Higher-Order Automated Theorem Proving

Michael Kohlhase

Computer Science
Carnegie Mellon University
Pittsburgh, Pa, USA

Fachbereich Informatik
Universität des Saarlandes
Saarbrücken, Germany

kohlhase@cs.cmu.edu

Unfinished Draft of June 30, 2004
Do not distribute!

Preface

In recent years, many theorem proving systems have been extended to higher-order logic (in particular based on variants of the simply typed lambda calculus). The two main reasons are that higher-order logic allows the quantification over functions, predicates and sets, and that the built-in notion of beta-eta reduction gives a convenient and transparent definition mechanism. The drawback of this approach is that the main computational device (higher-order unification, which is undecidable in general), becomes a major computational bottleneck.

In this book we survey the state of the art of higher-order automated theorem proving together a body of original research on the topic. We give a self-contained introduction to higher-order logic (chapters 1 and 2 introduce its syntax and semantics), introduce higher-order unification (chapter 3) and present the machine-oriented tableau and resolution calculi for higher-order logic (chapters 5 and 6). The main technical results will be the soundness and completeness theorems for these calculi. To obtain the completeness results (the calculi can prove every sentence that is valid in Henkin Semantics), we introduce a so-called model existence theorems in chapter 2.

Even with complete calculi, which make higher-order theorem proving semi-decidable, the problem is still intractable in practice. The main assumption of this book is that the introduction of additional knowledge into the theorem proving process and in particular into higher-order unification will make the search for higher-order unifiers (and thus for proofs in higher-order logic) more guided and ultimately yield more efficient higher-order theorem provers. The source of knowledge that we want to explore in this book is taxonomic information given in a concept hierarchy: We will introduce a sorted lambda-calculus (in chapter 7) where the universe is subdivided into various subsets (called sorts), and the signature is augmented by sort information. We introduce the technical device of term declarations, which give a very expressive mechanism for specifying (and using) sort information. After investigating the meta-theory of this logical system (sort preservation, subject reduction, ...) we will look at the sort computation and higher-order unification problems in this logic (chapter 8). The central result for this will be a structure theorem that links the structure of the formulae to the structure of their well-sortedness proofs.

We conclude the book by sketching how the methods developed in the first three parts can be extended for representing mathematical knowledge. We extend the sorted λ-calculus to a sorted higher-order logic (chapter 10), that can be seen as part of a hierarchy of interconnected representation languages for mathematics. Then propose a set of further language extensions that are aimed to come near the everyday mathematical language, the so-called mathematical vernacular (chapter 12).

Acknowledgments

I am indebted to my advisor Jörg Siekmann for introducing me to the field of artificial intelligence and giving me a feeling for the big picture of our field. I was very lucky that he focused my interest to the fascinating subject of higher-order deduction. His research group at the Universität des Saarlandes has provided the friendly, supportive, and extremely stimulating environment that is necessary for successful research.

During my visits to Carnegie Mellon University, Peter Andrews and Frank Pfenning have answered a great many questions, and have always had an open ear for my ideas; I would like to thank them for their patience and support.

The three main components of this thesis can be traced to the influence of these three people: Jörg Siekmann has introduced me to first-order resolution theorem proving and the dream of mechanizing mathematics. Peter Andrews had an important role in shaping my view of higher-order logic and deduction, while Frank Pfenning introduced me to type theory, and helped me to acquire most of the technical background that was necessary to complete the work reported in this book.

I would also like to explicitly thank my colleagues Franz Baader, Manfred Kerber, Daniel Nesmith, Christian Prehofer, Zhenyu Qian, Christoph Weidenbach, Gerald Klein, and Ortwin Scheja for valuable discussions and proofreading of earlier drafts of the thesis. Last but not least I want to thank Andrea Kohlhase, who has given me the moral support that is necessary to survive a substantial research project and who has taken up much of the burden of proofreading. More valuable even, she has always insisted that I try harder, when explanations were still unclear.

Finally, the "Studienstiftung des Deutschen Volkes" has had a twofold influence on the work reported in this thesis. At one of their summer academies in Campill, Italy, I had my very first positive experience with the field of artificial intelligence. This course, given by J"org Siekmann and Wolfgang Wahlster, later brought about my decision to change my research field from mathematics to deduction systems. Much later the Studienstiftung provided the research grant that made it possible to visit the Carnegie Mellon University for a semester without a lot of bureaucratic overhead.

The Deutsche Forschungsgemeinschaft (DFG) has financed much of the work reported first in my dissertation and then in this book in the collaborative research center (Sonderforschungsbereich) 378 "*Resource-Adaptive Cognitive Processes*" at Saarland University. Important parts of chapter 2 and sections 5.5 and 6.6 have been developed in collaboration with Christoph Benzmüller and was funded by the DFG in Project HOTEL. The final stages of preparing this book, were completed at Carnegie Mellon University during an extended research visit made possible by a DFG Heisenberg stipend.

Contents

arithmetic
calculus
Schickard
Pascal
Leibniz
Aristoteles
Leibniz
Boole
Frege
deduction system

Prologue

The field of mathematical logic has its roots in the effort of understanding the process of rational human reasoning. Since mathematical reasoning is human reasoning in its purest and most rigorous form, it is the most natural object for the investigation of this process.

If we look at the history of mathematics, we can observe a recurring pattern of change. Consider, for instance, the case of calculating with numbers, a task that has changed from a difficult job for highly paid specialists in Roman times to a task that is feasible for children in our century. What is the cause of this dramatic change? Of course the formalized reasoning procedures for arithmetic that we use nowadays. These so-called *calculi* consist of a set of rules that can be followed purely syntactically, but nevertheless manipulate arithmetic expressions in a correct and fruitful way. An essential prerequisite for syntactic manipulation is that the objects are given in a formal language suitable for the problem. For example, the introduction of the decimal system has been instrumental to the simplification of arithmetic mentioned above. When the arithmetical calculi were sufficiently well-understood and in principle a mechanical procedure, and when the art of clock-making was mature enough to design and build mechanical devices of an appropriate kind, the invention of calculating machines for arithmetic by Schickard (1623), Pascal (1642), and Leibniz (1671) was only a natural consequence.

Another important step for understanding the human reasoning process was the observation of Aristoteles, and other Greek philosophers, and later by Leibniz, Boole, and Frege, that mathematical methods (calculization) can be applied to the reasoning process itself. In particular, mathematical reasoning can be carried out by syntactically applying simple rules to formal expressions just like in the case of arithmetical calculi. This idea, to develop calculi for reasoning in analogy to those for arithmetic, has strongly influenced the development of formal languages, notions of semantics, and modern logical calculi. Just as the discovery of efficient calculi for arithmetic has led to the development of mechanical calculators, the discovery of logical calculi has led to the development of todays *deduction systems*, which are computer programs that perform reasoning tasks by operationalizing these calculi. The systems built so far can be roughly categorized into

unification
mating method
connection method
analytic tableaux
combinatorial explosion
AUTOMATH *Project*

the paradigms of interactive and fully automatic systems that aim at finding proofs for theorems with or without user interaction.

Automated theorem provers are usually based on refutation calculi that try to prove a theorem by deriving a contradiction from its negation and rely on unification [Rob65] as a central inference procedure. Unification algorithms (see [BS94] for a comprehensive survey) compute substitutions that equate given sets of terms. For refutation purposes only such substitutions (called most general) are needed from which all others can be recovered by instantiation. This property allows the refutation procedure to search for schematic proofs that represent all instances of the proof. Such refutation procedures are conceptually and computationally very simple, if the used sub-procedures, such as normal form reductions and unification, are decidable and of relatively low complexity. Therefore almost all automated theorem provers restrict the input language to some variant or subsystem of first-order logic, where unification is decidable and yields unique, most general unifiers.

The primary emphasis of research for fully automated systems lies in finding strong calculi and search strategies that restrict the search spaces associated with proof search. There are three classes of refutation calculi for automated theorem proving that turned out to be of primary importance, namely, resolution [Rob65, OS89], the mating/connection method [And81, Bib83], and analytic tableaux [Smu68, Fit90]. The resolution method has been further refined for a special treatment of the equality predicate in the Paramodulation [RW69], E-resolution [Mor69], and RUE-resolution [Dig79] calculi. Term Rewriting [KB70] systems, which have at first been developed for pure equational logic, have since been generalized to full first-order logic in the superposition calculi [BG98].

Automated theorem proving systems based on any of the calculi above have reached the power to solve non-trivial problems, but they are (like all search procedures) subject to the combinatorial explosion of the search spaces, and therefore have principal limits to the complexity of proofs that can be found. Thus in general they can approach the efficiency of mathematicians only in domains, where humans have very little or no intuition at all.

These limits have led part of the research community to investigate systems that rely on user interaction to find proofs. Since universal proof procedures do not play an important role in this paradigm, the community has developed highly expressive logical formalisms and calculi that are modeled after human reasoning in mathematical practice. This is essential for the concept of *interactive deduction systems*, because an expressive language allows for adequate formalizations of mathematical practice and also for short proofs that can be easily communicated. The AUTOMATH Project [dB80] has pioneered the area of mechanical proof checking by coding the total contents of a mathematical textbook [Lan30] in a formal language and proving all theorems in the accompanying proof system [Jut79].

Unfortunately, typical proofs in this and other systems are still so long and complex that almost all practical interactive systems provide a so-called "tactic mechanism" (giving rise to the name "tactic theorem proving"), which allows the user to write small specialized automatic reasoning procedures to relieve him of some of the routine work. Among the most influential tactic theorem provers are the Nuprl [CAB+86], the HOL [Gor85, GM93], the PVS [ORS92], and the KIV [HRS90, HRS91] systems, which were originally designed for program and hardware verification. Isabelle [PN90] and Elf [Pfe91] are examples for deduction systems that mechanize logical frameworks, i.e. systems where the logic language is powerful enough to allow the specification of object logics.

The main interest for higher-order automated theorem provers comes from the attempt to develop deduction systems that inherit the merits of both approaches to theorem proving, the interactive and the fully automatic one. If deduction systems are to be useful as assistants to mathematicians or in software verification, we will need the expressive formalisms (currently found in interactive systems) and the strong universal proof procedures (without the restriction to first-order logic), since otherwise formal deduction will be too tedious and expensive in practice.

Clearly the solution cannot only consist in the kind of automated theorem prover for expressive logics presented in this book, since such a system is probably even more subject to the combinatoric explosion found in first-order theorem provers. The answer will rather consist in a plethora of deductive systems (mathematical services) that are connected by the Internet, with a "mathematical software bus" to automated the communication and distribution of tasks (see [FHJ+99, FK99, AKR00, KF01] for ideas in this direction). With such a modular system design, it will be simpler to design semi-interactive deduction systems that have access to powerful automated theorem proving procedures (logic engines, such as resolution systems) to fill non-trivial gaps in the proofs the user develops interactively with the system (see [BCF+97] for an overview over the OMEGA system that attempts this). These logic engines must be automatic systems that can routinely solve non-trivial problems given in the expressive language of the interactive system calling them.

Part I

Higher-Order Logic

Higher-Order Logic[1]

EdNote(1)

[1] EdNote: say something here

Grundlagen-problem
Russell antinomy
Whitehead
Principia Mathematica
typed logic
higher-order logic

Introduction

A major driving force in the development of formal logics at the beginning of the twentieth century, was the attempt to apply formal (logical) methods mathematics itself aiming at a secure and rigorous axiomatic foundation (e.g. [Hil04, WR10]).

A solution of this *"Grundlagen" problem* requires a formal language to express all mathematical statements and a logical calculus that can formally derive all true statements. Hilbert's third ingredient of a solution: a requirement that this logical system should be strong enough to show its own consistency and completeness was later shown impossible by Kurt Gödel in [Göd31].

Even the endeavor of finding a logic that could express all of everyday mathematics turned out to be much more difficult than expected. For instance the well-known Russell-antinomies show that special precautions must be taken in order to prevent inconsistencies, even in the axiomatization of such basic objects as sets: In a naive formal system for set theory it is possible to write down a defining equation for the set $\mathcal{M}$ of all sets that do not contain themselves $(\forall Q.\mathcal{M}(Q) \equiv \neg Q(Q))$.This set is problematic, since it is not possible to decide whether $\mathcal{M}$ is a member of itself – a property that is deemed essential for a set.

Russell already suggested the use of typed logics [Rus08] as a possible remedy and used this approach together with Whitehead in the Principia Mathematica [WR10]. *Typed logics* classify objects by assigning a certain type to each object, and by restricting term formation to well-typed formulae. To be well-typed objects of functional type can only be applied to arguments whose type matches the type of their domain. The hierarchy of types naturally induces the notion of an order on types and objects.

In [Göd30] Gödel proved the completeness of a subsystem of the logic underlying Principia Mathematica (the so-called *first-order logic*), which allows quantification only over first-order variables (individuals). In contrast to this we will call logical systems that allow quantification over variables of arbitrary order *higher-order logics*. Unfortunately, first-order logic is so weak that neither Peano arithmetic nor, for instance, the theory of torsion groups[1] can be finitely axiomatized. Gödels later result [Göd31], stating that the full system of Principia Mathematica (and indeed any logical system that can formalize Peano arithmetic) is incomplete, has led to the dominance of first-order logic that we still experience today, even though only a fragment of mathematics can be adequately expressed. First-order logic is even more appealing, because the technical device of typing can be left implicit by distinguishing "terms" (denoting individuals) and "propositional formulae"

[1] Torsion groups are groups, where for any element a, there is a natural number n such that a^n is the neutral element.

Zermelo
Fraenkel
Gödel
comprehension axioms
axiom!comprehension

(denoting truth values).

Zermelo, Fraenkel, Gödel, and others used an encoding of mathematics into set theory [Zer08, Fra28, Neu28, Göd40, Ber41] that is itself axiomatizable in first-order logic to give an answer to the foundation problem for mathematics that is accepted by most mathematicians today. It is appealing, since it only uses first-order calculi and thus inherits all nice properties of first-order logic. Consequently, these ideas have been the basis for attempts to build automated theorem provers for mathematics [BLM+86, Qua92]. Note that these systems must be incomplete, as Peano arithmetic can be formalized in them via the encoding.

Mathematical vernacular usually uses a mixture of both: set theory and typed higher-order logics. Set theory provides a powerful tool for describing mathematical objects, and proofs are carried out in a logic implicitly typed by the choice of notation (n, m, k for natural numbers; f, g, h for functions …). Furthermore, explicit quantification over variables of higher type is widespread. Naturally, the opportunity to encode all of this into set theory is almost never used, since the encodings of the objects of interest become much too large and quite unwieldy. On the contrary, deduction is in general carried out in the respective technical language, that has been established for the particular mathematical field in question. Thus, if we view axiomatic set theory only as an answer to the foundation problem of mathematics, where the aim consists in demonstrating that the whole body of mathematics can be encoded into a consistent logical system, the problem can be viewed as solved and the technical inconvenience that, for instance, basic mathematical objects like functions have to be encoded as right total, left unique relations are irrelevant. This need for encoding into axiomatic set theory is clearly an obstacle for giving an adequate account of informal mathematical practice. The author believes that the choice of a typed higher-order logic is much more natural for the use in deduction systems, since conserves the structure in the objects, and thus allows to guide the search for proofs. After we have introduced higher-order logic and deduction, we will go another step in this direction and introduce higher-order logics, which have even more infrastructure for mathematical structure. In chapter 10, we will even set both axiomatic set theory and typed higher-order logics into relation and show that the two views do not exclude each other.

Higher-Order Logic and Simply Typed λ-Calculus

The expressiveness of the logical systems discussed so far (higher-order logics as well as axiomatic set theories) is guaranteed by the so-called *comprehension axioms*, which postulate the existence of all functions that can be expressed by well-formed formulae parameterized by free variables. Comprehension for functions (and sets) is generally taken for granted in mathe-

Cantor
Church
simple type theory
combinatory logic
Schönfinkel
Curry

matics ("everything that can be written down exists"), and implicitly used in function definitions like $f(x) = \sin^2(x)/2\pi$. Note that naive use of the comprehension principle for sets gave rise to the structurally similar $\mathcal{M}(Q) \equiv \neg Q(Q)$. The two approaches to the foundations of mathematics mainly differ in the restriction of the dangerous comprehension principle at work here. Type theoretic approaches restrict "what can be written down", whereas set-theoretic approaches replace a general comprehension principle by a set of weaker comprehension axioms for sets (the so-called ZFC axioms).

In the path we will pursue in this book, we use types and can therefore allow a general comprehension axiom schema of the following form.

$$\text{(Comp)} \qquad \exists F. \forall X_1 \ldots X_n. F X_1, \ldots X_n = \mathbf{A}$$

Unfortunately, this axiom schema stands for an infinite set of axioms makes a direct use of higher-order logic for the mechanization of mathematics impossible, because an automated theorem prover would have to be interactively supplied with the subset of comprehension axioms relevant for the problem at hand. Thus significant guidance would be left to the user of a deduction system, since the choice of the appropriate comprehension axiom (for instance, postulating the existence of a diagonal sequence in the proof of Cantor's theorem) is often a key idea to the proof.

This was one of the reasons for Alonzo Church to reformulate higher-order predicate logics to the (logically equivalent) system of *simple type theory*[2]. In this logical system the comprehension axioms are cast in an equality theory, which can even be directed to a confluent terminating reduction system, and is therefore decidable. We can understand this reformulation by the following argument. Take an instance of the original comprehension axiom $\exists F. \forall X. FX = \mathbf{A}$, where $\mathbf{A}$ is an arbitrary formula, and give the function F, which is guaranteed by this axiom, the name $\lambda X. \mathbf{A}$, then we are left with the assertion that $\forall X. (\lambda X. \mathbf{A})X = \mathbf{A}$. This can be instantiated by an arbitrary formula $\mathbf{B}$ to $(\lambda X. \mathbf{A})\mathbf{B} = [\mathbf{B}/X]\mathbf{A}$ which is just the definition of β-equality. Thus Church's simply typed λ-calculus is a very elegant and intuitive formulation of higher-order logic, since the syntax and semantics of typed predicate logics can be reobtained by simple definitions. To do this, we specialize the set of base types in $\Lambda^{\rightarrow}$ by restricting them to a type ι for individuals and a type o for truth values. To complete the logical system $\mathcal{HOL}$, we add compulsory logical constants $\neg$ of type $o \rightarrow o$ and $\wedge$ of type $o \rightarrow o \rightarrow o$ for the connectives, and, finally, the quantification constants Π^α of types $(\alpha \rightarrow o) \rightarrow o$ for quantification.

For further reading, we also want to mention an alternative, equivalent formulation of higher-order logic. *Combinatory logic* was developed by Schönfinkel [Sch24], and then thoroughly investigated by Curry and

[2] In fact, Church's type system is a simplification of Russell's system of ramified types [Rus08].

λ-calculus type theory

Feys [CF58] (for a modern treatment see [HS86]). In combinatory logic the role of λ-abstraction and β-conversion is taken up by the combinators K, S, I, and the axioms of weak combinatory reduction: $I\mathbf{A} \to \mathbf{A}$, $K\mathbf{A}\mathbf{B} \to \mathbf{A}$ and $S\mathbf{A}\mathbf{B}\mathbf{C} \to (\mathbf{A}\mathbf{C})(\mathbf{B}\mathbf{C})$. The simply typed version of this system of higher-order logic is equivalent to the simply typed λ-calculus, since each combinatory logic formula can directly be translated into a λ-calculus formula and vice versa. Moreover, the equality theories are coextensive modulo this translation. Combinatory logic has one great technical advantage: there is no need for λ-abstractions, and consequently, bound variables are not a problem. But it has the practical disadvantage that formulae are much more difficult to read for humans. While there have been attempts to mechanize mathematics on the basis of combinatory logics [Rob69b, Joh91, DJ92, Dou93, Joh93, Koh93], this is not the subject of this book, although it may become important in the future.

Independently from its logical origins the simply typed λ-calculus has become one of the most important tools of computer science for describing functions, programming languages, and more generally computability. This has spawned the development of a rich zoo of specialized type systems for various λ-calculi (see for instance [Tho91a]). Although the name "type theory" has originally been used by Church for his logical system (i.e. the simply typed λ-calculus augmented by logical constants and axioms), it has become customary to refer with this name to λ-calculi with powerful type systems but without logical constants or axioms. We will stick to this usage and call our λ-calculi higher-order logics whenever logical constants and axioms are present.

Semantics

In classical first-order predicate logic, it is rather simple to assess the deductive power of a calculus: first-order logic has a well-established and intuitive set-theoretic semantics, relative to which, completeness can easily be verified using for instance the abstract consistency method (see for instance the introductory textbooks [And86, Fit90]). This well-understood meta-theory has supported the development of calculi adapted to special applications — such as automated theorem proving (see for instance [BS98] for an overview).

In higher-order logics, the situation is rather different: the intuitive set-theoretic standard semantics cannot give a sensible notion of completeness, since it does not admit complete calculi [Göd31]. However, there is a more general notion of semantics (the so-called Henkin-models [Hen50]), that allows complete calculi and therefore sets the standard for deductive power of calculi.

Peter Andrews' "Unifying Principle for Type Theory" [And71] provides a method of higher-order abstract consistency that has become the standard tool for completeness proofs in higher-order logic, even though it can only be used to show completeness relative to a certain Hilbert style calculus $\mathfrak{T}$. A calculus $\mathcal{C}$ is called complete relative to a calculus $\mathcal{C}'$, iff $\mathcal{C}$ proves all theorems of $\mathcal{C}'$. Since $\mathfrak{T}$ is not complete with respect to Henkin models, the notion of completeness that can be established by this method is a strictly weaker notion than Henkin completeness.

As a consequence, the calculi developed for higher-order automated theorem proving [Hue73a, And71, Mil83, Koh95] and the corresponding theorem proving systems such as Tps [ABI+96a], or earlier versions of the authors' Leo[3] are not or cannot be proven complete with respect to Henkin models. Moreover, they are not even sound with respect to $\mathfrak{T}$, since all of them utilize η-conversion, which is not a theorem of $\mathfrak{T}$. In other words, their deductive power lies somewhere between $\mathfrak{T}$ and Henkin models.

In this situation, the aim of this chapter is to provide a semantical metatheory that will support the development of higher-order calculi for automated theorem proving like the corresponding methodology has in first-order logic. To reach this goal, we establish

- classes of models that adequately characterize the deductive power of existing theorem-proving calculi (making them sound and complete), and
- a standard methodology of abstract consistency methods (by providing the necessary model existence theorems, which extend Andrews' Unifying Principle), so that the completeness analysis for higher-order calculi will become a simple exercise like in first-order logic.

We investigate three notions of a model theory. The standard semantics is the intuitive semantics for mathematics, but it has the disadvantage that it does not admit complete calculi, for principal reasons [Göd31], as we have pointed out above. The Henkin semantics is somewhat less intuitive, but admits complete calculi [Hen50], and indeed we present a generalization of Henkin's original calculus from [Hen50] and prove its completeness with respect to this class of models. Unfortunately, current refutation calculi for automated theorem proving have problems with completeness for the Henkin semantics, since they fail to prove what we have called the axiom of truth values (cf. 2.3.6). Therefore we develop an even weaker semantics — that of Frege structures – which makes these refutation calculi complete. This notion of completeness is equivalent to the notion of relative completeness found in the literature [And71, Mil83, Pfe87].

Furthermore, we prove model existence theorems for Henkin models and Frege Structures. These theorems state that sets that have the property of being consistent in some abstract way (Γ_Σ-consistency) have a model. The

[3]Based on the results of section 6.6 the resolution calculus underlying newer versions of Leo [Ben97, BK98] can be proven complete for Henkin models based, see [BK97b].

proof of the model existence theorem for Frege structures is based on techniques from [And71], but we strengthen the notion of abstract consistency class by an additional saturatedness condition. This makes the valuations in the proof total, which in turn simplifies the proof and the semantical objects used in it. Furthermore, it allows to construct a Σ-term structure for any Γ_Σ-consistent set of sentences, which yields the model-theoretic result. Without saturatedness Andrews can only obtain partial valuations, and from this can only conclude $\mathfrak{T}$-consistency of Γ_Σ-consistent sets, where $\mathfrak{T}$ is a special Hilbert-Style calculus for higher-order logic. This result is a direct corollary of our's, since $\Sigma\mathfrak{T}$ is sound with respect to Frege structures. The model existence theorem for Henkin models needs further methods for the manipulation of equality and its connection to propositional equivalence in order to handle the axiom of truth values. These allow us to construct a Σ-congruence on the Σ-term structure that collapses the set $\mathcal{D}_o$ to the set $\{\mathtt{T}, \mathtt{F}\}$ of truth values, and thus constructs a Henkin model for any Γ_Σ-consistent set of sentences, provided that Γ_Σ is a saturated, extensional abstract consistency class.

Since an abstract consistency class Γ_Σ can be expressed in purely syntactic terms our model existence theorems can be used in a completeness proof for a refutation calculus $\mathcal{C}$ by showing that $\mathcal{C}$-consistency is an abstract consistency property and thus that $\mathcal{C}$-consistent sets of formulae are satisfiable. The contrapositive of this (unsatisfiable sets of formulae are $\mathcal{C}$-refutable) is just the assertion of the completeness for $\mathcal{C}$. We use this argument to give simple and elegant completeness proofs for the Hilbert-style calculi from [Hen50, And71]. These then result in compactness theorems for $\mathcal{HOL}$ with respect to Frege structures and general Σ-models.

$\mathcal{BT}$, $\mathcal{T}$ sets of base types and types
type
base type
type!base
functional type
type!functional

Chapter 1

Simply Typed λ-Calculus

In this section we review Church's simply typed λ-calculus $\Lambda^{\to}$. We will use it as an algebraic foundation for higher-order logic. We discuss the algebraic structure of well-formed formulae, and we give notions of algebraic semantics that are independent of the logical view of higher-order logic. For this we introduce the framework of Σ-structures, which provides a more algebraically flavored setting than, for example, the one given in [And86] or [HS86]. In particular, the notions of Σ-homomorphism and Σ-congruence will be useful later on.

We will introduce all concepts of $\Lambda^{\to}$ in the more general setting of partial functions, as they can be handled with little overhead, and we can also use them as a basis for the sorted λ-calculus $\Sigma\Lambda^{\to}$ and the sorted higher-order logic $\Sigma\mathcal{HOL}$. In $\Sigma\Lambda^{\to}$ and $\Sigma\mathcal{HOL}$ the type system is refined by a sort system, in which the domains of functions coincide with their domain sorts (which are subsets of the types), thus functions are total on their domains, but partial on the types.

1.1 Syntax of $\Lambda^{\to}$

Definition 1.1.1 (Types)
Let $\mathcal{BT}$ be a set of symbols, then the set $\mathcal{T}$ of **type**s is inductively defined to be the set $\mathcal{BT}$ together with all expressions $\alpha \to \beta$, where α and β are types. The functional type $\alpha \to \beta$ denotes the type of functions with domain α and codomain β. The types in $\mathcal{BT} \subset \mathcal{T}$ are called **base type**s, types of the form $\alpha \to \beta$ are called **functional type**s.

Example 1.1.2 If we have base types ι and o (for individuals and truth values; see section 1.2.1), then $\iota \to \iota$ is the type of functions from individuals, and $\iota \to o$ of unary predicates (functions from individuals to truth values).

Types for binary (and n-ary) functions can also be expressed using nested function types by a technique called “currying” (cf. 1.2.13): a binary func-

functional
$\mathbf{ln}(\alpha)$, $\mathbf{ord}(\alpha)$*length, order of* α
length
type!length
target type
type!target
order
type!order
$\mathbf{ord}(\alpha)$*order of a type*
variables
$\mathcal{V}, \mathcal{V}_\alpha$*set of variables (of type* α*)*
constant
Σ, Σ_α*set of constants (of type* α*)*
variable
$\mathit{wff}(\Sigma)$, $\mathit{wff}_\alpha(\Sigma)$*set of well-formed formulae*
well-formed
formula!well-formed

tion, say $+$ is given the type $\iota \to (\iota \to \iota)$, i.e. the type of a unary function that maps an argument (say the number 1) to a unary function (in this case the function that increments its argument by 1).

Note that while $\alpha \to (\beta \to \gamma)$ basically denotes a binary function, $(\alpha \to \beta) \to \gamma$ is the type of a *functional*, e.g. the differentiation operator, that takes a function (of type $\alpha \to \beta$) as an argument.

Notation 1.1.3 For the following we fix a set $\mathcal{BT}$ of base types and a set $\mathcal{T}$ of types induced by $\mathcal{BT}$. As syntactic variables for types we use lower case Greek letters. We use the convention of association to the right for omitting parentheses in types, thus $\alpha \to \beta \to \gamma$ is an abbreviation for $(\alpha \to (\beta \to \gamma))$. This way the type $\gamma := \beta_1 \to \ldots \to \beta_n \to \alpha$ denotes the type of n-ary functions, that take n arguments of the types $\beta_1, \ldots, \beta_n$ and have values of type α. To conserve even more space we use a kind of vector notation and abbreviate γ by $\overline{\beta_n} \to \alpha$.

Let

Definition 1.1.4 (Length and Order)
We define the **length** of a type α by setting $\mathbf{ln}(\alpha) := 0$, iff $\alpha \in \mathcal{BT}$ and $\mathbf{ln}(\alpha \to \beta) := 1 + \mathbf{ln}(\beta)$. Thus the length intuitively is the number of top level arrows $\to$ in a type. In other words a type α is functional, iff its length is positive. If $\tau(=)\overline{\beta^n} \to \gamma$, where $\gamma \in \mathcal{BT}$, then we call γ the **target type** of α.

We define the **order** $\mathbf{ord}(\alpha)$ of a type α as 0 for $\alpha \in \mathcal{BT}$ and as $1 + \max\{\mathbf{ord}(\alpha_i) \mid \alpha = \overline{\alpha} \to \beta, \beta \in \mathcal{BT}\}$ else. Thus the order of a type is intuitively the depth of nesting of $\to$ in a type.

Example 1.1.5 The length of $\iota \to \iota \to \iota$ is 2 and the order is 1, while $\mathbf{ln}((\iota \to \iota) \to \iota) = 1$ and $\mathbf{ord}((\iota \to \iota) \to \iota) = 2$.

For defining well-formed formulae we need a collection $\mathcal{V}$ of **variables** and a collection Σ of **constant**s. Since variables are much more volatile syntactic objects, which are frequently instantiated and renamed, we need an infinite supply of variables of any type. So we fix a countably infinite set $\mathcal{V}_\alpha$ of **variable**s of type α for every type $\alpha \in \mathcal{T}$.

Definition 1.1.6 (Well-Formed Formulae)
For each $\alpha \in \mathcal{T}$ we define the set $\mathit{wff}_\alpha(\Sigma)$ of **well-formed** formulae of type α inductively by

1. $\Sigma_\alpha \cup \mathcal{V}_\alpha \subseteq \mathit{wff}_\alpha(\Sigma)$
2. If $\mathbf{A} \in \mathit{wff}_{\beta\to\alpha}(\Sigma)$ and $\mathbf{B} \in \mathit{wff}_\beta(\Sigma)$, then $\mathbf{AB} \in \mathit{wff}_\alpha(\Sigma)$.
3. If $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma)$, then $(\lambda X_\beta.\mathbf{A}) \in \mathit{wff}_{\beta\to\alpha}(\Sigma)$.

We call formulae of the form **AB applications**, and formulae of the form $(\lambda X_\alpha\mathbf{.}\mathbf{A})$ **λ-abstractions**.

application
abstraction
$\mathbf{Free}(\mathbf{A})$*set of free variables of* $\mathbf{A}$
$\mathit{wff}(\Sigma, \Xi)$*set of well-formed formulae with free variables in* Ξ
$\mathbf{Bound}(\mathbf{A})$*set of bound variables of* $\mathbf{A}$
$\mathit{cwff}(\Sigma), \mathit{cwff}_\alpha(\Sigma)$*set of closed well-formed formulae*
bound
free

Remark 1.1.7 (Functions)
As the type $\alpha \to \beta$ already suggests, λ-abstractions can be viewed as representations of functions. $\lambda X_\beta\mathbf{.}\mathbf{A}_\alpha$ stands for the function that takes an argument $\mathbf{B}_\beta$ and gives the value of $\mathbf{A}$, where the "formal parameter" X_α has been bound to $\mathbf{B}_\beta$ (this intuition will be made formal by the theory of $\alpha\beta\eta$-equality in Definition 1.1.15).

In contrast to the classical mathematical notation e.g. $f(x) = 2x^2 + 3cx$, the λ-notation makes the intended arguments of the function very explicit: in our example, the variable x is an argument to f, but depending on the situation f often also also be viewed as a function in the parameter c. The λ-notation makes the intention which of the symbols the function depends upon explicit. In the latter situation, we would represent f as $\lambda c, x\mathbf{.}2x^2 + 3cx$

The λ-notation is especially convenient and useful for functionals (higher-order functions) such as the differentiation and integration operators, where $\frac{\partial}{\partial x} f$ becomes $\partial(\lambda x\mathbf{.}2x^2 + 3cx)$ and $\int_a^b \mathbf{A}dx$ is $\int_a^b \lambda X\mathbf{.}\mathbf{A}$ with functionals ∂ and $\int$ of types $(\iota \to \iota) \to \iota \to \iota$ and $\iota \to \iota \to (\iota \to \iota) \to \iota$.

Example 1.1.8 (Defining Concepts)
λ-terms can be used to define new concepts from known ones. Say that we have a signature that contains a constant $\times$ for multiplication, then we can use the λ-term $(\lambda X_\iota\mathbf{.} \times XX)$ for the square function.

Notation 1.1.9 We denote the constants by lower case letters and the variables by upper case letters and use bold upper case letters $\mathbf{A}_\alpha$, $\mathbf{B}_{\alpha\to\beta}$, $\mathbf{C}_\gamma \ldots$ as syntactical variables for well-formed formulae. The type of an object is denoted as a subscript, if it is not irrelevant or clear from the context.

In order to make the notation of well-formed formulae more legible, we use the convention that the group brackets (and) associate to the left and that the square dot **.** denotes a left bracket, whose mate is as far right as consistent with the brackets already present. Additionally, we combine successive λ-abstractions, so that the well-formed formula $(\lambda X^1\mathbf{.}\lambda X^2\mathbf{.} \ldots \lambda X^n\mathbf{.}\mathbf{A}\mathbf{E}^1 \ldots \mathbf{E}^m)$, which stands for $(\lambda X^1\mathbf{.}\lambda X^2\mathbf{.} \ldots (\lambda X^n\mathbf{.}(\cdots(\mathbf{A}\mathbf{E}^1)\mathbf{E}^2) \ldots \mathbf{E}^m)$, becomes $\lambda X^1 \ldots X^n\mathbf{.}\mathbf{A}\mathbf{E}^1 \ldots \mathbf{E}^m$, and in addition, we shorten the expression to $\lambda\overline{X^n}\mathbf{.}\mathbf{A}\overline{\mathbf{E}^m}$ by a kind of vector notation.

To avoid confusion with equality in the logic we denote the meta-logical relation of syntactic equality of well-formed formulae by $\equiv$.

Definition 1.1.10 Let $\mathbf{A}$ be a well-formed formula, then a variable X is called **bound** (**free**) in $\mathbf{A}$, iff it is (not) in a well-formed part of the form $(\lambda X\mathbf{.}\mathbf{B})$ in $\mathbf{A}$. The respective sets of variables are denoted by $\mathbf{Free}(\mathbf{A})$ and

closed
supp(σ)*support of the substitution* σ
SUB(Σ)*set of substitutions*
Intro(σ)*set of variables introduced by* σ
substitution
support
introduced by σ
variablesset of!introduced by σ
renaming substitution
substitution!renaming
idempotent
variable capture
variable capture
capture!variable

Bound(**A**). A well-formed formula is called **closed**, if it does not contain free variables. We denote the set of closed well-formed formulae of type α by $cwff_\alpha(\Sigma)$.

With the definition of free variables we can define sets of well-formed formulae that have restricted sets of free variables: let $\mathcal{W} \subset \mathcal{V}$ be a typed collection of variables, then we denote the set of well-formed formulae with free variables in $\mathcal{W}$ by

$$wff_\alpha(\Sigma; \mathcal{W}) := \{\mathbf{A} \in wff_\alpha(\Sigma) \mid \mathbf{Free}(\mathbf{A}) \subseteq \mathcal{W}\}$$

With this definition $cwff(\Sigma) = wff(\Sigma; \emptyset)$.

Since any formula **A** can only have finitely many variables there is always a finite set $\mathcal{W}$ of variables such that $\mathbf{A} \in wff_\alpha(\Sigma; \mathcal{W})$.

Definition 1.1.11 (Substitution)
We call an assignment σ into $wff(\Sigma)$ a **substitution**, iff its **support** $\mathbf{supp}(\sigma) := \{X \in \mathcal{V} \mid \sigma(X) \not\equiv X\}$ is finite. We write a substitution σ with $\mathbf{supp}(\sigma) = \{X^1, \ldots, X^n\}$ and $\sigma(X^i) = \mathbf{A}^i$ as $[\mathbf{A}^1/X^1], \ldots, [\mathbf{A}^n/X^n]$ or short $\overline{[\mathbf{A}^n/X^n]}$. If $\sigma = [\mathbf{A}/X]$, then we often write $\sigma(\mathbf{B})$ as $[\mathbf{A}/X]\mathbf{B}$. The set $\mathbf{Intro}(\sigma) := \bigcup_{i \leq n} \mathbf{Free}(\mathbf{A}^i)$ is called the set of variables **introduced by** σ. The set of substitutions is denoted by $\mathbf{SUB}(\Sigma)$. An injective substitution $\sigma := \overline{[\mathbf{A}^n/X^n]}$ is called **renaming substitution**, if the $\mathbf{A}^i$ are all variables.

A substitution σ is called **idempotent**, iff $\sigma \circ \sigma = \sigma$. Note that the condition $\mathbf{Intro}(\sigma) \cap \mathbf{supp}(\sigma) = \emptyset$ is a sufficient condition for σ to be idempotent [Sny91].

A substitution σ can always be extended to a mapping $\widehat{\sigma}$ on all formulae by requiring

1. $\widehat{\sigma} = \sigma$ on $\mathcal{V}$
2. $\widehat{\sigma}(c) = c$ for all constants $c \in \Sigma$
3. $\widehat{\sigma}(\mathbf{AB}) := (\widehat{\sigma}(\mathbf{A})\widehat{\sigma}(\mathbf{B}))$
4. $\widehat{\sigma}(\lambda X.\mathbf{A}) := (\lambda X.\widehat{\sigma}_{-X}(\mathbf{A}))$

It depends on the context, whether it is more convenient to view substitutions as functions with finite support or as Σ-homomorphisms, hence we take the liberty to switch our point of view whenever convenient.

Remark 1.1.12 (Variable Capture)
It is easy to see, that if there is a any well-formed part $(\lambda Y.\mathbf{C})$ of **B**, in which a variable X is free, $Y \in \mathbf{Free}(\mathbf{A})$, and $\mathbf{B}'$ is obtained from **B** by replacing all free occurrences of X with **A**, then $\mathbf{B}'$ has bound occurrences of variables Y that were free occurrences in **A**. We call this situation **variable capture** and need to avoid it for semantical correctness of instantiation. For

instance let $\mathbf{B} = (\lambda Y.X)$ and $\mathbf{A} = Y$, then $\mathbf{B}' = [\mathbf{A}/X]\mathbf{B} = (\lambda Y.Y)$. The problem is that Whereas $\mathbf{B}$ denotes a constant function, $\mathbf{B}'$ is the identity function, which is (in general) not constant.

$\mathbf{A}{=}_\alpha\mathbf{B}\mathbf{A}$ and $\mathbf{B}$ are alphabetical variants
alphabetical variant
α-equal
α-conversion
α-equality
$\rightarrow_\beta\beta$-reduction
$\rightarrow_\eta\eta$-reduction
$=_\beta, =_\eta, =_{\beta\eta}\beta\eta$-equality relations
one-step** $\beta\eta$**-reduction
redex
closure!transitive reflexive
relation!multi-step
relation!reduction
reduction relation

Definition 1.1.13 (Alphabetical Variant)
If a well-formed formula $\mathbf{B}$ is obtained from a well-formed formula $\mathbf{A}$ by replacing a subformula $(\lambda X_\alpha.\mathbf{C})$ of $\mathbf{A}$ such that $Y_\alpha \notin \mathbf{Free}(\mathbf{C})$ with $(\lambda Y_\alpha.[Y_\alpha/X_\alpha]\mathbf{C})$, then $\mathbf{B}$ is called an **alphabetical variant** of $\mathbf{A}$. We will also say that such formulae **α-equal** ($\mathbf{A}{=}_\alpha\mathbf{B}$) and call the induced equality relation on well-formed formulae **α-conversion** or **α-equality**.

General Assumption 1.1.14 In the following we will consider alphabetical variants as identical, and use the variable names in formulae just as a representational device, taking the liberty to rename bound variables wherever necessary. In particular, we can avoid variable capture, by renaming the bound variables in the argument of σ by α-conversion, so that the sets of bound and free variables are disjoint. Another, more algebraic way of avoiding variable capture is to assume α-equality to be built into the system and regard well-formed formulae as syntactically equal, iff they are alphabetical variants (α-equal). Formally we would replace the typed collection *wff*(Σ) by its quotient modulo α-conversion. We could also have used de Bruijn's indices [dB72], as a concrete implementation of this approach at the syntax level.

We will now introduce the notions of $\beta\eta$-conversion and reduction, which are closely related to each other. These equivalence relations together with α-conversion are are the equality theory built into the λ-calculus. Conceptually, they establish λ-abstractions as representations of functions, by giving interpretations to function application and function equality.

Definition 1.1.15 ($\beta\eta$-Reduction)
Let $\lambda \in \{\beta, \beta\eta, \eta\}$. We say that a well-formed formula $\mathbf{B}$ is obtained from a well-formed formula $\mathbf{A}$ by a **one-step $\beta\eta$-reduction** ($\mathbf{A} \rightarrow_{\beta\eta} \mathbf{B}$), if it is obtained by applying one of the following rules to a well-formed part (which we call a **λ-redex**) of $\mathbf{A}$.

β-Reduction $(\lambda X.\mathbf{C})\mathbf{D} \rightarrow_\beta [\mathbf{D}/X]\mathbf{C}$.

η-Reduction If X is not free in $\mathbf{C}$, then $(\lambda X.\mathbf{C}X) \rightarrow_\eta \mathbf{C}$.

As usual we denote the transitive, reflexive closure of a reduction relation $\rightarrow$ (i.e. the smallest transitive relation that contains it) with $\rightarrow^*$. Thus $\mathbf{A} \rightarrow^* \mathbf{B}$, iff there is a sequence of one-step reductions

$$\mathbf{A} \equiv \mathbf{A}^1 \rightarrow \cdots \rightarrow \mathbf{A}^n \equiv \mathbf{B}$$

equality!
normal form
higher-order logic
logic!higher-order
o, ι *types of truth values and individuals*
individual
truth value
$q^\alpha, \neg, \wedge, \Pi^\alpha$ *logical constants*
$\Sigma^{\mathcal{HOL}}$, *sig:empty′ signatures with logical constants*
logical constant
constant!logical
connective
quantifier
equality constant
constant!equality
parameter
proposition
sentence
$\vee, \Rightarrow, \Leftarrow, \Leftrightarrow, \forall, \exists$ *derived logical constants and quantifiers*
$\mathbf{A} =^\alpha \mathbf{B}, \mathbf{A} = \mathbf{B}$ *abbreviation for* $q^\alpha\mathbf{AB}$
$\forall X_\alpha.\mathbf{A}$ *abbreviation for* $\Pi^\alpha(\lambda X_\alpha.\mathbf{A})$

We will call the symmetric closures $=_\beta, =_\eta$, and $=_{\beta\eta}$ of these reduction relations the $\beta\eta$-**equality** relations. A formula that does not contain a $\beta\eta$-redex, and thus cannot be reduced by $\beta\eta$-reduction, is called a $\beta\eta$-**normal form**.

1.2 Applications of the Simply Typed λ-Calculus

In this section, we will fortify our intuition by using the simply typed λ-calculus as a basis for defining various logical applications. This will allow us not only to discuss various logical issues of higher order logic in an informal way before we embark into a more formal and semantics-oriented investigation in Chapter 2, but also to see the power of the simply typed λ-calculus.

1.2.1 λ-Calculus and (Higher-Order) Logics

We will now define a logical system $\mathcal{HOL}$ for **higher-order logic** by instantiating the set of types and constants suitably., which will be the basis for our automated theorem proving calculi in part **??**.

The intuition behind this is that if we assume that the set $\mathcal{BT}$ of base types is $\{o, \iota\}$ where the base type ι stands for the set of **individual**s and the type o for the **truth value**s, then we can distinguish classical syntactic classes of logical formulae like propositions, functions, predicates by the type of the objects alone. Moreover, since we allow objects of arbitrary types in $\Lambda^{\rightarrow}$, we can simply define the logical connectives and quantifiers needed for the inductive definitions of the logical language as constants.

Definition 1.2.1 (Logical Constants)
Let

$$\begin{array}{rcl}\Sigma^{\mathcal{HOL}} & := & \{[\neg\colon o \rightarrow o], [\wedge\colon o \rightarrow o \rightarrow o], [\Pi^\alpha\colon (\alpha \rightarrow o) \rightarrow o] \mid \alpha \in \mathcal{T}\} \\ \Sigma^{=} & := & \{[q^\alpha\colon \alpha \rightarrow \alpha \rightarrow o] \mid \alpha \in \mathcal{T}\}\end{array}$$

We call the constants $q^\alpha, \neg, \wedge, \Pi^\alpha$ **logical constant**s, since they will have a fixed interpretation in the respective semantics. More specifically $\neg, \wedge$ are called **connective**s and Π^α a **quantifier** in order to distinguish them from the **equality constant** q^α. Non-logical constants are called **parameter**s, since the choice of parameters determines the particular formulation of the logical system $\mathcal{HOL}$.

A well-formed formula of type o is called a **proposition**, and a closed proposition a **sentence**.

Notation 1.2.2 Since the constants q^α are intended to denote the equality relation, we use $\mathbf{A} =^\alpha \mathbf{B}$ or even $\mathbf{A} = \mathbf{B}$ as an abbreviation for $(q^\alpha\mathbf{AB})$.

Furthermore, we can obtain the connectives $\vee, \Rightarrow, \Leftrightarrow$ from the connectives defined so far, for instance, we take $\mathbf{A} \Rightarrow \mathbf{B}$ as an abbreviation for $\neg(\mathbf{A} \wedge \neg\mathbf{B})$. Finally, we use the infix notation for connectives, and write $\forall X_\alpha . \mathbf{A}$ as an abbreviation for $\Pi^\alpha(\lambda X_\alpha . \mathbf{A})$.

higher-order abstract syntax
syntax!higher-order abstract
$\amalg^\alpha$ *logical constant for existential quantification*
$\exists X_\alpha . \mathbf{A}$ *abbreviation for* $\amalg^\alpha(\lambda X_\alpha . \mathbf{A})$
order
$\mathit{wff}^n_\alpha(\Sigma) n^{\mathrm{th}}$ *order formulae in* $\mathit{wff}_\alpha(\Sigma)$
term
first-order term

Remark 1.2.3 (Higher-Order Abstract Syntax)
The intuition for the last notation is that in a logical language like $\Lambda^{\rightarrow}$ that provides one primitive binding and substitution mechanisms, it is possible to define other binding mechanisms (like quantifiers) in terms of it. This idea has been very influential under the name of **higher-order abstract syntax** and is crucial to the field of (meta)-logical frameworks, where this effect is used to define logics and reason about them in specialized λ-calculi (see e.g. [HP91] for an introduction).

In our case, we can think of Π^α as a predicate on predicates (sets) that that is true on all predicates that are universally true (see 2.1.1).

Similarly, we can define an existential predicate $\amalg^\alpha$ to be an abbreviation for $\lambda P_{\alpha\to o} . \neg\Pi^\alpha(\lambda Z_\alpha . \neg PZ)$. Note that $\amalg^\alpha \lambda X_\alpha . \mathbf{A} = \neg\Pi^\alpha(\lambda X_\alpha . \neg\mathbf{A}) = \neg\forall X_\alpha . \neg\mathbf{A}$, so we write $\exists X . \mathbf{A}$ for $\amalg^\alpha \lambda X_\alpha . \mathbf{A}$.

Before we go on with the exploration of higher-order logic as a formal system derived from the simply typed λ-calculus, let us make our ideas more precise about the order of formulae and of logical systems in general.

We will use of the order of types to distinguish subclasses of $\mathit{wff}(\Sigma)$. Generally one speaks of a system of second-order logic, if we allow quantification over functions and predicates. Note that functions and predicate variables are already present in first-order logic, one is only not allowed to quantify over them. So we speak of a system of third-order logic, iff there are constants for functionals (functions that take first-order functions as arguments) present and of fourth-order logic, when they can be quantified. Generally, we have the following ordering schema.

Definition 1.2.4 (n^{th} Order formulae)
A formula $\mathbf{A} \in \mathit{wff}(\Sigma)$ is called of **order**

$n = 2k$ iff the maximal order of types of symbols occurring in $\mathbf{A}$ is k,

$n = 2k - 1$ iff the maximal order of types of constants occurring in $\mathbf{A}$ is k, and that of variables is $k - 1$.

We denote the subsets of n^{th} order formulae in $\mathit{wff}_\alpha(\Sigma)$ by $\mathit{wff}^n_\alpha(\Sigma)$..

Generally, we speak of a system of higher-order (or ω-order) logic, iff quantification over variables of arbitrary order is allowed.

Remark 1.2.5 (First-Order Terms)
In classical accounts of logic, the set of first-order terms is built up from individual and function constants (of arity n) and individual variables. There

Leibniz formulation of equality
Leibniz formula
Leibniz
define
Meta-equality
object
semantical domain
constant symbol
Leibniz equality

a signature consists of set of $\mathcal{F}^n$ of n-ary functions (individual constants are treated as 0-ary functions).

Now let $\Sigma_{\alpha^n} := \mathcal{F}^n$ where $\alpha^n := \iota \to \iota \to \ldots \to \iota$, with $n+1$ occurrences of a type ι for individuals. Then function constants have types of order 1, while variables have order 0, and thus with the definition above, the set of first-order terms is isomorphic to the set $wff^1_\iota(\Sigma;\mathcal{W})\!\!\downarrow$, where $\mathcal{W}$ is a suitable set of variables of type ι. Indeed, first-order well-formed formulae cannot contain λ-abstractions: They cannot appear at argument positions of applications ($\mathbf{C}(\lambda X\mathbf{.A})$), since the corresponding function $\mathbf{C}$ would have a type of order 2, and they cannot appear at function position ($(\lambda X\mathbf{.A})\mathbf{C}$), since this would be a β-redex. Finally, they cannot appear at top position, since ι is a base type.

It is matter of folklore that equality can directly be expressed in $\mathcal{HOL}$ e.g. by the **Leibniz formula** for equality in terms of the other connectives from $\Sigma^{\mathcal{HOL}}$.

Definition 1.2.6 (Leibniz' Formulation for Equality)
We define the **Leibniz formula** for equality by

$$\mathbf{Q}^\alpha := (\lambda X_\alpha Y_\alpha \mathbf{.} \forall P_{\alpha\to o}\mathbf{.} PX \Rightarrow PY)$$

EdNote(2) With this definition, the formula $(\mathbf{A} \doteq^\alpha \mathbf{B})$[2], which we use as an abbreviation for $\mathbf{Q}^\alpha\mathbf{AB}$ β-reduces to $\forall P_{\alpha\to o}\mathbf{.}(P\mathbf{A}) \Rightarrow (P\mathbf{B})$, which can be read as: formulae $\mathbf{A}$ and $\mathbf{B}$ are not equal, iff there exists a discerning property P. In other words, $\mathbf{A}$ and $\mathbf{B}$ are equal, if they are indiscernible. Note that by contraposition we easily get the backward direction of ⇐ and hence it is sufficient to use ⇒ instead of ⇔. We will semantically justify this definition in Lemma 2.1.15.

In this book, we will differentiate between five different notions of equality. In order to prevent misunderstandings we explain these different notions together with their syntactical representation here:

Notation 1.2.7 If we *define* a concept we use $:=$ (e.g. let $\mathcal{D} := \{\mathtt{T},\mathtt{F}\}$) and $\equiv$ represents **Meta-equality**. We refer to the equality relation as an *object* of our *semantical domain*s with q; note that we possibly have one q^α in each domain $\mathcal{D}_\alpha$. The remaining two notions, $\doteq$ and $=$, are related to syntax. $=^\alpha$ may occur as a *constant symbol* of type α in a signature Σ and finally $\doteq^\alpha$ (and sometimes also $\mathbf{Q}^\alpha$) for *Leibniz equality.*

There are alternatives to obtaining a logical system by defining equality in terms of the logical connectives. We can for instance take the opposite approach using the definitions in Figure 1.1 and define the logical constants from a set of primitive equality constants (from $\Sigma^=$ in Definition 1.2.1). We

[2] EdNote: Chad: here equality

will discuss the semantical issues involved in these definitions in chapter 2. For the moment let us just note that under reasonable assumptions, both lead to equivalent logical systems.

conditional choice operator

Let us now discuss the definitions: truth is simple to define given equality, we simply make use of reflexivity. Universal quantification is defined as a predicate that checks whether its argument is equal to the universal set $(\lambda X.\mathbf{T}_o)$. The intuition behind the definition of conjunction is that $X \wedge Y$ is true, iff the pair (X, Y) is the pair $(\mathbf{T}_o, \mathbf{T}_o)$. Here a pair is modelled by abstraction over binary functions (the only thing we can rely on is argument order). Finally, falsity is using the difference between the constant function $\lambda X.\mathbf{T}_o$ and the identity function. Note that this only works since we have two truth values, (otherwise the functions coincide).

$\mathbf{T}_o$	$:=$	$q^o = q^o$
Π^α	$:=$	$q^{\alpha\to o}(\lambda X_\alpha.\mathbf{T}_o)$
$\wedge_{o\to o\to o}$	$:=$	$(\lambda X_o Y_o.(\lambda G_{o\to o\to o}.G\mathbf{T}_o\mathbf{T}_o) = (\lambda G_{o\to o\to o}.GXY))$
$\mathbf{F}_o$	$:=$	$(\lambda X_o.\mathbf{T}_o) = (\lambda X_o.X)$
$\neg_{o\to o}$	$:=$	$q^o\mathbf{F}_o$

Figure 1.1: Alternative Formalization for higher-Order Logic ($\mathcal{HOL}^=$)

Remark 1.2.8 (Further Logical Constants for $\mathcal{HOL}$)
Some formalizations of higher-order logic also consider the constants in Figure 1.2. These operators allow to specify further functions and relations in the λ-calculus.

The first two — **w** and **if** — are **conditional**s. **w** is a binary conditional, returning the second argument, when the first is true, **if** is the usual "if-then-else" conditional, it behaves like **w**, but returns the third argument, if the first is false. Using these operators, we can define e.g. step-functions.

The ι constant is Church's description operator that returns the unique element of a singleton set. It is more expressive than the conditionals, since they can be defined using ι: for instance we have $\mathbf{if} = \lambda X_o Y_\alpha Z_\alpha.\iota(\lambda W_\alpha.(X \wedge W = Y) \vee (\neg X \wedge W = Z)$. Its strength lies in the fact that for constructing an object (e.g. the value of a function) it is sufficient to be able to construct a λ-term for a property that uniquely determines it, i.e. describes (hence the name) it. This (as in the case of the conditional above) can often be done ($(X \wedge W = Y) \vee (\neg X \wedge W = Z)$ is the singleton with the right element), even if the object itself cannot be expressed as a λ-term.

The γ[3] constant is the **choice operator** that chooses a member from a non-empty set. Note that ι is a special case of the γ. EdNote(3)

[3] EDNOTE: Chad: why γ, and not ξ

axiom of choice
choice!axiom of
Dom(Φ), $\Im(\Phi)$*domain and image of* Φ
$\mathsf{A} \times \mathsf{B}$*Cartesian product*
Cartesian product

The existence of a choice operator — which is often stated as the **axiom of choice** (see below) — is such a strong assumption, that some mathematicians do not want to use it unrestrictedly.

Definition 1.2.9 (Axiom of Choice)
The following sentence is a formulation of the axiom of choice we will use in this book. For other formulations see e.g. [And86].

$$\exists C_{(\alpha\to o)\to\alpha}.\forall P_{\alpha\to o}.(\exists X_\alpha.PX) \Rightarrow P(CP)$$

name	op	type	example	Intuitive meaning
when	**w**	$o \to \alpha \to \alpha$	$\mathbf{w}\mathbf{A}_o\mathbf{B}_\alpha$	if **A**, then **B**
		$\forall X_\alpha.\mathbf{A} \Rightarrow (\mathbf{w}\mathbf{A}X) = X$		
if	**if**	$o \to \alpha \to \alpha \to \alpha$	$\mathbf{if}\mathbf{A}_o\mathbf{B}_\alpha\mathbf{C}_\alpha$	if **A**, then **B** else **C**
		$\forall X_\alpha.\mathbf{A} \Rightarrow (\mathbf{if}\mathbf{A}X\mathbf{B}) = X \wedge .\neg\mathbf{A} \Rightarrow (\mathbf{if}\mathbf{A}\mathbf{C}X) = X$		
descr.	ι	$(\alpha \to o) \to o$	$\iota\mathbf{P}_{\alpha\to o}$	a, if **P** is $\{\mathsf{a}\}$
		$\forall P_{\alpha\to o}.(\exists^1 X_\alpha.PX) \Rightarrow .\forall Y_\alpha.PY \Rightarrow \iota P = Y$		
choice	γ	$(\alpha \to o) \to o$	$\gamma\mathbf{P}_{\alpha\to o}$	some a in **P**, if $\mathbf{P} \neq \emptyset$
		$\forall P_{\alpha\to o}.(\exists X_\alpha.PX) \Rightarrow P(\gamma P)$		

Figure 1.2: Further logical operators in higher-order logic

1.2.2 λ-Calculus and Set Theory

The purpose of the following section is twofold. We will review the well-known set-theoretic foundations of functions and relations that will be used throughout this book (in particular in the formal semantics of $\Lambda^\to$ in section 1.4.2), and we will show how some of these concepts can be modeled in the simply typed λ-calculus $\Lambda^\to$ itself.

Note that in the definitions below (which can similarly be found in many introductory mathematics books, we will use set constructions in a naive way, which — as we have seen in the introduction — may be dangerous. In particular, we will use the set-construction $\{\mathsf{a} \mid \mathbf{A}\}$, which is dangerous, since it can also be used to construct Russell's paradox. Axiomatic set theories restrict the naive principle of set comprehension (every set that can be written down exists) by stating a carefully chosen set of axioms for safe set comprehension principles; For instance in definition 1.2.10, we use (implicitly) an axiom that guarantees the existence of sets of pairs. Similarly, all the other constructions can be carefully reconstructed in axiomatic set theories (see for instance [Ebb77] for an introduction).

Definition 1.2.10 (Relations)
Let A, B, and C be sets, then the **Cartesian product** $\mathsf{A} \times \mathsf{B}$ of A and B is

the set of pairs $\{(a, b) \mid a \in A, b \in B\}$. A **binary relation** Φ on $A \times B$ is a subset of the Cartesian product $\Phi \subseteq A \times B$, its **domain** $\mathbf{Dom}(\Phi)$ is the set $\{a \in A \mid \exists b.(a, b) \in \Phi\}$, its **image** $\Im(\Phi)$ is the set $\{b \in B \mid \exists a.(a, b) \in \Phi\}$. Let $\Psi \subseteq B \times C$ be another relation, then the **composition** of Φ and Ψ is defined by $\Psi \circ \Phi := \{(a, c) \mid \exists b.(a, b) \in \Phi, (b, c) \in \Psi\}$, if $\Im(\Phi) \subseteq \mathbf{Dom}(\Psi)$. The relation $\Phi^{-1} := \{(b, a) \mid (a, b) \in \Phi\} \subset B \times A$ is called the **inverse** relation for Φ. We call Φ **left** (**right**) **unique**, if it does not contain two different pairs having the same first (second) components. It is called **total**, if $\mathbf{Dom}(\Phi) = A$, and **surjective**, if $\Im(\Phi) = B$, in this case the inverse relation is total. Let $a \in \mathbf{Dom}(\Phi)$, then the **application** $\Phi(a)$ of Φ to $a \in A$ is the set $\{b \in B \mid (a, b) \in \Phi\}$. We sometimes write $\Phi(a, b)$, if $(a, b) \in \Phi$.

Definition 1.2.11 (Special Relations)
Let A be a set, then we call a relation $\Phi \subset A \times A$

- **reflexive**, iff $(a, a) \in \Phi$ holds for all $a \in A$.
- **symmetric**, iff $(a, b) \in \Phi$ implies $(b, a) \in \Phi$.
- **antisymmetric**, iff $(a, b) \in \Phi$ implies $(b, a) \notin \Phi$.
- **transitive**, iff $(a, b), (b, c) \in \Phi$ implies $(a, c) \in \Phi$.
- an **equivalence relation**, iff Φ is reflexive, symmetric, and transitive.
- a **quasi-ordering**, iff Φ is transitive and reflexive.

Let $\preceq$ be a quasi-ordering, then we call the relation $\sim\ :=\ \preceq \cap \preceq^{-1}$ the equivalence **induced** by $\preceq$, and $\prec\ := \{(a, b) \mid a \preceq b$ but $a \not\sim b\}$ the **strict ordering** for $\preceq$. A quasi-ordering $\preceq$ is called a **partial ordering**, iff $\sim$ is trivial, i.e. $x \sim y$, iff $x = y$. It is called **terminating** or **well-founded**, iff there are no infinite sequences $a_1, a_2, \ldots$ with $a_i \in A$ and $a_{i+1} \prec a_i$. A transitive relation Φ is called **confluent**, iff for all $a, b, c \in A$ with $\Phi(a, b)$ and $\Phi(a, c)$, there is $d \in A$ such that $\Phi(b, d)$ and $\Phi(c, d)$.

For an equivalence relation $\sim\ \subset A \times A$ we denote the **equivalence class** of $a \in A$ by $[\![a]\!] := [\![a]\!]_\sim := \{b \in A \mid b \sim a\}$.

Definition 1.2.12 (Partial Function)
We call a left unique relation a **partial function**. It is called **injective**, iff it is also right unique. We denote the family of all partial functions $\Phi \subseteq A \times B$ by $\mathcal{F}_p(A; B)$ and the family of all total relations by $\mathcal{R}^t(A; B)$. The set $\mathcal{F}(A; B) := \mathcal{R}^t(A; B) \cap \mathcal{F}_p(A; B)$ is called the set of **total function**s.

Let Φ be a partial function and $a \in \mathbf{Dom}(\Phi)$, then the **application** $\Phi(a)$ of a partial function Φ to $a \in A$ is the unique $b \in B$ such that $(a, b) \in \Phi$. In order to make the presentation of partial functions simpler, we introduce a

binary relation
relation!binary
domain
image
composition
inverse
relation!inverse
total
surjective
application
$\sim$*equivalence relation induced by a quasi-ordering*
$[\![f]\!]_\sim$*equivalence class of* f *under* $\sim$
reflexive
symmetric
antisymmetric
transitive
equivalence relation
quasi-ordering
induced
strict ordering
ordering!strict
partial ordering
terminating
well-founded
confluent
equivalence class
$\mathcal{F}_p(A; B), \mathcal{F}(A; B)$*families of partial and total functions*
$\varphi|_W$ *restriction of* Φ *to* W
$\Phi|\!|\Psi$*partial functions* Φ *and* Ψ *agree*
$\Phi, [b/a]$*augmented partial function*
$\Phi \cup \Psi$*union of partial functions*
Φ_{-c}*partial function* Φ *without value for* c
partial function
function!partial
injective
total function
function!total
application

function!composition
composition!function
restriction
function!restriction
agree
Cartesian product
n-ary
domain
image
Currying
extensionality

special symbol $\perp$ (for the undefined) and extend the definition of function application by $\Phi(\mathsf{c}) := \perp$, if $\mathsf{c} \in \mathsf{A}$, but $\mathsf{c} \notin \mathbf{Dom}(\Phi)$, or if $\mathsf{c} = \perp$ itself. This often allows to omitting the reasoning about domains. In particular, we have $\mathsf{f} \circ \mathsf{g}(\mathsf{a}) = \mathsf{f}(\mathsf{g}(\mathsf{a}))$ for $\mathsf{g} \in \mathcal{F}_p(\mathsf{A};\mathsf{B})$, $\mathsf{f} \in \mathcal{F}_p(\mathsf{B};\mathsf{C})$, and $\mathsf{a} \in \mathsf{A}$ independently of definedness considerations. Note that the symbol $\perp$ is not an object in any of the given sets, but rather a syntactic trick that eases notation. The so-called **function composition** $\circ$ is associative and therefore the sets $\mathcal{F}_p(\mathsf{A};\mathsf{A})$ and $\mathcal{F}(\mathsf{A};\mathsf{A})$ are monoids with this operation.

If $\Phi \in \mathcal{F}_p(\mathsf{A};\mathsf{B})$ and $\Psi \in \mathcal{F}_p(\mathsf{A};\mathsf{B})$ are partial functions such that $\Psi(\mathsf{a}) = \mathsf{b}$, but $\Psi(\mathsf{c}) = \Phi(\mathsf{c})$ for all $\mathsf{c} \neq \mathsf{a}$, then we denote Ψ by $\Phi, [\mathsf{b}/\mathsf{a}]$. For partial functions that can be presented by a finite set of pairs (e.g. substitutions and variable contexts), we often use the notation $\Phi := [\mathsf{b}^1/\mathsf{a}^1], \ldots, [\mathsf{b}^n/\mathsf{a}^n]$, if $\Phi = \{(\mathsf{a}^1, \mathsf{b}^1), \ldots, (\mathsf{a}^n, \mathsf{b}^n)\}$. Furthermore, we denote with $\Phi_{-\mathsf{c}}$ the partial function $\{(\mathsf{a}, \mathsf{b}) \in \Phi \mid \mathsf{a} \neq \mathsf{c}\}$.

Let $\mathsf{W} \subset \mathsf{A}$ and $\Phi \in \mathcal{F}_p(\mathsf{A};\mathsf{B})$, then the **restriction** of Φ to W is defined to be the function $\Phi|_{\mathsf{W}} := \{(\mathsf{a}, \mathsf{b}) \in \Phi \mid \mathsf{a} \in \mathsf{W}\}$. Note that $\Phi|_{\mathsf{W}} \in \mathcal{F}_p(\mathsf{W};\mathsf{B})$ and that with this definition Φ and $\Phi|_{\mathsf{W}}$ are equal iff $\mathsf{W} \cap \mathbf{Dom}(\Phi) = \mathbf{Dom}(\Phi)$. If Φ and Ψ are partial functions such that their restrictions on $\mathbf{Dom}(\Phi) \cap \mathbf{Dom}(\Psi)$ are identical, then we say that they **agree** and write $\Phi||\Psi$. In this case the set-theoretic union $\Phi \cup \Psi$ is again a partial function.

Remark 1.2.13 (n-ary Relations and Functions)
Let $\mathsf{A}_1, \ldots, \mathsf{A}_n$ be sets, then we can define the n-fold **Cartesian product** $\mathsf{A}_1 \times \cdots \times \mathsf{A}_n$ by $(\cdots(A_1 \times A_2) \times \cdots \times A_n)$, thus it is the set of ordered n-tuples $\{(\mathsf{a}_1, \ldots, \mathsf{a}_n) \mid \mathsf{a}_i \in \mathsf{A}_i\}$, where $(\mathsf{a}_1, \ldots, \mathsf{a}_n) := (\cdots(\mathsf{a}_1, \mathsf{a}_2), \ldots, \mathsf{a}_n)$. With this definition we can generalize the previous definitions for binary relations and unary functions to **n-ary** relations and n-ary functions. In particular, the **domain** $\mathbf{Dom}(\Phi)$ of an n-ary relation Φ is the set $\{(\mathsf{a}_1, \ldots, \mathsf{a}_{n-1}) \mid \exists \mathsf{b}. (\mathsf{a}_1, \ldots, \mathsf{a}_{n-1}, \mathsf{b}) \in \Phi\}$, the **image** $\Im(\Phi)$ of Φ is the set $\{\mathsf{a}_n \mid \exists \mathsf{a}_1, \ldots, \mathsf{a}_{a-1}. (\mathsf{a}_1, \ldots, \mathsf{a}_{n-1}, \mathsf{a}_n) \in \Phi\}$. We denote the family of n-ary total relations by $\mathcal{R}^t(\mathsf{A}_1, \ldots, \mathsf{A}_n; \mathsf{B})$ and adapt the other notions accordingly.

This construction implies that we can use unary functions instead of general n-ary ones. In particular, the well-known process of applying an n-ary function Φ to an n-tuple $(\mathsf{a}_1, \ldots, \mathsf{a}_n)$ can be considered as applying Φ to the sequence of values $\mathsf{a}_1, \ldots, \mathsf{a}_n$ one after the other. Thus the application of Φ to a tuple $(\mathsf{a}_1, \ldots, \mathsf{a}_k)$ yields an $(n-k)$-ary function that gives a_{n+1}, if applied to the tuple $(\mathsf{a}_{k+1}, \ldots, \mathsf{a}_n)$. This process is called **Currying**. Therefore $\mathcal{F}(\mathsf{A}_1, \ldots, \mathsf{A}_n; B)$ becomes $\mathcal{F}(\mathsf{A}_1; \mathcal{F}(\mathsf{A}_2; \ldots; \mathcal{F}(\mathsf{A}_n; \mathsf{B}) \ldots)$, and therefore we can restrict ourselves to unary functions.

Remark 1.2.14 (Extensionality)
Two partial functions $\mathsf{f}, \mathsf{g} \colon \mathsf{A} \longrightarrow \mathsf{B}$ are equal, iff they are equal as binary relations, that is, if $\mathbf{Dom}(\mathsf{f}) = \mathbf{Dom}(\mathsf{g})$ and for all $\mathsf{a} \in \mathbf{Dom}(\mathsf{f})$ we have $\mathsf{f}(\mathsf{a}) = \mathsf{g}(\mathsf{a})$. This property is called the **extensionality** of equality.

characteristic function
function!characteristic
ξ_{S}*characteristic function*

We have seen the set-theoretic approach to formalizing mathematics, which uses the notion of a set as a primitive notion and builds up the notion of a function from this. An alternative way is to build a higher-order typed set theory from the system $\mathcal{HOL}$, which takes the notion of a function as primitive, since it is built on top of the simply typed λ-calculus.

The most important concept is to model a set as its **characteristic function**: for any set $\mathsf{S} \subseteq \mathcal{D}_\alpha$, the characteristic function $\xi_{\mathsf{S}}:\mathcal{U} \longrightarrow \{\mathtt{T},\mathtt{F}\}$, is defined by $\xi_{\mathsf{S}}(\mathsf{a}) = \mathtt{T}$, iff $\mathsf{a} \in \mathsf{S}$. Since we have $\{\mathtt{T},\mathtt{F}\} = \mathcal{D}_o$, in $\mathcal{HOL}$, we can model a set $\mathsf{S} \subseteq \mathcal{D}_\alpha$ as a function $\mathsf{f}_{\mathsf{S}} \in \mathcal{D}_{\alpha\to o}$[1] and thus as a well-formed formula in $\mathit{wff}_{\alpha\to o}(\Sigma)$.

Definition 1.2.15 (Elementary Typed Set Theory)
We can now obtain the usual set-theoretic functions as abbreviations of λ-terms as indicated in Figure 1.3. These reveal their intended behavior by β-reduction: e.g. $\mathbf{A} \in^\alpha \mathbf{S}$ β-reduces to $\mathbf{SA}$ and $\mathbf{S} \subseteq^\alpha \mathbf{T} =_\beta \forall X_\alpha . X \in^\alpha \mathbf{S} \Rightarrow X \in^\alpha \mathbf{T}$ (which amounts to the usual set-theoretic definition of the subset relation). Note that the (binary) subset relation can also be used as the

$$\begin{array}{|lcl|}\hline \emptyset^\alpha & = & \lambda X_\alpha . \mathbf{F}_o \\ \in^\alpha & = & \lambda X_\alpha . \lambda S_{\alpha\to o} . SX \\ \cup^\alpha & = & \lambda S_{\alpha\to o} T_{\alpha\to o} . \lambda X_\alpha . SX \vee TX \\ \cap^\alpha & = & \lambda S_{\alpha\to o} T_{\alpha\to o} . \lambda X_\alpha . SX \wedge TX \\ \subseteq^\alpha & = & \lambda S_{\alpha\to o} T_{\alpha\to o} . \forall X_\alpha . SX \Rightarrow TX \\ \hline\end{array}$$

Figure 1.3: Definitions for Sets

(unary) power set operator, i.e. $\lambda X_{\alpha\to o} . X \subseteq^\alpha S$ is the set of all subsets of $S \subseteq \mathcal{D}_\alpha$.

For functions, we will use the definitions in Figure 1.4 for the concepts introduced in Definition 1.2.12.

In this manner, we can define all of the set-theoretic primitives as defined notions in $\mathcal{HOL}$. In particular, we can even use the first-order fragment of $\mathcal{HOL}$ (cf. 1.2.5) to embed axiomatic set theory into $\mathcal{HOL}$, including the definition of functions, as in Definition 1.2.12, so that the restriction in expressivity of typed set theory noted in footnote 1 on page xxxiii is not a problem of principle, but of notational convenience. Finally, let us discuss a couple of mathematical examples that will play a role in the following. $\mathcal{HOL}$

[1]Note that this construction does mean a certain restriction in the set-construction possibilities compared to axiomatic set theories, since sets can only contain elements of a common type. for instance, the standard set-theoretic construction of natural numbers as the set $\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}, \{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}\}, \ldots$ is impossible, since if $\emptyset$ is modeled as a function of type $\alpha \to o$, then $\{\emptyset\}$ must be of type $(\alpha \to o) \to o$.

$$
\begin{array}{rcl}
\mathbf{Dom}^{\alpha\beta} & = & \lambda F_{\alpha\to\beta}.\lambda X_{\alpha}.\exists Y_{\beta}.FX = Y \\
 & \widehat{=} & \text{Domain of a function} \\
\Im^{\alpha\beta} & = & \lambda F_{\alpha\to\beta}.\lambda X_{\beta}.\exists Y_{\alpha}.FY = X \\
 & \widehat{=} & \text{Image of a function} \\
\mathcal{F}^{\alpha\beta} & = & \lambda F_{\alpha\to\beta}.\lambda S_{\alpha\to o}.\lambda T_{\alpha\to o}.\forall X_{\alpha}.SX \Rightarrow T(FY) \\
 & \widehat{=} & \text{Function set constructor} \\
\mathsf{inj}^{\alpha\beta} & = & \lambda F_{\alpha\to\beta}.\lambda S_{\alpha\to o}.\forall X_{\alpha}, Y_{\alpha}.(SX \wedge SX \wedge FX = FY) \Rightarrow X = Y \\
 & \widehat{=} & \text{Injectivity on } \mathsf{S} \\
\mathsf{surj}^{\alpha\beta} & = & \lambda F_{\alpha\to\beta}.\lambda S_{\alpha\to o}.\lambda T_{\beta\to o}.\forall X_{\beta}.SX \Rightarrow \exists Y_{\alpha}.TX \wedge X = FY \\
 & \widehat{=} & \text{Surjectivity on } \mathsf{S} \text{ onto } \mathsf{T}
\end{array}
$$

Figure 1.4: Definitions for Functions

allows to give a direct account of Peano's axiomatization of natural numbers instead of the construction mentioned there.

Example 1.2.16 (Peano Axioms)
The natural numbers are formalized using the constant 0_ι for the number zero, and the constant $s_{\iota\to\iota}$ for the successor function. The set of natural numbers is represented by the predicate constant $\mathbb{N}_{\iota\to o}$ and and the following set of axioms.

1. $0 \in \mathbb{N}$ (zero is a natural number)
2. $s \in \mathcal{F}(\mathbb{N}\mathbb{N}) \to_\beta^* \forall X_\iota.(\mathbb{N}X) \Rightarrow \mathbb{N}(sX)$
   (the successor of a natural number is a natural number)
3. $\neg\exists X_\iota.(\mathbb{N}X) \wedge sX = 0$ (zero has no predecessor)
4. $(\mathsf{inj}s\mathbb{N}) \to_\beta^* \forall X_\iota, Y_\iota.\mathbb{N}X \wedge \mathbb{N}Y \wedge (sX = sY) \Rightarrow X = Y$
   (the successor function is one-one)
5. $\forall P_{\iota\to o}.(P0 \wedge \forall X_\iota.(\mathbb{N}X \wedge PX) \Rightarrow P(sX)) \Rightarrow .\forall Z_\iota.\mathbb{N}Z \Rightarrow PZ$
   (Induction Axiom: all properties P that hold on zero and with every natural number hold on its successor must hold on all natural numbers.)

Instead of using Peano's axioms to characterize the natural numbers as the inductively defined set of terms constructed by s and 0, we can represent them directly as closed λ-terms, the so-called **Church numerals**. The idea is to use variables for s and 0, and close the terms by λ-abstraction.

Example 1.2.17 (Church Numerals)
For any natural number $n \in \mathbb{N}$, we define the n^{th} **Church numeral** $\underline{n}$ by

$$\underline{n} := \lambda S_{\iota\to\iota}.\lambda 0_\iota.\underbrace{s(\ldots(s}_{n}\,0)\cdots)$$

All $\underline{n}$ have type $(\iota \to \iota) \to \iota \to \iota$, which we will abbreviate by ν.

We can interpret $\underline{n}$ as an iterator function that takes as arguments a function f and a start value a and iterates f n times on a, in particular $\underline{0}$ is just the constant function $\lambda S_{\iota\to\iota}.\lambda 0_\iota.0$.

$\underline{n}$ *The n^{th} Church numeral*
ν *The type of Church numerals*
iterator
function!iterator
Cantor

This allows us to define closed λ-terms for some arithmetical operations:

addition $\underline{+} := \lambda N_\nu M_\nu.\lambda S_{\alpha\to\alpha}.\lambda O_\alpha.NS(MSO)$ (N-fold iteration of S on MSO)

multiplication $\underline{\times} := \lambda N_\nu M_\nu.\lambda S_{\alpha\to\alpha}.\lambda O_\alpha.N(MS)O$ (N-fold iteration of MS (=+m) on O)

We leave exponentiation as an exercise to the reader. To convince ourselves that the operations defined above will compute $\underline{1+2}$ with β-reduction:

$$\begin{array}{rcl}
\underline{1+2} & = & (\lambda N_\nu M_\nu.\lambda S_{\alpha\to\alpha}.\lambda O_\alpha.NS(MSO))(\lambda S_{\iota\to\iota}.\lambda O_\iota.SO)(\lambda S_{\iota\to\iota}.\lambda O_\iota.S(SO)) \\
& \to_\beta & \lambda S_{\alpha\to\alpha}.\lambda O_\alpha.((\lambda S_{\iota\to\iota}.\lambda O_\iota.SO)S)((\lambda S_{\iota\to\iota}.\lambda O_\iota.S(SO))SO) \\
& \to_\beta^3 & \lambda S_{\alpha\to\alpha}.\lambda O_\alpha.(\lambda O_\iota.SO)(S(SO)) \\
& \to_\beta & \lambda S_{\alpha\to\alpha}.\lambda O_\alpha.S(S(SO)) \\
& = & \underline{3}
\end{array}$$

Somewhat surprisingly, we cannot define "decreasing" operations like subtraction or division using λ-terms alone (if we could, we could reduce arithmetic to $\beta\eta$-equality, which is decidable by 1.3.16). We will however be able to use higher-order unification for those operations, see example 3.5.3.

Even complex assertions, such as Cantor's theorem on the uncountability of real numbers can represented in higher-order logic in a natural way.

Example 1.2.18 (Cantor's Theorem)
The following formula is a variant of Cantor's theorem of the uncountability of $\mathbb{R}$.

$$\neg\exists F_{\iota\to\iota\to\iota}.F \in \mathcal{F}\mathbb{N}(\mathcal{F}\mathbb{N}\mathbb{N}) \wedge \mathsf{surj}F\mathbb{N}(\mathcal{F}\mathbb{N}\mathbb{N})$$

Instead of the real numbers, we simply take the sequences of natural numbers (functions from $\mathbb{N}$ to $\mathbb{N}$) here, ignoring the fact that real numbers only contain digits and that $0.9999999\ldots = 1$. This formula β-reduces to

$$\begin{array}{l}
\neg\exists F_{\iota\to\iota\to\iota}.\forall Z_\iota.(\mathbb{N}Z \Rightarrow \forall W_\iota.\mathbb{N}W \Rightarrow \mathbb{N}(FZW))\wedge \\
\qquad \forall G_{\iota\to\iota}.\forall Z_\iota.(\mathbb{N}Z \Rightarrow \mathbb{N}(GZ)) \Rightarrow \exists J_\iota.\mathbb{N}J \wedge FJ = G
\end{array}$$

after expanding all the definitions. The mapping F that is claimed not to exist maps natural numbers (type ι) to functions (type $\iota \to \iota$), thus it has type $\iota \to \iota \to \iota$.

In order to simplify the problem we take the type ι to be the set of natural numbers, thus Cantor's theorem has the form

$$\neg\exists F_{\iota\to\iota\to\iota}.\forall G_{\iota\to\iota}.\exists J_\iota.FJ = G$$

Montague
semantics construction procedure
construction procedure!semantics
procedure!semantics construction
compositional

We will use this simplified form as a running example for the automated theorem proving calculi in part **??**. In particular, the simplification makes it provable by current higher-order theorem provers.

In part **??** of this book, we will exhibit sorted variants of these calculi that provide special mechanisms (sorts) for treating information about domains and codomains of functions that reduce the search spaces induced by the original example to that of the simplified one.

1.2.3 Montague Grammar

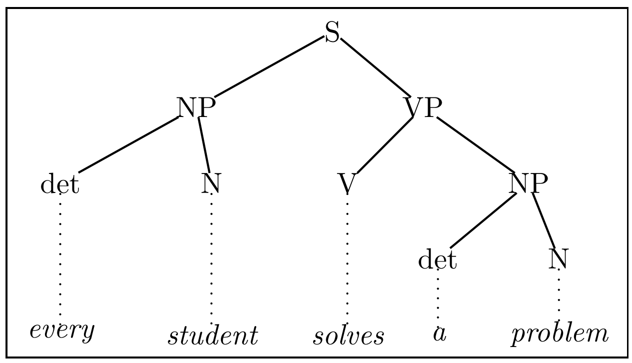
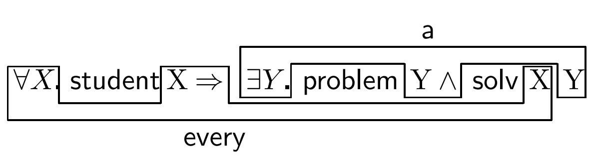


Figure 1.5: Every student solves a problem

Let us now take a look a different kind of application of β-reduction. Following [Mon74], the simply typed λ-calculus is used as a representation language for the semantics of (meaningful parts of) natural language utterances. The β-equality that is built into the represenation formalism makes it possible to define a compositional **semantics construction procedure**s. A semantics construction procedure, i.e. an algorithm that transforms parse trees that represent the syntactical form of a sentence as the one in Figure 1.5 into (a set of) logical formulae that represent its meaning. It is called **compositional**, iff the procedure not only computes the formulae, but does it in a way that gives a meaning to every node in the parse tree.

Consider for instance the sentence "*Every Student solves a problem*". This has the parse tree (syntactic analysis) given in Figure 1.5, but the logical representation $\forall X.\mathsf{student}X \Rightarrow \exists Y.\mathsf{problem}Y \wedge \mathsf{solv}XY$. If we take a look at the contributions of the respective words, then we obtain the following picture:



compositionality
computational linguistics

In particular the contribution of the determiners "*every*" and "*a*" cannot simply be localized. Richard Montague proposed to interpret the these words functionally (as higher-order relations between concepts) and use β-reduction to compute the semantics. For instance the semantics for the sentence "*A student sleeps*" is constructed in Figure 1.6. The syntactic analysis (in terms of the syntactic categories S: sentence, NP: nominal phrase, VP: verb phrase, DET: determiner, N: noun, V:verb) is isomorphic to the semantical analysis with respect to the following lexicon.

Word	Representation	Word	Representation
a	$\lambda P.\lambda Q.\exists X.PX \wedge QX$	*student*	$\lambda Z.\mathsf{student}Z$
every	$\lambda P.\lambda Q.\forall X.PX \Rightarrow QX$	*sleeps*	$\lambda W.\mathsf{sleep}W$

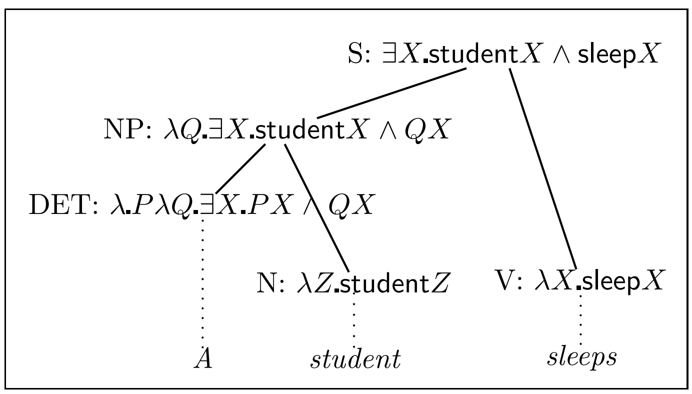


Figure 1.6: Compositional semantics construction

Therefore we have written it into a single tree. Note that the semantics (given as a λ-term) of the internal nodes of the tree has been constructed by applying the semantics of the left daughter to that of the right daughter, and β-reducing. For the semantics of the NP node we have

$$\begin{aligned}(\lambda P.\lambda Q\exists X.PX \wedge QX)(\lambda Y.\mathsf{student}Y) &\to_\beta \lambda Q.\exists X.(\lambda Y.\mathsf{student}Y)X \wedge QX \\ &\to_\beta \lambda Q.\exists X.\mathsf{student}X \wedge QX\end{aligned}$$

and for the sentence semantics

$$(\lambda Q.\exists X.\mathsf{student}X \wedge QX)(\lambda Z.\mathsf{sleep}Z) \longrightarrow_\beta \exists X.\mathsf{student}X \wedge \mathsf{sleep}X$$

Thus we can see that — given the right higher-order representations in the lexicon — the construction of the semantics of a natural language sentence can be seen as a totally logic-based process using β-reduction. Moreover, the semantics of any syntactically well-formed sentence fragment of the sentence can be given a semantics that only depends on that of its constituents. This property is called **compositionality** in computational linguistics.

head(A)*head of* **A**
head normal form
normal form!head
binder
matrix
head
rigid
flexible
projection
formula!projection
head redex
redex!head
head reduction step
reduction!head
internal

1.3 $\beta\eta$-Normal Forms

In this section we will establish the well-known strong normalization theorems for the simply typed λ-calculus. We will present a full proof of termination and confluence of the $\beta\eta$-reduction here, both since they are interesting in their own right, as well as in order to introduce the logical relation method due to Tait [Tai67] and Statman [Sta85], which we will need again in chapter 8. Before we do this, we need to establish some elementary definitions and results about $\beta\eta$-reduction.

Lemma 1.3.1 (β-Normal Form)
In a β-normal formula $\lambda\overline{X^n}.\mathbf{H}\overline{\mathbf{E}^m}$ where n and m are chosen maximally, $\mathbf{H}$ must be a constant or variable.

Proof: The assumption that $\mathbf{H}$ is an application contradicts the maximality of m, and assuming that $\mathbf{H}$ is a λ-abstraction $(\lambda Y.\mathbf{M})$ leads to a form $(\lambda\overline{X}.(\lambda Y.\mathbf{M})\overline{\mathbf{B}^k})$ that contains a β-redex and contradicts β-normality. □

Furthermore, we note that there cannot be β-redexes "between" the $\mathbf{E}^i$ (recall the bracketing conventions in 1.1.9), so that $\mathbf{A}$ is β-normal, iff the $\mathbf{E}^i$ are. This makes h a strong syntactic invariant of the λ-calculus and the form of $\mathbf{A}$ a good intermediate step in β-normalization.

Definition 1.3.2 (Head Normal Form)
Let $\mathbf{A} \equiv (\lambda\overline{X^n}.h\overline{\mathbf{E}^m})$ be a well-formed formula such that h is a constant or variable, then we say that $\mathbf{A}$ is in **head normal form**. The part $\lambda\overline{X^n}$ is called the **binder** of $\mathbf{A}$, the part $h\overline{\mathbf{E}^m}$ the **matrix**, and the constant or variable h is called the **head** of $\mathbf{A}$. We denote the head of $\mathbf{A}$ with $\mathbf{head}(\mathbf{A})$. We extend the function **head** by the definition $\mathbf{head}(\mathbf{A}) := k$, iff $\mathbf{A}$ is a k-projection formula. Otherwise the function **head** would be undefined for projection formulae, because the its head is some variable, whose name of no role outside the formula (we take alphabetic variants to be identical).

$\mathbf{A}$ is called **rigid**, iff h is a constant or a bound variable, otherwise **flexible**. If h is the bound variable X^k, then $\mathbf{A}$ is called a (k-)**projection** formula

A well-formed formula $\mathbf{A}$ that is not in head normal form must be of the form $\mathbf{A} \equiv (\lambda\overline{X}.(\lambda Y.\mathbf{M})\overline{\mathbf{B}^k})$. We call the redex $(\lambda Y.\mathbf{M})\mathbf{B}^1$ the **head redex** of $\mathbf{A}$ and the β-reduction step $(\lambda\overline{X}.(\lambda Y.\mathbf{M})\overline{\mathbf{B}^k}) \rightarrow_\beta (\lambda\overline{X}.[\mathbf{B}^1/Y]\mathbf{M}\mathbf{B}^2\ldots\mathbf{B}^k)$, which reduces this head redex, a **head reduction step**, any other β-reduction step will be called **internal**. We denote the head reduction relation by $\rightarrow^h$.

One simple but important step towards confluence of β-reduction is the following result.

Lemma 1.3.3 (Permutation of Head Reduction)
Let $\mathbf{A} \rightarrow^h \mathbf{B}$ and $\mathbf{A} \rightarrow^i \mathbf{C}$, then there is a formula $\mathbf{D}$, such that $\mathbf{B} \rightarrow^i \mathbf{D}$

and $\mathbf{C} \to^h \mathbf{D}$.

Proof: We conduct the proof by chasing the diagram in Figure 1.7 using lemma 1.3.4 below. □

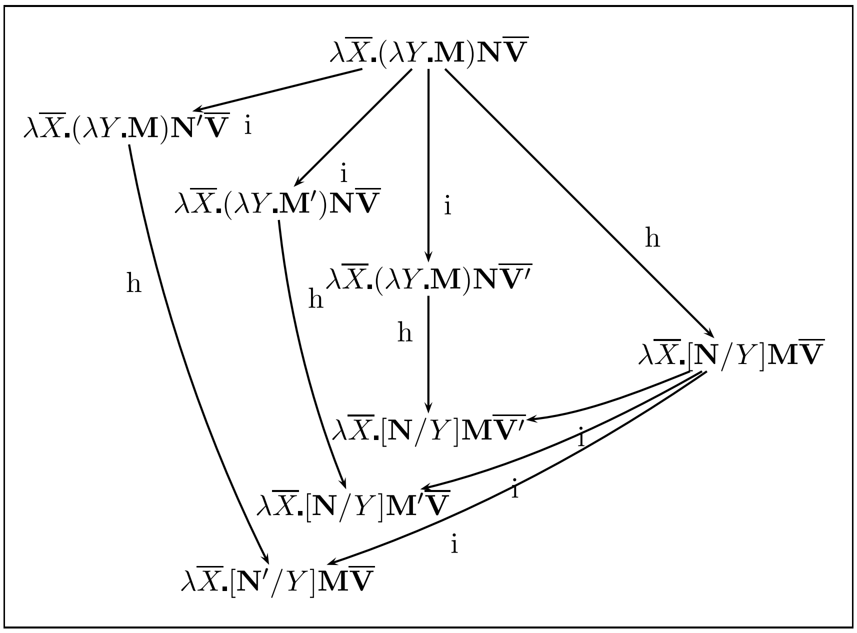


Figure 1.7: Permutation of head reduction

Lemma 1.3.4 (Substitutivity of $\beta\eta$-Reduction)
Let $\mathbf{A} \to_{\beta\eta} \mathbf{B}$, *then*

1. *any variable* $X \notin \mathbf{Free}(\mathbf{A})$ *is not free in* $\mathbf{B}$.
2. $[\mathbf{A}/X]\mathbf{C} \to_{\beta\eta} [\mathbf{B}/X]\mathbf{C}$
3. $[\mathbf{C}/X]\mathbf{A} \to_{\beta\eta} [\mathbf{C}/X]\mathbf{B}$

Proof: The first assertion is obvious, since X can only be free in $[\mathbf{D}/Y]\mathbf{E}$, if it already occurred freely in $(\lambda Y.\mathbf{E})\mathbf{D}$. The second assertion is a simple consequence of the fact that any $\beta\eta$-redex in $\mathbf{A}$ can also be reduced in $[\mathbf{A}/X]\mathbf{C}$.

For the third assertion it is sufficient to prove just the case of the contraction of one redex. For the case of β-contraction let $\mathbf{R} = (\lambda Y.\mathbf{E})\mathbf{D}$ be the redex in $\mathbf{A}$. Then $[\mathbf{C}/X]\mathbf{R} \equiv (\lambda Y.[\mathbf{C}/X]\mathbf{E})[\mathbf{C}/X]\mathbf{D}$, since by 1.1.14 we can assume that $Y \neq X$. This however β-contracts to $[[\mathbf{C}/X]\mathbf{D}/Y]([\mathbf{C}/X]\mathbf{E})$, which is an alphabetical variant of $[\mathbf{C}/X][\mathbf{D}/Y]\mathbf{E}$. For the η-case note that $[\mathbf{C}/X](\lambda Y.\mathbf{E}Y) \equiv (\lambda Y.[\mathbf{C}/X]\mathbf{E}Y) \to_\eta [\mathbf{C}/X]\mathbf{E}$. □

logical relation
relation!logical
induced logical relation
logical relation!induced
admissible

Having established these preliminaries, we can now come to the logical-relation method. We only develop the case of unary logical relations here, since this is sufficient for our purposes, see [Sta85] for the general theory.

Definition 1.3.5 (Logical Relation)
A unary typed relation $\mathcal{LR} \subset wff_\alpha(\Sigma)$ is called a **logical relation**, iff for all types $\alpha = \gamma \to \delta$ and all $\mathbf{A} \in wff_\alpha(\Sigma)$ we have $\mathcal{LR}_\alpha(\mathbf{A})$, iff $\mathcal{LR}_\delta(\mathbf{AC})$ for all $\mathbf{C} \in wff_\gamma(\Sigma)$ with $\mathcal{LR}_\gamma(\mathbf{C})$. Clearly, logical relations are totally determined by their values on base types.

For a substitution $\sigma \in \mathbf{SUB}(\Sigma)$, we say that σ is in $\mathcal{LR}$ (and write $\mathcal{LR}(\sigma)$), iff for all $X_\alpha \in \mathbf{Dom}(\sigma)$ we have $\mathcal{LR}_\alpha(\sigma(X))$.

For any other unary typed relation $\mathcal{SR}$, we call the (unique) logical relation $\mathcal{LR}$ that coincides with $\mathcal{SR}$ on base types the logical relation **induced** by $\mathcal{SR}$.

Definition 1.3.6 (Admissible)
Let $\mathcal{LR}$ be a logical relation, then a head reduction

$$(\lambda X_\alpha.\mathbf{A}_\alpha)\mathbf{B} \to_\beta [\mathbf{B}/X]\mathbf{A}$$

is called **admissible**, iff $\mathcal{LR}_\beta(\mathbf{B})$. A logical Relation R is said to be admissible if $\mathcal{LR}$ is closed under admissible head expansions. (Formally: Let α be a base type, and let $\mathbf{A}_\alpha, \mathbf{B}_\alpha$ be formulae, such that $\mathbf{A} \to^h \mathbf{B}$ is admissible, then a logical relation $\mathcal{LR}$ is called admissible if $\mathcal{LR}_\alpha(\mathbf{B})$ implies that $\mathcal{LR}_\alpha(\mathbf{A})$.)

The key tool for the logical relation method is the fundamental theorem for logical relations (1.3.8). This is a simple consequence of the following lemma, which uses a stronger assertion to make the induction go through.

Lemma 1.3.7 *Let $\mathcal{LR}$ be an admissible logical relation, such that $\mathcal{LR}_\alpha(\mathbf{C})$ for all $\mathbf{C} \in \mathcal{V}_\alpha \cup \Sigma_\alpha$. If $\theta \in \mathbf{SUB}(\Sigma)$ and $\mathcal{LR}(\theta)$, then $\mathcal{LR}_\alpha(\theta(\mathbf{A}))$ for all $\mathbf{A} \in wff_\alpha(\Sigma)$.*

Proof: We prove the assertion by induction on the structure of $\mathbf{A}$. If $\mathbf{A}$ is a constant, or a variable not in $\mathbf{Dom}(\theta)$, then the assertion trivial. If $\mathbf{A} = X_\alpha$ and $\mathcal{LR}_\alpha(\mathbf{A})$ by assumption. If $\mathbf{A} = \mathbf{B}_{\gamma\to\alpha}\mathbf{C}_\gamma$, then by inductive hypothesis we have $\mathcal{LR}_{\gamma\to\alpha}(\theta(\mathbf{B}))$ and $\mathcal{LR}_\gamma(\theta(\mathbf{C}))$ and therefore $\mathcal{LR}_\alpha(\theta(\mathbf{BC}))$ since $\mathcal{LR}$ is logical.

Finally, if $\mathbf{A} = \lambda X_\gamma.\mathbf{B}$, then we will show that $\mathcal{LR}_\delta(\theta(\mathbf{A})\mathbf{C})$ for all $\mathbf{C} \in wff_\gamma(\Sigma)$ with $\mathcal{LR}_\gamma(\mathbf{C})$. Without loss of generality, we can assume that $X \notin \mathbf{Dom}(\theta)$. Now let $\theta' := \theta, [\mathbf{C}/X_\gamma]$. Clearly, we have $\mathcal{LR}(\theta')$, Thus by induction hypothesis we have $\mathcal{LR}_\delta(\theta'(\mathbf{B}))$. Furthermore, $\theta(\mathbf{A})\mathbf{C} = (\lambda X_\gamma.\theta(\mathbf{B}))\mathbf{C}$ reduces to $\theta'(\mathbf{B})$ in an admissible head reduction and therefore we have $\mathcal{LR}_\delta((\lambda X_\gamma.\mathbf{B})\mathbf{C}))$ by admissibility of $\mathcal{LR}$. $\square$

Theorem 1.3.8 (Fundamental Theorem for Logical Relations) *Let $\mathcal{LR}$ be an admissible logical relation, such that $\mathcal{LR}_\alpha(\mathbf{C})$ for all $\mathbf{C} \in \mathcal{V}_\alpha \cup \Sigma_\alpha$. Then $\mathcal{LR}_\alpha(\mathbf{A})$ for all $\mathbf{A} \in wff_\alpha(\Sigma)$,.*

confluent at
terminating at

Proof: The empty substitution is obviously (vacuously) in $\mathcal{LR}$. ☐

Definition 1.3.9 (Confluent and Terminating at A)
We say that $\beta\eta$-reduction is **confluent at A**, iff for any formulae $\mathbf{B}$ and $\mathbf{C}$ with $\mathbf{A} \to^*_{\beta\eta} \mathbf{B}$ and $\mathbf{A} \to^*_{\beta\eta} \mathbf{C}$ there exists a formula $\mathbf{D}$, such that $\mathbf{B} \to^*_{\beta\eta} \mathbf{C}$ and $\mathbf{C} \to^*_{\beta\eta} \mathbf{D}$.

We say that $\beta\eta$-reduction is **terminating at A**, iff any sequence of $\beta\eta$-reductions starting with $\mathbf{A}$ terminates.

Lemma 1.3.10 *If $\beta\eta$-reduction is terminating at $\mathbf{C}$, and $\mathbf{B}$ is a subformula of $\mathbf{C}$, then $\beta\eta$-reduction is terminating at $\mathbf{B}$*

Proof: Any infinite reduction sequence from $\mathbf{B}$ can be transformed to one from $\mathbf{C}$ since $\beta\eta$-reduction is defined on subformulae. ☐

Definition 1.3.11 Let $\mathcal{SR}$ be the typed relation, such that $\mathcal{SR}_\alpha(\mathbf{A})$, iff $\beta\eta$-reduction is terminating and confluent at $\mathbf{A}_\alpha$. Furthermore let $\mathcal{LR}$ be the logical relation induced by $\mathcal{SR}$

The proof of termination and confluence will proceed in two steps: We will first show that $\mathcal{LR} \subseteq \mathcal{SR}$ and then that $\mathcal{LR}$ is admissible. Combining these results with the fundamental theorem for logical relations (1.3.8), we obtain that $\mathcal{LR}_\alpha(\mathbf{A})$ for any formula $\mathbf{A} \in wff_\alpha(\Sigma)$ and therefore $\mathcal{SR}_\alpha(\mathbf{A})$, which implies universal termination of $\beta\eta$-reduction.

Lemma 1.3.12 (Roller Coaster Lemma)

1. *Let h be a constant or variable of type $\overline{\delta^n} \to \alpha$ and $\mathcal{SR}_{\delta^i}(\mathbf{D}^i)$, then $\mathcal{LR}_\alpha(h\overline{\mathbf{D}^n})$.*

2. *$\mathcal{LR}_\alpha(\mathbf{A})$ implies $\mathcal{SR}_\alpha(\mathbf{A})$.*

Proof: We prove the assertions by a simultaneous induction over the type α. If α is a base type, then we obtain the fact that $\beta\eta$-reduction is confluent and terminating at $h\overline{\mathbf{D}^n}$, since the $\mathbf{D}^i$ are (since the arguments of h are independent, there can be no $\beta\eta$-redexes that are not in the $\mathbf{D}^i$), thus $\mathcal{LR}_\alpha(h\overline{\mathbf{D}^n})$ since $\alpha \in \mathcal{BT}$ ($\mathcal{SR} = \mathcal{LR}$ there). The second assertion is a trivial consequence of the definition of $\mathcal{LR}$.

For the inductive case of the first assertion let $\alpha = \beta \to \gamma$, and $\mathcal{LR}_\beta(\mathbf{B})$. By the second inductive hypothesis we have $\mathcal{SR}_\beta(\mathbf{B})$, by the first inductive hypothesis $\mathcal{LR}_\gamma(h\overline{\mathbf{B}^n}\mathbf{B})$. Thus $\mathcal{LR}_\alpha(h\overline{\mathbf{B}^n})$ by definition and we have proven the first assertion.

For the second assertion let $\mathcal{LR}_\alpha(\mathbf{A})$ and $X_\beta \notin \mathbf{Free}(\mathbf{A})$. With the first inductive hypothesis ($n = 0$) we have $\mathcal{LR}_\beta(X)$, and thus $\mathcal{LR}_\gamma(\mathbf{A}X)$ by definition. Now we see that $\beta\eta$-reduction is terminating at $\mathbf{A}$, since by the second inductive hypothesis we have $\mathcal{SR}_\gamma(\mathbf{A}X)$, in particular $\mathbf{A}X$ is terminating, so $\mathbf{A}$ is by 1.3.10. We obtain the confluence result from the observation that $\lambda X\textbf{.}\mathbf{C}X \to^*_{\beta\eta} \lambda X\textbf{.}\mathbf{E}$, whenever $\mathbf{C}X \to^*_{\beta\eta} \mathbf{E}$, which is easily verified by an induction over the length of the $\beta\eta$-reduction. So we have the following situation.

$$\begin{tikzcd}[column sep=small] & \mathbf{A} \arrow[dl] \arrow[dr] & \\ \mathbf{C} & & \mathbf{D} \end{tikzcd} \qquad \begin{tikzcd}[column sep=small] & \mathbf{A}X \arrow[dl] \arrow[dr] & \\ \mathbf{C}X \arrow[dr] & & \mathbf{D}X \arrow[dl] \\ & \mathbf{E} & \end{tikzcd} \qquad \begin{tikzcd}[column sep=small] & \mathbf{A} \arrow[dl] \arrow[dr] & \\ \mathbf{C} \arrow[dr] & & \mathbf{D} \arrow[dl] \\ & \lambda X\textbf{.}\mathbf{E} & \end{tikzcd}$$

From $\mathcal{SR}_\gamma(\mathbf{A}X)$, we know that there is a formula $\mathbf{E}$ that makes $\beta\eta$-reduction confluent at $\mathbf{A}X$, thus $\mathbf{C}$ and $\mathbf{D}$ can be reduced to $\lambda X\textbf{.}\mathbf{E}$. □

Lemma 1.3.13 $\mathcal{LR}$ *is admissible*

Proof: We have to show that $\mathcal{LR}_\alpha([\mathbf{B}/X_\beta]\mathbf{A})$ and $\mathcal{LR}_\beta(\mathbf{B})$ imply $\mathcal{LR}_\alpha((\lambda X_\beta\textbf{.}\mathbf{A})\mathbf{B})$. Now let $\mathcal{LR}_{\gamma^i}(\mathbf{C}^i)$, where $\alpha = \overline{\gamma} \to \delta$, such that $\delta \in \mathcal{BT}$. Then (by an iterated application of the definition of $\mathcal{LR}$) it is sufficient to show that $\mathcal{SR}_\delta((\lambda X\textbf{.}\mathbf{A})\mathbf{B}\overline{\mathbf{C}})$, since δ is a base type. We will now convince ourselves that $\beta\eta$-reduction is both terminating and confluent at $(\lambda X\textbf{.}\mathbf{A})\mathbf{B}\overline{\mathbf{C}}$.

For termination let us assume that $\mathcal{LR}_\alpha([\mathbf{B}/X]\mathbf{A})$ and $\mathcal{LR}_\beta(\mathbf{B})$, then by definition of $\mathcal{LR}$ we have $\mathcal{LR}_\delta([\mathbf{B}/X]\mathbf{A}\overline{\mathbf{C}})$ and thus $\mathcal{SR}_\delta([\mathbf{B}/X]\mathbf{A}\overline{\mathbf{C}})$. In particular $\mathcal{SR}_\alpha([\mathbf{B}/X]\mathbf{A})$ and $\mathcal{SR}_{\gamma^i}(\mathbf{C}^i)$ by 1.3.10. Furthermore we obtain $\mathcal{SR}_\beta(\mathbf{B})$ from $\mathcal{LR}_\beta(\mathbf{B})$ by 1.3.12. Finally, $\beta\eta$-reduction must be terminating at $\mathbf{A}$, since an infinite reduction from $\mathbf{A}$ would imply one from $[\mathbf{B}/X]\mathbf{A}$ by 1.3.4. Thus there cannot be an infinite sequence of reductions from $(\lambda X\textbf{.}\mathbf{A})\mathbf{B}\overline{\mathbf{C}}$ that only contracts redexes from $[\mathbf{B}/X]\mathbf{A}$ and the $\mathbf{C}^i$. Thus an infinite reduction sequence from $(\lambda X\textbf{.}\mathbf{A})\mathbf{B}\overline{\mathbf{C}}$ has the form

$$\begin{array}{rcl} (\lambda X\textbf{.}\mathbf{A})\mathbf{B}\overline{\mathbf{C}} & \to^*_\beta & (\lambda X\textbf{.}\mathbf{A}')\mathbf{B}'\overline{\mathbf{C}'} \\ & \to_\beta & [\mathbf{B}'/X]\mathbf{A}'\overline{\mathbf{C}'} \\ & \to^*_\beta & \ldots \end{array}$$

where $\mathbf{A} \to^*_\beta \mathbf{A}'$, $\mathbf{B} \to^*_\beta \mathbf{B}'$ and $\mathbf{C}^i \to^*_\beta \mathbf{C}'^i$ are finite. Thus $[\mathbf{B}/X]\mathbf{A} \to^*_\beta [\mathbf{B}'/X]\mathbf{A}'$ and in particular (in contradiction to our assumption), we have constructed an infinite reduction

$$[\mathbf{B}/X]\mathbf{A}\overline{\mathbf{C}} \to^*_\beta [\mathbf{B}'/X]\mathbf{A}'\overline{\mathbf{C}'} \to^*_\beta \ldots$$

Confluence at $(\lambda X\textbf{.}\mathbf{A})\mathbf{B}\overline{\mathbf{C}}$ is a consequence of permutability of head reduction (cf. 1.3.3). Let $(\lambda X\textbf{.}\mathbf{A})\mathbf{B}\overline{\mathbf{C}} \to^h [\mathbf{B}/X]\mathbf{A}\overline{\mathbf{C}}$ and furthermore

$(\lambda X.\mathbf{A})\mathbf{B}\overline{\mathbf{C}} \to_\beta \mathbf{C}$ and $(\lambda X.\mathbf{A})\mathbf{B}\overline{\mathbf{C}} \to_\beta \mathbf{D}$. We will assume that the latter two reductions are internal, since we are in a special case otherwise. By permutability, there we have $[\mathbf{B}/X]\mathbf{A}\overline{\mathbf{C}} \to^i_\beta \mathbf{C}'$ and $[\mathbf{B}/X]\mathbf{A}\overline{\mathbf{C}} \to^i_\beta \mathbf{C}'$, which can be joined to a formula $\mathbf{E}$ (we have assumed $\mathcal{SR}_\alpha([\mathbf{B}/X]\mathbf{A}\overline{\mathbf{B}})$).

normal form!β
decidable
head normal form
normal form!head

$$\begin{tikzcd}
(\lambda X.\mathbf{A})\mathbf{B}\overline{\mathbf{C}} \arrow[rr, "h"] \arrow[d, "i"] \arrow[dr, "i"] & & {[\mathbf{B}/X]\mathbf{A}\overline{\mathbf{C}}} \arrow[ddll, "i"] \arrow[dd, "i"] \\
\mathbf{C} \arrow[d, "h"'] & \mathbf{D} \arrow[dr, "h"'] & \\
\mathbf{C}' \arrow[dr] & & \mathbf{D}' \arrow[dl] \\
& \mathbf{E} &
\end{tikzcd}$$

□

Theorem 1.3.14 (βη-Normalization)
The β-, η-, and βη-reduction relations are terminating and confluent.

Proof: We prove the theorem by collecting the results obtained so far. By the previous Lemma 1.3.13, $\mathcal{LR}$ is admissible, and furthermore $\mathcal{LR}_\gamma(\mathbf{C})$ for all $\mathbf{C} \in \mathcal{V}_\gamma \cup \Sigma_\gamma$ by 1.3.12.1 ($n = 0$), so by the fundamental theorem for logical relations 1.3.8, we have $\mathcal{LR}_\alpha(\mathbf{A})$ for all $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma)$. However this yields $\mathcal{SR}_\alpha(\mathbf{A})$ by 1.3.12.2, which entails the assertion by definition, since $\mathbf{A}$ was arbitrary. □

Corollary 1.3.15 (Normal Forms)
For any formula $\mathbf{A}$ *there is a sequence of β-reductions* $\mathbf{A} \to^*_\beta \mathbf{B}$ *such that* $\mathbf{B}$ *is a β-**normal form**. Furthermore, for any derivation* $\mathbf{A}_\alpha \to^*_{\beta\eta} \mathbf{B}_\alpha$ *there is a derivation* $\mathbf{A}_\alpha \to^*_\beta \mathbf{C}_\alpha \to^*_\eta \mathbf{B}_\alpha$. *Thus we can compute βη-normal forms by first reducing to β-normal form and then further η-reducing to normal form.*

As a consequence, two formulae are βη-equal, iff their βη normal forms are alphabetical variants. Since βη-normal forms are effectively computable and α-equality is decidable (recursively go through the formulae keeping an renaming table for the bound variables), βη-equality is decidable.

Corollary 1.3.16 (Decidability of βη-Equality)
For any pair $\mathbf{A}, \mathbf{B}$ *of formula it is decidable whether* $\mathbf{A} =_{\beta\eta} \mathbf{B}$.

Remark 1.3.17 (Head Reduction Strategy)
Since β-reduction is confluent and terminating, the head reduction strategy (restricting β-reduction to the unique head redex) is complete for β-reduction to head normal form. This fact is convenient in some situations, where we want to fix a unique β-reduction sequence. For any formula $\mathbf{A}$ we call the formula $\mathbf{B}$ obtained with a maximal head reduction sequence from $\mathbf{A}$ the **head normal form** of $\mathbf{A}$. Of course the strategy of reducing a

$\eta[\cdot]$η-expanded form
$\mathbf{A}\!\downarrow$*long βη-normal form of* $\mathbf{B}$
expanded form!η
long head normal form
normal form!long η, βη
long βη-normal form

term to head normal form and then recursively head reducing the immediate subterms of the matrix yields a complete strategy for full β-reduction.

Definition 1.3.18 (Long $\beta\eta$-Normal Form)
Let $\mathbf{A} \equiv (\lambda\overline{X^n}.h\overline{\mathbf{E}^m})$ be a well-formed formula in head normal form such that the matrix $(h\overline{\mathbf{E}^m})$ is of type $\overline{\beta_k} \to \gamma$ and $\gamma \in \mathcal{BT}$, then the **η-expanded form** of $\mathbf{A}$ denoted by $\eta[\mathbf{A}]$ is defined to be

$$\lambda X^1 \ldots X^n Y^1_{\beta_1} \ldots Y^k_{\beta_k}.h\mathbf{E}^1 \ldots \mathbf{E}^m Y^1_{\beta_1} \ldots Y^k_{\beta_k}$$

where $Y^i_{\beta_i}$ are new variables of types β_i. We define the **long head normal form**, denoted by $\mathbf{A}\!\downarrow^h$, of a well-formed formula $\mathbf{A}$ to be the η-expanded form of the head normal form of $\mathbf{A}$. Similarly the **long $\beta\eta$-normal form** of $\mathbf{A}$, denoted by $\mathbf{A}\!\downarrow$, is the formula that we obtain by recursively extending this process to the arguments $\mathbf{E}^i$ and Y^j, thus

$$\mathbf{A}\!\downarrow = \lambda X^1 \ldots X^n Y^1_{\beta_1} \ldots Y^k_{\beta_k}.h\eta\left[\mathbf{E}^1\right] \ldots \eta[\mathbf{E}^m]\eta\left[Y^1_{\beta_1}\right] \ldots \eta\left[Y^k_{\beta_k}\right]$$

Theorem 1.3.19 (Head Normal Form Theorem)
For any formula $\mathbf{A}$ of type $\alpha = \overline{\beta_n} \to \gamma$ where $\gamma \in \mathcal{BT}$, the long head normal form of $\mathbf{A}$ has the form $\lambda\overline{X^n_{\beta_n}}.h\overline{\mathbf{U}}$, where h is a symbol and $h\overline{\mathbf{U}} \in wff_\gamma(\Sigma)$.

Remark 1.3.20 Since the η-expansion relation is a subrelation of the inverse of η-reduction, the η-normal forms of $\mathbf{A} \in wff(\Sigma)$ and $\eta[\mathbf{A}]$ are equal and therefore $\mathbf{A}$ and $\eta[\mathbf{A}]$ are η-equivalent.

We often use the η-expanded form rather than the η-reduced form of the β-normal form, because this normal form has better closure properties, e.g. if $\mathbf{A}$ is in β-reduced form, then $\eta[\mathbf{A}]$ is in β-reduced form as well, whereas the η-reduced form need not be. For a detailed discussion we refer to [Sny91].

1.4 Formal Semantics for $\Lambda^\to$

In this section we will formally develop in detail the set-theoretic semantics of the simply typed λ-calculus alluded to in section 1.2.2.

Intuitively, there is a universe $\mathcal{D}_\alpha$ for every type $\alpha \in \mathcal{T}$, such that $\mathcal{D}_{\alpha\to\beta}$ is the set of functions from $\mathcal{D}_\alpha$ to $\mathcal{D}_\beta$. Since this definition is too strong for the purposes of this book, where $\Lambda^\to$ is used as a theoretical basis for deduction in higher-order logic (it for instance entails extensionality and the existence of *all* functions), we will step by step specialize very general semantical structures (typed collections), which are hierarchies of sets indexed by types to the ones we will need later on (Σ-structures and Σ-algebras).

Definition 1.4.1 (Typed Collection)
A collection $\mathcal{D} := \mathcal{D}_\mathcal{T} := \{\mathcal{D}_\alpha \mid \alpha \in \mathcal{T}\}$ of sets $\mathcal{D}_\alpha$, indexed by the set $\mathcal{T}$

typed collection
type function
typed partial function
typed function/relation
constant
@application operator
pre-Σ-structure
carrier set
frame
universe
application operator
interpretation of constants
total
domain
functional
function application

of types, is called a **typed collection** (of sets). In the following we will always assume that $\mathcal{D}_\alpha \cap \mathcal{D}_\beta = \emptyset$, if $\alpha \neq \beta$. This allows us to define the **type function** $\tau\colon \bigcup_{\alpha\in\mathcal{T}} \mathcal{D}_\alpha \longrightarrow \mathcal{T}$ by $\tau(g) = \alpha$, iff $g \in \mathcal{D}_\alpha$. Let $\mathcal{D}_\mathcal{T}$ and $\mathcal{E}_\mathcal{T}$ be typed collections, then a collection $\mathcal{I} := \{\mathcal{I}^\alpha \in \mathcal{F}_p(\mathcal{D}_\alpha; \mathcal{E}_\alpha) \mid \alpha \in \mathcal{T}\}$ of partial functions is called a **typed partial function** $\mathcal{I}\colon \mathcal{D}_\mathcal{T} \longrightarrow \mathcal{E}_\mathcal{T}$.

It is often convenient to view a typed collection $\mathcal{D}_\mathcal{T}$ as the union $\bigcup_{\alpha\in\mathcal{T}} \mathcal{D}_\alpha$ and a typed function $\mathcal{I}\colon \mathcal{D}_\mathcal{T} \longrightarrow \mathcal{E}_\mathcal{T}$ as a function $\mathcal{I}\colon \bigcup_{\alpha\in\mathcal{T}} \mathcal{D}_\alpha \longrightarrow \bigcup_{\alpha\in\mathcal{T}} \mathcal{E}_\alpha$ with $\mathcal{I}^\alpha := \mathcal{I}|_{\mathcal{D}_\alpha}$ and $\Im(\mathcal{I}^\alpha) \subset \mathcal{E}_\alpha$. We take the liberty to switch the point of view whenever it is convenient. A collection $\{R_\alpha \subset \mathcal{D}_\alpha \times \mathcal{D}_\alpha \mid \alpha \in \mathcal{T}\}$ are called a **typed binary relation**.

Our treatment of $\Lambda^{\rightarrow}$ is parametric in the choice of **constant**s that are supplied. In particular, the algebraic structure of a pre-Σ-structure, which we are about to define, varies in the way the constants are interpreted.

1.4.1 Pre-Σ-Structures

The intuitive meaning for objects of functional type is a function, i.e. an object that can be applied to other objects of the appropriate type. Here we give a very abstract notion of an algebraic structure with function applications, which provides us with the basic vocabulary for the development of $\Lambda^{\rightarrow}$.

Definition 1.4.2 (Pre-Σ-Structure)
Let $\mathcal{D}_\mathcal{T}$ be a typed collection of sets,

$$@ := \{@^{\alpha\beta}\colon \mathcal{D}_{\alpha\rightarrow\beta} \times \mathcal{D}_\alpha \longrightarrow \mathcal{D}_\beta \mid \alpha, \beta \in \mathcal{T}\}$$

a typed family of partial functions, and let $\mathcal{I}\colon \Sigma \longrightarrow \mathcal{D}$ be a typed total function, then we call the triple $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ a partial **pre-Σ-structure**. The collection $\mathcal{D}$ is called the **carrier set** or the **frame** of $\mathcal{A}$, the set $\mathcal{D}_\alpha$ the **universe** of type α, the function @ the **application operator**, and the function $\mathcal{I}$ the **interpretation of constants**. A pre-Σ-structure is called **total**, iff @ is a collection of total functions. For an object $\mathsf{f} \in \mathcal{D}_{\alpha\rightarrow\beta}$ we define the **domain** of f as the set $\mathbf{Dom}(\mathsf{f}) := \{\mathsf{a} \in \mathcal{D}_\alpha \mid (\mathsf{f}, \mathsf{a}) \in \mathbf{Dom}(@)\}$.

We call a pre-Σ-structure $\mathcal{A} \equiv (\mathcal{D}, @, \mathcal{I})$ **functional**, iff the following statement holds for all $\mathsf{f}, \mathsf{g} \in \mathcal{D}_{\alpha\rightarrow\beta}$:

functional: $\mathsf{f} \equiv \mathsf{g}$, if for all $\mathsf{a} \in \mathcal{D}_\alpha$ $\mathsf{f}@\mathsf{a} \equiv \mathsf{g}@\mathsf{a}$.

Note that functionality is a restriction on the function universes only.

Remark 1.4.3 The application operator @ in a pre-Σ-structure is an abstract version of function application. As in the case with functions before

singleton pre-Σ-structure
structure!singleton
homomorphism
endomorphism
epimorphism
monomorphism
congruence
$[\![\mathsf{f}]\!]_\sim$ *equivalence class of* f *modulo* $\sim$
functionality
functional!congruence
congruence!functional
$\mathcal{D}^\sim$ *domain of equivalence classes modulo* $\sim$
$\mathcal{I}^\sim$ *interpretation of constants modulo* $\sim$
$@^\sim$ *application operator for equivalence classes modulo* $\sim$
$\pi_\sim$ *canonical projection*
canonical projection
$\mathcal{A}/_\sim$ *quotient pre-Σ-structure for the Σ-congruence* $\sim$

(cf. 1.2.13) it is no restriction to exclusively use a binary application operator, which corresponds to unary function application, since we can define higher-arity application operators from the binary one by setting

$$\mathsf{f}@(\mathsf{a}^1,\ldots,\mathsf{a}^n) := (\ldots(\mathsf{f}@\mathsf{a}^1)\ldots@\mathsf{a}^n)$$

Example 1.4.4 (Singleton Pre-Σ-Structure)
The triple $(\{\mathsf{a}\}\times\mathcal{T}, @^{\mathsf{a}}, \mathcal{I}^{\mathsf{a}})$, where $\mathsf{a}@^{\mathsf{a}}\mathsf{a} = \mathsf{a}$ and $\mathcal{I}^{\mathsf{a}}(c) = \mathsf{a}$ for all constants $c \in \Sigma$ is a (trivial) example for a total, functional pre-Σ-structures. It is called the **singleton pre-Σ-structure**.

Definition 1.4.5 (Σ-Homomorphism)
Let $\mathcal{A} := (\mathcal{D}, @^{\mathcal{A}}, \mathcal{I})$ and $\mathcal{B} := (\mathcal{E}, @^{\mathcal{B}}, \mathcal{J})$ be pre-Σ-structures. A **Σ-homomorphism** is a typed function $\kappa\colon \mathcal{D} \longrightarrow \mathcal{E}$ such that

1. $\kappa \circ \mathcal{I} \equiv \mathcal{J}$.

2. For all types $\alpha, \beta \in \mathcal{T}$, all $\mathsf{f} \in \mathcal{D}_{\alpha\to\beta}$, and $\mathsf{g} \in \mathcal{D}_\alpha$ we have: $\kappa(\mathsf{f})@^{\mathcal{B}}\kappa(\mathsf{g}) \equiv \kappa(\mathsf{f}@^{\mathcal{A}}\mathsf{g})$.

As usual we define a Σ-**endomorphism** κ on $\mathcal{A}$ to be a Σ-homomorphism $\kappa\colon \mathcal{A} \longrightarrow \mathcal{A}$, an Σ-**epimorphism** and a Σ-**monomorphism** to be surjective and injective Σ-homomorphisms respectively.

The most important method for constructing Σ-structures with given properties is well-known for algebraic structures and consists in building a suitable Σ-congruence and passing to the quotient structure. We will now develop the formal basis for it.

Definition 1.4.6 (Σ-Congruence)
Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ be a pre-Σ-structure, then a typed equivalence relation $\sim$ is called a Σ-**congruence** on $\mathcal{A}$, iff $\mathsf{f} \sim \mathsf{f}' \in \mathcal{D}_{\alpha\to\beta}$ and $\mathsf{g} \sim \mathsf{g}' \in \mathcal{D}_\alpha$ imply $\mathsf{f}@\mathsf{g} \sim \mathsf{f}'@\mathsf{g}'$.

Let $\mathsf{f} \in \mathcal{D}_\alpha$, then the equivalence class of f modulo $\sim$, $[\![\mathsf{f}]\!]_\sim$, is the set of all $\mathsf{g} \in \mathcal{D}_\alpha$, such that $\mathsf{f} \sim \mathsf{g}$.

A Σ-congruence is called **functional**, iff for all types α, β and all $\mathsf{f}, \mathsf{g} \in \mathcal{D}_{\alpha\to\beta}$ the fact that $\mathsf{f}@\mathsf{a} \sim \mathsf{g}@\mathsf{a}$ holds for all $\mathsf{a} \in \mathcal{D}_\beta$ implies $\mathsf{f} \sim \mathsf{g}$. Note that, since $\sim$ is a congruence, we also have the other direction, so we have $\mathsf{f}@\mathsf{a} \sim \mathsf{g}@\mathsf{a}$ for all $\mathsf{a} \in \mathcal{D}_\beta$, iff $\mathsf{f} \sim \mathsf{g}$.

Lemma 1.4.7 *The β and $\beta\eta$ equality relations $=_\beta$ and $=_{\beta\eta}$ are congruences on the pre-Σ-structures wff(Σ) and cwff(Σ).*

Proof: The congruence properties are a direct consequence of the fact that $\beta\eta$ reduction rules are defined to act on sub-term positions. ☐

quotient!pre-Σ-structure
pre-Σ-structure!quotient
canonical projection
projection!canonical
structure
variable assignment
φvariable assignment
evaluation function
ℰevaluation function
evaluation function
ℰ$_\varphi$evaluation function
evaluation function
assignment

Definition 1.4.8 (Quotient Pre-Σ-Structure)
Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ be a pre-Σ-structure, $\sim$ a Σ-congruence on $\mathcal{A}$, $\mathcal{D}^{\sim}_{\alpha} := \{[\![\mathsf{f}]\!]_{\sim} \mid \mathsf{f} \in \mathcal{D}_\alpha\}$, and $\mathcal{I}^{\sim}(c_\alpha) := [\![\mathcal{I}(c_\alpha)]\!]_{\sim}$ for all constants $c_\alpha \in \Sigma_\alpha$. Furthermore let $@^{\sim}$ be defined by $[\![\mathsf{f}]\!]_{\sim} @^{\sim} [\![\mathsf{a}]\!]_{\sim} := [\![\mathsf{f}@\mathsf{a}]\!]_{\sim}$. To see that this definition only depends on equivalence classes of $\sim$, consider $\mathsf{f}' \in [\![\mathsf{f}]\!]_{\sim}$ and $\mathsf{g}' \in [\![\mathsf{g}]\!]_{\sim}$, then $[\![\mathsf{f}@\mathsf{g}]\!]_{\sim} \equiv [\![\mathsf{f}'@\mathsf{g}]\!]_{\sim} \equiv [\![\mathsf{f}'@\mathsf{g}']\!]_{\sim} \equiv [\![\mathsf{f}@\mathsf{g}']\!]_{\sim}$. So $@^{\sim}$ is well-defined and total, thus $\mathcal{A}/_{\sim} := (\mathcal{D}^{\sim}, @^{\sim}, \mathcal{I}^{\sim})$ is also a pre-Σ-structure. We call $\mathcal{A}/_{\sim}$ the **quotient structure** of $\mathcal{A}$ for the relation $\sim$ and the typed function $\pi_\sim : \mathcal{A} \longrightarrow \mathcal{A}/_{\sim}$ that maps f to $[\![\mathsf{f}]\!]_{\sim}$ its **canonical projection**.

This definition is justified by the following theorem.

Theorem 1.4.9 *Let $\mathcal{A}$ be a pre-Σ-structure and let $\sim$ be an Σ-congruence on $\mathcal{A}$. The canonical projection $\pi_\sim$ is a surjective Σ-homomorphism. Furthermore, $\mathcal{A}/_{\sim}$ is functional iff $\sim$ is functional.*

Proof: Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ be a pre-Σ-structure. To convince ourselves that $\pi_\sim$ is indeed a surjective Σ-homomorphism, we note that by definition $\pi_\sim$ is surjective and $\mathcal{I}^{\sim} \equiv \pi_\sim \circ \mathcal{I}$. Now let $\mathsf{f} \in \mathcal{D}_{\beta\to\alpha}$, and $\mathsf{g} \in \mathbf{Dom}(\mathsf{f}) \subseteq \mathcal{D}_\beta$, then for all $\mathsf{g}' \in [\![\mathsf{g}]\!]_{\sim}$ we have $\mathsf{f}@\mathsf{g} = \mathsf{f}@\mathsf{g}'$ since $\sim$ is a congruence and therefore $\mathsf{g}' \in \mathbf{Dom}(\mathsf{f})$. Thus we have $[\![\mathsf{g}]\!]_{\sim} \equiv \pi_\sim(\mathsf{g}) \in \mathbf{Dom}([\![\mathsf{f}]\!]_{\sim}) \equiv \mathbf{Dom}(\pi_\sim(\mathsf{f}))$ and $\pi_\sim(\mathsf{f}) @^{\sim} \pi_\sim(\mathsf{g}) \equiv [\![\mathsf{f}]\!]_{\sim} @^{\sim} [\![\mathsf{g}]\!]_{\sim} \equiv [\![\mathsf{f}@\mathsf{g}]\!]_{\sim} \equiv \pi_\sim(\mathsf{f}@\mathsf{g})$.

The quotient construction trivializes $\sim$ to (meta-)equality, so functionality of $\sim$ is equivalent to functionality of $\mathcal{A}$. Formally we have $[\![\mathsf{f}]\!]_{\sim} \equiv [\![\mathsf{g}]\!]_{\sim}$, iff $\mathsf{f} \sim \mathsf{g}$, iff $f@\mathsf{a} \sim \mathsf{g}@\mathsf{a}$, iff $[\![\mathsf{f}@\mathsf{a}]\!]_{\sim} \equiv [\![\mathsf{g}@\mathsf{a}]\!]_{\sim}$, iff $[\![\mathsf{f}]\!]_{\sim} @^{\sim} [\![\mathsf{a}]\!]_{\sim} \equiv [\![\mathsf{g}]\!]_{\sim} @^{\sim} [\![\mathsf{a}]\!]_{\sim}$ for all $\mathsf{a} \in \mathcal{D}_\alpha$ and thus for all $[\![\mathsf{a}]\!]_{\sim} \in \mathcal{D}^{\sim}_{\alpha}$.

The assertion about totality is a consequence of the fact that $\pi_\sim$ is an epimorphism. □

A prominent example of a pre-Σ-structure is the collection of well-formed formulae.

Example 1.4.10 If we define $\mathbf{A}@\mathbf{B} := (\mathbf{A}\mathbf{B})$ for $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma)$ and $\mathbf{B} \in \mathit{wff}_\beta(\Sigma)$, then $@ : \mathit{wff}_{\alpha\to\beta}(\Sigma) \times \mathit{wff}_\alpha(\Sigma) \longrightarrow \mathit{wff}_\beta(\Sigma)$ is a total function. Thus $(\mathit{wff}(\Sigma), @, \mathrm{Id}_\Sigma)$ is a total pre-Σ-structure. The intuition behind this example is that we can think of the formula $\mathbf{A} \in \mathit{wff}_{\alpha\to\beta}(\Sigma)$ as a function

$$\mathbf{A} : \mathit{wff}_\alpha(\Sigma) \longrightarrow \mathit{wff}_\beta(\Sigma) \; ; \; \mathbf{B} \mapsto (\mathbf{A}\mathbf{B}) \; .$$

1.4.2 Σ-Structures

Σ-structures are pre-Σ-structures with a notion of evaluation for well-formed formulae in $\mathit{wff}(\Sigma)$.

Definition 1.4.11 (Assignment)
Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a pre-Σ-structure. A typed function $\varphi : \mathcal{V} \longrightarrow \mathcal{D}$ is called an **assignment** into $\mathcal{A}$.

evaluation function
structure!Σ
denotation

Definition 1.4.12 (Σ-Structure)
Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ be a pre-Σ-structure. We call a total typed mapping $\mathcal{E}\colon \mathcal{F}(\mathcal{V};\mathcal{D}) \times wff(\Sigma) \longrightarrow \mathcal{D}$ an **evaluation function** for $\mathcal{A}$, iff for any assignment φ into $\mathcal{A}$, we have

1. $\mathcal{E}_\varphi|_\Sigma \equiv \mathcal{I}$ and $\mathcal{E}_\varphi|_\mathcal{V} \equiv \varphi$
2. $\mathcal{E}_\varphi$ is a Σ-homomorphism
3. $\mathcal{E}_\varphi(\mathbf{A}) \equiv \mathcal{E}_\psi(\mathbf{A})$, whenever φ and ψ coincide on $\mathbf{Free}(\mathbf{A})$
4. $\mathcal{E}_\varphi(\lambda X.\mathbf{A})@\mathcal{E}_\varphi(\mathbf{B}) \equiv \mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\mathbf{B})/X]}(\mathbf{A}) \equiv \mathcal{E}_\varphi([\mathbf{B}/X]\mathbf{A})$

We call $\mathcal{A} := (\mathcal{D}, @, \mathcal{E})$ **Σ-structure**, iff $(\mathcal{D}, @, \mathcal{E}|_\Sigma)$ is a pre-Σ-structure and $\mathcal{E}$ is an evaluation function for $\mathcal{A}$. We call $\mathcal{E}_\varphi(\mathbf{A}_\alpha) \in \mathcal{D}_\alpha$ **denotation** of $\mathbf{A}_\alpha$ in $\mathcal{A}$ for φ.

If $\mathbf{A}$ is a closed formula, then $\mathcal{E}_\varphi(\mathbf{A})$ is independent of φ, since $\mathbf{Free}(\mathbf{A}) = \emptyset$. In these cases we sometimes drop the reference from $\mathcal{E}_\varphi(\mathbf{A})$ and simply write $\mathcal{E}(\mathbf{A})$.

Σ-structures generalize the notion of a Σ-algebras (Definition 1.4.29), which are the basis for Henkin models to the non-functional case. The existence of an evaluation function that meets the conditions above seems to be the weakest situation where one would like to speak of a model. Since Σ-structures need not be functional, we cannot use an evaluation function defined by homomorphic extension (see 1.4.25), as this requires functionality.

Example 1.4.13 The singleton pre-Σ-structure is a Σ-structure if we take $\mathcal{E}(\mathbf{A}) \equiv \mathsf{a}$, where a is the (unique) member of $\mathcal{D}_\alpha$. Note that in this Σ-structure $\mathcal{E}(\lambda X.X) \equiv \mathcal{E}_\varphi(\lambda X.Y)$.

For a detailed discussion on the closure conditions needed for the function universes to be rich enough, we refer the reader to [And72a, And73].

Note that the pre-Σ-structure $wff(\Sigma)$ from 1.4.10 cannot be made into a Σ-structure by providing an evaluation function, since there is no formula $\mathbf{C} \equiv \mathcal{E}_\varphi(\lambda X_\alpha.\mathbf{B}) \in wff_{\alpha\to\beta}(\Sigma)$ such that $\mathbf{C}@\mathbf{A} \equiv \mathbf{C}\mathbf{A} \equiv \mathcal{E}_{\varphi,[\mathbf{A}/X]}(\mathbf{B})$. In particular, the "obvious" choice $\lambda X_\alpha.\mathbf{B}$ for $\mathbf{C}$ does not work, since $(\lambda X_\alpha.\mathbf{B})\mathbf{A} \not\equiv \mathcal{E}_{\varphi,[\mathbf{A}/X]}(\mathbf{B})$. In fact, if $wff(\Sigma)$ were a Σ-structure, β-equality would have to be valid in $wff(\Sigma)$ (cf. 1.4.23), which it clearly is not.

General Assumption 1.4.14 For each $\alpha \in \mathcal{T}$ we assume the existence of a closed formula $\mathbf{G}^\alpha \in cwff_\alpha(\Sigma)$. This guarantees that types are not empty, i.e. that $\mathcal{D}_\alpha \neq \emptyset$ in any Σ-model $\mathcal{A} = (\mathcal{D}, @, \mathcal{E}, \upsilon)$, since $\mathcal{E}(\mathbf{G}^\alpha) \in \mathcal{D}_\alpha$ by definition of Σ-structures (1.4.12).

Automated deduction systems based on unification usually work with this implicit assumption, since otherwise the calculus becomes unsound: if the type α is empty, the formula $\forall X_\alpha.(pX) \wedge \neg(pX)$ is contradictory but at the same time satisfiable.

Note that it is sufficient to assume a closed formula $\mathbf{G}^\alpha$ for all $\alpha \in \mathcal{BT}$, since we can choose $\mathbf{G}^{\beta\to\alpha} := (\lambda X_\beta.\mathbf{G}^\alpha)$. Therefore each well-typed formula $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma)$ has a closed Σ-instance, that is, there exists a Σ-substitution $\sigma \in \mathbf{SUB}(\Sigma)$ such that $\sigma(\mathbf{A})$ is a closed formula. In this case the contradiction above really is unsatisfiable.

valid
satisfiable
quotient Σ-structure
structure!quotient
quotient!structure

Definition 1.4.15 (Valid Equation)
Let $\mathbf{A}, \mathbf{B} \in \mathit{wff}_\alpha(\Sigma)$ and $\mathcal{A} = (\mathcal{D}, @, \mathcal{E})$, then we call the equation $\mathbf{A} = \mathbf{B}$ **valid** in $\mathcal{A}$, iff $\mathcal{E}_\varphi(\mathbf{A}) = \mathcal{E}_\varphi(\mathbf{B})$ for all assignments φ into $\mathcal{A}$. We call $\mathbf{A} = \mathbf{B}$ **satisfiable** in $\mathcal{A}$, iff there is an assignment φ, such that $\mathcal{E}_\varphi(\mathbf{A}) = \mathcal{E}_\varphi(\mathbf{B})$.

We call $\mathbf{A} = \mathbf{B}$ valid (satisfiable) in a class $\mathcal{K}$ of Σ-structures, iff it is valid in every (satisfiable in some) $\mathcal{A} \in \mathcal{K}$.

The simplest way to make a given set of equations valid in a given structure is to pass to the quotient structure with respect to the induced Σ-congruence. Since we have already constructed the notion of quotient pre-Σ-structures, we only have to extend the notion of evaluation.

Definition 1.4.16 (Quotient Σ-Structure)
Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{E})$ be a Σ-structure, $\sim$ a Σ-congruence on $\mathcal{A}$ and let $\mathcal{A}/_\sim := (\mathcal{D}^\sim, @^\sim, \mathcal{E}^\sim)$ be the quotient pre-Σ-structure of $\mathcal{A}$, where $\mathcal{E}^\sim := \mathcal{E}|_\Sigma$.

For any assignment ψ into $\mathcal{A}/_\sim$, there exists an assignment φ into $\mathcal{A}$ such that $\psi \equiv \pi_\sim \circ \varphi$, since $\pi_\sim$ is a surjective Σ-homomorphism. So we can define $\mathcal{E}^\sim_\varphi$ as $\pi_\sim \circ \mathcal{E}_\psi$, and call $\mathcal{A}/_\sim := (\mathcal{D}^\sim, @^\sim, \mathcal{E}^\sim)$ the **quotient Σ-structure** of $\mathcal{A}$ modulo $\sim$.

Theorem 1.4.17 (Quotient Σ-Structure)
Let $\mathcal{A}$ be a Σ-structure and let $\sim$ be a Σ-congruence on $\mathcal{A}$, then $\mathcal{A}/_\sim$ is a Σ-structure, where every equation $\mathbf{A} \sim \mathbf{B}$ is valid.

Proof: We prove that $\mathcal{E}^\sim$ is a legal value function by verifying the conditions in 1.4.12. For this, let φ and ψ be assignments such that $\psi \equiv \pi_\sim \circ \varphi$. We have

1. $\mathcal{E}^\sim_\varphi|_\Sigma \equiv (\pi_\sim \circ \mathcal{E}_\psi)|_\Sigma \equiv \pi_\sim \circ \mathcal{E}_\psi|_\Sigma \equiv \pi_\sim \circ \mathcal{E} \equiv \mathcal{E}^\sim$ and
$\mathcal{E}^\sim_\varphi|_\mathcal{V} \equiv (\pi_\sim \circ \mathcal{E}_\psi)|_\mathcal{V} \equiv \pi_\sim \circ \mathcal{E}_\psi|_\mathcal{V} \equiv \pi_\sim \circ \psi \equiv \varphi$

2. $\mathcal{E}^\sim_\varphi \equiv \pi_\sim \circ \mathcal{E}_\psi$ is a Σ-homomorphism, since $\pi_\sim$ and $\mathcal{E}_\psi$ are.

3. $\mathcal{E}^\sim_\varphi(\mathbf{A}) \equiv [\![\mathcal{E}_\psi(\mathbf{A})]\!]_\sim \equiv [\![\mathcal{E}_{\psi'}(\mathbf{A})]\!]_\sim \equiv \mathcal{E}^\sim_{\varphi'}(\mathbf{A})$, iff φ and φ' coincide on $\mathbf{Free}(\mathbf{A})$, since this entails that ψ and ψ' do too.

4. $\mathcal{E}^\sim_\varphi([\mathbf{B}/X]\mathbf{A}) \equiv [\![\mathcal{E}_\psi([\mathbf{B}/X]\mathbf{A})]\!]_\sim \equiv [\![\mathcal{E}_{\psi,[\mathcal{E}_\psi(\mathbf{B})/X]}(\mathbf{A})]\!]_\sim$. With $[\![\mathcal{E}_\psi(\mathbf{B})]\!]_\sim \equiv \mathcal{E}^\sim_\varphi(\mathbf{B})$, we can see that this is $\mathcal{E}^\sim_{\varphi,[\mathcal{E}^\sim_\varphi(\mathbf{B})/X]}(\mathbf{A})$, and therefore $\pi_\sim \circ \psi, [\mathcal{E}_\psi(\mathbf{B})/X] \equiv \varphi, [\mathcal{E}^\sim_\varphi(\mathbf{B})/X]$ $\square$

term structure!β-
term structure!βη-
$\mathcal{TS}(\Sigma)^\beta$ β-term structure
$\mathcal{TS}(\Sigma)^{\beta\eta}$ βη-term structure
$\mathcal{TS}(\Sigma)$ term structure for Σ
term structure

Lemma 1.4.18 *$\beta\eta$-equality is a functional congruence on $\mathit{wff}(\Sigma)$. If Σ_α is infinite for all types $\alpha \in \mathcal{T}$, then $\beta\eta$-equality is also functional on $\mathit{cwff}(\Sigma)$.*

Proof: We will establish functionality of $=_{\beta\eta}$ on $\mathit{wff}(\Sigma)$ first and then use this to obtain the assertion for closed formulae.

Let $\mathbf{A}_{\gamma\to\alpha}\mathbf{C}_\gamma =_{\beta\eta} \mathbf{B}_{\gamma\to\alpha}\mathbf{C}$ for all $\mathbf{C}$, then in particular, for any variable $X \in \mathcal{V}_\gamma$ that is not free in $\mathbf{A}$ or $\mathbf{B}$, we have $\mathbf{A}X =_{\beta\eta} \mathbf{B}X$ and $\lambda X.\mathbf{A}X =_{\beta\eta} \lambda X.\mathbf{B}X$. By definition we have $\mathbf{A} =_\eta \lambda X_\alpha.\mathbf{A}X =_{\beta\eta} \lambda X_\alpha.\mathbf{B}X =_\eta \mathbf{B}$.

To show functionality of $\beta\eta$ on closed formulae, let $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_{\alpha\to\beta}(\Sigma)$, such that $\mathbf{A} \neq_{\beta\eta} \mathbf{B}$. Since $\beta\eta$ is functional on $\mathit{wff}(\Sigma)$, there must be a formula $\mathbf{C}$ with $\mathbf{AC} \neq_{\beta\eta} \mathbf{BC}$. Now let σ be a substitution that maps every variable in $\mathbf{Free}(\mathbf{C})$ to a distinct constant in Σ that does not occur in $\mathbf{C}$ (this has to exist, since we have assumed that all Σ_α are infinite and $\mathbf{Free}(\mathbf{C})$ is finite, so there are enough constants left). Then we have $\mathbf{AC}' \neq_{\beta\eta} \mathbf{BC}'$ as $\mathbf{C}$ and $\mathbf{C}'$ are structurally very similar and any $\beta\eta$ conversion sequence for $\mathbf{AC}' \neq_{\beta\eta} \mathbf{BC}'$ can be transformed into one for $\mathbf{AC} \neq_{\beta\eta} \mathbf{BC}$. Thus we have shown that $\mathbf{A} \neq_{\beta\eta} \mathbf{B}$ entails $\mathbf{AC}' \neq_{\beta\eta} \mathbf{BC}'$, which gives us the assertion. □

Remark 1.4.19 Note that the result for $\mathit{cwff}(\Sigma)$ is sharp. For instance, if $\Sigma = \{c_\iota\}$, then $\lambda X.X \neq_{\beta\eta} \lambda X.c$, but $(\lambda X.X)c =_{\beta\eta} c =_{\beta\eta} (\lambda X.c)c$, as $\{c\} = \mathit{cwff}_\iota(\Sigma)$ (it is a relatively simple exercise to extend this problem to more than one constant). The problem here is that we do not have a constant d_ι that would help distinguish the two functions. In $\mathit{wff}(\Sigma)$ we could always have used a variable.

Definition 1.4.20 (Term Structures for Σ)
Let $\mathit{cwff}(\Sigma)\!\downarrow_\beta$ be the collection of closed well-formed formulae in β-normal form and $\mathbf{A}@^\beta\mathbf{B}$ be the β-normal form of $\mathbf{AB}$. For the definition of an evaluation function let φ be an assignment into $\mathit{cwff}(\Sigma)\!\downarrow_\beta$. Note that $\sigma := \varphi|_{\mathbf{Free}(\mathbf{A})}$ is a substitution, since $\mathbf{Free}(\mathbf{A})$ is finite. Thus we can choose $\mathcal{E}^\beta_\varphi(\mathbf{A}) := \sigma(\mathbf{A})\!\downarrow_\beta$, where $\mathbf{A}\!\downarrow_\beta$ is the β-normal form of $\mathbf{A}$. We call $\mathcal{TS}(\Sigma)^\beta := (\mathit{cwff}(\Sigma)\!\downarrow_\beta, @^\beta, \mathcal{E}^\beta)$ the β-**term structure** for Σ.

Analogously, we can define $\mathcal{TS}(\Sigma)^{\beta\eta} := (\mathit{cwff}(\Sigma)\!\downarrow_{\beta\eta}, @^{\beta\eta}, \mathcal{E}^{\beta\eta})$ the $\beta\eta$-**term structure** for Σ.

The name *term structure* in the previous definition is justified by the following lemma.

Lemma 1.4.21 *$\mathcal{TS}(\Sigma)^\beta$ is a Σ-structure and $\mathcal{TS}(\Sigma)^{\beta\eta}$ is a functional Σ-structure.*

Proof: Note that $(\mathit{cwff}(\Sigma), @, \mathrm{Id}_\Sigma)$ is a pre-Σ-structure (see 1.4.10) and $\equiv_\beta$ is a Σ congruence (1.4.7). So $((\downarrow_\beta \mathit{cwff}(\Sigma)), @^\beta, \mathrm{Id}_\Sigma)$ is a pre-Σ-structure by theorem as constants are β-normal forms.

soundness!of β-equality ***complete***

As we have remarked in 1.4.13, *wff*(Σ) is not a Σ-structure, so we cannot use 1.4.17, but have to convince ourselves directly that $\mathcal{TS}(\Sigma)^\beta$ is a Σ-structure by verifying the conditions of 1.4.12. The first three are direct consequences of the definition of $\mathcal{E}^\beta$ as substitution application.

1. $\mathcal{E}^\beta_\varphi|_\Sigma \equiv \mathcal{E}^\beta \equiv \mathrm{Id}_\Sigma$ and $\mathcal{E}^\beta_\varphi|_\mathcal{V} \equiv \varphi$
2. $\mathcal{E}^\beta_\varphi$ is a Σ-homomorphism
3. $\mathcal{E}^\beta_\varphi(\mathbf{A}) \equiv \sigma(\mathbf{A}) \equiv \sigma'(\mathbf{A}) \equiv \mathcal{E}_{\varphi'}(\mathbf{A})$, iff φ and φ' coincide on $\mathbf{Free}(\mathbf{A})$
4. $\mathcal{E}^\beta_\varphi([\mathbf{B}/X]\mathbf{A}) \equiv \sigma([\mathbf{B}/X]\mathbf{A}) \equiv [\sigma(\mathbf{B})/X](\sigma'(\mathbf{A})) \equiv \sigma,[\sigma(\mathbf{B})/X]\mathbf{A} \equiv \mathcal{E}^\beta_{\varphi,[\mathcal{E}^\beta_\varphi(\mathbf{B})/X]}(\mathbf{A})$, where σ' is $\sigma,[X/X]$.

Since $=_{\beta\eta}$ is a sub-relation of $=_\beta$, a similar argument shows that $\mathcal{TS}(\Sigma)^{\beta\eta}$ is a Σ-structure. Furthermore, $=_{\beta\eta}$ is a functional Σ-congruence on *cwff*(Σ) (cf. 1.4.18), so we know by 1.4.2 that $\mathcal{TS}(\Sigma)^{\beta\eta}$ is functional. □

Remark 1.4.22 Note that $\mathcal{TS}(\Sigma)^\beta$ is not a functional Σ-structure since e.g. $(\lambda X_\gamma\mathbf{.}Y_{\gamma\rightarrow\delta}X)@^\beta\mathbf{C}_\gamma \equiv_\beta Y@^\beta\mathbf{C}$ for all $\mathbf{C}$ in $\mathcal{TS}_\gamma(\Sigma)^\beta$ but $\lambda X\mathbf{.}YX \not\equiv Y$.

In a general Σ-structure $\mathcal{A} := (\mathcal{D}, @, \mathcal{E})$, the constants are given a meaning by the interpretation function $\mathcal{I}\colon\Sigma \rightarrow \mathcal{D}$, and variables get their meaning by assignments $\varphi\colon\mathcal{V} \rightarrow \mathcal{D}$. Furthermore, the evaluation function has to respect instantiation as in first-order logic. This is enough to ensure soundness of β-equality. We do not have to show soundness of α-equality, since this is trivial as we have assumed alphabetic variants to be identical.

Lemma 1.4.23 (Soundness of β-equality)
Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{E})$ be a Σ-structure and φ an assignment into $\mathcal{A}$. Then $\mathcal{E}_\varphi((\lambda X\mathbf{.A})\mathbf{B}) \equiv \mathcal{E}_\varphi([\mathbf{B}/X]\mathbf{A})$.

Proof: By 1.4.12 $\mathcal{E}_\varphi$ is a Σ-homomorphism with $\mathcal{E}_\varphi(\lambda X\mathbf{.A})@\mathcal{E}_\varphi(\mathbf{B}) \equiv \mathcal{E}_\varphi([\mathbf{B}/X]\mathbf{A})$. □

On the other hand, we can see that $\alpha\beta$-equality ($\alpha\beta\eta$-equality) is **complete** for the class of (functional) Σ-structures, i.e. if the equation $\mathbf{A} = \mathbf{B}$ is valid, then $\mathbf{A}{=_{\beta\eta}}\mathbf{B}$. Thus $\alpha\beta$ equivalence fully characterizes equality in the class of all Σ-structures, while additional η-equality characterizes functionality.

Theorem 1.4.24 $\mathbf{A} = \mathbf{B}$ *is valid in the class of (functional) Σ-structures, iff* $\mathbf{A}{=_\beta}\mathbf{B}$ *(*$\mathbf{A}{=_{\beta\eta}}\mathbf{B}$*).*

Proof: The functional case is a simple consequence of the fact that $\mathcal{TS}(\Sigma)^{\beta\eta}$ is a total functional Σ-structure (cf. 1.4.21). The non-functional case is analogous using $\mathcal{TS}(\Sigma)^\beta$ instead. For closed equations the proof goes like

$\mathcal{I}_\varphi$ homomorphic extension of an assignment φ to wff(Σ)
homomorphic extension

this: if $\mathbf{A} = \mathbf{B}$ is valid in all Σ-structures, it must be in $\mathcal{TS}(\Sigma)$ and in particular $\mathbf{A}{\downarrow} = \mathcal{I}(\mathbf{A}) = \mathcal{I}(\mathbf{B}) = \mathbf{B}{\downarrow}$ and therefore $\mathbf{A} =_{\beta\eta} \mathbf{B}$.

EdNote(4)

If the equation is not closed, then the argument is a little more subtle due to the presence of free variables. We extend the signature Σ with constant c^i_α for each type α and each $i \in \mathbb{N}$[4]. Since we have assumed countably many variables per type, there is a bijection between the set of variables and the set of constants in Σ, which induces a variable assignment φ_Σ into $cwff(\Sigma)$ (each variable X_α is mapped to its associated constant $c^i_\alpha \in cwff_\alpha(\Sigma)$). Thus $\mathcal{I}_{\varphi_\Sigma}(\mathbf{A}) = \mathcal{I}_{\varphi_\Sigma}(\mathbf{B})$ is the long $\beta\eta$-normal form of $\varphi_\Sigma(\mathbf{A})$ and $\varphi_\Sigma(\mathbf{B})$. Since φ_Σ is a structure preserving homomorphism on well-formed formulae, $\varphi_\Sigma^{-1}(\mathcal{I}_{\varphi_\Sigma}(\mathbf{A}))$ is the is the long $\beta\eta$-normal form of both $\mathbf{A}$ and $\mathbf{B}$ and thus $\mathbf{A} =_{\beta\eta} \mathbf{B}$. □

1.4.3 Functional Σ-structures

For functional Σ-structures, there is another way to define evaluation: Since well-formed formulae are inductively built up from constants and variables φ and $\mathcal{I}$ can be extended to a Σ-homomorphism on well-formed formulae.

Definition 1.4.25 (Homomorphic Extension)
Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ be a functional pre-Σ-structure and let φ be an assignment into $\mathcal{A}$. Then the **homomorphic extension** $\mathcal{I}_\varphi$ of φ to $wff(\Sigma)$ is inductively defined to be a typed partial function $\mathcal{I}_\varphi \colon wff(\Sigma) \longrightarrow \mathcal{D}$ such that

1. $\mathcal{I}_\varphi(X) \equiv \varphi(X)$, if X is a variable,
2. $\mathcal{I}_\varphi(c) \equiv \mathcal{I}(c)$, if c is a constant,
3. $\mathcal{I}_\varphi(\mathbf{AB}) \equiv \mathcal{I}_\varphi(\mathbf{A})@\mathcal{I}_\varphi(\mathbf{B})$,
4. $\mathcal{I}_\varphi(\lambda X_\alpha.\mathbf{B}_\beta)$ is the function $\mathsf{f} \in \mathcal{D}_{\alpha\to\beta}$ such that $\mathsf{f}@\mathsf{a} := \mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{B})$ for all $\mathsf{a} \in \mathcal{D}_\alpha$. Note that f is unique, since we have assumed $\mathcal{A}$ to be functional.

We have to assume that the universes of functions $\mathcal{D}_{\alpha\to\beta}$ are rich enough to contain a value for all $\mathbf{A}_{\alpha\to\beta} \in wff_{\alpha\to\beta}(\Sigma)$ for this construction to yield a total function.

Lemma 1.4.26 *Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ be a functional pre-Σ-structure, then $\mathcal{E} \colon \varphi \mapsto \mathcal{I}_\varphi$ is an evaluation function for $\mathcal{A}$.*

Proof: To prove the assertion, we have to show the conditions of 1.4.12. The first one is trivially met by construction, the second is a direct consequence of the fact that $\mathcal{I}_\varphi \circ \mathrm{Id}_\Sigma \equiv \mathcal{I} \circ \mathrm{Id}_\Sigma \equiv \mathcal{I}$ on Σ.

[4] EdNote: assume sufficiently pure

evaluation!in functional structures

For the third condition, we prove that the value of a function depends only on its free variables (by induction on the structure of $\mathbf{A}$). The only interesting case is the one, where $\mathbf{A}$ is an abstraction, since the assertion is trivial for constants and variables, and a simple consequence of the inductive hypothesis for applications. So let $\mathbf{A} := (\lambda X.\mathbf{B})$, then $\mathcal{I}_\varphi(\mathbf{A})@\mathsf{a} \equiv \mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{B}) \equiv \mathcal{I}_{\psi,[\mathsf{a}/X]}(\mathbf{B}) \equiv \mathcal{I}_\psi(\mathbf{A})@\mathsf{a}$ by inductive hypothesis, since $\varphi, [\mathsf{a}/X]$ and $\psi, [\mathsf{a}/X]$ coincide on the free variables of $\mathbf{B}$. Thus we obtain the assertion from the definition of $\mathcal{I}_\varphi$.

Finally, we prove the fourth condition by induction on the structure of $\mathbf{A}$. If $\mathbf{A}$ is a constant or variable, then the assertion is trivial. The case where $\mathbf{A}$ is the application $\mathbf{CD}$ is entailed by the fact, that substitution and homomorphic extension are defined inductively on the structure of applications: We have

$$\begin{array}{rcl}\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{CD}) & \equiv & \mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{C})@\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{D}) \\ & \equiv & \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{C})@\mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{D}) \\ & \equiv & \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{CD})\end{array}$$

[5] If $\mathbf{A} \equiv (\lambda Y.\mathbf{D})$ and $\psi \equiv \varphi, [\mathsf{a}/Y]$, then EdNote(5)

$$\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A})@\mathsf{a} \equiv \mathcal{I}_\varphi(\lambda Y.[\mathbf{B}/X]\mathbf{D})@\mathsf{a} \equiv \mathcal{I}_\psi([\mathbf{B}/X]\mathbf{D}) \equiv \mathcal{I}_{\psi,[\mathcal{I}_\psi(\mathbf{B})/X]}(\mathbf{D})$$

by inductive hypothesis. Note that ψ and φ coincide on the free variables of $\mathbf{A}$ and $\mathbf{B}$, therefore by the third condition, which we have proven above, we have $\mathcal{I}_{\psi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{D}) \equiv \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\lambda Y.\mathbf{D})@\mathsf{a}$. This implies the assertion, since $\mathcal{A}$ is functional.[6] □ EdNote(6)

In fact, for functional Σ-structures, the two notions of evaluation coincide:

Lemma 1.4.27 (Evaluation in functional Σ-Structures)
If $\mathcal{A} := (\mathcal{D}, @, \mathcal{E})$ is a functional Σ-structure, then $\mathcal{E}_\varphi \equiv \mathcal{I}_\varphi$ for any assignment φ into $\mathcal{A}$.

Proof: Let $\mathbf{A} \in \mathit{wff}(\Sigma)$, we prove the assertion by induction over the depth[2] of $\mathbf{A}$. The assertion is trivial, if $\mathbf{A}$ is a constant or variable, and a simple consequence of the inductive hypothesis, if $\mathbf{A}$ is an application. So let $\mathbf{A} := (\lambda X.\mathbf{B})$, furthermore let Y be a variable not in $\mathbf{Free}(\mathbf{A})$ and $\psi := \varphi, [\mathsf{a}/Y]$. Then

$$\mathcal{E}_\varphi(\mathbf{A})@\mathsf{a} \equiv \mathcal{E}_\psi(\mathbf{A})@\mathsf{a} \equiv \mathcal{E}_\varphi(\mathbf{A})@\mathcal{E}_\psi(Y) \equiv \mathcal{E}_\psi(\mathbf{A}Y) \equiv \mathcal{E}_\psi([Y/X]\mathbf{B})$$

[5] EdNote: still to show the second equality!!!!

[6] EdNote: still to show the second equality!!!!

[2] The depth of a term is the depth (i.e. the length of the longest path) of the corresponding formula tree.

pre-Σ-algebra
full
structure!full
algebra!Σ

since β-equality is sound in Σ-structures. Now $[Y/X]\mathbf{B}$ is smaller than $\mathbf{A}$, so we can use the inductive hypothesis to obtain

$$\mathcal{E}_\varphi(\mathbf{A})@\mathsf{a} \equiv \mathcal{I}_\psi([Y/X]\mathbf{B}) \equiv \mathcal{I}_\psi(\mathbf{A}Y) \equiv \mathcal{I}_\varphi(\mathbf{A})@\mathcal{I}_\psi(Y) \equiv \mathcal{I}_\varphi(\mathbf{A})@\mathsf{a}$$

which entails the assertion since $\mathcal{A}$ is functional. □

Lemma 1.4.28 *Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{E})$ be a functional Σ-structure and X be a variable that is not free in $\mathbf{A}$, then $\mathcal{E}_\varphi(\lambda X.\mathbf{A}X) \equiv \mathcal{E}_\varphi(\mathbf{A})$ for all assignments φ into $\mathcal{A}$.*

Proof: With 1.4.12.3 and the fact that X is not free in $\mathbf{A}$ we have

$$\mathcal{E}_\varphi(\lambda X.\mathbf{A}X)@\mathsf{a} \equiv \mathcal{E}_{\varphi,[\mathsf{a}/X]}(\mathbf{A})@\mathcal{E}_{\varphi,[\mathsf{a}/X]}(X) \equiv \mathcal{E}_\varphi(\mathbf{A})@\mathsf{a}$$

which implies the assertion $\mathcal{E}_\varphi(\lambda X.\mathbf{A}X) \equiv \mathcal{E}_\varphi(\mathbf{A})$, as $\mathcal{A}$ is functional. □

We now specialize the notion of Σ-structures to the classical general model semantics for $\Lambda^{\rightarrow}$.

Definition 1.4.29 (Σ-Algebra)
A **pre-Σ-algebra** $\mathcal{A} := (\mathcal{D}, \mathcal{I})$ is a pre-Σ-structure $(\mathcal{D}, @, \mathcal{I})$ such that $\mathcal{D}_{\alpha\to\beta} \subseteq \mathcal{F}(\mathcal{D}_\alpha; \mathcal{D}_\beta)$ and $\mathsf{f}@\mathsf{a} \equiv \mathsf{f}(\mathsf{a})$. A pre-$\Sigma$-algebra is called **full**, iff $\mathcal{D}_{\alpha\to\beta} \equiv \mathcal{F}(\mathcal{D}_\alpha; \mathcal{D}_\beta)$. We call a pre-$\Sigma$-algebra an **Σ-algebra**, iff it is a Σ-structure.

Remark 1.4.30 Note that pre-Σ-algebras are functional, since they are defined as structures of mathematical functions. On the other hand, for any functional Σ-structure $\mathcal{A}$, we can define an isomorphic Σ-algebra $\mathcal{A}'$.

Proof: For a functional Σ-structure $\mathcal{A} := (\mathcal{D}, @, \mathcal{I})$ we define a Σ-algebra $\mathcal{A}' := (\mathcal{D}', \mathcal{I}')$ and a bijective Σ-homomorphism $\kappa\colon \mathcal{A} \longrightarrow \mathcal{A}'$ by an induction on the type:

- $\mathcal{D}'_\alpha := \mathcal{D}_\alpha$ for all $\alpha \in \mathcal{BT}$ and $\kappa := \mathrm{Id}_{\mathcal{D}}$; obviously κ is bijective.
- $\mathcal{D}'_{\alpha\to\beta} := \kappa(\mathcal{D}_{\alpha\to\beta})$ and $\kappa(\mathsf{f}) := \kappa \circ (@\mathsf{f}) \circ \kappa^{-1}$ for $\mathsf{f} \in \mathcal{D}_{\alpha\to\beta}$, where $@\mathsf{f}$ is the function in $\mathcal{D}_{\alpha\to\beta}$, such that $\mathsf{a} \mapsto \mathsf{f}@\mathsf{a}$. Note that with this construction κ is a homomorphism, since

  $$\kappa(\mathsf{f})(\kappa(\mathsf{a})) \equiv \kappa(\mathsf{f}@(\kappa^{-1}(\kappa(\mathsf{a})))) \equiv \kappa(\mathsf{f}@\mathsf{a})$$

  κ is surjective by construction and injective, since $\mathcal{A}$ is functional: If $\mathsf{f} \not\equiv \mathsf{g} \in \mathcal{D}_{\alpha\to\beta}$, then there is an $\mathsf{a} \in \mathcal{D}_\alpha$, such that $\mathsf{f}(\mathsf{a}) \not\equiv \mathsf{g}(\mathsf{a})$, in particular, we have

  $$\kappa(\mathsf{f}(\mathsf{a})) \equiv \kappa(\mathsf{f})@\kappa(\mathsf{a}) \not\equiv \kappa(\mathsf{f})@\kappa(\mathsf{a}) \equiv \kappa(\mathsf{g}(\mathsf{a}))$$

  since κ is injective on $\mathcal{D}_\beta$. Thus and therefore $\kappa(\mathsf{f}) \not\equiv \kappa(\mathsf{g})$, since $\kappa(\mathsf{f}), \kappa(\mathsf{g}) \in \mathcal{F}(\mathcal{D}_\alpha; \mathcal{D}_\beta)$

Now, we only have to choose $\mathcal{I}' := \kappa \circ \mathcal{I}$ to complete the construction of $\mathcal{A}'$. □

raw
judgment
inference rule
antecedent
succedent
axiom
proper inference rules
inference rule!proper
inference system

As a consequence, we can always consider functional Σ-structures as Σ-algebras.

1.5 Type Inference for $\Lambda^{\rightarrow}$

We now recast the definitions of well-formed formulae and λ-conversion as type inference systems. The notion of type inference system helps the analysis of λ-calculi such as $\Sigma\mathcal{HOL}$ (see section 7), where the type (or in that case the sort) does not only depend on the structure of the formula.

Type inference systems define judgments: in the case of the simply typed λ-calculus the primary judgment is that a formula $\mathbf{A}$ is of type α ($\Gamma \vdash_{\Sigma} \mathbf{A}\colon\alpha$). It is defined with respect to a variable context Γ, which gives local type information for the variables, and the signature Σ, which contains type information for constants. In the type-inference view, a formula $\mathbf{A}$ is called well-typed (with respect to Σ and Γ), iff the judgment $\Gamma \vdash_{\Sigma} \mathbf{A}\colon\alpha$ is derivable in an inference system $\Lambda^{\rightarrow}$.

Before we go into the details of the type inference system, let us generally fix the notations for inference systems and calculi.

Since we will use several inference systems and calculi in this book we will make the intuitive usage precise and give a (set-theoretic) definition.

Definition 1.5.1 (Inference System)
Each logical system is built up from syntactic objects like types, formulae, variables. These objects are called **raw** types, formulae,.... Let $\mathcal{J}$ be a relation on these entities, then we call $\mathcal{J}$ a **judgment schema** and a tuple $(\mathcal{E}^1,\ldots,\mathcal{E}^n) \in \mathcal{J}$ a **judgment**. An **inference rule** is an effectively computable relation

$$\mathcal{R} = \{(\mathcal{C}^1,\ldots,\mathcal{C}^m,\mathcal{D}) \mid \mathcal{C}^i \in \mathcal{J}^i, \mathcal{D} \in \mathcal{K}\}$$

on judgments such that $\mathcal{J}^i$ and $\mathcal{K}$ are judgment schemata. Inference rules are traditionally represented by a set of schemata of the form

$$\frac{\mathcal{C}^1,\ldots,\mathcal{C}^m}{\mathcal{D}}\,\mathcal{R}$$

where the part $\mathcal{C}^1,\ldots,\mathcal{C}^m$ is called **antecedent** and $\mathcal{D}$ is called the **succedent**. In order to give a finite presentation of an inference system the schemata may be schematic in the syntactic objects. Inference rules with empty antecedent are called **axiom**s and otherwise **proper inference rules**. An **inference system** $\mathcal{I}$ is a finite set of inference rules.

$\Phi \vdash_{\mathcal{I}} \mathbf{A}$ *derivation of* $\mathbf{A}$ *from* Φ *in system* $\mathcal{I}$
$\Phi \vdash_{\mathcal{R}} \mathcal{J}$ *one-step derivation with inference rule* $\mathcal{R}$
assertion
justification
support
derivation
proof-tree
derivation
hypothesis
structural ordering
ordering!structural
admissible
derivable
$[X\colon\alpha]$ *variable declaration*
variable declaration
declaration!variable
context
$\vdash_{ctx} \Gamma\Gamma$ *is a variable context*
$[X^1,\ldots,X^n\colon\alpha]$ *abbreviation for* $[X^1\colon\alpha],\ldots,[X^n\colon\alpha]$ *in variable contexts*

Definition 1.5.2 ($\mathcal{I}$-Derivation)
Let $\mathcal{I}$ be an inference system and $\mathcal{D}$ be a finite tree, where each node $\mathcal{N}$ in $\mathcal{D}$ is labeled with a triple $(\mathcal{J}, R, \{\mathcal{J}^1,\ldots,\mathcal{J}^n\})$ such that $R \in \mathcal{I}$ and $(\mathcal{J}^1,\ldots,\mathcal{J}^n,\mathcal{J}) \in R$. $\mathcal{J}$ is called the **assertion**, R the **justification**, and the set of $\mathcal{J}^i$ the **support** of $\mathcal{N}$. $\mathcal{D}$ is called an $\mathcal{I}$-**derivation**, iff each node $\mathcal{N}$ with label $(\mathcal{J}, R, \{\mathcal{J}^1,\ldots,\mathcal{J}^n\})$ has n children $\mathcal{N}^i$ with assertions $\mathcal{J}^i$. Note that, since the leaves of $\mathcal{I}$-derivations have no children, they have to be labeled with axioms. Because of the tree nature, we often call an $\mathcal{I}$-derivation a **proof-tree**.

Let $\mathcal{J}$ be a judgment and Φ be a set of judgments. We call an $\mathcal{I}$-derivation $\mathcal{D}$ an $\mathcal{I}$-**derivation** of $\mathcal{J}$ from the set Φ of **hypotheses**, if $\mathcal{J}$ is the assertion of the root of $\mathcal{D}$ and the supports of the leaves of $\mathcal{D}$ are subsets of Φ. If there exists an $\mathcal{I}$-derivation $\mathcal{D}$ of $\mathcal{J}$ from Φ, then we write $\Phi \vdash_{\mathcal{I}} \mathcal{J}$ or $\mathcal{D}\colon \Phi \vdash_{\mathcal{I}} \mathcal{J}$, if we want to specify the derivation. If $\mathcal{D}$ only consists of a single node labeled with the inference rule $\mathcal{R}$, then we often write $\Phi \vdash_{\mathcal{R}} \mathcal{J}$.

We will frequently prove our theorems by induction on the structure of the derivations involved in the assertion. For this we will use the **structural ordering** on derivations, which we can define by $\mathcal{D} \prec \mathcal{E}$ iff $\mathcal{D}$ is a subtree of $\mathcal{E}$.

Definition 1.5.3 (Admissible, Derivable)
Let $\mathcal{I}$ be an inference system, then an inference rule $\mathcal{R}$ is called **admissible** in $\mathcal{I}$, iff adding $\mathcal{R}$ to $\mathcal{I}$ does not change the set of judgments derivable from a given set of hypotheses. An inference rule $\mathcal{R}$ is called **derivable** in $\mathcal{I}$, iff for each $\mathcal{J}^1,\ldots,\mathcal{J}^n \vdash_{\mathcal{R}} \mathcal{J} \in \mathcal{R}$ there already exists an $\mathcal{I}$-derivation $\mathcal{J}^1,\ldots,\mathcal{J}^n \vdash_{\mathcal{I}} \mathcal{J}$.

Thus admissible rules can be added to an inference system without changing its theoretical properties. For practical reasoning applications, these added rules can sometimes make life much easier. Clearly, derivable rules of inference are also admissible.

Definition 1.5.4 (Variable Context)
Let X_α be a variable and α a type, then we call a pair $[X\colon\alpha]$ a **variable declaration** for X. We call a typed, partial function from variables to types a (variable) **context**. Thus a variable context can be expressed as a set of variable declarations.

Notation 1.5.5 We write the fact that a typed partial function $\Gamma\colon \mathcal{V} \longrightarrow \mathcal{T}$ is a variable context as $\vdash_{ctx} \Gamma$, and we generally use the symbols Γ, Δ, and Ξ for variable contexts. With our convention from 1.2.12 we have $\Gamma(X) = \alpha$ for $\Gamma := \Gamma', [X\colon\alpha]$, even if $\Gamma'(X) = \beta$.

In contrast to Definition 1.1.6 that directly defines the sets $\mathit{wff}_\alpha(\Sigma)$ of well-formed formulae based on the structure of raw formulae, the the type-inference view defines validity of judgments inductively on the structure of

raw *formula!raw* ***typing proofs*** ***well-typed***

the derivation of $\Gamma \vdash_{\Sigma} \mathbf{A} \colon \alpha$. In the simply typed λ-calculus these notions of inductions coincide, since the type inference system is structural, but in the case of $\Sigma\Lambda^{\rightarrow}$ (see section 7) they do not and the notion of an inference system is necessary to define well-typedness.

We now give an inference system for the judgments introduced so far. The judgment $\Gamma \vdash_{\Sigma} \mathbf{A} \colon \alpha$ holds, iff $\mathbf{A} \in \mathit{wff}_{\alpha}(\Sigma; \Gamma)$. Since the type inference system defines the well-formed formulae of $\Lambda^{\rightarrow}$ (those that have a type α), we also call it $\Lambda^{\rightarrow}$. We will maintain this practice in the following and identify the names of the logical systems with those of the inference systems defining them.

Definition 1.5.6 (Type Inference System $\Lambda^{\rightarrow}$)
The syntactic category of **raw** formulae consists of constants, variables, applications, and abstractions given by the context-free grammar below.

$$\begin{array}{rcl} \alpha \in \mathcal{T} & ::= & \iota \mid \ldots \mid \alpha \rightarrow \beta \\ \mathbf{A} & ::= & c \mid X \mid \mathbf{AB} \mid \lambda X_{\alpha} \mathbf{A} \end{array}$$

The inference system $\Lambda^{\rightarrow}$ for the judgment schema of well-typedness $\Gamma \vdash_{\Sigma} \mathbf{A} \colon \alpha$ is given by the schemata in Figure 1.8. We sometimes call $\Lambda^{\rightarrow}$-

$$\frac{c \in \Sigma_{\alpha}}{\Gamma \vdash_{\Sigma} c \colon \alpha}\ \mathit{wff}{:}\mathit{const} \qquad \frac{\Gamma(X) = \alpha}{\Gamma \vdash_{\Sigma} X \colon \alpha}\ \mathit{wff}{:}\mathit{var}$$

$$\frac{\Gamma \vdash_{\Sigma} \mathbf{A} \colon \beta \rightarrow \alpha \quad \Gamma \vdash_{\Sigma} \mathbf{B} \colon \beta}{\Gamma \vdash_{\Sigma} \mathbf{AB} \colon \alpha}\ \mathit{wff}{:}\mathit{app}$$

$$\frac{\Gamma, [X \colon \beta] \vdash_{\Sigma} \mathbf{A} \colon \alpha}{\Gamma \vdash_{\Sigma} (\lambda X_{\beta} . \mathbf{A}) \colon \beta \rightarrow \alpha}\ \mathit{wff}{:}\mathit{abs}$$

Figure 1.8: Inference Rules for Typing

derivations **typing proofs**, since they prove the typing judgment of the root node. It is obvious that this inference system for the well-typedness judgment is correct and complete with respect to the definition 1.1.6. Thus we can alternatively use this type inference system as a definition of well-typed formulae by specifying that any raw formula $\mathbf{A}$ is called **well-typed**, iff the judgment $\Sigma \vdash \mathbf{A} \colon \alpha$ is derivable in $\Lambda^{\rightarrow}$. Note that just like definition 1.1.6 this is an inductive definition. However, in contrast to the old definition this formally is inductive on the structure of $\Lambda^{\rightarrow}$-derivations rather than on the structure of formulae.

Remark 1.5.7 By inspection of the type inference system $\Lambda^{\rightarrow}$ above we see that the formulae in the succedent of the inference rules are partitioned by

structural
congruence relation
relation!congruence
term relation
closure!transitive reflexive
multi-step
relation!multi-step
length
ln($\mathcal{D}$)*length of the multi-step derivation* $\mathcal{D}$

the four possibilities for the structure of formulae (variable, constant, application, and abstraction), which is a disjoint partition of formulae. Thus the root node of any proof of a judgment $\Sigma \vdash \mathbf{A}\colon\alpha$ is uniquely determined by the structure of $\mathbf{A}$, and consequently, there is a tree-isomorphism between any typing proof for $\Sigma \vdash \mathbf{A}\colon\alpha$ and the formula $\mathbf{A}$ itself (when viewed as a tree). We call inference systems where this is the case **structural**. Note that structural type inference systems can be very conveniently inverted into type inference algorithms that recursively analyze the structure of formulae. For an example of a non-structural type inference system see $\Sigma\Lambda^{\rightarrow}$ in definition 7.2.7.

$$\frac{}{\Sigma \vdash (\lambda X_\beta.\mathbf{A})\mathbf{B} \rightarrow^t_\beta [\mathbf{B}/X]\mathbf{A}}\,\mathit{wf}{:}\beta{:}\mathit{top} \qquad \frac{X \notin \mathbf{Free}(\mathbf{A})}{\Sigma \vdash (\lambda X_\alpha.\mathbf{A}X) \rightarrow^t_\eta \mathbf{A}}\,\mathit{wf}{:}\eta{:}\mathit{top}$$

Figure 1.9: One-Step Top-Level Reductions

Now we use inference systems to analyze subsystems of $\beta\eta$-conversion.

Definition 1.5.8 (One-Step, Top-Level $\beta\eta$-Reductions)
$\beta\eta$-reduction can be formalized with a judgment $\Sigma \vdash \mathbf{A} \rightarrow_{\beta\eta} \mathbf{B}$, which is given by the inference rules for top-level reduction in Figure 1.9. Here we use the judgment $Y \in \mathbf{Free}(\mathbf{A})$, which we have defined in 1.1.10.

These one-step, top-level $\beta\eta$-reduction relations can be augmented to **congruence relation** in a very general manner by the rules in Figures 1.10 and 1.12.

Definition 1.5.9 (Congruence Closure for Single-Step Reduction)

Let $\mathcal{R}(\mathbf{A}, \mathbf{B})$ be a relation on well-formed formulae given by an inference system (such as $\mathbf{A} \rightarrow_{\beta\eta} \mathbf{B}$), then we obtain the **term relation** by adding the third set of inference rules in Figure 1.10 allow application of the top-level rules at arbitrary positions. In particular, there is a one-to-one correspondence of the position p, where the rule is applied and the *tr*:*app*, *tr*:*abs* derivation.

Definition 1.5.10 (Multi-Step Extension)
The first set of inference rules (transitivity and reflexivity) give us a **multi-step** relation, For any derivation $\mathcal{D}$ in a multi-step inference system, we define the **length** of $\mathcal{D}$ (written as $\mathbf{ln}(\mathcal{D})$) to be the number of *ms*:*trans* nodes in $\mathcal{D}$.

To highlight the fact that a $\beta\eta$-reduction sequence has more than one step we will often write $\Sigma \vdash \mathbf{A} \rightarrow^*_{\beta\eta} \mathbf{B}$.

equivalence relation
relation!equivalence

$$\frac{\Sigma \vdash \mathbf{A} \rightarrow^{t}_{\beta\eta} \mathbf{B}}{\Sigma \vdash \mathbf{A} \rightarrow^{1}_{\beta\eta} \mathbf{B}}\, tr{:}start \qquad \frac{\Sigma \vdash \mathbf{A} \rightarrow^{1}_{\beta\eta} \mathbf{B}}{\Sigma \vdash \lambda X_\alpha.\mathbf{A} \rightarrow^{1}_{\beta\eta} \lambda X_\alpha.\mathbf{B}}\, tr{:}abs$$

$$\frac{\Sigma \vdash \mathbf{A} \rightarrow^{1}_{\beta\eta} \mathbf{B}}{\Sigma \vdash \mathbf{AC} \rightarrow^{1}_{\beta\eta} \mathbf{BC}}\, tr{:}app{:}fn \qquad \frac{\Sigma \vdash \mathbf{C} \rightarrow^{1}_{\beta\eta} \mathbf{D}}{\Sigma \vdash \mathbf{AC} \rightarrow^{1}_{\beta\eta} \mathbf{BD}}\, tr{:}app{:}arg$$

Figure 1.10: Inference Rules for Single-Step Reduction

$$\frac{\Sigma \vdash \mathbf{A} \rightarrow^{1}_{\beta\eta} \mathbf{B} \quad \Sigma \vdash \mathbf{B} \rightarrow_{\beta\eta} \mathbf{C}}{\Sigma \vdash \mathbf{A} \rightarrow_{\beta\eta} \mathbf{C}}\, ms{:}trans \qquad \frac{\Sigma \vdash \mathbf{A}{:}\, \alpha}{\Sigma \vdash \mathbf{A} \rightarrow_{\beta\eta} \mathbf{A}}\, ms{:}ref$$

Figure 1.11: Inference Rules for Multi-step Reduction

Definition 1.5.11 (Equivalence Relation)
The *eq:sym* rule allows to turn any multi-step relation in to **equivalence relation**.

$$\frac{\Sigma \vdash \mathbf{A} \rightarrow_{\beta\eta} \mathbf{B}}{\Sigma \vdash \mathbf{A} =_{\beta\eta} \mathbf{B}}\, eq{:}start \qquad \frac{\Sigma \vdash \mathbf{A} =_{\beta\eta} \mathbf{B}}{\Sigma \vdash \mathbf{B} =_{\beta\eta} \mathbf{A}}\, eq{:}sym$$

Figure 1.12: Inference Rules for $\beta\eta$-Equality

Chapter 2

Higher-Order Logic

In this chapter we will present a formulation $\mathcal{HOL}$ of higher-order logic by giving the simply typed λ-calculus $\Lambda^{\rightarrow}$ developed so far, a logical interpretation. For this we specialize the type system and assume the existence of certain logical constants with a fixed interpretation. $\mathcal{HOL}$ is essentially the Andrews/Henkin version [Hen50, And71, And72b] of simple type theory. Since we have treated the syntax part in the last chapter, we will mainly be concerned with model theory in this.

Since the model theory of higher-order logic is inherently complex (see Figure 2.1), let us first informally discuss the issues treated formally in the rest of this chapter. We will discuss the model classes from bottom to top, from the most specific notion of standard models ($\mathfrak{ST}$)to the most general notion of υ-complexes, motivating the respective generalizations as we go along. In section 2.1, we will proceed the other way around, specializing the notion of a Σ-model ($\mathfrak{M}$) more and more.

The symbols in the boxes in Figure 2.1 denote model classes, the symbols labeling the arrows indicate the properties inducing the corresponding specialization, and the ∇-symbols next to the boxes indicate the clauses in the definition of abstract consistency classes (cf. 2.2.5) that are needed to establish a model existence theorem for this class of models.

A model existence theorem for a logical system $\mathcal{L}$ is a theorem of the form: *If a set of sentences Φ in $\mathcal{L}$ is a member of an abstract consistency class Γ, then there exists a $\mathcal{L}$-model for Φ.* It derives its importance from the fact that it allows to simplify completeness proofs with respect to $\mathcal{L}$: To prove completeness of a particular calculus $\mathcal{C}$, we only have to prove that the class Γ of sets of sentences Φ that are $\mathcal{C}$-consistent (cannot be refuted in $\mathcal{C}$) is an abstract consistency class. Then the model existence theorem tells us that $\mathcal{C}$-consistent sets of sentences are satisfiable in $\mathcal{L}$. Now we assume that a sentence $\mathbf{A}$ is valid in $\mathcal{L}$, so $\neg\mathbf{A}$ does not have a $\mathcal{L}$-model and is therefore $\mathcal{C}$-inconsistent. From this it is easy to verify that $\mathbf{A}$ is a theorem of $\mathcal{C}$. Note that with this argumentation the completeness proof for $\mathcal{C}$ condenses

standard model
model!standard
Henkin model

to verifying that Γ is an abstract consistency class, a task that does not refer to $\mathcal{L}$-models. Thus the usefulness of model existence theorems derives from the fact that it replaces the model-theoretic analysis in completeness proofs with the verification of some proof-theoretic conditions (membership in Γ). In this respect a model existence theorem is similar to a Herbrand Theorem, but it is easier to generalize to other logic systems like higher-order logic. The technique was developed for first-order logic by J. Hintikka and R. Smullyan [Hin55, Smu63, Smu68].

A **standard model** ($\mathfrak{ST}$, cf. Definition 2.1.8) for $\mathcal{HOL}$ provides a fixed set $\mathcal{D}_\iota$ of individuals, and a set $\mathcal{D}_o := \{\mathtt{T}, \mathtt{F}\}$ of truth values. All the domains for the complex types are defined inductively: $\mathcal{D}_{\alpha\to\beta}$ is the set of functions $f\colon \mathcal{D}_\alpha \to \mathcal{D}_\beta$. The evaluation function $\mathcal{I}_\varphi$ with respect to an interpretation $\mathcal{I}\colon \Sigma \to \mathcal{D}$ of constants and an assignment φ of variables is obtained by the standard homomorphic construction that evaluates a λ-abstraction with a function, whose operational semantics is specified by β-reduction.

One can reconstruct the key idea behind **Henkin models** ($\mathfrak{H}$, cf. Definition 2.1.8) by the following observation. If the set $\mathcal{D}_\iota$ is infinite, the set $\mathcal{D}_{\iota\to o}$ of sets of individuals must be uncountably infinite. On the other hand, any semantics that admits sound and complete calculi must have countable models, because of the compactness theorem that comes with a complete calculus. Leon Henkin generalized the class of admissible domains for functional types [Hen50]. Instead of requiring $\mathcal{D}_{\alpha\to\beta}$ to be the full set of functions, it is sufficient to require that $\mathcal{D}_{\alpha\to\beta}$ has enough members that any well-formed formula can be evaluated. In other words: the functional universes are rich enough to satisfy the comprehension axioms. Note that with this generalized notion of a model, there are fewer formulae that are valid in all models (intuitively, for any given formulae there are more possibilities for counter-models). In the particular case, the generalization to Henkin models, restricts the set of valid formulae sufficiently, so that all of them can be proven by a Hilbert-style calculus [Hen50].

It is matter of folklore that primitive equality (expressed by a primitive equality constant $= \in \Sigma$) is not strictly needed, since it can be expressed by the Leibniz formula. However, the Leibniz formula only really denotes the semantic equality relation if $\mathcal{D}_{\alpha\to o}$ contains enough properties to discern members of α; in fact, we need that for all $a \in \mathcal{D}_\alpha$, the singleton set $\{a\}$ is in $\mathcal{D}_{\alpha\to o}$ (cf. the proof of Lemma 2.1.15). On a similar note, Peter Andrews remarked in [And72a] that if the set $\mathcal{D}_{\alpha\to\alpha\to o}$ is so sparse that the semantic identity relation is not present, then it is possible to construct a Henkin model where Leibniz equality is non-extensional.

We are in the somewhat paradoxical situation, that Leibniz Equality (which is commonly used as a substitute for primitive equality) will only denote semantical equality, if we can guarantee that the identity relation is already present in the model (we call this property $\mathfrak{q}$, cf. Definition 2.1.4). Hence we introduce corresponding semantical structures, namely Henkin models without property $\mathfrak{q}$ ($\mathfrak{M}_{\beta\eta\mathfrak{b}}$), in which property $\mathfrak{q}$ is not necessarily valid and thus Leibniz equality does not necessarily denote the equality relation. An example of a theorem which is valid within the class of Henkin models but is not in the class of $\mathfrak{M}_{\beta\eta\mathfrak{b}}$'s is given by the axiom of functional extensionality for Leibniz equality $(\forall F_{\alpha\to\beta}.\forall G_{\alpha\to\beta}.(\forall X_\beta.FX \doteq GX) \Rightarrow F \doteq^\beta G)$, see lemma 2.1.16.

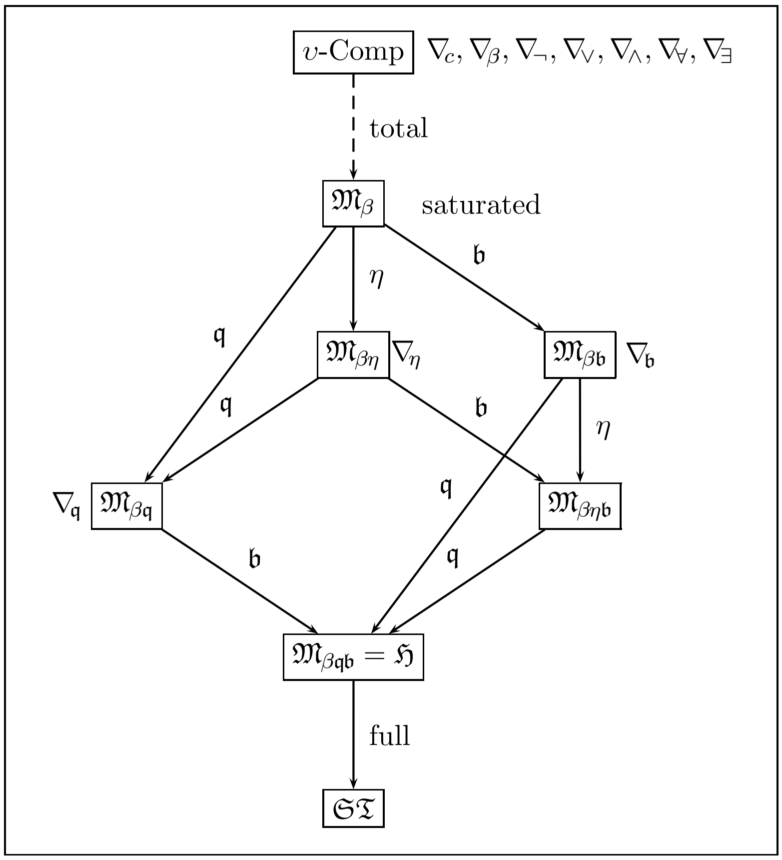


Figure 2.1: The landscape of Higher-Order Semantics

The next generalization of model classes comes from the fact that we want to characterize the deductive power of higher-order theorem provers mentioned above on a semantic level (we will take TPS [ABI+96a] as an example). Note that TPS cannot be complete with respect to Henkin models and is even not generally complete for $\mathfrak{M}_{\beta\eta\mathfrak{b}}$'s, although there is some 'extensionality treatment' build into the proof procedure. The incompleteness of TPS for Henkin models[1] can be seen from the fact that it fails to

[1] In case the extensionality axioms are not available in the search space. Note that one can add extensionality axioms to the calculus in order to achieve, at least in theory, Henkin completeness. But this increases the search space drastically and is not feasible in

functional Σ*-models*
model!functional Σ
interpretation!function
$\mathcal{I}$*interpretation function*
models!functional
property!$\mathfrak{b}$
$\mathfrak{b}$*property for models*
property!$\mathfrak{q}$
$\mathfrak{b}$*property for models*
model!Σ

refute formulae such as $c\mathbf{A}_o \wedge \neg c(\neg\neg\mathbf{A})$, where c is a constant of type $o \to o$ or $c(\lambda X_\alpha.\mathbf{B}X \wedge \mathbf{A}X) \Rightarrow c(\lambda X_\alpha.\mathbf{A}X \wedge \mathbf{B}X)$, where c is a constant of type $(\alpha \to o) \to o$. The problem in the former example is that the higher-order unification algorithm employed by TPS cannot determine that $\mathbf{A}$ and $\neg\neg\mathbf{A}$ denote identical semantic objects (by Boolean extensionality as already mentioned before), and thus returns failure instead of success. In the second example the principle of functional extensionality is needed in addition in order to prove the theorem.

The lack of completeness of refutation procedures like TPS occurs especially in these situations, where $\mathcal{HOL}$-formulae contain occurrences of propositional formulae dominated by uninterpreted constants or variables or where this problem is mixed with the problem of functional extensionality; in our examples the function constant c dominates the proposition $\mathbf{A}_o$ or the set expression $\lambda X_\alpha.\mathbf{B}X \wedge \mathbf{A}X$. To give a semantical characterisation of the deductive power of the TPS procedure, we have to generalize the class of Henkin models further, so that there are counter-models to the examples above. Obviously, this involves weakening the assumption that $\mathcal{D}_o \equiv \{\mathtt{T}, \mathtt{F}\}$ (we call this assumption for Henkin models property $\mathfrak{b}$), since this entails that the values of $\mathbf{A}$ and $\neg\neg\mathbf{A}$ are identical: In **functional** Σ**-models** ($\mathfrak{M}_{\beta\eta}, \mathfrak{M}_{\beta\eta\mathfrak{b}}, \mathfrak{M}_{\beta\mathfrak{q}}$, cf. Definitions 2.1.5 and 2.1.1) we only insist that there is a valuation υ of $\mathcal{D}_o$, i.e., a function $\upsilon\colon \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ that is coordinated with the functions $\mathcal{I}(\neg)$, $\mathcal{I}(\wedge)$, $\mathcal{I}(\Pi^\alpha)$ and (possibly) $\mathcal{I}(=^\alpha)$, where $\mathcal{I}$ is the interpretation function (i.e., a family of interpretation functions) for the constants given in the signature. Thus we have a notion of validity for Σ: we call a proposition $\mathbf{A}$ valid in $\mathcal{M} := (\mathcal{D}, \mathcal{I}, \upsilon)$ under an assignment φ, iff $\upsilon(\mathcal{I}_\varphi(\mathbf{A})) \equiv \mathtt{T}$. In our first example, there is a Σ-model structure $\mathcal{M} \equiv (\mathcal{D}, \mathcal{I}, \upsilon)$, where $\mathcal{I}_\varphi(\mathbf{A}) \not\equiv \mathcal{I}_\varphi(\neg\neg\mathbf{A})$, and therefore $\mathcal{I}_\varphi(c\mathbf{A}) \not\equiv \mathcal{I}_\varphi(c(\neg\neg\mathbf{A}))$ if we take $\mathcal{I}(c)$ to be the identity function on $\mathcal{D}_\alpha$. In particular, we can have $\upsilon(\mathcal{I}_\varphi(c\mathbf{A})) \not\equiv \upsilon(\mathcal{I}_\varphi(c(\neg\neg\mathbf{A})))$, and therefore $\upsilon(\mathcal{I}_\varphi(c\mathbf{A}_o \wedge \neg c(\neg\neg\mathbf{A}))) \equiv \mathtt{F}$, since υ is a valuation.

Clearly, for functional Σ-models we have the same choices concerning the role of equality, therefore, we distinguish the classes $\mathfrak{M}_{\beta\eta}$ and $\mathfrak{M}_{\beta\mathfrak{q}}$ of functional Σ-models without/with property $\mathfrak{q}$. Furthermore, we have the class $\mathfrak{M}_{\beta\eta\mathfrak{b}}$ of functional Σ-models with (only) property $\mathfrak{b}$, which expresses that the set of truth values contains exactly two elements, i.e., $\mathcal{D}_o \equiv \{\mathtt{T}, \mathtt{F}\}$. Since functional Σ-models with properties $\mathfrak{b}$ and $\mathfrak{q}$ are defined to be Σ-Henkin models, we can also view $\mathfrak{M}_{\beta\eta\mathfrak{b}}$ as "Henkin models without property $\mathfrak{q}$".

Finally, we drop the requirement of functional extensionality for Σ-**models**, which is encoded as property η (cf. Definition 2.1.1). This is the most general notion of semantics we will discuss in this book; we only insist that the evaluation function is a homomorphism which respects instantiation. In such models, a function is not uniquely determined by its behavior

practice.

$\mathfrak{M}_\beta$ model class
$\mathfrak{M}_{\beta\mathfrak{b}}$ model class
models!non-functional
semi-valuation
Hintikka set

on all possible arguments. For the construction of such models we therefore need labelings for functions (e.g., a green and a red version of a function f) in order to differentiate between them, even though they are functionally equivalent. As already done for functional Σ-models we analyze properties $\mathfrak{q}$ and $\mathfrak{b}$ for non-functional Σ-Models. Whereas $\mathfrak{b}$ may or may not hold for non-functional Σ-Models, it turns out that property $\mathfrak{q}$, i.e., the requirement that the intended equality relations are provided in the respective domains, only makes sense in connection with functionality. More precisely, property $\mathfrak{q}$ is not implied from functionality.

Peter Andrews has pioneered the construction of non-functional models with his υ-complexes in [And71]. These are even more general constructions than our Σ-models, since totality of the evaluation function is not assumed. His construction is based on Schütte's semi-valuation method [Sch60], which only needs partial valuations to construct a model for a given Hintikka set.

In this chapter, we concentrate on the other aspects of higher-order models and ensure totality of our evaluation functions by a saturation condition (cf. Definition 2.2.10) in our abstract consistency classes. This does not restrict the applicability of our model existence theorems, since saturation is relatively simple to prove for a given calculus (see [Koh98, BK97b]). For all of the notions of models (except naturally for standard models, where such a theorem cannot hold), we present model existence theorems tying the differentiating conditions of the models to suitable conditions in the abstract consistency classes (see section 2.2.4). We can use the classical construction in all cases: abstract consistent sets are extended to Hintikka sets (see section 2.2.2), which induce a valuation on a term structure (see Definition 1.4.20). In some cases, we have to take a quotient structure (see Definition 1.4.16) to ensure that the set of truth values is exactly $\{\mathtt{T}, \mathtt{F}\}$ for property $\mathfrak{b}$.

The simplest way to ensure property $\mathfrak{q}$ is by assuming that the signature contains a primitive logical constant for equality, which is evaluated as semantical identity (we call this property $\mathfrak{e}$). We will study this case in Section 2.2.3. On the one hand, the semantical situation becomes simpler now (cf. Figure 2.2), since $\mathfrak{M}_\beta$, $\mathfrak{M}_{\beta\eta}$ and $\mathfrak{M}_{\beta\mathfrak{q}}$ are identified, just as $\mathfrak{M}_{\beta\mathfrak{b}}$, $\mathfrak{M}_{\beta\eta\mathfrak{b}}$ and $\mathfrak{H}$, on the other hand, the existence of another logical constant induces further conditions in the definition of the abstract consistency classes.

2.1 Σ-Models

The standard example and intuitive semantic notion for higher-order logic is that of a Henkin model. This we can define from Σ-algebras by insisting on a suitable semantics for the type o of truth values. Since we also want to have a semantics where the extensionality axioms can fail to be valid, we generalize the notion of a Henkin model to the notion of a Σ-model.

Σ-model
valuation
Σ-valuation
satisfies
valid
υvaluation
$\models_\varphi$*semantical satisfiability*
$\models$*semantical validity*
valuation
model!Σ
satisfies
valid
truth
falsity
$\mathbf{T}_o$*truth, defined concept*
$\mathbf{F}_o$*falsity, defined concept*
property!q
property!η
property!b
q*property for models*
η*property for models*
b*property for models*

For this we use the notion of a Σ-valuation, which intuitively gives a truth-value interpretation to the domain $\mathcal{D}_o$ of a Σ-structure, which is consistent with the intuitive interpretations of the logical constants. Since models are semantic entities that are constructed first of all to make a statement about the truth or falsity of a formula, the requirement that there exists a Σ-valuation is perhaps the most general condition under which one wants to speak of a model. Thus we will define our most general notion of semantics as Σ-structures that have Σ-valuations.

Definition 2.1.1 (Σ-Model)
Let $\mathcal{A} := (\mathcal{D}, @, \mathcal{E})$ be a Σ-structure, then a surjective total function $\upsilon\colon \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ such that

1. $\upsilon(\mathcal{E}(\neg)@\mathsf{a}) \equiv \mathtt{T}$, iff $\upsilon(a) \equiv \mathtt{F}$,
2. $\upsilon(\mathcal{E}(\vee)@\mathsf{a}@\mathsf{b}) \equiv \mathtt{T}$, iff $\upsilon(\mathsf{a}) \equiv \mathtt{T}$ or $\upsilon(\mathsf{b}) \equiv \mathtt{T}$,
3. $\upsilon(\mathcal{E}(\Pi^\alpha)@\mathsf{f}) \equiv \mathtt{T}$, iff $\upsilon(\mathsf{f}@\mathsf{a}) \equiv \mathtt{T}$ for each $\mathsf{a} \in \mathcal{D}_\alpha$

is called a Σ-**valuation** for $\mathcal{A}$ and $\mathcal{M} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$ is called a Σ-**model** The class of all Σ-models is denoted by $\mathfrak{M}_\beta$.

We say that an assignment φ **satisfies** a formula $\mathbf{A} \in \mathit{wff}_o(\Sigma)$ in $\mathcal{M}$ ($\mathcal{M} \models_\varphi \mathbf{A}$), iff $\upsilon(\mathcal{E}_\varphi(\mathbf{A})) \equiv \mathtt{T}$ and that $\mathbf{A}$ is **valid** in $\mathcal{M}$, iff $\mathcal{M} \models_\varphi \mathbf{A}$ for all assignments φ. Finally, we say that $\mathcal{M}$ is a Σ-**model** for a set $H \subseteq \mathit{wff}_o(\Sigma)$ ($\mathcal{M} \models H$) iff $\mathcal{M}$ satisfies all $\mathbf{A} \in H$.

Lemma 2.1.2 (Truth and Falsity in Σ-Models)
Let $\mathcal{M} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$ be a Σ-model and φ an assignment. Furthermore let $\mathbf{T}_o := \mathbf{A}_o \vee \neg(\mathbf{A}_o)$ for some $\mathbf{A}_o \in \mathit{wff}_o$ and let $\mathbf{F}_o := \neg\mathbf{T}_o$. Then $\upsilon(\mathcal{E}_\varphi(\mathbf{T}_o)) \equiv \mathtt{T}$ and $\upsilon(\mathcal{E}_\varphi(\mathbf{F}_o)) \equiv \mathtt{F}$.

Proof: We have $\upsilon(\mathcal{E}_\varphi(\mathbf{T}_o)) \equiv \mathtt{T}$ iff $\upsilon(\mathcal{E}_\varphi(\mathbf{A}_o \vee \neg(\mathbf{A}_o))) \equiv \mathtt{T}$. Evaluation shows that this statement is equivalent to $\upsilon(\mathcal{E}_\varphi(\mathbf{A})) \equiv \mathtt{T}$ or $\upsilon(\mathcal{E}_\varphi(\mathbf{A})) \equiv \mathtt{F}$, which is valid since $\varphi : \mathcal{V}_o \to \mathcal{D}_o$ and $\upsilon : \mathcal{D}_o \to \{\mathtt{T}, \mathtt{F}\}$ are total functions.
Note further that $\upsilon(\mathcal{E}_\varphi(\mathbf{F}_o)) \equiv \mathtt{F}$ evaluates to $\upsilon(\mathcal{E}_\varphi(\mathbf{T}_o)) \equiv \mathtt{T}$, which we already know. □

Remark 2.1.3 We only constrain the functional behavior of the values of the logical constants. In particular this does not fully specify these values, since

- $\mathcal{M}$ need not be functional,
- and there can be more than two truth values.

Definition 2.1.4 (Properties $\mathfrak{q}$, $\mathfrak{b}$, and η) *model class!$\mathfrak{M}_{\beta\mathfrak{q}}$ model class!$\mathfrak{M}_{\beta\mathfrak{b}}$ model class!$\mathfrak{M}_{\beta\eta}$ model class!$\mathfrak{M}_{\beta\eta\mathfrak{b}}$ model class!$\mathfrak{M}_{\beta\mathfrak{q}\mathfrak{b}}$ $\mathfrak{q}$property for models property!$\mathfrak{q}$*

Given a Σ-model $\mathcal{M} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$, we say that $\mathcal{M}$ has *property*

η iff $\mathcal{M}$ is functional,

$\mathfrak{q}$ iff $\mathcal{M}$ has property η and for all $\alpha \in \mathcal{T}$ there is a function $\mathrm{q}^\alpha \in \mathcal{D}_{\alpha\to\alpha\to o}$, such that for all $\mathsf{a}, \mathsf{b} \in \mathcal{D}_\alpha$ we have $\upsilon(\mathrm{q}^\alpha @\mathsf{a}@\mathsf{b}) \equiv \mathtt{T}$ iff $\mathsf{a} \equiv \mathsf{b}$.

$\mathfrak{b}$ iff $\mathcal{D}_o$ has at most two elements. Note that $\mathcal{D}_o$ must always have at least the two elements $\mathcal{E}_\varphi(\mathbf{T}_o)$ and $\mathcal{E}_\varphi(\mathbf{F}_o)$ by Lemma 2.1.2, so we can assume without loss of generality that $\mathcal{D}_o \equiv \{\mathcal{E}_\varphi(\mathbf{F}_o) \equiv \mathtt{F}, \mathcal{E}_\varphi(\mathbf{T}_o) \equiv \mathtt{T}\}$ and that υ is the identity function.

Definition 2.1.5 (Specialized Model Classes)

We define special classes of Σ-models depending on the validity of the properties η, $\mathfrak{q}$ and $\mathfrak{b}$. Thus we obtain the specialized classes of Σ-models $\mathfrak{M}_{\beta\mathfrak{q}}, \mathfrak{M}_{\beta\mathfrak{b}}, \mathfrak{M}_{\beta\eta}, \mathfrak{M}_{\beta\eta\mathfrak{b}}$ by requiring that the properties specified in the index are valid.

Remark 2.1.6 (Property $\mathfrak{q}$)

The idea of property $\mathfrak{q}$ is to ensure for all types α that the intuitive equality relation $\mathrm{q}_{\alpha\to\alpha\to o}$, i.e., a functional congruence relation, is contained in $\mathcal{D}_{\alpha\to\alpha\to o}$. This ensures the existence of unit sets in the domains $\mathcal{D}_{\alpha\to o}$ which in turn makes Leibniz equality the intended equality relation, as the membership in this unit sets can be used as a strong argument in order to distinguish between different elements of $\mathcal{D}_\alpha$. For a detailed discussion see [And72a].

Property $\mathfrak{q}$ as stated in [BK97a] is not correct. Whereas the motivation for the formulation there was the same as sketched above, this formulation does unfortunately not ensure functionality (although this was intended), which is needed in the proof of Lemma 2.1.12[7]. We want to thank an EdNote(7) unknown referee of the Journal of Symbol Logic for pointing to this problem.

Remark 2.1.7 (Property $\mathfrak{q}$ in Henkin Models)

As Peter Andrews has noted in [And72a], Leon Henkin unintendedly introduced $\mathfrak{M}_{\beta\eta\mathfrak{b}}$ in [Hen50] instead of the class of Henkin models in the sense below. An element of $\mathfrak{M}_{\beta\eta\mathfrak{b}}$ does not necessarily have property $\mathfrak{q}$ and as Andrews has shown in [And72a], a consequence is that such an element may lack the principle of functional extensionality $\mathrm{EXT}_L^{\alpha\to\beta}$, which he corrected by introducing property $\mathfrak{q}$.

[7] EdNote: wrong reference

Henkin model
model!Henkin
standard model
model!standard
quotient!model
model!quotient
$\mathcal{A}/_{\sim}$ quotient model
congruence
quotient
model!quotient
extensionality!for Leibniz equality
extensionality!functional, for Leibniz equality
extensionality!Boolean, for Leibniz equality
extensionality!full, for Leibniz equality
$\mathrm{EXT}_L^{\alpha\to\beta}$ *functional extensionality for Leibniz equality*
EXT_L^{o} *Boolean extensionality for Leibniz equality*
axiom!extensionality!Leibniz equality

Definition 2.1.8 (Σ-Henkin Models)
A functional Σ-model is called a Σ-**Henkin model**, iff it has properties $\mathfrak{q}$ and $\mathfrak{b}$. The class of all Σ-Henkin models is denoted by $\mathfrak{H}$ or $\mathfrak{M}_{\beta\mathfrak{q}\mathfrak{b}}$. If furthermore, all domains $D_{\alpha\to\beta}$ are full then we call $\mathcal{H}$ a Σ-**standard model** ($\mathfrak{S}\mathfrak{T}$).

Now let us extend the notion of a quotient structure to Σ-models.

Definition 2.1.9 (Quotient Σ-model)
Let $\mathcal{M} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$ be a Σ-model, $\sim$ a congruence on the corresponding Σ-structure $\mathcal{A} := (\mathcal{D}, @, \mathcal{E})$, and $\mathcal{A}/_{\sim}$ be the quotient Σ-structure of $\mathcal{A} := (\mathcal{D}, @, \mathcal{E})$ modulo $\sim$ as defined in 1.4.16.

If $\upsilon(\mathbf{A}) \equiv \upsilon(\mathbf{B})$ for all $\mathbf{A}, \mathbf{B} \in \mathit{wff}_o(\Sigma)$ with $\mathbf{A} \sim \mathbf{B}$, then $\sim$ is called a **congruence** for $\mathcal{M}$. Then $\mathcal{M}/_{\sim} := (\mathcal{D}^{\sim}, @^{\sim}, \mathcal{E}^{\sim}, \upsilon^{\sim})$ is called the **quotient** Σ-model of $\mathcal{M}$ modulo $\sim$, if $\upsilon^{\sim}([\![\mathsf{a}]\!]_{\sim}) \equiv \upsilon(\mathsf{a})$ for all $\mathsf{a} \in \mathcal{D}_o$.

Remark 2.1.10 Note the importance of the additional requirement for functional congruence relations stated in 2.1.9. Without this requirement the quotient Σ-models are not well-defined.

Lemma 2.1.11 *Let $\mathcal{M}$ be a Σ-model, $H \subseteq \mathit{wff}_o(\Sigma)$, and $\sim$ be a congruence for $\mathcal{M}$, then $\mathcal{M}/_{\sim} \models_{\varphi} H$ iff $\mathcal{M} \models_{\varphi} H$.*

Proof: Let $\mathbf{A}_o \in H$. We have $\upsilon^{\sim}(\mathcal{E}^{\sim}_{\varphi}(\mathbf{A}_o)) \equiv \upsilon^{\sim}([\![\mathcal{E}_{\varphi}(\mathbf{A}_o)]\!]_{\sim}) \equiv \upsilon(\mathcal{E}_{\varphi}(\mathbf{A}_o))$. □

2.1.1 Leibniz Equality

Definition 2.1.12 (Extensionality for Leibniz equality)
We call the following formula schemata

$$\begin{array}{lcl}\mathrm{EXT}_L^{\alpha\to\beta} & := & \forall F_{\alpha\to\beta}\boldsymbol{.}\forall G_{\alpha\to\beta}\boldsymbol{.}(\forall X_{\beta}\boldsymbol{.}FX \doteq GX) \Rightarrow F \doteq^{\beta} G \\ \mathrm{EXT}_L^{o} & := & \forall A_o\boldsymbol{.}\forall B_o\boldsymbol{.}(A \Leftrightarrow B) \Leftrightarrow A \doteq^{o} B\end{array}$$

the **axioms of full extensionality for Leibniz equality**; we refer to the first as **axiom of functional extensionality** and to the latter formula as the **extensionality axiom for truth values**. Note that $\mathrm{EXT}_L^{\alpha\to\beta}$ specifies functionality of the relation denoted by the Leibniz formula $\doteq$. We will use the terms functionality and extensionality interchangeably.

Remark 2.1.13 (Extensionality)
We distinguish the notions of functionality and extensionality for truth values, that are often referred to simply as "extensionality". The functionality axiom formalizes mathematical practice to view functions simply as relations between input and output, abstracting away from aspects, such as runtime complexity. The second axiom formalizes the property of a logical system to be extensional in the philosophical logic sense, i.e. that equivalent

formulae can be substituted for each other in any context, requiring that the denotation of a formula may not depend on the context. This form is usually assumed in mathematics, but not when it comes e.g. to formalizing beliefs or knowledge. There the sentences "I know that the morning star is actually the evening star" and "I know that the morning star is actually the morning star" are extensionally equal (both are actually the planet Venus), but the first sentence might actually be false, depending on my knowledge of astronomy. *equality!Leibniz*

Lemma 2.1.14 *The axioms* EXT^o_L *are not valid in the class of* Σ*-models.*

Proof: Let $(\mathcal{D}, @, \mathcal{I})$ be any standard Σ-algebra with $\mathcal{D}_o = \{a, b, c\}$, and let the map $\upsilon\colon \mathcal{D}_o \longrightarrow \{\mathtt{T}, \mathtt{F}\}$ be defined by $\upsilon(a) = \upsilon(b) = \mathtt{T}$ and $\upsilon(c) = \mathtt{F}$. Furthermore, let the interpretation function $\mathcal{I}$ behave on connectives and quantifiers as indicated by the following schemata (that for ∨ is induced by the ones for ∧ and ¬):

$\mathcal{I}(\neg)$	a	b	c
	c	c	a

$\mathcal{I}(\wedge)$	a	b	c
a	a	b	c
b	b	b	c
c	c	c	c

$\mathcal{I}(\vee)$	a	b	c
a	a	a	a
b	a	a	a
c	a	a	c

$$\mathcal{I}(\Pi^\alpha)@f = \begin{cases} a, & \text{if } f@g \in \{a, b\} \text{ for all } g \in \mathcal{D}_\alpha \\ c, & \text{if } f@g = c \text{ for some } g \in \mathcal{D}_\alpha \end{cases}$$

We have $\mathcal{I}(X \doteq Y) = \mathcal{I}(\forall P.\neg PX \vee PY) = c$, since $\mathcal{M}$ is standard and $\mathcal{I}_{[p/P]}\neg PX \vee PY = c$ for $p \in \mathcal{D}_{o\to o}$ with $p(a) = a$, $p(b) = c$, $p(c) = c$, which gives $\mathcal{V}_\varphi(\forall X, Y_o.X \Leftrightarrow Y \Leftrightarrow X \doteq Y) = \mathtt{F}$.[8] □ EdNote(8)

Lemma 2.1.15 (Leibniz Equality in Σ-models)
Let $\mathcal{M} := (\mathcal{D}, @, \mathcal{E}, \upsilon)$ *be a* Σ*-model and* φ *be an assignment.*

1. *If* $\mathcal{E}_\varphi(\mathbf{A}) \equiv \mathcal{E}_\varphi(\mathbf{B})$*, then* $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{B})) \equiv \mathtt{T}$.
2. *If* $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{b}}$ *and* $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq^o \mathbf{B})) \equiv \mathtt{T}$*, then* $\mathcal{E}_\varphi(\mathbf{A}) \equiv \mathcal{E}_\varphi(\mathbf{B})$.
3. *If* $\mathcal{M} \in \mathfrak{M}_{\beta\mathsf{q}}$ *and* $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{B})) \equiv \mathtt{T}$*, then* $\mathcal{E}_\varphi(\mathbf{A}) \equiv \mathcal{E}_\varphi(\mathbf{B})$.

Proof: Let $\mathsf{a}, \mathsf{b} \in \mathcal{D}_\alpha$ and $\psi := \varphi, [\mathsf{a}/X], [\mathsf{b}/Y]$.

1. We show that $\upsilon(\mathcal{E}_\varphi(\doteq^\alpha)@\mathsf{a}@\mathsf{b}) \equiv \mathtt{T}$ if $\mathsf{a} \equiv \mathsf{b}$, which entails the assertion. By definition $\mathcal{E}_\varphi(\doteq^\alpha) \equiv \mathcal{E}_\varphi(\lambda X, Y.\forall P.PX \Rightarrow PY)$ and thus $\mathcal{E}_\varphi(\doteq^\alpha)@\mathsf{a}@\mathsf{b} \equiv \mathcal{E}_\psi(\forall P.PX \Rightarrow PY)$. Now let $\mathsf{r} \in \mathcal{D}_{\alpha\to o}$, then $\upsilon(\mathcal{E}_{\psi,[\mathsf{r}/P]}(PX)) \equiv \mathsf{r}@\mathsf{a} \equiv \mathtt{F}$ or $\upsilon(\mathcal{E}_{\psi,[\mathsf{r}/P]}(PY)) \equiv \mathsf{r}@\mathsf{b} \equiv \mathsf{r}@\mathsf{a} \equiv \mathtt{T}$, since υ is total and $\mathsf{a} \equiv \mathsf{b}$. So we see that $\upsilon(\mathcal{E}_\varphi(\doteq)@\mathsf{a}@\mathsf{b}) \equiv \upsilon(\mathcal{E}_{\psi,[\mathsf{r}/P]}(PX \Rightarrow PY)) \equiv \mathtt{T}$ for all $\mathsf{r} \in \mathcal{D}_{\alpha\to o}$, which yields the assertion.

[8] EdNote: check this, in which class is this model? b is not valid, but q is since it is standard. discuss this here using the material near the crossref

extensionality!in models

2. First note that by property $\mathfrak{b}$ we have $\mathcal{D}_o \equiv \{\mathtt{T}, \mathtt{F}\}$ and υ is the identity function on $\mathcal{D}_o$. Let us assume that $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq^o \mathbf{B})) \equiv \mathcal{E}_\psi(\forall P.P\mathbf{A} \Rightarrow P\mathbf{B}) \equiv \mathtt{T}$ but $\mathcal{E}_\varphi(\mathbf{A}) \not\equiv \mathcal{E}_\varphi(\mathbf{B})$, which means that either $\mathcal{E}_\varphi(\mathbf{A}) \equiv \mathtt{T}$ and $\mathcal{E}_\varphi(\mathbf{B}) \equiv \mathtt{F}$ or vice a versa. In the first case we choose a predicate $r := \mathcal{E}_\varphi(\lambda X_o.X_o)$ and get from the first assumption that $\mathcal{E}_{\varphi,[\mathsf{r}/P]}(P\mathbf{A}) \equiv \mathcal{E}_{\varphi,[\mathsf{r}/P]}(\mathbf{A}) \equiv \mathcal{E}_\varphi(\mathbf{A}) \equiv \mathtt{T}$ or $\mathcal{E}_{\varphi,[\mathsf{r}/P]}(P\mathbf{B}) \equiv \mathcal{E}_{\varphi,[\mathsf{r}/P]}(\mathbf{B}) \equiv \mathcal{E}_\varphi(\mathbf{B}) \equiv \mathtt{T}$, which gives us the contradiction. Note that P does not occur free in $\mathbf{A}$ or $\mathbf{B}$ by definition of $\doteq$.

   The second case is analogous with $r := \mathcal{E}_\varphi(\lambda X_o.\neg X_o)$.

3. We show that if $\upsilon(\mathcal{E}_\varphi(\doteq^\alpha)@\mathsf{a}@\mathsf{b}) \equiv \mathtt{T}$, then $\mathsf{a} \equiv \mathsf{b}$, which entails the assertion. Suppose $\mathsf{a} \not\equiv \mathsf{b} \in \mathcal{D}_\alpha$ and $\mathsf{r} \equiv \mathrm{q}^\alpha@\mathsf{a}$, where $\mathrm{q}^\alpha \in \mathcal{D}_{\alpha\to\alpha\to o}$ is the function guaranteed by property q. We know that $\mathrm{q}^\alpha@\mathsf{a}@\mathsf{a} \equiv \mathtt{T}$ and $\mathrm{q}^\alpha@\mathsf{a}@\mathsf{b} \equiv \mathtt{F}$, since $\mathsf{a} \not\equiv \mathsf{b}$ by assumption. Hence $\upsilon(\mathcal{E}_\varphi(\doteq^\alpha)@\mathsf{a}@\mathsf{b}) \equiv \upsilon(\mathcal{E}_\psi(\forall P.PX \Rightarrow PY) \equiv \mathtt{F}$ for $\psi := \varphi, [\mathsf{a}/X], [\mathsf{b}/Y]$, since $\upsilon(\mathcal{E}_{\psi,[\mathsf{r}/P]}(PX \Rightarrow PY)) \equiv \mathtt{F}$, as $\upsilon(\mathcal{E}_{\psi,[\mathsf{r}/P]}(PX)) \equiv \mathrm{q}^\alpha@\mathsf{a}@\mathsf{a} \equiv \mathsf{r}@\mathsf{a} \equiv \mathtt{T}$ and $\upsilon(\mathcal{E}_{\psi,[\mathsf{r}/P]}(PY)) \equiv \mathrm{q}^\alpha@\mathsf{a}@\mathsf{b} \equiv \mathsf{r}@\mathsf{b} \equiv \mathtt{F}$. □

Lemma 2.1.16 (Extensionality in Σ-models)

1. *There exists a model* $\mathcal{M} \in \mathfrak{M}_\beta$ *which is not functional.*
2. *There exists a model* $\mathcal{M} \in \mathfrak{M}_{\beta\eta\mathfrak{b}}$ *for which* $\mathrm{EXT}_L^{\alpha\to\beta}$ *is not valid.*
3. *There exists a model* $\mathcal{M} \in \mathfrak{M}_{\beta\mathsf{q}}$ *for which* EXT_L^o *is not valid.*
4. $\mathrm{EXT}_L^{\alpha\to\beta}$ *is valid in* $\mathcal{M}$*, if* $\mathcal{M} \in \mathfrak{M}_{\beta\mathsf{q}}$.
5. EXT_L^o *is valid in* $\mathcal{M}$*, if* $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{b}}$.

As a consequence the following table characterizes the different properties of the introduced semantical structures. If a formula is valid for a certain semantical structure we use a '+' and a '−' otherwise. Each entry is further marked with a justification referring to one of the above statements.

valid in	$\mathfrak{M}_\beta/\mathfrak{M}_{\beta\eta}$	$\mathfrak{M}_{\beta\mathsf{q}}$	$\mathfrak{M}_{\beta\mathfrak{b}}/\mathfrak{M}_{\beta\eta\mathfrak{b}}$	$\mathfrak{M}_{\beta\mathsf{q}\mathfrak{b}}$
$\mathrm{EXT}_L^{\alpha\to\beta}$	$-_{(2)}$	$+_{(4)}$	$-_{(2)}$	$+_{(4)}$
EXT_L^o	$-_{(3)}$	$-_{(3)}$	$+_{(5)}$	$+_{(5)}$

EdNote(9) **Proof:** Let υ be a Σ-valuation[9] and let $\mathcal{M}$ be a Σ-model based on the β-term structure $\mathcal{TS}(\Sigma)^\beta$ for Σ, i.e., $\mathcal{M} := (\mathit{cwff}(\Sigma)\!\downarrow_\beta, @^\beta, \mathcal{E}^\beta, \upsilon)$. Note that $\mathcal{M}$ need not be functional by Remark 1.4.22 and 2.1.3.

[9] EdNote: We do not know that this exists. Refer to a model existence theorem or get one another way.

equality! properties of Leibniz equality

For the proof of 2. we refer to [And72a], where Andrews constructs a functional Σ-Model (actually a $\mathcal{M} \in \mathfrak{M}_{\beta\eta\mathfrak{b}}$) that lacks the principle of functional extensionality of Leibniz equality.

For 3. note we note that the model constructed in Lemma 2.1.14 is in $\mathfrak{M}_{\beta\mathfrak{q}}$, since it is standard.

Next we consider 4.: Let $\psi := \varphi, [\mathsf{f}/F], [\mathsf{g}/G]$. From $\mathcal{V}_\psi(\forall A_\alpha\text{.}FA \doteq GA) \equiv \mathtt{T}$ we get that for all $\mathsf{a} \in \mathcal{D}_\alpha$ $\mathcal{V}_{\psi,[\mathsf{a}/A]}(FA \doteq GA) \equiv \mathtt{T}$. By Lemma 2.1.15(3) we can conclude that $\mathcal{E}_{\psi,[\mathsf{a}/A]}(FA) \equiv \mathcal{E}_{\psi,[\mathsf{a}/A]}(GA)$ for all $\mathsf{a} \in \mathcal{D}_\alpha$ and hence $\mathcal{E}_{\psi,[\mathsf{a}/A]}(F)@\mathcal{E}_{\psi,[\mathsf{a}/A]}(A) \equiv \mathcal{E}_{\psi,[\mathsf{a}/A]}(G)@\mathcal{E}_{\psi,[\mathsf{a}/A]}(A)$ for all $\mathsf{a} \in \mathcal{D}_\alpha$. By definition of property $\mathfrak{q}$, which includes property η and thus ensures functionality, we get $\mathcal{E}_\psi(F) \equiv \mathcal{E}_\psi(G)$. This finally gives us that $\mathcal{V}_\psi(F \doteq^{\alpha\to\beta} G) \equiv \mathtt{T}$ with Lemma 2.1.15(1).

And finally in 5. we have that for all $\mathsf{a}, \mathsf{b} \in \mathcal{D}_o$ and all assignments φ: $\upsilon(\mathcal{E}_{\varphi,[\mathsf{a}/A][b/B]}(A \Leftrightarrow B)) \equiv \mathtt{T}$, iff $\upsilon(\mathcal{E}_{\varphi,[\mathsf{a}/A][\mathsf{b}/B]}(A)) \equiv \upsilon(\mathcal{E}_{\varphi,[\mathsf{a}/A][\mathsf{b}/B]}(B))$. From $\mathfrak{b}$ we further know that υ is the identity function and hence this statement is valid, iff $\mathcal{E}_{\varphi,[\mathsf{a}/A][\mathsf{b}/B]}(A) \equiv \mathcal{E}_{\varphi,[\mathsf{a}/A][\mathsf{b}/B]}(B)$. For the left to right direction of our statement we can now apply Lemma 2.1.15(1) and in the right to left direction the assertion follows with 2.1.15(2). □

Next we discuss the role of Leibniz equality within the different semantic structures.

Theorem 2.1.17 (Properties of Leibniz Equality)
Let $\mathcal{M}$ be a Σ-model. For all assignments φ and all terms $\mathbf{A}, \mathbf{B}, \mathbf{C} \in wff_\alpha(\Sigma)$ and $\mathbf{F}, \mathbf{G} \in wff_{\alpha\to\beta}(\Sigma)$ we have:

$\mathfrak{M}_\beta$ *If $\mathcal{M} \in \mathfrak{M}_\beta$, then $\upsilon \circ \mathcal{E}_\varphi(\doteq^\alpha)$ is an equivalence relation on $\mathcal{D}_\alpha$. In particular:*

- **re:** $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{A})) \equiv \mathtt{T}$.
- **sy:** *If $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{B})) \equiv \mathtt{T}$, then $\upsilon(\mathcal{E}_\varphi(\mathbf{B} \doteq^\alpha \mathbf{A})) \equiv \mathtt{T}$.*
- **tr:** *If $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{B})) \equiv \mathtt{T}$ and $\upsilon(\mathcal{E}_\varphi(\mathbf{B} \doteq^\alpha \mathbf{C})) \equiv \mathtt{T}$, then $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{C})) \equiv \mathtt{T}$.*
- **co:** *If $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{B})) \equiv \mathtt{T}$, then $\upsilon(\mathcal{E}_\varphi(\mathbf{FA} \doteq^\beta \mathbf{FB})) \equiv \mathtt{T}$.*[10] EdNote(10)

$\mathfrak{M}_{\beta\mathfrak{q}}$ *If $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{q}}$, then $\mathcal{E}_\varphi(\doteq^\alpha)$ is the equality relation on $\mathcal{D}_\alpha$.*

Proof:

$\mathfrak{M}_\beta$ **re** $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{A})) \equiv \mathtt{T}$, iff for all $\mathsf{p} \in \mathcal{D}_{\alpha\to o}$ we have that $\upsilon(\mathcal{E}_{\varphi,[\mathsf{p}/P]}(P\mathbf{A})) \equiv \mathtt{F}$ or $\upsilon(\mathcal{E}_{\varphi,[\mathsf{p}/P]}(P\mathbf{A})) \equiv \mathtt{T}$, which is obvious since υ is total and surjective.

[10] EdNote: I have moved this into the $\mathfrak{M}_\beta$ case, check whether true.

equality!primitive
valuation!with equality
property $\mathfrak{e}$
$\mathfrak{e}$property for models
model!Henkin, with primitive equality
model!with primitive equality
Henkin model with primitive equality

sy Suppose $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq^\alpha \mathbf{B})) \equiv \mathtt{T}$, but $\upsilon(\mathcal{E}_\varphi(\mathbf{B} \doteq^\alpha \mathbf{A})) \equiv \mathtt{F}$. From the latter we get that $\upsilon(\mathcal{E}_{\varphi,[\mathsf{p}/P]}(P\mathbf{B})) \equiv \mathtt{T}$ and $\upsilon(\mathcal{E}_{\varphi,[\mathsf{p}/P]}(P\mathbf{A})) \equiv \mathtt{F}$ for some $\mathsf{p} \in \mathcal{D}_{\alpha\to o}$. Without loss of generality, let $\mathsf{p} := \mathcal{E}_\varphi(V)$ for a fresh variable $V \in \mathcal{V}_{\alpha\to o}$. From the former assumption we know that for all $\mathsf{q} \in \mathcal{D}_{\alpha\to o}$ we have $\upsilon(\mathcal{E}_{\varphi,[\mathsf{q}/P]}(P\mathbf{A})) \equiv \mathtt{F}$ or $\upsilon(\mathcal{E}_{\varphi,[\mathsf{q}/P]}(P\mathbf{B})) \equiv \mathtt{T}$ and hence $\upsilon(\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\lambda X_\bullet \neg VX)/P]}(P\mathbf{A})) \equiv \mathtt{F}$ or $\upsilon(\mathcal{E}_{\varphi,[\mathcal{E}_\varphi(\lambda X_\bullet \neg VX)/P]}(P\mathbf{B})) \equiv \mathtt{T}$ which is equivalent with $\upsilon(\mathcal{E}_{\varphi,[\mathsf{p}/P]}(P\mathbf{A})) \equiv \mathtt{T}$ or $\upsilon(\mathcal{E}_{\varphi,[\mathsf{p}/P]}(P\mathbf{B})) \equiv \mathtt{F}$ and contradicts the latter assumption.

tr, co Similar to **sy** choosing $\mathsf{q} := \mathsf{p}$ and $\mathsf{q} := \mathcal{E}_\varphi(\lambda Z_\bullet V(\mathbf{F}Z))$ respectively to obtain the contradiction.

$\mathfrak{M}_{\beta\mathfrak{q}}$ By 2.1.15.1 and 2.1.15.3

2.1.2 Primitive Equality

The situation of higher-order semantics becomes much simpler if we introduce equality as a primitive logical constant $=$ in Σ, which we will assume for the rest of this section. Since $=$ is logical, we have to specialize the notion of Σ-valuation (cf. 2.1.1) by requiring that $\upsilon(\mathcal{E}(=^\alpha)@\mathsf{a}@\mathsf{b}) \equiv \mathtt{T}$, iff $\mathsf{a} \equiv \mathsf{b}$. In this case, we call υ a Σ-*valuation with equality.*

Furthermore, we say that a Σ-model $\mathcal{M}$ has *property* $\mathfrak{e}$, iff $\mathcal{M}$ has property η and for all types α, $\upsilon(\mathcal{E}(=^\alpha)@\mathsf{a}@\mathsf{b}) \equiv \mathtt{T}$, iff $\mathsf{a} \equiv \mathsf{b}$.

Definition 2.1.18 (Henkin Model with Primitive Equality)
A (functional) Σ-model, which has property $\mathfrak{e}$ is called a (functional) Σ-*model with primitive equality* and a functional one with additional property $\mathfrak{b}$ is called a Σ-*Henkin model with primitive equality.* The class of all Σ-models with primitive equality are denoted by $\mathfrak{M}_{\beta\mathfrak{e}}$ and the class of all Σ-Henkin model with primitive equality by $\mathfrak{M}_{\beta\mathfrak{e}\mathfrak{b}}$.

Remark 2.1.19 (Property $\mathfrak{e}$ implies $\mathfrak{q}$)
It is easy to see that property $\mathfrak{e}$ implies $\mathfrak{q}$. The only difference between these properties is that property $\mathfrak{q}$ only ensures the existence of the intended semantical equality relation, while $\mathfrak{e}$ additionally requires that our new logical connectives $=^\alpha$ are indeed associated with this relations. The connection between property $\mathfrak{q}$ and $\mathfrak{e}$ is already discussed in [And72a]. Andrews concludes that it seems natural to require the existence of logical connectives $=^\alpha$ in the signature if one is interested in extensionality. In this book we are especially interested in shedding some light on both: in extensionality of Leibniz equality in case $=^\alpha \notin \Sigma$ and in extensionality of Leibniz equality and/or primitive equality in case $=^\alpha \in \Sigma$.

As property $\mathfrak{e}$ implies $\mathfrak{q}$ which in turn implies property η, it is easy to see that the landscape of higher-order semantics from Figure 2.1 at Page lxiii collapses to the one in Figure 2.2.

extensionality!for primitive equality
extensionality!functional, for primitive equality
extensionality!Boolean, for primitive equality
extensionality!full, for primitive equality
$\mathrm{EXT}^{\alpha\to\beta}$ *functional extensionality for primitive equality*
EXT^{o} *Boolean extensionality for primitive equality*
axiom!extensionality
extensionality!for primitive and Leibniz equality

$$\begin{tikzcd}
\boxed{\upsilon\text{-Comp}} \arrow[d, dashed, "\text{total}"] & \nabla_c, \nabla_\beta, \nabla_\neg, \nabla_\vee, \nabla_\wedge, \nabla_\forall, \nabla_\exists \\
\boxed{\mathfrak{M}_{\beta\mathfrak{e}}} \arrow[d, "\mathfrak{b}"] & \text{saturated}, \nabla_\mathfrak{q}, \nabla_{=^r}, \nabla_{=^s} \\
\boxed{\mathfrak{M}_{\beta\mathfrak{e}\mathfrak{b}} \equiv \mathfrak{H}_\mathfrak{e}} \arrow[d, "\text{full}"] & \nabla_\mathfrak{b} \\
\boxed{\Sigma\mathcal{HOL}_\mathfrak{e}} &
\end{tikzcd}$$

Figure 2.2: The landscape of Higher-Order Semantics with primitive equality

Definition 2.1.20 (Extensionality for primitive equality)
Analogous to the extensionality axioms for Leibniz equality, we can define such for primitive equality.

$$\begin{array}{lll}
\mathrm{EXT}^{\alpha\to\beta} & := & \forall F_{\alpha\to\beta}\text{.}\forall G_{\alpha\to\beta}\text{.}(\forall X_\beta\text{.}FX = GX) \Rightarrow F =^\beta G \\
\mathrm{EXT}^{o} & := & \forall A_o\text{.}\forall B_o\text{.}(A \Leftrightarrow B) \Leftrightarrow A =^o B
\end{array}$$

the **axioms of full extensionality for primitive equality**.

The following lemma shows that in a Σ-model with full equality the denotations of primitive equations and corresponding Leibniz equations are identical modulo υ.

Lemma 2.1.21 (Primitive and Leibniz equality)
If $\mathcal{M} := (\mathcal{D}, @, \mathcal{E}, \upsilon) \in \mathfrak{M}_{\beta\mathfrak{e}}$, *we have that* $\upsilon(\mathcal{E}_\varphi(\mathbf{A} = \mathbf{B})) \equiv \upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq \mathbf{B}))$ *for all* $\mathbf{A}, \mathbf{B} \in \mathit{wff}(\Sigma)$.

Proof: By Lemma 2.1.15(3) we have $\upsilon(\mathcal{E}_\varphi(\mathbf{A} \doteq \mathbf{B})) \equiv \mathtt{T}$, iff $\mathcal{E}_\varphi(\mathbf{A}) \equiv \mathcal{E}_\varphi(\mathbf{B})$, since each $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}}$ is also in $\mathfrak{M}_{\beta\mathfrak{q}}$. By property $\mathfrak{e}$ this is equivalent to $\upsilon(\mathcal{E}_\varphi(\mathbf{A} = \mathbf{B})) \equiv \mathtt{T}$. □

Remark 2.1.22 The previous lemma shows that in definition 1.2.1 we can indeed use the Leibniz property to treat equality as a defined notion, if we take care to ensure that our Σ-models contain the identity relation for each type (which is obviously the case for Σ-models with equality). Thus we would principally not have needed to assume the constants q^α in our

extensionality!in models with primitive equality
extensionality!Boolean, for primitive equality
extensionality!full, for primitive equality
$\mathrm{EXT}^{\alpha\to\beta}$ functional extensionality for primitive equality
BegNP(11)
EXT^{o} Boolean extensionality for primitive equality
EdNote(12)
extensionality!Boolean, for Leibniz equality
extensionality!full, for Leibniz equality
$\mathrm{EXT}_L^{\alpha\to\beta}$ functional extensionality for Leibniz equality
EXT_L^{o} Boolean extensionality for Leibniz equality
EndNP(11)

signature. The critical part in this choice is that for ensuring the correct meaning for $\mathbf{Q}^\alpha$ we have to require the existence of the identity relation for each type in each Σ-model (see [And72a] for a discussion in the context of Henkin models). This requirement is automatically met, if we have constants $q^\alpha \in \overline{\Sigma}$, so it seems natural to treat equality as primitive.

Lemma 2.1.23 *The axioms* EXT^o *are not valid in the class of* Σ*-models.*[12]

Proof: Let $(\mathcal{D}, @, \mathcal{I})$ be the model constructed in the proof of Lemma 2.1.14 and $\varphi(X) = a$ and $\varphi(Y) = b$, then we can see that $\mathcal{I}_\varphi(X \Rightarrow Y) = a$ and $\mathcal{I}_\varphi(X \Leftarrow Y) = a$, thus $\mathcal{I}_\varphi(X \Leftrightarrow Y) = a$, but $\mathcal{I}(X = Y) = c$. So we have $\mathcal{I}_\varphi(\forall X, Y_o. X \Leftrightarrow Y \wedge X = Y) = c$, and thus $\mathcal{V}_\varphi(\forall X, Y_o. X \Leftrightarrow Y \wedge X = Y) = \mathtt{F}$. □

Lemma 2.1.24 (Extensionality for primitive equality)

1. *There exists a* $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}}$ *for which* EXT^o *and* EXT_L^o *are not valid.*
2. $\mathrm{EXT}^{\alpha\to\beta}$ *and* $\mathrm{EXT}_L^{\alpha\to\beta}$ *are valid in* $\mathcal{M}$*, if* $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}}$.
3. EXT^o *and* EXT_L^o *are valid in* $\mathcal{M}$*, if* $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{eb}}$.

Thus we can extend the table in Lemma 2.1.16 to the following one:

valid in	$\mathfrak{M}_{\beta\mathfrak{e}}$	$\mathfrak{M}_{\beta\mathfrak{eb}}$
$\mathrm{EXT}_L^{\alpha\to\beta}, \mathrm{EXT}^{\alpha\to\beta}$	$+$	$+$
$\mathrm{EXT}_L^o, \mathrm{EXT}^o$	$-$	$+$

Proof: The assertions follow from their respective counterparts in Lemma 2.1.16(3) with Lemma 2.1.21 and the fact that each $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}}$ is also in $\mathfrak{M}_{\beta\mathfrak{q}}$. □

Theorem 2.1.25 *Let* $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}}$*, then* $\upsilon \circ \mathcal{E}_\varphi(\doteq^\alpha)$ *and* $\upsilon \circ \mathcal{E}_\varphi(=^\alpha)$ *are equivalence relations on* $\mathcal{D}_\alpha$ *for all assignments* φ*. If* $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{eb}}$*, then* $\mathcal{E}_\varphi(\doteq^\alpha) \equiv \mathcal{E}_\varphi(=^\alpha)$ *is the equality relation on* $\mathcal{D}_\alpha$.

Proof: If $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{e}}$, then the proofs for $\doteq$ are provided by Lemma 2.1.17. For $=$ they follow immediately with 2.1.21.

If $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{eb}}$, then the argumentation is analogous, and we first show that $\doteq$ and $=$ denote a functional congruence relation. As furthermore υ is the identity relation on $\mathcal{D}_o$ we have that $\doteq$ and $=$ indeed denote the intended semantical equality relations. □

[11] New Part:
[12] EdNote: where is this used (cross-reference)

Remark 2.1.26 (Skolem Paradox)
Skolem paradox!Skolem Cantor

Now we can point out an interesting difference between standard models and Henkin models. One consequence of the model existence theorem (cf. 2.2.32) is the compactness theorem 2.3.22 that states that any satisfiable set of formulae has a countable model. Cantor's theorem 1.2.18 says that the set of sequences of natural numbers is uncountable. The so-called Skolem **paradox** consists in the fact that since Cantor's theorem is valid in all Σ-models (cf. 6.2.6 for a proof from Peano's axioms) it must also be valid in the countable model guaranteed by the compactness theorem.

We can solve this seeming contradiction by taking another look at the exact representation of the theorem. The formula from 1.2.18 only makes an assertion about the cardinality only in standard models that contain all possible functions. In Σ-models it only asserts that $\mathcal{D}_{\iota\to\iota\to\iota} \subseteq \mathcal{F}(\mathcal{D}_\iota;\mathcal{D}_{\iota\to\iota})$ does not contain surjective functions. In particular, we can see that the countable models of Cantor's theorem cannot be standard models.

Remark 2.1.27 If we were only interested in analyzing $\mathcal{HOL}$ with respect to the Henkin semantics, we could have simplified the presentation of the theory by only assuming the equality constants, defining the other logical constants by the following definitions, and treating them as defined formulae: $\neg := (q^o\mathbf{F}_o)$, $\Pi^\alpha := (q^{\alpha\to o}(\lambda X_\alpha\text{.}\mathbf{T}_o))$ and $\wedge := (\lambda X_oY_o\text{.}(\lambda G\text{.}G\mathbf{T}_o\mathbf{T}_o) = (\lambda G\text{.}GXY))$.

Furthermore, we could have weakened the conditions for υ to be a Σ-valuation by only requiring that $\upsilon \circ \mathcal{I}(q^\alpha)$ is the identity relation on $\mathcal{D}_\alpha$, since the definitions above entail that $\upsilon := \mathrm{Id}_{\mathcal{D}_o}$ is a Σ-valuation. Let us consider the cases of 2.1.1.

1. This case is trivial, since $\mathcal{I}(q^\alpha)$ is the identity relation on $\mathcal{D}_\alpha$ by definition.

2. We have $\mathcal{I}(\neg)@\mathsf{a} = \mathcal{I}(q^o\mathbf{F}_o)@\mathsf{a} = \mathcal{I}(q^o)@\mathtt{F}@\mathsf{a} = \mathtt{T}$, iff $\mathsf{a} = \mathtt{F}$, since $\mathcal{D}_o = \{\mathtt{T}, \mathtt{F}\}$.

3. $\mathcal{I}(\Pi^\alpha)@\mathsf{f} = \mathcal{I}(q^{\alpha\to o}(\lambda X_\alpha\text{.}\mathbf{T}_o)@\mathsf{f} = \mathtt{T}$, iff $\mathcal{I}(\lambda X_\alpha\text{.}\mathbf{T}_o) = \mathsf{f}$, iff $\mathtt{T} = \mathcal{I}(\mathbf{T}_o) = \mathcal{I}(\lambda X_\alpha\text{.}\mathbf{T}_o)@\mathsf{a} = \mathsf{f}@\mathsf{a}$ for all $\mathsf{a} \in \mathcal{D}_\alpha$, since $\mathcal{M}$ is functional.

4. Let $a, b \in \mathcal{D}_o$ and $\varphi = [a/X], [b/Y]$, then $\mathcal{I}(\wedge)@a@b = \mathtt{T}$, iff $\mathcal{I}_\varphi(\lambda G\text{.}G\mathbf{T}_o\mathbf{T}_o) = \mathcal{I}_\varphi(\lambda G\text{.}GXY)$, which is the case, iff for all $g \in \mathcal{D}_{o\to o\to o}$ we have $g@\mathtt{T}@\mathtt{T} = g@a@b$, since $\mathcal{M}$ is functional. This is clearly equivalent to the condition that $a = \mathtt{T} = b$.

Note that a construction like the one above is not possible in the case of Σ-models, since the proof of the condition for ¬ requires that $\mathcal{D}_o$ has exactly the elements $\mathtt{T}$ and $\mathtt{F}$, as we take all elements that are not $\mathtt{T}$ to be false.

Hintikka set
extension lemma
closed under subsets

Remark 2.1.28 Let $\mathcal{M} = (\mathcal{D}, \mathcal{I})$ and $\mathcal{N} = (\mathcal{E}, \mathcal{J})$ be Henkin models, and let $\kappa\colon \mathcal{M} \longrightarrow \mathcal{N}$ be a Σ-homomorphism, then by 2.1.2 and 1.4.5(2) we have $\kappa(\mathtt{T}) = \kappa(\mathcal{I}_\varphi(\mathbf{T}_o)) = \mathcal{J}_{\kappa\circ\varphi}(\mathbf{T}_o) = \mathtt{T}$ and similarly $\kappa(\mathtt{F}) = \mathtt{F}$.

Remark 2.1.29 Note that the class of Henkin models defined above is rich in non-standard models[2], since we do not require it to contain a description function. In this detail we differ from most systems of higher-order logic (cf. [Rus08, Chu40, Hen50, And71, And86]), which do require the existence of description or choice operators, which guarantee more functions (see 1.2.8). Even though ours may not be the most interesting notion of model, we choose not to deal with descriptions in this book, which focuses on treating calculi for automated deduction.

2.2 Model Existence

In this section we introduce the model existence theorems for the different semantical notions discussed in Section 2.1. These theorems have the following form, where $* \in \{\mathfrak{M}, \mathfrak{M}_{\beta\mathfrak{b}}, \mathfrak{M}_{\beta\eta}, \mathfrak{M}_{\beta\mathfrak{q}}, \mathfrak{M}_{\beta\eta\mathfrak{b}}, \mathfrak{M}_{\beta\mathfrak{qb}}\, \mathfrak{M}_{\beta\mathfrak{e}}, \mathfrak{M}_{\beta\mathfrak{eb}}\}$:

> *Theorem (Model Existence): For a given abstract consistency class $\mathfrak{Acc}_*$ and a set $H \in \mathfrak{Acc}_*$ there is a Σ-model $\mathcal{M}$ of H, such that $\mathcal{M} \in \mathfrak{M}_*$.*

The most important tools used in the proofs of the model existence theorems are the so-called Σ-Hintikka sets. These sets are maximal elements in abstract consistency classes, and allow computations that resemble those in the considered semantical structures (e.g., Σ-Henkin models). These allow us to construct $*$-valuations for the term structures that turn those into $*$-models.

The key step in the proof of the model existence theorems is an extension lemma, which guarantees a Σ-Hintikka set $\mathcal{H}$ for any set H of sentences in Γ_Σ. Apart from this, the proofs for the model existence theorems are standard.

2.2.1 Abstract Consistency

Let us now review a few technicalities that we will need for the proofs of the model existence theorems.

Definition 2.2.1 (Compactness)
Let $\mathcal{C}$ be a class of sets.

1. $\mathcal{C}$ is called **closed under subsets**, iff for all sets S and T the following condition holds: if $S \subseteq T$ and $T \in \mathcal{C}$, then $S \in \mathcal{C}$.

[2] In our notion of Henkin model we cannot guarantee the existence of, for instance, step functions, i.e. functions $f \in \mathcal{F}(\mathcal{D}_\alpha; \mathcal{D}_\beta)$ that are constant on subsets of $\mathcal{D}_{\alpha\to o}$ given by predicates in $\mathcal{D}_{\alpha\to o}$.

2. $\mathcal{C}$ is called **compact**, iff for every set S the following condition holds: $S \in \mathcal{C}$, iff every finite subset of S is a member of $\mathcal{C}$.

compact
sufficiently Σ-pure
witness constant
constant!witness
$S * a$union of S with singleton a
∇_*abstract consistency property
abstract consistency property
abstract consistency class!properties for

Lemma 2.2.2 *If $\mathcal{C}$ is compact, then $\mathcal{C}$ is closed under subsets.*

Proof: Suppose $S \subseteq T$ and $T \in \mathcal{C}$. Every finite subset A of S is a finite subset of T, and since $\mathcal{C}$ is compact, we know that $A \in \mathcal{C}$. Thus $S \in \mathcal{C}$. □

Definition 2.2.3 (Sufficiently Pure)
Let Σ be a signature and $\mathcal{T}$ be a set of Σ-sentences. $\mathcal{T}$ is called **sufficiently Σ-pure**, iff for each type α there is a set of constants $\mathcal{P}_\alpha \subseteq \Sigma_\alpha$ with equal cardinality to $\mathit{wff}_\alpha(\Sigma)$, such that the elements of $\mathcal{P}$ do not occur in $\mathcal{T}$.

We will always presuppose that sets of sets of sentences are sufficiently Σ-pure in order to have enough witness constants. This can be obtained in practice by enriching the signature with spurious constants. Another way would be to use specially marked variables (which may never be instantiated) as in [Koh94].

Notation 2.2.4 For reasons of legibility we will write $S * a$ for $S \cup \{a\}$, where S is a set. We will use this notation with the convention that $*$ associates to the left.

Definition 2.2.5 (Properties for Abstract Consistency Classes)
Let Γ_Σ be a class of sets of Σ-sentences. We need the following conditions, where $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_o(\Sigma)$ and $\mathbf{F}, \mathbf{G} \in \mathit{cwff}_{\alpha\to\beta}(\Sigma)$:

∇_c If $\mathbf{A}$ is atomic, then $\mathbf{A} \notin \Phi$ or $\neg\mathbf{A} \notin \Phi$.

$\nabla_\neg$ If $\neg\neg\mathbf{A} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$.

∇_β If $\mathbf{A} \in \Phi$ and $\mathbf{B}$ is the β-normal form of $\mathbf{A}$, then $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

∇_η If $\mathbf{A} \in \Phi$ and $\mathbf{B}$ is the $\beta\eta$-normal form of $\mathbf{A}$, then $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

$\nabla_\vee$ If $\mathbf{A} \vee \mathbf{B} \in \Phi$, then $\Phi * \mathbf{A} \in \Gamma_\Sigma$ or $\Phi * \mathbf{B} \in \Gamma_\Sigma$.

$\nabla_\wedge$ If $\neg(\mathbf{A} \vee \mathbf{B}) \in \Phi$, then $\Phi \cup \{\neg\mathbf{A}, \neg\mathbf{B}\} \in \Gamma_\Sigma$.

$\nabla_\forall$ If $\Pi^\alpha\mathbf{F} \in \Phi$, then $\Phi * \mathbf{F}\mathbf{W} \in \Gamma_\Sigma$ for each $\mathbf{W} \in \mathit{cwff}_\alpha(\Sigma)$.

$\nabla_\exists$ If $\neg\Pi^\alpha\mathbf{F} \in \Phi$, then $\Phi * \neg(\mathbf{F}w) \in \Gamma_\Sigma$ for any constant $w \in \Sigma_\alpha$, which does not occur in Φ.

$\nabla_\mathfrak{b}$ If $\neg(\mathbf{A} \doteq^o \mathbf{B}) \in \Phi$, then $\Phi \cup \{\mathbf{A}, \neg\mathbf{B}\} \in \Gamma_\Sigma$ or $\Phi \cup \{\neg\mathbf{A}, \mathbf{B}\} \in \Gamma_\Sigma$.

$\nabla_\mathfrak{q}$ If $\neg(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G}) \in \Phi$, then $\Phi * \neg(\mathbf{F}w \doteq^\beta \mathbf{G}w) \in \Gamma_\Sigma$ for any constant $w \in \Sigma_\alpha$, which does not occur in Φ.

abstract consistency classes
$\mathfrak{Acc}_*$ *abstract consistency class*
abstract consistency class
conversion!α-
abstract consistency classes!non-atomic
consistency!non-atomic

(Additional abstract consistency conditions for primitive equality will be introduced later in Section 2.2.3.)

Remark 2.2.6 Note that for the connectives $\vee, \Pi^\alpha$ there are two conditions — a positive and a negative one — given in the definition above, namely $\nabla_\vee/\nabla_\wedge$ for $\vee$ and $\nabla_\forall/\nabla_\exists$ for Π^α. For $\doteq^o$ and $\doteq^{\alpha\to\beta}$ the situation is different, as we need only conditions for the negative cases. The positive cases can be inferred at the level of Hintikka sets by expanding the Leibniz definition of equality (see the proofs of $\overline{\nabla}_{\mathfrak{q}}^+$ in Lemma 2.2.16 and $\overline{\nabla}_{\mathfrak{b}}^+$ in Lemma 2.2.18).

Definition 2.2.7 (Abstract Consistency Classes)
Let Σ be a signature and Γ_Σ be a class of sets of Σ-sentences that is closed under subsets. Using the properties from the previous definition we introduce the following abstract consistency classes:

$\mathfrak{Acc}_\beta$ If $\nabla_c, \nabla_\neg, \nabla_\beta, \nabla_\vee, \nabla_\wedge, \nabla_\forall$ and $\nabla_\exists$ are valid for Γ_Σ, then Γ_Σ is called an **abstract consistency class** for Σ-models ($\mathfrak{Acc}_\beta$).

Based upon this definition we introduce the following specialized abstract consistency classes: $\mathfrak{Acc}_{\beta\mathfrak{b}}, \mathfrak{Acc}_{\beta\eta}, \mathfrak{Acc}_{\beta\mathfrak{q}}, \mathfrak{Acc}_{\beta\eta\mathfrak{b}}, \mathfrak{Acc}_{\beta\mathfrak{q}\mathfrak{b}}$, where we indicate by indices which additional properties from $\{\nabla_\eta, \nabla_{\mathfrak{q}}, \nabla_{\mathfrak{b}}\}$ are required.

Sometimes we do not want to differentiate between the particular notions above. In this cases we simply speak of an *abstract consistency class*, with which we refer to an arbitrary one in $\{\mathfrak{Acc}_\beta, \mathfrak{Acc}_{\beta\mathfrak{b}}, \mathfrak{Acc}_{\beta\eta}, \mathfrak{Acc}_{\beta\mathfrak{q}}, \mathfrak{Acc}_{\beta\eta\mathfrak{b}}, \mathfrak{Acc}_{\beta\mathfrak{q}\mathfrak{b}}\}$.

Remark 2.2.8 Note that $\mathfrak{Acc}_{\beta\eta}$ corresponds to the abstract consistency property discussed by Andrews in [And71]. The only (technical) difference is that Andrews does not consider α-conversion as built-into the logic but needs a condition similar to ∇_β that requires α-standardized forms to be abstract consistent.

Lemma 2.2.9 (Non-atomic Consistency)
Let Γ_Σ be an abstract consistency class and $\mathbf{A} \in cwff_o(\Sigma)$, then for all $\Phi \in \Gamma_\Sigma$ we have $\mathbf{A} \notin \Phi$ or $\neg\mathbf{A} \notin \Phi$.

Proof: Let $\mathbf{A} \in wff_o(\Sigma)$ and $\Phi \in \Gamma_\Sigma$, such that $\mathbf{A} \in \Phi$. By ∇_β we can assume that $\mathbf{A}$ is a β-normal form. So we prove the assertion by an induction over the structure of $\mathbf{A}$.

If $\mathbf{A}$ is atomic we get the assertion immediately by ∇_c. If $\mathbf{A}$ is not atomic, then its head must be a logical constant, therefore we can proceed by a case-analysis over the connectives and quantifiers.

Suppose $\mathbf{A}$ has the form $\neg\mathbf{B}$ and $\{\neg\mathbf{B}, \neg\neg\mathbf{B}\} \subseteq \Phi$. By $\nabla_\neg$ we know that $\{\neg\mathbf{B}, \mathbf{B}\} \cup \Phi \in \Gamma_\Sigma$, which contradicts the induction hypotheses. Now suppose

$\mathbf{A}$ has the form $\mathbf{B}\vee\mathbf{C}$ and $\{\mathbf{B}\vee\mathbf{C}, \neg(\mathbf{B}\vee\mathbf{C})\} \subseteq \Phi$. By $\nabla_\vee$, $\nabla_\wedge$ we know that $\{\mathbf{B}\vee\mathbf{C}, \neg(\mathbf{B}\vee\mathbf{C}), \mathbf{B}, \neg\mathbf{B}, \neg\mathbf{C}\}\cup\Phi \in \Gamma_\Sigma$ or $\{\mathbf{B}\vee\mathbf{C}, \neg(\mathbf{B}\vee\mathbf{C}), \mathbf{C}, \neg\mathbf{B}, \neg\mathbf{C}\}\cup\Phi \in \Gamma_\Sigma$. In both cases the contradiction is given by the induction hypotheses. Suppose $\mathbf{A}$ has the form $\Pi(\lambda X.\mathbf{B})$ and $\{\Pi(\lambda X.\mathbf{B}), \neg\Pi(\lambda X.\mathbf{B})\} \subseteq \Phi$. Now let $w \in \Sigma_\alpha$ be a constant that does not occur in Φ, by $\nabla_\exists$, $\nabla_\forall$ and ∇_β we know that $\{\Pi(\lambda Y.\mathbf{B}), \neg\Pi(\lambda Y.\mathbf{B}), [w/Y]\mathbf{B}, \neg[w/Y]\mathbf{B}\} \cup \Phi \in \Gamma_\Sigma$ which contradicts the induction hypothesis. □

In contrast to [And71], we work with saturated abstract consistency classes in order to obtain total Σ-valuations, which makes the proofs of the model existence theorem much simpler and, e.g., yields much more natural models.

Definition 2.2.10 (Saturated)
We call an abstract consistency class Γ_Σ **atomically saturated**, iff for all $\Phi \in \Gamma_\Sigma$ and for all atomic sentences $\mathbf{A} \in cwff_o(\Sigma)$, we have $\Phi * \mathbf{A} \in \Gamma_\Sigma$ or $\Phi * \neg\mathbf{A} \in \Gamma_\Sigma$. If this property holds for all sentences $\mathbf{A} \in cwff_o(\Sigma)$, then we call Γ_Σ **saturated**.

Remark 2.2.11 Clearly, not all abstract consistency classes are saturated, since the empty set is one that is not, even if Σ is empty.

In the definition of abstract consistency class, we only had to require atomic consistency, i.e., that there are no atomic propositions that contradict each other in one abstract consistent set, to ensure consistency (cf. 2.2.9). The conjecture is that a similar theorem can be proven for saturatedness:

> **Conjecture:** *Let Γ_Σ be an atomic saturated abstract consistency class. Then there exists a saturated abstract consistency class $\Gamma_\Sigma \subseteq \Gamma'_\Sigma$.*

Such a result would be of practical importance, as would reduce the problem of proving saturatedness of a given calculus to proving atomic saturatedness.

Lemma 2.2.12 *Let Γ_Σ be a saturated abstract consistency class, $\Phi \in \Gamma_\Sigma$ and $\mathbf{A}$ an atomic sentence. Then $\Phi * (\mathbf{A} \vee \neg\mathbf{A}) \in \Gamma_\Sigma$.*

Proof: Since Γ_Σ is saturated and $\Phi \in \Gamma_\Sigma$, we must have $\Phi * (\mathbf{A} \vee \neg\mathbf{A}) \in \Gamma_\Sigma$ or $\Phi * \neg(\mathbf{A} \vee \neg\mathbf{A}) \in \Gamma_\Sigma$. We prove the assertion by refuting the second alternative. If $\Phi * \neg(\mathbf{A} \vee \neg\mathbf{A}) \in \Gamma_\Sigma$, then $\Phi \cup \{\neg(\mathbf{A} \vee \neg\mathbf{A}), \neg\mathbf{A}, \neg\neg\mathbf{A}, \mathbf{A}\} \in \Gamma_\Sigma$ by $\nabla_\wedge$ and $\nabla_\neg$. Since $\mathbf{A}$ is an atomic sentence we get a contradiction with Lemma 2.2.9. □

Lemma 2.2.13 (Compactness of abstract consistency classes)
For each abstract consistency class Γ_Σ there exists an abstract consistency class Γ'_Σ of the same type, such that $\Gamma_\Sigma \subseteq \Gamma'_\Sigma$, and Γ'_Σ is compact. Furthermore Γ'_Σ is saturated if Γ_Σ is.

Proof: (following and extending [And86], proposition no. 2506)
We choose $\Gamma'_\Sigma := \{\Phi \subseteq \mathit{cwff}_o(\Sigma) \mid \text{every finite subset of } \Phi \text{ is in } \Gamma_\Sigma\}$. Now suppose that $\Phi \in \Gamma_\Sigma$. Γ_Σ is closed under subsets, so every finite subset of Φ is in Γ_Σ and thus $\Phi \in \Gamma'_\Sigma$. Hence $\Gamma_\Sigma \subseteq \Gamma'_\Sigma$.

Next let us show that each Γ'_Σ is compact. Suppose $\Phi \in \Gamma'_\Sigma$ and Ψ is an arbitrary finite subset of Φ. By definition of Γ'_Σ all finite subsets of Φ are in Γ_Σ and therefore $\Psi \in \Gamma'_\Sigma$. Thus all finite subsets of Φ are in Γ'_Σ whenever Φ is in Γ'_Σ. On the other hand, suppose all finite subsets of Φ are in Γ'_Σ. Then by the definition of Γ'_Σ the finite subsets of Φ are also in Γ_Σ, so $\Phi \in \Gamma'_\Sigma$. Thus Γ'_Σ is compact.

Next we show that if Γ_Σ satisfies ∇_*, then Γ'_Σ satisfies ∇_*, by considering the cases of Definition 2.2.7. First note that by Lemma 2.2.2 we have that Γ'_Σ is closed under subsets.

∇_c Let $\Phi \in \Gamma'_\Sigma$ and suppose there is an atom $\mathbf{A}$, such that $\{\mathbf{A}, \neg\mathbf{A}\} \subseteq \Phi$. Then $\{\mathbf{A}, \neg\mathbf{A}\} \in \Gamma_\Sigma$ contradicting ∇_c.

$\nabla_\neg$ Let $\Phi \in \Gamma'_\Sigma$, $\neg\neg\mathbf{A} \in \Phi$, Ψ be any finite subset of $\Phi * \mathbf{A}$ and $\Theta := (\Psi \setminus \{\mathbf{A}\}) * \neg\neg\mathbf{A}$. Θ is a finite subset of Φ, so $\Theta \in \Gamma_\Sigma$. Since Γ_Σ is an abstract consistency class and $\neg\neg\mathbf{A} \in \Theta$, we get $\Theta * \mathbf{A} \in \Gamma_\Sigma$ by $\nabla_\neg$. We know that $\Psi \subseteq \Theta * \mathbf{A}$ and Γ_Σ is closed under subsets, so $\Psi \in \Gamma_\Sigma$. Thus every finite subset Ψ of $\Phi * \mathbf{A}$ is in Γ_Σ and therefore by definition $\Phi * \mathbf{A} \in \Gamma'_\Sigma$.

$\nabla_\beta, \nabla_\eta, \nabla_\vee, \nabla_\wedge, \nabla_\forall, \nabla_\exists$ Analogous to $\nabla_\neg$.

$\nabla_\mathfrak{q}$ Let $\Phi \in \Gamma'_\Sigma$, $\neg(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G}) \in \Phi$ and Ψ be any finite subset of $\Phi * \neg(\mathbf{F}w \doteq \mathbf{G}w)$, where $w \in \Sigma_\alpha$ is a constant that does not occur in Φ. We show that $\Psi \in \Gamma_\Sigma$. Clearly $\Theta := (\Psi \setminus \{\neg(\mathbf{F}w \doteq \mathbf{G}w)\}) * \neg(\mathbf{F} \doteq \mathbf{G})$ is a finite subset of Φ and therefore $\Theta \in \Gamma_\Sigma$. Since Γ_Σ satisfies $\nabla_\mathfrak{q}$ and $\neg(\mathbf{F} \doteq \mathbf{G}) \in \Theta$, we have $\Theta * \neg(\mathbf{F}w \doteq \mathbf{G}w) \in \Gamma_\Sigma$ by $\nabla_\mathfrak{q}$. Furthermore, $\Psi \subseteq \Theta * \neg(\mathbf{F}w \doteq \mathbf{G}w)$ and Γ_Σ is closed under subsets, so $\Psi \in \Gamma_\Sigma$. Thus every finite subset Ψ of $\Phi * \neg(\mathbf{F}w \doteq \mathbf{G}w)$ is in Γ_Σ, therefore by definition we have $\Phi * \neg(\mathbf{F}w \doteq \mathbf{G}w) \in \Gamma'_\Sigma$.

$\nabla_\mathfrak{b}$ Let $\Phi \in \Gamma'_\Sigma$ with $\neg(\mathbf{A} \doteq \mathbf{B}) \in \Phi$ but $\Phi \cup \{\mathbf{A}, \neg\mathbf{B}\} \notin \Gamma_\Sigma$ and $\Phi \cup \{\neg\mathbf{A}, \mathbf{B}\} \notin \Gamma_\Sigma$. Then there exists finite subsets Φ_1 and Φ_2 of Φ, such that $\Phi_1 * \{\mathbf{A}, \neg\mathbf{B}\} \notin \Gamma_\Sigma$ and $\Phi_2 * \{\neg\mathbf{A}, \mathbf{B}\} \notin \Gamma_\Sigma$. Now we choose $\Phi_3 := \Phi_1 \cup \Phi_2 * \neg(\mathbf{A} \doteq \mathbf{B})$. Obviously Φ_3 is a finite subset of Φ and therefore $\Phi_3 \in \Gamma_\Sigma$. Since Γ_Σ satisfies $\nabla_\mathfrak{b}$, we have that $\Phi_3 \cup \{\mathbf{A}, \neg\mathbf{B}\} \in \Gamma_\Sigma$ or $\Phi_3 \cup \{\neg\mathbf{A}, \mathbf{B}\} \in \Gamma_\Sigma$. From this and the fact that extensional abstract consistency classes are closed under subsets we get that $\Phi_1 \cup \{\mathbf{A}, \neg\mathbf{B}\} \in \Gamma_\Sigma$ or $\Phi_2 \cup \{\neg\mathbf{A}, \mathbf{B}\} \in \Gamma_\Sigma$, which contradicts our assumption.

For the proof that Γ'_Σ is saturated, let $\Phi \in \Gamma'_\Sigma$, but neither $\Phi * \mathbf{A}$ nor $\Phi * \neg\mathbf{A}$ be in Γ'_Σ. Then there are finite subsets Φ^+ and Φ^- of Φ, such that $\Phi^+ * \mathbf{A} \notin \Gamma_\Sigma$

and $\Phi^- * \neg\mathbf{A} \notin \Gamma_\Sigma$ (since all finite subsets of Φ are in Γ_Σ). As $\Psi := \Phi^+ \cup \Phi^-$ is a finite subset of Φ, we have $\Psi \in \Gamma_\Sigma$. Furthermore, $\Psi * \mathbf{A} \in \Gamma_\Sigma$ or $\Psi * \neg\mathbf{A} \in \Gamma_\Sigma$, because Γ_Σ is saturated. Γ_Σ is closed under subsets, so $\Phi^+ * \mathbf{A} \in \Gamma_\Sigma$ or $\Phi^- * \neg\mathbf{A} \in \Gamma_\Sigma$. This is a contradiction, so we can conclude that if $\Phi \in \Gamma'_\Sigma$, then $\Phi * \mathbf{A} \in \Gamma'_\Sigma$ or $\Phi * \neg\mathbf{A} \in \Gamma'_\Sigma$. □

Hintikka set maximal
Hintikka Lemma!for $\mathfrak{Acc}_\beta$
$\overline{\nabla}_*$ Hintikka set property

2.2.2 Hintikka Sets

Now we define Hintikka sets, which are maximal elements in an abstract consistency class. Hintikka sets connect syntax with semantics as they provide the basis for the model constructions in the model existence theorem 2.2.32.

Definition 2.2.14 (Σ-Hintikka Set)
Let Γ_Σ be an abstract consistency class, then a set $\mathcal{H}$ is called a **Σ-Hintikka set** for Γ_Σ, iff it is **maximal** in Γ_Σ, i.e. iff for each sentence $\mathbf{D} \in cwff_o(\Sigma)$ such that $\mathcal{H} * \mathbf{D} \in \Gamma_\Sigma$, we already have $\mathbf{D} \in \mathcal{H}$.

In the following we discuss properties of Σ-Hintikka sets. Since we have different types of abstract consistency classes, depending on the additional requirements $\eta, \mathfrak{q}$ and $\mathfrak{b}$, we have to discuss different Hintikka lemmata.

Theorem 2.2.15 (Hintikka Lemma for $\mathfrak{Acc}_\beta$)
If Γ_Σ is a saturated $\mathfrak{Acc}_\beta$ and $\mathcal{H}$ is maximal in Γ_Σ, then the following statements hold for all $\mathbf{A}, \mathbf{B} \in cwff_o(\Sigma)$, $\mathbf{F} \in cwff_{\alpha\to o}(\Sigma)$ and $\mathbf{C}, \mathbf{D}, \mathbf{E} \in cwff_\alpha(\Sigma)$:

$\overline{\nabla}_c^1$ $\mathbf{A} \notin \mathcal{H}$ *or* $\neg\mathbf{A} \notin \mathcal{H}$.

$\overline{\nabla}_c^2$ $\mathbf{A} \in \mathcal{H}$, *iff* $\neg\mathbf{A} \notin \mathcal{H}$.

$\overline{\nabla}_c^3$ $\neg\mathbf{A} \in \mathcal{H}$, *iff* $\mathbf{A} \notin \mathcal{H}$.

$\overline{\nabla}_\neg$ $(\neg\neg\mathbf{A}) \in \mathcal{H}$, *iff* $\mathbf{A} \in \mathcal{H}$.

$\overline{\nabla}_\beta$ *If* $\mathbf{A} =_\beta \mathbf{B}$, *then* $\mathbf{A} \in \mathcal{H}$, *iff* $\mathbf{B} \in \mathcal{H}$.

$\overline{\nabla}_\vee$ $(\mathbf{A} \vee \mathbf{B}) \in \mathcal{H}$, *iff* $\mathbf{A} \in \mathcal{H}$ *or* $\mathbf{B} \in \mathcal{H}$.

$\overline{\nabla}_\wedge$ $\neg(\mathbf{A} \vee \mathbf{B}) \in \mathcal{H}$, *iff* $\neg\mathbf{A} \in \mathcal{H}$ *and* $\neg\mathbf{B} \in \mathcal{H}$.

$\overline{\nabla}_\forall$ $\Pi^\alpha\mathbf{F} \in \mathcal{H}$, *iff for each* $\mathbf{D} \in cwff_\alpha(\Sigma)$ *we have* $\mathbf{F}\mathbf{D} \in \mathcal{H}$.

$\overline{\nabla}_\exists$ $\neg\Pi^\alpha\mathbf{F} \in \mathcal{H}$, *iff there is a* $\mathbf{D} \in cwff_\alpha(\Sigma)$, *such that* $\neg\mathbf{F}\mathbf{D} \in \mathcal{H}$.

$\overline{\nabla}_{\doteq}^r$ $\mathbf{A} \doteq^\alpha \mathbf{A} \in \mathcal{H}$

$\overline{\nabla}_{\doteq}^s$ *If* $\mathbf{F}[\mathbf{C}]_p \in \mathcal{H}$ *and* $\mathbf{C} \doteq^\alpha \mathbf{D} \in \mathcal{H}$, *then* $\mathbf{F}[\mathbf{D}]_p \in \mathcal{H}$

$\overline{\nabla}_{\doteq}^{sy}$ $\mathbf{C} \doteq^\alpha \mathbf{D} \in \mathcal{H}$, *iff* $\mathbf{D} \doteq^\alpha \mathbf{C} \in \mathcal{H}$

$\overline{\nabla}_{\dot{=}}^{tr}$ *$\mathbf{C} \doteq^{\alpha} \mathbf{D} \in \mathcal{H}$ and $\mathbf{D} \doteq^{\alpha} \mathbf{E} \in \mathcal{H}$, then $\mathbf{C} \doteq^{\alpha} \mathbf{E} \in \mathcal{H}$*

$\overline{\nabla}_{t}$ *$(\mathbf{A} \vee \neg\mathbf{A}) \in \mathcal{H}$ for any sentence $\mathbf{A}$.*

$\overline{\nabla}_{\dot{=}}^{tf-}$ *$\neg(\mathbf{T}_o \doteq \mathbf{F}_o) \in \mathcal{H}$, if $\mathbf{T}_o$ and $\mathbf{F}_o$ are defined as in Lemma 2.1.2.*

Proof:

$\overline{\nabla}_{c}^{1}$ By 2.2.9.

$\overline{\nabla}_{c}^{2}$**,**$\overline{\nabla}_{c}^{3}$ Both are direct consequences of the saturation of Γ_Σ and $\overline{\nabla}_{c}^{1}$.

$\overline{\nabla}_{\neg}$ If $\neg\neg\mathbf{A} \in \mathcal{H}$, then $\mathcal{H} * \mathbf{A} \in \Gamma_\Sigma$ by $\nabla_{\neg}$. The maximality of $\mathcal{H}$ now gives us that $\mathbf{A} \in \mathcal{H}$. To obtain the converse, let us assume that $\mathbf{A} \in \mathcal{H}$. Then by $\overline{\nabla}_{c}^{2}$ we know that $\neg\mathbf{A} \notin \mathcal{H}$ and by $\overline{\nabla}_{c}^{3}$ $\neg\neg\mathbf{A} \in \mathcal{H}$.

$\overline{\nabla}_{\beta}$ Suppose $\mathbf{A} =_{\beta} \mathbf{B}$. Since β-reduction is terminating and confluent there is unique $\mathbf{C}$, such that $\mathbf{C}$ is the β-normal-form of $\mathbf{A}$ and $\mathbf{B}$. Without loss of generality we show that if $\mathbf{A} \in \mathcal{H}$, then $\mathbf{B} \in \mathcal{H}$. For that we suppose that $\mathbf{A} \in \mathcal{H}$ but $\mathbf{B} \notin \mathcal{H}$. From the latter we get by $\overline{\nabla}_{c}^{3}$ that $\neg\mathbf{B} \in \mathcal{H}$. Note that the β-normal-form of $\mathbf{A}$ is $\mathbf{C}$ and of $\neg\mathbf{B}$ is $\neg\mathbf{C}$. By ∇_{β} and the maximality of $\mathcal{H}$ we know that $\{\mathbf{C}, \neg\mathbf{C}\} \in \mathcal{H}$, which contradicts $\overline{\nabla}_{c}^{1}$.

$\overline{\nabla}_{\vee}$ We get the first direction by $\nabla_{\vee}$ and the maximality of $\mathcal{H}$. For the converse direction let us assume that $\mathbf{A} \in \mathcal{H}$ or $\mathbf{B} \in \mathcal{H}$ but $(\mathbf{A} \vee \mathbf{B}) \notin \mathcal{H}$. Then by $\overline{\nabla}_{c}^{3}$ we get $\neg(\mathbf{A} \vee \mathbf{B}) \in \mathcal{H}$ and by the first direction of $\overline{\nabla}_{\wedge}$ we have $\{\neg\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$ which contradicts the assumption with $\overline{\nabla}_{c}^{1}$.

Note that the argumentation is not circular. In both cases we use the forward direction of the counterpart to verify the backward direction, whereas forward directions are proven directly. The same holds for the proofs of $\overline{\nabla}_{\forall}$ and $\overline{\nabla}_{\exists}$ below.

$\overline{\nabla}_{\wedge}$ Analogous to the $\overline{\nabla}_{\vee}$ case.

$\overline{\nabla}_{\forall}$ Again, we get the first direction by $\nabla_{\forall}$ and the maximality of $\mathcal{H}$. For the converse direction let us assume that $(\mathbf{FD}) \in \mathcal{H}$ for each $\mathbf{D} \in cwff_{\alpha}(\Sigma)$, but $\Pi^{\alpha}\mathbf{F} \notin \mathcal{H}$. Then $\neg\Pi^{\alpha}\mathbf{F} \in \mathcal{H}$ by $\overline{\nabla}_{c}^{3}$ and by the first direction of $\overline{\nabla}_{\exists}$ there is a$\mathbf{D} \in cwff_{\alpha}(\Sigma)$, such that $\neg(\mathbf{FD}) \in \mathcal{H}$ which is a contradiction.

$\overline{\nabla}_{\exists}$ Analogous to $\overline{\nabla}_{\forall}$.

$\overline{\nabla}_{\dot{=}}^{r}$ Suppose $\mathbf{A} \doteq^{\alpha} \mathbf{A} \notin \mathcal{H}$. By $\overline{\nabla}_{c}^{2}$, the definition of $\doteq$ and $\overline{\nabla}_{\exists}$ we have $\neg(\neg\mathbf{QA} \vee \mathbf{QA})) \in \mathcal{H}$ for a $\mathbf{Q} \in cwff_{\beta\to o}(\Sigma)$. Applying $\overline{\nabla}_{\wedge}$ contradicts $\overline{\nabla}_{c}^{2}$.

$\overline{\nabla}_{\dot{=}}^{s}$ Suppose $\mathbf{F}[\mathbf{C}]_p \in \mathcal{H}$ and $\mathbf{C} \doteq \mathbf{D} \in \mathcal{H}$. From the latter we obtain $(\lambda P.\neg P\mathbf{C} \vee P\mathbf{D})(\lambda X.\mathbf{F}[X]_p) \in \mathcal{H}$ by the definition of $\doteq$ and $\overline{\nabla}_{\forall}$. Note that X is free for $\mathbf{F}[Y]_p$ so we have $\neg\mathbf{F}[\mathbf{C}]_p \vee \mathbf{F}[\mathbf{D}]_p \in \mathcal{H}$ by $\overline{\nabla}_{\beta}$. From this we conclude with $\overline{\nabla}_{\vee}$ that $\neg\mathbf{F}[\mathbf{C}]_p \in \mathcal{H}$ or $\mathbf{F}[\mathbf{D}]_p \in \mathcal{H}$. Since the first option contradicts our assumption with $\overline{\nabla}_{c}^{1}$, it must be the case that $\mathbf{F}[\mathbf{D}]_p \in \mathcal{H}$.

$\overline{\nabla}_*$ *Hintikka set property*
Hintikka lemma!for $\mathfrak{Acc}_{\beta\eta}$
$\overline{\nabla}_*$ *Hintikka set property*
Hintikka lemma!for $\mathfrak{Acc}_{\beta\mathfrak{q}}$

$\overline{\nabla}_{\dot{=}}^{sy}$ By $\overline{\nabla}_{\dot{=}}^{r}$ and $\overline{\nabla}_{\dot{=}}^{s}$.

$\overline{\nabla}_{\dot{=}}^{tr}$ By $\overline{\nabla}_{\dot{=}}^{r}$ and$\overline{\nabla}_{\dot{=}}^{s}$.

$\overline{\nabla}_{t}$ Saturation of Γ_Σ and maximality of $\mathcal{H}$ entails that $\mathbf{A} \in \mathcal{H}$ or $\neg\mathbf{A} \in \mathcal{H}$. We now get the assertion by $\overline{\nabla}_{\vee}$.

$\overline{\nabla}_{\dot{=}}^{tf-}$ Let us assume that $\mathbf{T}_o \doteq \mathbf{F}_o \in \mathcal{H}$. By definition of Leibniz equality (instantiating the universal by the identity), we have $\mathbf{T}_o \Rightarrow \mathbf{F}_o \in \mathcal{H}$, which is equivalent to $\mathbf{F}_o \in \mathcal{H}$ by $\overline{\nabla}_{\vee}$, this contratricts the fact that $\mathbf{T}_o \in \mathcal{H}$, (which be know by $\overline{\nabla}_{t}$). □

Theorem 2.2.16 (Hintikka Lemma for $\mathfrak{Acc}_{\beta\eta}$)
If Γ_Σ is a saturated $\mathfrak{Acc}_{\beta\eta}$ and $\mathcal{H}$ is maximal in Γ_Σ, then for all $\mathbf{A},\mathbf{B},\mathbf{C} \in$ cwff$_o(\Sigma)$

$\overline{\nabla}_{\eta}$ *If $\mathbf{A}=_{\beta\eta}\mathbf{B}$, then $\mathbf{A} \in \mathcal{H}$ iff $\mathbf{B} \in \mathcal{H}$.*

Proof: Analogous to $\overline{\nabla}_{\beta}$ in Theorem 2.2.15 □

Theorem 2.2.17 (Hintikka Lemma for $\mathfrak{Acc}_{\beta\mathfrak{q}}$)
If Γ_Σ is a saturated $\mathfrak{Acc}_{\beta\mathfrak{q}}$ and $\mathcal{H}$ is maximal in Γ_Σ, then for all $\mathbf{C} \in$ cwff$_\alpha(\Sigma)$, $\mathbf{F},\mathbf{G} \in$ cwff$_{\alpha\to\beta}(\Sigma)$:

$\overline{\nabla}_{\mathfrak{q}}^{-}$ *$\neg(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G}) \in \mathcal{H}$, iff there is a $\mathbf{C} \in$ cwff$_\alpha(\Sigma)$, such that $\neg(\mathbf{FC} \doteq^{\beta} \mathbf{GC}) \in \mathcal{H}$.*

$\overline{\nabla}_{\mathfrak{q}}^{+}$ *$\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G} \in \mathcal{H}$, iff $\mathbf{FC} \doteq^{\beta} \mathbf{GC} \in \mathcal{H}$ for all $\mathbf{C} \in$ cwff$_\alpha(\Sigma)$*

Proof:

$\overline{\nabla}_{\mathfrak{q}}^{-}$ We get the first direction by $\nabla_{\mathfrak{q}}$ and the maximality of $\mathcal{H}$. For the converse let us suppose that $\neg(\mathbf{FC} \doteq \mathbf{GC}) \in \mathcal{H}$ but $\neg(\mathbf{F} \doteq \mathbf{G}) \notin \mathcal{H}$. From the latter we know by $\overline{\nabla}_{c}^{2}$, that $\mathbf{F} \doteq \mathbf{G} \in \mathcal{H}$ and by $\overline{\nabla}_{\dot{=}}^{s}$ we have that $\neg(\mathbf{GC} \doteq \mathbf{GC}) \in \mathcal{H}$ which contradicts $\overline{\nabla}_{\dot{=}}^{r}$ and $\overline{\nabla}_{c}^{1}$.[13] EdNote(13)

[13]EdNote: The converse direction does not need $\nabla_{\mathfrak{q}}$

$\overline{\nabla}_*$ Hintikka set property
Hintikka lemma!for $\mathfrak{Acc}_{\beta\mathfrak{b}}$

$\overline{\nabla}_{\mathfrak{q}}^{+}$ Suppose $\mathbf{F} \doteq \mathbf{G} \in \mathcal{H}$ but $\mathbf{FC} \doteq \mathbf{GC} \notin \mathcal{H}$, which means by $\overline{\nabla}_c^3$, that $\neg(\mathbf{FC} \doteq \mathbf{GC}) \in \mathcal{H}$. From this we get by the definition of $\doteq$, $\overline{\nabla}_\exists$ and $\overline{\nabla}_\beta$, that $\neg(\neg\mathbf{Q}(\mathbf{FC}) \vee \mathbf{Q}(\mathbf{GC})) \in \mathcal{H}$ for some $\mathbf{Q} \in \mathit{wff}_{\alpha\to o}(\Sigma)$. On the other hand we know from $\mathbf{F} \doteq \mathbf{G} \in \mathcal{H}$ by the definition of $\doteq$ and $\overline{\nabla}_\forall$ that $(\lambda P_{(\alpha\to\beta)\to o}.\neg P\mathbf{F} \vee P\mathbf{G})(\lambda X_{\alpha\to\beta}.\mathbf{Q}(X\mathbf{C})) \in \mathcal{H}$, and hence by $\overline{\nabla}_\beta$ that $\neg\mathbf{Q}(\mathbf{FC}) \vee \mathbf{Q}(\mathbf{GC}) \in \mathcal{H}$, which contradicts $\overline{\nabla}_c^1$. For the converse assume that $\mathbf{FC} \doteq \mathbf{GC} \in \mathcal{H}$ for all $\mathbf{C} \in \mathcal{H}$ but $\mathbf{F} \doteq \mathbf{G} \notin \mathcal{H}$. We get by $\overline{\nabla}_c^3$ that $\neg(\mathbf{F} \doteq \mathbf{G}) \in \mathcal{H}$ which contradicts the assumption with $\overline{\nabla}_{\mathfrak{q}}^{-}$ and $\overline{\nabla}_c^1$. □

Theorem 2.2.18 (Hintikka Lemma for $\mathfrak{Acc}_{\beta\mathfrak{b}}$)
If Γ_Σ is a saturated $\mathfrak{Acc}_{\beta\mathfrak{b}}$ and $\mathcal{H}$ is maximal in Γ_Σ, then for all $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_o(\Sigma)$:

$\overline{\nabla}_{\mathfrak{b}}^{-}$ $\neg(\mathbf{A} \doteq^o \mathbf{B}) \in \mathcal{H}$, *iff* $\{\neg\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H}$ *or* $\{\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$

$\overline{\nabla}_{\mathfrak{b}}^{+}$ $(\mathbf{A} \doteq^o \mathbf{B}) \in \mathcal{H}$, *iff* $\{\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H}$ *or* $\{\neg\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$.

$\overline{\nabla}_{\doteq}^{\Leftrightarrow}$ $(\mathbf{A} \Leftrightarrow \mathbf{B}) \in \mathcal{H}$, *iff* $(\mathbf{A} \doteq^o \mathbf{B}) \in \mathcal{H}$.

$\overline{\nabla}_{\doteq}^{c}$ *Either* $\mathbf{A} \doteq^o \mathbf{B} \in \mathcal{H}$ *or* $\mathbf{A} \doteq^o \neg\mathbf{B} \in \mathcal{H}$.

$\overline{\nabla}_{\doteq}^{tf+}$ *Either* $\mathbf{A} \doteq^o \mathbf{T}_o \in \mathcal{H}$ *or* $\mathbf{A} \doteq^o \mathbf{F}_o \in \mathcal{H}$.

Proof:

$\overline{\nabla}_{\mathfrak{b}}^{-}$ We get the first direction by the definition of $\doteq$, $\nabla_{\mathfrak{b}}$ and the maximality of $\mathcal{H}$. Now assume that $\{\neg\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H}$ or $\{\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$ but $\neg(\mathbf{A} \doteq \mathbf{B}) \notin \mathcal{H}$. From the latter we know by the definition of $\doteq$ and $\overline{\nabla}_\forall$ that $\{(\lambda P_{o\to o}.\neg P\mathbf{A} \vee P\mathbf{B})(\lambda X_o.X), (\lambda P_{o\to o}.\neg P\mathbf{A} \vee P\mathbf{B})(\lambda X_o.\neg X)\} \subseteq \mathcal{H}$ and by $\overline{\nabla}_\beta$ and $\overline{\nabla}_\vee$ that one of $\{\neg\mathbf{A}, \neg\neg\mathbf{A}\}$, $\{\mathbf{B}, \neg\mathbf{B}\}$, $\{\neg\mathbf{A}, \neg\mathbf{B}\}$ or $\{\mathbf{B}, \neg\neg\mathbf{A}\}$ must be a subset of $\mathcal{H}$. All four cases contradict $\overline{\nabla}_c^1$ together with the assumption.

$\overline{\nabla}_{\mathfrak{b}}^{+}$ Since Γ_Σ is saturated we have $\mathbf{A} \in \mathcal{H}$ or $\neg\mathbf{A} \in \mathcal{H}$. From this we easily get the first direction by $\overline{\nabla}_{\doteq}^{s}$. For the converse suppose that $\{\mathbf{A}, \mathbf{B}\} \in \mathcal{H}$ or $\{\neg\mathbf{A}, \neg\mathbf{B}\} \in \mathcal{H}$ but $(\mathbf{A} \doteq \mathbf{B}) \notin \mathcal{H}$ which means by $\overline{\nabla}_c^3$ that $\neg(\mathbf{A} \doteq \mathbf{B}) \in \mathcal{H}$. By $\overline{\nabla}_{\mathfrak{b}}^{-}$ we have $\{\neg\mathbf{A}, \mathbf{B}\} \in \mathcal{H}$ or $\{\mathbf{A}, \neg\mathbf{B}\} \in \mathcal{H}$. In each of the four cases the contradiction follows by $\overline{\nabla}_c^1$.

$\overline{\nabla}_{\doteq}^{\Leftrightarrow}$ If we assume $(\mathbf{A} \Leftrightarrow \mathbf{B}) \in \mathcal{H}$, then by the definition of $\Leftrightarrow$ and $\overline{\nabla}_\wedge$ we have $\{\neg\mathbf{A} \vee \mathbf{B}, \mathbf{A} \vee \neg\mathbf{B}\} \subseteq \mathcal{H}$, and by $\overline{\nabla}_\vee$ that $\{\neg\mathbf{A}, \mathbf{A}\} \subseteq \mathcal{H}$ or $\{\neg\mathbf{B}, \mathbf{B}\} \subseteq \mathcal{H}$ or $\{\neg\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$ or $\{\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H}$. Note that the first two alternatives are impossible because of $\overline{\nabla}_c^1$. Now we assume that

primitive equality
$\mathfrak{Acc}_*$ *abstract consistency class*
abstract consistency properties!with primitive equality
$\nabla_{\doteq}$ *abstract consistency property*
$\nabla_{=}$ *abstract consistency property*
abstract consistency properties

$\mathbf{A} \doteq \mathbf{B} \notin \mathcal{H}$ from which we obtain by $\overline{\nabla}_c^3$ and $\overline{\nabla}_{\mathfrak{b}}^-$ that $\{\neg\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H}$ or $\{\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$. We have to consider four cases and in each case we get a contradiction with $\overline{\nabla}_c^1$.

For the converse direction, we first note that $\mathbf{A} \Rightarrow \mathbf{A} \in \mathcal{H}$ by $\overline{\nabla}_t$, and thus $\mathbf{A} \Leftrightarrow \mathbf{A} \in \mathcal{H}$. If we assume $\mathbf{A} \doteq \mathbf{B}$, then $\mathbf{A} \Leftrightarrow \mathbf{B} \in \mathcal{H}$ follows by $\overline{\nabla}_{\doteq}^s$.

$\overline{\nabla}_{\doteq}^c$ Assume that $\mathbf{A} \doteq \mathbf{B} \notin \mathcal{H}$ and $\mathbf{A} \doteq \neg\mathbf{B} \notin \mathcal{H}$. By $\overline{\nabla}_c^3$ we have $\neg(\mathbf{A} \doteq \mathbf{B}) \in \mathcal{H}$ and $\neg(\mathbf{A} \doteq \neg\mathbf{B}) \in \mathcal{H}$, and by $\overline{\nabla}_{\mathfrak{b}}^-$ we get from the former that $\{\neg\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H}$ or $\{\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$ and from the latter that $\{\neg\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$ or $\{\mathbf{A}, \neg\neg\mathbf{B}\} \subseteq \mathcal{H}$. We have to consider four cases and in each we get a contradiction with $\overline{\nabla}_c^1$. Analogous we can show with $\overline{\nabla}_{\mathfrak{b}}^+$ that $\mathbf{A} \doteq \mathbf{B} \in \mathcal{H}$ and $\mathbf{A} \doteq \neg\mathbf{B} \in \mathcal{H}$ leads to a contradiction.

$\overline{\nabla}_{\doteq}^{tf+}$ Follows immediately from $\overline{\nabla}_{\doteq}^c$. □

2.2.3 Primitive Equality

We now define abstract consistency properties for primitive equality. For this we have different options, e.g., we could introduce primitive equality by postulating $=$ to be a functional congruence relation or alternatively we could state properties connecting $=$ with $\doteq$.

Our concrete choice, namely a property postulating reflexivity and substitutivity of $=$, is motivated from a practical point of view, as we believe that reflexivity and substitutivity are more easy to verify in practical applications.

Definition 2.2.19 (Abstract Consistency for Primitive Equality) Let Σ be signature and let Γ_Σ be a $\mathfrak{Acc}_\beta$, then we define the following condition, where $\Phi \in \Gamma_\Sigma$:

$\nabla_{\mathfrak{e}}^r$ $\neg(\mathbf{A} =^\alpha \mathbf{A}) \notin \Phi$.

$\nabla_{\mathfrak{e}}^s$ If $\mathbf{F}[\mathbf{A}]_p \in \Phi$ and $\mathbf{A} = \mathbf{B} \in \Phi$, then $\Phi * \mathbf{F}[\mathbf{B}]_p \in \Gamma_\Sigma$.

Using these properties we introduce the abstract consistency classes $\mathfrak{Acc}_{\beta\mathfrak{e}}$ and $\mathfrak{Acc}_{\beta\mathfrak{eb}}$ based upon the definition of $\mathfrak{Acc}_\beta$: An $\mathfrak{Acc}_\beta$ ($\mathfrak{Acc}_{\beta\mathfrak{b}}$) that also satisfies $\nabla_{\mathfrak{e}}^r$ and $\nabla_{\mathfrak{e}}^s$ is an $\mathfrak{Acc}_{\beta\mathfrak{e}}$ ($\mathfrak{Acc}_{\beta\mathfrak{eb}}$).

Instead of using reflexivity and substitution principles we can also introduce the following alternative definition for abstract consistency with primitive equality:

Definition 2.2.20 (Alternative Abstract Consistency) Let Σ be signature and let Γ_Σ be a $\mathfrak{Acc}_\beta$, then we define the following condition, where $\Phi \in \Gamma_\Sigma$:

$\doteq$Leibniz equality
=primitive equality
equality!Leibniz
equality!primitive

$\nabla^{\doteq}_{=}$ If $\neg(\mathbf{A} = \mathbf{B}) \in \Phi$, then $\Phi * \neg(\mathbf{A} \doteq \mathbf{B}) \in \Gamma_\Sigma$.

$\nabla^{=}_{\doteq}$ If $\neg(\mathbf{A} \doteq \mathbf{B}) \in \Phi$, then $\Phi * \neg(\mathbf{A} = \mathbf{B}) \in \Gamma_\Sigma$.

Using these properties instead of $\nabla_{\mathfrak{e}}^{r}$ and $\nabla_{\mathfrak{e}}^{s}$ we can introduce analogues of the abstract consistency classes $\mathfrak{Acc}_{\beta\mathfrak{e}}$ and $\mathfrak{Acc}_{\beta\mathfrak{eb}}$ above.

Remark 2.2.21 Just as in the case with Leibniz equality, we can extend an abstract consistency class with primitive equality so that it is compact.

Proof: We proceed just as in the proof of Lemma 2.2.13 but check the cases for $\nabla_{\mathfrak{e}}^{r}$ and $\nabla_{\mathfrak{e}}^{s}$.

For $\nabla_{\mathfrak{e}}^{r}$ let $\Phi \in \Gamma'_\Sigma$ and suppose there is an $\mathbf{A} \in cwff_o$ with $\neg(\mathbf{A} = \mathbf{A}) \in \Phi$. Then $\{\neg(\mathbf{A} = \mathbf{A})\} \in \Gamma_\Sigma$ contradicting $\nabla_{\mathfrak{e}}^{r}$.

For $\nabla_{\mathfrak{e}}^{s}$ let $\Phi \in \Gamma'_\Sigma$, $\{\mathbf{F}[\mathbf{A}]_p, \mathbf{A} = \mathbf{B}\} \subseteq \Phi$, Ψ be any finite subset of $\Phi * \mathbf{F}[\mathbf{B}]_p$ and $\Theta := (\Psi \setminus \{\mathbf{F}[\mathbf{B}]_p\}) \cup \{\mathbf{F}[\mathbf{A}]_p, \mathbf{A} = \mathbf{B}\}$. Θ is a finite subset of Φ, so $\Theta \in \Gamma_\Sigma$. Since Γ_Σ is an $\mathfrak{Acc}_{\beta\mathfrak{eb}}$ and $\{\mathbf{F}[\mathbf{A}]_p, \mathbf{A} = \mathbf{B}\} \subseteq \Theta$, we get $\Theta * \mathbf{F}[\mathbf{B}]_p \in \Gamma_\Sigma$ by $\nabla_{\mathfrak{e}}^{s}$. We know that $\Psi \subseteq \Theta * \mathbf{F}[\mathbf{B}]_p$ and Γ_Σ is closed under subsets, so $\Psi \in \Gamma_\Sigma$. Thus every finite subset Ψ of $\Phi * \mathbf{F}[\mathbf{B}]_p$ is in Γ_Σ and therefore by definition $\Phi * \mathbf{F}[\mathbf{B}]_p \in \Gamma'_\Sigma$. $\square$

The next lemma shows the connection between Leibniz equality and primitive equality for $\mathfrak{Acc}_{\beta\mathfrak{e}}$.

Lemma 2.2.22 (Leibniz versus Primitive Equality)
Let Γ_Σ be a saturated $\mathfrak{Acc}_{\beta\mathfrak{e}}$. For all $\Phi \in \Gamma_\Sigma$, all $\mathbf{A}, \mathbf{B} \in wff_o(\Sigma)$ and $\mathbf{F}, \mathbf{G} \in wff_{\alpha\to\beta}(\Sigma)$ holds:

1. *If $\neg(\mathbf{A} \doteq^\alpha \mathbf{B}) \in \Phi$, then $\Phi * \neg(\mathbf{A} =^\alpha \mathbf{B}) \in \Gamma_\Sigma$*
2. *If $\neg(\mathbf{A} =^\alpha \mathbf{B}) \in \Phi$, then $\Phi * \neg(\mathbf{A} \doteq^\alpha \mathbf{B}) \in \Gamma_\Sigma$*
3. *If $\mathbf{A} \doteq^\alpha \mathbf{B} \in \Phi$, then $\Phi * \mathbf{A} =^\alpha \mathbf{B} \in \Gamma_\Sigma$*
4. *If $\mathbf{A} =^\alpha \mathbf{B} \in \Phi$, then $\Phi * \mathbf{A} \doteq^\alpha \mathbf{B} \in \Gamma_\Sigma$*
5. *If $\neg(\mathbf{F} =^{\alpha\to\beta} \mathbf{G}) \in \Phi$, then $\Phi * \neg(\mathbf{F}w =^\beta \mathbf{G}w) \in \Gamma_\Sigma$ for any constant $w \in \Sigma_\alpha$, which does not occur in Φ.*

Proof:

1. Suppose $\neg(\mathbf{A} \doteq^\alpha \mathbf{B}) \in \Phi$, but $\Phi * \neg(\mathbf{A} =^\alpha \mathbf{B}) \notin \Gamma_\Sigma$. Since Γ_Σ is saturated we have $\Phi * \mathbf{A} =^\alpha \mathbf{B} \in \Gamma_\Sigma$ and by $\nabla_{\mathfrak{e}}^{s}$, that $\Phi * \mathbf{A} =^\alpha \mathbf{B} * \neg(\mathbf{B} \doteq^\alpha \mathbf{B}) \in \Gamma_\Sigma$. From the definition of $\doteq$ we further conclude with $\nabla_\exists$ that $\Phi * \mathbf{A} =^\alpha \mathbf{B} * \neg(\mathbf{B} \doteq^\alpha \mathbf{B}) * \neg(\neg p\mathbf{B} \lor p\mathbf{B}) \in \Gamma_\Sigma$ for any constant $p \in \Sigma_{\alpha\to o}$. From this we get the contradiction with $\nabla_\land$ and Lemma 2.2.9.

2. Suppose $\neg(\mathbf{A} =^\alpha \mathbf{B}) \in \Phi$, but $\Phi * \neg(\mathbf{A} \doteq^\alpha \mathbf{B}) \notin \Gamma_\Sigma$. Since Γ_Σ is saturated we have $\Phi * \mathbf{A} \doteq^\alpha \mathbf{B} \in \Gamma_\Sigma$ and by definition of $\doteq$, $\nabla_\forall$ and the subset closure of Γ_Σ that $\Phi * (\lambda P_{\alpha\to o}\boldsymbol{.}\neg P\mathbf{A} \vee P\mathbf{B})(\lambda X_\alpha\boldsymbol{.}\mathbf{A} = X) \in \Gamma_\Sigma$. By ∇_β, $\nabla_\vee$ and the subset closure of Γ_Σ we finally get that $\Phi * \neg(\mathbf{A} = \mathbf{A}) \in \Gamma_\Sigma$ or $\Phi * \mathbf{A} = \mathbf{B} \in \Gamma_\Sigma$. The former is contradictory with $\nabla_{\mathfrak{e}}^r$ and Lemma 2.2.9, and the latter with the assumption $\neg(\mathbf{A} =^\alpha \mathbf{B}) \in \Phi$ and Lemma 2.2.9.

$\overline{\nabla}_*$ *Hintikka set property* *Hintikka lemma!for $\mathfrak{Acc}_{\beta\mathfrak{e}}$*

3. Suppose $\mathbf{A} \doteq^\alpha \mathbf{B} \in \Phi$, but $\Phi * \mathbf{A} =^\alpha \mathbf{B} \notin \Gamma_\Sigma$. Since Γ_Σ is saturated we have $\Phi * \neg(\mathbf{A} =^\alpha \mathbf{B}) \in \Gamma_\Sigma$ and by (2) and the subset closure of Γ_Σ that $\Phi * \neg(\mathbf{A} \doteq^\alpha \mathbf{B}) \in \Gamma_\Sigma$ which contradicts the assumption with Lemma 2.2.9.

4. Analogous to (3) with (1).

5. From $\neg(\mathbf{F} =^{\alpha\to\beta} \mathbf{G}) \in \Phi$ we can derive with (2), $\nabla_{\mathfrak{q}}$, (1) and the subset closure of Γ_Σ that $\Phi * \neg(\mathbf{F}w =^{\alpha\to\beta} \mathbf{G}w) \in \Gamma_\Sigma$. □

We now discuss two new Hintikka lemmata, which take the logical nature of $=$ into account

Theorem 2.2.23 (Hintikka Lemma for $\mathfrak{Acc}_{\beta\mathfrak{e}}$)
If Γ_Σ is a saturated $\mathfrak{Acc}_{\beta\mathfrak{e}}$ and $\mathcal{H}$ is maximal in Γ_Σ, then the following statements hold for all $\mathbf{A}, \mathbf{B}, \mathbf{C} \in cwff_\alpha(\Sigma)$, $\mathbf{F}, \mathbf{G} \in cwff_{\alpha\to\beta}(\Sigma)$ and $\mathbf{D}, \mathbf{E} \in cwff_o(\Sigma)$:

$\overline{\nabla}_=^r$ $(\mathbf{A} =^\alpha \mathbf{A}) \in \mathcal{H}$.

$\overline{\nabla}_=^s$ *If* $\mathbf{D}[\mathbf{A}]_p \in \mathcal{H}$ *and* $\mathbf{A} =^\alpha \mathbf{B} \in \mathcal{H}$, *then* $\mathbf{D}[\mathbf{B}]_p \in \mathcal{H}$.

$\overline{\nabla}_=^{sy}$ $\mathbf{A} =^\alpha \mathbf{B} \in \mathcal{H}$, *iff* $(\mathbf{B} =^\alpha \mathbf{A}) \in \mathcal{H}$.

$\overline{\nabla}_=^{tr}$ $\mathbf{A} =^\alpha \mathbf{B} \in \mathcal{H}$ *and* $\mathbf{B} =^\alpha \mathbf{C} \in \mathcal{H}$, *then* $\mathbf{A} =^\alpha \mathbf{C} \in \mathcal{H}$

$\overline{\nabla}_=^{q-}$ $\neg(\mathbf{F} =^{\alpha\to\beta} \mathbf{G}) \in \mathcal{H}$, *iff there is a* $\mathbf{C} \in wff_\alpha(\Sigma)$, *such that* $\neg(\mathbf{FC} =^\beta \mathbf{GC}) \in \mathcal{H}$.

$\overline{\nabla}_=^{q+}$ $\mathbf{F} =^{\alpha\to\beta} \mathbf{G} \in \mathcal{H}$, *iff* $\mathbf{FC} =^\beta \mathbf{GC} \in \mathcal{H}$ *for all* $\mathbf{C} \in cwff_\alpha(\Sigma)$.

$\overline{\nabla}_=^{\doteq+}$ $\mathbf{A} \doteq^\alpha \mathbf{B} \in \mathcal{H}$, *iff* $\mathbf{A} =^\alpha \mathbf{B} \in \mathcal{H}$.

$\overline{\nabla}_=^{\doteq-}$ $\neg(\mathbf{A} \doteq^\alpha \mathbf{B}) \in \mathcal{H}$, *iff* $\neg(\mathbf{A} =^\alpha \mathbf{B}) \in \mathcal{H}$.

Proof: We obtain $\overline{\nabla}_=^{\doteq+}$ by maximality of $\mathcal{H}$, 2.2.22(3) and 2.2.22(4). Similarly we get $\overline{\nabla}_=^{\doteq-}$, only with 2.2.22(1) and 2.2.22(2). The other assertions are direct consequence of these and their respective counterparts in Theorem 2.2.15. □

model existence
extension lemma
abstract extension lemma

Theorem 2.2.24 (Hintikka Lemma for $\mathfrak{Acc}_{\mathfrak{M}_{\beta\mathfrak{eb}}}$)
If Γ_Σ is a saturated $\mathfrak{Acc}_{\mathfrak{M}_{\beta\mathfrak{eb}}}$ and $\mathcal{H}$ is maximal in Γ_Σ, then for all $\mathbf{A}, \mathbf{B} \in cwff_o(\Sigma)$:

$\overline{\nabla}_{=}^{b-}$ $\neg(\mathbf{A} =^o \mathbf{B}) \in \mathcal{H}$, *iff* $\{\neg\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H}$ *or* $\{\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$.

$\overline{\nabla}_{=}^{b+}$ $\mathbf{A} =^o \mathbf{B} \in \mathcal{H}$, *iff* $\{\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H}$ *or* $\{\neg\mathbf{A}, \neg\mathbf{B}\} \subseteq \mathcal{H}$.

$\overline{\nabla}_{=}^{\Leftrightarrow}$ $\mathbf{A} \Leftrightarrow \mathbf{B} \in \mathcal{H}$, *iff* $\mathbf{A} = \mathbf{B} \in \mathcal{H}$.

$\overline{\nabla}_{=}^{c}$ *Either* $\mathbf{A} = \mathbf{B} \in \mathcal{H}$ *or* $\mathbf{A} = \neg\mathbf{B} \in \mathcal{H}$.

Proof: The statements follow directly from their counterparts $\overline{\nabla}_{\mathfrak{b}}^{-}$ — $\overline{\nabla}_{\doteq}^{c}$ in Lemma 2.2.18 with the help of $\overline{\nabla}_{=}^{\doteq+}$ and $\overline{\nabla}_{=}^{\doteq-}$. □

2.2.4 Model Existence Theorems

We shall now present the proof of the abstract extension lemma, which will nearly immediately yield the model existence theorems. For the proof we adapt the construction of Henkin's completeness proof from [Hen50, Hen96].

Theorem 2.2.25 (Abstract Extension Lemma)
Let Σ be a signature, Γ_Σ be a compact abstract consistency class and let $H \in \Gamma_\Sigma$ be sufficiently Σ-pure. Then there exists a Σ-Hintikka set $\mathcal{H}$ for Γ_Σ, such that $H \subseteq \mathcal{H}$.

Proof: We construct $\mathcal{H}$ by inductively constructing a sequence of sets $\mathcal{H}^i$ such that $\mathcal{H}^i \in \Gamma_\Sigma$. Then the Σ-Hintikka set is $\mathcal{H} := \bigcup_{i\in\mathbb{N}} \mathcal{H}^i \in \Gamma_\Sigma$.
Let $\mathbf{A}_1, \mathbf{A}_2, \ldots$ be an enumeration of $cwff_o(\Sigma)$. We define $\mathcal{H}^0 := H$ and the set $\mathcal{H}^{n+1}$ according to the following Table 2.3. Since the construction is uniform for all kinds of abstract consistency classes $\mathcal{H}^{n+1}$ depends on the respective kind of abstract consistency class Γ_Σ we are interested in and in the properties of $\mathbf{A}_n$ with respect to this Γ_Σ. (*How to read the table*: Assume Γ_Σ is an an $\mathfrak{Acc}_{\beta\mathfrak{q}}$ and $\mathbf{A}_n$ is of form $\neg(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G})$. The table defines $\mathcal{H}^{n+1}$ to be $\mathcal{H}^n * \mathbf{A}_n * \neg(\mathbf{F}w \doteq^{\beta} \mathbf{G}w)$ for a fresh $w \in \Sigma_\alpha$ in case $\mathbf{A}_n \in \Gamma_\Sigma$ and $\mathcal{H}^n$ otherwise.)

Next we show by induction, that $\mathcal{H}^n \in \Gamma_\Sigma$ for all $n \in \mathbb{N}$. The base case holds by construction (for all kinds of abstract consistency classes). So let $\mathcal{H}^n \in \Gamma_\Sigma$. We have to show that $\mathcal{H}^{n+1} \in \Gamma_\Sigma$. This is trivial in case $\mathcal{H}^n * \mathbf{A}_n \notin \Gamma_\Sigma$ (again for all abstract consistency classes). In case $\mathcal{H}^n * \mathbf{A}_n \in \Gamma_\Sigma$ we have to consider four sub-cases:

1. If $\mathbf{A}_n$ is of the form $\neg\Pi^\alpha\mathbf{B}$, then we get the conclusion trivially by $\nabla_\exists$ (for all cases).

$\dot\sim_{\mathcal{H}}$ functional congruence relation

$\mathcal{H}^{n+1}$		$\mathfrak{Acc}_{\beta}/\mathfrak{Acc}_{\beta\mathfrak{b}}/$ $\mathfrak{Acc}_{\beta\eta}/\mathfrak{Acc}_{\beta\eta\mathfrak{b}}$	$\mathfrak{Acc}_{\beta\mathfrak{q}}/\mathfrak{Acc}_{\beta\mathfrak{qb}}/$	$\mathfrak{Acc}_{\beta\mathfrak{e}}/\mathfrak{Acc}_{\beta\mathfrak{eb}}/$
$\mathcal{H}^n * \mathbf{A}_n \notin \Gamma_\Sigma$		$\mathcal{H}^n$	$\mathcal{H}^n$	$\mathcal{H}^n$
$\mathcal{H}^n * \mathbf{A}_n \in \Gamma_\Sigma$ and	$\mathbf{A}_n$ of form $\neg\Pi^\alpha\mathbf{B}$	$\mathcal{H}^n * \mathbf{A}_n * \neg\mathbf{B}w$	$\mathcal{H}^n * \mathbf{A}_n * \neg\mathbf{B}w$	$\mathcal{H}^n * \mathbf{A}_n * \neg\mathbf{B}w$
	$\mathbf{A}_n$ of form $\neg(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G})$	$\mathcal{H}^n * \mathbf{A}_n$	$\mathcal{H}^n * \mathbf{A}_n * \neg(\mathbf{F}w \doteq^{\beta} \mathbf{G}w)$	$\mathcal{H}^n * \mathbf{A}_n * \neg(\mathbf{F}w \doteq^{\beta} \mathbf{G}w)$
	$\mathbf{A}_n$ of form $\neg(\mathbf{F} =^{\alpha\to\beta} \mathbf{G})$	$\mathcal{H}^n * \mathbf{A}_n$	$\mathcal{H}^n * \mathbf{A}_n$	$\mathcal{H}^n * \mathbf{A}_n * \neg(\mathbf{F}w =^{\beta} \mathbf{G}w)$
	$\mathbf{A}_n$ other form	$\mathcal{H}^n * \mathbf{A}_n$	$\mathcal{H}^n * \mathbf{A}_n$	$\mathcal{H}^n * \mathbf{A}_n$
$w \in \Sigma_\alpha$ is a constant which is fresh for $\mathcal{H}^n$				

Figure 2.3: Hintikka set construction: How to construct sets $\mathcal{H}^{n+1}$ from $\mathcal{H}^n$

2. If $\mathbf{A}_n$ is of form $\neg(\mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G})$ the conclusion is either trivial (for $\mathfrak{Acc}_{\beta}$, $\mathfrak{Acc}_{\beta\mathfrak{b}}$, $\mathfrak{Acc}_{\beta\eta}$a or $\mathfrak{Acc}_{\beta\eta\mathfrak{b}}$), or follows by $\nabla_{\mathfrak{q}}$ for $\mathfrak{Acc}_{\beta\mathfrak{q}}$ or $\mathfrak{Acc}_{\beta\mathfrak{qb}}$. In case of $\mathfrak{Acc}_{\beta\mathfrak{e}}$ or $\mathfrak{Acc}_{\beta\mathfrak{eb}}$ we obtain the assertion by $\nabla_{\mathfrak{q}}$ and Lemma 2.2.22(1).
3. If $\mathbf{A}_n$ is of form $\neg(\mathbf{F} =^{\alpha\to\beta} \mathbf{G})$ the conclusion is either trivial (for $\mathfrak{Acc}_{\beta}$, $\mathfrak{Acc}_{\beta\mathfrak{b}}$, $\mathfrak{Acc}_{\beta\eta}$, $\mathfrak{Acc}_{\beta\eta\mathfrak{b}}$, $\mathfrak{Acc}_{\beta\mathfrak{q}}$, $\mathfrak{Acc}_{\beta\mathfrak{qb}}$) or follows from 2.2.22(5).
4. If $\mathbf{A}_n$ is of any other form, then the conclusion is trivial (for all cases).

Since Γ_Σ is compact, we also have $\mathcal{H} \in \Gamma_\Sigma$.

Now we know that our inductively defined set $\mathcal{H}$ is indeed in Γ_Σ and that $H \subseteq \mathcal{H}$. It only remains to show that $\mathcal{H}$ is maximal in Γ_Σ. So let $\mathbf{A}_n \in \mathit{cwff}_o(\Sigma)$ be the n-th sentence from the above sequence, such that $\mathcal{H} * \mathbf{A}_n \in \Gamma_\Sigma$. Since $\mathcal{H}$ is closed under subsets we know that $\mathcal{H}^n * \mathbf{A}_n \in \Gamma_\Sigma$. By definition of $\mathcal{H}^{n+1}$ we conclude that $\mathbf{A}_n \in \mathcal{H}^{n+1}$ and hence $\mathbf{A}_n \in \mathcal{H}$. $\square$ BegNP(14)

Next we define two congruence relations, $\dot\sim_{\mathcal{H}}$ and $\sim_{\mathcal{H}}$, which we need in the model existence theorems below in order to build quotient models. $\dot\sim_{\mathcal{H}}$ is just the congruence induced by Leibniz equality in the Hintikka set $\mathcal{H}$.

Definition 2.2.26 (Congruence Relation $\dot\sim_{\mathcal{H}}$)
Let Γ_Σ be an abstract consistency class and $\mathcal{H}$ be a Hintikka set for Γ_Σ. For all $\mathbf{A}, \mathbf{B} \in \mathit{wff}(\Sigma)$ we define $\mathbf{A}_\gamma \dot\sim_{\mathcal{H}} \mathbf{B}_\gamma$, iff $\mathbf{A} \doteq^\gamma \mathbf{B} \in \mathcal{H}$.

Lemma 2.2.27 (Properties of $\dot\sim_{\mathcal{H}}$)
Let Γ_Σ be an abstract consistency class and $\mathcal{H}$ be a Hintikka set for Γ_Σ. Then $\dot\sim_{\mathcal{H}}$ is a functional congruence relation, if Γ_Σ is an $\mathfrak{Acc}_{\beta\mathfrak{qb}}$.

Proof: $\dot\sim_{\mathcal{H}}$ is a functional congruence relation by $\overline{\nabla}^r_{\doteq}$, $\overline{\nabla}^{sy}_{\doteq}$, $\overline{\nabla}^{tr}_{\doteq}$, $\overline{\nabla}^-_{\mathfrak{q}}$ and $\overline{\nabla}^+_{\mathfrak{q}}$, which are valid in case Γ_Σ is an $\mathfrak{Acc}_{\beta\mathfrak{qb}}$. $\square$

[14]New Part:

$\sim_{\mathcal{H}}$functional congruence relation
functional!congruence relation

Unfortunately, this only is a congruence, if property $\mathfrak{q}$ is present. Therefore we have to define another, more complicated relation $\sim_{\mathcal{H}}$ to do the job in the case where we do not have $\mathfrak{q}$. Its construction employs a function Υ_γ which we define first.

Definition 2.2.28 Let $\mathbf{A}, \mathbf{B} \in \mathit{wff}_\alpha(\Sigma)$ be terms. We define a function $\Upsilon_\alpha\colon \mathit{wff}_\alpha(\Sigma)^2 \longrightarrow \mathbb{N}$, which computes the number of recursively embedded functionally comparable arguments which differ syntactically but might be semantically equal.

$$\Upsilon_\alpha(\mathbf{A}, \mathbf{B}) = \begin{cases} 1 + \Upsilon_{\beta\to\alpha}(\mathbf{F}, \mathbf{G}) & \text{if } \mathbf{A} \equiv (\mathbf{F}_{\beta\to\alpha}\mathbf{C}_\beta) \text{ and } \mathbf{B} \equiv (\mathbf{G}_{\beta\to\alpha}\mathbf{D}_\beta) \\ \quad + \Upsilon_\beta(\mathbf{C}, \mathbf{D}) & \quad \text{and } \mathbf{C}_\beta \not\equiv \mathbf{D}_\beta \\ 0 & \text{else} \end{cases}$$

Definition 2.2.29 (Congruence Relation $\sim_{\mathcal{H}}$)
Let Γ_Σ be an abstract consistency class and $\mathcal{H}$ be a Hintikka set for Γ_Σ. For all $\mathbf{A}, \mathbf{B} \in \mathit{wff}(\Sigma)$ we define $\mathbf{A}_\gamma \sim_{\mathcal{H}} \mathbf{B}_\gamma$ by a case-analysis on $\Upsilon_\gamma(\mathbf{A}, \mathbf{B})$:

1. If $\Upsilon_\gamma(\mathbf{A}, \mathbf{B}) = 0$, then

$$\mathbf{A}_\gamma \sim_{\mathcal{H}} \mathbf{B}_\gamma, \text{ iff } \begin{cases} (i) & \{\mathbf{A}, \mathbf{B}\} \subseteq \mathcal{H} \text{ or } \{\mathbf{A}, \mathbf{B}\} \cap \mathcal{H} \equiv \emptyset & \text{if } \gamma \equiv o \\ (ii) & \mathbf{A} \equiv \mathbf{B} & \text{if } \gamma \equiv \iota \\ (iii) & (\mathbf{A}\mathbf{E}) \sim_{\mathcal{H}} (\mathbf{B}\mathbf{E}) \text{ for all } \mathbf{E} \in \mathit{wff}_\alpha(\Sigma) & \text{if } \gamma \equiv \alpha \to \beta \end{cases}$$

2. If $\Upsilon_\gamma(\mathbf{A}, \mathbf{B}) > 0$, i.e. $\mathbf{A} \equiv (\mathbf{F}_{\beta\to\gamma}\mathbf{C}_\beta), \mathbf{B} \equiv (\mathbf{G}_{\beta\to\gamma}\mathbf{D}_\beta)$ and $\mathbf{C}_\beta \not\equiv \mathbf{D}_\beta$, then

$$\mathbf{A}_\gamma \sim_{\mathcal{H}} \mathbf{B}_\gamma \quad \text{iff} \quad \mathbf{F} \sim_{\mathcal{H}} \mathbf{G} \quad \text{and} \quad \mathbf{C} \sim_{\mathcal{H}} \mathbf{D}$$

To see that this $\dot\sim_{\mathcal{H}}$ is well-defined we observe that the different cases are mutually independent and that all possible cases are covered. To convince ourselves that the recursion terminat es we define the following termination order: $(\mathbf{A}_\gamma, \mathbf{B}_\gamma) \lhd (\mathbf{A}'_\beta, \mathbf{B}'_\beta)$, iff $(\Upsilon_\gamma(\mathbf{A}, \mathbf{B}), \mathbf{ord}(\gamma)) <_{lex} (\Upsilon_\beta(\mathbf{A}', \mathbf{B}'), \mathbf{ord}(\beta))$, where $<_{lex}$ is the lexicographical order on tupels of numerals and $\mathbf{ord}(\beta)$ refers to the order of type β as, for instance, defined in [Bar84] (if $\alpha \in \mathcal{BT}$ then order-of$(\alpha) \equiv 1$; if $\alpha \equiv \beta \to \gamma$ then order-of$(\alpha) \equiv \max(\text{order-of}(\beta) + 1, \text{order-of}(\gamma))$. With $\lhd$ we can easily verify termination since both recursive calls are decreasing with respect to this order; case (iii) leaves the first ordering criterion constant and decreases the second, which is the order of the terms to be compared, and case (iii) decreases the first ordering criterion, which is the number of recursively embedded arguments which are functionally comparable but not identical.

Lemma 2.2.30 (Properties of $\sim_{\mathcal{H}}$)
If Γ_Σ is an abstract consistency class and $\mathcal{H}$ be a Hintikka set for Γ_Σ, then $\sim_{\mathcal{H}}$ is a functional congruence relation.

Proof: We show the assertion by induction on $n \equiv \Upsilon_\gamma(\mathbf{A}, \mathbf{B})$. In case (iii) we employ an encapsulated induction on the order of type γ. *model existence theorem*

Base case ($n \equiv 0$): We have to distinguish three cases with respect to γ. Note that since $n \equiv 0$, i.e. all functionally comparable arguments in $\mathbf{A}$ and $\mathbf{B}$ are already identical, functionality needs not to be adressed here[15]. Hence it remains to validate reflexivitiy, symmetry, and transitivity. EdNote(15) In case (i) and (ii) where $\gamma \in \{o, \iota\}$ the assertion is trivial. In case (iii) where $\gamma \equiv \alpha \to \beta$ we employ an induction on the order k of type γ. The base case is ($k \equiv 1$) and we know that $\mathbf{A}_\gamma \equiv \mathbf{B}_\gamma$ iff $(\mathbf{AE})_\beta \equiv (\mathbf{BE})_\beta$ for all $\mathbf{E} \in \mathit{wff}_\alpha$. Since β must be in $\{o, i\}$ we get reflexivity, symmetry, and transitivity by cases (i) and (ii) from above. In the step case ($k > 1$) we get the assertion by the induction hypotheses (if $(\mathbf{AE})$ and $(\mathbf{BE})$ still have functional type) or by cases (i) and (ii) (if we end up with a base type). This concludes the base case of our outer induction proof.

Step case ($n > 0$): Here $\mathbf{A} \equiv (\mathbf{F}_{\beta\to\gamma}\mathbf{C}_\beta)$ and $\mathbf{B} \equiv (\mathbf{G}_{\beta\to\gamma}\mathbf{D}_\beta)$. Reflexivity, symmetry, and transitivity follow immediately by the induction hypotheses. Functionality is obvious as this is exactly what is expressed here. $\square$ EndNP(14)

Remark 2.2.31 Note that in Lemma 2.1.16 $\mathrm{EXT}_L^{\alpha\to\beta}$ does not hold for $\doteq$ and hence $\dot\sim_{\mathcal{H}}$ is not a functional congruence in case Γ_Σ is not at least an $\mathfrak{Acc}_{\beta\mathfrak{q}}$. Hence $\dot\sim_{\mathcal{H}}$ is unsuitable for the model construction of an $\mathcal{M} \in \mathfrak{M}_{\beta\eta\mathfrak{b}}$ (or $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{b}}$) from a given $\mathcal{M}' \in \mathfrak{M}_{\beta\eta}$ (or $\mathcal{M}' \in \mathfrak{M}_\beta$) as demonstrated below but fits well for the construction of an $\mathcal{M} \in \mathfrak{M}_{\beta\mathfrak{qb}}$. Fortunately the relation $\sim_{\mathcal{H}}$ is already a functional congruence in case Γ_Σ is an $\mathfrak{Acc}_{\beta\eta}$.

We now use the Σ-Hintikka sets, guaranteed by Lemma 2.2.25, to construct a Σ-valuation for the Σ-term structure that turns it into the desired model $\mathcal{M}$.

Theorem 2.2.32 (Model Existence Theorem)
Let Γ_Σ be a saturated $\mathfrak{Acc}$ and let $H \in \Gamma_\Sigma$ be a sufficiently Σ-pure set of sentences. For all $i \in \{\beta, \beta\eta, \beta\eta\mathfrak{b}, \beta\mathfrak{q}, \beta\mathfrak{qb}, \beta\mathfrak{e}, \beta\mathfrak{eb}\}$ (cf. Def. 2.1.5, 2.1.8 and 2.1.18) we have: If Γ_Σ is an $\mathfrak{Acc}_i$ (cf. Def. 2.2.7 and 2.2.19), then there exists a countable model $\mathcal{M} \in \mathfrak{M}_i$ that satisfies H.

Proof: Let Γ_Σ be an abstract consistency class. We can assume without loss of generality (cf. Lemma 2.2.13) that Γ_Σ is compact, so the preconditions of 2.2.25 are met, and therefore there exists a Σ-Hintikka set $\mathcal{H} \subseteq \mathit{wff}_o(\Sigma)$ for Γ_Σ, such that $H \subseteq \mathcal{H}$.

Now, for each different kind of abstract consistency class, we will construct a countable model $\mathcal{M}^{\mathcal{H}}$ of the corresponding type. These model

[15] EdNote: I do not understand this

$\mathfrak{Acc}_*$ *abstract consistency class*
$\mathfrak{Acc}_*$ *abstract consistency class*
$\mathfrak{Acc}_*$ *abstract consistency class*

constructions closely reflect the relations of the different model types as discussed in Section 2.1 and shown in Figures 2.1 and 2.2. We start with the construction of an $\mathcal{M}_1^{\mathcal{H}} \in \mathfrak{M}_\beta$ and an $\mathcal{M}_2^{\mathcal{H}} \in \mathfrak{M}_{\beta\eta}$ based upon the non-functional term structure $\mathcal{TS}(\Sigma)^\beta$ and the functional $\mathcal{TS}(\Sigma)^{\beta\eta}$. The remaining model constructions are then based upon these two basic constructions.

$\mathfrak{M}_\beta$ Let Γ_Σ be an $\mathfrak{Acc}_\beta$. Given the Σ-Hintikka set $\mathcal{H}$ with $H \subseteq \mathcal{H}$ from above, we choose $\upsilon(\mathbf{C}) := \mathtt{T}$, iff $\mathbf{C} \in \mathcal{H}$. Note that we have $\upsilon(\mathbf{C}) := \mathtt{F}$, iff $\neg\mathbf{C} \in \mathcal{H}$ by $\overline{\nabla}_c^2$. By $\overline{\nabla}_\beta$ we know that υ is well-defined on $cwff(\Sigma){\downarrow}_\beta$ and by $\overline{\nabla}_c^2$, $\overline{\nabla}_c^3$ we have that υ is a total function on $\mathcal{TS}_o(\Sigma)^\beta$.

Furthermore by $\overline{\nabla}_c^2$, $\overline{\nabla}_c^3$, $\overline{\nabla}_\vee$ and $\overline{\nabla}_\forall$ we have that υ is a Σ-valuation of the Σ-term structure $\mathcal{TS}(\Sigma)^\beta$ and thus $\mathcal{M}_1^{\mathcal{H}} := (\mathcal{TS}(\Sigma)^\beta, \upsilon)$ is a Σ-model by construction. We have $\mathcal{M}_1^{\mathcal{H}} \models H$, since $H \subseteq \mathcal{H}$. Note that $\mathcal{M}_1^{\mathcal{H}}$ is indeed countable, since the sets of well-typed formulae are countable.

$\mathfrak{M}_{\beta\eta}$ Let Γ_Σ be an $\mathfrak{Acc}_{\beta\eta}$ and hence also an $\mathfrak{Acc}_\beta$. Analogous to the previous case we construct the countable Σ-model $\mathcal{M}_2^{\mathcal{H}} := (\mathcal{TS}(\Sigma)^{\beta\eta}, \upsilon)$ with $\mathcal{M}_2^{\mathcal{H}} \models H$. Note that in this case υ is well-defined on $\mathcal{TS}_o(\Sigma)^{\beta\eta}$ because of $\overline{\nabla}_\eta$. By Lemma 1.4.21 we know that $\mathcal{M}_2^{\mathcal{H}}$ is functional and hence $\mathcal{M}_2^{\mathcal{H}}$ is in $\mathfrak{M}_{\beta\eta}$.

We proceed with the construction of an $\mathcal{M}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{q}}$ and $\mathcal{M}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{b}}$ based upon the previous construction of an $\mathcal{M}_1^{\mathcal{H}} \in \mathfrak{M}_\beta$ and accordingly of an $\mathcal{M}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{q}}$ and an $\mathcal{M}^{\mathcal{H}} \in \mathfrak{M}_{\beta\eta\mathfrak{b}}$ based upon an $\mathcal{M}_2^{\mathcal{H}} \in \mathfrak{M}_{\beta\eta}$. Thus we start out with a countable Σ-model $\mathcal{M}_1^{\mathcal{H}} := (\mathcal{TS}(\Sigma)^\beta, \upsilon)$ or $\mathcal{M}_2^{\mathcal{H}} := (\mathcal{TS}(\Sigma)^{\beta\eta}, \upsilon)$, such that $\mathcal{M}_i^{\mathcal{H}} \models H$, for $i = 1, 2$. Property $\mathfrak{q}$ is easy to verify, as it follows from the properties discussed in the Hintikka-lemmata 2.2.15 and 2.2.17.

$\mathfrak{M}_{\beta\mathfrak{q}}$ Let Γ_Σ be an $\mathfrak{Acc}_{\beta\mathfrak{q}}$. From $\overline{\nabla}_{\doteq}^r$, $\overline{\nabla}_{\doteq}^{sy}$, $\overline{\nabla}_{\doteq}^{tr}$, $\overline{\nabla}_{\mathfrak{q}}^-$ and $\overline{\nabla}_{\mathfrak{q}}^+$ we can derive that $\doteq^\alpha$ is indeed the q^α required by property $\mathfrak{q}$ and hence $\mathcal{M}^{\mathcal{H}}$ is a countable Σ-model in $\mathfrak{M}_{\beta\mathfrak{q}}$.

To verify property $\mathfrak{b}$ instead we have to construct an $\mathcal{M}^{\mathcal{H}}$ from $\mathcal{M}_i^{\mathcal{H}}$ $(i = 1, 2)$ by reducing the set of truth values to $\{\mathtt{T}, \mathtt{F}\}$, which can be done with the help of a functional congruence relation.

$\mathfrak{M}_{\beta\mathfrak{b}}$,$\mathfrak{M}_{\beta\eta\mathfrak{b}}$ Let Γ_Σ be an $\mathfrak{Acc}_{\beta\mathfrak{q}}$ or $\mathfrak{Acc}_{\beta\eta\mathfrak{b}}$. By Lemma 2.2.30, $\overline{\nabla}_{\mathfrak{b}}^+$ and $\overline{\nabla}_c^3$ we can show that the relation $\sim_{\mathcal{H}}$ defined in 2.2.29 is a functional Σ-congruence for $\mathcal{M}_i^{\mathcal{H}}$ and thus, by Lemma 2.1.11, the quotient structure $\mathcal{M}_i^{\mathcal{H}}/_{\sim_{\mathcal{H}}}$ is a functional Σ-model that satisfies H. From $\overline{\nabla}_c^2$, $\overline{\nabla}_c^3$ and the choice of υ we conclude that $\sim_{\mathcal{H}}$ has exactly two equivalence classes on $\mathcal{TS}_o(\Sigma)^{\beta\eta}$. Thus we have $\mathcal{D}_o \equiv \{\mathtt{T} := [\![\mathbf{T}_o]\!]_\sim, \mathtt{F} := [\![\mathbf{F}_o]\!]_\sim\}$, if we define $\mathbf{T}_o$ and $\mathbf{F}_o$ as in Lemma 2.1.2. Using $\overline{\nabla}_t$ and $\overline{\nabla}_c^2$ we further get that

υ is the identity relation. Finally note that $\mathcal{M}_i^{\mathcal{H}}/_{\sim_{\mathcal{H}}}$ is countable since $\mathcal{M}_i^{\mathcal{H}}$ is.

$\mathfrak{Acc}_*$ *abstract consistency class*
$\mathfrak{Acc}_*$ *abstract consistency class*
$\mathfrak{Acc}_*$ *abstract consistency class*
$\mathfrak{Acc}_*$ *abstract consistency class*

We finish the constructions for the cases without a primitive notion of equality with the construction of a Σ-Henkin model $\mathcal{N}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{qb}}$ in case we are considering an $\mathfrak{Acc}_{\beta\mathfrak{qb}}$.

We start with $\mathcal{M}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{q}}$ guaranteed by the discussion above. Analogous to the construction of an $\mathcal{M}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{b}}$, we make use of a functional congruence relation in order to construct a quotient model which fulfills property $\mathfrak{b}$. But instead of the relation $\sim_{\mathcal{H}}$ we had to use before, we apply the simpler relation $\dot{\sim}_{\mathcal{H}}$ which is a functional congruence relation for the elements of $\mathfrak{M}_{\beta\mathfrak{qb}}$.

$\mathfrak{H} \equiv \mathfrak{M}_{\beta\mathfrak{qb}}$ By Lemma 2.2.30 and $\overline{\nabla}_{\mathfrak{b}}^{+}$,$\overline{\nabla}_{c}^{3}$ we know that the relation $\dot{\sim}_{\mathcal{H}}$ is a functional Σ-congruence for $\mathcal{M}^{\mathcal{H}}$, so the quotient structure $\mathcal{N}^{\mathcal{H}} := \mathcal{M}^{\mathcal{H}}/_{\dot{\sim}_{\mathcal{H}}} \in \mathfrak{M}_{\beta\mathfrak{q}}$ with $\mathcal{N}^{\mathcal{H}} \models H$ by Lemma 2.1.11. Now we can conclude that $\mathcal{D}_o \equiv \{\mathtt{T}, \mathtt{F}\}$ with the same argumentation as in the $\mathfrak{M}_{\beta\eta\mathfrak{b}}$ case.

It remains to discuss the cases with primitive equality and we start with the $\mathcal{M}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{q}}$, resp. $\mathcal{M}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{qb}}$ from above.

$\mathfrak{M}_{\beta\mathfrak{e}}$ Let Γ_Σ be an $\mathfrak{Acc}_{\beta\mathfrak{e}}$ and hence an $\mathfrak{Acc}_{\beta\mathfrak{q}}$. We construct the countable $\mathcal{E}^{\mathcal{H}} \in \mathfrak{M}_{\beta\mathfrak{q}}$ with $\mathcal{E}^{\mathcal{H}} \models H$ as discussed above. It remains to show that property $\mathfrak{e}$ is valid for $\mathcal{E}^{\mathcal{H}}$ which follows from property $\mathfrak{q}$ by $\overline{\nabla}_{=}^{\dot{=}+}$ and $\overline{\nabla}_{=}^{\dot{=}-}$.

$\mathfrak{M}_{\beta\mathfrak{eb}}$ This case is analogous: Let Γ_Σ be an $\mathfrak{Acc}_{\beta\mathfrak{eb}}$ and hence an $\mathfrak{Acc}_{\beta\mathfrak{qb}}$. We construct a countable $\mathcal{E}^{\mathcal{H}} := (\mathcal{TS}(\Sigma)^{\beta\eta}, \upsilon) \in \mathfrak{H}$ with $\mathcal{E}^{\mathcal{H}} \models H$. Again property $\mathfrak{e}$ is valid for $\mathcal{E}^{\mathcal{H}}$ by property $\mathfrak{q}$, $\overline{\nabla}_{=}^{\dot{=}+}$, and $\overline{\nabla}_{=}^{\dot{=}-}$. □

This proof concludes the section on model existence theorems. We have proven a set of proof-theoretic conditions that is sufficient to guarantee the existence of a model. The next step will be to apply this method to particular abstract consitency classes, namely those that are generated by logical calculi for higher-order logic. The result of this investigation will be compleness results for these calculi.

2.3 Classical Calculi for Higher-Order Logic

In this section we introduce the syntactic counterparts of the entailment relation and fix the formalism for the calculi $\mathfrak{T}$, $\mathfrak{T}_\eta$, $\mathfrak{T}_{\eta\mathfrak{b}}$, and $\mathfrak{T}_{\mathfrak{e}}$ which are simple and intuitive generalizations of a commonly used Hilbert-style calculus for first-order logic.

Henkin
$\Phi \Vdash_{\mathcal{C}} \mathbf{A}$ *derivation of* $\mathbf{A}$ *from* Φ *in a calculus* $\mathcal{C}$
$\Vdash_{\mathcal{C}} \mathbf{A}$ $\mathbf{A}$ *is a* $\mathcal{C}$*-theorem*
calculus
provable
proof
theorem
sound
complete
refutation complete
consistent
inconsistent
consistent with
absolutely consistent
consistency wrt. negation

The calculus $\mathfrak{T}_{\eta\mathfrak{b}}$ was in troduced by Leon Henkin in [Hen50] and proven Henkin-complete there. Peter Andrews generalized his argument to a model existence theorem for υ-complexes in [And71], which he used to prove $\mathfrak{T}$ complete for this class.

In the literature [And71, Hue72, Mil83] the deductive power of machine-oriented calculi is determined relative to these calculi. We will now use the semantic tools developed above to re-obtain the completeness results mentioned above and characterize the deductive power in terms of our semantics.

Definition 2.3.1 (Calculus)
A **calculus** $\mathcal{C}$ is an inference system for the judgment $\Vdash \mathbf{A}$, i.e. $\mathbf{A}$ is **provable** in $\mathcal{C}$. Since $\Vdash$ is a unary relation, it is customary to drop it, and to consider inference rules of a calculus $\mathcal{C}$ just as relations over propositions.

Let $\mathbf{A}$ be a proposition and Φ be a set of sentences. Using the nomenclature for inference systems we call a $\mathcal{C}$-derivation $\mathcal{D}$ a $\mathcal{C}$-**proof** of $\mathbf{A}$ from the set Φ of hypotheses, if $\mathbf{A}$ is the assertion of the root of $\mathcal{D}$, and the supports of the leaves of $\mathcal{D}$ are subsets of Φ. If there exists a $\mathcal{C}$-derivation of $\mathbf{A}$ from Φ, then we write $\Phi \Vdash_{\mathcal{C}} \mathbf{A}$. Let $\mathcal{C}$ be a calculus, then a proposition $\mathbf{A}$ is called a $\mathcal{C}$-**theorem**, iff there exists a $\mathcal{C}$-derivation of $\mathbf{A}$ from the empty set of hypotheses.

We will now review some fundamental properties of calculi.

Definition 2.3.2 (Sound, Complete, Saturated)
Let $\mathcal{C}$ be a calculus and $\mathcal{K}$ a class of Σ-models, then $\mathcal{C}$ is called

1. **sound** with respect to $\mathcal{K}$, iff each $\mathcal{C}$-theorem is valid in $\mathcal{K}$.

2. **complete** with respect to $\mathcal{K}$, iff each valid sentence in $\mathcal{K}$ is a $\mathcal{C}$-theorem.

3. **refutation complete** with respect to $\mathcal{K}$, iff each $\mathcal{K}$-unsatisfiable sentence $\mathbf{A}$ can be refuted in $\mathcal{C}$, i.e. there is a $\mathcal{C}$-derivation of an elementary contradiction from $\mathbf{A}$.

Definition 2.3.3 ($\mathcal{C}$-Consistent)
Let $\mathcal{C}$ be a calculus, then a set Φ of propositions is called $\mathcal{C}$-**consistent**, iff there is a formula $\mathbf{A}$, such that $\Phi \nVdash_{\mathcal{C}} \mathbf{A}$, and $\mathcal{C}$-**inconsistent** otherwise. We call a set Ψ $\mathcal{C}$-**consistent with** a set Φ, iff $\Phi \cup \Psi$ is $\mathcal{C}$-consistent.

We will now relate the notion of consistency defined above (which is sometimes called **absolutely consistent**) with the often used notion of consistency by elementary contradiction, which is sometimes called **consistency wrt. negation**. For this, we will have to assume certain minimal properties of calculi, which we will collect under the label of a "reasonable calculus". Of course all calculi discussed in this book are reasonable.

reasonable
$\mathfrak{T}_*$ higher-order calculi
Andrews

Definition 2.3.4 (Reasonable Calculus)
We will call a calculus $\mathcal{C}$ **reasonable**, iff it is sound and complete for propositional logic, enjoys a deduction theorem i.e. if $\mathcal{H}, \mathbf{A} \Vdash_{\mathcal{C}} \mathbf{B}$, then $\mathcal{H} \Vdash_{\mathcal{C}} \mathbf{A} \Rightarrow \mathbf{B}$. and admits universal instantiation and $\forall X_o. \forall Y_o. \neg X \Rightarrow .X \Rightarrow Y$. In other words: modus ponens is an admissible rule of inference (cf. 1.5.3) and the universal formula is a $\mathcal{C}$-theorem.

Lemma 2.3.5 (Inconsistency and Contradiction)
Let $\mathcal{C}$ be a reasonable calculus, then a set Φ is $\mathcal{C}$-inconsistent, iff there is a formula $\mathbf{A} \in \mathit{wff}_o(\Sigma)$ with $\Phi \Vdash_{\mathcal{C}} \mathbf{A} \wedge \neg\mathbf{A}$

Proof: Let $\Phi \Vdash_{\mathcal{C}} \mathbf{A} \wedge \neg\mathbf{A}$, and $\mathbf{B} \in \mathit{wff}_o(\Sigma)$ an arbitrary proposition, then by two applications of modus ponens and univeral instantiation, we obtain $\Phi \Vdash_{\mathcal{C}} \mathbf{B}$, so Φ is absolutely inconsistent. Conversely, if Φ is (absolutely) inconsistent, then $\Phi \Vdash_{\mathcal{C}} \mathbf{A}$ and $\Phi \Vdash_{\mathcal{C}} \neg\mathbf{A}$, so Φ is inconsistent with respect to negation. □

In particular we see that the choice of the elementary contradictory formula $\mathbf{A} \wedge \neg\mathbf{A}$ is not important.

Definition 2.3.6 (The Calculi $\mathfrak{T}$, $\mathfrak{T}_\eta$ and $\mathfrak{T}_{\eta\mathfrak{b}}$)
The calculus $\mathfrak{T}$ consists of the inference rules in Figure 2.4. We obtain the calculus $\mathfrak{T}_\eta$ by adding the η-conversion rule $\mathfrak{T}(\eta)$ and finally, $\mathfrak{T}_{\eta\mathfrak{b}}$ with the rule $\mathfrak{T}(\mathfrak{b})$ for full extensionality (see Figure 2.5). These rules correspond to the $\mathrm{EXT}_L^{\alpha\to\beta}$ and EXT_L^{o}[16] axioms. We collectively denote these calculi with $\mathfrak{T}_*$.[17] EdNote(16) EdNote(17)

Note that if $\Vdash_\Sigma^{\mathcal{T}} \mathbf{A}$, then we have $\mathbf{A}\colon o$, since we have taken care to require this for the propositional axioms, and the proper inference rules conserve this property.

Remark 2.3.7 (Choice of Logical Constants)
When defining a higher-order logic one has the choice, either to assume the the logical connectives and quantifiers as primitive and to define the semantics of equality via the Leibniz formula (see 1.2.6 and 2.1.1) or to take equality as primitive, and to define the logical constants from it (see Figure 1.1). We have taken the first approach here with the $\mathfrak{T}_*$ calculi. An alternative would have been to take second approach and assume the inference rule $\mathfrak{T}(\mathfrak{e})$ in Figure 2.5. Peter Andrews takes this approach in his book on higher-order logic [And86]. He takes the equality constants from $\Sigma^=$ as the only primitive logical constants (cf. 1.2.1) and views all $\mathit{wff}(\Sigma^{\mathcal{HOL}})$ formulae as abbreviations of these via the definitions in Figure 1.1. Consequently, he only needs a calculus that treats equality: the calculus $\mathfrak{T}_\mathfrak{e}$

[16] EdNote: Chad: $\mathfrak{T}_\mathfrak{q}$ corresponds to $\mathrm{EXT}_L^{\alpha\to\beta}$ correct?
[17] EdNote: What about the calculi for $\mathfrak{q}, \mathfrak{e}$, etc?

Leibniz
Leibniz equality
equality!Leibniz

$$\frac{}{\Vdash_\Sigma^{\iota} (\mathbf{P}_o \vee \mathbf{P}) \Rightarrow \mathbf{P}} \qquad \frac{}{\Vdash_\Sigma^{\iota} \mathbf{P}_o \Rightarrow (\mathbf{P} \Rightarrow \mathbf{P})}$$

$$\frac{}{\Vdash_\Sigma^{\iota} (\mathbf{P}_o \vee \mathbf{Q}_o) \Rightarrow (\mathbf{Q} \vee \mathbf{P})}$$

$$\frac{}{\Vdash_\Sigma^{\iota} (\mathbf{P}_o \Rightarrow \mathbf{Q}_o) \Rightarrow .(\mathbf{R}_o \vee \mathbf{P}) \Rightarrow (\mathbf{R} \vee \mathbf{Q})}$$

$$\frac{}{\Vdash_\Sigma^{\iota} \Pi^\alpha \mathbf{A}_{\alpha\to o} \Rightarrow \mathbf{A}X_\alpha} \qquad \frac{X \notin \mathbf{Free}(\mathbf{A})}{\Vdash_\Sigma^{\iota} \forall X_\alpha.(\mathbf{A}_o \vee \mathbf{B}_{\alpha\to o}X) \Rightarrow .\mathbf{A} \vee \Pi^\alpha \mathbf{B}}$$

$$\frac{\mathbf{A} =_\beta \mathbf{B} \quad \Vdash_\Sigma^{\iota} \mathbf{A}}{\Vdash_\Sigma^{\iota} \mathbf{B}}\,\mathfrak{T}(\beta) \qquad \frac{\Vdash_\Sigma^{\iota} \mathbf{A} \Rightarrow \mathbf{B} \quad \Vdash_\Sigma^{\iota} \mathbf{A}}{\Vdash_\Sigma^{\iota} \mathbf{B}}\,\mathfrak{T}(MP)$$

$$\frac{\Vdash_\Sigma^{\iota} \mathbf{A}X}{\Vdash_\Sigma^{\iota} \mathbf{A}\mathbf{B}}\,\mathfrak{T}(Subst) \qquad \frac{\Vdash_\Sigma^{\iota} \mathbf{A}X}{\Vdash_\Sigma^{\iota} \Pi^\alpha \mathbf{A}}\,\mathfrak{T}(UG)$$

Figure 2.4: Inference rules for $\mathfrak{T}$

only consists of $\mathfrak{T}(\mathfrak{e})$ and $\mathfrak{T}(\beta)$. In [And86], he and shows that the $\mathfrak{T}$ rules are admissible in $\mathfrak{T}_\mathfrak{e}$.

On the other hand, the $\mathfrak{T}(\mathfrak{e})$ rule is clearly admissible if we interpret equality as Leibniz equality, so the choice of logical constants is immaterial for our purposes.

In the rest of this section, we will use our more refined model existence theorems to analyze soundness and completeness of the $\mathfrak{T}_*$ calculi. Since most of the results hold for all calculi, we will use $\mathfrak{T}_*$ with the understanding that $* \in \{\eta, \mathfrak{b}\}^*$.

$$\frac{\mathbf{A} =_\eta \mathbf{B} \quad \Vdash_\Sigma^{\iota} \mathbf{A}}{\Vdash_\Sigma^{\iota} \mathbf{B}}\,\mathfrak{T}(\eta) \qquad \frac{\Vdash_\Sigma^{\iota} \mathbf{A} \Leftrightarrow \mathbf{B}}{\Vdash_\Sigma^{\iota} \mathbf{A} = \mathbf{B}}\,\mathfrak{T}(\mathfrak{b})$$

$$\frac{\Vdash_\Sigma^{\iota} \forall X_\alpha.\mathbf{F}X \doteq^\beta \mathbf{G}X}{\Vdash_\Sigma^{\iota} \mathbf{F} \doteq^{\alpha\to\beta} \mathbf{G}}\,\mathfrak{T}(\mathfrak{q})$$

$$\frac{\Vdash_\Sigma^{\iota} \mathbf{A}[\mathbf{C}]_p \quad \Vdash_\Sigma^{\iota} \mathbf{C} = \mathbf{D}}{\Vdash_\Sigma^{\iota} \mathbf{A}[\mathbf{D}]_p}\,\mathfrak{T}(\mathfrak{e})$$

Figure 2.5: Extensions of $\mathfrak{T}$

tautology

Theorem 2.3.8 (Soundness)
$\mathfrak{T}_*$ is sound for $\mathfrak{M}_*$.

Proof: The validity of the propositional axioms and the inference rules $\mathfrak{T}(MP)$ and $\mathfrak{T}(UG)$ is an immediate consequence of the definition of Σ-valuations. For the $\mathfrak{T}(Subst)$ rule let $\Gamma, [X{:}\alpha] \vdash_\Sigma \mathbf{A}{:}o$, $\Gamma' \vdash_\Sigma \mathbf{B}{:}\alpha$ and $\mathcal{M} \models \mathbf{A}$, then by the substitution value property (1.4.12.4) we have $\upsilon(\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A})) = \upsilon(\mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{A})) = \mathtt{T}$ for all Γ-assignments φ. $\mathfrak{T}(\beta)$ is sound all Σ-structures and therefore in all Σ-models by Lemma 1.4.23. Thus all inference rules of $\mathfrak{T}$ are sound in the class $\mathfrak{M}$ of Σ-models.

$\mathfrak{T}(\eta)$ is sound in functional Σ-structures and therefore in $\mathfrak{M}_{\beta\eta}$ by Lemma 1.4.28. Finally, soundness of $\mathfrak{T}(\mathfrak{b})$ is a consequence of Theorem 1.4.24. □

Moreover, we have seen in section 2.1 that the characterizations above are sharp. We now present a some results on $\mathfrak{T}_*$-provability, which simplify the completeness proof for the $\mathfrak{T}_*$ calculi.

Lemma 2.3.9 *Let $\mathcal{M} = (\mathcal{D}, @, \mathcal{I}, \upsilon) \in \mathfrak{M}_*$, such that $\mathcal{V}_\varphi(\mathbf{B}) = \mathtt{F}$. If $\mathbf{A}^1, \ldots, \mathbf{A}^n \Vdash_{\mathfrak{T}_*} \mathbf{B}$, then $\mathcal{M} \models \neg\mathbf{A}^1 \vee \ldots \vee \neg\mathbf{A}^n$.*

Proof: We show the assertion by induction on the structure of the $\mathfrak{T}_*$-derivation $\mathcal{D}{:}\,\Phi \Vdash_\Sigma^{\mathcal{C}} \mathbf{B}$. In the base (axiom) case, $\mathbf{B}$ must be an instance of an axiom of $\mathfrak{T}_*$, which cannot be, since $\mathbf{B}$ is false in $\mathcal{M}$ and $\mathfrak{T}_*$ is sound for $\mathfrak{M}_*$, thus the base case is vacuously true. For the inductive case let and let $\mathcal{D}$ end in an application of the rule $\mathcal{R} := \mathbf{A}^1, \ldots, \mathbf{A}^n \Vdash \mathbf{B} \in \mathfrak{T}_*$. Since $\mathfrak{T}_*$ is sound with respect to $\mathfrak{M}_*$, we have $\mathcal{V}_\varphi(\mathbf{A}^1, \ldots, \mathbf{A}^n \Rightarrow \mathbf{B}) = \mathtt{T}$ by $\mathfrak{T}(MP)$, and therefore $\mathcal{V}_\varphi(\neg\mathbf{A}^1 \vee \ldots \vee \neg\mathbf{A}^n) = \mathtt{T}$. Therefore one of the premises of $\mathcal{R}$ is false in $\mathcal{M}$, and by induction we get the assertion. □

Lemma 2.3.10 *Let $\mathcal{M} \in \mathfrak{M}_*$, then $\Psi(\mathcal{M}) := \{\mathbf{A} \in wff_o(\Sigma) \mid \mathcal{M} \models \mathbf{A}\}$ is $\mathfrak{T}_*$-consistent.*

Proof: We prove the contrapositive: let $\Psi(\mathcal{M})$ be $\mathfrak{T}_*$-inconsistent, then there must be a finite $\mathfrak{T}_*$-inconsistent subset $\Phi := \{\mathbf{A}^1, \ldots, \mathbf{A}^n\}$ of $\Psi(\mathcal{M})$, then $\mathbf{A}^1, \ldots, \mathbf{A}^n \Vdash_{\mathfrak{T}_*} \mathbf{F}_o$, and therefore one of the $\mathbf{A}^i$ is false in $\mathcal{M}$ by the first assertion, which contradicts the assumption. □

Definition 2.3.11 (Tautology)
We call a proposition $\mathbf{A}$ a **tautology**, iff it is a substitution instance of a proposition $\mathbf{P}$ that only contains logical connectives and variables of type o, and is valid in all Σ-models.

Note that the validity of $\mathbf{P}$ only depends on the assignment for propositional variables in $\mathbf{P}$.

Lemma 2.3.12 (Rule P)
If $\mathcal{C}$ is a reasonable calculus and $\mathbf{A}$ is a tautology, then $\Vdash_{\mathcal{C}} \mathbf{A}$.

saturated

Proof: Let $\mathbf{P}$ be the proposition such that $\mathbf{A} = \sigma(\mathbf{P})$, and $\mathbf{P}$ only contains propositional variables and connectives. By definition $\mathcal{C}$ is complete for propositional logic, so there is a $\mathfrak{T}_*$-proof $\mathcal{D}$: $\Vdash_{\mathfrak{T}_*} \mathbf{P}$. One application of the substitution rule now gives the assertion. $\square$

Theorem 2.3.13 *$\mathfrak{T}_*$ are reasonable.*

Proof: It is well-known that the inference rules in Figure 2.4 are sound and complete for propositional logic. Modus ponens and universal instantiation are inference rules, and the universal formula is easy to prove. The proof of the deduction theorem is a simple induction over the structure of the $\mathfrak{T}_*$-proof involved, we refer to lemma 5240 in [And86] and to [Hen50] for a proof. $\square$

Definition 2.3.14 (Saturated)
We will call a calculus **saturated**, iff for all $\mathcal{C}$-consistent sets Φ of propositions and any proposition $\mathbf{A}$ we have $\Phi * \mathbf{A}$ is $\mathcal{C}$-consistent or $\Phi * \neg\mathbf{A}$ is $\mathcal{C}$-consistent.

Lemma 2.3.15 *Reasonable calculi are saturated.*

Proof: Let $\mathcal{C}$ be a reasonable calculus, $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ be $\mathcal{C}$-inconsistent, then we show that Φ is $\mathcal{C}$-inconsistent, and obtain the assertion. By definition we have $\Phi * \mathbf{A} \Vdash_{\mathcal{C}} \mathbf{F}_o$, and by the deduction theorem $\Phi \Vdash_{\mathcal{C}} \mathbf{A} \Rightarrow \mathbf{F}_o$, and thus $\Phi \Vdash_{\mathcal{C}} \neg\mathbf{A}$ by "rule P", similarly $\Phi \Vdash_{\mathcal{C}} \neg\mathbf{A} \Rightarrow \mathbf{F}_o$ and thus $\Phi \Vdash_{\mathcal{C}} \mathbf{F}_o$ by $\mathfrak{T}(MP)$. Thus Φ is $\mathcal{C}$-inconsistent by definition. $\square$

Lemma 2.3.16 *Let Φ be a set of propositions and $\mathcal{C}$ a reasonable calclus, then*

1. *Φ is $\mathcal{C}$-inconsistent with $\{\mathbf{A}^1, \ldots, \mathbf{A}^n\}$, iff $\Phi \Vdash_{\mathcal{C}} \neg\mathbf{A}^1 \vee \ldots \vee \neg\mathbf{A}^n$.*
2. *if Φ is $\mathcal{C}$-consistent and $\Vdash_{\mathcal{C}} \mathbf{C} \Rightarrow \mathbf{A}^1 \wedge \ldots \wedge \mathbf{A}^n$ for some $\mathbf{C} \in \Phi$ and $n \geq 1$, then $\Phi \cup \{\mathbf{A}^1, \ldots, \mathbf{A}^n\}$ is $\mathcal{C}$-consistent.*

Proof: The first assertion is a consequence Lemma 2.3.9: if Φ is $\mathfrak{T}_*$-inconsistent with $\mathbf{A}^1, \ldots, \mathbf{A}^n$, then $\Phi, \mathbf{A}^1, \ldots, \mathbf{A}^n \Vdash_{\mathfrak{T}_*} \neg\mathbf{F}_o$, for some $1 \leq i \leq n$, so $\Phi \Vdash_{\mathfrak{T}_*} \mathbf{A}^1 \Rightarrow \mathbf{\ldots} \Rightarrow \mathbf{A}^n \Rightarrow \neg\mathbf{F}_o$, and therefore $\Phi \Vdash_{\mathfrak{T}_*} \neg\mathbf{A}^1 \vee \ldots \vee \neg\mathbf{A}^n$.

This entails the second assertion: If $\Phi \cup \{\mathbf{A}^1, \ldots, \mathbf{A}^n\}$ must be $\mathfrak{T}_*$-inconsistent, then $\Phi \Vdash_{\mathfrak{T}_*} \mathbf{A}^1, \ldots, \mathbf{A}^n$ by 1. and $\Phi \Vdash_{\mathfrak{T}_*} \neg\mathbf{C}$ by rule P. Thus Φ is $\mathfrak{T}_*$-inconsistent, which contradicts the assumption. $\square$

Now, we use the model existence theorems for $\mathcal{HOL}$ to give short and elegant proofs of completeness for $\mathfrak{T}_*$.

Theorem 2.3.17 *The class $\Gamma^*_\Sigma := \{\Phi \subset \mathit{wff}_o(\Sigma) \mid \Phi$ is $\mathfrak{T}_*$-consistent$\}$ is an $\mathfrak{Acc}_{\mathfrak{M}_*}$*

Proof: Obviously Γ_Σ^* is closed under subsets, since any subset of a $\mathfrak{T}_*$-consistent set is $\mathfrak{T}_*$-consistent. Γ_Σ^* is saturated by 2.3.15.

∇_c By definition 2.3.3, the set $\{\mathbf{A}, \neg\mathbf{A}\}$ is $\mathfrak{T}_*$-inconsistent, since $\mathfrak{T}_*$ admits Rule P (cf 2.3.12) and $\mathbf{A} \wedge \neg\mathbf{A} \Rightarrow \mathbf{F}_o$ is tautologous; this establishes ∇_c as $\mathfrak{T}_*$-inconsistency is closed under supersets.

$\nabla_\neg$ Let Φ be $\mathfrak{T}_*$-consistent; $\neg\neg\mathbf{A} \Rightarrow \mathbf{A}$ is tautologous and thus $\neg\neg\mathbf{A} \in \Phi$ implies that $\Phi * \mathbf{A}$ is $\mathfrak{T}_*$-consistent by Lemma 2.3.16(2).

∇_β Let $\mathbf{A}{=_\beta}\mathbf{B}$, the inference rule $\mathfrak{T}(\beta)$ together with the deduction theorem 2.3.13 yield $\mathbf{A} \Rightarrow \mathbf{B}$, thus we can use Lemma 2.3.16(2) as in $\nabla_\neg$.

∇_η as in ∇_β, but with the additional rule $\mathfrak{T}(\eta)$ (this clause only holds in $\mathfrak{T}_\eta$ and $\mathfrak{T}_\mathfrak{e}$).

$\nabla_\vee$ If Φ is $\mathfrak{T}_*$-consistent, and $\Phi{*}\mathbf{A}$ and $\Phi{*}\mathbf{B}$ are both $\mathfrak{T}_*$-inconsistent, then $\Phi \Vdash_{\mathfrak{T}_*} \neg\mathbf{A}$ and $\Phi \Vdash_{\mathfrak{T}_*} \neg\mathbf{B}$, so $\Phi \Vdash_{\mathfrak{T}_*} \neg(\mathbf{A} \vee \mathbf{B})$ by rule P (cf. 2.3.12), therefore $(\mathbf{A} \vee \mathbf{B}) \notin \Phi$, which is just the contrapositive of $\nabla_\vee$

$\nabla_\wedge$ As $\nabla_\neg$, but with the tautologies $(\mathbf{A} \wedge \mathbf{B}) \Rightarrow \mathbf{A}$ and $(\mathbf{A} \wedge \mathbf{B}) \Rightarrow \mathbf{B}$.

$\nabla_\forall$ The inference rule $\mathfrak{T}(UG)$ together with the deduction theorem 2.3.13 yield $\Pi^\alpha\mathbf{A} \Rightarrow \mathbf{A}\mathbf{B}$, thus we can use Lemma 2.3.16(2) as in $\nabla_\neg$.

$\nabla_\exists$ Let $\Phi \subseteq \mathit{wff}_o(\Sigma)$ and X new. We assume that $\neg\Pi^\alpha\mathbf{A} \in \Phi$ and Φ is $\mathfrak{T}_*$-consistent, but $\Phi * \neg(\mathbf{A}X)$ is $\mathfrak{T}_*$-inconsistent. So there is a $\mathfrak{T}_*$-derivation $\mathcal{D}\colon \Phi \Vdash_{\mathfrak{T}_*} \mathbf{A}X$ by 2.3.16(1) and 2.3.12. By adding an application of $\mathfrak{T}(UG)$ the root of $\mathcal{D}'$ we obtain a $\mathfrak{T}_*$-derivation of $\Phi \Vdash_{\mathfrak{T}_*} \Pi\mathbf{A}$, which contradicts our assumption that Φ is $\mathfrak{T}_*$-consistent. Thus $\neg(\mathbf{A}X) \notin \Phi$.

$\nabla_\mathfrak{b}$ We show the contrapositive and assume that both $\Phi \cup \{\mathbf{A}, \neg\mathbf{B}\} \notin \Gamma_\Sigma^*$ and $\Phi \cup \{\neg\mathbf{A}, \mathbf{B}\} \notin \Gamma_\Sigma^*$. So we have $\Phi \Vdash_{\mathfrak{T}_*} \neg\neg\mathbf{A} \vee \neg\mathbf{B}$ and $\Phi \Vdash_{\mathfrak{T}_*} \neg\mathbf{A} \vee \neg\neg\mathbf{B}$ by Lemma 2.3.16(1), and $\Phi \Vdash_{\mathfrak{T}_*} \mathbf{B} \Rightarrow \mathbf{A}$ and $\Phi \Vdash_{\mathfrak{T}_*} \mathbf{A} \Rightarrow \mathbf{B}$ by Rule P. Thus we have $\Phi \Vdash_{\mathfrak{T}_*} \mathbf{B} \Leftrightarrow \mathbf{A}$, and $\Phi \Vdash_{\mathfrak{T}_*} \mathbf{B} \doteq^o \mathbf{A}$ by $\mathfrak{T}(\mathfrak{b})$. Thus $\Phi * \neg\mathbf{A} \doteq^o \mathbf{B}$ must be $\mathfrak{T}_*$-inconsitent, which yields the assertion.

$\nabla_\mathfrak{q}$ This is a direct consequence of the extensionality rule $\mathfrak{T}(\mathfrak{q})$. $\square$

Corollary 2.3.18 (Henkin's Theorem for $\mathfrak{T}_*$)
Every $\mathfrak{T}_*$-consistent set of sentences has a countable $\mathfrak{M}_*$-model.

Proof: Let Φ be a $\mathfrak{T}_*$-consistent set of sentencens. By Theorem 2.3.17 we know that the class of sets of $\mathfrak{T}_*$-consistent propositions constitute a saturated, $\mathfrak{Acc}_{\mathfrak{M}_*}$, thus the model existence theorem 2.2.32 guarantees a countable $\mathfrak{M}_*$ model for Φ. $\square$

Corollary 2.3.19 (Completeness Theorem for $\mathfrak{T}_*$)
We have $\mathbf{A} \Vdash_{\mathfrak{T}_*} \mathbf{B}$*, iff* $\mathbf{A} \models \mathbf{B}$ *in* $\mathfrak{M}_*$.

Remark 2.3.20 $\mathfrak{T}$ and $\mathfrak{T}_\eta$ are not complete with respect to Henkin models.

Proof sketch: We do not have the means of proving this lemma here, since the argumentation involves the use of the resolution calculus $\mathcal{HR}$ presented in chapter 6, so we only sketch the proof and refer to 6.5.9. Let $\mathbf{A} := (cb)$, and $\mathbf{B} := c(\neg\neg b)$, where c and b are constants of type $o \to o$ and o. The proof of the assertion has three ingredients:

- $\mathbf{C} := \neg\mathbf{A} \wedge \mathbf{B}$ is not $\mathcal{HR}$-refutable.
- $\mathfrak{T}_\eta$ is sound and refutation-complete for $\mathfrak{M}_{\beta\eta}$ (see 2.3.19).
- and $\mathcal{HR}$ is sound and complete for $\mathfrak{M}_{\beta\eta}$ (6.5.6).

$\neg\mathbf{C}$ cannot be derivable in $\mathfrak{T}_\eta$, since otherwise $\mathbf{C}$ could be refuted in $\mathcal{HR}$, as $\mathcal{HR}$ is refutation complete. The same argumentation holds for $\mathfrak{T}$, since it is weaker lacking the η-axiom of $\mathfrak{T}$. □

Remark 2.3.21 The completeness theorem 2.3.19 says that the formula $\mathbf{C}$ that was used to show incompleteness 2.3.20 of $\mathfrak{T}_\eta$ is a $\mathfrak{T}_{\eta\mathfrak{b}}$-theorem. Here we sketch the direct proof.

We have $\Vdash_{\mathfrak{T}_{\eta\mathfrak{b}}} b \Leftrightarrow .\neg\neg b$ and by extensionality $\Vdash_{\mathfrak{T}_{\eta\mathfrak{b}}} b = .\neg\neg b$, which expands to $\Vdash_{\mathfrak{T}_{\eta\mathfrak{b}}} \forall P_{o\to o}.Pb \Rightarrow P.\neg\neg b$ and by substitution $\Vdash_{\mathfrak{T}_{\eta\mathfrak{b}}} cb \Rightarrow c.\neg\neg b$. □

Finally we can use the completeness theorems obtained so far to prove a compactness theorem for our semantics.

Corollary 2.3.22 (Compactness Theorem for $\mathfrak{T}_*$)
Let Φ be a set of sentences, then Φ has $\mathfrak{M}_*$-model, iff every finite subset of Φ has a $\mathfrak{M}_*$-model.

Proof: Let every finite subset Ψ of Φ be satisfiable by a $\mathfrak{M}_*$-model then Ψ is $\mathfrak{T}_*$-consistent by 2.3.19, so Φ is $\mathfrak{T}_*$-consistent (every $\mathfrak{T}_*$-proof is finite), and thus satisfiable by a $\mathfrak{M}_*$-model by 2.3.19. □

2.4 Other Applications of Model Existence

An application of model existence theorems that we will not discuss in this book is that they allow for very simple (but non-constructive) proofs of cut-elimination theorems. In [And71] Peter Andrews applies his "Unifying Principle" to cut-elimination in a non-extensional sequent calculus, by proving the calculus complete (relative to $\mathfrak{T}$) both with and without the cut rule and concludes that cut-elimination is valid for this calculus. In the extensional case, where a cut-elimination theorem can be found in [Tak68, Tak87], we can directly model a cut-elimination proof after Andrews' approach, using the model existence theorem for Henkin models.

A related application lies in proof transformation for higher-order logics [Mil83, Pfe87]. Here, proofs found by higher-order automated theorem provers can be transformed into other calculi, such as natural deduction- or sequent calculi that form the basis of tactic-based theorem provers for classical logics like ISABELLE [Pau94] or ΩMEGA [BCF+97]. Dale Miller's original proof transformation system for TPS [Mil83], uses Andrews' "Unifying principle" and only works for non-extensional calculi like higher-order matings. Frank Pfenning's later extensions (by equality and extensionality) build on various cut-elimination theorems. Again, the methods developed in this chapter can shed some light on the situation.

In all these applications, the leverage added by our analysis in this chapter is that we can now extend non-extensional results to extensional cases. However, the generalized model classes have a merit of their own, for instance in higher-order logic programming [NM94], where the denotational semantics of programs can induce non-standard meanings for the classical connectives. For instance, given a SLD-like search strategy as in λ-PROLOG [Mil91], conjunction is not commutative any more. Therefore, various authors have proposed model-theoretic semantics, where property $\mathfrak{b}$ fails. For instance David Wolfram uses Andrew's v-complexes [Wol94] as a semantics for λ-PROLOG and Gopalan Nadathur uses "labelled structures" for the same purpose in [NM94]. It is plausible to assume that the results presented here will be useful for further development in this direction.

Part II

Higher-Order Automated Theorem Proving

EdNote(18)

[18] EdNote: abstracthere

Hilbert
calculus!refutation
refutation
goal-oriented
unification
calculus!analytic
analytic
tableaux
resolution
calculus!generative
generative
clause normal form
modus ponens
empty clause

The field of automated theorem proving is one of the oldest branches of Artificial Intelligence; it has contributed general logic-based methods, which now form one of the fundamental paradigms of artificial intelligence research. Before we start our technical development of higher-order automated theorem proving calculi, we survey the history and state of the art.

Automated Theorem Proving

Automated reasoning programs usually use calculi optimized towards mechanization on a machine. In contrast to the Hilbert-style calculi discussed in the last chapter, which derive theorems from generally accepted truths (axioms), machine-oriented calculi are usually **refutation calculi**, which show validity of a given formula by assuming its negation, and showing that it leads to a contradiction. This simple trick usually makes the calculi much more **goal-oriented** preventing unnecessary search. Moreover, machine-oriented calculi often use (meta)-variables to delay the decision about the instantiation of universal quantifiers. These variables are maintained in the proof and instantiations are only computed when a potential contradiction has been found by a process called **unification**, which finds most general solutions for term equations.

There are hundreds of calculi built on these ideas, which can be categorized into two classes. *Analytical calculi* such as *tableaux* [DGHP99] develop the formula they try to refute in a case-analysis tree, showing that every case leads to a branch with an explicit contradiction. The *resolution* calculus [Rob65] is paradigmatic for the other class (*generative calculi*). It first converts formulae into a specialized form, called *clause normal form* (a conjunction of disjunctions, the clauses), and generates new clauses by the resolution rule, a generalization of modus ponens. A refutation has been found in this calculus, if the empty clause is derived. As an empty disjunction it can never be true, it serves as an elementary contradiction for resolution.

Generally, automated theorem provers are computer programs that implement semi-decision procedures by heuristically searching for a proof of a given conjecture in the space all possible proofs. Available systems for first-order logic have reached a great maturity in the last 45 years, both from the point of logical sophistication (they are available for a wide variety of logics and employ calculi specialized towards mechanized search), as well as from an engineering point of view (systems like **Otter**, **Spass**, or **WaldMeister** can productively search through spaces of literally billions of formulae even on traditional hardware at the rate of tens of thousands of logical inferences per second, see [Sut00]).

It is safe to characterize the state of the art by the following dichotomy: In applications where humans have little or no intuitions that guide the search for solutions, automated theorem proving techniques can be vastly more efficient than humans; General-purpose automated theorem provers have been known to prove problems in mathematics, some that were open for decades, e.g. the Robbins conjecture [McC97].

In applications where humans have obtained intuitions by training, they usually outclass the general-purpose systems dramatically. Sometimes, the human intuitions can be formalized and used to build special-purpose theorem proving systems. In these cases, the systems can compete again, for instance in the case of elementary plane geometry, where systems can prove all known theorems in a matter of seconds each [Wu00].

Machine-Oriented Calculi for $\mathcal{HOL}$

The history of building automated theorem provers for higher-order logic is almost as old as the field of deduction systems itself. In fact, one of the first attempts to build a semi-automated deduction systems (SAM [BEG$^+$64, GOBS69]) did not restrict itself to first-order logic but instead used higher-order logic.

Since then, a wide range of methods for higher-order automated theorem proving has been proposed. In [Rob68, Rob69a] Robinson presented a proof procedure that is essentially a tableau implementation of the calculi given in [Sch60, Tak53]. A similar procedure was later implemented in [Hib73] and successfully applied to problems from number theory. In [Rob69b] Robinson proposes to translate a problem given in higher-order logic into combinatory logic and to give it to a conventional first-order Resolution/Paramodulation theorem prover that has also been given an axiomatization of combinatory logic. In [Dar71] Darlington presents a resolution procedure that allows limited second-order formulae in order to handle induction schemata. He employs the unification algorithms from [Gou66]. Andrews proposed a resolution calculus for full higher-order logic [And71], and for the completeness proof pioneers the higher-order abstract consistency methods presented in the last chapter. This technique is probably more important than the particular calculus itself, which lacks unification just like the calculi discussed so far, and is therefore not practically applicable.

The first successful attempts to mechanize and implement higher-order logic were those of Huet [Hue72] and Jensen and Pietrzykowski [Pie73, JP73, JP76]. They combine the resolution principle with *higher-order unification*, which we discuss in detail in chapter 3. The unification problem in typed λ-calculi is naturally much more complex than that for first-order terms,

Robinson
Robinson
Darlington
Andrews
Huet
Jensen
Pietrzykowski
higher-order unification

flexible literals

since it has to take the theory of λ-equality into account.

Granted a good understanding of higher-order unification, higher-order theorem proving is still a complex issue. Since higher-order unification is undecidable, incorporating unification as a primitive sub-process would not result in an effectively computable calculus. As a remedy, the unification process can be delayed by capturing the unification problems as constraints and effectively interleaving the search for contradictions with the search for unifiers. For instance, Huet's resolution calculus works on a generalization of first-order clauses that also incorporate unification constraints. The resolution rule only cuts literals of complementary polarity from the clauses and adds the appropriate pair to the unification constraint, which is checked for unifiability at the end of the deduction, once an empty clause has been found. Clearly this strategy is not viable for practical applications, since any two literals can be resolved upon, and it is not the case that successive instantiation constrains the search space.

Furthermore, in contrast to first-order refutation theorem proving not all instantiations that are necessary in a refutation can be obtained by unification, since the heads of flexible literals (i.e. literals, where the head is a predicate variable) have to be instantiated with formulae that contain the logical constants $\wedge, \neg, \Pi$, and $=$. These substitutions cannot be found using unification, since the needed head symbols are not even present in the clause set, as they have been eliminated in the clause normal form transformation. Huet solves this problem by introducing special "splitting" inference rules that provide the instantiations by enumerating all possible substitutions. This also solves a problem with Skolemization: It necessary to instantiate a predicate variable occurring negatively by a subformula $\mathbf{A}$. Clearly it is not sound to Skolemize $\mathbf{A}$ before instantiation, since P occurs under a negative prefix. But $\mathbf{A}$ is a subformula of a clause $\mathbf{C}$ and must therefore be Skolemized before refutation.

Unfortunately, the blind enumeration approach can hardly be called practical, since these inference rules are infinitely branching, but a better solution for the general problem remains still to be found. It seems probable that Bledsoe's "set variables" method [Ble77, Ble79] from the context of set-theoretic theorem proving might give some heuristics or even provide a direction towards a complete mechanization.

Finally, naive Skolemization [Sko19] is not sound for higher-order logic [And73]: it is possible to prove an instance of the axiom of choice. The problem with this is that the axiom of choice is independent of $\mathcal{HOL}$ (see for instance 2.1.29), so naive Skolemization is only sound in formulations of higher-order logic that do assume the axiom of choice. Mathematicians who usually assume it, normally do so explicitly, and like to be aware of its usage, since it can lead to mathematically controversial results. Naive Skolemization would introduce it through the back-door and would ruin the explicitness that we had hoped to gain by logic-based formalization. We will spend chapter 4 on this problem and two possible solutions.

primitive substitutions

In chapters 5 and 6 we provide the basis for further development by giving a full account of a higher-order tableau calculus $\mathcal{HT}$ (that is similar in spirit to the TPS procedure [Mil83, And89]) and a higher-order resolution calculus that is a rational reconstruction of Huet's calculus [Hue72, Hue73a]. In contrast to Huet's calculus, we allow an interleaving of the search for contradiction and unification (constraint simplification) to help detect clashes early. Since the substitution may well change the propositional structure of the clause by instantiating a predicate variable, we have to renormalize the clause on the fly. This interleaving also makes it possible to use a variant of Andrews' primitive substitutions [And89] for instantiating flexible literals, which is conceptually much simpler than Huet's splitting rules. Another effect of this interleaving of unification and refutation is that we can augment unification by a rule for treating logical equivalences, that makes it possible to treat embedded occurrences of propositional subformulae. This is necessary for treating propositional extensionality and achieving completeness with respect to Henkin models (see sections 5.5 and 6.6).

Chapter 3

Higher-Order Unification

In this chapter we review higher-order unification and pre-unification. The algorithms will be central inference procedures for the refutation calculi presented in chapters 5 and 6. Building upon the notion of general binding we give a set of transformations for general unification and pre-unification, which will be shown correct and complete based on methods from [Sny91].

3.1 Preliminaries

Unification was first explicitly used in the context of the first-order resolution calculus [Rob65] to find common instances of first-order atoms, although the ideas can be traced back to Herbrand [Her30]. There it enjoys the desirable property that for any pair of first-order terms that have a common instance, a most general unifier can be computed (in linear time). Since this allows to generate the instances needed in a proof on the fly and search for a proof on the most general level, the use of unification has dramatically reduced search spaces. Consequently, unification has become one of the primary computation procedures in automated theorem proving. It has spawned a whole research area [BS94, Kir90], that concentrates on the problem of making a set of formulae equal in some given theory, and finding algorithms for that.

In full generality, the unification problem can be characterize as follows. Let $\mathcal{L}$ be a formal language and $\mathcal{K}$ a corresponding class of models. Then, the unification problem for a subclass $\mathcal{T}$ of models is to determine, whether an equation $\mathbf{A} =^? \mathbf{B}$ is satisfiable in $\mathcal{T}$, i.e. whether for any model $\mathcal{M} \in \mathcal{T}$, there is an assignment φ, such that $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$.

For first-order unification, $\mathcal{L}$ is the set of first-order terms for some signature Σ and $\mathcal{T}$ is the class of standard Σ-algebras. Since first-order terms do not contain function variables, $\beta\eta$-redexes cannot be introduced, so two terms are equal in all standard models, iff they are syntactically equal. This problem is computationally so simple, that it can be solved in linear time,

nullary
Huet
Jensen
Pietrzykowski
pre-unification
explicit substitutions
substitution!explicit

and the set of solutions can be represented by unique minimal substitutions.

The unification problem in typed λ-calculi is naturally much more complex than that for first-order terms, since it has to take the theory of $\beta\eta$-equality into account. Indeed the higher-order unification problem is that for well-typed formulae and Σ-structures.

Higher-order unification has first been investigated in depth by Gould in [Gou66], who already identified the problem as nullary, i.e. complete sets of unifiers need not always have most general elements. Even though Gould's unification algorithms (and also those of Darlington [Dar68] and Ernst [Ern71]) are not complete, the early work identified the major difficulties, and lead to the solution of the problem by Huet, Jensen, and Pietrzykowski [Hue72, Hue76, JP73, JP76]. Huet proved that third-order unification is undecidable [Hue73b], a result that was independently obtained by Lucchesi [Luc72], refined by [Bax78], and finally extended to second-order logic [Gol81, Far91a]. The last result gives a sharp classification of the undecidability of higher-order unification: if the language has one binary function constant, then unification is undecidable; if there are only unary function symbols, then unification is decidable: it can easily be seen to be equivalent to associative unification, which is decidable [Mak77], and has at most infinitely many most general unifiers [Plo72].

The first implementations of higher-order unification already revealed that the search space for unifiers is far too large to be feasible for practical applications. Huet developed a restriction of the higher-order unification problem (*pre-unification*) that is sufficient for the completeness of refutation procedures, where one is only interested in the solvability of unification problems rather than in the unifiers themselves. Although the pre-unification problem is also undecidable, a simple modification of Huet's algorithm enumerates sets of most general solutions to unifiable problems. Moreover the application to almost all practical problems yields small sets of most general unifiers. For a modern presentation of higher-order (pre-)unification we refer the reader to [Sny91].

A totally different approach to higher-order unification is taken by Dougherty in [Dou93], where he gives a unification algorithm for combinatory logic that is based on a first-order narrowing method for the theory of strong combinatory equality (which is equivalent to $\beta\eta$-equality modulo translation). A third approach to higher-order unification is that of so-called **explicit substitutions** (see e.g. [DHK95, DHKP96]), where the formula language is augmented by an explicit concept of substitution application. The first-order theory of substitution application is treated by first-order narrowing techniques.

Another research topic in the field of higher-order unification that is motivated by practical applications is the search for subclasses of higher-order formulae that enjoy a tractable unification problem. A subclass that is particularly interesting because its unification problem is decid-

higher-order pattern
$\preceq$instantiation relation
more general
instance
relation!instance
equal over

able, unitary [Mil92] (solvable unification problems always have unique most general unifiers), and linear [Qia93] (that can be computed in linear space and time) is that of *higher-order patterns* introduced by Miller for the higher-order logic programming language λ-PROLOG [Mil91]. This subclass has since proven its usefulness for higher-order equality reasoning [Nip91, Pre94b, Pre94a] and logical frameworks [Pfe91]. In some cases it is possible to relax the conditions of higher-order patterns and still obtain a decidable unification problem. For instance, unification of pairs consisting of one higher-order pattern and one second-order formula are still decidable [Pre94b, Pre94a].

The question of higher-order unification, where the theory of $\beta\eta$-equality has been augmented by a further equational theory such as associativity or commutativity, has led to algorithms for general- [Sny90] and modular *higher-order E-unification* [NQ91, Mül93, MW94, Web93]. The modular algorithms are, in fact, more of a combination method that allows to combine higher-order unification with existing first-order unification algorithms. For an application of the associative commutative higher-order patterns see [QW94]. In [Joh91, DJ92] the methods developed by Dougherty for unification with combinators have been extended to higher-order E-unification by employing combination methods for first-order narrowing.

In contrast to the higher-order unification problem, where the issue of decidability is basically well-understood, the question of decidability of *higher-order matching* (for given formulae $\mathbf{A}$ and $\mathbf{B}$ find a substitution σ such that $\sigma(\mathbf{A}){=_{\beta\eta}}\mathbf{B}$) is still largely open. In [Hue76] Huet was able to prove decidability of second-order matching and recently Dowek and Padovani have extended this result to third- and fourth-order formulae [Dow92, Pad96]. Even though various authors have studied other special cases [Zai87, Wol93, CQ94, Cur93] the decidability of the general higher-order matching problem is still open.

Definition 3.1.1 (More General)
Let $\mathbf{A}, \mathbf{B} \in \mathit{wff}_\alpha(\Sigma)$, then we say that $\mathbf{A}$ is **more general** than $\mathbf{B}$, iff there exists a substitution $\sigma \in \mathbf{SUB}(\Sigma)$ such that $\sigma(\mathbf{A}){=_{\beta\eta}}\mathbf{B}$. In this case we call $\mathbf{B}$ a **$\beta\eta$-instance** of $\mathbf{A}$, and denote this fact by $\mathbf{A} \preceq_{\beta\eta} \mathbf{B}$ (or simply $\mathbf{A} \preceq \mathbf{B}$). Note that the resulting **instance relation** $\preceq$ is a partial ordering relation on well-typed formulae.

If we want to extend the instance ordering to substitutions, we have to take care to fix a domain of comparison, in order to get good results.

Definition 3.1.2 Let σ, θ be substitutions with and $\mathcal{W} \subseteq \mathcal{V}$, then

- σ and θ are **$\beta\eta$-equal over** $\mathcal{W}$ ($\sigma{=_{\beta\eta}}\theta[\mathcal{W}]$), iff for all $X \in \mathcal{W}$ we have $\sigma(X){=_{\beta\eta}}\theta(X)$.

approximation
unification problem
pair
existential
variable!existential
universal
variable!universal
flexible
rigid
$\mathbf{U}(\mathcal{E})/\mathbf{Mat}(\mathcal{E})$*set of unifiers/matchers of the system* $\mathcal{E}$
higher-order unifier
unifier!higher-order
unifiable
higher-order matcher
matcher!higher-order

- σ is more general than θ over $\mathcal{W}$ ($\sigma \preceq_{\beta\eta} \theta[\mathcal{W}]$), iff there is a substitution ρ such that $\theta =_{\beta\eta} \rho \circ \sigma[\mathcal{W}]$.

- σ is an **approximation** of θ, iff $\theta \circ \sigma =_{\beta\eta} \theta[\mathbf{Dom}(\sigma)]$. For any set $\Psi \subseteq \mathbf{SUB}(\Sigma)$, we say that σ approximates Ψ, iff σ approximates all $\theta \in \Psi$.

3.2 Unification Problems

In this section we define higher-order unification and present the unification problems, the basic structures that are manipulated by our unification algorithms. Originally a higher-order unification problem consists in finding a substitution σ such that $\sigma(\mathbf{A}) =_{\beta\eta} \sigma(\mathbf{B})$ for given formulae $\mathbf{A}$ and $\mathbf{B}$.

Most unification algorithms solve unification problems by transforming sets of pairs $\mathbf{A} =^? \mathbf{B}$ of formulae to a so-called solved form, from which a solution can be directly read off. The presentation of unification algorithms as inference rules is a variant of the presentation as systems of transformations, which was introduced in [MM73].

Definition 3.2.1 (Unification Problem)
Let Γ and Υ be variable contexts, then we call a triple $\exists\Gamma.\forall\Upsilon.\mathbf{C}$ a **unification problem**, iff $\mathbf{C}$ is of the form $\mathbf{C} \equiv \mathbf{P}^1 \wedge \ldots \wedge \mathbf{P}^n$, where the so-called **pairs** $\mathbf{P}^i$ are of the form $\mathbf{P}^i \equiv (\mathbf{A}^i =^? \mathbf{B}^i)$ for some $\mathbf{A}^i, \mathbf{B}^i \in \mathit{wff}(\Sigma; \Gamma \cup \Upsilon)$ or $\mathbf{C} = \mathbf{T}_o$. We will call the variables in $\mathbf{Dom}(\Gamma)$ **existential** and those in $\mathbf{Dom}(\Upsilon)$ **universal**. Intuitively existential variables are those we want to solve for, and the universal variables are used as place-holders for variables that were originally bound.

We call a formula $\mathbf{A}$ in a pair $\mathbf{A} =^? \mathbf{B}$ **flexible**, iff $\mathbf{head}(\mathbf{A})$ is an existential variable and **rigid** otherwise (i.e. if it is a constant or a universal variable).

Let $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{F}$ be a unification problem, to ease the load of notation we will write $\sigma =_{\beta\eta} \theta[\mathcal{E}]$ for $\sigma =_{\beta\eta} \theta[\Gamma]$ and sometimes even drop the $[\Xi]$ in $\sigma \preceq \theta[\Xi]$, if it is clear from the context.

Definition 3.2.2 (Higher-Order Unifier/Matcher)
We will call a substitution σ with $\mathbf{supp}(\sigma) \subseteq \mathbf{Dom}(\Gamma)$ and $\mathbf{Intro}(\sigma) \cap \mathbf{Dom}(\Upsilon) = \emptyset$ a **higher-order unifier** of a unification problem $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{F}$, if σ solves all pairs in $\mathcal{F}$, i.e. $\sigma(\mathbf{A}) =_{\beta\eta} \sigma(\mathbf{B})$ for all pairs $\mathbf{A} =^? \mathbf{B} \in \mathcal{F}$. We call a unification problem $\mathcal{E}$ **unifiable**, iff there is a unifier for $\mathcal{E}$.

We call a substitution σ a **higher-order matcher** for $\mathcal{E}$, iff $\sigma(\mathbf{A}) =_{\beta\eta} \mathbf{B}$ for all pairs $\mathbf{A} =^? \mathbf{B} \in \mathcal{F}$.

The sets of higher-order unifiers and matchers of a unification problem $\mathcal{E}$ are denoted as $\mathbf{U}(\mathcal{E})$ and $\mathbf{Mat}(\mathcal{E})$.

unifiers!complete set
most general unifier
unifier!most general
minimal

Remark 3.2.3 (Higher-Order Unification and Validity)
The logical form of the unification problems is motivated by the fact that a unification problem $\mathcal{E}$ is unifiable, iff it is valid in the class of all Σ-models (cf. 1.4.29). In this discussion, we will use the notational variants $\exists[X\colon\alpha],[Y\colon\beta].\forall[Z\colon\gamma].$ and $\exists X_\alpha, Y_\beta.\forall Z_\gamma.$ interchangeably.

Indeed if $\mathcal{M} = (\mathcal{D}, \mathcal{I})$ is a Σ-algebra, then $\mathcal{M} \models \mathcal{E}$, iff there is a variable assignment φ into $\mathcal{D}$ with $\mathbf{Dom}(\varphi) = \mathbf{Dom}(\Gamma)$, such that $\mathcal{I}_\varphi(\forall\Upsilon.\mathcal{F}) = \mathtt{T}$, since $\mathcal{E}$ is closed. So $\mathcal{F}$ is valid in $\mathcal{M}$, iff for all variable assignments ψ into $\mathcal{D}$ with $\mathbf{Dom}(\psi) = \mathbf{Dom}(\Upsilon)$, such that $\mathcal{I}_{\varphi\cup\psi}(\mathcal{F}) = \mathtt{T}$.

To show our claim of equivalence above, let (without restriction of generality) $\mathcal{F} \equiv \mathbf{A} =^? \mathbf{B}$. Now let θ be a higher-order unifier of $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{F}$ and ψ an arbitrary variable assignment into $\mathcal{D}$ with $\mathbf{Dom}(\psi) = \mathbf{Dom}(\Upsilon)$. To show that $\mathcal{E}$ is valid, we choose $\varphi := \{[\mathcal{I}_\psi(\theta(X))/X] \mid X \in \mathbf{supp}(\theta)\}$. We know that $\theta(\mathbf{A}) =_{\beta\eta} \theta(\mathbf{B})$, so we have $\mathcal{I}_\psi(\theta(\mathbf{A})) = \mathcal{I}_\psi(\theta(\mathbf{B}))$ by Theorem 1.4.24 and $\mathcal{I}_{\varphi\cup\psi}(\mathbf{A}) = \mathcal{I}_{\varphi\cup\psi}(\mathbf{B})$ by the substitution-value property (see 1.4.26 and 1.4.12(4)). This shows the first part of our equivalence.

For the other direction, let $\mathcal{E}$ be valid. In particular, it is valid in $\mathcal{TS}(\Sigma)^{\beta\eta}$, so we have a variable assignment φ into $\mathcal{TS}(\Sigma)^{\beta\eta}$ with $\mathbf{Dom}(\varphi) = \mathbf{Dom}(\Gamma)$, such that $\mathcal{I}_{\varphi\cup\psi}(\mathcal{F}) = \mathtt{T}$ for all assignments ψ into $\mathcal{TS}(\Sigma)^{\beta\eta}$ with $\mathbf{Dom}(\psi) = \mathbf{Dom}(\Upsilon)$. By the construction of $\mathcal{TS}(\Sigma)^{\beta\eta}$, the assignment φ is a ground substitution, so $\mathbf{Intro}(\varphi) = \emptyset$ and thus $\mathbf{Intro}(\varphi) \cap \mathbf{Dom}(\Upsilon) = \emptyset$. Obviously this makes φ into a higher-order unifier of $\mathcal{E}$.

Definition 3.2.4 (Complete Set of Higher-Order Unifiers)
Let $\mathcal{E}$ be a unification problem, then a subset $\Psi \subset \mathbf{U}(\mathcal{E})$ is called a **complete set of unifiers** of $\mathcal{E}$, iff for all $\theta \in \mathbf{U}(\mathcal{E})$ there is a $\sigma \in \Psi$ with $\sigma \preceq_{\beta\eta} \theta[\mathcal{E}]$.

If the singleton set $\{\sigma\} \subset \mathbf{U}(\mathcal{E})$ is a complete set of unifiers for $\mathcal{E}$, then σ is called a **most general unifier** for $\mathcal{E}$. We call a complete set $\Phi \in \mathbf{U}(\mathcal{E})$ of unifiers **minimal** or a **set of most general unifiers**, iff any $\sigma, \theta \in \Phi$ are $\preceq$-incomparable, i.e. we do not have $\sigma \preceq_{\beta\eta} \theta[\mathcal{E}]$.

The corresponding notions for the matching problem are defined accordingly.

Example 3.2.5 (Higher-Order Unifiers)
Let us consider the unification problem $\mathcal{E} := \exists F_{\iota\to\iota}.\forall X_\iota.F(fX) =^? f(FX)$, where f is a constant of type $\iota \to \iota$. The set $\{\sigma_n := [\lambda Z.f^n Z/F] \mid n \in \mathbb{N}\}$ is a complete set of unifiers for $\mathcal{E}$ that is also minimal: Obviously $\sigma_n(F(fX)) =_{\beta\eta} f^{n+1}X =_{\beta\eta} \sigma_n(f(FX))$, and the σ_i are $\preceq$-incomparable for distinct i, as they are all closed.

The cardinality of complete sets of unifiers gives rise to a classification of theories.

unification hierarchy
unitary
finitary
infinitary
nullary
$\mathcal{E}_\sigma$ the unification problem associated to σ
solved in
solved form
solved part

Definition 3.2.6 (Unification Hierarchy)
A theory $\mathcal{T}$ is called

unitary iff any $\mathcal{T}$-unification problem $\mathcal{E}$ is unsolvable or has a most general unifier,

finitary iff any $\mathcal{T}$-unification problem has a finite set of most general unifiers, but there is one problem that has at least two.

infinitary iff any $\mathcal{T}$-unification problem has a set of most general unifiers, but there is a problem that has infinitely many incomparable ones.

nullary else, i.e. there are $\mathcal{T}$-unification problems that do not have minimal and complete sets of unifiers.

Thus with example 3.2.5 we can see that higher-order unification is at least infinitary. In fact, higher-order unification is nullary [Gou66]. Let us now come to the definition of solved forms, i.e. unification problems where a most general unifier can be simple read off.

Definition 3.2.7 (Solved Form)
Let $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{F}$ be a unification problem with $\Gamma(X) = \alpha$. Then we call a pair $X =^? \mathbf{A}$ **solved in** $\mathcal{E}$, iff $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma;\Gamma)$ and X is not free in $\mathbf{A}$ or elsewhere in $\mathcal{F}$. Note that in particular, the variables from Υ may not occur in $\mathbf{A}$.

We call a variable X **solved in** $\mathcal{E}$, iff there is a solved pair $X =^? \mathbf{A}$ in $\mathcal{F}$, and we call $\mathcal{E}$ in **solved form**, if all of its pairs are solved in $\mathcal{E}$.

A unification problem $\mathcal{E} = \exists\Gamma.\forall\Upsilon.X^1 =^? \mathbf{A}^1 \wedge \ldots \wedge X^n =^? \mathbf{A}^n$ in solved form determines a unique substitution $\sigma_\mathcal{E} := [\mathbf{A}^1/X^1], \ldots, [\mathbf{A}^n/X^n]$ that will turn out to be the unique most general unifier of $\mathcal{E}$ (cf. 3.2.8). This result justifies the name. Note that σ is idempotent by definition, since $\mathbf{Dom}(\sigma) \cap \mathbf{Intro}(\sigma) = \emptyset$. Conversely, any idempotent substitution $\sigma \in \mathbf{SUB}(\Sigma;\Gamma)$ uniquely determines a solved unification problem $\exists\Gamma.\mathcal{E}_\sigma$, where $\mathcal{E}_\sigma \exists\Gamma.X_1 =^? \sigma(X_1) \wedge \ldots \wedge X_n =^? \sigma(X_n)$ and $\mathbf{Dom}(\sigma) = \{X^1, \ldots, X^n\}$. Note that any unification problem $\mathcal{E}$ can always be written as $\exists\Gamma.\forall\Upsilon.\mathcal{F} \wedge \mathcal{G}$, where $\mathcal{G}$ is the set of pairs in $\mathcal{E}$ that are solved in $\mathcal{E}$. We call $\mathcal{G}$ the **solved part** of $\mathcal{E}$. Note that $\mathcal{G} = \mathcal{E}_\sigma$, for $\sigma = \sigma_\mathcal{G}$.

Transformation-based unification methods attempt to reduce the input systems to solved systems, which represent their unifiers. The fundamental connection between solved systems and unifiers is the following fact, which shows that solved systems indeed represent their own solutions.

Lemma 3.2.8 *Let $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{E}_\sigma$ be a unification problem in solved form, then σ is a most general unifier for $\mathcal{E}$. In particular, for any unifier θ of $\mathcal{E}$ we have $\theta =_{\beta\eta} \theta \circ \sigma[\mathcal{E}]$.*

Proof: Clearly σ is an well-defined substitution, since $\mathcal{E}$ is in solved form. Moreover $\sigma(X) = \sigma(\mathbf{A})$ for any pair $X =^? \mathbf{A} \in \mathcal{E}_\sigma$, thus σ is a unifier for $\mathcal{E}$. If $\theta \in \mathbf{SUB}(\Sigma; \Delta \to \Gamma)$ is a unifier for $\mathcal{E}$, then $\theta \circ \sigma(X^i) \equiv \theta(\mathbf{A}^i) =_{\beta\eta} \theta(X^i)$ and $\theta(Y) = \theta \circ \sigma(Y)$ for $Y \notin \mathbf{Dom}(\Gamma)$, so that indeed $\theta =_{\beta\eta} \theta \circ \sigma$. □

complete unification procedure
unification procedure
success node
node!failure
failure node
node!failure

With similar methods, we can show the following useful observation.

Lemma 3.2.9 (Approximation Lemma)
If σ approximates $\mathbf{U}(\exists\Gamma.\forall\Upsilon.\mathcal{E})$, then $\mathbf{U}(\exists\Gamma.\forall\Upsilon.\mathcal{E}) = \mathbf{U}(\exists\Gamma.\forall\Upsilon.\mathcal{E} \wedge \mathcal{E}_\sigma)$.

Proof: We only have to show that $\mathbf{U}(\exists\Gamma.\forall\Upsilon.\mathcal{E} \wedge \mathcal{E}_\sigma) \subseteq \mathbf{U}(\exists\Gamma.\forall\Upsilon.\mathcal{E})$, since the other direction is trivial. So let $\theta \in \mathbf{U}(\exists\Gamma.\forall\Upsilon.\mathcal{E})$, then σ approximates θ, so $\theta =_{\beta\eta} \theta \circ \sigma[\mathbf{Dom}(\sigma)]$ by definition. So for any pair $X =^? \sigma(X)$ in $\mathcal{E}_\sigma$, (note that $X \in \mathbf{Dom}(\sigma)$ here), we have $\theta(X) =_{\beta\eta} \theta \circ \sigma(X) = \theta(\sigma(X))$ and thus $\theta \in \mathbf{U}(\exists\Gamma.\forall\Upsilon.\mathcal{E}_\sigma)$, and finally $\theta \in \mathbf{U}(\exists\Gamma.\forall\Upsilon.\mathcal{E} \wedge \mathcal{E}_\sigma)$. □

The next lemma will be used to show that we need not be concerned with solved pairs, when computing unifiers. It is therefore consistent with the intuition that the solved part of a system is merely a record of an answer substitution being constructed.

Lemma 3.2.10 $\mathbf{U}(\exists\Gamma.\forall\Upsilon.\mathcal{E} \wedge \mathcal{E}_\sigma) = \mathbf{U}(\exists\Gamma.\forall\Upsilon.\sigma(\mathcal{E}) \wedge \mathcal{E}_\sigma)$

Proof: Clearly we have $\theta \in \mathbf{U}(\exists\Gamma.\forall\Upsilon.\mathcal{E} \wedge \mathcal{E}_\sigma)$, iff $\theta \in \mathbf{U}(\exists\Gamma.\forall\Upsilon.\mathcal{E}_\sigma) \cap \mathbf{U}(\exists\Gamma.\forall\Upsilon.\mathcal{E})$, so $\theta = \theta \circ \sigma$ by 3.2.8. Now $\theta \circ \sigma \in \mathbf{U}(\exists\Gamma.\forall\Upsilon.\mathcal{E})$, iff $\theta \in \mathbf{U}(\exists\Gamma.\forall\Upsilon.\sigma(\mathcal{E}))$, which gives the assertion. □

Definition 3.2.11 (Complete Unification Procedure)
An inference system $\mathcal{I}$ is called a **complete unification procedure**, iff for every unification problem $\mathcal{E}$ and each substitution $\theta \in \mathbf{U}(\mathcal{E})$ there is a system $\mathcal{F}$ in solved form such that $\mathcal{E} \vdash_{\mathcal{I}} \mathcal{F}$ and $\sigma_{\mathcal{F}} \preceq \theta[\mathcal{E}]$.

Remark 3.2.12 The algorithms for unification consist in the process of systematically exploring the search trees generated by the respective inference systems from a root $\mathcal{E}$. The leaves of these trees are labeled with unification problems, where none of the rules apply. A leaf is called a **success node**, if the corresponding unification problem is in solved form, otherwise it is called a **failure node**.

We will prove (cf. 3.5.11) that the set of substitutions corresponding to the unification problems of the success nodes of the unification tree are complete sets of unifiers of $\mathcal{E}$.

3.3 Simplification

We present a unification algorithm that solves unification problems by transforming them into solved form. For a given unification problem $\mathcal{E}$ it returns a complete set of unifiers, if $\mathcal{E}$ is unifiable, and fails otherwise. It is a generalization of Huet's higher-order unification algorithm [Hue72], as presented in [Sny90].

Definition 3.3.1 ($\mathcal{SIM}$: Simplification of Unification Problems) Let $\mathcal{SIM}$ be the inference system given in Figure 3.1 This set of rules

$$\frac{\exists\Gamma.\forall\Upsilon.(\lambda X_\alpha.\mathbf{A}) =^? (\lambda Y_\alpha.\mathbf{B}) \wedge \mathcal{E} \quad Z \notin \mathbf{Dom}(\Gamma, \Upsilon)}{\exists\Gamma.\forall\Upsilon, [Z{:}\,\alpha].[Z/X]\mathbf{A} =^? [Z/Y]\mathbf{B} \wedge \mathcal{E}}\,\mathcal{SIM}(\alpha)$$

$$\frac{\exists\Gamma.\forall\Upsilon.(\lambda X_\alpha.\mathbf{A}) =^? \mathbf{B} \wedge \mathcal{E} \quad Z \notin \mathbf{Dom}(\Gamma, \Upsilon)}{\exists\Gamma.\forall\Upsilon, [Z{:}\,\alpha].[Z/X]\mathbf{A} =^? (\mathbf{B}Z) \wedge \mathcal{E}}\,\mathcal{SIM}(\eta)$$

$$\frac{\exists\Gamma.\forall\Upsilon.\mathcal{E} \wedge \mathbf{T}_o}{\exists\Gamma.\forall\Upsilon.\mathcal{E}}\,\mathcal{SIM}(\top_o) \qquad \frac{\exists\Gamma.\forall\Upsilon.\mathbf{A} =^? \mathbf{A} \wedge \mathcal{E}}{\exists\Gamma.\forall\Upsilon.\mathcal{E}}\,\mathcal{SIM}(triv)$$

$$\frac{\exists\Gamma.\forall\Upsilon.h\overline{\mathbf{U}^n} =^? h\overline{\mathbf{V}^n} \wedge \mathcal{E} \quad h \in \Sigma \cup \mathbf{Dom}(\Upsilon)}{\exists\Gamma.\forall\Upsilon.\mathbf{U}^1 =^? \mathbf{V}^1 \wedge \ldots \wedge \mathbf{U}^n =^? \mathbf{V}^n \wedge \mathcal{E}}\,\mathcal{SIM}(dec)$$

$$\frac{\begin{array}{c}\mathbf{Dom}(\Upsilon) \cap \mathbf{Free}(\lambda\overline{X_n}.\mathbf{A}) = \emptyset \quad \Gamma(F) = \overline{\alpha_n} \to \beta \\ F \in \mathbf{Free}(\mathcal{E}) \quad F \notin \mathbf{Free}(\lambda\overline{X^n}.\mathbf{A}) \\ \exists\Gamma.\forall\Upsilon, [X^i{:}\,\alpha_i].F\overline{X^n} =^? \mathbf{A} \wedge \mathcal{E}\end{array}}{\exists\Gamma.\forall\Upsilon.F =^? (\lambda\overline{X^n}.\mathbf{A}) \wedge [(\lambda\overline{X^n}.\mathbf{A})/F]\mathcal{E}}\,\mathcal{SIM}(elim)$$

Figure 3.1: Simplification rules for Higher-Order Unification

is used with the convention that all formulae are eagerly reduced to head normal form, i.e. each rule consists of two parts, first applying the transformation, and then reducing to head normal form. Furthermore, we apply these rules with the understanding that the operators $\wedge$ and $=^?$ are commutative $(\mathcal{E}\wedge\mathcal{F}) = (\mathcal{F}\wedge\mathcal{E})$ and associative $(\mathcal{E}\wedge(\mathcal{F}\wedge\mathcal{G})) = ((\mathcal{E}\wedge\mathcal{F})\wedge\mathcal{G})$. Note that, in contrast to [Mil92], we consider the quantifier prefixes as declarations and therefore do not need inference rules for quantifier exchange.

We will now show two important properties of simplification: $\mathcal{SIM}$ converses unifiers and is strongly normalizing.

Lemma 3.3.2 *If $\mathcal{D}{:}\,\mathcal{E} \vdash_{\mathcal{SIM}} \mathcal{E}'$, then $\mathbf{U}(\mathcal{E}) = \mathbf{U}(\mathcal{E}')[\mathcal{E}]$.*

Proof: Clearly it suffices to show the assertion for the case, where $\mathcal{D}$ consists of a single rule application. The assertion is trivial for $\mathcal{SIM}(triv)$ and

dp(**A**), **dp**(θ)*depth of* **A**, θ ***depth***

$\mathcal{SIM}(\top_o)$. For $\mathcal{SIM}(\alpha)$ let

$$\frac{\exists\Gamma.\forall\Upsilon.(\lambda X_\alpha.\mathbf{A})=^?(\lambda Y_\alpha.\mathbf{B}) \quad Z\notin\mathbf{Dom}(\Gamma,\Upsilon)}{\exists\Gamma.\forall\Upsilon'.[Z/X]\mathbf{A}=^?[Z/Y]\mathbf{B}\wedge\mathcal{E}}\,\mathcal{SIM}(\alpha)$$

where $\Upsilon' := \Upsilon, [Z{:}\,\alpha]$.

Furthermore, let θ be a substitution, then $\theta(\lambda X.\mathbf{A}) = \theta(\lambda Y.\mathbf{B})$, iff $\theta(\lambda Z.[Z/X]\mathbf{A}) = \theta(\lambda Z.[Z/Y]B)$ by α-conversion, iff $(\lambda Z.\theta([Z/X]\mathbf{A}) = (\lambda Z.\theta([Z/Y]B)$, by definition and since $Z \notin \mathbf{Dom}(\Gamma,\Upsilon) \supseteq \mathbf{Free}(\mathbf{A}) \cup \mathbf{Free}(\mathbf{B})$. However the last condition is equivalent to $\theta([Z/X]\mathbf{A}) = \theta([Z/Y]B)$ by *ms:trans*. Thus the sets of substitutions that solve $\mathcal{E}$ and $\mathcal{E}'$ are identical.

In the presence of $\mathcal{SIM}(\alpha)$ the rule $\mathcal{SIM}(\eta)$ is equivalent to a direct consequence of η-conversion: The set of unifiers does not change by replacing **B** by its η-expansion $(\lambda X_\alpha.\mathbf{B}X)$. Now we obtain the assertion for $\mathcal{SIM}(\eta)$ by that for $\mathcal{SIM}(\alpha)$.

In the $\mathcal{SIM}(dec)$ case we have

$$\frac{\exists\Gamma.\forall\Upsilon.h\overline{\mathbf{U}^n}=^?h\overline{\mathbf{V}^n}\wedge\mathcal{E} \quad h\in\Sigma\cup\mathbf{Dom}(\Upsilon)}{\exists\Gamma.\forall\Upsilon.\mathbf{U}^1=^?\mathbf{V}^1\wedge\ldots\wedge\mathbf{U}^n=^?\mathbf{V}^n\wedge\mathcal{E}}\,\mathcal{SIM}(dec)$$

Let $\theta \in \mathbf{U}(\exists\Gamma.\forall\Upsilon.\mathbf{U}^1=^?\mathbf{V}^1\wedge\ldots\wedge\mathbf{U}^n=^?\mathbf{V}^n\wedge\mathcal{E})$, so $\theta(\mathbf{U}^i)=_\beta\theta(\mathbf{V}^i)$ for all $1 \le i \le n$ and therefore

$$\theta(h\overline{\mathbf{U}^n}) \equiv h\overline{\theta(\mathbf{U}^n)}=_\beta h\overline{\theta(\mathbf{V}^n)} \equiv \theta(h\overline{\mathbf{V}^n})$$

Thus for any atom h we have $\theta \in \mathbf{U}(\mathcal{E})$.

For $\mathcal{SIM}(elim)$ let $(F\overline{X})=^?\mathbf{A}$ be the pair that the rule acts upon. We show that for an arbitrary unifier θ of $\mathcal{E}$, the formula $\lambda\overline{X}.\mathbf{A}$ is more general than $\theta(F)$. So let θ be an arbitrary unifier of $\mathcal{E}$ then

$$\theta(F)=_{\beta\eta}\theta(\lambda\overline{X}.F\overline{X}) = \lambda\overline{X}.\theta(F\overline{X})=_{\beta\eta}\lambda\overline{X}.\theta(\mathbf{A}) = \theta(\lambda\overline{X}.\mathbf{A})$$

since the X^i are not in $\mathbf{Dom}(\theta)$. This is just the claim with θ as the instantiating substitution. Now we obtain the assertion by 3.2.10. ☐

For the measures used in the termination result we will need the concept of the depth of a formula, which we will now define.

Definition 3.3.3 (Depth)
We define the **depth dp**(**A**) of a formula **A** inductively on the structure of **A** by setting the depth of variables and constants to 0 and the depths of applications and λ-abstractions to the maximum of depths of the immediate subformulae incremented by 1.

depth
substitution!depth
multiset
multiplicity
multiset ordering
ordering!multiset

This definition is only a special case of the general definition of depth for trees, setting the depth of leaves to 0. It gives us a notion of depth for $\Sigma\Lambda^{\rightarrow}$-derivations, that we use in the following. We say that the **depth** of a substitution θ is the maximum of the $\mathbf{dp}(\theta(X))$ for all $X \in \mathbf{Dom}(\theta)$.

Example 3.3.4 We have $\mathbf{dp}(X) = 0$ and $\mathbf{dp}(\lambda X.X) = 1$, examples of formulae of depth 2 are $\lambda X.(fXX)$ and $f(gX)$. Note that the order of a type (see 1.1.4) is just the depth of a type interpreted as a tree.

Furthermore, we will need the concept of a **multiset**, intuitively, this is a set that allows multiple occurrences of elements. For instance $S := \{\mathsf{a}, \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{c}, \mathsf{c}\}$ is a multiset, in which a occurs twice and c with **multiplicity** three. It is well-known (see [Sny90] for details) that an ordering $\prec$ on elements can be extended to ordering on multisets, the so-called **multiset ordering**: $M = M' * \mathsf{a} \prec_{mul} M' \cup N$, where $\mathsf{a} \prec \mathsf{b}$ for all $\mathsf{b} \in N$. For instance we have $S \prec_{mul} \{\mathsf{a}, \mathsf{a}, \mathsf{a}, \mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{c}\}$, if $\mathsf{a} \prec \mathsf{c}$.

Lemma 3.3.5 *Simplification with $\mathcal{SIM}$ is terminating and confluent up to α-equivalence.*

Proof: The proof goes by close inspection of the $\mathcal{SIM}$ inference rules: Confluence is a consequence of the fact that for any pair only one rule is applicable (the rule is determined by the top formula constructors of the side). Thus the variation is in the choice of pairs to operate upon, but this choice is unproblematic, since only the $\mathcal{U}\Lambda(elim)$ rules affects the remaining pairs of a unification problem at all but only transforms $\mathcal{SIM}$-irreducible pairs to reducible ones and does not alter rule applicability of $\mathcal{SIM}$-reducible ones.

For termination let $\mu(\mathcal{E})$ be the triple $(\mu_1(\mathcal{E}), \mu_2(\mathcal{E}), \mu_3(\mathcal{E}))$, where

1. $\mu_1(\mathcal{E})$ is the number of is unsolved variables in $\mathcal{E}$,
2. $\mu_2(\mathcal{E})$ is the multiset of $\mathbf{dp}(\mathbf{A}) + \mathbf{dp}(\mathbf{B})$ for all pairs $\mathbf{A} =^? \mathbf{B}$ in $\mathcal{E}$, and
3. $\mu_3(\mathcal{E})$ is the multiset of types or pairs in $\mathcal{E}$.

Furthermore, let $\prec$ be the strict lexicographic ordering for the obvious component orderings. Then we can see that the rule $\mathcal{U}\Lambda(elim)$ decreases μ_1 (but may increase μ_2). The rule $\mathcal{SIM}(\eta)$ leaves the set of unsolved variables invariant, but decreases the depth of the pair it acts on. Furthermore, $\mu_2(\lambda X.\mathbf{A} =^? \mathbf{B}) = \mathbf{dp}(\lambda X.\mathbf{A}) + \mathbf{dp}(\mathbf{B}) = \mathbf{dp}([Z/X]\mathbf{A}) + \mathbf{dp}(\mathbf{B}Z) = \mu_2([Z/X]\mathbf{A} =^? \mathbf{B}Z)$, and all the other rules of $\mathcal{SIM}$ decrease the term depths or (in the case of $\mathcal{SIM}(dec)$ exchange a pair for a set of pairs of smaller depth). Thus all of these rules decrease μ_2 and preserve μ_1. Thus well-foundedness of $\prec$ gives termination of $\mathcal{SIM}$. $\square$

Thus we can use it to reduce unification problems to a unique normal form (we assumed implicit α-conversion for unification problems) which we call

$\mathcal{SIM}$-**normal form**. By inspection we can easily see, that in $\mathcal{SIM}$-normal forms all pairs must be of the form $h\overline{\mathbf{U}}=^?k\overline{\mathbf{V}}$, where h and k are constants or variables. Furthermore this normal form is effectively computable, since $\beta\eta$-normal forms are 1.3.14.

normal form
normal form!simplification
$\mathcal{G}^h_\alpha$, $\mathcal{G}^j_\alpha$ *general/j-projection binding*
general binding
binding!general
imitation binding
projection binding
binding!projection
approximation

Unfortunately, $\mathcal{SIM}$ does not transform arbitrary unification problems to solved form ($\exists[F{:}\,\alpha\to\beta]\mathbf{.}Fa_\alpha =^? b_\beta$ is in $\mathcal{SIM}$-normal form, but not solved). However we can easily verify by inspection of the rules that all $\mathcal{SIM}$ normal forms consist of pairs of the form $h\overline{\mathbf{U}}=^?k\overline{\mathbf{V}}$, such that, if the heads h and k are constants or non-positive variables, then $h \neq k$. Since such rigid/rigid pairs are not solvable, we only have to concern ourselves with flex/rigid and flex/flex pairs in the following.

3.4 General Bindings

One of the key steps in higher-order unification consists in solving the following problem: given a type α and a symbol (constant or variable) h, find the most general well-typed formula of type α that has head h. Such formulae are called general bindings (cf. 3.4.1) of type α for the head h.

Definition 3.4.1 (General Binding)
Let h be a constant or variable of type $\overline{\delta_n}\to\gamma$ and $\alpha = \overline{\beta^k}\to\gamma$, such that $\gamma\in\mathcal{BT}$, then then we call the formula

$$\mathcal{G}^h_\alpha := \lambda\overline{X^k_\beta}\mathbf{.}h(H^1\overline{X})\ldots(H^n\overline{X})$$

the **general binding** of head h and type α, if the H^i are new variables of types $\overline{\beta_k}\to\delta_i$. If h is a constant or free variable, then we call $\mathcal{G}^h_\alpha$ an **imitation binding**, else (h is the bound variable X^j) a j-**projection binding**. In the latter case we write $\mathcal{G}^j_\alpha$, since the name of the bound variable carries no meaning outside of $\mathcal{G}^j_\alpha$.

Note that this definition is unique up to the names of the new variables H^i and only depends on the signature Σ and the context Γ (if h is a free variable in $\mathbf{Dom}(\Gamma)$). If we want to characterize this dependency, we write $\mathcal{G}^h_\alpha(\Sigma,\Gamma,\mathcal{C})$, where $\mathcal{C}$ is a variable context $[H^1{:}\,\overline{\beta}\to\delta^1],\ldots,[H^n{:}\,\overline{\gamma}\to\delta^n]$ that assigns the appropriate types to the H^i. Finally, for given type α, and head h we collect the imitation binding and all projection bindings (there need not be projection bindings for all types) in set of **approximation**s of h and α

$$\mathcal{A}^h_\alpha(\Sigma,\Gamma,\mathcal{C}) := \{\mathcal{G}^h_\alpha(\Sigma,\Gamma,\mathcal{C})\}\cup\{\mathcal{G}^j_\alpha(\Sigma,\Gamma,\mathcal{C})\} \mid j\leq \mathbf{ln}(\alpha)\}$$

The significance of this class of formulae is given by the following theorem, which is a simple consequence of the normal form Lemma (cf. 1.3.1).

Theorem 3.4.2 (General Binding Theorem)
Let $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma;\Gamma)$ *be a formula with* $\mathbf{head}(\mathbf{A}) = h$*, then there exists a substitution* $\rho \in \mathbf{SUB}(\Sigma;\mathcal{C} \to \Gamma)$*, such that* $\rho(\mathbf{G}) =_{\beta\eta} \mathbf{A}$ *where* $\mathbf{G} = \mathcal{G}^h_\alpha(\Sigma,\Gamma,\mathcal{C})$ *is the general binding for* h *and* α*. Moreover, if* $\mathbf{A}$ *is a head normal form, then* $\mathbf{dp}(\rho) < \mathbf{dp}(\mathbf{A})$*.*

Proof: We can assume without loss of generality, that $\mathbf{A}$ is a head normal form, since the first part of the assertion is invariant under $\beta\eta$-equality. So let $\mathbf{A} = \lambda \overline{X^k_\beta}.h\overline{\mathbf{U}^n}$. Obviously the substitution

$$\rho := [\lambda \overline{X^k_\beta}.\mathbf{U}^1/H^1], \ldots [\lambda \overline{X^k_\beta}.\mathbf{U}^n/H^n]$$

is sufficient for the equality, since

$$(\lambda X^k_\beta.\mathbf{U}^i)\overline{X^k} \to^*_\beta \mathbf{U}^i$$

and furthermore meets the depth conditions, since the matrix of each of the $\rho(H^i)$ is a strict subformula of the matrix $\mathbf{A}$ and the binders are identical. □

Example 3.4.3 (Imitation Bindings)
If we assume the signature $\Sigma = [0\colon\iota], [-\colon\iota \to \iota], [+\colon\iota \to \iota \to \iota]$, then we have the following imitation bindings:

$$\begin{array}{rcl}
\mathcal{G}^0_\iota & = & 0 \\
\mathcal{G}^0_{\iota\to\iota} & = & \lambda X_\iota.0 \\
\mathcal{G}^-_\iota & = & -H_\iota \\
\mathcal{G}^-_{\iota\to\iota} & = & \lambda X_\iota. -(H_\iota X) \\
\mathcal{G}^+_{\iota\to\iota} & = & \lambda X_\iota. +(H^1_\iota X)(H^2_\iota X)
\end{array}$$

If we come back to the example about elementary set theory (see Definition 1.2.15), then we have (note that general bindings are $\beta\eta$ long forms)

$$\begin{array}{rcl}
\mathcal{G}^\emptyset_o & = & \emptyset H_\alpha \\
\mathcal{G}^\emptyset_{\alpha\to o} & = & \lambda X_\alpha.\emptyset(H_{\alpha\to\alpha}X) \\
\mathcal{G}^\cup_{\alpha\to o} & = & \lambda X_\alpha. \cup (H^1_{\alpha\to\alpha\to o}X)(H^2_{\alpha\to\alpha\to o}X)(H^3_{\alpha\to\alpha}X)
\end{array}$$

Example 3.4.4 (Projection Bindings)
The name of projection bindings comes from the fact that the canonical projections are examples, consider for instance the following projection bindings:

$$\mathcal{G}^1_{\iota\to\iota\to\iota\to\iota} = \lambda X_\iota Y_\iota Z_\iota.X \quad \mathcal{G}^2_{\iota\to\iota\to\iota\to\iota} = \lambda X_\iota Y_\iota Z_\iota.Y \quad \mathcal{G}^3_{\iota\to\iota\to\iota\to\iota} = \lambda X_\iota Y_\iota Z_\iota.Z$$

Note that projection bindings are heavily dependent on the type, all three examples have a type of the form $\alpha \to \alpha$.

$\mathcal{U}\Lambda$transformations for unification

$$\begin{aligned}
\mathcal{G}^1_{\iota\to\iota} &= \lambda X_\iota . X \\
\mathcal{G}^1_{(\iota\to\iota)\to\iota\to\iota} &= \lambda X_{\iota\to\iota} Y_\iota . X(H_{(\iota\to\iota)\to\iota\to\iota} XY) \\
\mathcal{G}^2_{(\iota\to\iota)\to\iota\to\iota} &= \lambda X_{\iota\to\iota} Y_\iota . Y
\end{aligned}$$

In the first case α is the base type ι, and there is only the identity as the 1-projection. In the second and third case α is functional, and we have more than one projection[1].

Moreover, not every type admits projection bindings, e.g. $(\iota \to o)$ has none, and $(\iota \to o \to \iota)$ only admits 1-projections.

Remark 3.4.5 (Construction of General Bindings)
The construction of the general bindings can be rationalized from the requirements of being the most general formula of a given type and head, if we restrict ourselves to long $\beta\eta$ normal forms (see Definition 1.3.18): Let $\alpha = \overline{\beta_k} \to \gamma$, then $\mathcal{G}^h_\alpha$ must be of the form $\lambda \overline{X^k_\alpha}.\mathbf{C}$, where $\mathbf{C}$ is of type γ, since $\mathcal{G}^h_\alpha$ is $\beta\eta$-long and has type α. As $\mathbf{head}(\mathcal{G}^h_\alpha) = h$ and $h\colon \overline{\delta_m} \to \gamma$, we have $\mathbf{C} = h\mathbf{D}^1_{\delta_1} \ldots \mathbf{D}^m_{\delta_m}$. Now, we have to choose the arguments $\mathbf{D}^i$, since we want to be most general, the first idea is that they should be variables, but then in any instance of $\mathcal{G}^h_\alpha$, the arguments would be independent of the bound variables X^i (e.g. $\lambda X.hX$ is not an instance of $\lambda X.hH$). Therefore we have to choose the $\mathbf{D}^i$ as applications of new variables H^i to the bound variables, i.e. $\mathbf{D}^i = (H\overline{X})$ (in this case e.g. $\lambda X.hX$ is a $\beta\eta$-instance of $\lambda X.h(HX)$, with $\rho = [\lambda X.X/H]$). The types of the new variables H^i are determined by the requirement that $\mathbf{D}^i$ have types δ_i: we must have $H\colon \overline{\beta_k} \to \delta_i$.

Thus we see that a simple requirement analysis fully determines the form of the general bindings up to the names of the new variables H^i.

3.5 General Higher-Order Unification

Now we can use the general bindings to extend simplification with $\mathcal{SIM}$ to full higher-order unification. The remaining cases are the flex/rigid and flex-flex pairs of the form $F_\alpha\overline{\mathbf{U}^n} =^? h\overline{\mathbf{V}^m}$. Here the intuition is to bind the head variable F_α to a general binding of the correct type (α) and of a head that makes decomposition applicable (i.e the head must be h or bound).

Definition 3.5.1 ($\mathcal{U}\Lambda$: Transformations for Unification)
Let $\mathcal{U}\Lambda$ be the system $\mathcal{SIM}$, augmented by the inference rules in Figure 3.2.

[1] In particular, in the case of a polymorphic λ-calculus that allows type variables, there are in general infinitely many projection bindings, which makes higher-order unification a computationally very hard problem there.

Just as in $\mathcal{SIM}$ leave the associativity and commutativity of $\wedge$ and $=^?$ implicit. Furthermore, all formulae are eagerly reduced to $\mathcal{SIM}$-normal form, i.e. each rule consists of two parts, first applying the transformation, and maximally $\mathcal{SIM}$-reducing afterwards.

$$\frac{\exists \Gamma.\forall \Upsilon.h\overline{\mathbf{U}^n}=^?h\overline{\mathbf{V}^n} \wedge \mathcal{E} \quad h \in \mathbf{Dom}(\Gamma)}{\exists \Gamma.\forall \Upsilon.\mathbf{U}^1=^?\mathbf{V}^1 \wedge \ldots \wedge \mathbf{U}^n=^?\mathbf{V}^n \wedge \mathcal{E}}\,\mathcal{U}\Lambda(dec)$$

$$\frac{\exists \Gamma.\forall \Upsilon.F_\alpha\overline{\mathbf{U}}=^?h\overline{\mathbf{V}} \wedge \mathcal{E} \quad \mathbf{G} \in \mathcal{A}^h(\Sigma,\Delta,\mathcal{C})}{\exists \Gamma,\mathcal{C}.\forall [\mathbf{G}/F]\Upsilon.F=^?\mathbf{G} \wedge [\mathbf{G}/F](F\overline{\mathbf{U}}=^?h\overline{\mathbf{V}} \wedge \mathcal{E})}\,\mathcal{U}\Lambda(\mathit{flex/rig})$$

$$\frac{\exists \Gamma.\forall \Upsilon.F_\alpha\overline{\mathbf{U}}=^?H_\beta\overline{\mathbf{V}} \wedge \mathcal{E} \quad \mathbf{G} \in \mathcal{A}^h(\Sigma,\Delta,\mathcal{C})}{\exists \Gamma,\mathcal{C}.\forall [\mathbf{G}/F]\Upsilon.F=^?\mathbf{G} \wedge [\mathbf{G}/F](F\overline{\mathbf{U}}=^?H\overline{\mathbf{V}} \wedge \mathcal{E})}\,\mathcal{U}\Lambda(guess)$$

Figure 3.2: Rules for general Higher-Order Unification

Note that type conditions in the inference rules enforce the preservation of well-typedness of the unifiers.

Remark 3.5.2 Our $\mathcal{U}\Lambda(\mathit{flex/rig})$ rule subsumes Huet's rules of imitation ($\mathbf{G}$ has head h) and projection ($\mathbf{G}$ is a projection binding) transformations (see [Sny91]), since $\mathcal{A}^h(\Sigma,\Delta,\mathcal{C})$ contains imitation and projection bindings.

In order go get an intuition for the unification procedure let us work an example. We come back to the Church numerals defined above (see Example 1.2.17) and use higher-order unification to solve linear equations over natural numbers. These are a good benchmark for one's unification skills, since it is easy to generate arbitrarily complex examples.

Example 3.5.3 (Higher-Order Unification for Church Numerals) To keep the unification problems manageable, we will solve the single relatively simple equation $\underline{X+1}=\underline{3}$ and hope to obtain the single higher-order unifier $[\underline{2}/X]$.

After translation and initial β-reduction, we obtain the unification problem:

$$\mathcal{E}_1 := \exists X_\nu.\lambda S_{\iota\to\iota}O_\iota.XS(SO)=^?\lambda S_{\iota\to\iota}O_\iota.S(S(SO))$$

applying $\mathcal{SIM}(\alpha)$ twice leaves us with the problem

$$\mathcal{E}_2 := \exists X_\nu.\forall S_{\iota\to\iota},O_\iota.XS(SO)=^?S(S(SO))$$

The pair $XS(SO)=^?S(S(SO))$ is flex/rigid (S is universal), so we can apply $\mathcal{U}\Lambda(\mathit{flex/rig})$. The imitation binding $\mathcal{G}^S_\nu = \lambda Z_{\iota\to\iota}W_\iota.S(H^1ZW)$, where H is a new variable of type ν, cannot be eliminated by $\mathcal{SIM}(elim)$, since

Free($\mathcal{G}_\nu^S$) contains the universal variable S. So we have the choice between the 1- and 2-projections. Using $\mathcal{G}_\nu^2 = \lambda Z_{\iota\to\iota} W_\iota.W$, leads to the pair $SO =^? S(S(S(O)))$, which is unsolvable, since it is rigid, so we use $\mathcal{U}\Lambda(\mathit{flex}/\mathit{rig})$ with $\mathcal{G}_\nu^1 = \lambda Z_{\iota\to\iota} W_\iota.Z(H_\nu^1 ZW)$, and arrive at

$$\mathcal{E}_2 := \exists X_\nu, H_\nu^1.\forall S_{\iota\to\iota}, O_\iota.X =^? \mathcal{G}_\nu^1 \wedge S(H^1 S(SO)) =^? S(S(SO))$$

Note that the imitation step included eliminating the new variable X and reducing to head normal form. The unsolved pair is rigid/rigid, therefore, we decompose using $\mathcal{SIM}(dec)$ and obtain

$$\mathcal{E}_4 := \exists X_\nu, H_\nu^1.\forall S_{\iota\to\iota}, O_\iota.X =^? \mathcal{G}_\nu^1 \wedge H^1 S(SO) =^? S(SO)$$

The pair $H^1 S(SO) =^? S(SO)$ is again flex/rigid, the imitation binding is impossible and the 2-projection leads to immediate failure (as above), so we again use $\mathcal{U}\Lambda(\mathit{flex}/\mathit{rig})$ with $\mathcal{G}_\nu^1$ and obtain (after decomposition)

$$\begin{aligned}\mathcal{E}_5 := \exists X_\nu, H_\nu^1, H_\nu^2.\forall S_{\iota\to\iota}, O_\iota.&\\ &X =^? \lambda Z_{\iota\to\iota} W_\iota.Z(Z(H^2 ZW)) \wedge \\ &H^1 =^? \lambda Z_{\iota\to\iota} W_\iota.Z(H^2 ZW) \wedge H^2 S(SO) =^? SO\end{aligned}$$

This time, we apply $\mathcal{U}\Lambda(\mathit{flex}/\mathit{rig})$ with the 2-projection binding $\mathcal{G}_\nu^2 = \lambda Z_{\iota\to\iota} W_\iota.W$, to the unsolved pair and obtain

$$\begin{aligned}\mathcal{E}_6 := \exists X_\nu, H_\nu^1, H_\nu^2.\forall S_{\iota\to\iota}, O_\iota.&\\ &X =^? (\lambda Z_{\iota\to\iota} W_\iota.Z(ZW)) \wedge H^1 =^? \lambda Z_{\iota\to\iota} W_\iota.ZW \wedge \\ &H^2 =^? \lambda Z_{\iota\to\iota} W_\iota.W \wedge SO =^? SO\end{aligned}$$

which is solved, if we delete the trivial pair by $\mathcal{SIM}(triv)$. The 1-projection would have lead to the pair $H^3 S(SO) =^? O$, which is unsolvable, since the imitation binding is forbidden, and both projections will result a left hand side that has head $S \neq O$.

Now, we can read off the most general unifier

$$\sigma_{\mathcal{E}_6} = [\lambda Z_{\iota\to\iota} W_\iota.Z(ZW)/X], [\lambda Z_{\iota\to\iota} W_\iota.ZW/H^1], [\lambda Z_{\iota\to\iota} W_\iota.W/H^2]$$

of the $\mathcal{E}_6$. Restricting $\mathcal{E}_{\mathcal{E}_6}$ to the variables in the original problem $\mathcal{E}_1$ we arrive at the unifier $[\underline{2}/X]$ as expected.

Let us now consider an example, where $\mathcal{U}\Lambda$ is branching and nonterminating.

Example 3.5.4 (Branching Unification)
We will compute the higher-order unifiers from Example 3.2.5 above where f is a constant of type $\iota \to \iota$:

$$\mathcal{E} := \exists F_{\iota\to\iota}.\forall X_\iota.F(fX) =^? f(FX)$$

We had claimed that $\mathbf{U}(\mathcal{E}) = \{\sigma_n := [\lambda Z. f^n Z/F] \mid n \in \mathbb{N}\}$.

Obviously the pair in $\mathcal{E}$ is flex/rigid, so we apply $\mathcal{U}\Lambda(\mathit{flex/rig})$ and have two choices (imitation and 1-projection). With the 1-projection binding $\mathcal{G}^1_{\iota\to\iota} = \lambda Z_\iota. Z$, we arrive at the first solution $\sigma_0 = [\lambda Z. f^0 Z/F] = [\lambda Z. Z/F]$. The imitation branch ($\mathcal{G}^f_{\iota\to\iota} = \lambda Z_\iota. f(H^1 Z)$) gives us the unification problem

$$\mathcal{E}_1 := \exists F_{\iota\to\iota}, H^1_{\iota\to\iota}. \forall X_\iota. F =^? \lambda Z. f(H^1 Z) \wedge H^1(fX) =^? f(H^1 X)$$

Obviously the unsolved pair $H^1(fX) =^? f(H^1 X)$ in $\mathcal{E}_1$ is just an alphabetical variant of the original pair in $\mathcal{E}$. Therefore, we have the same two choices as above. 1-projection will give us the solution $\sigma_1 = [\lambda Z_\iota. fZ/F]$ and imitation will start the cycle again, generating all solutions σ_i along the way.

For the soundness proof for $\mathcal{U}\Lambda$ we start out with some useful technical lemmata.

Lemma 3.5.5 *Let $\mathcal{E} \vdash_{\mathcal{U}\Lambda} \mathcal{E}'$ by a single application of $\mathcal{U}\Lambda(dec)$ to a pair $H\overline{\mathbf{U}^n} =^? H\overline{\mathbf{V}^n}$, then for any substitution θ we have*

1. *If $H \in \mathbf{supp}(\theta)$, then $\theta \in \mathbf{U}(\mathcal{E}')$ implies that $\theta \in \mathbf{U}(\mathcal{E})$.*
2. *If $H \notin \mathbf{supp}(\theta)$, then $\theta \in \mathbf{U}(\mathcal{E})$, iff $\theta \in \mathbf{U}(\mathcal{E}')$.*

Proof: Let $\theta \in \mathbf{U}(\mathcal{E}')$, so $\theta(\mathbf{U}^i) =_\beta \theta(\mathbf{V}^i)$ for all $1 \leq i \leq n$ and therefore

$$\theta(H\overline{\mathbf{U}^n}) \equiv \theta(H)\overline{\theta(\mathbf{U}^n)} =_\beta \theta(H)\overline{\theta(\mathbf{V}^n)} \equiv \theta(H\overline{\mathbf{V}^n})$$

Thus for any atom H we have $\theta \in \mathbf{U}(\mathcal{E})$. Now let $H \notin \mathbf{Dom}(\theta)$ and $\theta \in \mathbf{U}(\mathcal{E})$, then $\theta(H) = H$, so in this case we have $\theta \in \mathbf{U}(\mathcal{E}')$. □

In other words, $\mathcal{U}\Lambda(dec)$ does not preserve unifiers. This behavior is illustrated by the fact that $Xa =^? Xb$ has the unifier $[\lambda X. Y/X]$, while after decomposition with $\mathcal{U}\Lambda(dec)$, the problem $a =^? b$ is unsolvable, since a and b are distinct constants.

The other $\mathcal{U}\Lambda$ rules do not conserve unifiers either (otherwise, we would have incorporated them into simplification). By applying the rules $\mathcal{U}\Lambda(\mathit{flex/rig})$ and $\mathcal{U}\Lambda(\mathit{guess})$ we effectively commit ourselves to a particular approximation of a solution, and thus cannot reasonably expect to conserve the set of unifiers.

Lemma 3.5.6 *If $\mathcal{E} \vdash_{\mathcal{U}\Lambda} \mathcal{E}'$ by a $\mathcal{U}\Lambda$-derivation only containing applications of the rules $\mathcal{U}\Lambda(\mathit{flex/rig})$ and $\mathcal{U}\Lambda(\mathit{guess})$, then $\mathbf{U}(\mathcal{E}') \subseteq \mathbf{U}(\mathcal{E})$.*

Proof: The transformations $\mathcal{U}\Lambda(\mathit{flex/rig})$ and $\mathcal{U}\Lambda(\mathit{guess})$ can be divided into three parts, first adding a pair $X =^? \mathbf{G}$, then eliminating the variable, and finally $\mathcal{SIM}$-reducing. Clearly adding a new pair does not create new unifiers, so we must have $\mathbf{U}(\mathcal{E} \wedge X =^? \mathbf{A}) \subset \mathbf{U}(\mathcal{E})$. Thus obtain the assertion with 3.2.10 and 3.3.2. □

Theorem 3.5.7 (Soundness of $\mathcal{U}\Lambda$)
If $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{E}' \vdash_{\mathcal{U}\Lambda} \mathcal{F}$ such that $\mathcal{F}$ is in solved form, then the substitution $\sigma_{\mathcal{F}}|_{\mathbf{Dom}(\Gamma)} \in \mathbf{U}(\mathcal{E})$.

Proof: We prove $\sigma_{\mathcal{F}} \in \mathbf{U}(\mathcal{E})$ by induction on the length of the transformation sequence using the above lemmata in the induction step. The restriction of $\sigma_{\mathcal{F}}$ does not affect the fact that $\sigma_{\mathcal{F}}|_{\mathbf{Dom}(\Gamma)}$ still unifies $\mathcal{E}$. □

So if the algorithm $\mathcal{U}\Lambda$ returns a substitution θ for an initial system $\mathcal{E}$, then θ is indeed a unifier for $\mathcal{E}$. The main result of this section is the converse, namely, that given an initial unification problem $\mathcal{E}$ and a unifier θ, the algorithm $\mathcal{U}\Lambda$ can compute a unifier σ of $\mathcal{E}$, which is more general than θ.

As higher-order unification is undecidable [Gol81], our set of transformations cannot be terminating in general. We will prove, that $\mathcal{U}\Lambda$ is a complete unification procedure, that is, if for any given $\theta \in \mathbf{U}(\mathcal{E})$ there is a $\mathcal{U}\Lambda$-derivation $\mathcal{E} \vdash_{\mathcal{U}\Lambda} \mathcal{F}$ such that $\mathcal{F}$ is a unification problem in solved form, and $\sigma_{\mathcal{F}}$ is more general than θ. For this we only need termination for $\mathcal{U}\Lambda$ inference rules that approximate θ.

Definition 3.5.8 Let $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{E}'$ be a unification problem and θ be a substitution, we set

$$\mu(\mathcal{E},\theta) := (\mu_1(\mathcal{E},\theta), \mu_2(\mathcal{E}))$$

where $\mu_1(\mathcal{E},\theta)$ is the multiset of depths of $\theta(X)$, where $X \in \mathbf{Dom}(\theta)$ is unsolved in $\mathcal{E}$ and $\mu_2(\mathcal{E})$ is the multiset of depths of formulae in $\mathcal{E}$. Furthermore, let $\prec$ be the strict lexicographic ordering for the obvious component orderings.

The following measure provides the basis for defining the approximating rule applications and for proving their termination.

Lemma 3.5.9 (Semi-Termination for $\mathcal{U}\Lambda$)
Let $\mathcal{E}$ be an unsolved unification problem in $\mathcal{SIM}$-normal form and $\theta \in \mathbf{U}(\mathcal{E})$, then there exists a unification problem $\mathcal{E}'$ and a substitution θ', such that

$$\mathcal{E} \vdash_{\mathcal{U}\Lambda} \mathcal{E}' \qquad \theta \equiv \theta'[\mathcal{E}] \qquad \theta' \in \mathbf{U}(\mathcal{E}') \qquad \mu(\mathcal{E}',\theta') \prec \mu(\mathcal{E},\theta)$$

Proof: Let $\theta \in \mathbf{U}(\mathcal{E})$ and $\mathbf{A} =^? \mathbf{B}$ an unsolved pair in $\mathcal{E}$. As $\mathcal{E}$ is in $\mathcal{SIM}$-normal form we have $\mathbf{A} \equiv \mathbf{M}\overline{\mathbf{U}}$ and $\mathbf{B} \equiv \mathbf{N}\overline{\mathbf{V}}$, where $\mathbf{M}$ and $\mathbf{N}$ are symbols. Note that $\mathbf{M}$ and $\mathbf{N}$ cannot be equal constants, and moreover we cannot have $\mathbf{A} =_{\beta\eta} \mathbf{B}$.

If $\mathbf{M} = \mathbf{N}$ is a variable not in $\mathbf{supp}(\theta)$, then $\mathcal{U}\Lambda(dec)$ applies. By 3.5.5 we have $\theta \in \mathbf{U}(\mathcal{E}')$ and $\mu(\mathcal{E}',\theta) \prec \mu(\mathcal{E},\theta)$, since $\mu_1(\mathcal{E}',\theta) \preceq \mu_1(\mathcal{E},\theta)$ and $\mu_2(\mathcal{E}') \prec \mu(\mathcal{E})$.

Otherwise either $\mathbf{M} \neq \mathbf{N}$ or $\mathbf{M} = \mathbf{N} \in \mathbf{supp}(\theta)$. In both cases, since $\mathcal{E}$ is unifiable, either $\mathbf{M}$ or $\mathbf{N}$ is an unsolved variable $F_\alpha \in \mathbf{supp}(\theta)$. Without loss of generality we assume that $\mathbf{M} = F$ and set $h := \mathbf{head}(\theta(F))$. By the general binding theorem 3.4.2 there exists a general binding $\mathbf{G} \in \mathcal{A}_\alpha^h$ and a substitution ρ, such that $\rho(\mathbf{G}) =_{\beta\eta} \theta(F)$. Now,

- if $\mathbf{head}(\mathbf{N}) \notin \mathbf{supp}(\theta)$, then $\mathcal{U}\Lambda(\mathit{flex/rig})$ applies.
- if $\mathbf{head}(\mathbf{N}) \in \mathbf{supp}(\theta)$ then $\mathcal{U}\Lambda(\mathit{guess})$ applies.

In all these cases we set $\theta' := \theta \cup \rho$ and have $\theta = \theta'[\mathcal{E}]$, since $\mathbf{supp}(\rho)$ consists of new variables and $\theta' \in \mathbf{U}(\mathcal{E}')$ by 3.5.7.

Now by the general binding theorem, $\mathbf{dp}(\rho) < \mathbf{dp}(\theta(F))$, furthermore, $\mathcal{U}\Lambda(\mathit{guess})$ and $\mathcal{U}\Lambda(\mathit{flex/rig})$ remove the variable F from the set of variables in $\mathbf{supp}(\theta)$ that are not solved in $\mathcal{E}$, but adds $\mathbf{supp}(\rho)$ to the set, therefore we have $\mu_1(\mathcal{E}', \theta') \prec \mu_1(\mathcal{E}, \theta)$. Thus we have $\mu(\mathcal{E}', \theta') \prec \mu(\mathcal{E}, \theta)$. □

If we call such a transformation μ-**prescribed**, then each application of a μ-prescribed transformation decreases the well-founded measure μ. Thus any sequence of μ-prescribed transformations must terminate. The previous lemma also guarantees that any system obtained by exhaustively applying μ-prescribed transformations to a unifiable system must be solved, since otherwise it guarantees another μ-prescribed transformation.

Corollary 3.5.10 *If $\mathcal{E}$ is a unifiable unification problem such that no μ-prescribed transformation rule from $\mathcal{U}\Lambda$ is applicable, then $\mathcal{E}$ is in solved form.*

Theorem 3.5.11 (Completeness Theorem for $\mathcal{U}\Lambda$)
For any unification problem $\mathcal{E}$ and any substitution $\theta \in \mathbf{U}(\mathcal{E})$, there is a $\mathcal{U}\Lambda$-derivation $\mathcal{E} \vdash_{\mathcal{U}\Lambda} \mathcal{F}$ such that $\mathcal{F}$ is in solved form and $\sigma_\mathcal{F} \preceq_{\beta\eta} \theta[\mathcal{E}]$.

Proof: Let $\mathcal{D}: \mathcal{E} \vdash_{\mathcal{U}\Lambda} \mathcal{F}$ be a maximal μ-prescribed $\mathcal{U}\Lambda$-derivation out of $\mathcal{E}$. By 3.5.9 this is always finite, so we can prove the assertion by induction on the number n of nodes in $\mathcal{D}$. If $n = 0$, then $\mathcal{E}$ is in solved form and $\sigma_\mathcal{E}$ is a most general unifier for $\mathcal{E}$. In particular, we have $\sigma_\mathcal{E} \preceq_{\beta\eta} \theta[\mathcal{E}]$.

If $n > 0$, then there is a μ-prescribed inference $\mathcal{E} \vdash \mathcal{E}'$ and a substitution θ' satisfying 3.5.9. By inductive hypothesis there is a $\mathcal{U}\Lambda$-derivation $\mathcal{E}' \vdash_{\mathcal{U}\Lambda} \mathcal{F}$ such that $\sigma_\mathcal{F} \preceq_{\beta\eta} \theta'[\mathcal{E}']$. By 3.5.7 we have $\sigma_\mathcal{F} \in \mathbf{U}(\mathcal{E}') \subseteq \mathbf{U}(\mathcal{E})$. Furthermore, by inspection of the inference rules we see that $\mathcal{U}\Lambda$ rules only expand the set of existential variables in $\mathcal{E}$, so $\sigma_\mathcal{F} \preceq_{\beta\eta} \theta'[\mathcal{E}']$ implies $\sigma_\mathcal{F} \preceq_{\beta\eta} \theta'[\mathcal{E}]$, which in turn yields the assertion since $\theta' = \theta[\mathcal{E}]$ by 3.5.9. □

If we combine the soundness results theorem 3.5.7 with the completeness result from theorem 3.5.11, we can characterize the set of solutions found by the algorithm $\mathcal{U}\Lambda$ by the following corollary.

Corollary 3.5.12 *For any unification problem $\mathcal{E}$ the set*

$$\mathcal{U}\Lambda(\mathcal{E}) := \{\sigma_{\mathcal{F}} \mid \mathcal{E} \vdash_{\mathcal{U}\Lambda} \mathcal{F} \text{ and } \mathcal{F} \text{ is in solved form}\}$$

is a complete set of unifiers for $\mathcal{E}$.

3.6 Pre-Unification ($\mathcal{P}\Lambda$)

pre-equality
pre-unifier
unifier!pre
$\mathbf{PU}(\mathcal{E})$*set of pre-unifiers for $\mathcal{E}$*
wff:pre:topinference rules for pre-equality
$=^p$*pre-equality*
$\Gamma \Vdash^p_\Sigma \mathcal{E}\mathcal{E}$ *is pre-unifiable*
$\mathcal{E}_p$*subsystem of pre-solved pairs in $\mathcal{E}$*
pre-solved
solved!pre
pre-solved form
solved form!pre-solved form!pre-solved

The rule $\mathcal{U}\Lambda(guess)$ gives rise to a serious explosion of the search space for unifiers, which makes general higher-order unification in this form impractical. Huet's solution to this problem was to redefine the higher order unification problem to a form sufficient for refutation purposes: for the pre-unification problem flex/flex pairs are considered already solved, since they can always be trivially solved by binding the head variables to special constant functions that identify the formulae by absorbing their arguments.

Definition 3.6.1 (Pre-Equality)
Let $\mathbf{A} =^p_{\beta\eta} \mathbf{B}$ be the **pre-equality** judgment defined by the inference system for typed $\beta\eta$-equality augmented by the following inference rule

$$\frac{F, G \in \mathcal{V}}{F\overline{\mathbf{U}^n} =^p G\overline{\mathbf{V}^k}} \; \mathit{wff{:}pre{:}top}$$

Let $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{E}'$ be a unification problem, then we call a substitution σ a **pre-unifier** of the pair $\mathbf{A} =^? \mathbf{B} \in \mathcal{E}'$, iff $\sigma(\mathbf{A}) =^p_{\beta\eta} \sigma(\mathbf{B})$. We denote the set of pre-unifiers by $\mathbf{PU}(\mathcal{E})$.

Definition 3.6.2 (Pre-Solved Form)
Let $\mathcal{E}$ be a unification problem, then we call a pair $F\overline{\mathbf{U}^k} =^? G\overline{\mathbf{V}^n}$ **pre-solved** in $\mathcal{E}$, iff F and G are variables. A unification problem $\mathcal{E}$ is in **pre-solved form**, iff all pairs in $\mathcal{E}$ are solved or pre-solved. In this case we write $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{E}_\sigma \wedge \mathcal{E}_p$, where the pairs in $\mathcal{E}_p$ are pre-solved, but unsolved.

Remark 3.6.3 (Trivial Unifiers for Pre-Solved Pairs)
Let $\mathcal{P} := F_\alpha\overline{\mathbf{U}^n} =^? G_\beta\overline{\mathbf{V}^m}$ be a pre-solved pair with $\alpha = \overline{\alpha^n} \to \gamma$, $\beta = \overline{\beta^m} \to \gamma$, then

$$\sigma_{\mathcal{P}} := [\lambda X^1_{\alpha^1} \ldots X^n_{\alpha^n}.H/F], [\lambda X^1_{\beta^1} \ldots X^m_{\beta^m}.H/G]$$

is a unifier for $\mathcal{P}$, since $\sigma_{\mathcal{P}}(F\overline{\mathbf{U}^n}) =_{\beta\eta} H =_{\beta\eta} \sigma_{\mathcal{P}}(G\overline{\mathbf{V}^m})$.

Thus pre-unifiers can always be extended to unifiers by finding trivial unifiers for the pre-solved pairs. Let $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{E}_\sigma \wedge \mathcal{E}_p$ be a unification problem in pre-solved form, then we can construct a unifier θ for $\mathcal{E}_p$ as above, and see that $\theta \cup \sigma$ is a unifier of $\mathcal{E}$. Therefore a unification problem $\mathcal{E}$ is pre-unifiable, iff it is unifiable.

$\mathcal{P}\Lambda$ transformations for pre-unification
transformation!pre-unification
pre-unification!transformation

Definition 3.6.4 ($\mathcal{P}\Lambda$: Transformations for Pre-Unification)
We define the set $\mathcal{P}\Lambda$ of transformations for pre-unification by modifying $\mathcal{U}\Lambda$ (3.5.1): the inference rules of $\mathcal{P}\Lambda$ are obtained from their $\mathcal{U}\Lambda$ counterparts by requiring that they may not be performed on a pre-solved pair. Thus the rules $\mathcal{U}\Lambda(dec)$ and $\mathcal{U}\Lambda(guess)$ are never applicable, and we only have $\mathcal{U}\Lambda(\mathit{flex}/\mathit{rig})$ left.

With this definition we immediately obtain the soundness and completeness of $\mathcal{P}\Lambda$. For the completeness result we need an analogue of the termination lemma (3.5.9) which can be obtained by observing that the restriction of the $\mathcal{U}\Lambda$ inference rules to $\mathcal{P}\Lambda$ inference rules by forbidding applications of $\mathcal{U}\Lambda(\mathit{flex}/\mathit{rig})$ to flex-flex-pairs does not affect the result.

Lemma 3.6.5 (Semi-Termination for $\mathcal{P}\Lambda$)
Let $\mathcal{E}$ be an unification problem in $\mathcal{SIM}$-normal form that is not pre-solved and $\theta \in \mathbf{PU}(\mathcal{E})$, then there exists a unification problem $\mathcal{E}'$ and a substitution θ', such that

$$\mathcal{E} \vdash_{\mathcal{U}\Lambda} \mathcal{E}' \qquad \theta \equiv \theta'[\mathcal{E}] \qquad \theta' \in \mathbf{U}(\mathcal{E}') \qquad \mu(\mathcal{E}', \theta') \prec \mu(\mathcal{E}, \theta)$$

Proof: Let $\mathcal{E} \in \mathbf{U}(\mathcal{E})$ and $\mathbf{A} {=^?} \mathbf{B}$ a pair in $\mathcal{E}$ that is not pre-solved. As $\mathcal{E}$ is in $\mathcal{SIM}$-normal form we can without loss of generality assume that $\mathbf{A} \equiv F\overline{\mathbf{U}}$, where $F_\alpha \in \mathbf{supp}(\theta)$ is an unsolved variable and $\mathbf{B}$ is rigid. By the general binding theorem 3.4.2 there exists a general binding $\mathbf{G} \in \mathcal{A}_\alpha^{\mathbf{head}(\theta(F))}$ and a substitution ρ, such that $\rho(\mathbf{G}) {=_{\beta\eta}} \theta(F)$. Thus $\mathcal{U}\Lambda(\mathit{flex}/\mathit{rig})$ applies, since $\mathbf{N}$ is rigid. In all these cases we set $\theta' := \theta \cup \rho$ and have $\theta = \theta'[\mathcal{E}]$, since $\mathbf{supp}(\rho)$ consists of new variables and $\theta' \in \mathbf{U}(\mathcal{E}')$ by Theorem 3.5.7.

The depth condition can be shown just as in the case for $\mathcal{U}\Lambda$ (cf. Lemma 3.5.9). $\square$

Theorem 3.6.6 (Soundness and Completeness for $\mathcal{P}\Lambda$)
For any unification problem $\mathcal{E}$ and any $\theta \in \mathbf{PU}(\mathcal{E})$, there is a $\mathcal{P}\Lambda$-derivation $\mathcal{E} \vdash_{\mathcal{P}\Lambda} \mathcal{F}$ such that $\mathcal{F}$ is in pre-solved form, $\sigma_\mathcal{F} \in \mathbf{PU}(\mathcal{E})$ and $\sigma_\mathcal{F} \preceq^p_{\beta\eta} \theta[\mathcal{E}]$.

Proof sketch: Soundness is immediate, since all $\mathcal{P}\Lambda$-transformations are special cases of the $\mathcal{U}\Lambda$-transformations, so the assertion can be obtained with the same methods as 3.5.7.

The rest of the proof is completely analogous to that in of 3.5.11 using Lemma 3.6.5 instead of 3.5.9 $\square$

By a close inspection of the rules, we give an even finer characterization of the $\mathcal{P}\Lambda$ transformations and the higher-order pre-unification problem in the unification hierarchy (cf. Definition 3.2.6).

Theorem 3.6.7 (Pre-Unification is Infinitary)
The Higher-order pre-unification problem is of type infinitary in the unification hierarchy. $\mathcal{P}\Lambda$ is a complete, minimal pre-unification procedure. That

is: if $\mathcal{E}$ is a unification problem, then the set $\mathcal{P}\Lambda(\mathcal{E}) := \{\sigma \mid \mathcal{E} \vdash_{\mathcal{P}\Lambda} \mathcal{E}_\sigma\}$ is a complete set of most-general pre-unifiers for $\mathcal{E}$.

natural language
language!natural
linguistics
semantics!natural language

Proof: The fact that $\mathcal{P}\Lambda$ is a complete higher-order pre-unification procedure is an immediate consequence of Theorem 3.6.6. Minimality is a consequence of the fact that in the search tree for pre-unifiers, at any $\mathcal{U}\Lambda$(*flex/rig*) node (these are the only ones with branching, since the $\mathcal{SIM}$ transformations are deterministic) we only have the choice between the imitation- and possibly some projection bindings. Since the sets of instances are disjoint (they have different heads), the solutions must be incomparable, and as a consequence most general.

As we have seen in Example 3.2.5, higher-order (pre)-unification is at least infinitary. Since for any unification problem $\mathcal{E}$ the set $\mathcal{P}\Lambda(\mathcal{E})$ is countable (it is produced as the success leaves of a (possibly infinite) tree), and all $\theta \in \mathcal{P}\Lambda(\mathcal{E})$ are minimal, it cannot be nullary. □

3.7 HOU in Natural Language Semantics

As we have seen above, higher-order unification is an inference procedure that allows to solve equations containing function and predicate variables. In Example 3.5.3, higher-order unification was used to construct a function — the representation of a natural number. In this section we will present a different kind of application of higher-order unification, where a predicate is constructed: Dalrymple, Shieber and Pereira have pioneered higher-order unification as an inference procedure for the reconstruction of linguistically unspecified material [DSP91a] in a Montegovian framework of natural language semantics (see section 1.2.3).

We will briefly sketch the underlying ideas in case of verb-phrase ellipsis; i.e. the phenomenon that parts of natural language sentences (here verb phrases) can be replaced by utterances like “does too”. For a thorough treatment of cases like focus constructions, second-occurrence expressions, and adverbial quantification, the reader is referred to [GK96b] and the references in [GKK99].

The basic idea [DSP91b] underlying the use of higher-order unification for natural language semantics is very simple: Following [Mon74], the simply typed λ-calculus is used as a semantic representation language while semantically under-specified elements (e.g. anaphoric references or ellipses) are represented by free variables whose value is determined by solving higher-order equations. For instance, the discourse “*Dan likes his wife. Peter does too*” has has the semantic representation

$$\mathbf{S} := \mathsf{like}(\mathsf{dan}, \mathsf{w\_of}(\mathsf{dan})) \wedge R(\mathsf{peter})$$

where the value of the predicate variable R is determined by the equation

$$\mathsf{like}(\mathsf{dan}, \mathsf{w\_of}(\mathsf{dan})) =^? R(\mathsf{dan})$$

sloppy reading
strict

The fact that higher-order equations can have more than one most general solution is used to account for ambiguity, in this case the so-called sloppy/strict ambiguity Higher-order unification (see Figure 3.3 for a trace

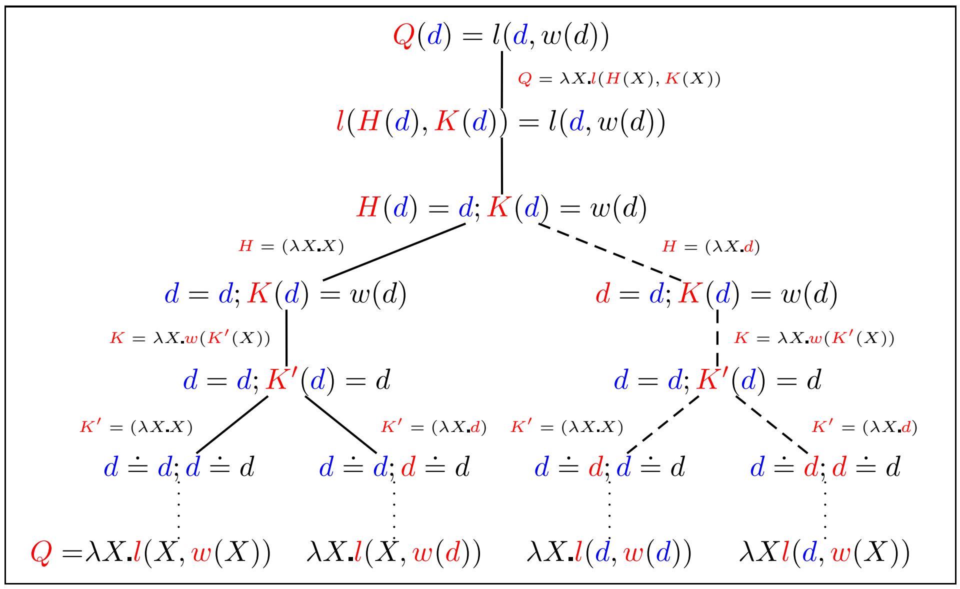


Figure 3.3: Computing the sloppy/strict Ambiguity

of the algorithm) calculates the solutions

$$\sigma_1 := [\lambda X.\mathsf{like}(X, \mathsf{w\_of}(X))/R] \quad \text{and} \quad \sigma_2 := [\lambda X.\mathsf{like}(X, \mathsf{w\_of}(\mathsf{dan}))/R]$$

which correspond to the intended **sloppy reading** *Peter loves Peter's wife* and the **strict** reading *Peter loves Dan's wife*

$$\sigma_1(\mathbf{S}) = \mathsf{like}(\mathsf{dan}, \mathsf{w\_of}(\mathsf{dan})) \wedge \mathsf{like}(\mathsf{peter}, \mathsf{w\_of}(\mathsf{peter}))$$
$$\sigma_2(\mathbf{S}) = \mathsf{like}(\mathsf{dan}, \mathsf{w\_of}(\mathsf{dan})) \wedge \mathsf{like}(\mathsf{peter}, \mathsf{w\_of}(\mathsf{dan}))$$

However, higher-order unification also yields two other solutions

$$\sigma_3 := [\lambda Z.\mathsf{like}(\mathsf{dan}, \mathsf{w\_of}(\mathsf{dan}))/R] \quad \text{and} \quad \sigma_4 := [\lambda Z.\mathsf{like}(\mathsf{dan}, \mathsf{w\_of}(Z))/R]$$

that are clearly not the desired readings of the discourse:

$$\sigma_3(\mathbf{S}) = \mathsf{like}(\mathsf{dan}, \mathsf{w\_of}(\mathsf{dan})) \wedge \mathsf{like}(\mathsf{dan}, \mathsf{w\_of}(\mathsf{dan}))$$
$$\sigma_4(\mathbf{S}) = \mathsf{like}(\mathsf{dan}, \mathsf{w\_of}(\mathsf{dan})) \wedge \mathsf{like}(\mathsf{dan}, \mathsf{w\_of}(\mathsf{peter}))$$

To remedy this shortcoming, Dalrymple, Shieber, and Pereira, propose an informal restriction, the **primary occurrence restriction**, which deletes any solution from the set of linguistically valid solutions, which contains a pre-determined so-called **primary occurrence** (in our case the first occurrence of dan in **S**). In [GK96b, Gar97, GK97, GKK99] a variant of $\Lambda^{\rightarrow}$, the **colored λ-calculus** [HK00] that allows color annotations to symbol occurrences is used as a tool to specify the interface between the classical semantic construction process (using higher-order unification) and other sources of linguistic information (which are coded into color information).

primary occurrence
occurrence!primary
colored λ-calculus
λ-calculus!colored
patterns
higher-order patterns
higher-order pattern
pattern!higher-order
pattern unification problem
unification problem!pattern
pattern substitution
substitution!pattern

This approach has been successfully been extended to other parallelism-based phenomena [Gar97], focus constructions [GK96a, Pul97] and finally corrections [GKvL96], which combine focus and parallelism (see section 5.6)

3.8 Higher-Order Patterns

There are certain syntactic fragments of $\Lambda^{\rightarrow}$, where the higher-order unification problem has better properties than in the general case. We will concentrate on two fragments, that of *higher-order patterns*, where the problem is unitary and that of *higher-order matching*, where one side of each pair in unification problems is closed (i.e does not contain existential variables).

We first specialize the higher-order unification algorithm to higher-order patterns.

Definition 3.8.1 (Higher-Order Pattern)
We call a formula $\mathbf{A} \in \mathit{wff}(\Sigma)$ a **higher-order pattern**, iff any occurrence of a free variable F in $\mathbf{A}$ must be in a subformula $\mathbf{B}$ of $\mathbf{A}$ of the form $FX^1 \ldots X^n$, where the X^i are bound in $\mathbf{A}$. In other words, all free variables of a higher-order pattern occur at the leaves, or applied to a list of distinct bound variables.

We will call a formula $\mathbf{A}$ in a unification problem $\mathcal{E} := \exists\Gamma\mathbf{.}\forall\Upsilon\mathbf{.}\mathcal{E} \wedge \mathbf{A} =^? \mathbf{B}$ a higher-order pattern, iff the condition holds for $F \in \mathbf{Dom}(\Gamma)$ and X^i bound in $\mathbf{A}$ or in $\mathbf{Dom}(\Upsilon)$; i.e. the universally quantified variables from the declaration are also allowed for the as the arguments for the existential ones (which take the rôle of the free ones). We will call $\mathcal{E}$ a **pattern unification problem**, iff all formulae occurring in its pairs are higher-order patterns.

Finally, we will call a substitution $\sigma \in \mathbf{SUB}$ a **pattern substitution**, iff for all X in $\mathbf{Dom}(\sigma)$, $\sigma(X)$ is a higher-order pattern.

Example 3.8.2 Let f, a be constants and F, G be variables of appropriate type, then $\lambda XYZ\mathbf{.}FZX$ and $\lambda X\mathbf{.}f(\lambda Y\mathbf{.}FXY)a(GX)$ are higher-order patterns, while Fa, $\lambda X\mathbf{.}FXa$, $\lambda X\mathbf{.}FXX$, and $\lambda XY\mathbf{.}FX(YX)$ are not. Furthermore, all first-order formulae and all closed formulae are higher-order patterns, since they do not contain free function variables. Finally, rigid general bindings are higher-order patterns, while flexible are not in general.

For the analysis of higher-order patterns we will often use the following notion:

partial permutation
permutation!partial
partial permutation
binding context
context!binding
subterm-closed
closed!subterm

Definition 3.8.3 (Partial Permutation)
We say that φ is a **partial permutation** from k into n, if φ is an injective mapping from $\{1, \ldots, k\}$ into $\{1, \ldots, n \geq k\}$.

Lemma 3.8.4 *The class of higher-order patterns is closed under the application of pattern substitutions and β-reduction.*

Proof: We will only summarize the main idea of the proof, which can be made formal by a simple simultaneous induction over the structure of the formulae involved.

For the substitution case let σ be a pattern substitution with $F \in \mathbf{Dom}(\sigma)$ and $\sigma(F) = \lambda\overline{Y^n}.\mathbf{C}$. We consider a subformula $\mathbf{B} = FX^{\varphi(1)} \ldots XF^{\varphi(n)}$ of a higher-order pattern $\mathbf{A}$. Then $\sigma(\mathbf{B}) = (\lambda\overline{Y^n}.\mathbf{C})\overline{X^{\varphi(n)}}$ which is obviously a higher-order pattern (only the variable F has been eliminated). Furthermore, β-reduction of $\sigma(\mathbf{B})$ will only replace the variables Y^i in $\mathbf{C}$ that were bound in $\sigma(F)$ for the X^i that are bound in $\mathbf{A}$.

The assertion for β-reduction is now a simple consequence of the observation that in a β-redex $\mathbf{A} := (\lambda X.\mathbf{B})\mathbf{C}$ in a higher-order pattern, $\mathbf{C}$ must be pattern as well (the binding contexts of $\mathbf{A}$ and $\mathbf{B}$ are the same), so $[\mathbf{C}/X]$ is a pattern substitution. $\square$

Remark 3.8.5 (Patterns are not Subterm-Closed)
Note that the property of being a higher-order pattern is not **subterm-closed**, i.e. subformulae of higher-order patterns are in general not again higher-order patterns. Consider for instance $\lambda X.Xa$ for some constant a, where the subterm Xa is not a higher-order pattern, since the free variable X is applied to a constant.

The syntactic fragment of higher-order patterns allows to specialize the unification rules from Figure 3.2.

Definition 3.8.6 (Transformations for Pattern Unification)
The inference rules for $\mathcal{U}$Pat are those of $\mathcal{U}\Lambda$, where the $\mathcal{U}\Lambda$(*guess*) rule is is replaced by the flex/flex rules in Figure 3.4. The first one handles the case, where the heads are identical and the second one, where they are distinct.

Note that these rules always transform pattern unification problems to ones that are patterns: for the simplification rules that is straightforward (even decomposition does not make a problem, even though patterns are not subterm closed). In the flex/rigid case conservation is a consequence of the fact that rigid approximating bindings are higher-order patterns, and the flex/flex case, where $\mathcal{U}\Lambda$ would have used flexible bindings is treated by $\mathcal{U}$Pat(*same*) and $\mathcal{U}$Pat(*diff*), which conserves higher-order patterns by construction.

Soundness of these rules immediately comes from the fact that they are specializations of the $\mathcal{U}\Lambda$ rules. We will now show completeness.

$\Upsilon = [X^1{:}\,\alpha_1],\ldots,[X^n{:}\,\alpha_n]$, $\Gamma(F) = \overline{\alpha_k} \to \beta$, $\Gamma(G) = \overline{\alpha_l} \to \beta$
φ, ψ partial permutations from k to n; χ from l to n.

$$\frac{\exists\Gamma.\forall\Upsilon.\mathcal{E} \wedge F\overline{X^{\varphi(k)}} =^? F\overline{X^{\psi(k)}}}{\exists\Gamma, [H{:}\,\overline{\alpha_{\rho(l)}} \to \beta].\forall\Upsilon.\mathcal{E} \wedge \begin{array}{l} F\overline{X^{\varphi(k)}} =^? F\overline{X^{\psi(k)}} \wedge \\ F =^? \lambda\overline{Y^k_{\alpha_{\varphi(k)}}}.H\overline{Y^{\rho(l)}} \end{array}} \,\mathcal{U}\mathrm{Pat}(\mathit{same})$$

ρ is a partial permutation from k to l, such that $\rho(i) = \varphi(j)$, iff $\varphi(j) = \psi(j)$ i.e. ρ picks out all arguments, where φ and ψ coincide.

$$\frac{\exists\Gamma.\forall\Upsilon.\mathcal{E} \wedge F\overline{X^{\varphi(k)}} =^? G\overline{X^{\chi(l)}}}{\exists\Gamma, [H{:}\,\overline{\alpha_{\varphi'(m)}} \to \beta].\forall\Upsilon.\mathcal{E} \wedge \begin{array}{l} F\overline{X^{\varphi(k)}} =^? G\overline{X^{\chi(l)}} \wedge \\ F =^? \lambda\overline{Y^k_{\alpha_{\varphi(k)}}}.H\overline{Y^{\varphi'(m)}} \wedge \\ G =^? \lambda\overline{Y^l_{\alpha_{\chi(l)}}}.H\overline{Y^{\chi'(m)}} \end{array}} \,\mathcal{U}\mathrm{Pat}(\mathit{diff})$$

φ' and χ' are partial permutations from m into k, l, such that $\varphi'(m) = i$ and $\chi'(m) = j$, iff $\varphi(i) = \chi(j)$.

Figure 3.4: $\mathcal{U}$Pat transformations for higher-order pattern unification

Lemma 3.8.7 *If $\mathcal{E}$ is a pattern unification problem and $\mathcal{E} \vdash_{\mathcal{U}\mathrm{Pat}} \mathcal{F}$, then* $\mathbf{UPat}(\mathcal{E}) = \mathbf{UPat}(\mathcal{F})$.

Proof: The assertion can be verified by a simple inspection of the rules. For $\mathcal{SIM}$ we have already verified this in 3.3.2. Furthermore for higher-order patterns the $\mathcal{U}\Lambda(\mathit{flex}/\mathit{rig})$ rule is deterministic, that is, all but the imitation or one projection immediately lead to failure. Thus by Lemma 3.5.9 $\mathcal{U}\Lambda(\mathit{flex}/\mathit{rig})$ must conserve the set of unifiers.

For $\mathcal{U}$Pat(same) we have $\mathcal{E} = \exists\Gamma.\forall\Upsilon.F\overline{X^{\varphi(k)}} =^? F\overline{X^{\psi(l)}} \wedge \mathcal{E}$, where $\Upsilon = [X^1{:}\,\alpha_1],\ldots,[X^n{:}\,\alpha_n]$, $\Gamma(F) = \overline{\alpha_k} \to \beta$, and φ, ψ are partial permutations from k to n. Now, for any solution $\sigma \in \mathbf{UPat}(\mathcal{E})$ we have $\sigma(F){=_{\beta\eta}}\lambda\overline{Y_k}.\mathbf{A}$, since σ is $\beta\eta$-long. Now let us assume that $\mathbf{A}$ contains an occurrence of Y_i with $\varphi(i) \neq \psi(i)$, then $\sigma(F)\overline{X^{\varphi(k)}}{=_{\beta\eta}}\overline{[X^{\varphi(k)}/Y^k]}\mathbf{A}$ and $\sigma(F)\overline{X^{\psi(k)}}{=_{\beta\eta}}\overline{[X^{\psi(k)}/Y^k]}\mathbf{A}$, so these differ there. This contradicts the assumption that σ is a unifier of $\mathcal{E}$. Thus $\mathbf{A}$ can only contain occurrences of Y^i where $\varphi(i) = \psi(i)$ and therefore is an instance of $H\overline{Y^{\rho(l)}}$ as in $\mathcal{U}$Pat(same).

Finally, for $\mathcal{U}$Pat(diff) we use a similar argumentation with $\sigma(F){=_{\beta\eta}}\lambda\overline{Y_k}.\mathbf{A}$ and $\sigma(G){=_{\beta\eta}}\lambda\overline{Z_l}.\mathbf{B}$. Since σ is a unifier of $\mathcal{E}$, we have

$$\overline{[X^{\varphi(k)}/Y^k]}\mathbf{A}{=_{\beta\eta}}\sigma(F)\overline{X^{\varphi(k)}}{=_{\beta\eta}}\sigma(G)\overline{X^{\psi(l)}}{=_{\beta\eta}}\overline{[X^{\psi(l)}/Z^l]}\mathbf{B}$$

so in particular if Y_i occurs in $\mathbf{A}$, then Z_j must occur in $\mathbf{B}$ at the same

~~restricted~~ *unification problem!restricted*

position for some i, j, where $\varphi(i) = \psi(j)$, and vice versa. If any other Y_i and Z_j would occur, then the terms would differ. Furthermore, $\mathbf{A}$ and $\mathbf{B}$ must be equal up to these differences. This concludes the proof of the assertion. □

We will now present the termination proof from [Qia96], since we will need it for the generalization to the sorted case in 9.3. Since there is no direct proof that uses well-founded measure on the set of unification problems, we will first establish some technical infrastructure.

Definition 3.8.8 (Restricted)
We call a pattern unification problem $\mathcal{E}$ **restricted**, iff $\mathbf{A}^i = \lambda\overline{X^n}.F^i\overline{Y^n}$, such that

- $F^i \in \mathbf{Dom}(\Gamma)$ are distinct
- $X^i \neq Y^i$ and $\{F^1, \ldots, F^n\} \cap \bigcup_{i=1,\ldots,n} \mathbf{Free}(\mathbf{A}^i) = \emptyset$

For a unification problem $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathbf{A}^1 =^? \mathbf{B}^1 \wedge \ldots \mathbf{A}^n =^? \mathbf{B}^n$ let $\mu_1(\mathcal{E})$ be the number of unsolved free variables in $\mathcal{E}$ and $\mu_2(\mathcal{E}) := \sum_{k=1}^{n}(\mu(\mathbf{A}^k) + \mu(\mathbf{B}^k))$, where $\mu_2(\mathbf{A})$ is the number of occurrences of function symbols and bound variables that are not arguments to free variables. Finally, let
EdNote(19) $\mu(\mathcal{E}) := (\mu_1(\mathcal{E}), \mu_2(\mathcal{E}))$.[19]

Lemma 3.8.9 1. *Transformations $\mathcal{U}\Lambda(dec)$ and $\mathcal{U}\mathrm{Pat}(same)$ are not applicable to restricted unification problems.*

2. *Transformations $\mathcal{U}\Lambda(elim)$ and $\mathcal{U}\mathrm{Pat}(same)$ conserve restrictedness of unification problems.*

3. *A sequence of transformations $\mathcal{U}\Lambda(elim)$ and $\mathcal{U}\mathrm{Pat}(same)$ issuing from a restricted unification problem must always terminate.*

EdNote(20) **Proof:** bla[20] □

Lemma 3.8.10
Let $\mathcal{E} \vdash_{\mathcal{U}\mathrm{Pat}} \mathcal{F}$, where $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\lambda\overline{X^n}.F\overline{Y^n} =^? \mathbf{B}$ is restricted and $\mathcal{F} = \mathcal{E}_\sigma$ is solved, then $\mathbf{Dom}(\sigma)$ contains only F, $\mathbf{Free}(\mathbf{B})$ and new variables H^j. Furthermore, $\sigma(G) = \lambda\overline{Z}.H\overline{Z'}$ for all $G \in \mathbf{Dom}(\sigma) \cap \mathbf{Free}(\mathbf{A})$.

EdNote(21) **Proof:** bla[21] □

Theorem 3.8.11 (Termination of $\mathcal{U}\mathrm{Pat}$)
There are no infinite $\mathcal{U}\mathrm{Pat}$-derivation.

EdNote(22) **Proof:** bla[22] □ *decidable unitary*

Theorem 3.8.12 (Completeness for $\mathcal{U}$Pat)
Let $\mathcal{E}$ be an unification problem, then $\mathcal{U}$Pat is terminating and confluent from $\mathcal{E}$ and yields an irreducible problem $\mathcal{F}$, such that either

- *$\mathcal{F}$ is solved and $\sigma_{\mathcal{F}}$ is a most general unifier of $\mathcal{E}$ or*
- *$\mathcal{F}$ is not solved and $\mathcal{E}$ is not unifiable.*

Proof: We have proven termination in the lemma above. Confluence can be verified by a simple inspection of the rules. For $\mathcal{SIM}$ we have already verified this in 3.3.5. Furthermore for higher-order patterns the $\mathcal{U}\Lambda(\mathit{flex}/\mathit{rig})$ rule is deterministic, that is, all but the imitation or one projection immediately lead to failure. Finally, the rules $\mathcal{U}$Pat(*diff*) and $\mathcal{U}$Pat(*same*) have disjoint accessibility conditions and cannot lead to divergence. □

Now the completeness result above directly yields the fact that there can be at most one most general solution to a pattern unification problem.

Corollary 3.8.13 *Higher-order pattern unification is decidable and unitary, i.e. pattern unification problems have at most one most general unifier.*

3.9 Second-Order Matching

It is not known whether the higher-order matching problem in typed λ-calculi is decidable in general. However, there are large subclasses where matching is known to be decidable, for instance, the class of higher-order patterns (this is a simple consequence of Theorem 3.8.13 or that of third- and fourth-order formulae [Dow92, Pad96]. Since the last result is rather involved, we will only present the case of second-order matching from [Hue76].[23] EdNote(23)

[19]EdNote: move somewhere appropriate
[20]EdNote: coming up
[21]EdNote: coming up
[22]EdNote: coming up
[23]EdNote: Aus Christian Prehofer's Diss kopieren.

Skolem function
Skolem
function!Skolem
choice function
function!choice
Andrews

Chapter 4

Witnesses, Skolemization, and Variable Conditions

In this chapter we review techniques for dealing with variable dependencies introduced by quantification in refutation calculi. The default mechanism for doing this in first-order logic is Skolemization [Sko19]: a closed quantified formula $\forall\overline{X^n}.\exists Y.\mathbf{A}$ is replaced by a formula $\forall\overline{X}.[f\overline{X^n}/Y]\mathbf{A}$, where f is a new function constant (the so-called **Skolem function**), which makes the dependency implicit in the mixed quantifier prefix explicit. Of course only those variables X that really occur in $\mathbf{A}$ matter, so another form of Skolemization replaces formulae a $\exists Y.\mathbf{A}$ with $[f\overline{X}/Y]\mathbf{A}$, where $\{X^1,\ldots,X^n,Y\}$ is the set of free variables in $\mathbf{A}$.

In higher-order logic where we have access to function variables, we can also use existentially quantified variables instead of function constants (recall that we work in a refutation setting) and replace $\exists Y_\beta.\mathbf{A}$ with $\exists F_{\overline{\alpha_n}\to\beta}.[F\overline{X}/Y_\beta]\mathbf{A}$, if the X^i have types α_i instead. Instead of getting rid of an existential quantifier, we have reduced a potentially problematic $\forall\exists$ prefix to a much more well-behaved $\exists\forall$-prefix.

We still have another way of higher-order Skolemization built on a similar intuition: recall that in higher-order abstract syntax (cf 1.2.3) $\exists Y_\beta.\mathbf{A}$ is an abbreviation of $\amalg^\beta\mathbf{B}$ with $\mathbf{B} := \lambda Y_\beta.\mathbf{A}$, i.e the set of objects b, such that $[\mathsf{b}/X]\mathbf{A}$ holds. Now, the predicate $\amalg^\beta$ holds, if the set $\mathbf{B}$ is non-empty, so we can replace $\amalg^\beta\mathbf{B}$ with $\mathbf{B}(f\mathbf{B})$, where f is a choice function (cf. 1.2.8) that returns some such b in $\mathbf{B}$. This Skolemization rule was proposed by Peter Andrews in [And71].

In all approaches the dependency of Y on the values of the universal variable X^i is coded into a function f. In first-order logic, this transformation of formulae preserves satisfiability and is therefore sufficient for refutation theorem proving. Note that it does not conserve models, since the existential is in effect interpreted as unique: the function f can only return a single value.

Unfortunately naive applications of all of these Skolemization approaches are not sound for higher-order logic [And73] – or for many other non-classical logics: it is possible to prove instances of the axiom of choice using naive Skolemization (see Examples 4.0.1 and 4.0.2). The problem with this is that the axiom of choice is independent of $\mathcal{HOL}$ (see [And72b, Mil83, Mil87], so naive Skolemization is only sound in formulations of higher-order logic that do assume the axiom of choice. Mathematicians who usually assume it, normally do so explicitly, and like to be aware of its usage, since it can lead to mathematically controversial results. Naive Skolemization would introduce it through the back-door and would ruin the explicitness that we had hoped to gain by logic-based formalization.

Let us now consider (for the didactic sake) a simple higher-order extension of a tableau calculus using higher-order unification and Andrews' Skolemization, and use it to prove a variant of the axiom of choice by introducing a Skolem function and massaging away its argument to show the problems. A sound version of this calculus will be discussed in detail in chapter 5.

Example 4.0.1 (Proving the Axiom of Choice)
We will prove the version of the axiom of choice given in Definition 1.2.9.

$$\exists C_{(\iota\to o)\to\iota}. \forall P_{\iota\to o}. (\exists X_\iota. PX) \Rightarrow P(CP) \tag{4.1}$$

by negating it, and developing it in a tableau using naive Skolemization, so we try to refute (we have already pushed the negation through the quantifiers)

$$\forall C_{(\iota\to o)\to\iota}. \exists P_{\iota\to o}. (\exists X_\iota. PX) \wedge \neg P(CP) \tag{4.2}$$

The tableau rule for universal quantifiers, just drops the $\forall C_{(\iota\to o)\to\iota}.$

$$\exists P_{\iota\to o}. (\exists X_\iota. PX) \wedge \neg P(CP) \tag{4.3}$$

Now, we Skolemize: we replace P with $f\mathbf{A}$, where f is a Skolem constant of type $((\iota \to o) \to o) \to \iota \to o$ and $\mathbf{A} = \lambda P_{\iota\to o}. (\exists X_\iota. PX) \wedge \neg P(CP)$ to obtain

$$(\exists X_\iota. f\mathbf{A}X) \wedge \neg f\mathbf{A}(C(f\mathbf{A})) \tag{4.4}$$

The rule for conjunctions just puts both conjuncts on the branch,

$$\exists X_\iota. f\mathbf{A}X \tag{4.5}$$

$$\neg f\mathbf{A}(C(f\mathbf{A})) \tag{4.6}$$

Now, we Skolemize again replacing X by $g\mathbf{B}$ in (4.5), where g is a Skolem constant of type $(\iota \to o) \to \iota$ and $\mathbf{B} = \lambda X_\iota. f\mathbf{A}X$ to get

$$f\mathbf{A}(g\mathbf{B}) \tag{4.7}$$

Obviously, we can close the branch, if we can unify

$$\exists C_{(\iota\to o)\to\iota}.f\mathbf{A}(g\mathbf{B})=^? f\mathbf{A}(C(f\mathbf{A})) \tag{4.8}$$

we will do so, using techniques from chapter 3. After decomposition, deleting the trivial equation $\mathbf{A}=^?\mathbf{A}$, and reorienting we obtain

$$\exists C_{(\iota\to o)\to\iota}.C(f\mathbf{A})=^? g\mathbf{B} \tag{4.9}$$

Now, imitate C with $\lambda Z_{\iota\to o}.g(H^1Z)$, simplify (expanding $\mathbf{B}$) to arrive at

$$\exists H^1_{(\iota\to o)\to\iota\to o}.\forall W_\iota.H^1(f\mathbf{A})W=^?\neg f\mathbf{A}W \tag{4.10}$$

Imitating H^1 with $\lambda Z_{\iota\to o}W_\iota.\neg(H^2ZW)$ obtains

$$\exists H^2_{(\iota\to o)\to\iota\to o}.\forall W_\iota.H^2(f\mathbf{A})W=^? f\mathbf{A}W \tag{4.11}$$

Now we 1-project H^2 with $\lambda Z_{\iota\to o}W_\iota.Z(H^3ZW)$ and get (after decomposition)

$$\exists H^3_{(\iota\to o)\to\iota\to o}.\forall W_\iota.H^3(f\mathbf{A})W=^? W \tag{4.12}$$

which can be solved by 2-projecting H^3 with $\lambda Z_{\iota\to o}W_\iota.W$. All in all C is instantiated to $\lambda Z_{\iota\to o}.g(\lambda X_\iota.\neg ZX)$.

In hindsight it is not surprising that can prove the axiom of choice, if we appeal to a choice function (that is what a Skolem function really is) to express variable dependencies. But even if we move to the more conventional form of Skolemization, that replaces $\Pi\mathbf{A}$ with $\mathbf{A}(f\overline{X})$, we can still prove the following choice principle, which (as a schema) is equivalent to the choice principle above (see [ABI+96b, p.333] for a proof of the non-trivial direction of the equality found by the Tps system) and therefore also independent of $\mathcal{HOL}$.

Example 4.0.2 (Proving Relational Choice)
We prove the following choice principle which intuitively says that inside any relation R, whose 2-projection is always non-empty, we can a find a function F.

$$\forall R_{\alpha\to\alpha\to o}.(\forall X_\alpha.\exists Y_\alpha.RXY)\Rightarrow(\exists F_{\alpha\to\alpha}.\forall Z_\alpha.RZ(FZ)) \tag{4.13}$$

by refuting (again, negation has been pushed through)

$$\exists R_{\alpha\to\alpha\to o}.(\forall X_\alpha.\exists Y_\alpha.RXY)\wedge(\forall F_{\alpha\to\alpha}.\exists Z_\alpha.\neg RZ(FZ)) \tag{4.14}$$

Skolemizing R and decomposing the conjunction yields the equations

$$\forall X_\alpha.\exists Y_\alpha.rXY \tag{4.15}$$
$$\forall F_{\alpha\to\alpha}.\exists Z_\alpha.\neg rZ(FZ) \tag{4.16}$$

raising
Benzmüller
Wirth
ranked Skolemization
Skolemization!ranked

we drop the universals and Skolemize Y with gX and Z with hF

$$rX(gX) \tag{4.17}$$

$$\neg r(hF)(F(hF)) \tag{4.18}$$

This tableau branch is closed, if we can unify

$$\exists F_{\alpha\to\alpha} X_\alpha . rX(gX) =^? r(hF)(F(hF)) \tag{4.19}$$

decomposition leads to two equations, the first one $X =^? hF$ is solved, so we directly eliminate X and arrive at

$$\exists F_{\alpha\to\alpha} . g(hF) =^? F(hF) \tag{4.20}$$

Imitation and subsequent 1-projection lead to the solution $[g/F]$.

Dale Miller has investigated the proof theory of this problem in depth [Mil83, Mil87] in the context of expansion trees and higher-order matings [ALCMP84], and further developed this technique in [Mil91, Mil92] for the context of higher-order unification and logic programming. His methods guarantee soundness of Skolemization by explicitly keeping track of the variable dependencies coming from the quantifier prefix and modifying the classical higher-order unification procedure to reject all solutions that do not conform to these restrictions. This bookkeeping can either be implemented using an augmented notion of Skolemization, its dual (called **raising** [Mil92], which we will not cover in this book) or a direct representation in dependency relations.

We will look at the first in the next section, and augment Miller's proof theoretic account (see [Mil83, Mil87, Mil91, Mil92]) by a model-theoretic argument.

For the development of our higher-order refutation calculi, we will choose the third option and present a direct representation of the variable dependencies, that was first used in the author's dissertation [Koh94], and that generalizes to synthetic refutation calculi like resolution. The author is grateful to Christoph Benzmüller and by Claus-Peter Wirth for pointing out a problem in the definition of variable conditions in [Koh94], it has been fixed by an idea from [Wir98].

We will use sections 4.2 to introduce the syntax and semantics of variable conditions and section 4.3 to adapt higher-order unification to the new setting.

4.1 Ranked Skolemization

In this section we will discuss Dale Miller's method of **ranked Skolemization** [Mil83, Mil87] and augment his proof-theoretic account with a model-theoretic view.

To recapitulate: Skolemization is a syntactic trick that allows to employ the occurs-check of unification to reject any instantiation that does not obey the semantic restrictions imposed by $\forall\exists$-quantifications. In first-order logic, this trick is unproblematic: $\forall X.\exists Y.\mathbf{A}$ is satisfiable, iff $\forall X.[fX/Y]\mathbf{A}$ is, where f is a Skolem function. The two formulae are not equivalent, since there may well be several Y for a given X that make $\mathbf{A}$ true, but the function f can only have one value for a particular X. Thus semantically — as we have seen in the discussion above — the Skolem functions are choice functions. In first-order logic, it is unproblematic to add these to the signature, since the Skolem function only occurs in a **Skolem term** $f\mathbf{B}$ that represents an individual — the witness for Y — that we know exists. In higher-order logic, a naive application of the same intuition causes problems, since the notion of arity has been lost (due to Currying).

Skolem term
Curry
rank
f_α^k *Skolem constant*
Σ^{Sk}, Σ^* *Skolem signature*
Skolem constant
constant!Skolem
arity
Skolem
Skolem term
term!Skolem
necessary arguments
arguments!necessary

If we look at the "choice proof" examples above from a technical point of view, we see that in the instantiation $[g/F]$ in Example 4.0.2, the Skolem function g, that was introduced to keep track of the dependence of Y on X (by the witness term $g(X)$) has lost its argument, thereby exposing the bare choice function g necessary for "proving" the choice principle. In Example 4.0.1, the situation is a little more complex, the "proof" uses the fact that in the instantiation $\lambda Z_{\iota\to o}.g(\lambda X_\iota.\neg ZX)$, the argument of the Skolem function g contains the variable Z that is abstracted over outside, giving the whole term functional capabilities.

Dale Miller's solution to this problem was to decorate Skolem constants with arities called **rank** and restrict the set of well-formed formulae to those, where Skolem functions have all their necessary arguments. We will consider this fragment and use it as a basis for the rest of this section.

Definition 4.1.1 (Skolem Constant)
Let $\alpha \in \mathcal{T}$ and $\mathbf{ln}(\alpha) \geq k$, then we assume the existence of a typed collection $\Sigma^{Sk} := \{\Sigma_\alpha^k \mid \alpha \in \mathcal{T}; \mathbf{ln}(\alpha) \geq k\}$, of countably infinite sets of constants that is disjoint from Σ and $\mathcal{V}$. We will call f_α^k a **Skolem constant** of **arity** k and denote $\Sigma \cup \Sigma^{Sk}$ with Σ^*.

Definition 4.1.2 (Herbrand Universe)
Let $\mathit{wff}^*(\Sigma)$ be the smallest fragment of $\mathit{wff}(\Sigma^*)$, such that

1. All variables and constants are in $\mathit{wff}^*(\Sigma)$.

2. If $\alpha = \overline{\beta_k} \to \gamma$ and $f_\alpha^k \in \Sigma_\alpha^k$, then $f_\alpha^k\overline{\mathbf{A}^k} \in \mathit{wff}_\gamma^*(\Sigma)$ provided that the $\mathbf{A}^i$ are in $\mathit{wff}_{\beta_i}^*(\Sigma)$.

   We say that $f_\alpha^k\overline{\mathbf{A}^k} \in \mathit{wff}_\gamma^*(\Sigma)$ is a **Skolem term** and call the $\mathbf{A}^i$ its **necessary arguments**

3. If $\mathbf{A} \in \mathit{wff}_{\alpha\to\beta}^*(\Sigma)$ and $\mathbf{B} \in \mathit{wff}_\alpha^*(\Sigma)$, then $\mathbf{AB} \in \mathit{wff}_\beta^*(\Sigma)$.

ranked
Herbrand
Herbrand Universe
Universe!Herbrand
Herbrand formula
formula!Herbrand
Herbrand substitution
substitution!Herbrand

4. If $\mathbf{A} \in \mathit{wff}^*_\alpha(\Sigma)$ and X is not free in any necessary argument of any Skolem term occurring in $\mathbf{A}$, then $\lambda X_\beta . \mathbf{A} \in \mathit{wff}^*_{\beta\to\alpha}(\Sigma)$.

We call $\mathit{wff}^*(\Sigma)$ the (**ranked**) **Herbrand Universe** for Σ^* and say that $\mathbf{A}$ is a **Herbrand formula**, if $\mathbf{A} \in \mathit{wff}^*(\Sigma)$. Finally, we call a substitution σ a **Herbrand substitution**, if $\sigma(X)$ is a Herbrand formula for all $X \in \mathbf{Dom}(\sigma)$. Note that since we have assumed that $k \leq \mathbf{ln}(\alpha)$ a Skolem constant f^k_α, can occur applied to more than k arguments in a Herbrand formula.

Example 4.1.3 (Herbrand Formulae)
Any formulae in $\mathit{wff}(\Sigma)$ is a Herbrand formula (it does not contain Skolem constants). The formulae $f^1(g^2XY)$, and $\lambda Z . f^1\mathbf{A}Z$ are too, if $Z \notin \mathbf{Free}(\mathbf{A})$, whereas f^1, and $\lambda Z . f^1 Z$, and $\lambda Z . f^1 Z\mathbf{A}$ are not.

Lemma 4.1.4 *$\mathit{wff}^*(\Sigma)$ is closed under Herbrand substitutions, i.e. if $\mathbf{A} \in \mathit{wff}^*_\alpha(\Sigma)$ and σ is a Herbrand substitution, then $\sigma(\mathbf{A}) \in \mathit{wff}^*_\alpha(\Sigma)$.*

Proof: We prove the assertion for $\sigma = [\mathbf{B}/X]$ by an induction over the definition of Herbrand formulae; the general case follows by a straightforward induction over the length of σ.

If $\mathbf{A}$ is a variable or parameter, then $\sigma(\mathbf{A})$ is either $\mathbf{A}$ or $\mathbf{B}$, in both cases, the assertion is trivial. If $\mathbf{A}$ is a Skolem term with head f^k, then $\sigma(\mathbf{A})$ is one for f^k too, furthermore, the inductive hypothesis entails that the necessary arguments of $\sigma(\mathbf{A})$ are Skolem formulae and therefore, we have the assertion. The case where $\mathbf{A}$ is a general application is a simple consequence of the inductive hypothesis. This leaves us with the abstraction case.

So let $\mathbf{A} \equiv \lambda Y_\beta . \mathbf{C}_\gamma$ and $\alpha = \beta \to \gamma$. We can assume that $Y \neq X$ and $Y \notin \mathbf{Free}(\mathbf{B})$ since $\mathit{wff}^*(\Sigma)$ is clearly closed under α-conversion and we have assumed that substitution application renames variables to avoid capture. By inductive hypothesis, we have $[\mathbf{B}/X]\mathbf{C} \in \mathit{wff}^*_\gamma(\Sigma)$, but Y is not free in any necessary argument of $\mathbf{B}$ and Y is not free in $\mathbf{B}$, so Y is not free in any necessary argument in $[\mathbf{B}/X]\mathbf{C}$ and therefore, $\sigma(\mathbf{A}) = \lambda Y . \sigma(\mathbf{C})$ is a Herbrand formula. □

Lemma 4.1.5 *The Herbrand Universe is closed under β-reduction and η-equality.*

Proof: The assertion for β-reduction is a simple consequence of the previous Lemma. For η-equality let $\mathbf{A} = \lambda X . \mathbf{B}X$, where $X \notin \mathbf{Free}(\mathbf{B})$, then X is not free in any necessary argument of $\mathbf{B}$ and for that matter of $\mathbf{B}X$, so $\mathbf{A}$ is a Herbrand formula, if $\mathbf{B}$ is. For the converse direction note that $\mathbf{A}$ can only be a Herbrand formula if $\mathbf{B}X$ is one too, and X is not free in any necessary argument in $\mathbf{B}X$. Therefore, $\mathbf{B}X$ must have been constructed by the application rule, and therefore $\mathbf{B}$ must be a Herbrand formula too. □

Σ^*-*correspondence*
correspondence!Σ^*
evaluation function
algebra!Σ^*
Miller
analytic calculus
calculus!analytical
calculus!generative
generative calculus
variable condition

Now, let us come to the semantics of Herbrand formulae. We cannot just use $\Sigma^* = \Sigma \cup \Sigma^{SK}$-models, since otherwise denotations (via $\mathcal{I}$) of the Skolem functions would end up in the model — making the choice principles valid as we have seen above. Therefore we come up with a specialized interpretation function for Σ^{Sk} constants that does not map into the carrier set.

Definition 4.1.6 (Σ^*-Models)
Let $\mathcal{M} = (\mathcal{D}, @, \mathcal{I})$ be a Σ-algebra, then we call the a function $\mathcal{F}$ on Σ^* with $\mathcal{F}(\Sigma^k_{\overline{\beta_k} \to \gamma}) \subseteq \mathcal{F}(\mathcal{D}_{\beta_1} \times \cdots \times \mathcal{D}_{\beta_k}; \mathcal{D}_\delta)$ a Σ^***-correspondence** for $\mathcal{M}$. We define an **evaluation function** $\mathcal{I}^*_\varphi$ on $\mathit{wff}^*(\Sigma)$ for $\mathcal{M}^*$ by setting $\mathcal{I}^*_\varphi = (\mathcal{I} \cup \mathcal{F})_\varphi$ in definition 1.4.25, and call $\mathcal{A}^* := (\mathcal{D}, @, \mathcal{I}, \mathcal{F})$ a Σ^*-algebra for $\mathcal{A}$.

This notion of evaluation is well-defined on Herbrand formulae since Herbrand terms always have their necessary arguments: By a simple induction over the structure of $\mathbf{A}$, we can see that $\mathcal{I}^*(\mathcal{A}) \in \mathcal{D}_\alpha$ for all $\mathbf{A} \in \mathit{wff}^*_\alpha(\Sigma)$.

In other words, we evaluate Herbrand formulae just as ordinary formulae, only Skolem functions are evaluated using $\mathcal{F}$. As a consequence, the substitution-value theorem is trivial. It is just as simple to establish a substitution-value theorem for Σ^*-algebras

Theorem 4.1.7 (Substitution-Value Theorem for Σ^*-Algebras)
We have $\mathcal{I}^*_\varphi(\lambda X.\mathbf{A})@\mathcal{I}^*_\varphi(\mathbf{B}) \equiv \mathcal{I}^*_{\varphi,[\mathcal{I}^*_\varphi(\mathbf{B})/X]}(\mathbf{A}) \equiv \mathcal{I}^*_\varphi([\mathbf{B}/X]\mathbf{A})$ *if* $\lambda X.\mathbf{A}, \mathbf{B} \in \mathit{wff}^*(\Sigma)$.

Thus we have a extended the semantics developed in Chapter 2 to account for Skolem functions in the Herbrand universe, in a way that preserves all the necessary meta-properties without entailing any choice principles. Furthermore, this semantics coincides with the original one on formulae that do not contain Skolem terms. Therefore this semantics is very well-suited to argue about soundness and completeness of refutation calculi that employ ranked Skolemization.

We will not discuss Dale Miller's other alternative to Skolemization in this book, since it is tied to one specific higher-order refutation approach: expansion trees and higher-order mating search. He uses an explicit relation on nodes in expansion trees that represents the variable dependencies. The mating search calculus is modified to insure the invariant that this relation is acyclic. In the next section we will present a similar method that is not limited to analytical calculi can also be applied to generative calculi like resolution (see section 4.4 for a discussion on the differences of the approaches).

4.2 Variable Conditions

Like in Skolemization, the approach presented in this section assumes special constants as witnesses for existential variables, but chooses to represent the dependencies in a special function, the **variable condition**.

Bibel
Wirth
witness constant
constant!witness
$\mathcal{W}, \mathcal{W}(\mathbf{A})$*set of witness constants*
variable!condition
illegal
legal
substitution
$\mathcal{R}$*-substitution*

In a first-order setting an alternative to Skolemization similar to Miller's dominance relations has independently been considered by Wolfgang Bibel in [Bib82], variable conditions have recently been taken up in a first-order setting by Claus-Peter Wirth in [Wir98, Wir00].

Definition 4.2.1 (Witness Constants)
We assume a typed collection $\mathcal{W} := \{\mathcal{W}_\alpha \mid \alpha \in \mathcal{T}\}$ of **witness constant**s, such that each of the sets $\mathcal{W}_\alpha := \{v_\alpha, w_\alpha, w^1_\alpha, w^2_\alpha, \ldots\}$ is countably infinite. Note that we have already done something similar — if less explicit — by assuming that signatures are always sufficiently pure in Definition 2.2.3.

For a formula $\mathbf{A} \in \mathit{wff}(\Sigma \cup \mathcal{W})$, we will write $\mathcal{W}(\mathbf{A})$ for the set of witness constants in $\mathcal{W}$ occurring in $\mathbf{A}$.

Definition 4.2.2 (Variable Condition, $\mathcal{R}$-Substitution)
We will call a partial function $\mathcal{R}$ that maps witness constants to sets of variables a **variable condition**. A witness constant $w \in \mathcal{W}$ is $\mathcal{R}$**-illegal** for a variable $X \in \mathcal{V}$, iff $X \in \mathcal{R}(w)$ and a formula $\mathbf{A}$ is $\mathcal{R}$**-legal** for X, if $X \notin \mathbf{Free}(\mathbf{A})$ and $\mathbf{A}$ does not contain witness constants that are $\mathcal{R}$-illegal for X.

Finally, a substitution σ is called an $\mathcal{R}$**-substitution**, if for all $X \in \mathbf{Dom}(\sigma)$, the formula $\sigma(X)$ is $\mathcal{R}$-legal for X.

For a given variable condition $\mathcal{R}$ and an $\mathcal{R}$-substitution $[\mathbf{A}/X]$ we will often need the following variable condition

$$\mathcal{R}(\mathbf{A}/X)(w) := \begin{cases} \mathcal{R}(w) & \text{if} \quad X \notin \mathcal{R}(w) \\ (\mathcal{R}(w) \setminus \{X\}) \cup \mathbf{Free}(\mathbf{A}) \cup \mathcal{R}(\mathcal{W}(\mathbf{A})) & \text{if} \quad X \in \mathcal{R}(w) \end{cases}$$

We now proceed to give semantics to formulae with witnesses under variable conditions, which uses the notion of $\mathcal{R}$-correspondences as a semantic counterpart of variable conditions that specify the dependencies of variables. These $\Sigma\mathcal{R}$-algebras use a notion of evaluation where the values of witness constants (which in general depend on the value of the variables in its $\mathcal{R}$-image) are is determined with respect to the given $\mathcal{R}$-correspondence. Note that we only restrict ourselves to $\Sigma\mathcal{R}$-algebras instead of $\Sigma\mathcal{R}$-structures out of notational convenience; We can do the same construction for non-functional Σ-structures by just extending condition 1.4.12.1 in exactly the same way. Theorem 4.2.6 shows that the content of this chapter carries over to the non-functional case.

Definition 4.2.3 ($\mathcal{R}$-Correspondence)
Let $\mathcal{A} = (\mathcal{D}, \mathcal{I})$ be a Σ-algebra and $\mathcal{R}$ a variable condition. If $\mathcal{R}(w_\beta) = \{X^1_{\alpha_1}, \ldots, X^n_{\alpha_n}\}$, then a total function

$$\mathsf{f}_w : \mathcal{D}_{\alpha_1} \times \cdots \times \mathcal{D}_{\alpha_n} \longrightarrow \mathcal{D}_\beta$$

is called an **$\mathcal{R}$-function** for w_β in $\mathcal{A}$. We call a complete set $\mathcal{F} := \{\mathsf{f}_w \mid w_\beta \in \mathbf{Dom}(\mathcal{R})\}$ of $\mathcal{R}$-functions an **$\mathcal{R}$-correspondence** for $\mathcal{A}$. Note that we need a value $\mathsf{f}_w \in \mathcal{D}_\beta$ in $\mathcal{F}$, even if $n = 0$.

$\mathcal{R}$-function
$\mathcal{R}$-correspondence
alternative value function
value function!alternative
induced variable correspondence
variable correspondence!induced

In this definition we need to assume a canonical total ordering on variables to fix argument order for $\mathcal{R}$-functions. The ordering itself is immaterial, we only have to be sure that it is always the same.

Definition 4.2.4 ($\mathcal{F}$-Evaluation)
If $\mathcal{F}$ is an $\mathcal{R}$-correspondence for $\mathcal{M}$ and φ is an assignment into $\mathcal{M}$, then we define the **alternative value function** $\mathcal{I}_\varphi^{\mathcal{F}}$ by the new base case in the definition of homomorphic extension (cf. 1.4.25) for witness constants $w_\beta \in \mathcal{W}$

$$\mathcal{I}_\varphi^{\mathcal{F}}(w_\beta) := \begin{cases} \mathcal{I}(w_\beta), & \text{if} \quad w_\beta \notin \mathbf{Dom}(\mathcal{R}) \\ \mathsf{f}_w(\overline{\varphi(X_n)}), & \text{if} \quad \mathcal{R}(w_\beta) = \{X^1, \ldots, X^n\} \end{cases}$$

Note that $\mathcal{I}_\varphi^{\mathcal{F}}(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{A})$, if $\mathbf{A}$ does not contain witness constants in $\mathbf{Dom}(\mathcal{R})$.

We call a triple $\mathcal{A} = (\mathcal{D}, \mathcal{I}, \mathcal{F})$ a $\Sigma\mathcal{R}$-algebra, iff $\mathcal{F}$ is total. Classes of $\Sigma\mathcal{R}$-models can be defined just like in chapter 2.

For this variant notion of evaluation, we have a substitution-value theorem that is similar to that for the original value function $\mathcal{I}_\varphi$. For this we will need the notion of an induced $\mathcal{R}$-correspondence.

Definition 4.2.5 (Induced $\mathcal{R}$-Correspondence)
Let $\mathcal{R}$ be a variable condition, $\mathcal{F}$ a $\mathcal{R}$-correspondence, and $\mathsf{f}_w \in \mathcal{F}$. Furthermore let $\mathbf{A}$ be a formula with

$$\mathcal{R}(w) = \{\overline{X^n}\} \quad \text{and} \quad \mathbf{Free}(\mathbf{A}) = \{\overline{Y^m}\} \quad \text{and} \quad \bigcup_{w \in \mathcal{W}(\mathbf{A})} \mathcal{R}(w) = \{\overline{Z^l}\}$$

then we define a function $\mathsf{f}_w(\mathbf{A}/X)$ by setting $\mathsf{f}_w(\mathbf{A}/X) := \mathsf{f}_w$ if $X \notin \mathcal{R}(w)$ and $\mathsf{f}_w(\mathbf{A}/X) := \mathsf{g}$, where

$$\begin{array}{rcl} \mathsf{g}(\overline{\mathsf{d}^{n+m+l-1}}) & := & \mathsf{f}_w(\mathsf{a}^1, \ldots, \mathsf{a}^{i-1}, \mathcal{I}_\varphi^{\mathcal{F}}(\mathbf{A}), \mathsf{a}^{i+1}, \ldots, \mathsf{a}^n) \\ \varphi & := & [\mathsf{b}^1/Y^1], \ldots, [\mathsf{b}^m/Y^m], [\mathsf{c}^1/Z^1], \ldots, [\mathsf{c}^1/Z^1] \end{array}$$

and the sequence $\overline{\mathsf{d}^{n+m+l-1}}$ of arguments to g is the set $\{\mathsf{a}^1, \ldots, \mathsf{a}^{i-1}, \mathsf{a}^{i+1}, \ldots, \mathsf{a}^n, \overline{\mathsf{b}^m}, \overline{\mathsf{c}^l}\}$ ordered with respect to the variable ordering mentioned above.

Now we can set $\mathcal{F}(\mathbf{A}/X) := \{\mathsf{f}_w(\mathbf{A}/X) \mid \mathsf{f}_w \in \mathcal{F}\}$. Clearly, $\mathcal{F}(\mathbf{A}/X)$ is an $\mathcal{R}(\mathbf{A}/X)$-correspondence by construction, since $\mathcal{R}(\mathbf{A}/X)(w) = \mathcal{R}(w) \cup \{Y^1, \ldots, Y^m, Z^1, \ldots, Z^l\}$, therefore, we call $\mathcal{F}(\mathbf{A}/X)$ the $\mathcal{R}(\mathbf{A}/X)$-correspondence **induced** by $[\mathbf{A}/X]$.

Theorem 4.2.6 (Substitution-Value Theorem for $\mathcal{I}_\varphi^\mathcal{F}$) *Let $\mathcal{M} = (\mathcal{D}, \mathcal{I})$ be a Σ-algebra, $\mathcal{R}$ a variable condition for $\mathcal{M}$ and $\mathbf{A}$ is $\mathcal{R}$-legal for X, then*

$$\mathcal{I}_\varphi^{\mathcal{F}(\mathbf{A}/X)}([\mathbf{A}/X]\mathbf{B}) = \mathcal{I}_{\varphi,[\mathcal{I}_\varphi^\mathcal{F}(\mathbf{A})/X]}^\mathcal{F}(\mathbf{B})$$

Proof: We prove the assertion by an induction over the structure of $\mathbf{B}$. Since the inductive cases are standard, we will only present the base cases of the induction. If $\mathbf{B}$ is a proper constant c, then

$$\mathcal{I}_\varphi^{\mathcal{F}(\mathbf{A}/X)}([\mathbf{A}/X]\mathbf{B}) = \mathcal{I}_\varphi^{\mathcal{F}(\mathbf{A}/X)}(c) = \mathcal{I}(c) = \mathcal{I}_{\varphi,[\mathcal{I}_\varphi^\mathcal{F}(\mathbf{A})/X]}^\mathcal{F}(c)$$

and analogously if $\mathbf{B} \neq X$ is a variable. If $\mathbf{B} = X$, then

$$\begin{aligned}
\mathcal{I}_\varphi^{\mathcal{F}(\mathbf{A}/X)}([\mathbf{A}/X]\mathbf{B}) &= \mathcal{I}_\varphi^{\mathcal{F}(\mathbf{A}/X)}(\mathbf{A}) \\
&= \varphi, [\mathcal{I}_\varphi^{\mathcal{F}(\mathbf{A}/X)}(\mathbf{A})/X](X) \\
&= \mathcal{I}_{\varphi,[\mathcal{I}_\varphi^{\mathcal{F}(\mathbf{A}/X)}(\mathbf{A})/X]}^\mathcal{F}(X) \\
&= \mathcal{I}_{\varphi,[\mathcal{I}_\varphi^\mathcal{F}(\mathbf{A})/X]}^\mathcal{F}(X)
\end{aligned}$$

The last step is correct, since $[\mathbf{A}/X]$ is a $\mathcal{R}$-substitution: We have $X \notin \mathcal{R}(w)$ for any $w \in \mathcal{W}(\mathbf{A})$, therefore $f_w \in \mathcal{F}$ and $f'_w \in \mathcal{F}(\mathbf{A}/X)$ coincide for $w \in \mathcal{W}(\mathbf{A})$. Thus we have $\mathcal{I}_\varphi^{\mathcal{F}(\mathbf{A}/X)}(\mathbf{A}) = \mathcal{I}_\varphi^\mathcal{F}(\mathbf{A})$, which concludes the proof for the $\mathbf{B} = X$ case.

Now let $\mathbf{B} = w \in \mathcal{W}$ and $\mathcal{R}(w) = \{X^1, \ldots, X^n\}$, then we have

$$\mathcal{R}(\mathbf{A}/X)(w) = \{X^1, \ldots, X^n, Y^1, \ldots, Y^m, Z^1, \ldots, Z^l\}$$

and (by construction of $\mathcal{F}(\mathbf{A}/X)$)

$$\begin{aligned}
&\mathcal{I}_\varphi^{\mathcal{F}(\mathbf{A}/X)}([\mathbf{A}/X]\mathbf{B}) \\
&\quad = \mathcal{I}_\varphi^{\mathcal{F}(\mathbf{A}/X)}(w) \\
&\quad = f'_w(\varphi(X^1), \ldots, \varphi(X^n), \varphi(Y^1), \ldots, \varphi(Y^m), \varphi(Z^1), \ldots, \varphi(Z^l)) \\
&\quad = f_w(\varphi(X^1), \ldots, \varphi(X^n), \mathcal{I}_\varphi^\mathcal{F}(\mathbf{A})) \\
&\quad = \mathcal{I}_{\varphi,[\mathcal{I}_\varphi^\mathcal{F}(\mathbf{A})/X]}^\mathcal{F}(w) \\
&\quad = \mathcal{I}_{\varphi,[\mathcal{I}_\varphi^\mathcal{F}(\mathbf{A})/X]}^\mathcal{F}(\mathbf{B})
\end{aligned}$$

□

However, we do not have the condition 3 for an evaluation function of Σ-structures (cf. 1.4.12), but the following:

Lemma 4.2.7 *Let $\mathcal{R}$ be a variable condition and $\mathcal{F}$ a $\mathcal{R}$-correspondence, then $\mathcal{I}_\varphi^\mathcal{F}(\mathbf{A}) = \mathcal{I}_\psi^\mathcal{F}(\mathbf{A})$ for all assignments φ and ψ that coincide on* $\mathbf{Free}(\mathbf{A}) \cup \mathcal{R}(\mathcal{W}(\mathbf{A}))$.

Proof: The proof is identical to that for $\mathcal{I}_\Phi$ in the proof of Lemma 1.4.26. *unifier!$\mathcal{R}$* We only have to consider the new case of a witness constant $w \in \mathbf{Dom}(\mathcal{R}) \cap \mathcal{W}(\mathbf{A})$ for $\mathcal{I}_\varphi^\mathcal{F}$. There we have

$$\mathcal{I}_\varphi^\mathcal{F}(w) = \mathsf{f}_w(\overline{\varphi(X^n)}) = \mathsf{f}_w(\overline{\psi(X^n)}) = \mathcal{I}_\psi^\mathcal{F}(w)$$

since $\mathcal{R}(w_\beta) = \{X^1, \ldots, X^n\} \subseteq \mathcal{R}(\mathcal{W}(\mathbf{A}))$. $\square$

4.3 HO Unification with Variable Conditions

To use higher-order unification in higher-order refutation calculi like tableaux (cf. section 5) and resolution (cf. section 6), we have to generalize unification problems to take variable conditions into account. These have to be respected in order to obtain a sound calculus. We can think of variable conditions as generalizations of the quantifier prefixes of unification problems in the last chapter. The witness constants take the part of the universally quantified variables: just like these, they may never be instantiated. Furthermore, universally quantified variables were not allowed to appear in solutions, in other words they are illegal for the existential ones. Thus the quantifier prefix $\exists\Gamma.\forall\Upsilon.$ can also be modeled by the variable condition $\mathcal{R} := \{w \mapsto \mathbf{Dom}(\Gamma) \mid w \in \mathbf{Dom}(\Upsilon)\}$.

In this section we will study the higher-order pre-unification for unification problems of the form $\mathcal{E} := \mathcal{R}.\mathcal{F}$, where $\mathcal{F}$ is a conjunction of pairs of formulae in *wff*$(\Sigma \cup \mathcal{W})$. A higher-order $\mathcal{R}$-unifier is a $\mathcal{R}$-substitution that unifies $\mathcal{F}$. Full higher-order unification can be done by similar methods, but we will not pursue it here, since we do not need it in our theorem proving applications.

Dale Miller has studied a generalization of the unification problem with $\exists\forall$-prefix in [Mil92], where he allows arbitrary quantifier alternations, for instance $\mathcal{E} := \forall V_\alpha.\exists X_\beta.\forall W_\gamma.\exists Y_\delta.\mathcal{F}$. With an argumentation similar to that in Remark 3.2.3, we can see that the validity problem for such formulae equivalent to the $\mathcal{R}$-unification problem $\mathcal{E} := \mathcal{R}.\mathcal{F}$, where $\mathcal{R} = \{V \mapsto \{X, Y\}, W \mapsto \{Y\}\}$.

Similar restrictions on the occurrences of constants have been studied for combination of first-order E-unification algorithms (see for instance [BS00]). There, subterms dominated by a function symbol from theory E_1 may contain constants of theory E_2 that need to be eliminated in order for the E_1-unification algorithm to be applicable. The dominance relation leads to dependencies like our variable conditions.

Let us now begin with the technical development of higher-order $\mathcal{R}$-unification. To arrive at a higher-order $\mathcal{R}$-unification algorithm, we have to define solved forms.

solved
solved
$\mathcal{RPU}\Lambda$Rules for $\mathcal{R}$-pre-unification

Definition 4.3.1 ($\mathcal{R}$-solved)
We will call a pair $X \neq^? \mathbf{C}$ **solved** in a unification problem $\mathcal{R}.X \neq^? \mathbf{C} \wedge \mathcal{E}$, iff $\mathbf{C}$ is $\mathcal{R}$-legal for X, and $X \notin \mathbf{Free}(\mathcal{E})$.

The analogue of Lemma 3.2.8, i.e. that $\mathcal{R}$-solved unification problems have most general $\mathcal{R}$-unifiers is immediate. The pre-unification rules have to be slightly changed; the ones that differ from the $\mathcal{P}\Lambda$ rules (cf 3.6.4) are given in Figure 4.1 (we call the changed set of rules $\mathcal{RPU}\Lambda$). The main difference is that we use rule $\mathcal{SIM}(fun)$ instead of $\mathcal{SIM}(\alpha)$ and $\mathcal{SIM}(\eta)$, that $\mathcal{SIM}(elim)$ presupposes $\mathcal{R}$-legality, and finally $\mathcal{U}\Lambda(flex/rig)$ adapts the variable condition for the newly introduced variables.

$$\frac{\mathcal{R}.\mathbf{A}_{\alpha\to\beta} \neq^? \mathbf{B}_{\alpha\to\beta} \wedge \mathcal{E} \quad w \text{ new}}{\mathcal{R} * \{w \mapsto \mathbf{Free}(\mathbf{A}) \cup \mathbf{Free}(\mathbf{B})\}.\mathbf{A}w \neq^? \mathbf{B}w}\ \mathcal{SIM}(fun)$$

$$\frac{\mathcal{R}.F\overline{w^n} =^? \mathbf{A} \wedge \mathcal{E} \qquad \begin{array}{c} F \in \mathbf{Free}(\mathcal{E}) \quad w^i \in \mathcal{W} \\ (\lambda\overline{w^n}.\mathbf{A})\ \mathcal{R}\text{-legal for } F \end{array}}{\mathcal{R}.F =^? (\lambda\overline{w^n}.\mathbf{A}) \wedge [(\lambda\overline{w^n}.\mathbf{A})/F]\mathcal{E}}\ \mathcal{SIM}(elim)$$

$$\frac{\mathcal{R}.F_\alpha\overline{\mathbf{U}} =^? h\overline{\mathbf{V}} \wedge \mathcal{E} \quad \mathbf{G} \in \mathcal{A}^h(\Sigma \cup \mathcal{W}) \quad F \notin \mathcal{R}(h)}{\mathcal{R}(\mathbf{G}/F).F =^? \mathbf{G} \wedge [\mathbf{G}/F](F\overline{\mathbf{U}} =^? h\overline{\mathbf{V}} \wedge \mathcal{E})}\ \mathcal{U}\Lambda(flex/rig)$$

Figure 4.1: $\mathcal{RPU}\Lambda$-Rules for $\mathcal{R}$-Pre-Unification

Lemma 4.3.2 (Simplification Preserves $\mathcal{R}$-Unifiers)
If $\mathcal{D}: \mathcal{E} \vdash_{\mathcal{SIM}} \mathcal{E}'$, then $\mathcal{R}\mathbf{U}(\mathcal{E}) = \mathcal{R}\mathbf{U}(\mathcal{E}')[\mathcal{E}]$ and $\mathcal{R}\mathbf{PU}(\mathcal{E}) = \mathcal{R}\mathbf{PU}(\mathcal{E}')[\mathcal{E}]$.

Proof: For $\mathcal{SIM}(fun)$ let σ be a $\mathcal{R}$-unifier of a pair $\mathbf{A} =^? \mathbf{B}$, then $\sigma(\mathbf{A}w) = \sigma(\mathbf{A})w =_{\beta\eta} \sigma(\mathbf{B})w = \sigma(\mathbf{B}w)$. For the other direction let σ be a $\mathcal{R} * \{w \mapsto \mathbf{Free}(\mathbf{A}) \cup \mathbf{Free}(\mathbf{B})\}$-unifier of $\mathbf{A}w \neq^? \mathbf{B}w$. Then we have $\sigma(\mathbf{A})w = \sigma(\mathbf{A}w) =_{\beta\eta} \sigma(\mathbf{B}w) = \sigma(\mathbf{B})w$, and as a consequence $\lambda w.\sigma(\mathbf{A})w =_{\beta\eta} \lambda w.\sigma(\mathbf{B})w$. Since σ is a $\mathcal{R} * \{w \mapsto \mathbf{Free}(\mathbf{A}) \cup \mathbf{Free}(\mathbf{B})\}$-substitution, we have $w \notin \mathcal{W}(\sigma(X))$ for any $X \in \mathbf{Free}(\mathbf{A}) \cup \mathbf{Free}(\mathbf{B})$. Together with the fact that w was new, and thus cannot occur in $\mathbf{A}$ and $\mathbf{B}$, we have $w \notin \mathcal{W}(\sigma(\mathbf{A})) \cup \mathcal{W}(\sigma(\mathbf{B}))$. Thus we have $\sigma(\mathbf{A}) =_{\beta\eta} \sigma(\mathbf{B})$ by η-equality.

For $\mathcal{SIM}(elim)$ the assertion is trivial, since the variable condition is left untouched. □

This immediately gives soundness. The completeness result can be obtained with the methods for higher-order pre-unification using a variant of Lemma 3.6.5:

Lemma 4.3.3 (Semi-Termination for $\mathcal{RPU}\Lambda$)
Let $\mathcal{E} = \mathcal{R}.\mathcal{F}$ be an unsolved unification problem in $\mathcal{SIM}$-normal form and

$\theta \in \mathcal{R}\mathbf{PU}(\mathcal{E})$, then there exists a unification problem $\mathcal{E}' = \mathcal{R}'.\mathcal{F}'$ and a $\mathcal{R}'$-substitution θ', such that

$$\mathcal{E} \vdash_{\mathcal{RPU}\Lambda} \mathcal{E}' \qquad \theta \equiv \theta'[\mathcal{E}] \qquad \theta' \in \mathcal{R}\mathbf{PU}(\mathcal{E}') \qquad \mu(\mathcal{E}',\theta') \prec \mu(\mathcal{E},\theta)$$

Proof: We argue just as in Lemma 3.6.5, and construct θ' as we have done there. The only interesting case for variable conditions is the flex/rigid case with an unsolved pair $\mathbf{A} =^? \mathbf{B}$ where $\mathbf{head}(\mathbf{A}) = F$ and $\theta(F) = \lambda Z.h\overline{\mathbf{U}}$ (in all other cases $\mathcal{R}' = \mathcal{R}$). So we have $\mathbf{G} = \lambda\overline{Z}.h\overline{\mathbf{V}}$, and $\rho = \overline{[\lambda Z.\mathbf{U}^i/H^i]}$ where $\mathbf{V}^i = H^i\overline{Z}$ and choose $\mathcal{R}' := \mathcal{R}(\mathbf{G}/F)$. Now we have to show that $\theta' = \theta \cup \rho$ is a $\mathcal{R}'$-substitution.

We first convince ourselves that θ is. So let $X \in \mathbf{Dom}(\theta)$ and $v \in \mathcal{W}(\theta(X))$; we show that v is $\mathcal{R}'$-legal for X, i.e. that $X \notin \mathcal{R}'(v)$ by examining the cases in the construction of $\mathcal{R}' = \mathcal{R}(\mathbf{G}/F)$. If $F \notin \mathcal{R}(v)$, then $\mathcal{R}'(v) = \mathcal{R}(v)$ and $X \notin \mathcal{R}(v)$, since $\theta(X)$ is $\mathcal{R}$-legal for X. If $F \in \mathcal{R}(v)$, then $\mathcal{R}'(v) = \mathcal{R}(v) \cup \{H^1,\ldots,H^n\} \cup \mathcal{R}(\mathcal{W}(\mathbf{G}))$. X cannot be in the first two components, since $\theta(X)$ is $\mathcal{R}$-legal for X and $H^i \notin \mathbf{Dom}(\theta)$ since they were new. Now, we have to consider two cases: if $h \notin \mathcal{W}$, then $\mathcal{W}(\mathbf{G}) = \emptyset$ (all other symbols are variables) and we are done. If $h \in \mathcal{W}$, then $h \in \mathcal{W}(\theta(F))$, since $h = \mathbf{head}(\theta(F))$.[24] EdNote(24)

Now, it only remains to show that ρ is a $\mathcal{R}'$-substitution. So let $v \in \mathcal{W}(\rho(H^i)) = \mathcal{W}(\lambda Z.\mathbf{U}^i)$, then $v \in \mathcal{W}(\theta(F))$ since $\theta(F)$ is $\mathcal{R}$-legal for F, thus $\mathcal{R}'(v) = \mathcal{R}(v)$ and $H^i \notin \mathcal{R}(v)$, since it is new. □

Corollary 4.3.4 (Soundness and Completeness for $\mathcal{RPU}\Lambda$)
For any unification problem $\mathcal{E} = \mathcal{R}.\mathcal{F}$ and any $\theta \in \mathbf{PU}(\mathcal{E})$, there is a $\mathcal{RPU}\Lambda$-derivation $\mathcal{E} \vdash_{\mathcal{RPU}\Lambda} \mathcal{E}' = \mathcal{R}'.\mathcal{F}'$ such that $\mathcal{E}'$ is in $\mathcal{R}'$-pre-solved form, $\sigma_{\mathcal{E}'} \in \mathbf{PU}(\mathcal{E})$ and $\sigma_{\mathcal{E}'} \preceq^p_{\beta\eta} \theta[\mathcal{E}]$.

4.4 Relating the Approaches

Note that Miller's mixed-prefix fragment is less expressive than our's, since due to the linear ordering of quantifier prefix, he cannot express the variable condition $\{v \mapsto \{X\}, w \mapsto \{Y\}\}$:

prefix	forbids	allows
$\forall V.\forall W.\exists X.\exists Y.$		$[W/X],[V/Y]$
$\exists X.\forall V.\exists Y.\forall W.$	$[W/X]$	
$\exists X.\forall W.\exists Y.\forall V.$	$[V/Y]$	
$\exists X.\exists Y.\forall V.\forall W.$	$[W/X],[V/Y]$	

[24] EdNote: To show that $X \notin \mathcal{R}(h)$, how do we do this? This is false: take $\theta = [v/X],[\lambda Z.h]$, where $h, v \in \mathcal{W}$. Furthermore take $\mathcal{R}(h) = \{X\}$ and $\mathcal{R}(v) = \emptyset$. Then $\mathbf{G} = \lambda Z.h$ and $\mathcal{R}(\mathbf{G}/F)(v) = \mathcal{R}(h) = \{X\}$. We have to restrict the condition in the unification rule. Think of the Skolemized version first.

The notion of a Σ^*-correspondence and a $\mathcal{R}$-correspondence are very similar. We can in fact give a model-preserving transformation between the approaches:

Let $\mathbf{A} \in \mathit{wff}^*(\Sigma)$ and $\mathcal{T}$ the set of Skolem subterms of $\mathbf{A}$. Now, let $\mathcal{W}$ be a set of witnesses and $\omega{:}\,\mathcal{T} \longrightarrow \mathcal{W}$ a type-preserving bijection. We can obtain a formula $\Omega\textsc{mega}(\mathbf{A}) \in \mathit{wff}(\Sigma \cup \mathcal{W})$ by replacing every occurrence of $\mathbf{B} \in \mathcal{T}$ in $\mathbf{A}$ by $\omega(\mathbf{B})$. The corresponding variable condition is given by $\mathcal{R} = \mathbf{Free} \circ \omega^{-1}$, i.e. $\mathcal{R}(w)$ is the set of free variables of the Skolem term corresponding to w under ω. To see that this mapping respects models, let $\mathcal{M}^*(\mathcal{D}, @, \mathcal{I}, \mathcal{F})$ be a Σ^*-algebra with $\mathcal{I}^*_\varphi(\mathbf{A}) = \mathsf{a}$. Now let $\Omega(\mathcal{F})$ be the $\mathcal{R}$-correspondence defined by $\mathsf{f}_w(\mathsf{a}_1, \ldots, \mathsf{a}_k) := \mathcal{I}^*_\psi(\omega^{-1}(w))$, where $\mathcal{R}(w) = \{X^1, \ldots, X^n\}$ and $\psi = [\mathsf{a}_1/X^1] \ldots [\mathsf{a}_n/X^n]$. This makes $\Omega(\mathcal{M}^*) = (\mathcal{D}, @, \mathcal{I}, \Omega(\mathcal{F}))$ into a $\Sigma\mathcal{R}$-algebra. Finally, we convince ourselves that $\mathcal{I}^{\Omega(\mathcal{F})}_\varphi(\Omega(\mathbf{A})) = \mathcal{I}^*_\varphi(\mathbf{A})$, in other words, our mapping Ω is model-preserving. We prove the claim by induction on the structure of $\Omega(\mathcal{A})$; the only interesting case is where $\Omega(\mathbf{A})$ is a witness, say w with $\mathcal{R}(w) = \{X^1, \ldots, X^n\}$. Thus we have

$$\mathcal{I}^{\Omega(\mathcal{F})}_\varphi(\Omega(\mathbf{A})) = \mathsf{f}_w(\overline{\varphi(X^n)}) = \mathcal{I}^*_\varphi(\omega^{-1}(w)) = \mathcal{I}^*_\varphi(\mathbf{A})$$

The transformation in the other direction is even simpler: Let $\mathcal{W}$ be a set of witness constants and $\mathcal{R}$ a variable condition over $\mathcal{W}$, then we choose $\Sigma^k_{\overline{\beta_k} \to \gamma} := \{f^k_w \mid \mathcal{R}(w_\gamma) = \{X^1_{\beta_1}, \ldots, X^n_{\beta_k}\}\}$. With this construction, any $\mathcal{R}$-correspondence $\mathcal{F}$ is also a Σ^*-correspondence. Obviously, replacing all witness constants w occurring in a formula $\mathbf{A} \in \mathit{wff}(\Sigma \cup \mathcal{W})$ by Skolem formulae $(f^k_w \overline{X^k})$ will yield a formula $\mathbf{A}^*$ in $\mathit{wff}^*(\Sigma)$. To see that $\mathcal{I}^{\mathcal{F}}_\varphi(\mathbf{A}) = \mathcal{I}^*_\varphi(\mathbf{A}^*)$, we again only have to look at the case, where $\mathbf{A}$ is a witness w with $\mathcal{R}(w) = \{X^1, \ldots, X^n\}$.

$$\mathcal{I}^{\mathcal{F}}_\varphi(\mathbf{A}) = \mathsf{f}_w(\overline{\varphi(X^n)}) = \mathcal{I}^*(f^k_w)@(\overline{\varphi(X^n)}) = \mathcal{I}^*_\varphi(f^k_w \overline{X^k}) = \mathcal{I}^*_\varphi(\mathbf{A}')$$

which concludes the argument.

Chapter 5

Higher-Order Tableaux

In this chapter we present a higher-order tableau calculus. First-order tableaux were introduced by Beth [Bet69] and Hintikka [Hin55] and later unified by Smullyan [Smu68]. The free variable tableaux method has its origin in the work of Prawitz [Pra60] and has further been elaborated by Reeves [Ree87] and Fitting [Fit90].

Generalizing these methods, we develop a tableau calculus $\mathcal{HT}$, which is strongly related to the higher-order matings method [And89] of Andrews and Miller's expansion trees [Mil83, Pfe87]. Just like tableau calculi these methods build trees annotated with formulae by decomposing the propositional structure of the input formulae. The most significant difference is that the test of inconsistency is done by finding a "spanning mating" instead of closing all branches. In this perspective expansion trees correspond to standard tableaux whereas the higher-order mating method corresponds to free variable tableaux, and we have adapted ideas from both for the development of $\mathcal{HT}$ and $\mathcal{HTE}$. Therefore, the methods presented in this chapter can be directly carried over to a completeness proof for the higher-order matings method that is still missing in the literature.

Weaker variants of the calculi $\mathcal{HT}$ and $\mathcal{HTE}$ have been first considered in [Koh95], it differs from the present version in the treatment of functional extensionality. In this respect, the calculi from [Koh95] were not complete with respect to Σ-structures. Christoph Benzmüller has noted this shortcoming and proposed the rule $\mathcal{HT}(leib)$ in [Ben97].

The case of standard tableaux for higher-order logic is a simple extension of first-order tableau methods to the higher-order language. The only significant difference is that $\alpha\beta\eta$-equality has to be built in by keeping formulae in normal form. Therefore we will only concern ourselves with free variable tableaux.

labeled proposition
proposition!labeled
literal
atomic
rigid
literal!rigid
flexible
literal!flexible
unification pair
$\mathbf{A}_\alpha \neq^? \mathbf{B}_\alpha$ *unification pair*
$\cdot\mathcal{E}_\mathcal{T}$ *unfication problem of* $\mathcal{R}.\mathcal{T}$
higher-order tableau
tableau!higher-order
tableau!construction rules
rules!tableau construction
unification problem of
has substitution
substitution!of tableau

5.1 The Calculus $\mathcal{HT}$

Now, we can come to the definition of the higher-order tableau calculi. We will start by defining the necessary data structures; see 5.1.5 for an example.

Definition 5.1.1 (Literal)
We will call a proposition $\mathbf{A}^v$, where $v \in \{\mathtt{T}, \mathtt{F}\}$ a **labeled proposition** and a **literal** if $\mathbf{A}$ is **atomic**, i.e. if the head of $\mathbf{A}$ is a parameter or a variable. In accordance with Definition 1.3.2, we call a literal **rigid** in the first case and **flexible** otherwise.

We will call a literal of the form $(\mathbf{A}_\alpha = \mathbf{B}_\alpha)^\mathtt{F}$ a **unification pair**, and write it as $\mathbf{A}_\alpha \neq^? \mathbf{B}_\alpha$ to emphasize this.

Definition 5.1.2 (Higher-Order Tableau)
Let $\mathcal{R}$ be a variable condition. We will call a pair $\mathcal{R}.\mathcal{T}$, where $\mathcal{T}$ is a tree whose nodes are labeled with labeled propositions or unification constraints a **higher-order tableau**, iff it can be constructed by the tableau **construction rules** in Figures 5.1 to 5.3. We will call the unification problem $\mathcal{R}.\mathcal{P}_1 \wedge \ldots \wedge \mathcal{P}_n$ the **unification problem of** $\mathcal{R}.\mathcal{T}$ (and write it as $\mathcal{R}.\mathcal{E}_\mathcal{T}$), iff the set of unification pairs at the leaves of $\mathcal{T}$ is $\{\mathcal{P}_1^\mathtt{F}, \ldots, \mathcal{P}_n^\mathtt{F}\}$. We will often use $\mathbf{PU}(\mathcal{T})$ for $\mathbf{PU}(\mathcal{R}.\mathcal{E}_\mathcal{T})$.

Finally, if $\mathcal{R}.\mathcal{E}$ is the unification problem of $\mathcal{R}.\mathcal{T}$, and $\mathcal{E}_\theta$ is the solved part of $\mathcal{E}$, then we say that $\mathcal{R}.\mathcal{T}$ **has substitution** θ.

Remark 5.1.3 (Unification Constraint)
The unification constraint of a tableau represents the unification problem that has to be solved in order to close the branch. Since we are in a refutation theorem proving setting, it is expressed by negative literals. We will see that the $\mathcal{HT}(cut)$ rule residuates these unification pairs as necessary conditions to close a branch.

The structural rules in Figure 5.1 recursively build up the tableau tree by decomposing the logical structure of proper labeled formulae and adding new nodes and branches, managing the variable condition along the way. Note that the quantifier rules transform an formula $(\mathcal{Q}.\mathcal{E})^\mathtt{F}$, where $\mathcal{Q}$ is an arbitrary quantifier prefix and $\mathcal{E}$ a set of pairs into a tableau with unfication problem $\mathcal{R}.\mathcal{E}$ (see the discussion on page cxlix).

Since higher-order unification is undecidable, we need an explicit rule $\mathcal{HT}(cut)$ (see Figure 5.2) for cutting complementary formulae: If $\mathbf{A}^v$ and $\mathbf{B}^w$ where $v \neq w$ occur in a branch $\mathcal{B}$ of $\mathcal{R}.\mathcal{T}$, then the rule $\mathcal{HT}(cut)$ introduces the unification pair $\mathbf{A} \neq^? \mathbf{B}$, which has to be processed before $\mathcal{B}$ can be closed in order to conserve soundness. Tableau instantiation is only permitted for $\mathcal{R}$-solved pairs, where the most general unifier is obvious ($\mathcal{HT}(subst)$ in Figure 5.2).

induced substitution
substitution!induced
induced
primitive substitution
substitution!primitive
flexible literal
literal!flexible

$$\frac{(\mathbf{A}\vee\mathbf{B})^{\mathtt{T}}}{\mathbf{A}^{\mathtt{T}} \mid \mathbf{B}^{\mathtt{T}}}\,\mathcal{HT}(\wedge) \qquad \frac{(\neg\mathbf{A})^{\mathtt{T}}}{\mathbf{A}^{\mathtt{F}}}\,\mathcal{HT}(\neg^{\mathtt{F}}) \qquad \frac{(\Pi^{\alpha}\mathbf{A})^{\mathtt{T}}}{(\mathbf{A}X_{\alpha})^{\mathtt{T}}}\,\mathcal{HT}(all)$$

$$\frac{(\mathbf{A}\vee\mathbf{B})^{\mathtt{F}}}{\begin{array}{c}\mathbf{A}^{\mathtt{F}}\\ \mathbf{B}^{\mathtt{F}}\end{array}}\,\mathcal{HT}(\vee) \qquad \frac{(\neg\mathbf{A})^{\mathtt{F}}}{\mathbf{A}^{\mathtt{T}}}\,\mathcal{HT}(\neg^{\mathtt{T}}) \qquad \frac{(\Pi^{\alpha}\mathbf{A})^{\mathtt{F}}}{(\mathbf{A}w_{\alpha})^{\mathtt{F}}}\,\mathcal{HT}(ex)^{1}$$

[1] w_α new and $\mathcal{R}^1 = \mathcal{R} * \{w_\alpha \mapsto \mathbf{Free}(\mathbf{A}) \cup \mathcal{R}(\mathcal{W}(\mathbf{A}))\}$

Figure 5.1: Structural Rules of $\mathcal{HT}$

For a given tableau $\mathcal{R}.\mathcal{T}$, we will call the combined substitution $\theta_{\mathcal{T}}$ generated by all applications of $\mathcal{HT}(subst)$ in $\mathcal{T}$ the substitution **induced** by $\mathcal{R}.\mathcal{T}$. Note that σ is a $\mathcal{R}$-substitution. The $\mathcal{HT}(leib)$ rule allows to interpret unification pairs as Leibniz equalities. This rule has been introduced by Christoph Benzmüller in [Ben97] and can be seen as a restricted form of primitive substitution introducing the necessary structure for Leibniz equality. Note that the inverse of this rule is admissible in $\mathcal{HT}$, as we can see in Figure 5.7. Since it is not clear whether all instantiations that are necessary for a proof can be found by the unification rules in Figure 5.3, $\mathcal{HT}$ utilizes a **primitive substitution** rule that allows to instantiate flexible literals with general bindings for arbitrary connectives and quantifiers.

Unification constraints can be structurally simplified by the dual variants of the modified unification rules from section 4.3, they are given in Figure 5.2. All other substitution pairs have to be treated by the unification

$$\frac{\begin{array}{c}\mathbf{A}^{v}\\ \mathbf{B}^{w}\end{array}}{\mathbf{A}\neq^{?}\mathbf{B}}\,\mathcal{HT}(cut) \qquad \frac{X\neq^{?}\mathbf{C} \quad \mathbf{C}\ \mathcal{R}\text{-legal for } X}{\mathbf{C}\neq^{?}\mathbf{C}}\,\mathcal{HT}(subst)^{1}$$

$$\frac{\mathbf{A}\neq^{?}\mathbf{B}}{\mathbf{A}=\mathbf{B}^{\mathtt{F}}}\,\mathcal{HT}(leib) \qquad \frac{X_{\alpha}\overline{\mathbf{U}^{n}}^{v} \quad \mathbf{K}\in\mathcal{A}^{k}(\Sigma) \quad k\in\Sigma^{\mathcal{HOL}}}{X\overline{\mathbf{U}^{n}}^{v} \mid X\neq^{?}\mathbf{K}}\,\mathcal{HT}(prim)^{2}$$

[1] the resulting tableau is of the form $\mathcal{R}(\mathbf{C}/X).[\mathbf{C}/X]\mathcal{T}$
[2] the new pair $X\neq^{?}\mathbf{K}$ is eagerly eliminated by $\mathcal{HT}(subst)$

Figure 5.2: Constraint and instantiation rules in $\mathcal{HT}$

rule from. Note that the rules from Figure 5.3 together with $\mathcal{HT}(subst)$ directly correspond to the rules of higher-order pre-unification [Hue75]. With these rules we use the tableau mechanism to construct unification trees (see Theorem 3.6.6). All of these rules are used with the understanding that all

closed!branch
flexible
rigid
closed!tableau
tableau!refutation
refutation!tableau
proof!tableau

formulae are reduced $\beta\eta$-normal form after each rule application.

$$\frac{F_\alpha\overline{\mathbf{U}}\neq^? h\overline{\mathbf{V}} \quad \mathbf{G}\in\mathcal{A}^h_\alpha(\Sigma)}{F\neq^?\mathbf{G}_\alpha \;\Big|\; F\overline{\mathbf{U}}\neq^? h\overline{\mathbf{V}}}\,\mathcal{HT}(\mathit{flex/rig})$$

$$\frac{h\overline{\mathbf{U}^n}\neq^? h\overline{\mathbf{V}^n} \quad h\in\Sigma\cup\mathcal{W}}{\mathbf{U}^1\neq^?\mathbf{V}^1 \;\Big|\; \ldots \;\Big|\; \mathbf{U}^n\neq^?\mathbf{V}^n}\,\mathcal{SIM}(dec)$$

$$\frac{\mathbf{A}_{\alpha\to\beta}\neq^?\mathbf{B}_{\alpha\to\beta} \quad w_\alpha \text{ new}}{\mathbf{A}w_\alpha\neq^?\mathbf{B}w_\alpha}\,\mathcal{SIM}(\mathit{fun})^1$$

[1] $w_\alpha\in\mathcal{W}$ new and
$\mathcal{R}^1=\mathcal{R}*\{w\mapsto\mathbf{Free}(\mathbf{A})\cup\mathbf{Free}(\mathbf{B})\}$

Figure 5.3: Unification rules of $\mathcal{HT}$

Definition 5.1.4 (Closed Tableau)
We call a branch $\mathcal{B}$ in a higher-order tableau $\mathcal{T}$ **closed**, iff $\mathcal{B}$ ends in a flex-flex pair or a trivial pair $\mathbf{A}\neq^?\mathbf{A}$. Note that closed branches need not stay closed, since the heads of flex-flex pairs can be instantiated making them rigid. Recall that a formula is called **flexible**, if the head is a free (positive) variable and **rigid** otherwise. In this case unification has to be resumed on the particular branch. The $\mathcal{HT}(\mathit{subst})$ rule immediately closes the branch $\mathcal{B}$ that ends in a $\mathcal{R}$-solved pair.

A tableau $\mathcal{R}.\mathcal{T}$ is called **closed**, iff each branch of $\mathcal{R}.\mathcal{T}$ is closed. For a proposition $\mathbf{A}\in \mathit{wff}_o(\Sigma)$ we write $\vdash_{\mathcal{HT}}\mathbf{A}$, if there is a closed higher-order tableau $\mathcal{R}.\mathcal{T}$ with $\mathbf{A}^{\mathsf{F}}$ at the root. In this case we call $\mathcal{R}.\mathcal{T}$ a $\mathcal{HT}$ **refutation** for $\mathbf{A}^{\mathsf{F}}$ and a $\mathcal{HT}$**-proof** for $\mathbf{A}$.

Example 5.1.5 (A $\mathcal{HT}$-Proof of Cantor's Theorem)
We show that there cannot be a surjective mapping from the natural numbers to infinite sequences of natural numbers (see example 1.2.18). In order to simplify the problem we take the type ι to be the set of natural numbers ($\mathbb{N}=\lambda X_\alpha.\mathbf{T}_o$), thus Cantor's theorem has the form

$$\neg\exists F_{\iota\to\iota\to\iota}.\forall G_{\iota\to\iota}.\exists J_\iota.FJ=G$$

To be able to prove the theorem we need a property of the natural numbers (instead of the Peano Axioms we add the fact that $\forall X_\iota.\neg X=sX$ that the successor function has no fixed point). To keep the presentation short, we will add an extensionality axiom, which is not strictly needed. Figure 5.4 shows the tableau proof of cantor's theorem. Note that if we collect the bindings for G, then we obtain the instance $\lambda Z_\iota . s(fZZ)$, which corresponds to the incremented diagonal sequence that serves as a counterexample to countability in Cantor's original proof.

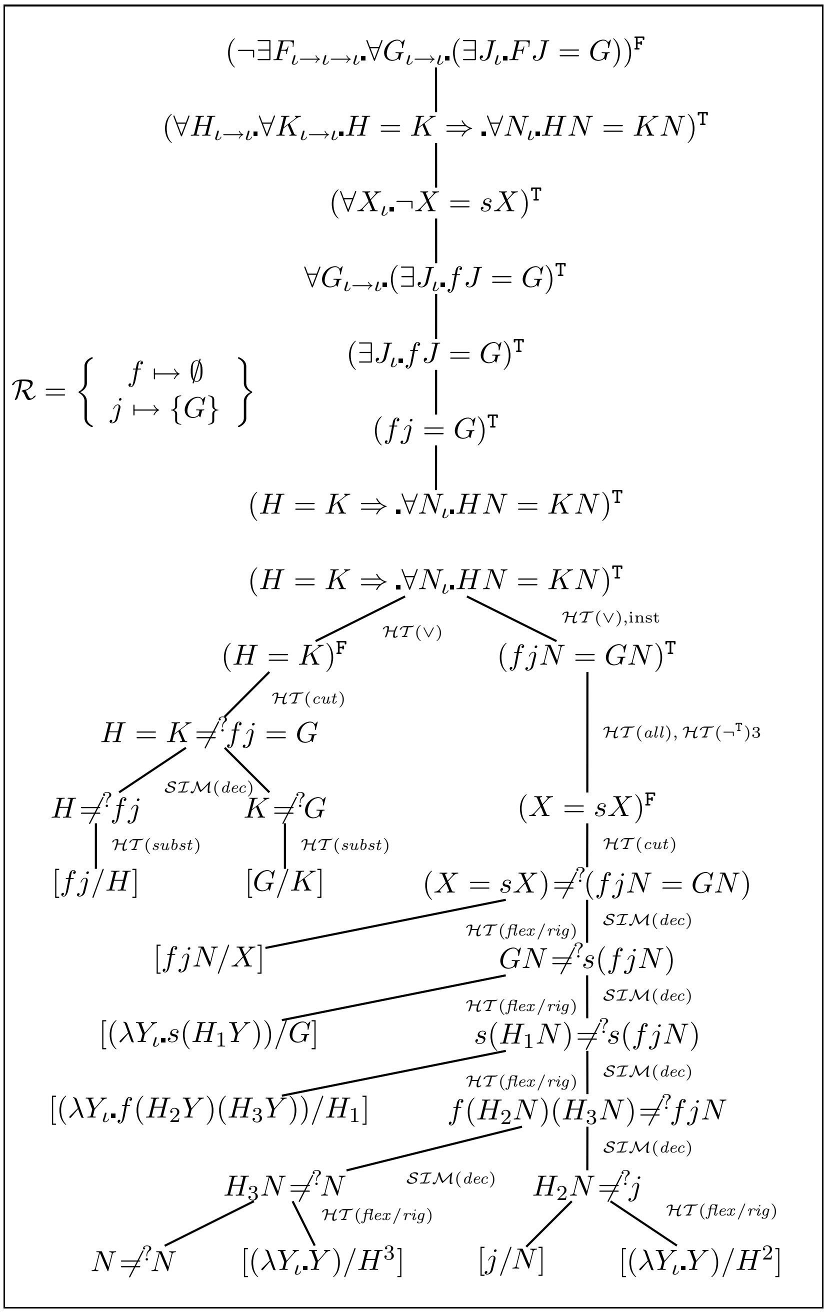


Figure 5.4: Tableau for Cantor's Theorem

$\mathcal{R}$-satisfiable

5.2 Soundness

We now proceed to give a definition of tableau satisfiability based on the notion of evaluation developed in section 4.2.

Definition 5.2.1 (Tableau Satisfiability)
We say that a labeled formula $\mathbf{A}^{v}$ is **$\mathcal{R}$-satisfiable** in a Σ-model $\mathcal{M} = (\mathcal{D}, \mathcal{I}, \upsilon)$, iff there is an $\mathcal{R}$-correspondence $\mathcal{F}$ for $\mathcal{M}$, such that $\upsilon(\mathcal{I}_{\varphi}^{\mathcal{F}}(\mathbf{A})) = v$ for all assignments φ; analogously for a pair $\mathbf{A} \neq^{?} \mathbf{B}$, iff $\mathcal{I}_{\varphi}^{\mathcal{F}}(\mathbf{A}) \neq \mathcal{I}_{\varphi}^{\mathcal{F}}(\mathbf{B})$. We call a (branch in a) tableau $\mathcal{R}.\mathcal{T}$ satisfiable, iff there is a single assignment φ, that satisfies all labeled formulae (on that branch) of $\mathcal{R}.\mathcal{T}$.

Note that closed branches cannot be satisfiable (since they contain complementary labeled formulae) and closed tableaux must be unsatisfiable, since all branches are closed. As a consequence, closed tableaux are unsatisfiable.[25]

EdNote(25)

Theorem 5.2.2 (Model-Preservation for $\mathcal{HT}$)
$\mathcal{HT}$ preserves Σ-models.

Proof: Let $\mathcal{R}'.\mathcal{T}'$ be a tableau obtained from $\mathcal{R}.\mathcal{T}$ by a tableau construction rule and $\mathcal{M}$ be a Σ-model, then we show that $\mathcal{M} \models \mathcal{R}.\mathcal{T}$, iff $\mathcal{M} \models \mathcal{R}'.\mathcal{T}'$. We only present the proof for the rules $\mathcal{HT}(ex)$ and $\mathcal{HT}(subst)$, since $\mathcal{SIM}(fun)$ is analogous to $\mathcal{HT}(ex)$ and all others are unproblematic, because the variable condition is not altered by the transformation.

In the case of $\mathcal{HT}(ex)$ $\mathcal{T}'$ is obtained from $\mathcal{T}$ by adding a formula $\mathbf{A}w_{\alpha}^{\mathtt{F}}$ ($\mathcal{T}$ contains $\Pi^{\alpha}\mathbf{A}^{\mathtt{F}}$), and we have $\mathcal{R}' = \mathcal{R} * \{w \mapsto W\}$, where $W := \mathbf{Free}(\mathbf{A}) \cup \mathcal{R}(\mathcal{W}(\mathbf{A})) = \{X^1, \ldots, X^n\}$. If $\mathcal{M} = (\mathcal{D}, \mathcal{I}, \upsilon) \models \mathcal{R}.\mathcal{T}$, then there is an $\mathcal{R}$-correspondence $\mathcal{F}$ for $\mathcal{M}$ such that for all assignments φ there is a labeled proposition or unification pair in $\mathcal{R}.\mathcal{T}$ that is satisfied by φ in $\mathcal{M}$.

Since $\mathcal{M} \models \Pi^{\alpha}\mathbf{A}^{\mathtt{F}}$ we have $\upsilon(\mathcal{I}_{\varphi}^{\mathcal{F}}(\Pi^{\alpha}\mathbf{A})) = \mathtt{F}$, thus there is an $\mathsf{a} \in \mathcal{D}_{\alpha}$ such that $\mathcal{I}_{\varphi}^{\mathcal{F}}(\mathbf{A})(\mathsf{a}) = \mathcal{I}_{\psi}^{\mathcal{F}}(\mathbf{A}Y) = \mathtt{F}$, where Y is a new variable and $\psi := \varphi, [\mathsf{a}/Y]$. This a only depends on $\psi|_{W}$, and as a consequence on the values of φ on W, since $Y \notin W$ by Lemma 4.2.7. Since we have made no assumptions on φ, we can construct an n-ary function F_w that maps each tuple $(\psi(X_1), \ldots, \psi(X_n))$ to the set of a, such that $\upsilon(\mathcal{I}_{\varphi}^{\mathcal{F}}(\mathbf{A})(\mathsf{a})) = \mathtt{F}$. These sets are always nonempty, so by the axiom of choice (not the one on the object level, which we have not included into $\mathcal{HOL}$ but the one on the meta level we are reasoning on) there is a total function f_w mapping each tuple $(\psi(X_1), \ldots, \psi(X_n))$ to some element of $\mathsf{F}_w(\psi(X_1), \ldots, \psi(X_n))$. Thus $\mathcal{F}' := \mathcal{F} * \mathsf{f}_w$ is an $\mathcal{R}'$-correspondence. Furthermore, we have

$$\mathcal{I}_{\varphi}^{\mathcal{F}'}(\mathbf{A}w) = \mathcal{I}_{\varphi}^{\mathcal{F}}(\mathbf{A})(\mathsf{f}_w(\varphi(X_1), \ldots, \varphi(X_n))) = \mathtt{F}$$

[25] EdNote: Work in $\Sigma\mathcal{R}$-models here.

for all assignments φ into $\mathcal{M}$ and thus $\mathcal{M} \models \mathcal{D}$ by definition.

For the converse direction let $\mathcal{M} \models \mathcal{D}$. We assume the existence of an $\mathcal{R}'$-correspondence $\mathcal{F}'$ for $\mathcal{M}$ such that $\upsilon(\mathcal{I}_\varphi^{\mathcal{F}'}(\mathbf{A}w)) = \mathtt{F}$ for all assignments φ. Since $w \in \mathbf{Dom}(\mathcal{R}')$ there must be a function $\mathsf{f}_w : \mathcal{D}_{\alpha_1} \times \cdots \times \mathcal{D}_{\alpha_n} \longrightarrow \mathcal{D}_\beta$ in $\mathcal{F}'$. Let $\mathcal{F} := \mathcal{F}' \setminus \{\mathsf{f}_w\}$, then $\mathcal{F}$ is an $\mathcal{R}$-correspondence and $\mathcal{I}_\varphi^{\mathcal{F}}(\mathbf{A}w) = \mathcal{I}_\varphi^{\mathcal{F}}(\mathbf{A})(\mathsf{f}_w(\varphi(X_1), \ldots, \varphi(X_n)))$, and therefore $\upsilon(\mathcal{I}_\varphi^{\mathcal{F}}(\Pi^\alpha \mathbf{A})) = \mathtt{F}$, since υ is a valuation. Since we have taken φ to be an arbitrary valuation, we have $\mathcal{M} \models \mathcal{R}.\mathcal{T}$.

For the $\mathcal{HT}(subst)$ case we have an $\mathcal{R}$-solved pair $X =^? \mathbf{C}$ in $\mathbf{T}$, and instantiate $\mathbf{T}$ by the $\mathcal{R}$-substitution $[\mathbf{C}/X]$. The new variable condition $\mathcal{R}'$ is $\mathcal{R}(\mathbf{C}/X)$. Now, any $\Sigma\mathcal{R}$-model of $\mathcal{R}.\mathcal{T}$ can be extended to a $\Sigma\mathcal{R}(\mathbf{C}/X)$-model for $\mathcal{R}(\mathbf{C}/X).[\mathbf{C}/X]\mathcal{T}$ by the substitution value theorem 4.2.6: the truth values of the formulae in the tableau are not changed by the instantiation as we have $\mathcal{I}^{\mathcal{F}}_{\varphi,[\mathcal{I}_\varphi^{\mathcal{F}}(\mathbf{C})/X]}(\mathbf{A}) = \mathcal{I}_\varphi^{\mathcal{F}(\mathbf{A}/X)}([\mathbf{C}/X]\mathbf{A})$. □

Corollary 5.2.3 (Soundness of $\mathcal{HT}$)
$\mathcal{HT}$-theorems are valid in the class of Σ-models.

Proof: Let $\mathbf{A}$ be a $\mathcal{HT}$-theorem, then by definition $\mathbf{A}^{\mathtt{F}}$ has a closed tableau $\mathcal{R}.\mathcal{T}$. As $\mathcal{HT}$ is model-preserving, there cannot be a model $\mathcal{M} = (\mathcal{D}, \upsilon, \mathcal{I})$ with $\mathcal{I}_\varphi(\mathbf{A}) = \mathtt{F}$, in other words, $\mathbf{A}$ is valid. □

5.3 Completeness of $\mathcal{HT}$

A central part of the completeness proofs for unification-based refutation calculi are the lifting properties. The central lifting theorem for $\mathcal{HT}$ states that any given tableau refutation for $\theta(\mathbf{A})$, can be lifted to a tableau refutation for $\mathbf{A}$. For the construction of a lifted $\mathcal{HT}$-tableau $\mathcal{R}.\mathcal{T}$ for a proposition $\mathbf{A}$ from a tableau $\mathcal{R}_\theta.\mathcal{T}_\theta$ for $\theta(\mathbf{A})$ it is crucial to maintain a tight correspondence $\omega : \mathcal{T}_\theta \longrightarrow \mathcal{T}$ between $\mathcal{R}_\theta.\mathcal{T}_\theta$ and $\mathcal{R}.\mathcal{T}$ that respects labels and is compatible with θ, i.e. for any node $\mathcal{N}$ in $\mathcal{T}$ with labeled formula $\mathbf{A}^v$ we have $\omega_{\mathcal{N}}(\theta(\mathbf{A})) = \mathbf{A}$. The main difficulty with lifting properties in higher-order logic is the fact that due to the existence of predicate variables at the head of formulae, the propositional structure of formulae can change during instantiation. For instance if $\theta(F_{\alpha\to o}) = \lambda X_\alpha.GX \lor p_o$, and $\mathbf{A}^{\mathtt{T}} = Fa^{\mathtt{T}}$ is the formula of $\mathcal{N}$, then $\mathcal{HT}(\lor)$ is applicable in $\mathcal{T}_\theta$ but not in the fragment of $\mathcal{T}$ already constructed. The solution of this problem is to apply $\mathcal{HT}(prim)$ with a suitable imitation binding $\mathcal{G}^{\lor}_{\alpha\to o} = \lambda X_\alpha.(H^1X) \lor (H^2X)$ and to obtain a node $\mathcal{N}'$ with formula $(H^1a \lor H^2a)^{\mathtt{T}}$, to which $\mathcal{HT}(\lor)$ can be applied. Since $\mathcal{G}^{\lor}_{\alpha\to o}$ is more general than $\theta(F)$ there is a substitution ρ, such that $\theta(F) = \rho(\mathcal{G}^{\lor}_{\alpha\to o})$, therefore $\omega_{\mathcal{N}'}(\theta'(H^1a \lor H^2a)^{\mathtt{T}}) = (H^1a \lor H^2a)^{\mathtt{T}}$ where $\theta' = \theta \cup \rho$.

embedding!tableau
tableau!embedding
***covers* Φ *mod* θ**
compatible

Definition 5.3.1 (Tableau Embedding)
Let $\mathcal{R}.\mathcal{T}$ and $\mathcal{R}'.\mathcal{T}'$ be higher-order tableaux such that Φ and Φ' are the respective sets of nodes in $\mathcal{R}.\mathcal{T}$ and $\mathcal{R}'.\mathcal{T}'$, then we call an injection $\omega\colon \Phi \longrightarrow \Phi'$ a **tableau embedding**, iff

- it preserves dominance in trees (i.e. if node $\mathcal{M}$ dominates node $\mathcal{N}$ in $\mathcal{T}$, then $\omega(\mathcal{M})$ dominates $\omega(\mathcal{N})$ in $\mathcal{T}'$) and furthermore
- it preserves truth values, i.e. $\omega(\mathbf{M}^\alpha)$ is of the form $\mathbf{N}^\alpha$. Here (and in the following) we also use ω for the mapping on formulae induced by ω ($\mathbf{M}$ and $\mathbf{N}$ are the formulae at nodes $\mathcal{M}$ and $\mathcal{N} = \omega(\mathcal{M})$).

For the lifting argument it will also be important to monitor the solvability of the unification constraint represented by the collection of leaves of a tableau. In particular, we will have to relate the sets of unification pairs of the basic and the lifted tableau modulo the connecting substitution θ. The condition we want to maintain is that the set of solutions of the lifted constraints, together with those induced by θ should always be the set of solutions of the ground constraints.

Definition 5.3.2 (Covers mod θ)
Let $\Phi \subseteq \mathbf{SUB}(\Sigma;\Delta \to \Xi)$, and $\Psi \in \mathbf{SUB}(\Sigma;\Gamma \to \Xi)$ be sets of substitutions and let $\theta \in \mathbf{SUB}(\Sigma;\Gamma \to \Delta)$. Furthermore let $\tau_\theta(\rho) := \rho \circ \theta$ for all $\rho \in \Phi$, then we say that Ψ **covers Φ mod** θ, iff $\tau_\theta(\Phi) = \Psi$.

Lemma 5.3.3 *Let θ be a $\mathcal{R}$-substitution, then $\mathbf{PU}(\mathcal{R}(\theta).\theta(\mathcal{E}))$ covers $\mathbf{PU}(\mathcal{R}.\mathcal{E} \vee \mathcal{E}_\theta)$ mod θ, i.e. $\tau_\theta(\mathbf{PU}(\mathcal{R}(\theta).\theta(\mathcal{E}))) = \mathbf{PU}(\mathcal{R}.\mathcal{E} \vee \mathcal{E}_\theta)$.*

Proof: We have $\tau_\theta(\mathbf{PU}(\mathcal{R}(\theta).\theta(\mathcal{E})) \subseteq \mathbf{PU}(\mathcal{R}.\mathcal{E} \vee \mathcal{E}_\theta)$, since for any substitution $\sigma \in \mathbf{PU}(\mathcal{R}(\theta).\theta(\mathcal{E}))$ we have $\tau_\theta(\sigma) = \sigma \circ \theta$ is a pre-unifier of θ and of $\mathcal{E}_\theta$ (θ is the most general unifier of $\mathcal{E}_\theta$).

On the other hand, any pre-unifier ρ of $\mathcal{R}.\mathcal{E} \vee \mathcal{E}_\theta$ is also one of $\mathcal{R}.\mathcal{E}_\theta$, so we have $\rho = \rho' \circ \theta = \tau_\theta(\rho')$ by Lemma 3.2.8, since θ is the most general unifier of $\mathcal{R}.\mathcal{E}_\theta$. Thus we have $\mathbf{PU}(\mathcal{R}.\mathcal{E} \vee \mathcal{E}_\theta) \subseteq \tau_\theta(\mathbf{PU}(\mathcal{R}(\theta).\theta(\mathcal{E}))$, which completes the proof. $\square$

Definition 5.3.4 (θ-Compatible)
Let $\mathcal{R}.\mathcal{T}$ and $\mathcal{R}_\theta.\mathcal{T}_\theta$ be higher-order tableaux and θ an $\mathcal{R}$-substitution. We say that a tableau embedding $\omega\colon \mathcal{T}_\theta \longrightarrow \mathcal{T}$ is **θ-compatible**, iff $\omega(\theta(\mathbf{M}^\alpha)) = \mathbf{M}^\alpha$ and moreover $\mathbf{PU}(\mathcal{R}.\mathcal{E}_\mathcal{T})$ covers $\mathbf{PU}(\mathcal{R}_\theta.\mathcal{E}_{\mathcal{T}_\theta})$ mod θ, i.e. $\tau_\theta(\mathbf{PU}(\mathcal{T}_\theta)) = \mathbf{PU}(\mathcal{T})$.

Lemma 5.3.5 (Mimicking head $\vee$)
For any $\mathbf{A} \in wf\!f_o(\Sigma)$ and any substitution θ, such that $\theta(\mathbf{A}) = \theta(\mathbf{B}) \vee \theta(\mathbf{C})$, there is a tableau $\mathcal{T}_\vee$ with substitution $\theta_\vee$, root $\mathbf{A}^\mathtt{T}$, and the two open leaves $\mathbf{B}^\mathtt{T}$ and $\mathbf{C}^\mathtt{T}$ and a substitution $\rho_\vee$ with $\mathbf{Dom}(\rho_\vee) \cap \mathbf{Free}(\mathbf{A}) = \emptyset$ and $\theta = \rho_\vee \circ \theta_\vee[\mathbf{Dom}(\theta_\vee)]$.

Proof: Let $\mu(\theta, \mathbf{A})$ be the multiset of term depths in $\theta(\mathbf{Free}(\mathbf{A}))$. We prove the assertion by well-founded induction over the multi-set ordering for natural numbers. If $\mathbf{A} = \mathbf{B} \vee \mathbf{C}$, then $\mathcal{T}$ consists of a single application of $\mathcal{HT}(\vee)$ and $\theta_\vee = \rho_\vee = \emptyset$. Note that this subsumes the base case where $\mu(\theta, \mathbf{A}) = \emptyset$, since then $\theta(\mathbf{A}) = \mathbf{A}$.

If $\mathbf{head}(\mathbf{A}) \neq \vee$, then $\mathbf{A}$ must be flexible, i.e. $\mathbf{A} = P_\alpha \overline{\mathbf{U}^n}$ for some variable $P_\alpha \in \mathbf{Dom}(\theta)$ and $h = \mathbf{head}(\theta(P)) \in \{\vee, j\}$. Therefore, by the general binding theorem 3.4.2 there is a partial binding $\mathbf{G}_\alpha^h$, a substitution ρ such that $\rho(\mathbf{G}_\alpha^h) =_{\beta\eta} \theta(P_\alpha)$. Now let $\theta' := \theta \cup \rho$, then $\theta'([\mathbf{G}_\alpha^h/P]\mathbf{A}) = \theta(\mathbf{A})$. Furthermore $\mu(\theta', [\mathbf{G}_\alpha^h/P]\mathbf{A}) < \mu(\theta, \mathbf{A})$, since $\mathbf{dp}(\rho) < \mathbf{dp}(\theta(P))$ and $P \notin \mathbf{Free}([\mathbf{G}_\alpha^h/P]\mathbf{A})$. So by inductive hypothesis, there is a tableau $\mathcal{T}'_\vee$ with substitution $\theta'_\vee$ and root $[\mathbf{G}_\alpha^h/P]\mathbf{A}$ and a substitution $\rho'_\vee$, such that $\theta' = \rho'_\vee \circ \theta'_\theta[\mathbf{Dom}(\theta'_\vee)]$. If we combine $\mathcal{T}'_\vee$ with a $\mathcal{HT}(prim)$ step at the root, we have the tableau shown in Figure 5.5 which has root $\mathbf{A}$.

Now let $\theta_\vee := \theta'_\vee \cup [\mathbf{G}_\alpha^h/P]$ and $\rho_\vee := \rho'_\vee \cup \rho$, then

$$\begin{aligned}
\rho_\vee \circ \theta_\vee &= (\rho'_\vee \cup \rho) \circ (\theta'_\vee \cup [\mathbf{G}_\alpha^h/P]) \\
&= (\rho'_\vee \circ \theta'_\vee) \cup (\rho \circ [\mathbf{G}_\alpha^h/P]) \\
&= (\rho'_\vee \circ \theta'_\vee) \cup ([\theta(P)/P]) = \theta[\mathbf{Dom}(\theta_\vee)]
\end{aligned}$$

since $\mathbf{Dom}(\theta_\vee) = \mathbf{Dom}(\theta'_\vee) * P$ by construction. □

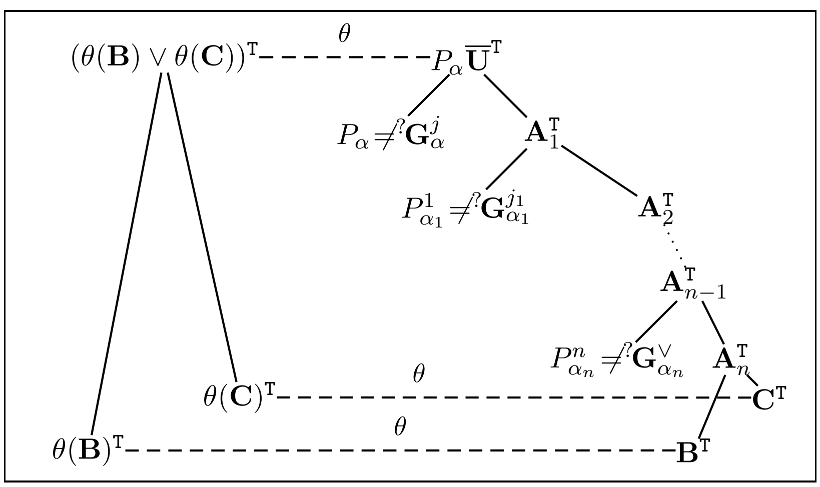


Figure 5.5: Tableau Lifting

Clearly, similar results for the other structural $\mathcal{HT}$*rules* can be shown by exactly the same methods. Note that the mapping given by the dashed lines in Figure 5.5 interpreted from left to right is a θ-compatible tableau embedding, since there are no unification constraints present. We will use this in the proof of the next Result, which incorporates the main part of the proof of the tableau lifting result in Theorem 5.3.7 (we need a strengthened assertion to make the induction go through).

Lemma 5.3.6 *Let* $\mathbf{A} \in wff_o(\Sigma)$*,* θ *a substitution, and* $\mathcal{R}_\theta.\mathcal{T}_\theta$ *be a tableau for* $\theta(\mathbf{A})$*, then there is a tableau* $\mathcal{R}.\mathcal{T}$ *for* $\mathbf{A}$ *and a* θ*-compatible tableau inclusion* $\omega\colon \mathcal{R}_\theta.\mathcal{T}_\theta \longrightarrow \mathcal{R}.\mathcal{T}$*.*

Proof: We construct $\mathcal{R}.\mathcal{T}$, and ω inductively on the structure of $\mathcal{T}_\theta$. If $\mathcal{R}_\theta.\mathcal{T}_\theta$ is an initial tableau, then we we take ω to be the trivial tableau embedding between one-node tableaux. ω is θ-compatible, since there are no unification pairs present.

Now let us assume that the assertion holds for $\omega\colon \mathcal{R}_\theta.\mathcal{T}_\theta \longrightarrow \mathcal{R}.\mathcal{T}_\theta$ and θ and look at possible extensions of $\mathcal{T}_\theta$. If $\mathcal{T}_\theta$ is extended by $\mathcal{HT}(\vee)$ at node $\mathcal{N}_\theta$ to $\mathcal{T}'_\theta$, then $\mathcal{N}_\theta$ must be labeled with $\theta(\mathbf{A})^{\mathtt{T}} = (\theta(\mathbf{A}) \vee \theta(\mathbf{B})^{\mathtt{T}})$. As ω is θ-compatible, the node $\omega(\mathcal{N}_\theta)$ in $\mathcal{T}$ must have the label $\mathbf{A}^{\mathtt{T}}$. Lemma 5.3.5 guarantees a tableau $\mathcal{T}_\vee$ with root $\mathbf{A}$, leaves $\mathbf{B}$ and $\mathbf{C}$, and substitution $\theta_\vee$.

Let $\mathcal{T}'$ be $\mathcal{T}$ extended with $\mathcal{T}_\vee$ at $\omega(\mathcal{N}_\theta)$ and $\mathcal{R}' := \mathcal{R}(\theta_\vee)$, then $\mathcal{E}_{\mathcal{T}'} = \theta_\vee(\mathcal{E}_\mathcal{T})$, since $\theta_\vee$ is the substitution of $\mathcal{T}_\vee$ and $\mathcal{T}_\vee$ does not have unsolved constraints. Now let ω' be the extension of ω that maps the nodes with labels $\theta(\mathbf{B})$ and $\theta(\mathbf{C})$ to the leaves with labels $\mathbf{B}$ and $\mathbf{C}$ of $\mathcal{T}_\vee$. If we set $\widetilde{\theta} := \theta_\vee \circ \theta$, then it only remains to show that ω' is $\widetilde{\theta}$-compatible.

$$\begin{array}{rcl} \mathbf{PU}(\mathcal{R}'.\mathcal{E}_{\mathcal{T}'}) & = & \mathbf{PU}(\mathcal{R}(\theta_\vee).\theta_\vee(\mathcal{E}_\mathcal{T}) \vee \mathcal{E}_\theta) \\ & \equiv_{\theta_\vee} & \mathbf{PU}(\mathcal{R}.\mathcal{E}_\mathcal{T}) \\ & \equiv_\theta & \mathbf{PU}(\mathcal{R}(\theta).\theta(\mathcal{E}_\mathcal{T})) \end{array}$$

by Lemma 5.3.3 and inductive hypothesis. Thus Lemma **??** completes the assertion for the $\mathcal{HT}(\vee)$ case.

The cases for the other structural $\mathcal{HT}$ rules can be shown by exactly the same methods, using the respective analogs of Lemma 5.3.5.

In the $\mathcal{HT}(cut)$ case, we have have the left situation below, so we can extend $\mathcal{T}$ by the right tableau.

$$\frac{\begin{array}{c}\theta(\mathbf{A})^v \\ \theta(\mathbf{B})^w\end{array}}{\theta(\mathbf{A}) \neq^? \theta(\mathbf{B})}\,\mathcal{HT}(cut) \qquad\qquad \frac{\begin{array}{c}\mathbf{A}^v \\ \mathbf{B}^w\end{array}}{\mathbf{A} \neq^? \mathbf{B}}\,\mathcal{HT}(cut)$$

Since tableau embeddings do not extend to unification pairs, we can take $\widetilde{\omega} = \omega$ and $\widetilde{\theta} = \theta$. This extension adds a new pair $\theta(\mathbf{A}) \neq^? \theta(\mathbf{B})$ to the unification constraints of $\mathcal{T}_\theta$ and $\mathbf{A} \neq^? \mathbf{B}$ of those of $\mathcal{T}$. Now, lemma 5.3.3 gives

$$\mathbf{PU}(\mathcal{R}(\theta).\mathcal{E}_{\mathcal{T}_\theta} \wedge \theta(\mathbf{A}) \neq^? \theta(\mathbf{B})) \equiv_\theta \mathbf{PU}(\mathcal{R}.\mathcal{E}_\mathcal{T} \wedge \mathcal{E}_\theta \wedge \mathbf{A} \neq^? \mathbf{B})$$

Similarly, the simplification rule $\mathcal{SIM}(dec)$ does not affect the tableau embedding ω, since that is restricted to labeled formulae and θ, since $\mathcal{SIM}(dec)$ it conserve sets of pre-unifiers. Finally, for $\mathcal{HT}(subst)$ and $\mathcal{HT}(prim)$ which instantiate the tableau by a substitution $[\mathbf{A}/X]$ where $X \notin \mathbf{Dom}(\theta)$, thus we can lift their applications directly (i.e. create the lifted tableau, by applying exactly the same substitution to $\mathcal{T}$).

Now we have examined all the cases of tableau construction and have thus completed the proof of the assertion. $\square$

Corollary 5.3.7 (Tableau Lifting)
Let $\mathbf{A} \in \mathit{wff}_o(\Sigma)$ and θ a substitution, then $\mathbf{A}$ has a $\mathcal{HT}$-proof provided $\theta(\mathbf{A})$ has one.

Proof: Let $\mathcal{R}_\theta.\mathcal{T}_\theta$ be a tableau proof for $\theta(\mathbf{A})$. By Lemma 5.3.6, there is a tableau $\mathcal{R}.\mathcal{T}$ for $\mathbf{A}$ and a θ-compatible tableau inclusion $\omega\colon \mathcal{R}_\theta.\mathcal{T}_\theta$. Now as $\mathcal{T}_\theta$ is closed, we have $\mathbf{PU}(\mathcal{R}_\theta.\mathcal{E}_{\mathcal{T}_\theta}) \neq \emptyset$ and the leaves of $\mathcal{T}$ can only be unification constraints (ω is a tableau embedding). Moreover, ω is θ-compatible, so $\mathbf{PU}(\mathcal{R}_\theta.\mathcal{E}_{\mathcal{T}_\theta}) \equiv_\theta \mathbf{PU}(\mathcal{R}.\mathcal{E}_\mathcal{T})$, so $\emptyset \neq \mathbf{PU}(\mathcal{R}.\mathcal{E}_\mathcal{T} \vee \mathcal{E}_\theta) \subseteq \mathbf{PU}(\mathcal{R}.\mathcal{E}_\mathcal{T})$ in other words, $\mathcal{R}.\mathcal{T}$ is closed and yields the assertion. $\square$

Example 5.3.8 (Tableau Lifting)
Consider the concrete lifting situation in Figure 5.6 with the instantiated version on the right hand side.

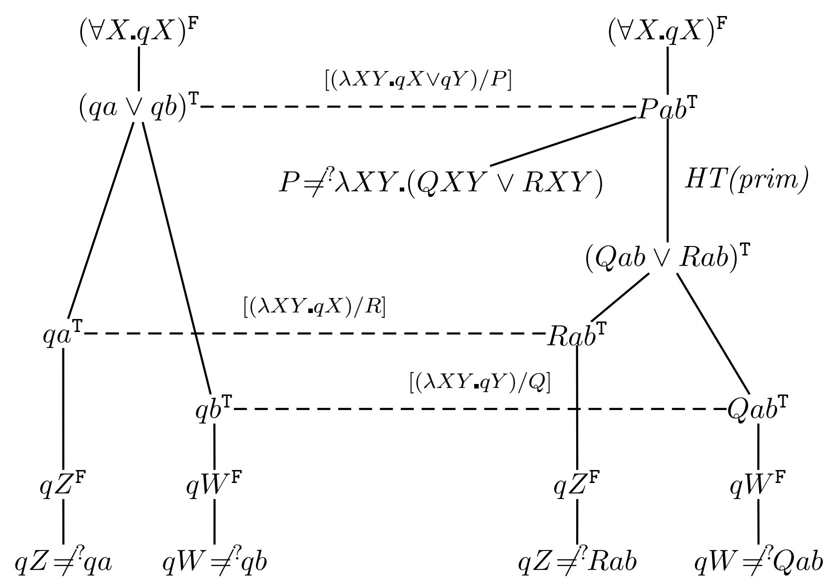
Theorem 5.3.9 (Completeness)
$\mathcal{HT}$ is refutation complete with respect to the class of Σ-models, i.e. if Φ is a valid set of sentences, then there is a closed higher-order tableau for $\emptyset.\Phi^{\mathsf{F}}$.

Proof: Completeness of $\mathcal{HT}$ can be proven using the model existence theorem 2.2.32 by verifying that the class Γ_Σ defined by

$$\Gamma_\Sigma := \{\Phi \subset \mathit{wff}_o(\Sigma) \mid \not\vdash_{\mathcal{HT}} \Phi^{\mathsf{T}}\}$$

is a saturated abstract consistency class. For conditions 1. and 3.– 5., and 7. in the definition of abstract consistency class (cf. definition 2.2.7)this can be achieved with the usual techniques (see for instance [Fit90]). Condition 2. is trivially met, since all formulae are kept in $\beta\eta$-normal form and we have treated condition 6. in the tableau lifting theorem above.

For 8. let $\Phi = \Phi' * \neg(\mathbf{A} =^{\alpha\to\beta} \mathbf{B})$ and let $\mathcal{R}.\mathcal{T}$ be a closed tableau for $\Phi * (\neg\mathbf{A}w = \mathbf{B}w)$ for some witness constant w that does not occur in Φ. The first tableau in Figure 5.7 is a $\mathcal{HT}$-refutation of Φ. Note that the witness constant r is eliminated again from the unification constraint by $\mathcal{SIM}(dec)$, therefore the intermediate variable condition $\mathcal{R}' := \mathcal{R} * \{r \mapsto \mathbf{Free}(\mathbf{A}) \cup \mathbf{Free}(\mathbf{B})\}$ in Figure 5.7 can be simplified to $\mathcal{R}$.



$\mathbf{PU}(qZ \neq^? qa \vee qW \neq^? qb) = [a/Z], [b/W] =_\theta \mathbf{PU}(qW \neq^? Qab \vee qZ \neq^? Rab)$,
where $\theta = [(\lambda XY.qX)/R], [(\lambda XY.qY)/Q], [(\lambda XY.(QXY \vee RXY))/P]$

Figure 5.6: A concrete tableau lifting situation

Φ'	$(\mathbf{A} = \mathbf{B})^{\mathtt{F}}$
$\mathbf{A} =^{\alpha\to\beta} \mathbf{B}^{\mathtt{F}}$	$(\forall P_{\alpha\to o}.P\mathbf{A} \Rightarrow P\mathbf{B})^{\mathtt{F}}$
$\mathbf{A} \neq^? \mathbf{B}$	$r\mathbf{A}^{\mathtt{T}}$
$\mathbf{A}w_\alpha \neq^? \mathbf{B}w$	$r\mathbf{B}^{\mathtt{F}}$
$(\mathbf{A}w =^\beta \mathbf{B}w)^{\mathtt{F}}$	$r\mathbf{A} \neq^? r\mathbf{B}$
$\mathcal{T}'$	$\mathbf{A} \neq^? \mathbf{B}$

(The first step in the left tableau abbreviates the right.)

Figure 5.7: Functional Extensionality for Γ_Σ.

To show saturation of Γ_Σ we assume that $\Phi * \mathbf{A} \notin \Gamma_\Sigma$ and $\Phi * \neg\mathbf{A} \notin \Gamma_\Sigma$ for some proposition $\mathbf{A} \in \mathit{wff}_o(\Sigma)$ and some $\Phi \in \Gamma_\Sigma$. In other words, there are $\mathcal{HT}$-refutations for both $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$, and therefore there is one for $\Phi \cup \{\mathbf{A} \vee \neg\mathbf{A}\}$, and consequently there is a $\mathcal{HT}$-refutation for Φ, since
EdNote(26) tautologies can be eliminated in $\mathcal{HT}$[26]. This contradicts our assumption that $\Phi \in \Gamma_\Sigma$. □

[26] EdNote: Show that somewhere

5.4 Optimizations for Primitive Substitutions

primitive substitution
substitution!primitive

The primitive substitution rules have originally been introduced by Peter Andrews in [And89]. Note that the set of general bindings is infinite, since we need one for every quantifier Π^α and the set of types is infinite. Thus in contrast to the goal-directed search for instantiations in unification, the rule $\mathcal{HT}(prim)$ performs blind search and even worse, is infinitely branching. Therefore, the problem of finding instantiations for predicate variables is conceived as the limiting factor to higher-order automated theorem proving.

Remark 5.4.1 (Atomic Cut and $\mathcal{HT}(prim)$ in $\mathcal{HT}$)
From the completeness proof above we can directly verify, that the $\mathcal{HT}(cut)$ rule can be restricted to the case where the formulae $\mathbf{A}^\mathsf{T}$ and $\mathbf{B}^\mathsf{F}$ are literals. Indeed, the base cases of the definition of abstract consistency class only mention atomic formulae and the lifting argument lifts atomic cuts to atomic cuts. This restriction considerably restricts the possibilities of applications of $\mathcal{HT}(cut)$ and thus prunes the search spaces associated with proof search in $\mathcal{HT}$ considerably.

In Example 5.3.8, we have shown a special instance of the lifting process used in 5.3.7. This uses $\mathcal{HT}(prim)$ to instantiate the variable P with a disjunction binding so that the literal $P\overline{\mathbf{U}}^\mathsf{T}$ can be split into the two tableau branches present in the ground proof. Indeed this is necessary, if we want to prune search spaces by restricting $\mathcal{HT}(cut)$ to atomic cuts. However, if we do not impose this restriction, then $\exists P.(Pab) \Rightarrow \forall X.qX$ has a closed $\mathcal{HT}$-tableau that does not need $\mathcal{HT}(prim)$ (see Figure 5.8). This suggests that without the restriction to atomic cuts, $\mathcal{HT}$ can handle many theorems without $\mathcal{HT}(prim)$. Clearly, however, we have to extend the $\mathcal{HT}(cut)$ rule by a variant, that acts on arbitrary pairs of formulae $\mathbf{A}^v$, $\mathbf{B}^v$ with equal annotation in the branch (introducing a negation operator for soundness).

$$\frac{\begin{array}{c}\mathbf{A}^v\\ \mathbf{B}^v\end{array}}{\mathbf{A} \neq^? \neg\mathbf{B}} \qquad \frac{\begin{array}{c}\mathbf{A}^v\\ \mathbf{B}^v\end{array}}{\neg\mathbf{A} \neq^? \mathbf{B}}$$

Since the primitive substitution rule $\mathcal{HT}(prim)$ basically employs blind search for a propositional substitution, this variant of $\mathcal{HT}$ may well be more efficient, since unification generates the necessary instantiations in a goal-driven manner from the material already present in the tableau.

Note that we probably cannot hope that $\mathcal{HT}$ is complete without $\mathcal{HT}(prim)$, as the following examples (due to Chad E. Brown) suggest:

$$\exists S_o \forall Y_\iota.(S \Rightarrow p_{\iota\to o}) \land \exists V_\iota.pV \Rightarrow S$$
$$\forall A, B_{\alpha\to o}.\exists C_{\alpha\to o}.((A \subseteq C) \land (B \subseteq C)) \land \forall D_{\alpha\to o}.A \subseteq D \land B \subseteq D \Rightarrow C \subseteq D$$

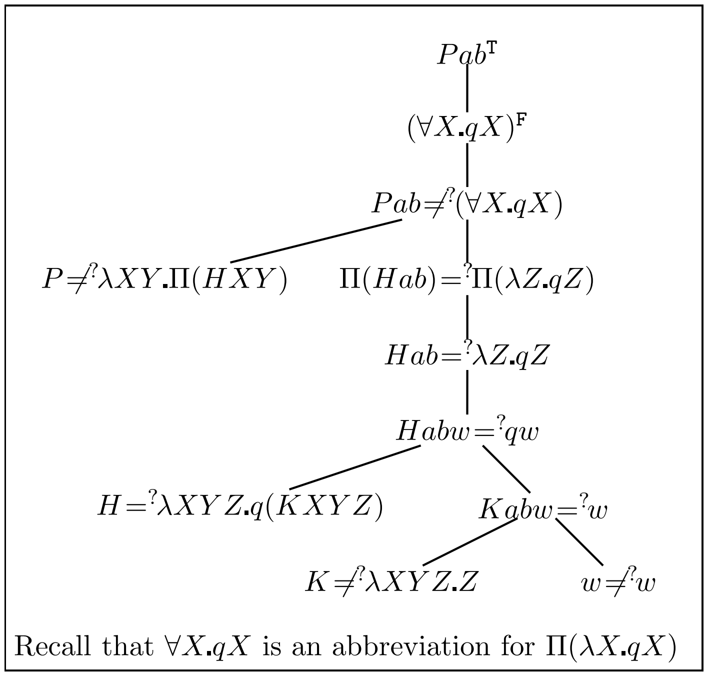


Figure 5.8: A closed tableau without $\mathcal{HT}(prim)$.

In the first case we want $[\forall Y. pY/S]$ and in the second one $C = A \cup B$, both of which are not not present in any subformula of the assertion. Even if the calculus is not complete, it might well be able to handle for instance generalizations in inductive proofs. Consider for instance the formula $\forall X_\iota. X + (X + X) = (X + X) + X$, which can only be proven by a detour to general associativity of $+$.

5.5 Extensional Tableaux

In the previous Section we have proven completeness of $\mathcal{HT}$ with respect to Σ-models, which characterizes traditional higher-order refutation calculi, but is not a very intuitive notion of semantics. Indeed $\mathcal{HT}$ is not complete with respect to Henkin models. Consider for instance the formula $\mathbf{C} := \neg(c_{o \to o} b_o) \vee (c(\neg\neg b))$ which cannot be proven in $\mathcal{HT}$ — the middle tableau in Figure 5.9 shows the maximal tableau for $\mathbf{C}$, which cannot be closed, since no $\mathcal{HT}$ construction rule applies to the last pair. This lack of completeness is unfortunate, since the class of Henkin models is the most intuitive one that admits complete calculi. In particular, our mathematical intuition which conforms to Henkin models would make us believe that $\mathbf{C}$ should be refutable, because $\neg\neg b$ is provably equivalent to b. This example shows us that in extensional calculi we have to deal with propositions that appear in the arguments of parameters. The simplest approach to build a calculus that can refute $\mathbf{C}$ is to add the equational theory $b = \neg\neg b$ to higher-order unification. Even though this approach is intuitive, it does not solve the general problem of incorporating extensionality into higher-order tableaux. In fact, we can generalize the formula $\mathbf{C} := (cb) \vee \neg c(\neg\neg b)$ to $\mathbf{C}' := (c\mathbf{A}) \vee \neg(c\mathbf{B})$, where $\mathbf{A}$ and $\mathbf{B}$ are arbitrary propositions. Now $\mathbf{C}'$ is valid in the class of Henkin models, iff $\mathbf{A} \Leftrightarrow \mathbf{B}$ is valid. So the approach of enhancing the unification would require augmenting the unification procedure by the theory of logical equivalence, which would enable the unification procedure to prove any theorem by unifying it with some elementary tautology like $\forall X_o. X \vee \neg X$.

The semantic problem with completeness behind this example is maybe best illustrated by the fact that in Σ-models equivalence fails to entail equality means that substitutivity of equivalence does not hold for Σ-models (recall the Leibniz definition of equality) and thus "buried occurrences" propositions cannot be substituted for equivalent ones. Thus the remedy for the incompleteness of $\mathcal{HT}$ is a tableau construction rule that relates equality in unification constraints with equivalence.

Definition 5.5.1 (Extensional Tableau Calculus $\mathcal{HTE}$)
The tableau construction rules for $\mathcal{HTE}$ consist of those for $\mathcal{HT}$ augmented with the one from Figure 5.9. This rule interprets unification pairs of type

$$\frac{\mathbf{A}_o \neq^? \mathbf{B}_o}{\begin{array}{c|c} \mathbf{A}^{\mathsf{F}} & \mathbf{A}^{\mathsf{T}} \\ \mathbf{B}^{\mathsf{T}} & \mathbf{B}^{\mathsf{F}} \end{array}}\ \mathcal{HTE}(ext) \qquad \begin{array}{c} (\neg(cb) \vee c(\neg\neg b))^{\mathsf{F}} \\ \neg(cb)^{\mathsf{F}} \\ c(\neg\neg b)^{\mathsf{F}} \\ cb^{\mathsf{T}} \\ cb \neq^? c(\neg\neg b) \\ b \neq^? \neg\neg b \end{array} \qquad \begin{array}{c} b \neq^? \neg\neg b \\ \begin{array}{c|c} \neg\neg b^{\mathsf{F}} & \neg\neg b^{\mathsf{T}} \\ b^{\mathsf{T}} & b^{\mathsf{F}} \\ b^{\mathsf{F}} & b^{\mathsf{T}} \\ b \neq^? b & b \neq^? b \end{array} \end{array}$$

Figure 5.9: Extensional Tableaux in $\mathcal{HTE}$

o as equivalences.

In particular we can continue our example above with the right sub-tableau in Figure 5.9. Note that $\mathcal{HTE}$ completely blurs the distinction between propositional reasoning by decomposition of formulae and substitution construction by unification constraint solving. In particular we can indeed prove a sentence $\mathbf{A}$ by unifying it with a simple tautology, such as $\forall X_o. X \vee \neg X$. We will now explore this symmetry of propositional reasoning and unification.

Theorem 5.5.2 (Soundness and Completeness)
$\mathcal{HTE}$ is sound and refutation complete, i.e. a proposition $\mathbf{A} \in \mathit{wff}_o(\Sigma)$

non-degenerate
degenerate
formula!degenerate

is valid in all Henkin models, iff there is a closed higher-order extensional tableau for $\emptyset.\mathbf{A}^{\mathtt{F}}$.

Proof: Soundness of $\mathcal{HTE}(ext)$ is immediate: If $\mathbf{A}$ and $\mathbf{B}$ have different truth values, then one must be $\mathtt{T}$, while the other must be $\mathtt{F}$. Completeness is a consequence of the model existence theorem for Henkin models 2.2.32. To show that

$$\Gamma_\Sigma := \{\Phi \subset \mathit{wff}_o(\Sigma) \mid \not\vdash_{\mathcal{HTE}} \Phi^{\mathtt{T}}\}$$

is a saturated abstract consistency class, we build on 5.3.9, so we now only have to verify the condition 9.

$$\begin{array}{|c|c|}
\hline
\multicolumn{2}{|c|}{\Phi'} \\
\multicolumn{2}{|c|}{(\mathbf{A} =^o \mathbf{B})^{\mathtt{F}}} \\
\multicolumn{2}{|c|}{\mathbf{A} \neq^? \mathbf{B}} \\
\mathbf{A}^{\mathtt{F}} & \mathbf{A}^{\mathtt{T}} \\
\mathbf{B}^{\mathtt{T}} & \mathbf{B}^{\mathtt{F}} \\
\mathcal{T}^1 & \mathcal{T}^2 \\
\hline
\end{array}$$

Figure 5.10: Extensionality for Γ_Σ

Let $\Phi = \Phi' * \neg(\mathbf{A} =^o \mathbf{B})$, and $\mathcal{R}.\mathcal{T}^1$ a closed tableau for $\Phi \cup \{\neg\mathbf{A}, \mathbf{B}\}$ and finally $\mathcal{R}.\mathcal{T}^2$ one for $\Phi \cup \{\mathbf{A}, \neg\mathbf{B}\}$ then the tableau in Figure 5.10 is a $\mathcal{HTE}$-refutation for Φ. Here Q is a new negative variable of type $\alpha \to o$. $\square$

Now that we have seen how we can achieve completeness with respect to Henkin models, let us note that completeness with respect Σ-models is still an interesting property for a higher-order calculus: For the fragment of formulae where every propositional subformula (i.e. one of type o) is dominated only by logical constants — let us call such formulae **non-degenerate** — the two notions of validity coincide. Since for most applications degenerate do not occur, if suffices to employ calculi without rules such as $\mathcal{HTE}(ext)$.

To put the ideas developed in this section into a broader perspective (we will make use of this in the next section) let us digress to a more general look at automatic theorem proving. Theorem proving is a syntactic process of making judgments about the validity of formulae in all models.

In propositional logic, formulae are built up from propositional variables, and the logical connectives $\neg$ and $\vee$. While the variables can be arbitrarily interpreted (to be either $\mathtt{T}$ or $\mathtt{F}$), the connectives $\neg$ and $\vee$ are interpreted to denote the negation and disjunction functions on the set of truth values. Thus the class of models consists only of the $\{\neg, \vee\}$-algebra with carrier set $\mathcal{D}_o = \{\mathtt{T}, \mathtt{F}\}$ where $\mathcal{I}(\neg)$ and $\mathcal{I}(\vee)$ are the well-known functions.

In first-order logic there is a clear conceptual distinction between terms (syntactic objects that denote individuals) and formulae (syntactic objects that denote truth values). Formulae are built up from atoms, the symbols $\neg$

and $\vee$, and quantification. Atoms take the place of propositional variables, whereas $\neg$, $\vee$, and quantification have fixed interpretations. Atoms are built up from predicate symbols and terms, which in turn are built up from function symbols, individual constants, and variables, all of which can be freely interpreted. Thus the class of models for first-order logic consists of some universe $\mathcal{D}_\iota$ of individuals and $\mathcal{D}_o$ with a fixed interpretation for $\neg$, $\vee$, and quantification.

Skolemization eliminates the treatment of quantification into a preprocess in refutation-based theorem proving. For instance, resolution-based calculi consist of the propositional rules (computation in the fixed part $\mathcal{D}_o$) and the unification procedure, which amounts to solving term equations in all models. Since the term algebra is the free algebra, it is sufficient to solve the term equations there.

Let us summarize these ideas. Due to the strong division of the model theory into a fixed part $\mathcal{D}_o$ and a free part $\mathcal{D}_\iota$, first-order theorem proving can be divided into a propositional part (acting on first-order formulae) and a term part (unification), which do not interfere. In higher-order logic (here simple type theory) we do not have this clear division. In particular, there are formulae, where symbols with a fixed interpretation are dominated (in the scope or subterms of arguments) by symbols with a flexible interpretation.

We have presented a calculus where the unification procedure calls the theorem proving procedure recursively on demand, i.e. whenever it encounters a propositional pair. In effect, this approach breaks down the distinction between unification and refutation and treat both processes in one uniform calculus.

5.6 Higher-Order Unification with Equivalence

In the introduction of the last section we have seen that in order to obtain a Henkin-complete tableau refutation procedure, we essentially have to augment higher-order unification with logical equivalences, and in fact, we have done just that with our calculus $\mathcal{HTE}$. It can either be seen as a Henkin-complete refutation calculus (if we start out with the negation of the theorem we are trying to prove) or as a higher-order unification calculus with equivalence (HOUE). In this section we will show an application of the second intuition, in natural language semantics extending and building on the material in section 3.7. We will take another parallelism phenomenon as an example: Corrections are utterances such as (5.1b–d) where a discourse participant (B) corrects the utterance of some other discourse participant (A).

(5.1) a. *A: Jon likes Mary.*

topic
sentence topic

b. *B: No, PETER likes Mary.*

c. ⋆ *B: No, PETER likes Sarah.*

d. *B: No, PETER likes [the woman with the red hat]*

Since the well-formedness of corrections depends on prosodic stress (focus), we have capitalized the stressed parts of the sentences in the example above. It is assumed that a correction is well-formed, if the correctum and correction are parallel and only in the stressed part. In particular, the utterance in (5.1c) does not form a well-formed correction for (5.1a), since the source utterance and the target are not parallel (Mary contrasts with Sarah which is not stressed). Following the ideas of [DSP91b, GKvL96] we can model the requirements corrections place on context with a variant of the equations used in section 3.7: If **S** and **T** are the source (correctum) and target (correction) sentences, then we use the equation

(5.2) a. $X(\mathbf{A}^1, \ldots, \mathbf{A}^n) = \mathbf{S}$ *and* $X(\mathbf{B}^1, \ldots, \mathbf{B}^n) = \mathbf{T}$

where X is the variable for the sentence **topic** (the parallel part) and the $\mathbf{A}^i$ and $\mathbf{B}^i$ are the respective source and target focus particles (the stressed parts). Note that in contrast to the verb phrase ellipsis case, there can be more than one contrasting part, e.g. "B: No, PETER likes SARAH." would be a perfectly good correction of (5.1.a) with two focus particles.

The pair of higher-order equations in (5.2) states that the semantics of the source utterance (Speaker A) and that of the must be contrastively parallel to that of the target utterance (Speaker B), i.e both have a common part (modeled by the predicate variable X in the equations) and can differ only in the contrastive elements $\mathbf{A}_i$ and $\mathbf{B}_i$. For (5.1) we have the equations

(5.3) a. $X(jon) = likes(jon, mary)$

b. $X(peter) = likes(peter, mary)$;
solution: $X = \lambda X.likes(X, mary)$

c. $X(peter) = likes(peter, sarah)$;
no solution

d. $X(peter) = \exists^1 x.(w(x) \wedge wrh(x) \wedge like(j, x))$;
solution ???

(5.3a) is the source equation and (5.3b-d) are the target equations corresponding to the corrections (5.1b-d). The HOU analysis predicts that (5.1b) is a correct correction to (5.1a) — the topic (computed as the value for X) is *liking Mary* — while (5.1c) is not. An interpretation with topic *liking* with additional contrastive element *Sarah* is impossible, since it is not prosodically marked. Finally, the analysis predicts that (5.1d) is incorrect, since HOU fails — even in a situation, where *Mary* is the unique woman with a red hat — as it is too weak to identify the semantic equivalence of the two descriptions of *Mary*.

Even though HOU considers $\beta\eta$-equality of formulae, it does not take into account the semantics of the logical connectives and quantifiers contained in the logical representation of natural language utterances. This is exactly the problem that prevents HOU from finding a solution in (5.3.d).

How does HOUE account for such examples? To show this, we now sketch the main steps of the unification process for example (5.3.d) with equations:

$$\begin{array}{ll} X(p) & = like(p, s) \\ X(j) & = \exists x(w(x) \wedge wrh(x) \wedge unique(x) \wedge like(j, x)) \end{array}$$

These are solved in a context, where Sarah is the only woman with a red hat. The HOUE method is given access to the hypotheses $unique(s)$, $w(s)$ and $wrh(s)$ by adding them to the initial tableau. In a first step, we solve the first equation to $An = \lambda z.like(z, s)$ and obtain the following tableau:

$$\begin{array}{c} unique(s) \\ w(s) \\ wrh(s) \\ X(p) = like(p, s) \\ \vdots \\ like(j, s) = \exists x(w(x) \wedge wrh(x) \wedge unique(x) \wedge like(j, x)) \end{array}$$

The HOUE rule $\mathcal{HTE}(ext)$ now splits the initial equation into two branches. The first one has the form

$$\begin{array}{c} like(j, s) \\ \neg\exists x(w(x) \wedge \ldots \wedge like(j, x)) \\ \begin{array}{c|c|c|c} \neg w(z) & \neg wrh(z) & \neg unique(z) & \neg like(j, z) \\ * \, [z = s] & *[z = s] & *[z = s] & *[z = s] \end{array} \end{array}$$

and contains the formulae $like(j, s)$ and $(\neg\exists x(w(x) \wedge wrh(x) \wedge unique(x) \wedge like(j, x)))$. The latter is universally quantified and can therefore be developed into four branches $\neg w(z)$, $\neg wrh(z)$, $\neg unique(z)$, and $\neg like(j, z)$. The first three branches can be closed using the hypotheses on Sarah and the last one with the first formula, all by binding the new variable z to s. The second branch has the form

$$\begin{array}{c} \neg like(j, s) \\ \exists x(\ldots \wedge unique(x) \wedge like(j, x)) \\ unique(c) \\ like(j, c)) \\ \vdots \\ c = s \\ like(j, s) \\ * \, [] \end{array}$$

and consists of the formulae $\neg like(j,s)$ and $\exists x(w(x) \wedge wrh(x) \wedge unique(x) \wedge like(j,x))$, which is developed into the single branch containing the conjuncts $w(c)$, $wrh(c)$, $unique(c)$, and $like(j,c))$, where c is a Skolem constant for x. Here an expansion of the definition of uniqueness

$$unique(x) \Leftrightarrow \forall z(w(z) \wedge wrh(z) \leftrightarrow x = z)$$

closes the branch (if Sarah and c are unique, then $s = c$).

By now, it should be clear that our treatment will also encounter no particular problem in dealing with examples such as (5.4) and (5.5) below. The first example relies on the world-knowledge that marrying is a symmetric relation (both partners have to say "yes I do"), whereas the second relies on the fact that getting wounded is synonymous to being hurt by someone/thing. Once these equivalences are taken into account, the HOUE analysis of corrections will correctly predict that these examples are well–formed.

(5.4) a. *A: Jon married Sarah*
b. *B: No, Sarah married PETER*

(5.5) a. *A: Sarah hurt Paul.*
b. *B: No, PETER was wounded.*

We have seen that a deaccented anaphor must either have a semantic representation which syntactically unifies with that of its antecedent, or be semantically equivalent to this antecedent. To show that this is a necessary condition, we need to provide some ill–formed examples in which neither condition holds. Such examples are given when the correction contains a destressed pronoun whose source parallel element is either an indefinite (5.6) or a quantifier (5.7).

(5.6) a. *Jon eats an*$_1$ *apple.*
b. $*$ *No, PETER eats it*$_1$.

(5.7) a. *Jon kissed most*$_1$ *women at the party yesterday.*
b. $*$ *No, PETER kissed them*$_1$.

In both cases, the semantic representation of the pronoun in the correction fails to syntactically unify with the semantic representation of its antecedent. Neither can it be proved that *it* and *them* are semantically equivalent to *an apple* and *most women at the party* respectively. Therefore, unification fails correctly ruling out (5.6) and (5.7). The logical reason for this e.g. in (5.6), is that while the second equation $X(p) = eat(p,Y)$ can be solved to $X = \lambda X.eat(X,Y)$ yielding the negated $\neg(eat(j,Y) \Leftrightarrow \exists x(ap(X) \wedge eat(j,X))$, this cannot be refuted. Example (5.6) is in fact ambiguous between a specific reading of the indefinite *an apple* and a non-specific one. In the first case, the indefinite denote uniquely so that *it* in (5.6.nb) refers to this unique apple. Since it is denotationally equivalent with its antecedent, HOUE will succeed. In the second case, there is no unique apple salient in the context, hence *it* and *an apple* cannot be denotationally equivalent. Therefore HOUE fails. The above discussion focuses on this second possibility.

DeMorgan
$\mathcal{R}.\mathbf{C} \vee \mathcal{E}$ *(generalized) clause*
generalized clause
clause!generalized
clause
clause part
unification constraint
$\Box$*empty clause*

Chapter 6

$\mathcal{HR}$: Resolution for $\mathcal{HOL}$

In this chapter we present a variant of Huet's "Constrained Resolution" calculus [Hue72]. Since resolution calculi operate on formulae in clause normal form, we will begin with a discussion of an inference system $\mathcal{RC}$ that transforms arbitrary formulae into clause normal forms, conserving satisfiability.

6.1 Reduction to Clause Normal Form

One of the most prominent features of resolution calculi is that they manipulate formulae in clause (conjunctive) normal form. The conjunctive normal form is a prenex normal form, where all existential quantifications have been eliminated (using techniques shown in chapter 4) and where the matrix has been transformed by DeMorgan laws such that the matrix is a conjunction of disjunctions and such that negations have minimal scope. This normal form is traditionally written in clause form where the quantifier prefix is dropped, and the matrix is written as a set of clauses, which are in turn sets of literals. This set notation emphasizes the commutativity, associativity, and idempotence of conjunction and disjunction.

In $\mathcal{HR}$ we take clauses to be triples: disjunctions of literals together with unification problems and a a variable condition.

Definition 6.1.1 (Clause)
Let $\mathcal{R}$ be a variable condition, then we call a pair $\mathcal{C} := \mathcal{R}.\mathbf{C}$ a **generalized clause**, if $\mathbf{C}$ is a disjunction of labeled formulae in $\mathit{wff}_o(\Sigma \cup \mathbf{Dom}(\mathcal{R}))$. We call $\mathcal{C}$ a **clause**, iff $\mathbf{C}$ is a disjunction of literals.

Just as in the case of tableaux, we will partition $\mathbf{C}$ by the type of literals: The disjunction of all proper literals in $\mathbf{C}$ is called the **clause part** of $\mathcal{C}$, while the disjunction of unification constraints in $\mathbf{C}$ is called the **unification constraint** of $\mathcal{D}$.

In the following we will identify clauses that only differ in the ordering of literals, and we will often treat clauses as sets or multisets of literals.

initial
clause!initial,terminal,empty
terminal
empty clause
$\mathcal{RC}$ calculus for reduction into clause normal form

Definition 6.1.2 (Empty Clause)
We call a clause **initial**, iff its unification constraint is empty, and **terminal**, iff if does not contain any proper literals, i.e. $n = 0$. In accordance with the practice from first-order resolution we denote the empty clause (it is initial and terminal) by $\Box$.

Remark 6.1.3 (Unification Constraint)
In clauses, the unification constraint plays a similar role as in the tableau setting (cf. 5.1.3), only that in the resolution setting it lives at the clause level (resolution is a synthetic calculus):

They represent the unification problem that has to be solved in order to obtain an empty clause. We will see that the $\mathcal{HR}(R)$ rule residuates a resolution pair as necessary conditions do the cut.

In the following we will often write $\mathbf{PU}(\mathcal{C})$ for $\mathbf{PU}(\mathcal{R}.\mathcal{E})$, if $\mathcal{C} = \mathcal{R}.\mathbf{C} \vee \mathcal{E}$.

We present the process of transforming a sentence $\mathbf{A}$ into clause normal form as a calculus $\mathcal{RC}$, in order to facilitate the study of the interaction with the resolution calculus $\mathcal{HR}$ defined below.

Definition 6.1.4 (Reduction Rules ($\mathcal{RC}$))
EdNote(27) The objects manipulated by the $\mathcal{RC}$-calculus are generalized clauses.[27]

We use the rules in Figure 6.1 with the convention that $\vee$ is associative (as we have already suggested by leaving out the parentheses) and commutative. We have specified an explicit inference rule for exercising the idempotence of $\vee$ to collapse multiple occurences of literals in clauses in order to obtain tighter control over $\mathcal{RC}$-derivations in the proofs of the lifting lemmata. Furthermore, after each application the formulae in the new clauses are reduced to $\beta\eta$-normal form. We can extend this calculus to act on sets of generalized clauses. Since the notions are equivalent, we will always adopt the notion most convenient for our purposes.

Example 6.1.5 We illustrate the clause normal form transformation on the axiom of mathematical induction (cf. 1.2.16). We start out with the initial generalized clause generated from the axiom by labelling it with $\mathtt{T}$ and decorating it with the empty variable condition.

1.		$\emptyset.(\forall P_{\iota\to o}.(P0 \wedge \forall X_\iota.PX \Rightarrow P(sX)) \Rightarrow .\forall Z_\iota.\mathbb{N}Z \Rightarrow PZ)^{\mathtt{T}}$
2.	$(\mathcal{RC}(\forall)1)$	$\emptyset.(P0 \wedge \forall X_\iota.PX \Rightarrow P(sX)) \Rightarrow .\forall Z_\iota.\mathbb{N}Z \Rightarrow PZ^{\mathtt{T}}$
3.	$(\mathcal{RC}(\vee)2)$	$\emptyset.(P0 \wedge \forall X_\iota.PX \Rightarrow P(sX))^{\mathtt{F}} \vee (\forall Z_\iota.\mathbb{N}Z \Rightarrow PZ)^{\mathtt{T}}$
4.	$(\mathcal{RC}(\vee), \mathcal{RC}(\forall)3)$	$\emptyset.P0^{\mathtt{F}} \vee (\forall X_\iota.PX \Rightarrow P(sX))^{\mathtt{F}} \vee (\mathbb{N}Z \Rightarrow PZ)^{\mathtt{T}}$
5.	$(\mathcal{RC}(\exists), \mathcal{RC}(\vee)4)$	$\mathcal{R}.P0^{\mathtt{F}} \vee (Pw \Rightarrow P(sw))^{\mathtt{F}} \vee (\mathbb{N}Z)^{\mathtt{F}} \vee PZ^{\mathtt{T}}$
6.	$(\mathcal{RC}(\wedge l)5)$	$\mathcal{R}.P0^{\mathtt{F}} \vee Pw^{\mathtt{T}} \vee \mathbb{N}Z^{\mathtt{F}} \vee PZ^{\mathtt{T}}$
7.	$(\mathcal{RC}(\wedge r)5)$	$\mathcal{R}.P0^{\mathtt{F}} \vee P(sw)^{\mathtt{F}} \vee \mathbb{N}Z^{\mathtt{F}} \vee PZ^{\mathtt{T}}$

[27] EdNote: convert the whole chapter/book to a language, where $\vee$ is primitive

CNF(Φ)*the clause normal form of a set Φ of sentences*
clause normal form
normal form!clause

$$\frac{\mathcal{R}.\mathbf{C} \vee (\neg\mathbf{A})^{\mathsf{T}}}{\mathcal{R}.\mathbf{C} \vee \mathbf{A}^{\mathsf{F}}}\,\mathcal{RC}(\neg^{\mathsf{T}}) \qquad \frac{\mathcal{R}.\mathbf{C} \vee (\neg\mathbf{A})^{\mathsf{F}}}{\mathcal{R}.\mathbf{C} \vee \mathbf{A}^{\mathsf{T}}}\,\mathcal{RC}(\neg^{\mathsf{F}})$$

$$\frac{\mathcal{R}.\mathbf{C} \vee (\mathbf{A} \wedge \mathbf{B})^{\mathsf{T}}}{\mathcal{R}.\mathbf{C} \vee \mathbf{A}^{\mathsf{T}}}\,\mathcal{RC}(\wedge l) \qquad \frac{\mathcal{R}.\mathbf{C} \vee (\mathbf{A} \wedge \mathbf{B})^{\mathsf{T}}}{\mathcal{R}.\mathbf{C} \vee \mathbf{B}^{\mathsf{T}}}\,\mathcal{RC}(\wedge r)$$

$$\frac{\mathcal{R}.\mathbf{C} \vee (\mathbf{A} \wedge \mathbf{B})^{\mathsf{F}}}{\mathcal{R}.\mathbf{C} \vee \mathbf{A}^{\mathsf{F}} \vee \mathbf{B}^{\mathsf{F}}}\,\mathcal{RC}(\vee) \qquad \frac{\mathcal{R}.\mathbf{C} \vee (\Pi^{\alpha}\mathbf{A})^{\mathsf{T}}}{\mathcal{R}.\mathbf{C} \vee \mathbf{A}X^{\mathsf{T}}}\,\mathcal{RC}(\forall)$$

$$\frac{\mathcal{R}.\mathbf{C} \vee (\Pi^{\alpha}\mathbf{A})^{\mathsf{F}} \quad w_{\alpha} \text{ new}}{\mathcal{R} * \{w_{\alpha} \mapsto \mathbf{Free}(\mathbf{A}) \cup \mathcal{R}(\mathcal{W}(\mathbf{A}))\}.\mathbf{C} \vee (\mathbf{A}w_{\alpha})^{\mathsf{F}}}\,\mathcal{RC}(\exists)$$

$$\frac{\mathcal{R}.\mathbf{M}^{\alpha} \vee \mathbf{M}^{\alpha} \vee \mathbf{C}}{\mathcal{R}.\mathbf{M}^{\alpha} \vee \mathbf{C}}\,\mathcal{RC}(coll)$$

Figure 6.1: Reduction to Clause Normal Form

Here $\mathcal{R} = \{w_\iota \mapsto \{P\}\}$. Note that the last two formulae are clauses, since the disjuncts are literals. Thus they constitute the clause normal from of the induction axiom.

Lemma 6.1.6 *The reduction relation induced by $\mathcal{RC}$ on sets of generalized clauses is confluent, terminating, and the $\mathcal{RC}$-normal forms are clauses. Furthermore, all maximal $\mathcal{RC}$-derivations without $\mathcal{RC}(coll)$ are of equal length.*

Proof: For the confluence note that the rules of $\mathcal{RC}$ act only on one labeled proposition in the clause without changing the others, and applicability of the rules is determined by the head symbol of the chosen proposition.

By a simple induction over the number of logical constants that occur at top level in a generalized clause we observe that the $\mathcal{RC}$-rules can only be applied finitely often to a finite set of generalized clauses, so the reduction relation is terminating. Since every rule eliminates exactly one logical symbol, the length is independent of the derivation. □

Definition 6.1.7 (Clause Normal Form)
Let $\mathcal{C}$ be a generalized clause, then we call the set $\mathbf{CNF}(\mathcal{C})$ of clauses that are derivable from $\mathcal{C}$ in $\mathcal{RC}$ the **clause normal form** of $\mathcal{C}$. If $\mathbf{A}$ is a proposition, then we call the set $\mathbf{CNF}(\emptyset.\mathbf{A}^{\mathsf{F}})$ the clause normal form of $\mathbf{A}$, and denote it with $\mathbf{CNF}(\mathbf{A})$. If $\Phi = \{\mathbf{A}_1, \ldots, \mathbf{A}_n\}$ is a set of sentences, then we call the set $\mathbf{CNF}(\Phi) := \bigcup_{i \leq n} \mathbf{CNF}(\mathbf{A}_i)$ the clause normal form of Φ.

Remark 6.1.8 Note that only top level occurrences of propositional subformulae are considered in clause normal form. In general there can be *buried* *satisfied* *valid*

"buried" propositional subformulae in general form in clauses. For instance, if $\mathbf{A} := \forall X_{o\to o}\mathbf{.}\forall Y_o\mathbf{.}X(\neg\neg Y) \vee \neg(XY)$, then clearly $\vdash_\Sigma \mathbf{A}\colon o$ and

$$\mathbf{CNF}(\mathbf{A}) = \emptyset\mathbf{.}(X\mathbf{.}\neg\neg Y)^{\mathsf{T}} \vee \mathbf{.}(XY)^{\mathsf{F}}$$

Before we can show correctness of the clause form transformations, we have to take a look at the concept of satisfiability for clauses (which contain Skolem functions). This relies on the same ideas as the tableau satisfiability from Definition 5.2.1. The Skolem functions are interpreted in a Skolem extension of a given Model, and the set of denotations of the Skolem functions form a $\mathcal{R}_\Gamma$-correspondence.

Definition 6.1.9 (Validity for Clauses)
Let $\mathcal{M}^* = (\mathcal{D}, @, \mathcal{I}, \upsilon, \mathcal{F})$ be a Σ^*-model, $\mathcal{C}$ be a generalized clause, and φ a Γ-assignment into $\mathcal{M}$, then we say that a labeled proposition $\mathbf{M}^\alpha$ in $\mathcal{C}$ is **satisfied** by φ in $\mathcal{M}^*$, iff $\upsilon(\mathcal{I}^*_\varphi(\mathbf{M})) = \alpha$. Analogously for a pair $\mathbf{A} \neq^? \mathbf{B}$ in $\mathcal{C}$, iff $\mathcal{I}^*_\varphi(\mathbf{A}) \neq \mathcal{I}^*_\varphi(\mathbf{B})$. We say that $\mathcal{C}$ is **valid** in $\mathcal{M}$ iff for each Γ-assignment φ, there is a labeled proposition $\mathbf{M} \in \mathcal{C}$ that is satisfied by φ in $\mathcal{M}$.

General Assumption 6.1.10 We consider free variables in clause as universally quantified and use α-conversion for clauses, implicitly renaming them whenever variable disjointness is required.

Remark 6.1.11 For generalized clauses of the form $\emptyset\mathbf{.A}^{\mathsf{T}}$, the notion of validity for clauses coincides with the classical notion, as defined in 2.1.1. Indeed if the variable condition is empty, and the variable context does not contain negative variables, the variable correspondence must be empty too. Since $\mathbf{A}$ is a sentence, its validity is independent of the assignment considered.

Theorem 6.1.12 *Let $\mathcal{C} \vdash_{\mathcal{RC}} \mathcal{D}$ and $\mathcal{M}$ be a Σ-model, then $\mathcal{M} \models \mathcal{C}$, iff $\mathcal{M} \models \mathcal{D}$.*

Proof: The result can be obtained by exactly the same arguments as in the proof of Theorem 5.2.2 □

If we instantiate this result with maximal $\mathcal{RC}$-derivations, we obtain the following clause normal form theorem.

Theorem 6.1.13 (Clause Normal Form Theorem)
Let Φ be a set of sentences, and let $\mathcal{M}$ be a Σ-model, then $\mathcal{M} \models \Phi$, iff $\mathcal{M} \models \mathbf{CNF}(\Phi)$.

Proof: For any sentence $\mathbf{A} \in \Phi$ we have $\mathcal{M} \models \mathbf{A}$, iff $\mathcal{M} \models \emptyset\mathbf{.A}^{\mathsf{T}}$ (cf. 6.1.11) and by 6.1.12 $\mathcal{M} \models \mathcal{C}$ for any clause $\mathcal{C} \in \mathbf{CNF}(\mathbf{A})$. We obtain the assertion by extending this argument to the set Φ. □

Remark 6.1.14 The empty clause □ is unsatisfiable with respect to Σ-models.

6.2 The Resolution Calculus $\mathcal{HR}$

$\mathcal{HR}$*higher-order resolution for* $\mathcal{HOL}$

Now we turn to the actual resolution calculus $\mathcal{HR}$. The previous results set the stage by giving a semantic justification of a resolution calculus that proves sentences $\mathbf{A}$ by converting $\emptyset.\mathbf{A}^{\mathsf{F}}$ to clause normal form and then by deriving the empty clause $\Box$ from that.

In contrast to Huet's calculus we allow pre-unification transformations to be applied to clauses during the resolution process. This generalization allows us to investigate more realistic strategies than in Huet's calculus, which uses the "lazy unification" strategy, that only allows unification to happen after a terminal clause has been derived.

Definition 6.2.1 (Higher-Order Resolution ($\mathcal{HR}$))
The calculus $\mathcal{HR}$ is a variant of Huet's resolution calculus from [Hue72], and has the rules of inference given in Figure 6.2: the first two operate on the

$$\frac{\mathcal{R}.\mathbf{N}^\alpha \lor \mathbf{C} \quad \mathcal{R}'.\mathbf{M}^\beta \lor \mathbf{D} \quad \alpha \neq \beta}{\mathcal{R} \cup \mathcal{R}'.\mathbf{C} \lor \mathbf{D} \lor \mathbf{M} \neq^? \mathbf{N}}\ \mathcal{HR}(R)$$

$$\frac{\mathcal{R}.\mathbf{M}^\alpha \lor \mathbf{N}^\alpha \lor \mathbf{C}}{\mathcal{R}.\mathbf{M}^\alpha \lor \mathbf{C} \lor \mathbf{M} \neq^? \mathbf{N}}\ \mathcal{HR}(F)$$

$$\frac{\mathcal{R}.F\overline{\mathbf{U}^k}^\alpha \lor \mathbf{C} \quad \mathbf{P} \in \mathcal{A}_\alpha^{\{\land,\neg\}\cup\{\Pi^\beta \mid \beta\in\mathcal{T}\}}(\Sigma)}{\mathcal{R}.F\overline{\mathbf{U}^k}^\alpha \lor \mathbf{C} \lor F \neq^? \mathbf{P}}\ \mathcal{HR}(P)$$

$$\frac{\mathcal{R}.\mathbf{C} \lor X \neq^? \mathbf{B} \quad X \neq^? \mathbf{B}\ \mathcal{R}\text{−solved}}{\mathcal{C} \in \mathbf{CNF}(\mathcal{R}(\mathbf{B}/X).[\mathbf{B}/X]\mathbf{C})}\ \mathcal{HR}(E)$$

Figure 6.2: Higher-Order Resolution ($\mathcal{HR}$)

clause parts. For the $\mathcal{HR}(R)$ rule we assume that the clauses are variable-disjoint. Note that this assumption does not result in a loss of generality, since we can always rename the variables in a clause, since they are implicitly universally quantified (cf. 6.1.10). For manipulating the unification constraints $\mathcal{HR}$ utilizes the $\Sigma\mathcal{PT}$ rules $\Sigma\mathcal{PT}(\mathit{flex}-\mathit{rig})$ and $\Sigma\mathcal{PT}(\mathit{guess})$ (cf. 9.2.7) by extending them to clauses in the obvious way. The rule $\mathcal{HR}(P)$ generates instantiations for flexible literal, binding them to general bindings whose head if some logical constant.

$\mathcal{HR}$ has one further inference rule $\mathcal{HR}(E)$ which propagates partial solutions from the constraints to the clause part, and thus help detect clashes early. Since the instantiation may well change the propositional structure of the clause by instantiating a predicate variable, we have to renormalize the clause on the fly.

$\Phi \vdash_{\mathcal{HR}} \Box$ $\mathcal{HR}$-refutation of Φ refutable!$\mathcal{HR}$ refutation!$\mathcal{HR}$

Definition 6.2.2 We call a set Φ of well-formed sentences $\mathcal{HR}$-**refutable**, iff $\Box$ is derivable from the set of clauses $\mathbf{CNF}(\Phi)$. A $\mathcal{HR}$-derivation $\mathcal{R}$ of an empty clause $\Box$ from a set $\mathcal{C}$ of clauses is called a $\mathcal{HR}$-**refutation** of $\mathcal{C}$. By a slight abuse of notation we call a sentence $\mathbf{A}$ $\mathcal{HR}$-refutable, iff $\mathbf{CNF}(\mathbf{A}^{\mathtt{F}})$ is $\mathcal{HR}$-refutable.

Example 6.2.3 Let $\Sigma := \{[c: o \to o], [b: o]\}$, $\mathbf{A} := (cb)$, and $\mathbf{B} := c(\neg\neg b)$, then we can convince ourselves that there is no $\mathcal{HR}$-refutation of $\mathbf{C} := \neg\mathbf{A} \wedge \mathbf{B}$: The clause normal form of $\mathbf{C}$ is $\emptyset.\{(cb)^{\mathtt{F}}, \emptyset.c(\neg\neg b)^{\mathtt{T}}\}$. Clearly the only rule that applies to $\mathbf{CNF}(\mathbf{C})$ is $\mathcal{HR}(R)$ yielding $\emptyset.(cb) \neq^? c(\neg\neg b)$, which simplifies to the unsolvable constraint $\emptyset.b \neq^? (\neg\neg b)$.

We conclude our exposition of $\mathcal{HR}$ with two examples.

Example 6.2.4 (The Successor has no Fixed Point)
The clause normal form of the Peano axioms (cf. 1.2.16) is the following:

1. $\emptyset.\mathbb{N}0^{\mathtt{T}}$
2. $\emptyset.\mathbb{N}X^{\mathtt{F}} \vee \mathbb{N}(sX)^{\mathtt{T}}$
3. $\emptyset.\mathbb{N}X^{\mathtt{F}} \vee sX = 0^{\mathtt{F}}$
4. $\emptyset.sX = sY^{\mathtt{F}} \vee X = Y^{\mathtt{T}}$
5. $\mathcal{R}.P0^{\mathtt{F}} \vee Pw^{\mathtt{T}} \vee \mathbb{N}Z^{\mathtt{F}} \vee PZ^{\mathtt{T}}$
6. $\mathcal{R}.P0^{\mathtt{F}} \vee P(sw)^{\mathtt{F}} \vee \mathbb{N}Z^{\mathtt{F}} \vee PZ^{\mathtt{T}}$

where $\mathcal{R} = \{w_\iota \mapsto \{P\}\}$ as we have seen in Example 6.1.5. Then we have the clauses coming from the theorem $\forall X_\iota.\mathbb{N}X \Rightarrow \neg(sX) = X^{\mathtt{F}}$

7. $\mathcal{R}'.\mathbb{N}v^{\mathtt{T}}$
8. $\mathcal{R}'.sv = v^{\mathtt{T}}$

Where $\mathcal{R}' = \{v \mapsto \emptyset\}$. Now we have the following $\mathcal{HR}$-derivation:

9.	$(\mathcal{HR}(R)5,7)$	$\mathcal{R} \cup \mathcal{R}'.P0^{\mathtt{F}} \vee Pw^{\mathtt{T}} \vee PZ^{\mathtt{T}} \vee Z \neq^? v$
10.	$(\mathcal{HR}(E)9)$	$\mathcal{R} \cup \mathcal{R}'.P0^{\mathtt{F}} \vee Pw^{\mathtt{T}} \vee Pv^{\mathtt{T}}$
11.	$(\mathcal{HR}(P)10)$	$\mathcal{R} \cup \mathcal{R}'.P0^{\mathtt{F}} \vee Pw^{\mathtt{T}} \vee Pv^{\mathtt{T}} \vee P \neq^? \lambda Z.\neg(Q_{\iota\to o}Z)$
12.	$(\mathcal{HR}(E)11)$	$\mathcal{R}^1.Q0^{\mathtt{T}} \vee Qw^{\mathtt{F}} \vee Qv^{\mathtt{F}}$
13.	$(\mathcal{HR}(R)12,8)$	$\mathcal{R}^1.Q0^{\mathtt{T}} \vee Qw^{\mathtt{F}} \vee Qv \neq^? sv = v$
14.	$(\mathcal{HR}(E)13)$	$\mathcal{R}^2.s0 = 0^{\mathtt{T}} \vee sw = w^{\mathtt{F}}$
15.	$(\mathcal{HR}(R)1,3)$	$\emptyset.s0 = 0^{\mathtt{F}}$
16.	$(\mathcal{HR}(R)14,15)$	$\mathcal{R}^2.sw = w^{\mathtt{F}}$

Where $\mathcal{R}^1 = \{w \mapsto \{Q\}, v \mapsto \emptyset\}$ and $\mathcal{R}^2 = \{w \mapsto \emptyset, v \mapsto \emptyset\}$. Note that we used $Q \neq^? \lambda X.sZ = Z$ in step 14.

Analogously, we obtain from 6. instead from 5:

17.	$(\mathcal{HR}(R)14,15)$	$\mathcal{R}^2.s(sw) = sw^{\mathtt{T}}$
18.	$(\mathcal{HR}(R)4,17)$	$\mathcal{R}^2.X = Y^{\mathtt{T}} \vee sX = sY \neq^? s(sw) = sw$
19.	$(\mathcal{SIM}(dec)18)$	$\mathcal{R}^2.X = Y^{\mathtt{T}} \vee X \neq^? sw \vee Y \neq^? w$
20.	$(\mathcal{HR}(E)19)$	$\mathcal{R}^2.sw = w^{\mathtt{T}}$
21.	$(\mathcal{HR}(R)16,20)$	$\Box$

This example shows that we can in principle use $\mathcal{HR}$ to mechanize induction theorem proving (which is not a big surprise).

splitting rule
rule!splitting
Andrews

Remark 6.2.5 (Splitting)
In his "constrained resolution" calculus Huet uses a set of rules for flexible literals, which he calls **splitting rule**s instead of the rule $\mathcal{HR}(P)$ (which is essentially due to Peter Andrews, cf. [ALCMP84, And89]). We have adapted the splitting rules to the $\mathcal{HR}$ setting in Figure 6.3. They are equivalent to the $\mathcal{HR}(P)$ rule in the presence of $\mathcal{HR}(E)$. In contrast to $\mathcal{HR}(P)$, which only binds the head variable of the flexible literal they also perform some clause normalization. Note that Huet does not need a primitive binding for projection for flexible variables, since this can be obtained in the subsequent unification.

We feel that Andrew's primitive substitution rule together with clause normalization on the fly gives a better separation of causes and generalizes better to other settings like extensional higher-order resolution, which we will discuss in section 6.6. Note that Huet can only avoid on the fly clause normalization, since he defers all unification until a terminal clause has been found.

$$\frac{\mathcal{R}.\mathbf{L}^{\mathtt{T}} \vee \mathbf{C}}{\mathcal{R}.P_o^{\mathtt{T}} \vee Q_o^{\mathtt{T}} \vee \mathbf{C} \vee \mathbf{L} \neq^{?} (P \vee Q)} \qquad \frac{\mathcal{R}.\mathbf{L}^{\mathtt{F}} \vee \mathbf{C}}{\mathcal{R}.P_o^{\mathtt{T}} \vee \mathbf{C} \vee \mathbf{L} \neq^{?} (P \vee Q)}$$

$$\frac{\mathcal{R}.\mathbf{L}^{\mathtt{T}} \vee \mathbf{C}}{\mathcal{R}.P_o^{\mathtt{T}} \vee \mathbf{C} \vee \mathbf{L} \neq^{?} \neg P} \qquad \frac{\mathcal{R}.\mathbf{L}^{\mathtt{F}} \vee \mathbf{C}}{\mathcal{R}.P_o^{\mathtt{F}} \vee \mathbf{C} \vee \mathbf{L} \neq^{?} \neg P}$$

$$\frac{\mathcal{R}.\mathbf{L}^{\mathtt{T}} \vee \mathbf{C}}{\mathcal{R}.R_{\alpha\to o} Z_\alpha^{\mathtt{T}} \vee \mathbf{C} \vee \mathbf{L} \neq^{?} \Pi^\alpha R} \qquad \frac{\mathcal{R}.\mathbf{L}^{\mathtt{T}} \vee \mathbf{C}}{\mathcal{R}'.R_{\alpha\to o} w^{\mathtt{T}} \vee \mathbf{C} \vee \mathbf{L} \neq^{?} \Pi^\alpha R}$$

$P_o, Q_o, R_{\alpha\to o}$ new, $\mathcal{R}' = \mathcal{R} * \{w_\alpha \mapsto \mathbf{Free}(\mathbf{A}) \cup \mathcal{R}(\mathcal{W}(\mathbf{A}))\}$

Figure 6.3: Huet's Splitting rules

Example 6.2.6 (Cantor's theorem)
To show the differences to the tableau setting (cf. 5.1.5), we will now prove Cantor's theorem (see 1.2.18) in $\mathcal{HR}$. We will use primed rules like $\mathcal{HR}(R)'$ as abbreviation for the rule application with subsequent unification.

1.	*given*	$\mathcal{R}.(fj = G)^{\mathtt{T}}$
2.	*given*	$\emptyset.H = K^{\mathtt{F}} \vee HN = KN^{\mathtt{T}}$
3.	*given*	$\emptyset.X = sX^{\mathtt{F}}$
4.	$(\mathcal{HR}(R)'1, 2)$	$\mathcal{R}.fjN = GN^{\mathtt{T}}$
5.	$(\mathcal{HR}(R)4, 3)$	$\mathcal{R}.fjN = GN \neq^{?} X = sX$

from here on, unification goes just like in the tableau proof in 5.1.5.

6.3 Soundness

Soundness of $\mathcal{HR}$ is relatively simple, since we have treated the interesting cases on section 6.1, and the remaining rules do not affect variable conditions.

Lemma 6.3.1 *Let Φ be a set of clauses and $\Phi \vdash_{\mathcal{HR}} \mathcal{D}$, then for any $\Sigma\mathcal{R}$-model $\mathcal{M}$ we have $\mathcal{M} \models \mathcal{D}$, if $\mathcal{M} \models \Phi$.*

Proof: Let $\mathcal{D}: \Phi \vdash_{\mathcal{HR}} \mathcal{D}$, we prove the assertion by induction over the structure of $\mathcal{D}$. If $\mathcal{D}$ is the empty $\mathcal{HR}$-derivation, then the assertion is trivial. If $\mathcal{D}$ is obtained from Φ by $\mathcal{HR}(R)$, then we have the following situation:

$$\cfrac{\cfrac{\mathcal{D} \quad \mathcal{D}' \quad \mathcal{R}'.\mathbf{M}^\beta \vee \mathbf{D}}{\mathcal{R}.\mathbf{N}^\alpha \vee \mathbf{C} \quad\quad \alpha \neq \beta}}{\mathcal{R} \cup \mathcal{R}'.\mathbf{C} \vee \mathbf{D} \vee \mathbf{M} \neq^? \mathbf{N}}\ \mathcal{HR}(R)$$

By inductive hypothesis, there is a $\mathcal{R}$-correspondence $\mathcal{F}$ and a $\mathcal{R}'$-correspondence $\mathcal{F}'$, such that for all assignments φ and φ' we have $\mathcal{I}^{\mathcal{F}}_{\varphi}(\mathbf{N}) = \alpha$ or $\mathcal{I}^{\mathcal{F}}_{\varphi}(\mathbf{L}) = \gamma$ for some $\mathbf{L}^\gamma \in \mathbf{C}$ and $\mathcal{I}^{\mathcal{F}'}_{\varphi'}(\mathbf{M}) = \beta$ or $\mathcal{I}^{\mathcal{F}'}_{\varphi'}(\mathbf{K}) = \delta$ for some $\mathbf{K}^\delta \in \mathbf{D}$.

Clearly, $\mathcal{F} \cup \mathcal{F}'$ is a $\mathcal{R} \cup \mathcal{R}'$-assignment, since the clauses are variable-disjoint. We now convince ourselves that $\mathcal{M} \models^{\mathcal{F}\cup\mathcal{F}'}_{\varphi\cup\varphi'} \mathcal{R} \cup \mathcal{R}'.\mathbf{C} \vee \mathbf{D} \vee \mathbf{M} \neq^? \mathbf{N}$. We have two cases: if $\mathcal{I}^{\mathcal{F}}_{\varphi}(\mathbf{L}) = \gamma$ or $\mathcal{I}^{\mathcal{F}'}_{\varphi'}(\mathbf{K}) = \delta$, then the claim is trivial, since extending the $\mathcal{R}$-correspondence $\mathcal{F}$ by $\mathcal{F}'$ will not change the value of $\mathbf{L}$ ($\mathbf{Dom}(\mathcal{R}')$ and $\mathcal{W}(\mathbf{L})$ are disjoint). In the other case we have

$$\mathcal{I}^{\mathcal{F}\cup\mathcal{F}'}_{\varphi\cup\varphi'}(\mathbf{M}) = \mathcal{I}^{\mathcal{F}}_{\varphi}(\mathbf{M}) = \alpha \neq \beta = \mathcal{I}^{\mathcal{F}'}_{\varphi'}(\mathbf{N}) = \mathcal{I}^{\mathcal{F}\cup\mathcal{F}'}_{\varphi\cup\varphi'}(\mathbf{N})$$

and in particular, $\mathcal{I}^{\mathcal{F}\cup\mathcal{F}'}_{\varphi\cup\varphi'}(\mathbf{M}) \neq \mathcal{I}^{\mathcal{F}\cup\mathcal{F}'}_{\varphi\cup\varphi'}(\mathbf{N})$.

Now the only remaining non-trivial case is that of $\mathcal{HR}(E)$, since the case of $\mathcal{HR}(F)$ is analogous to $\mathcal{HR}(R)$, and the rules $\mathcal{HR}(P)$, $\mathcal{\Sigma PT}(\mathit{flex}-\mathit{rig})$, and $\mathcal{\Sigma PT}(\mathit{guess})$ only add pairs, which only weaken disjunctions.

The $\mathcal{HR}(E)$ inference rule can be divided into two parts, first instantiating a clause $\mathcal{C} = \mathcal{R}.\mathbf{C} \vee X \neq^? \mathbf{B}$ to a generalized clause $\mathcal{C}' := \mathcal{R}(\mathbf{B}/X).[\mathbf{B}/X]\mathbf{C}$, and then reducing it to clause normal form. Obviously any $\Sigma\mathcal{R}$-model of $\mathcal{C}$ can be extended to a $\Sigma\mathcal{R}(\mathbf{C}/X)$-model for $\mathcal{R}(\mathbf{B}/X).[\mathbf{B}/X]\mathbf{C}$ by the substitution value theorem 4.2.6: the truth values of the formulae in the tableau are not changed by the instantiation as we have $\mathcal{I}^{\mathcal{F}}_{\varphi,[\mathcal{I}^{\mathcal{F}}_{\varphi}(\mathbf{C})/X]}(\mathbf{A}) = \mathcal{I}^{\mathcal{F}(\mathbf{A}/X)}_{\varphi}([\mathbf{B}/X]\mathbf{A})$.

Thus soundness of this case is a consequence of the inductive hypothesis and the clause normal form theorem 6.1.13. $\square$

clause isomorphism
isomorphism!clause (set)
clause set isomorphism
compatible

Theorem 6.3.2 *The $\mathcal{HR}$ calculus is sound, i.e. if* $\mathbf{A}$ *is a sentence such that* $\mathbf{A}$ *is $\mathcal{HR}$-refutable, then* $\mathbf{A}$ *is unsatisfiable with respect to Σ-models.*

Proof: Let $\Phi = \mathbf{CNF}(\mathbf{A}^{\mathsf{F}})$. If $\Phi \vdash_{\mathcal{HR}} \Box$, then Φ is unsatisfiable with respect to Σ-models by 6.3.1 and 6.1.14. Now the clause normal form theorem 6.1.13 gives the desired result. $\Box$

6.4 Lifting Properties for $\mathcal{HR}$

Note that since $\mathcal{HR}$ differs from Huet's constrained resolution in that we allow interleaving higher-order unification and resolution, the lifting lemmata become considerably more complex and take up most of the work in the completeness proof for $\mathcal{HR}$.

The proof of the lifting theorem constructs a lifted $\mathcal{HR}$-refutation of a proposition $\mathbf{A}$ from one for $\theta(\mathbf{A})$. Just as in the tableau calculus discussed in the last chapter we will need to maintain a tight correspondence $\omega\colon \mathcal{C}_\theta \longrightarrow \mathcal{C}$ (which we formalize in the notion of a clause set isomorphism) between the clauses in the derivations that respects labels and is compatible with θ, i.e. for any Literal $\mathbf{L}^\alpha$ in $\mathcal{C}$ we have $\omega(\theta(\mathbf{L})) = \mathbf{L}$.

Definition 6.4.1 (Clause Set Isomorphism)
Let $\mathcal{C}$ and $\mathcal{C}'$ be generalized clauses such that Φ and Φ' are the respective sets of proper labeled formulae in $\mathcal{C}$ and $\mathcal{C}'$, then we call a bijection $\omega\colon \Phi \longrightarrow \Phi'$ a **clause isomorphism**, iff $\omega(\mathbf{M}^\alpha)$ is of the form $\mathbf{N}^\alpha$, i.e. the intended truth values match.

Let Φ and Ψ be sets of clauses, then a bijection $\omega\colon \Phi \longrightarrow \Psi$ together with a family of mappings $\omega_{\mathcal{C}}\colon \mathcal{C} \longrightarrow \omega(\mathcal{C})$ for all $\mathcal{C} \in \Phi$ is called a **clause set isomorphism**, iff all $\omega_{\mathcal{C}}$ are clause isomorphisms.

In the following we will write clauses as $\mathcal{C} = \mathcal{R}\mathbf{.C} \vee \mathcal{E}$ with the intended meaning that $\mathbf{C}$ is the clause of $\mathcal{C}$ and $\mathcal{E}$ its unification constraint.

Definition 6.4.2 (θ-Compatible)
Let θ be a $\mathcal{R}$-substitution, $\mathcal{C} = \mathcal{R}\mathbf{.C} \vee \mathcal{F}$ and $\mathcal{C}_\theta = \mathcal{R}_\theta\mathbf{.C}_\theta \vee \mathcal{F}_\theta$ generalized clauses, and $\omega\colon \mathcal{C}_\theta \longrightarrow \mathcal{C}$ a clause isomorphism. We say that ω is **θ-compatible**, iff $\omega(\theta(\mathbf{M})^\alpha) = \mathbf{M}^\alpha$ for all labeled formulae $\mathbf{M}$ in $\mathbf{C}$ and moreover (cf. Definition 5.3.2) $\tau_\theta(\mathbf{PU}(\mathcal{C}_\theta)) = \mathbf{PU}(\mathcal{C})$.

We call a clause set isomorphism ω **θ-compatible**, iff all component clause isomorphisms $\omega_{\mathcal{C}_\theta}$ are. Similarly we can define (θ-compatible) isomorphisms of derivations as isomorphisms of the underlying trees such that corresponding nodes have clause isomorphisms.

We need the following technical lemmata, which allows us to mimic $\mathcal{RC}$-derivations by $\mathcal{HR}$-derivations using $\mathcal{HR}(P)$ inferences and factorization. To

EdNote(28) ease the load of notation we will leave out the variable correspondences from the clauses and unification problems.[28]

Before we can prove the first lifting lemma, we need the following result:

Lemma 6.4.3 (Anti-Lifting for $\mathcal{RC}$)
Let $\mathcal{C} \vdash_{\mathcal{RC}} \mathcal{D}$, and $\omega : \mathcal{C}_\theta \longrightarrow \mathcal{C}$ be a θ-compatible clause isomorphism, then there is a $\mathcal{RC}$-derivation $\mathcal{C}_\theta \vdash_{\mathcal{RC}} \mathcal{D}_\theta$, and a θ-compatible clause isomorphism $\omega' : \mathcal{D}_\theta \longrightarrow \mathcal{D}$.

Proof: By induction on the structure of $\mathcal{R} : \mathcal{C} \vdash_{\mathcal{RC}} \mathcal{D}$. If $\mathcal{R}$ is empty, then the assertion is trivial for $\omega' := \omega$. If $\mathcal{R}$ begins with $\mathcal{RC}(\vee)$, then $\mathcal{C} = \mathcal{R}.(\mathbf{A} \wedge \mathbf{B})^{\mathsf{F}} \vee \mathcal{C}'$, and $\mathcal{R} : \mathcal{R}.(\mathbf{A} \wedge \mathbf{B})^{\mathsf{F}} \vee \mathcal{C}' \vdash \mathcal{R}.\mathbf{A}^{\mathsf{F}} \vee \mathbf{B}^{\mathsf{F}} \vee \mathcal{C}' \vdash \mathcal{D}$. Now since ω is θ-compatible, $\mathcal{C}_\theta = \mathcal{R}_\theta.(\theta(\mathbf{A}) \wedge \theta(\mathbf{B}))^{\mathsf{F}} \vee \mathcal{C}'_\theta$, and we have $\mathcal{C}_\theta \vdash_{\mathcal{RC}(\vee)} \mathcal{R}_\theta.\theta(\mathbf{A})^{\mathsf{F}} \vee \theta(\mathbf{B})^{\mathsf{F}} \vee \mathcal{C}'_\theta$. Let $\widetilde{\omega}$ be the the obvious $\vee$-refinement of ω, then we $\widetilde{\omega}$ is trivially θ-compatible. So we obtain the assertion for the $\mathcal{RC}(\vee)$ case by inductive hypothesis. The other cases are analogous and we have proven the assertion. □

The problem the $\mathcal{RC}$ lifting adds to the lifting techniques discussed in section 5.3 is that the $\mathcal{HR}(E)$ step eagerly reduces to clause normal form. Thus we have to consider the possibility that the general binding $\mathbf{G}$ in the the primitive substitution may be a projection that projects up arbitrarily complex propositional subformulae, which trigger normalization sequences of length greater than one. Thus the statement of the analogon to Lemma 5.3.5 is a little more involved.

Lemma 6.4.4 (Mimicking)
Let $\mathbf{A} \in wff_o(\Sigma)$ and θ a $\mathcal{R}$-substitution, and $\omega : \mathcal{C}_\theta \longrightarrow \mathcal{C}$ a θ-compatible clause isomorphism, where $\mathcal{C}_\theta = \mathcal{R}_\theta.\theta(\mathbf{A}) \vee \mathbf{C}_\theta$ and $\mathcal{C} = \mathcal{R}.\mathbf{A} \vee \mathbf{C}$. If $\theta(\mathbf{A})$ is not a disjunction of literals, then there is a $\mathcal{HR}$-derivation $\mathcal{R}.\mathbf{A} \vee \mathbf{C} \vdash_{\mathcal{HR}} \mathcal{C}' := \mathcal{R}'.\mathbf{A}' \vee \mathbf{C}'$ of positive length, a substitution θ' with $\theta = \theta'[\mathbf{Dom}(\theta)]$, a $\mathcal{RC}$-derivation $\mathcal{C}_\theta \vdash_{\mathcal{RC}} \mathcal{C}_{\theta'}$ and a θ'-compatible clause isomorphism $\omega' : \mathcal{C}_{\theta'} \longrightarrow \mathcal{C}'$.

Proof: Let $\mu(\theta, \mathbf{A})$ be the multiset of term depths in $\theta(\mathbf{Free}(\mathbf{A}))$. We prove the assertion by well-founded induction over the multi-set ordering for natural numbers.

If $\mathbf{head}(\mathbf{A}) \in \{\wedge, \neg, \Pi^\alpha\}$, then $\mathbf{head}(\theta(\mathbf{A})) = \mathbf{head}(\mathbf{A})$ and we can apply the same $\mathcal{RC}$ rule that eliminates the head to both. The obvious refinement ω of ω' gives the assertion. Note that this subsumes the base case where $\mu(\theta, \mathbf{A}) = \emptyset$, since then $\theta(\mathbf{A}) = \mathbf{A}$ and $\mathbf{head}(\mathbf{A}) = \mathbf{head}(\theta(\mathbf{A})) \in \{\wedge, \neg, \Pi^\alpha\}$.

If $\mathbf{head}(\mathbf{A}) \notin \{\wedge, \neg, \Pi^\alpha\}$, then $\mathbf{A}$ must be flexible, i.e. $\mathbf{A} = P_\alpha \overline{\mathbf{U}^n}$ for some variable $P_\alpha \in \mathbf{Dom}(\theta)$ and $h = \mathbf{head}(\theta(P)) \in \{j, \wedge, \neg, \Pi^\alpha\}$. Therefore, by the general binding theorem 3.4.2 there is a partial binding $\mathbf{G}^h_\alpha$,

[28] EdNote: maybe

a substitution ρ such that $\rho(\mathbf{G}_\alpha^h) =_{\beta\eta} \theta(P_\alpha)$. Now let $\theta^h := \theta_{-P} \cup \rho$ and $\mathcal{C}^h := \mathcal{R}(\mathbf{G}_\alpha^h/P)\mathbf{.}[\mathbf{G}_\alpha^h/P]\mathbf{A} \vee [\mathbf{G}_\alpha^h/P]\mathbf{C}$, and ω^h the obvious variant of ω (instantiation does not change the distribution of labeled formulae). To see that ω^h is θ^h-compatible, note that $\tau_\theta(\mathbf{PU}(\mathcal{F})) = \mathbf{PU}(\mathcal{F}_\theta)$, where $\mathcal{F}$ and $\mathcal{F}$ are the unification constraints of $\mathcal{C}$ and $\mathcal{C}_\theta$. So every $\sigma \in \mathbf{PU}(\mathcal{F})$

$$\begin{aligned}
\tau_\theta(\mathbf{PU}(\mathcal{C})) &= \tau_{\theta^h}(\tau_{[\mathbf{G}_\alpha^h/P]}(\mathbf{PU}(\mathcal{C}))) \\
&= \tau_{\theta^h}(\tau_{[\mathbf{G}_\alpha^h/P]}(\mathbf{PU}(\mathcal{R}\mathbf{.}\mathcal{E}))) \\
&= \tau_{\theta^h}(\mathbf{PU}(\mathcal{R}(\mathbf{G}_\alpha^h/P)\mathbf{.}[\mathbf{G}_\alpha^h/P]\mathcal{E})) \\
&= \tau_{\theta^h}(\mathbf{PU}(\mathcal{C}^h))
\end{aligned}$$

As ω is θ-compatible, we have $\tau_\theta(\mathbf{PU}(\mathcal{C})) = \mathbf{PU}(\mathcal{C}_\theta)$, so we have $\tau_{\theta^h}(\mathbf{PU}(\mathcal{C}^h)) = \mathbf{PU}(\mathcal{C}_\theta)$, in other words, the obvious extension ω' of ω to $\mathcal{C}^h$ is θ^h-compatible. Furthermore, $\mu(\theta^h, [\mathbf{G}_\alpha^h/P]\mathbf{A}) < \mu(\theta, \mathbf{A})$, since $\mathbf{dp}(\rho) < \mathbf{dp}(\theta(P))$ and $P \notin \mathbf{Free}([\mathbf{G}_\alpha^h/P]\mathbf{A})$.

So by inductive hypothesis, there is a $\mathcal{HR}$-derivation $\widetilde{\mathcal{D}}{:}\,\mathcal{C}^h \vdash_{\mathcal{HR}} \mathcal{C}' := \mathcal{R}'\mathbf{.A}' \vee \mathbf{C}'$ of positive length and a substitution $\widetilde{\theta}$ with $\theta^h = \widetilde{\theta}[\mathbf{Dom}(\theta^h)]$. Now let $\theta' := \widetilde{\theta} \circ [\mathbf{G}_\alpha^h/P]$ and $\mathcal{D}'$ be $\widetilde{\mathcal{D}}$ prefixed by $\mathcal{HR}(P)$. We have

$$\theta'(P) = \widetilde{\theta}(\mathbf{G}_\alpha^h) = \rho(\mathbf{G}_\alpha^h) = \theta(P)$$

by construction and for $P \neq X \in \mathbf{Dom}(\theta)$ we have $\theta'(X) = \widetilde{\theta}(X) = \theta(X)$, since $\widetilde{\theta} = \theta^h = \theta \cup \rho = \theta[\mathbf{Dom}(\theta)]$. So we have established $\theta' = \theta[\mathbf{Dom}(\theta)]$.

The existence of $\mathcal{C}_\theta$ and ω' is a simple consequence of Lemma 6.4.3.[29] $\Box$ EdNote(29)

Lemma 6.4.5 (Lifting Lemma for $\mathcal{RC}$)
Let $\mathcal{C}, \mathcal{C}_\theta$, and $\widetilde{\mathcal{C}}_\theta$ be generalized clauses and let $\mathcal{C}_\theta \vdash_{\mathcal{RC}} \widetilde{\mathcal{C}}_\theta$ be a maximal $\mathcal{RC}$-derivation. Furthermore, let θ be a $\mathcal{C}$-substitution, and $\omega{:}\,\mathcal{C}_\theta \longrightarrow \mathcal{C}$ a θ-compatible clause isomorphism. Then there exists a $\mathcal{HR}$-derivation $\mathcal{C} \vdash_{\mathcal{HR}} \widetilde{\mathcal{C}}$, a $\widetilde{\mathcal{C}}$-substitution $\widetilde{\theta}$ with $\widetilde{\theta} \equiv \theta[\mathbf{Dom}(\theta)]$, and a $\widetilde{\theta}$-compatible clause isomorphism $\widetilde{\omega}{:}\,\widetilde{\mathcal{C}}_\theta \longrightarrow \widetilde{\mathcal{C}}$, so that we have the following commutative diagram.

$$\begin{array}{ccc}
\mathcal{C}_\theta & \vdash_{\mathcal{RC}} & \widetilde{\mathcal{C}}_\theta \\
\theta \Big\downarrow \omega & & \widetilde{\theta} \Big\downarrow \widetilde{\omega} \\
\mathcal{C} & \vdash_{\mathcal{HR}} & \widetilde{\mathcal{C}}
\end{array}$$

Proof: Let $\mathcal{D}_\theta{:}\,\mathcal{C}_\theta \vdash_{\mathcal{RC}} \mathcal{C}'_\theta \vdash_{\mathcal{RC}(coll)} \widetilde{\mathcal{C}}_\theta$ where the $\mathcal{RC}$-derivation $\widetilde{\mathcal{D}}_\theta{:}\,\mathcal{C}'_\theta \vdash_{\mathcal{RC}} \widetilde{\mathcal{C}}_\theta$ consists entirely of $\mathcal{RC}(coll)$ steps collapsing duplicate literals.

We prove the assertion in two steps: first we construct a $\mathcal{HR}$-derivation $\mathcal{R}'{:}\,\mathcal{C} \vdash_{\mathcal{HR}} \mathcal{C}'$ and then extend this appropriately by $\mathcal{HR}(F)$ steps to account for idempotence. As we have seen in Lemma 6.1.6, the number of nodes

[29] EdNote: check that, do we need the inductive hypothesis for that?

in maximal $\mathcal{RC}$-derivations that do not exercise idempotence of $\vee$ is independent of the concrete $\mathcal{RC}$-derivation. So we construct $\mathcal{R}'$ by induction on the number of nodes in $\mathcal{D}_\theta$. If $\mathcal{D}_\theta$ is empty, then $\mathcal{C}'_\theta = \mathcal{C}_\theta$, so we obtain the assertion with $\mathcal{C}' := \mathcal{C}$, $\theta' := \theta$, and the empty $\mathcal{HR}$-derivation. If $\mathcal{D}_\theta$ is non-empty, $\mathcal{C}_\theta$ is of the form $\mathcal{R}_\theta.\theta(\mathbf{A}) \vee \mathbf{C}_\theta$ and $\mathcal{C} \equiv \mathcal{R}.\mathbf{A} \vee \mathcal{C}$, where $\theta(\mathbf{A})$ is not a disjunction of literals. So we can apply Lemma 6.4.4 and obtain a $\mathcal{HR}$-derication $\mathcal{R}.\mathbf{A} \vee \mathbf{C} \vdash_{\mathcal{HR}} \mathcal{C}'' := \mathcal{R}'.\mathbf{A}'' \vee \mathbf{C}''$ of positive length, a substitution θ'' with $\theta = \theta''[\mathbf{Dom}(\theta)]$, a $\mathcal{RC}$-derivation $\mathcal{C}_\theta \vdash_{\mathcal{RC}} \mathcal{C}_{\theta''}$ and a θ''-compatible clause isomorphism $\omega'' : \mathcal{C}_{\theta''} \longrightarrow \mathcal{C}''$.

Since we do not exercise idempotence of $\vee$ and $\mathcal{RC}$ is confluent, there must be an $\mathcal{RC}$-derivation $\mathcal{D}''_\theta : \mathcal{C}''_\theta \vdash_{\mathcal{RC}} \mathcal{C}'_\theta$ that has fewer nodes than $\mathcal{D}_\theta$. Thus we obtain the assertion with the inductive hypothesis by combining $\mathcal{HR}$-derivations according to the following diagram:

$$\begin{array}{ccccc}
\mathcal{C}_\theta & \vdash_{\mathcal{RC}} & \mathcal{C}''_\theta & \vdash_{\mathcal{RC}} & \mathcal{C}'_\theta \\
\theta \big\downarrow \omega & & \theta'' \big\downarrow \omega'' & & \theta' \big\downarrow \omega' \\
\mathcal{C} & \vdash_{\mathcal{HR}(P)} & \mathcal{C}'' & \vdash^{IH}_{\mathcal{HR}} & \mathcal{C}'
\end{array}$$

This completes our first goal.

Now $\widetilde{\mathcal{C}_\theta}$ is obtained from $\mathcal{C}'_\theta$ by collapsing duplicate literals, which we simulate by $\mathcal{HR}(F)$ steps on the more general side. Let $\widetilde{\theta} := \theta'$, we proceed by constructing a $\widetilde{\theta}$-compatible clause isomorphism $\widetilde{\omega}$ from ω' by induction on the structure $\widetilde{\mathcal{D}_\theta} : \mathcal{C}'_\theta \vdash_{\mathcal{RC}} \widetilde{\mathcal{C}_\theta}$. If $\widetilde{\mathcal{D}_\theta}$ is empty, then we take $\widetilde{\omega} := \omega'$, and there is nothing to do. If $\widetilde{\mathcal{C}_\theta}$ is nonempty, then it must be of the form

$$\cfrac{\cfrac{\mathcal{C}'_\theta}{\widetilde{\theta}(\mathbf{M}^\alpha) \vee \widetilde{\theta}(\mathbf{N}^\alpha) \vee \widetilde{\theta}(\mathbf{C}) \vee \mathcal{F}}}{\widetilde{\theta}(\mathbf{M}^\alpha) \vee \widetilde{\theta}(\mathbf{C}) \vee \mathcal{F}}\ \mathcal{RC}(coll)$$

where $\widetilde{\theta}(\mathbf{M}) \equiv \widetilde{\theta}(\mathbf{N})$, then we can imitate the step with the following one

$$\cfrac{\cfrac{\mathcal{C}'}{\mathbf{M}^\alpha \vee \mathbf{N}^\alpha \vee \mathbf{C} \vee \mathcal{E}}}{\mathbf{M}^\alpha \vee \mathbf{C} \vee \mathbf{M} \neq^? \mathbf{N} \vee \mathcal{E}}\ \mathcal{HR}(F)$$

Furthermore, we have $\mathbf{PU}(\mathcal{F}) = \mathbf{PU}(\mathcal{E} \vee \widetilde{\theta}(\mathbf{M}) \neq^? \widetilde{\theta}(\mathbf{N}))$ since $\widetilde{\theta}(\mathbf{M}) \equiv \widetilde{\theta}(\mathbf{N})$. Moreover we have $\mathbf{PU}(\mathcal{F}) \equiv_{\widetilde{\theta}} \mathbf{PU}(\mathcal{E})$ by inductive hypothesis and thus $\mathbf{PU}(\mathcal{F}) \equiv_{\widetilde{\theta}} \mathbf{PU}(\mathcal{E} \vee \mathbf{M} \neq^? \mathbf{N})$. Thus $\widetilde{\omega} := \omega_{-\mathbf{N}^\alpha}$ is a $\widetilde{\theta}$-compatible clause isomorphism, so we obtain the assertion by inductive hypothesis.

Finally, it only remains to combine the two $\mathcal{HR}$-derivations, we have constructed in this proof, according to the following commutative diagram:

$$\begin{array}{ccccc}
\mathcal{C}_\theta & \vdash_{\mathcal{RC}} & \mathcal{C}'_\theta & \vdash_{\mathcal{RC}} & \widetilde{\mathcal{C}}_\theta \\
\theta \big\downarrow \omega & & \theta' \big\uparrow \omega' & & \widetilde{\theta} \big\downarrow \widetilde{\omega} \\
\mathcal{C} & \vdash_{\mathcal{HR}} & \mathcal{C}' & \vdash_{\mathcal{HR}} & \widetilde{\mathcal{C}}
\end{array}$$

□

Lemma 6.4.6 ($\mathcal{RC}$-Normalization Lifting Lemma)
Let θ be a substitution and $\mathbf{A} \in wff_o(\Sigma)$, then there is a $\mathcal{HR}$-derivation $\mathcal{D}$ of a set Φ of initial clauses from $\mathbf{CNF}(\mathbf{A}^\alpha)$, a substitution $\widetilde{\theta}$, and a $\widetilde{\theta}$-compatible clause set isomorphism $\widetilde{\omega}: \mathbf{CNF}(\theta(\mathbf{A})^\alpha) \longrightarrow \Phi$, so that we have the following commutative diagram.

$$\begin{array}{ccccc}
\theta(\mathbf{A})^\alpha & & \vdash_{\mathcal{RC}} & & \mathbf{CNF}(\theta(\mathbf{A})^\alpha) \\
\theta \big\downarrow \emptyset & & & & \widetilde{\theta} \big\downarrow \widetilde{\omega} \\
\mathbf{A}^\alpha & \vdash_{\mathcal{RC}} & \mathbf{CNF}(\mathbf{A}^\alpha) & \vdash_{\mathcal{HR}} & \Phi
\end{array}$$

Proof: Let $\Theta = \mathbf{CNF}(\theta(\mathbf{A})^\alpha)$ and $\Psi = \mathbf{CNF}(\mathbf{A}^\alpha)$. Furthermore, let $\mathcal{C} \in \Psi$ be a clause and $\mathcal{R}: \mathbf{A}^\alpha \vdash_{\mathcal{RC}} \mathcal{C}$ its reduction to clause form. Clearly the same $\mathcal{RC}$-reductions also apply to $\theta(\mathbf{A}^\alpha)$, since if $\mathbf{head}(\mathbf{A}) \in \{\vee, \neg, \Pi^\alpha\}$, then $\mathbf{head}(\mathbf{A}) = \mathbf{head}(\theta(\mathbf{A}))$. Thus we have an $\mathcal{RC}$-derivation $\mathcal{D}: \theta(\mathbf{A}^\alpha) \vdash_{\mathcal{RC}} \mathcal{C}_\theta$ and a clause isomorphism $\omega: \mathcal{C}_\theta \longrightarrow \mathcal{C}$. As the constraint parts of $\mathcal{C}$ and $\mathcal{C}_\theta$ are empty, ω is θ-compatible by 5.3.3 (where we take $\mathcal{E} = \mathbf{T}_o$).

Now $\mathcal{C}_\theta$ need not be a clause yet, so let $\mathcal{R}: \mathcal{C}_\theta \vdash_{\mathcal{RC}} \mathcal{C}'_\theta \in \Theta$ be a maximal $\mathcal{RC}$-derivation. By the lifting lemma for $\mathcal{RC}$ (6.4.5) there is a clause $\mathcal{C}'$, a $\mathcal{HR}$-derivation $\mathcal{D}: \mathcal{C} \vdash_{\mathcal{HR}} \mathcal{C}'$, a substitution $\theta' = \theta_{\mathcal{C}'}$ and a θ'-compatible clause isomorphism $\omega' = \omega_{\mathcal{C}'}: \mathcal{C}'_\theta \longrightarrow \mathcal{C}'$.

Thus for any clause $\mathcal{C} \in \Psi$ and each clause $\mathcal{C}_\theta \in \Theta$ we have a $\mathcal{HR}$-derivation $\mathcal{C} \vdash_{\mathcal{HR}} \mathcal{C}'$ and a θ'-compatible clause isomorphism $\omega': \mathcal{C}'_\theta \longrightarrow \mathcal{C}'$. Hence we obtain the assertion by collecting all such clauses $\mathcal{C}'$ in the set Φ. If we take care to keep the domains of the contexts in the respective derivations disjoint, then the substitutions $\theta_{\mathcal{C}'}$ all have the form $\theta \cup \rho_{\mathcal{C}'}$, where all $\rho_{\mathcal{C}'}$ have disjoint domains, so we can construct a single substitution $\widetilde{\theta} := \theta \cup \bigcup_{\mathcal{C}' \in \Theta} \rho_{\mathcal{C}'}$, which verifies the assertion. □

Lemma 6.4.7 (Lifting Lemma for $\mathcal{HR}$)
Let $\omega: \Theta \longrightarrow \Psi$ be a θ-compatible clause set isomorphism and $\mathcal{D}: \Theta \vdash_{\mathcal{HR}} \mathcal{C}_\theta$ such that the constraint of $\mathcal{C}_\theta$ is pre-unifiable, then there is a $\mathcal{HR}$-derivation $\Psi \vdash_{\mathcal{HR}} \mathcal{C}$ and a θ'-compatible clause isomorphism $\omega': \mathcal{C}_\theta \longrightarrow \mathcal{C}$ for a substitution θ' with $\theta' = \theta[$

$Dom(\theta)$], so that we have the following commutative diagram.

$$\begin{tikzcd} \Theta & \vdash_{\mathcal{HR}} & \mathcal{C}_\theta \arrow[d, "\theta'"', "\omega'"] \\ \Psi \arrow[from=u, "\theta"', "\omega"] & \vdash_{\mathcal{HR}} & \mathcal{C} \end{tikzcd}$$

Proof: We prove the assertion by induction on the structure of $\mathcal{D}$. If $\mathcal{D}$ is the empty tree, then we choose $\omega' := \omega$ and $\theta' := \theta$, and obtain the assertion from the definition of clause set isomorphism.

If $\mathcal{D}$ ends in a $\mathcal{HR}(R)$ step, then we have the following situation:

$$\dfrac{\dfrac{\Theta}{\dfrac{}{\mathcal{C}'_\theta}}\,\mathcal{D}' \qquad \dfrac{\Theta}{\dfrac{}{\mathcal{C}''_\theta}}\,\mathcal{D}''}{\mathcal{C}_\theta}\,\mathcal{HR}(R)$$

By inductive hypothesis we have $\mathcal{HR}$-derivations $\mathcal{R}' \colon \Psi \vdash_{\mathcal{HR}} \mathcal{C}'$ and $\mathcal{R}'' \colon \Psi \vdash_{\mathcal{HR}} \mathcal{C}''$ and θ'-compatible clause isomorphisms $\omega' \colon \mathcal{C}'_\theta \longrightarrow \mathcal{C}'$ and $\omega'' \colon \mathcal{C}''_\theta \longrightarrow \mathcal{C}''$ for a substitution θ'. We can restrict ourselves to a single substitution θ' here, since the parent clauses can be renamed to suitable α-variants that have disjoint variable contexts. Since ω' and ω'' are clause isomorphisms, we must have $\mathcal{C}' = \mathbf{A}^\alpha \vee \mathbf{C}' \vee \mathcal{E}'$ and $\mathcal{C}'' = \mathbf{B}^\beta \vee \mathbf{C}'' \vee \mathcal{E}''$ with $\alpha \neq \beta$, and $\mathcal{D}$ must be of the form

$$\dfrac{\dfrac{\Theta}{\dfrac{}{\theta(\mathbf{A})^\alpha \vee \theta(\mathbf{C}') \vee \mathcal{F}'}}\,\mathcal{D}' \qquad \dfrac{\Theta}{\dfrac{}{\theta(\mathbf{B})^\beta \vee \theta(\mathbf{C}'') \vee \mathcal{F}''}}\,\mathcal{D}''}{\theta(\mathbf{C}) \vee \mathcal{F} \vee \theta(\mathbf{A}) \neq^? \theta(\mathbf{B})}\,\mathcal{HR}(R)$$

where $\mathbf{C} := \mathbf{C}' \vee \mathbf{C}''$ and $\mathcal{F} := \mathcal{F}' \vee \mathcal{F}''$. Let $\mathcal{R}$ be the following $\mathcal{HR}$-derivation:

$$\dfrac{\dfrac{\Psi}{\dfrac{}{\mathbf{A}^\alpha \vee \mathbf{C}' \vee \mathcal{E}'}}\,\mathcal{R}' \qquad \dfrac{\Theta}{\dfrac{}{\mathbf{B}^\beta \vee \mathbf{C}'' \vee \mathcal{E}''}}\,\mathcal{R}''}{\mathbf{C} \vee \mathcal{E} \vee \mathbf{A} \neq^? \mathbf{B}}\,\mathcal{HR}(R)$$

where $\mathcal{E} := \mathcal{E}' \vee \mathcal{E}''$. Now we trace the clause isomorphism through the $\mathcal{HR}$-derivation. Let us first treat clauses as multisets to avoid the hassle with factoring. By θ'-compatibility from the inductive hypothesis we have $\mathbf{PU}(\mathcal{E}') \equiv_{\theta'} \mathbf{PU}(\mathcal{F}')$ and $\mathbf{PU}(\mathcal{E}'') \equiv_{\theta'} \mathbf{PU}(\mathcal{F}'')$. Moreover we have

$$\mathbf{PU}(\mathcal{E} \vee \mathbf{A} \neq^? \mathbf{B}) \equiv_{\theta'} \mathbf{PU}(\mathcal{F} \wedge \theta'(\mathbf{A}) \neq^? \theta'(\mathbf{B}))$$

which proves that the obvious choice for ω is θ'-compatible. The $\mathcal{HR}(F)$ case is analogous.

tautology
elementary

$\mathcal{HR}(P)$ and $\mathcal{UA}(\mathit{flex}/\mathit{rig})$ share the following common structure: they introduce a general binding $F \neq^? \mathbf{G}$ for a variable $F \notin \mathbf{Dom}(\theta)$. Since $F \notin \mathbf{Dom}(\theta)$, we can lift the rule applications directly (i.e. create a more general $\mathcal{HR}$-derivation by introducing exactly the same general binding), and obtain the assertion by the methods we have exemplified for the $\mathcal{HR}(R)$ case above. For the $\mathcal{HR}(E)$ case we have the following situation

$$\frac{\theta(\mathbf{C}) \vee \mathcal{F} \vee F \neq^? \mathbf{A}}{\mathbf{D} \vee [\mathbf{A}/F]\mathcal{F} \vee F \neq^? \mathbf{A}}\,\mathcal{HR}(E)$$

where $[\mathbf{Z}/F]\theta(\mathbf{C}) \vdash_{\mathcal{RC}} \mathbf{D}$, so we obtain the assertion with 6.4.5. $\Box$

Now we have the fragments that we will piece together to the proof of the main lifting theorem.

Theorem 6.4.8 ($\mathcal{HR}$-Refutation Lifting)
Let $\mathbf{A} \in \mathit{wff}_o(\Sigma)$ and $\theta \in \mathbf{SUB}$, then $\mathbf{A}$ is $\mathcal{HR}$-refutable, if $\theta(\mathbf{A})$ is.

Proof: Let $\Theta = \mathbf{CNF}(\theta(\mathbf{A}))$ and $\Psi = \mathbf{CNF}(\mathbf{A})$, then by the $\mathcal{RC}$-normalization lifting lemma 6.4.6 there is a set Φ of clauses, a substitution θ' with $\theta \equiv \theta'[\mathbf{Dom}(\theta)]$, and a θ'-compatible clause set isomorphism $\omega\colon \Phi \longrightarrow \Theta$. Furthermore, there is a $\mathcal{HR}$-derivation $\mathcal{H}\colon \Psi \vdash_{\mathcal{HR}} \Phi$. We continue the proof according to the following diagram:

$$\begin{tikzcd}
\theta(\mathbf{A}) \arrow[d, "\theta"] & & \vdash_{\mathcal{RC}} & & \Theta \arrow[d, "\theta'"', "\omega'"] & \vdash^{\mathcal{D}}_{\mathcal{HR}} & \Box \arrow[d, "\theta''"', "\omega''"] & & \\
\mathbf{A} & \vdash_{\mathcal{RC}} & \mathbf{CNF}(\mathbf{A}) & \vdash^{\mathcal{H}}_{\mathcal{HR}} & \Phi & \vdash_{\mathcal{HR}} & \mathcal{F} & \vdash_{\mathcal{HR}} & \Box
\end{tikzcd}$$

Let $\mathcal{D}$ be the $\mathcal{HR}$-refutation of $\theta(\mathbf{A})$, i.e. $\mathcal{D}\colon \Theta \vdash_{\mathcal{HR}} \Box$, then by the lifting lemma for $\mathcal{HR}$ (6.4.7) there is a $\mathcal{HR}$-derivation $\mathcal{R}'\colon \Psi \vdash_{\mathcal{HR}} \mathbf{C} \vee \mathcal{F}$ and a θ''-compatible clause isomorphism $\omega''\colon \mathcal{F} \longrightarrow \mathcal{E}$. Thus $\mathbf{C}$ is the empty disjunction and $\mathbf{PU}(\emptyset) \equiv_{\theta''} \mathbf{PU}(\mathcal{F})$. Now, $\mathbf{PU}(\emptyset)$ is non-empty, so $\mathcal{F}$ is unifiable, so there is a $\Sigma\mathcal{PT}$-derivation that transforms $\mathcal{F}$ to pre-solved form by 3.6.6. Let $\mathcal{R}''$ be the corresponding $\mathcal{HR}$-derivation, then we obtain a $\mathcal{HR}$-refutation of $\mathbf{A}$ by $\mathcal{H}\colon \Psi \vdash^{\mathcal{H}}_{\mathcal{HR}} \Psi \vdash^{\mathcal{R}'}_{\mathcal{HR}} \mathcal{F} \vdash^{\mathcal{R}''}_{\mathcal{HR}} \mathcal{F}$. Note that since $\mathcal{HR}(E)$ drops the solved pairs it acts upon, $\mathcal{E}$ is empty, so we have derived the empty clause and proven the assertion. $\Box$

Definition 6.4.9 (Tautology)
A clause $\mathcal{C} = \mathbf{M}^{\mathtt{T}} \vee \mathbf{N}^{\mathtt{F}} \vee \mathbf{C} \vee \mathcal{E}$ is called a **tautology**, if $\mathbf{M} =_{\beta\eta} \mathbf{N}$ and $\mathcal{E}$ is pre-solved. $\mathcal{C}$ is called **elementary**, iff $\mathbf{C}$ empty.

Remark 6.4.10 If $\mathcal{C}$ is an elementary tautology and $\Phi, \mathcal{C} \vdash_{\mathcal{HR}} \Box$, then $\Phi \vdash_{\mathcal{HR}} \Box$.

Proof: Let $\Phi, \mathcal{C} \vdash_{\mathcal{HR}} \Box$, by the lifting lemma (6.4.7) we have a $\mathcal{HR}$-refutation $\Phi * \mathcal{C}'$ where $\mathcal{C}' = P^{\mathtt{F}} \vee P^{\mathtt{T}}$. Note that the only inference rule that can be applied to $\mathcal{C}'$ is a $\mathcal{HR}(R)$ step of the form

$$\frac{\mathcal{D} = \mathbf{B}^{\alpha} \vee \mathbf{C} \quad P^{\mathtt{F}} \vee P^{\mathtt{T}}}{\mathcal{D}' = \mathbf{C} \vee P^{\alpha} \vee P \neq^{?} \mathbf{B}} \mathcal{HR}(R)$$

Clearly any $\mathcal{HR}$-derivation using $\mathcal{D}'$ can use $\mathcal{D}$ instead. $\Box$

6.5 Completeness of $\mathcal{HR}$

We now investigate the relative completeness of $\mathcal{HR}$ using the model existence theorems from chapter 2. The main technical result of this section will be that that the class $\Gamma_{\Sigma} := \{\Phi \subset wff_o(\Sigma) \mid \Phi \not\vdash_{\mathcal{HR}} \Box\}$ is an abstract consistency class. This gives us completeness of $\mathcal{HR}$.

Lemma 6.5.1 (Disjunction Lemma)
Let Φ be a set of clauses and $\mathbf{A}, \mathbf{B} \in wff_o(\Sigma)$, such that $\mathbf{Free}(\mathbf{A}) \cap \mathbf{Free}(\mathbf{B}) = \emptyset$. If $\Phi * \mathcal{R}.\mathbf{A} \vdash_{\mathcal{HR}} \mathbf{C}$, then $\Phi * \mathcal{R}'.\mathbf{A} \vee \mathbf{B} \vdash_{\mathcal{HR}} \mathbf{C}$ or $\Phi * \mathcal{R}.\mathbf{A} \vee \mathbf{B} \vdash_{\mathcal{HR}} \mathcal{R}'.\mathbf{C} \vee \mathbf{B}$.

Proof: Let $\mathcal{D}: \Phi * \mathcal{R}.\mathbf{A} \vdash_{\mathcal{HR}} \mathcal{R}'.\mathbf{C}$, then we prove the assertion by induction over $\mathcal{D}$. If $\mathcal{D}$ is empty, then $\mathcal{R}'.\mathbf{C} \in \Phi$ or $\mathbf{C} = \mathbf{A}$. In both cases we obtain the assertion with the empty $\mathcal{HR}$-derivation.

If $\mathcal{D}$ ends in $\mathcal{HR}(R)$, then $\mathcal{D}$ is of the form

$$\frac{\dfrac{\Phi * \mathcal{R}.\mathbf{A}}{\mathcal{R}^1\mathbf{N}^{\alpha} \vee \mathbf{C}_1} \mathcal{D}' \qquad \dfrac{\Phi * \mathcal{R}.\mathbf{A}}{\mathcal{R}^2.\mathbf{M}^{\beta} \vee \mathbf{C}_2} \mathcal{D}'' \qquad \alpha \neq \beta}{\mathcal{R}^1 \cup \mathcal{R}^2.\mathbf{C}_1 \vee \mathbf{C}_2 \vee \mathbf{M} \neq^{?} \mathbf{N}} \mathcal{HR}(R)$$

By inductive hypothesis we have $\mathcal{HR}$-derivations

1. $\Phi * \mathcal{R}.\mathbf{A} \vee \mathbf{B} \vdash \mathcal{R}^1\mathbf{N}^{\alpha} \vee \mathbf{C}_1$ or $\Phi * \mathcal{R}.\mathbf{A} \vee \mathbf{B} \vdash_{\mathcal{HR}} \mathcal{R}^1.\mathbf{N}^{\alpha} \vee \mathbf{C}_1 \vee \mathbf{B}$

2. $\Phi * \mathcal{R}.\mathbf{A} \vee \mathbf{B} \vdash \mathcal{R}^2.\mathbf{M}^{\beta} \vee \mathbf{C}_2$ or $\Phi * \mathcal{R}.\mathbf{A} \vee \mathbf{B} \vdash_{\mathcal{HR}} \mathcal{R}^2\mathbf{M}^{\beta} \vee \mathbf{C}_2 \vee \mathbf{B}$

Thus we have four cases, but as all of them can be treated with the same methods, we show the most complex one, where for both applications of the inductive hypothesis the latter alternative is assumed. In this case, we have

$$\dfrac{\dfrac{\dfrac{\dfrac{\Phi * \mathcal{R}.\mathbf{A} \vee \mathbf{B}}{\mathcal{R}^1.\mathbf{N}^\alpha \vee \mathbf{C}_1 \vee \mathbf{B}}\,\mathcal{D}' \qquad \dfrac{\Phi * \mathcal{R}.\mathbf{A} \vee \mathbf{B}}{\mathcal{R}^2.\mathbf{M}^\beta \vee \mathbf{C}_2 \vee \mathbf{B}}\,\mathcal{D}'' \qquad \alpha \neq \beta}{\mathcal{R}^1 \cup \mathcal{R}^2.\mathbf{C}_1 \vee \mathbf{C}_2 \vee \mathbf{M} \neq^? \mathbf{N} \vee \mathbf{B}_1 \vee \mathbf{B}_2}\,\mathcal{HR}(R)}{\mathcal{R}^1 \cup \mathcal{R}^2.\mathbf{C}_1 \vee \mathbf{C}_2 \vee \mathbf{M} \neq^? \mathbf{N} \vee \mathbf{B}_1 \vee \mathbf{B}_2 \vee \mathbf{B}_1 \neq^? \mathbf{B}_2}\,\mathcal{HR}(F)}{\mathcal{R}'.\mathbf{C}_1 \vee \mathbf{C}_2 \vee \mathbf{M} \neq^? \mathbf{N} \vee \mathbf{B}_1 \vee \mathbf{B}_2 \vee \mathbf{B}}$$

Note that the $\mathbf{B}_1$ and $\mathbf{B}_2$ come from renaming the $\mathbf{B}$ in the $\mathcal{HR}(R)$ step, which makes them different. Therefore we have to factor them, the unification of this pair computes a renaming substitution, which makes them equal.

The $\mathcal{HR}(F)$ case is similar but much less complex, since we only need one application of the inductive hypothesis and avoid renaming. In the $\mathcal{HR}(E)$ case we have the following situation:

$$\dfrac{\dfrac{\Phi * \mathbf{A}}{\mathcal{C} = \mathbf{C} \vee F \neq^? \mathbf{E}}\,\mathcal{D}'}{\mathbf{D} \vee F \neq^? \mathbf{E}}\,\mathcal{HR}(E)$$

where $[\mathbf{E}/F]\mathbf{C} \vdash_{\mathcal{RC}} \mathbf{D}$ by a maximal $\mathcal{RC}$-derivation. By inductive hypothesis we have $\Phi * \mathbf{A} \vee \mathbf{B} \vdash \mathbf{C} \vee F \neq^? \mathbf{E}$ or $\Phi * \mathbf{A} \vee \mathbf{B} \vdash_{\mathcal{HR}} \mathbf{C} \vee F \neq^? \mathbf{E} \vee \mathbf{B}$. Note that $[\mathbf{A}/F]\mathbf{B} = \mathbf{B}$ since $\mathbf{Free}(\mathbf{B}) \cap \mathbf{Free}(\mathbf{A}) = \emptyset$, therefore we obtain the assertion by a single application of $\mathcal{HR}(E)$.

The remaining cases $\mathcal{HR}(P)$, $\Sigma\mathcal{PT}(\mathit{flex}-\mathit{rig})$, and $\Sigma\mathcal{PT}(\mathit{guess})$ are nearly trivial, since they simply add a pair to $\mathcal{C}$, thus we obtain the assertion directly by the inductive hypothesis. □

Lemma 6.5.2 ($\mathcal{HR}$ is Saturated)
Let $\Phi \subset \mathit{wff}_o(\Sigma)$ be $\mathcal{HR}$-consistent and $\mathbf{A} \in \mathit{wff}_o(\Sigma)$ be atomic, then $\Phi * \mathbf{A}$ is $\mathcal{HR}$-consistent or $\Phi * \neg\mathbf{A}$ is $\mathcal{HR}$-consistent.

Proof: To see that $\mathcal{HR}$ is saturated, let $\Psi := \mathbf{CNF}(\Phi)$, then $\mathbf{CNF}(\Phi * \mathbf{A}) = \Psi * \mathcal{R}.\mathbf{A}^{\mathsf{F}}$ and $\mathbf{CNF}(\Phi * \neg\mathbf{A}) = \Psi * \mathcal{R}.\mathbf{A}^{\mathsf{T}}$. To show the converse of the assertion we assume that both $\Phi * \mathbf{A}$ and $\Phi * \neg\mathbf{A}$ are $\mathcal{HR}$-inconsistent, i.e. there are $\mathcal{HR}$-refutations $\mathcal{D}\colon \Psi * \mathcal{R}.\mathbf{A}^{\mathsf{T}} \vdash_{\mathcal{HR}} \Box$ and $\mathcal{D}'\colon \Psi * \mathcal{R}.\mathbf{A}^{\mathsf{F}} \vdash_{\mathcal{HR}} \Box$.

By applying lemma 6.5.1 to $\mathcal{D}$ we obtain a $\mathcal{HR}$-refutation $\Psi * \mathcal{R}.\mathbf{A}^{\mathsf{T}} \vee \mathbf{A}^{\mathsf{F}}$ or a $\mathcal{HR}$-derivation $\Psi * \mathcal{R}.\mathbf{A}^{\mathsf{T}} \vee \mathbf{A}^{\mathsf{F}} \vdash_{\mathcal{HR}} \mathcal{R}.\mathbf{A}^{\mathsf{F}}$. Now lemma 6.4.10 guarantees a $\mathcal{HR}$-refutation of Φ, which proves the claim in this case, or a $\mathcal{HR}$-derivation $\mathcal{D}''\colon \Phi \vdash_{\mathcal{HR}} \mathcal{R}.\mathbf{A}^{\mathsf{F}}$. This can be combined with the derivation $\mathcal{D}'$ to to a refutation of Φ. □

Lemma 6.5.3 *Let Φ and Ψ be finite sets of sentences, and $\Phi \vee \Psi := \{\mathbf{A} \vee \mathbf{B} \mid \mathbf{A} \in \Phi, \mathbf{B} \in \Psi\}$. Furthermore, let $\Phi \vdash_{\mathcal{HR}} \Box$ and $\Psi \vdash_{\mathcal{HR}} \Box$, then $\Phi \vee \Psi \vdash_{\mathcal{HR}} \Box$.*

Proof: Let $\mathbf{B} \in \Psi$, $\Phi = \{\mathbf{A}^1, \ldots, \mathbf{A}^n\}$, and $\Phi^i := \{\mathbf{A}^1 \vee \mathbf{B}, \ldots, \mathbf{A}^i \vee \mathbf{B}, \mathbf{A}^{i+1}, \ldots, \mathbf{A}^n\}$, then we first convince ourselves by induction on i, that $\Phi^i \vdash_{\mathcal{HR}} \Box$ or $\Phi^i \vdash_{\mathcal{HR}} \mathbf{B}$. If $i = 0$, then $\Phi = \Phi^0 \vdash_{\mathcal{HR}} \Box$ by assumption. If $i > 0$, then $\Phi^{i-1} \vdash_{\mathcal{HR}} \Box$ or $\Phi^{i-1} \vdash_{\mathcal{HR}} \mathbf{B}$ by inductive hypothesis, thus we obtain the assertion by the lemma 6.5.1.

In particular, for $i = n$ we have $\mathcal{D}_{\mathbf{B}}\colon \Phi^n = \Phi \vee \{\mathbf{B}\} \vdash_{\mathcal{HR}} \Box$ or $\mathcal{D}_{\mathbf{B}}\colon \Phi \vee \{\mathbf{B}\} \vdash \mathbf{B}$ for all $\mathbf{B} \in \Psi$. If the first case is assumed for some $\mathbf{B} \in \Psi$, then we have proven the assertion. Otherwise we have $\Phi \vee \Psi \vdash_{\mathcal{HR}} \Psi$ by combining all $\mathcal{HR}$-derivations $\mathcal{D}_{\mathbf{B}}$, which we can combine with the $\mathcal{HR}$-refutation of Ψ to obtain the assertion. $\Box$

EdNote(30)

Lemma 6.5.4 *If $\Phi * \mathbf{A}^\alpha \vdash_{\mathcal{HR}} \mathbf{B} \vee \mathcal{E}$, then $\Phi \vdash_{\mathcal{HR}} \mathbf{B} \vee \mathcal{E}$, or there is a literal $\mathbf{C}^\beta$ with $\beta \neq \alpha$, such that $\Phi \vdash_{\mathcal{HR}} \mathbf{B} \vee \mathbf{C}^\beta \vee \mathcal{F}$, and $\mathbf{U}(\mathcal{E}) = \mathbf{U}(\mathcal{F} \wedge \mathbf{A} \neq^? \mathbf{C})$.*[30]

Proof: We prove the assertion by an induction on the structure of the $\mathcal{HR}$-derivation $\mathcal{R}\colon \Phi * \mathbf{A}^\alpha \vdash_{\mathcal{HR}} \mathbf{B} \vee \mathcal{E}$.

If $\mathcal{R}$ ends in $\mathcal{HR}(R)$, then we have the following situation:

$$\dfrac{\dfrac{\Phi * \mathbf{A}^\alpha}{\mathbf{M} \vee \mathbf{L} \vee \mathcal{E}'}\mathcal{R}' \qquad \dfrac{\Phi * \mathbf{A}^\alpha}{\mathbf{N} \vee \mathbf{O} \vee \mathcal{E}''}\mathcal{R}''}{\mathbf{M} \vee \mathbf{N} \vee \mathcal{E}' \vee \mathcal{E}'' \vee \mathbf{L} \neq^? \mathbf{O}}\,\mathcal{HR}(R)$$

where $\mathbf{B} = \mathbf{M} \vee \mathbf{N}$ and $\mathcal{E} = \mathcal{E}' \vee \mathcal{E}'' \vee \mathbf{L} \neq^? \mathbf{O}$. Now we have to consider two cases:

Let us first assume that both $\mathbf{M} \vee \mathbf{L} \vee \mathcal{E}'$ and $\mathbf{N} \vee \mathbf{O} \vee \mathcal{E}''$ are distinct from $\mathbf{A}^\alpha$, so we can apply the inductive hypothesis to $\mathcal{R}'$ and $\mathcal{R}''$ (this gives us 4 cases, of which we will only consider the most complicated one, where both residuated a literal $\mathbf{C}^\beta$), giving us $\mathcal{HR}$-derivations $\mathcal{D}'\colon \Phi\colon \mathbf{M} \vee \mathbf{L} \vee \mathbf{C}^\beta \vee \mathcal{F}'$ and $\mathcal{D}''\colon \Phi\colon \mathbf{N} \vee \mathbf{O} \vee \mathbf{D}^\beta \vee \mathcal{F}''$. Using these let $\mathcal{D}$ be the following $\mathcal{HR}$-derivation

$$\dfrac{\dfrac{\dfrac{\Phi}{\mathbf{M} \vee \mathbf{L} \vee \mathbf{C}^\beta \vee \mathcal{F}'}\mathcal{D}' \qquad \dfrac{\Phi}{\mathbf{N} \vee \mathbf{O} \vee \mathbf{D}^\beta \vee \mathcal{F}''}\mathcal{D}''}{\mathbf{M} \vee \mathbf{N} \vee \mathbf{C}^\beta \vee \mathbf{D}^\beta \vee \mathcal{F}' \vee \mathcal{E}'' \vee \mathbf{L} \neq^? \mathbf{O}}\,\mathcal{HR}(R)}{\mathbf{M} \vee \mathbf{N} \vee \mathbf{C}^\beta \vee \mathcal{F}' \vee \mathcal{F}'' \vee \mathbf{L} \neq^? \mathbf{O} \vee \mathbf{C} \neq^? \mathbf{D}}\,\mathcal{HR}(F)$$

If we choose $\mathcal{F} := \mathcal{F}' \vee \mathcal{F}'' \vee \mathbf{L} \neq^? \mathbf{O} \vee \mathbf{C} \neq^? \mathbf{D}$, then indeed $\mathcal{D}\colon \Phi \vdash \mathbf{B} \vee \mathbf{C}^\beta \vee \mathcal{F}$, so we only have to verify the condition on the unification constraints.

[30] EdNote: Where is this used?

By inductive hypothesis, we have $\mathbf{U}(\mathcal{E}') = \mathbf{U}(\mathcal{F}' \vee \mathbf{A} \not=^? \mathbf{C})$ and $\mathbf{U}(\mathcal{E}'') = \mathbf{U}(\mathcal{F}'' \vee \mathbf{A} \not=^? \mathbf{D})$ and thus

$$\begin{array}{rcl}
\mathbf{U}(\mathcal{E}) & = & \mathbf{U}(\mathcal{E}' \vee \mathcal{E}'' \vee \mathbf{L} \not=^? \mathbf{O}) \\
& = & \mathbf{U}(\mathcal{F}' \vee \mathcal{F}'' \vee \mathbf{A} \not=^? \mathbf{C} \vee \mathbf{A} \not=^? \mathbf{D} \vee \mathbf{L} \not=^? \mathbf{O}) \\
& = & \mathbf{U}(\mathcal{F}' \vee \mathcal{F}'' \vee \mathbf{L} \not=^? \mathbf{O} \vee \mathbf{A} \not=^? \mathbf{C} \vee \mathbf{C} \not=^? \mathbf{D}) \\
& = & \mathbf{U}(\mathcal{F} \vee \mathbf{A} \not=^? \mathbf{C})
\end{array}$$

which completes the proof of this case.

In the case that the resolution step actually involves $\mathbf{A}^\alpha$, we are in a slightly more specific situation:

$$\dfrac{\dfrac{\Phi * \mathbf{A}^\alpha}{\mathbf{B} \vee \mathbf{L} \vee \mathcal{E}'}\mathcal{R}' \qquad \mathbf{A}^\alpha}{\mathbf{B} \vee \mathcal{E}' \vee \mathbf{L} \not=^? \mathbf{A}}\mathcal{HR}(R)$$

where $\mathcal{E} = \mathcal{E}' \vee \mathbf{L} \not=^? \mathbf{A}$. Now we can again apply the inductive hypothesis to $\mathcal{R}'$ (again we will only consider the residuating case), giving us a $\mathcal{HR}$-derivation $\mathcal{D}' \colon \Phi \colon \mathbf{M} \vee \mathbf{L} \vee \mathbf{C}^\beta \vee \mathcal{F}'$. Using this let $\mathcal{D}$ be the following $\mathcal{HR}$-derivation

$$\dfrac{\dfrac{\Phi}{\mathbf{B} \vee \mathbf{L} \vee \mathbf{C}^\beta \vee \mathcal{F}'}\mathcal{D}'}{\mathbf{B} \vee \mathbf{C}^\beta \vee \mathcal{F}' \vee \mathbf{L} \not=^? \mathbf{A}}\mathcal{HR}(F)$$

If we choose $\mathcal{F} := \mathcal{F}' \vee \mathbf{L} \not=^? \mathbf{A}$, then indeed $\mathcal{D} \colon \Phi \vdash \mathbf{B} \vee \mathbf{C}^\beta \vee \mathcal{F}$, so we only have to verify the condition on the unification constraints. By inductive hypothesis, we have $\mathbf{U}(\mathcal{E}') = \mathbf{U}(\mathcal{F}' \vee \mathbf{A} \not=^? \mathbf{C})$ and thus

$$\begin{array}{rcl}
\mathbf{U}(\mathcal{E}) & = & \mathbf{U}(\mathcal{E}' \vee \mathbf{L} \not=^? \mathbf{A}) \\
& = & \mathbf{U}(\mathcal{F}' \vee \mathbf{A} \not=^? \mathbf{C} \vee \mathbf{L} \not=^? \mathbf{A}) \\
& = & \mathbf{U}(\mathcal{F}' \vee \mathbf{L} \not=^? \mathbf{A} \vee \mathbf{A} \not=^? \mathbf{C}) \\
& = & \mathbf{U}(\mathcal{F} \vee \mathbf{A} \not=^? \mathbf{C})
\end{array}$$

which completes the proof of the resolution case. The cases of the other $\mathcal{HR}$-inference rules are direct consequences of the inductive hypotheses. □

Now we are in the position to attack the completeness proof for $\mathcal{HR}$. We use the model existence theorem for Σ-models, which we have proven in chapter 2.

Theorem 6.5.5 *Let $\Gamma_\Sigma := \{\Phi \subset wff_o(\Sigma) \mid \Phi \not\vdash_{\mathcal{HR}} \Box\}$, then Γ_Σ is a saturated abstract consistency class.*

Proof: We verify the properties of 2.2.7 for some set $\Phi \in \Gamma_\Sigma$ of sentences. In the following let $\Theta := \mathbf{CNF}(\Phi)$.

Let $\mathbf{A} \in \Phi$ be an atom such that $\mathbf{A}, \neg\mathbf{A} \in \Phi$, then we have $\emptyset.\mathbf{A}^{\mathtt{T}}, \emptyset.\mathbf{A}^{\mathtt{F}} \in \Theta$ for the corresponding unit clauses. So we obtain the assertion with the following $\mathcal{HR}$-derivation

$$\dfrac{\dfrac{\emptyset.\mathbf{A}^{\mathtt{T}} \quad \emptyset.\mathbf{A}^{\mathtt{F}}}{\emptyset.\mathbf{A} \neq^? \mathbf{A}}\,\mathcal{HR}(R)}{\Box}\,\mathcal{SIM}$$

The remaining cases all share the following form: if $\mathbf{A} \in \Phi$, then $\Phi \cup \Psi \in \Gamma_\Sigma$ for some set of formulae Ψ. So we have to prove that, if $\mathbf{A} \in \Phi$ and $\Phi \not\vdash \Box$, then $\Phi \cup \mathbf{CNF}(\Psi) \not\vdash_{\mathcal{HR}} \Box$, or equivalently, that the existence of a $\mathcal{HR}$-refutation of $\Phi \cup \mathbf{CNF}(\Psi)$ guarantees a $\mathcal{HR}$-refutation of Φ. Condition 2.2.7($\nabla_\vee$) is a direct consequence of 6.5.3. For the cases ∇_β, $\nabla_\neg$, and $\nabla_\wedge$, the argumentation is trivial, since any $\mathcal{HR}$-refutation of $\Phi * \mathbf{A}{\downarrow}$, $\Phi * \neg\neg\mathbf{A}$ or $\Phi * \neg(\mathbf{A} \vee \mathbf{B})$ is also one for Φ, because the clause normal forms of $\mathbf{A}{\downarrow}$, $\neg\neg\mathbf{A}$ and that of $\mathbf{A}$ are identical, and finally, $\mathbf{CNF}(\neg(\mathbf{A} \vee \mathbf{B})) = \mathbf{CNF}(\{\neg\mathbf{A}, \neg\mathbf{B}\})$.

To verify 2.2.7($\nabla_\forall$) let $\Pi^\alpha\mathbf{A} \in \Phi$ and $\mathcal{D}_\theta$ be a $\mathcal{HR}$-refutation of $\Psi_\theta := \Phi * \mathbf{AB}$, where $\theta = [\mathbf{B}/X]$. Let $\Phi = \Phi' * \Pi^\alpha\mathbf{A}$, then $\mathcal{D}_\theta$ is also a $\mathcal{HR}$-refutation of $\Psi'_\theta := \Phi' \cup \{\mathbf{A}X, \Pi^\alpha\mathbf{A}\}$, since the clause normal forms of Ψ_θ and Ψ'_θ are equal. Now let $\Psi = \{\mathbf{A}X, \mathbf{A}Y\}$ and $\theta = [\mathbf{B}/Y]$, then $\theta(\Psi) = \Psi'_\theta$, so by 6.4.8 there is a $\mathcal{HR}$-refutation $\mathcal{D}$ of Ψ. As the clause normal forms of $\mathbf{A}X$ and $\mathbf{A}Y$ are α-variants, $\mathcal{D}$ is also a $\mathcal{HR}$-refutation of Φ.

For 2.2.7($\nabla_\exists$) let $\neg\Pi^\alpha\mathbf{A} \in \Phi$ and $\mathcal{D}$ be a $\mathcal{HR}$-refutation of $\Theta \cup \mathbf{CNF}(\mathcal{R}.\mathbf{A}w^{\mathtt{F}})$ for some witness constant $w \in \mathcal{W}$ that does not occur in Φ and $\mathcal{R} = [\emptyset/w]$. Note that the clause normal forms of $\Theta \cup \mathbf{CNF}(\mathcal{R}.\mathbf{A}w^{\mathtt{F}})$ and Φ are equal up to the choice of the witness constant w, since those of $\emptyset.(\Pi^\alpha\mathbf{A})^{\mathtt{F}}$ and $\mathcal{R}.(\mathbf{A}w)^{\mathtt{F}}$ are, so $\mathcal{D}$ is also a $\mathcal{HR}$-refutation of $\Theta \cup \mathbf{CNF}(\mathbf{A}w^{\mathtt{F}})$.

Finally Lemma 6.5.2 directly implies that Γ_Σ is saturated. $\Box$

Now the completeness theorem is only a simple corollary.

Corollary 6.5.6 (Completeness of $\mathcal{HR}$)
Let Φ be a finite set of sentences. If Φ is unsatisfiable in the class of Σ-models, then $\Phi \vdash_{\mathcal{HR}} \Box$.

Proof: Let $\Gamma_\Sigma := \{\Psi \subset \mathit{wff}_o(\Sigma) \mid \Psi \not\vdash_{\mathcal{HR}} \Box\}$. If Φ were not $\mathcal{HR}$-refutable, then $\Phi \in \Gamma_\Sigma$ by construction. Furthermore, by 6.5.5 Γ_Σ is a saturated abstract consistency class and therefore by 2.2.32 there is a Σ-model $\mathcal{M} \models \Phi$, which contradicts the assumption. $\Box$

Theorem 6.5.7 *A sentence $\mathbf{A}$ is valid in the class of Σ-models, iff $\neg\mathbf{A}$ is $\mathcal{HR}$-refutable.*

Proof: The result is an immediate consequence of 6.5.6 and 6.3.2. $\square$

Theorem 6.5.8 (Relative Completeness of $\mathcal{HR}$)
Let $\mathbf{A}$ *be a sentence, then* $\vdash_{\mathfrak{T}_\eta} \mathbf{A}$*, iff* $\mathbf{A}^{\mathtt{F}} \vdash_{\mathcal{HR}} \square$.

Proof: By the previous theorem and 2.3.19. $\square$

Now we can prove the statement about the completeness of our Hilbert-Style calculi $\mathfrak{T}_*$, which we made above in chapter 2.

Corollary 6.5.9 $\mathfrak{T}$ *and* $\mathfrak{T}_\eta$ *are not complete with respect to Henkin models.*

Proof: As we have seen in 6.2.3, there is a formula $\mathbf{C}$ that is not $\mathcal{HR}$-refutable. With the relative completeness theorem 6.5.8 above it is clear that $\neg\mathbf{C}$ is not derivable in $\mathfrak{T}_\eta$. Since the calculus $\mathfrak{T}$ is weaker than $\mathfrak{T}_\eta$ (because it lacks the η-axiom) the result also holds for $\mathfrak{T}$. $\square$

6.6 Resolution for Extensional Higher-Order Logic

To account for extensionality we propose the following two rules:

$$\frac{\mathbf{C} \vee \mathbf{A} \neq^? \mathbf{B}}{\mathbf{CNF}(\neg(\mathbf{A} \Leftrightarrow \mathbf{B})) \vee \mathbf{C}} \mathcal{ER}(Ref)$$

$$\frac{(P\mathbf{A})^{\mathtt{F}} \vee (P\mathbf{B})^{\mathtt{F}} \vee \mathbf{C}}{Q(\mathbf{A}X)^{\mathtt{F}} \vee Q(\mathbf{B}X)^{\mathtt{F}} \vee \mathbf{C}} \mathcal{ER}(Ext)$$

Obviously the first rule amounts to the recursive call of the refutation procedure. In our example above we have the following $\mathcal{HR}$-derivation $\mathcal{A}$

$$\cfrac{\cfrac{\cfrac{(cb)^{\mathtt{F}} \quad c(\neg\neg b)^{\mathtt{T}}}{c(\neg\neg b) \neq^? (cb)} \mathcal{HR}(R)}{(\neg\neg b) \neq^? b} \mathcal{SIM}}{b} \mathcal{ER}(Ref)$$

similarly we have a $\mathcal{HR}$-derivation $\mathcal{A}'$ of b, since the clause normal form of $\neg(\mathbf{A} \Leftrightarrow \mathbf{B})$ is $\{b^{\mathtt{T}}, b^{\mathtt{F}}\}$. Thus we can complete the refutation with

$$\cfrac{\cfrac{\cfrac{\mathcal{A}}{b} \quad \cfrac{\mathcal{A}'}{\neg b}}{b \neq^? b} \mathcal{HR}(R)}{\square} \mathcal{SIM}$$

Part III

Sorted Methods

Draft
30/6/2004

Draft 30/6/2004

[31] EdNote(31)

[31] EdNote: abstract here

Sorts in First-Order Automated Theorem Proving

The introduction of *sorted logics* has been one of the most successful contributions to first-order automated deduction. Sort techniques consist in syntactically distinguishing between objects of different semantic classes (foxes, wolves, numbers, points, lines, etc.); the essential idea behind sorted logic is to assign sorts (specifying the membership in some class) to objects and restrict the range of variables to particular sorts. Sorted logics have already been studied very early from a theoretical point of view by Herbrand [Her30], Schmidt [Sch38, Sch51], Wang [Wan52] and Oberschelp [Obe62]. This work has culminated in the so-called relativization theorems that no additional expressivity can be gained by introducing sorts. The relativization method encodes all sorted first-order formulae, proofs, and models into unsorted first-order logic, in such a way that entailment and provability are preserved. For instance, the formula $\forall X_{\mathbb{A}}.\mathbf{A}$ is transformed to $\forall X.\mathbb{A}(X) \Rightarrow \mathbf{Rel}(\mathbf{A})$, where $\mathbb{A}$ is a new unary predicate of the unsorted language. Declarations like $[c{::}\mathbb{A}]$ or $[\mathbb{A} \leq \mathbb{B}]$ from the signature are relativized to new "signature axioms" $\mathbb{A}(c)$ and $\forall X.\mathbb{A}(X) \Rightarrow \mathbb{B}(X)$. On the model theoretic side the algebras of partial functions are transformed into algebras of total functions by extending partial functions arbitrarily. So-called sort theorems now verify the coordination of the two notions of relativizations by stating that a sorted sentence $\mathbf{A}$ is satisfiable, iff its relativization $\mathbf{Rel}(\mathbf{A})$ has a model that also satisfies the signature axioms. Thus from a purely theoretical point of view sorted logics are rather uninteresting.

However from a computational perspective, they are, since the exploitation of sort information in the search for proofs can dramatically reduce the search space associated with theorem proving (see e.g. [Wal85]), and hence the resulting sorted calculi are much more efficient for deduction purposes. In the context of first-order logic, sort information has been successfully employed by Walther [Wal83, Wal85, Wal88], Schmidt-Schauß [SS86, SS89], Cohn [Coh87, Coh89, Coh92], Frisch [Fri90, CF92], Weidenbach [Wei95, Wei96a], and others.

In unsorted logics the only way to express the knowledge that an object is a member of a certain class of objects is through the use of unary predicates, such as the predicate $\mathbb{N}$ in the formulae $(\mathbb{N}2)$, i.e. "2 is a natural number" or $\neg(\mathbb{N}\mathrm{Peter})$ with the meaning "Peter is not a natural number". This leads to a multitude of unit clauses in the deduction that only carry the sort information. Furthermore, in unsorted logics quantification is unrestricted, whereas in practice one often wants quantifications to range only over the objects in a certain class. The latter kind of quantification can always be formulated with formulae like $\forall X.(\mathbb{N}X) \Rightarrow (X \geq 0)$. But the approach is unsatisfactory, because inter alia the derivation of the nonsensical formula $(\mathbb{N}\mathrm{Peter}) \Rightarrow (\mathrm{Peter} \geq 0)$ is permitted, even though $(\mathrm{Peter} \geq 0)$ can never be derived because of $\neg(\mathbb{N}\mathrm{Peter})$.

restricted quantification
variable declaration
constant declaration
subsort declaration
order-sorted
partial function
term declarations
term declaration

Sorted logics remedy this situation by assigning sorts to terms and require that formulae meet certain restrictions to denote meaningful objects: an application $(\mathbf{AB})$ is called *well-sorted*, iff there are sorts $\mathbb{A}$ and $\mathbb{B}$ such that $\mathbf{A}$ is of sort $\mathbb{B} \rightarrow \mathbb{A}$ and $\mathbf{B}$ is of sort $\mathbb{B}$. In this case the sort of $(\mathbf{AB})$ is $\mathbb{A}$. Furthermore, sorted logics provide mechanisms for restricted quantification, where the truth value of a formula $\forall X_{\mathbb{A}}.\mathbf{A}$ depending on the instances of $\mathbf{A}$ with respect to objects of sort $\mathbb{A}$. Thus in a sorted logic the quantified formula above would read $\forall X_{\mathbb{N}}.(\geq X 0)$ where $\geq$ would be declared to be a binary relation on $\mathbb{N}$ and 0 to be of sort $\mathbb{N}$.

The set of declarations for sort information is traditionally called the *signature* of the sorted logic. Classical sorted logics know three mechanisms for declaring this sort information:

- Variables can be restricted to sorts via declarations of the form $[X{::}\mathbb{A}]$ where X is a variable and $\mathbb{A}$ is a sort. In fact, most sorted logics postulate a total sort function, that associates a unique sort with each variable.

- Constants and functions can be declared to belong to certain sorts by declarations of the form $[c{::}\mathbb{A}]$ and $[f{::}\mathbb{A} \rightarrow \mathbb{B}]$.

- Subsort information can be declared by declarations of the form $[\mathbb{A} \leq \mathbb{B}]$. This induces a subsort relation on sorts, which is the smallest partial ordering that contains these subsort declarations. The subsort relation plays such a central role in sorted logics that these are often called "order-sorted". It is a useful notion to employ, since it allows the specification of hierarchies of sorts, which encode the definitional taxonomies of objects that play a great role in mathematics. Examples of such taxonomies are the hierarchies in algebra (semigroups, monoids, rings, fields,...) or numbers (natural numbers, integers, rationals, complex,...).

Naturally, these declarations have an effect on the notion of a model of a logic system. The carrier set $\mathcal{D}$ has subsets $\mathcal{D}_{\mathbb{A}}$ corresponding to the sorts. The correct semantical notion for functions is that of partial functions, which obey the declarations of the signature, i.e. if $[f{::}\mathbb{A} \rightarrow \mathbb{B}]$ is declared in the signature, the denotation of f must be a partial function that is total on $\mathcal{D}_{\mathbb{A}}$, and moreover $f(\mathcal{D}_{\mathbb{A}}) \subseteq \mathcal{D}_{\mathbb{B}}$. For each subsort declaration $[\mathbb{A} \leq \mathbb{B}]$ we must have $\mathcal{D}_{\mathbb{A}} \subseteq \mathcal{D}_{\mathbb{B}}$. This semantics reflects the fact that humans use certain classes to structure the universe and that mathematicians naturally use variables and functions restricted to these classes. Thus the sorted models are closer to the intuition of mathematicians than the unsorted ones.

In his sorted logic with *term declarations* [SS89] Manfred Schmidt-Schauß relaxes the implicit condition that only the sorts of constants and variables can be declared, and allows declarations of the form $[\mathbf{A}{::}\mathbb{A}]$, called *term declarations*, where $\mathbf{A}$ can be an arbitrary formula. The idea of term declarations

conditional term declarations
dynamic sorts

is that there can be sort information within the structure of a formula, if the formula matches a certain schema (a term declaration). Consider, for instance, the addition function, which (semantically) we would like to have the sort $\mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$ where $\mathbb{N}$ is the sorts of natural numbers. If we also have a sort for the even numbers $\mathbb{E}$, then we might want to specify that the expression $[\mathbf{A} + \mathbf{A}]$ is an even number, even if $\mathbf{A}$ is not. This information can be formalized by declaring the formula $[X_{\mathbb{N}} + X_{\mathbb{N}}]$ to be of sort $\mathbb{E}$ using a term declaration. In this expressive system term declarations of the form $[X_{\mathbb{A}}::\mathbb{B}]$ entail that $\mathbb{A}$ is a subsort of $\mathbb{B}$ and induce the intended subsort ordering on the set of sorts.

Research in sorted first-order logics and sorted deduction has primarily centered around the following topics:

***Unification*:** Here the impetus has been in developing expressive sorted logics, finding unification algorithms, and proving complexity results for them. For instance, unification in sorted logics is decidable and unitary, if the subsort relation is tree-like, and finitary, in the case where only function declarations are allowed [Wal84, Wal88]. Schmidt-Schauss proved that in the case of general term declarations, sorted unification is undecidable and infinitary. However, there are restricted classes of term declarations where sorted unification is decidable (cf. [Wei96a] for a compilation of known decidability and complexity results).

***Sorted refutation calculi*:** The sorted resolution calculi of Walther [Wal83, Wal87], and Schmidt-Schauß [SS86, SS87, SS89] simply substitute sorted unification for unsorted unification to obtain a sorted refutation calculus (the power and generality of this approach has been made explicit by Frisch in [Fri90]). Here the taxonomic theory in the signature is fixed in advance and is completely separated from the object theory. Thus the sort theory can only influence the deduction by restraining the search space. The calculi of Cohn [Coh87, Coh89, Coh92], Beierle et al. [BHP$^+$92], and Weidenbach [WO90] even allow conditional term declarations. This mechanism allows the derivation of more sort information during a proof and thus makes the sort information that constrains the proof search more and more concise during the proof search. The most extreme calculus is the resolution calculus with dynamic sorts [Wei95, Wei96a], where the sort theory and the object theory are completely mixed, but the unification procedure used in the resolution inference always takes the current state of the sort theory into account.

***Logic programming*:** Since logic programming languages like Prolog are based on fragments of first-order logics and the operational semantics can be seen as a very restricted form of resolution, there are various order-sorted logic programming languages (such as, for instance TEL [SNGM87, Smo89] or the many-sorted language GÖDEL [HL94]) that take advantage of the sort mechanisms. In this setting the aspect of search control — the user has a clear understanding of the operational semantics (the search behavior) and uses it for programming — is not as important as the greater conciseness of problem formulations. Since sorted programming languages should not be less efficient than unsorted ones on unsorted problems, compilers of such languages often try to precompute sorts of terms at compile-time in order to reduce the complexity of the sorted unification needed at runtime. Practical type systems also provide mechanisms of polymorphism and type reconstruction to ease the burden of typing for the user.

analysis
domain
codomain
image
support
contravariance
covariance

Sorted λ-Calculi

The question of the behavior of higher-order logic under the constraints of a full sorted type structure is a natural one to ask, in particular, since calculi in this system promise the development of more powerful deduction systems for real mathematics.

In typed λ-calculi the idea of declaring sort information is very natural, as all objects are already typed, which amounts to a – very coarse – division of the universe into classes. The type system is merely refined by the sorts. This gives rise not only to new classes of objects (sorts $\mathbb{A}, \mathbb{B}$), but also of functions (sort $\mathbb{A} \to \mathbb{B}$), where domains and codomains are just the sorts. Thus a sorted higher-order logic seems to be the most adequate system, for example, to formalize analysis, since we now have constructs for the domain and codomain, the image and the support of a function, and for function restriction within the system.

Huet was the first to propose the study of a sorted version of higher-order logic in an appendix to [Hue72]. The unification problem in extensions of this system has since been studied by Nipkow and Qian [NQ92] and Pfenning and the author [KP93]. Furthermore, typed λ-calculi with order-sorted type structures have been of interest in the programming language community as a theoretical basis for object-oriented programming and for more expressive formalisms for higher-order algebraic specifications [Qia91, Car84, BL90, Pie91].

Nipkow and Qian [NQ92] consider a collection of sort systems parameterized by rules for contravariance in the domain sort. This principle states that, if $\mathbb{A} \leq \mathbb{B}$, then $\mathbb{B} \to \mathbb{C} \leq \mathbb{A} \to \mathbb{C}$ and (semantically) corresponds to implicit function restriction. The dual covariance principle states that, if $\mathbb{A} \leq \mathbb{B}$, then $(\mathbb{C} \to \mathbb{A}) \leq (\mathbb{C} \to \mathbb{B})$. The paper presents a unification algorithm for the resulting sorted calculi. Functional formulae in these calculi in general do not have unique supporting sorts. The consequent difficulties

intersection sorts
regular
extensionality

with extensionality are solved by studying unification under sorted equalities that have been restricted to appropriate domain sorts — such restrictions enable specification of a well-defined η-rule. Constant overloading and functional base sorts are not present in these calculi.

In [KP93] Pfenning and the author consider a calculus $\Lambda^{\rightarrow\&}$ with intersection sorts. The intersection operator & on sorts provides sorts $\mathbb{A}\&\mathbb{B}$ denoting the intersection of the sets denoted by $\mathbb{A}$ and $\mathbb{B}$. This calculus also supports contravariance in the domain sort and constant overloading. Permitting intersection sorts makes it possible to define a minimal sort for every formula, so that all signatures are regular. In this setting problems with extensionality are alleviated by allowing only typed abstractions and by defining a formula $\lambda X.\mathbf{M}$ to have the sort $\mathbb{A} \to \mathbb{B}$, iff $\mathbf{M}$ has sort $\mathbb{B}$ assuming that X has sort $\mathbb{A}$. η-equality is then a typed relation which preserves the sorts of formulae. This calculus has been generalized by Pfenning [Pfe92] to a λ-calculus with dependent types, which will be used as a logical framework extending LF in the Elf programming language [Pfe91].

The calculi mentioned above only allow for sorting the universe of individuals, so they are not directly comparable, in terms of expressive power, with the one presented in this book. Indeed, these calculi represent a principally different approach to deduction which appears to call for a semantics where functions are total functions on the types and where the sort information only specifies the behavior of these functions when restricted.

The works of Cardelli [Car88], Bruce and Longo [BL90], Curien and Ghelli [CG91], and Pierce [Pie91] treat variants of the system $F_{\leq}$ which encompass polymorphic intersection types (i.e. intersection types whose variables are explicitly quantified) and the interaction between these types, and various subsort relations. These calculi serve as computational models for functional programming languages and are much more expressive than those studied here, but since they are not intended for deduction purposes, their unification problems have not yet been addressed. In such calculi subsort declarations are not required to respect the functional structure of types, rendering the decidability of sort assignment a very complex issue.

Bill Farmer develops a version of higher-order logic LUTINS [Far91a, Far91b], where all objects of functional type are partial functions and uses it as the working language of the IMPS [FGT93] system, an interactive deduction system that has been used to formalize and prove a large variety of theories in mathematics. In this setting sorts are a derived notion, they are used as a device to characterize certain partial functions as total on the sets represented by the sorts and make computation much more efficient using this information. Experiments with the IMPS system have shown that the greater expressiveness of sorted logics is invaluable for formalizing mathematical statements and finding proofs for them.

$\Sigma\Lambda^{\rightarrow}$: A Sorted λ-Calculus

The logical system $\Sigma\mathcal{HOL}$ (sorted higher-order logic or classical sorted type theory) developed in this book is a sorted higher-order logic that is an instance of a sorted λ-calculus $\Sigma\Lambda^{\rightarrow}$ with term declarations and functional base sorts. We now proceed to discuss the primary features of $\Sigma\Lambda^{\rightarrow}$ and our results.

Since the terms "type" and "sort" are not used uniformly in the literature, let us now take a look at the underlying principles and fix the usage for this book. Both terms refer to the idea of annotating syntactic objects with semantic information about class membership. The notion of "type" has first been introduced by Russell to avoid paradoxes and antinomies in higher-order logics. As we have seen above, the term "sort" comes from first-order deduction systems, where the mechanism is used for representing part of an axiomatization into sort information that can be efficiently manipulated by the calculus. In typed λ-calculi the type mechanism is used in both ways without properly distinguishing them. In $\Sigma\Lambda^{\rightarrow}$ we want to make the type mechanism of simply typed λ-calculus more expressive without losing consistency of the language. Thus we separate both uses into a simple type system (for the safety aspect) and a sort system (for the additional expressiveness). In particular, we will use the term "type" to refer to a mechanism that is used for the safety aspect and the term "sort" for the representation aspect. Clearly the sort system has to conform to the type system in some way in order to ensure that no antinomies can be imported via the sort declarations. In our case, the sort system is a refinement of the underlying type system, and sorted operations will turn out to be refinements of their unsorted counterparts; in particular, well-sorted formulae are still well-typed in $\Sigma\Lambda^{\rightarrow}$.

The calculus $\Sigma\Lambda^{\rightarrow}$ differs from the systems described above in the following three principal ways:

***Functional base sorts*:** In addition to partitioning the function universes into the classes $\mathbb{A} \rightarrow \mathbb{B}$ of functions defined by domains $\mathbb{A}$ and codomains $\mathbb{B}$, the sort system of $\Sigma\Lambda^{\rightarrow}$ allows base sorts of functional type, i.e. base sorts that denote subclasses of the function classes $\mathbb{A} \rightarrow \mathbb{B}$. Syntactically, each sort $\mathbb{A}$ comes with a type $\tau(\mathbb{A})$, and — if is of functional type — also with a domain sort $\mathfrak{d}(\mathbb{A})$ and a codomain sort $\mathfrak{r}(\mathbb{A})$. Semantically, the sorts $\mathbb{C}$ denote subsets $\mathcal{D}_{\mathbb{C}}$ of the family of partial functions of type α, where α is the type of $\mathbb{C}$.

***Term Declarations*:** The term declaration mechanism is much more powerful than the declaration schemata proposed in the λ-calculi mentioned in 6.6. $\Sigma\Lambda^{\rightarrow}$ can be seen as a unifying framework which subsumes most known declaration mechanisms. For instance, sort inclusion (a concept that is primitive to most sorted logics) is a derivable

BegNP(32) *restrictive* *descriptive* *extensionality*

mechanism in $\Sigma\Lambda^{\rightarrow}$.

***Extensionality*:** As we have seen above, there is an issue about whether to model function restriction as a "built in" of the system, and as a consequence have contravariance rules in the subsort system. In this book, we will explore both possibilities, refining the system $\Sigma\Lambda^{\rightarrow}$ to a *restrictive* sort system $\Sigma^r\Lambda^{\rightarrow}$ and a *descriptive* sort system $\Sigma^d\Lambda^{\rightarrow}$. The first takes seriously the mathematical intuition that functions have uniquely specified domains. In $\Sigma^r\Lambda^{\rightarrow}$, formulae of functional type have unique supporting sorts, i.e. if a formula $\mathbf{A}$ has sorts $\mathbb{A}$ and $\mathbb{B}$, then $\mathfrak{d}(\mathbb{A}) = \mathfrak{d}(\mathbb{B})$. Consequently, our subsort relation cannot be contravariant in the domain sort. This property of $\Sigma\Lambda^{\rightarrow}$ allows us to give a meaningful account of the extensionality principle $(\forall X{::}\mathbb{A}.fX = gX \Rightarrow f = g)$, which relies on the concept of unique supporting sorts (unique domains of functions), even in a context without typed or sorted equality. In $\Sigma^d\Lambda^{\rightarrow}$, function restriction is built-in, here the problem of identifying supporting sorts can be circumvented by requiring a sorted notion of equality $=^{\mathbb{A}}$. For instance, if $\mathbb{R}$ ($\mathbb{P}$) is the sort of real (positive real numbers) and i (a) is the identity (absolute value) function on the real numbers, then we would have $i =^{\mathbb{P}} a$ but not $i =^{\mathbb{R}} a$. In systems like $\Sigma^r\Lambda^{\rightarrow}$ the extensionality principle has the form $\forall X{::}\mathbb{A}.fX =^{\mathbb{B}} gX \Rightarrow f =^{\mathbb{A}\rightarrow\mathbb{B}} g$ independent of the intuitive domain of f and g.

EndNP(32)

The aspects of an extensional sort system and functional base sorts are genuinely higher-order notions, and do not occur in first-order logic. Furthermore, they interact in a very subtle way. We have investigated a subsystem of $\Sigma\Lambda^{\rightarrow}$ that only allows signatures consisting of constant declarations and thus treats the interaction of functional base sorts and extensionality in isolation in [JK93, JK94]. In contrast to this subsystem, the powerful mechanism of term declarations in $\Sigma\Lambda^{\rightarrow}$ allows a straightforward specification of many mathematical concepts (cf. examples 7.7.22, 8.4.12, and 10.5.2).

The idea of using term declarations as a general framework for sorted logics was used in [SS89], so $\Sigma\Lambda^{\rightarrow}$ is a generalization of the system presented there. Achieving a formalization for the term declaration mechanism in the context of λ-calculi and higher-order accounts for most of the technical difficulties that we deal with in this book. The task is so difficult, since term declarations heavily interact with β-conversion (see the discussion in 7.2.10).

We have focused on the term declaration mechanism, since it yields a very general basis for understanding sorted λ-calculi at least in the context of Church-style logical systems for formalizing mathematics. Clearly intensional λ-calculi like Martin-Löf type theory [ML94] or the calculi of constructions [CH85] have much more general type systems, but these systems

[32] New Part:

are usually only used via the propositions as types isomorphism, and thus these sorts do not appear as a mechanism on the object level, which is just what we are interested in. Moreover, since the term declaration mechanism is in no way tied to extensionality or functional base sorts, we conjecture that it can be added to most type systems in the literature.

restrictive sort discipline
descriptive sort discipline

Chapter 7

Sorted λ-Calculus

In this section we define a sorted λ-calculus $\Sigma\Lambda^{\rightarrow}$ with term declarations and functional base sorts. This system is a generalization of the first-order system described in [SS89] and the system $\Lambda^{\rightarrow}$ presented in chapter 1. We start out with the system $\Lambda^{\rightarrow}$ and add more syntactic information to the formulae in order to distinguish certain well-typed formulae as well-sorted. Since well-sortedness is not a structural property, we give a sort inference system for well-sortedness.

As we have mentioned in the introduction to part **??**, types were developed as a syntactic means to distinguish mathematical objects of fundamentally differing nature and thereby eliminate antinomies and paradoxes from the formal system. We have seen that types (as a mechanism equivalent to flat sort hierarchies) can also serve as a powerful representation mechanism that allows to formalize disjoint sets as differing types. In $\Sigma\Lambda^{\rightarrow}$ we want to make the type mechanism more expressive without losing the safety aspect in terms of antinomies of a type system. Thus we separate both aspects into a simple type system (for the safety aspect) and a sort system (for the additional expressiveness). In fact, we only need one base type symbol ι for individuals in the presence of a sort system, since all other distinctions can and should be made within the sort system.

Clearly the sort system has to conform to the type system in some way, in order to ensure that no antinomies can be imported via the sort declarations. In our case the sort system is a refinement of the underlying type system and sorted operations will turn out to be refinements of their unsorted counterparts. In particular, well-sorted formulae are still well-typed.

We will present two variants of a sorted λ-calculus $\Sigma\Lambda^{\rightarrow}$ that we will develop simultaneously, since they share many of their characteristics and methods. In $\Sigma^r\Lambda^{\rightarrow}$ sorts are used to *restrict* the domains of functions that are not defined on all of the universe, and in $\Sigma^d\Lambda^{\rightarrow}$, they are used to *describe* the behavior of restrictions of a function. Whereas in $\Sigma^r\Lambda^{\rightarrow}$, the fact that a function f has sort $\mathbb{A} \rightarrow \mathbb{B}$ implies that it is defined exactly on $\mathcal{D}_{\mathbb{A}}$

$\mathcal{BS}$, $\mathcal{S}$ sets of base sorts and sorts
$\mathfrak{d}(\mathbb{A}), \mathfrak{r}(\mathbb{A}), \tau(\mathbb{A})$ domain, codomain sort and type of $\mathbb{A}$
sort system
base sort
sort!base
sort
domain sort
codomain sort

and that it is therefore an error to apply it to any argument outside of that domain, in $\Sigma^d\Lambda^{\rightarrow}$, it only means that **A** is defined at least on $\mathcal{D}_{\mathbb{A}}$. In both cases the value of f on $g \in \mathcal{D}_{\mathbb{A}}$ will be in $\mathcal{D}_{\mathbb{B}}$.

The system $\Sigma^r\Lambda^{\rightarrow}$ was developed in the authors dissertation [Koh94], while $\Sigma^d\Lambda^{\rightarrow}$ builds on [KP93] and [Kle96]. Our main results for $\Sigma\Lambda^{\rightarrow}$ are that sorted $\beta\eta$-reduction is terminating and confluent and moreover conserves sorts of formulae. The proof of the β-reduction results makes use of the fact that Σ-substitutions (well-sorted substitutions) also do. These results show that even though there is a strong interaction between sorted $\beta\eta$-conversion and term declarations (proofs of well-sortedness are totally independent of the structure of formulae) the system $\Sigma\Lambda^{\rightarrow}$ still satisfies the basic properties required for speaking of a λ-calculus.

7.1 Sorts

Sorting the universe of individuals gives rise to new classes of functions, whose domains and codomains are just the sorts. In addition to this essentially first-order way of sorting the function universes, the classes of functions defined by domains and codomains can be further divided into subclasses, since functions are explicit objects of type theory. Sorts of functional type, i.e. base sorts that denote classes of functions, are introduced. Syntactically each sort comes with a type, and — if it is of functional type — also with domain and codomain sorts.

Definition 7.1.1 (Sort System)
A **sort system** is a quadruple $(\mathcal{S}, \mathcal{BS}, \mathfrak{r}, \mathfrak{d})$, where

1. $\mathcal{BS} := \mathcal{BS}_{\mathcal{T}} := \{\mathcal{BS}_\alpha \mid \alpha \in \mathcal{T}\}$ is a typed collection of sets (of symbols), called **base sort**s, which we assume not to contain any types (we always want to be able to distinguish sorts and types).

2. the collection **sort**s $\mathcal{S}$ is the closure of $\mathcal{BS}$ under function construction, i.e. $\mathcal{S}$ is a typed collection that contains $\mathcal{BS}$ such that for any $\mathbb{A} \in \mathcal{S}_\alpha$ and $\mathbb{B} \in \mathcal{S}_\beta$, we have $\mathbb{A} \rightarrow \mathbb{B} \in \mathcal{S}_{\alpha\rightarrow\beta}$.

3. the **domain sort** function $\mathfrak{d}$ is a function $\mathfrak{d}\colon \mathcal{BS}_{\alpha\rightarrow\beta} \longrightarrow \mathcal{S}_\alpha$.

4. the **codomain sort** function $\mathfrak{r}$ is a function $\mathfrak{r}\colon \mathcal{BS}_{\alpha\rightarrow\beta} \longrightarrow \mathcal{S}_\beta$.

If the context is clear we will often denote the sort system $(\mathcal{S}, \mathcal{BS}, \mathfrak{d}, \mathfrak{r})$ only by $\mathcal{S}$.

BegNP(33)

[33] New Part:

Remark 7.1.2 (Finiteness and Sorts)
Since we are ultimately interested in computational properties of $\Sigma\Lambda^{\rightarrow}$, we will assume that $\mathcal{BS}$ is finite. As a consequence, for each type α, the set $\mathcal{S}_\alpha$ is finite ($\mathbf{ln}(\alpha)$ limits the number of base sorts that can appear in $\mathbb{A} \in \mathcal{S}_\alpha$).

Definition 7.1.3 (Sorts)
Let $\mathcal{S}$ be a sort system and $\mathbb{A} \in \mathcal{S}$. Remember that $\mathcal{S}$ is a typed collection (cf. 1.4.1) induces a type function. We call the type $\tau(\mathbb{A}) \in \mathcal{T}$ the **type** of the sort $\mathbb{A}$. If $\tau(\mathbb{A}) \notin \mathcal{BT}$, then we call $\mathbb{A}$ a **functional sort** ($\mathbb{A} \in \mathcal{S}^f$) and otherwise **non-functional** ($\mathbb{A} \in \mathcal{S}^{nf}$).

As in the typed case, the **length** of a sort is defined by the number of top level arrows: $\mathbf{ln}(\mathbb{A}) := 0$, iff $\mathbb{A} \in \mathcal{BS}$ and $\mathbf{ln}(\mathbb{A} \to \mathbb{B}) := 1 + \mathbf{ln}(\mathbb{B})$ otherwise. Intuitively, if $\mathbf{ln}(\mathbb{A}) = n$, then $\mathbb{A} = \overline{\mathbb{B}^n} \to \mathbb{C}$, where $\mathbb{C} \in \mathcal{BS}$ (we call $\mathbb{C}$ the **target sort**). Note that we can have $\mathbf{ln}(\mathbb{A}) \neq \mathbf{ln}(\tau(\mathbb{A}))$ due to the existence of functional base sorts.

Notation 7.1.4 We denote sorts with uppercase symbols like $\mathbb{A}$, $\mathbb{B}$, $\mathbb{C}$, or $\mathbb{D}$. For $\mathbb{C} := (\mathbb{A} \to \mathbb{B}) \in \mathcal{S}$ we define $\mathfrak{r}(\mathbb{C}) := \mathbb{B}$, $\mathfrak{d}(\mathbb{C}) := \mathbb{A}$, thus we can extend the functions $\mathfrak{r}$ and $\mathfrak{d}$ to $\mathcal{S}^f$. Furthermore, we use $\mathfrak{r}$ and $\mathfrak{d}$ on types in the obvious way.

We often use the shorthands $\mathfrak{d}^k(\mathbb{A})$ and $\mathfrak{r}^k(\mathbb{A})$ for the k^{th} **domain sort** and the k-fold **codomain sort** of $\mathbb{A}$, which we inductively define by

$$\begin{array}{rclcrcl} \mathfrak{r}^0(\mathbb{A}) & = & \mathbb{A} & \quad & \mathfrak{r}^{i+1}(\mathbb{A}) & = & \mathfrak{r}(\mathfrak{r}^i(\mathbb{A})) \\ \mathfrak{d}^0(\mathbb{A}) & = & \mathbb{A} & \quad & \mathfrak{d}^{i+1}(\mathbb{A}) & = & \mathfrak{d}(\mathfrak{r}^i(\mathbb{A})) \end{array}$$

Remark 7.1.5 By defining the sets of sorts as typed collections we have enforced that the sorts refine an existing type structure. We could have defined the sort system without reference to types by making the domain sort and codomain sort information part of the signature (cf. 7.2.7), but we prefer to have the type information as a useful intuition in the background and keep the definition of the sorted signatures comparatively simple.

Definition 7.1.6 (Trivially Sorted)
Since we are ultimately interested in sorted formulae and their typed counterparts, we only consider sort systems where τ is surjective. Thus we can pick a local inverse, i.e. an injection $\sharp\colon \mathcal{T} \longrightarrow \mathcal{S}$ with $\flat(\sharp(\alpha)) = \alpha$. We sometimes also denote τ by $\flat$, when we want to stress the property of being an inverse of $\sharp$.

We call a sort system **trivially sorted**, iff $\tau\colon \mathcal{BS} \longrightarrow \mathcal{BT}$ is a bijection. In this case there can be no functional base sorts, since all base sorts are of base type. Moreover the functions $\flat$ and $\sharp$ are bijections and inverses, so the sort system $\mathcal{S}$ is isomorphic to $\mathcal{T}$ and the functions $\mathfrak{d}$ and $\mathfrak{r}$ are trivial. If we only have one base type, then we call the sort system **one-sorted**.

$\mathbf{ln}(\mathbb{A})$*length of* $\mathbb{A}$
τ *type function*
$\mathcal{S}^f, \mathcal{S}^{nf}$ *set of functional/non-functional sorts*
type!of a sort
End NP(33)
functional sort
sort!(non)-functional
non-functional
length!of a sort
target sort
sort!target
$\mathbb{A}, \mathbb{B}, \mathbb{C}, \ldots$ *sorts*
$\mathfrak{d}^k(\mathbb{A})/\mathfrak{r}^k(\mathbb{A})$ n^{th} *domain/codomain sort of* $\mathbb{A}$
domain sort!k^{th}
domain sort!k-fold
$\sharp, \flat$ *correspondences of types with sorts*
trivially sorted
sorted!trivially/one
one-sorted

$[X::\mathbb{A}]$*variable declaration*
variable declaration
declaration!variable
context
$\vdash_{ctx}\Gamma\Gamma$ *is a variable context*
$[X^1,\ldots,X^n::\mathbb{A}]$*abbreviation for* $[X^1::\mathbb{A}],\ldots,[X^n::\mathbb{A}]$ *in variable contexts*
raw

7.2 Well-Sorted Formulae

Next we introduce the concept of well-sortedness for well-formed formulae. Just as well-typedness (cf. section 1.5) it is defined with respect to a variable context, which gives local sort information for the variables, and a signature, which contains sort information for formula schemata (term declarations). A formula $\mathbf{A}$ is called well-sorted with respect to a signature Σ and a context Γ, iff the judgment $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ is derivable in the inference system $\Sigma\Lambda^{\to}$.

One of the difficulties in devising a formal system with term declarations is that the signature, needed for defining well-sortedness, itself contains formulae that have to be well-sorted. We have another case of recursive dependency: on the one hand we need well-sortedness conditions on the inference rules for sorted $\beta\eta$-reduction, and on the other hand we need an inference rule $ws{:}\beta\eta$ that guarantees formulae that are $\beta\eta$-equivalent to have identical sets of sorts. Therefore we need to combine the inference systems for valid signatures, well-sortedness, and sorted $\beta\eta$-reduction into one large sort inference system $\Sigma\Lambda^{\to}$.

Definition 7.2.1 (Variable Context)
Let X_α be a variable and $\mathbb{A}$ a sort, then we call a pair $[X::\mathbb{A}]$ a **variable declaration** for X, iff $\tau(\mathbb{A}) = \alpha$. We call a typed, partial function from variables to sorts a (variable) **context**. Thus a variable context is just a set of variable declarations.

Notation 7.2.2 We write the fact that a typed partial function $\Gamma\colon \mathcal{V} \longrightarrow \mathcal{S}$ is a variable context as $\vdash_{ctx} \Gamma$, and we generally use the symbols Γ, Δ, and Ξ for variable contexts. With our convention from 1.2.12 we have $\Gamma(X) = \mathbb{A}$ for $\Gamma := \Gamma', [X::\mathbb{A}]$, even if $\Gamma'(X) = \mathbb{B}$. We sometimes abbreviate contexts $[X^1::\mathbb{A}],\ldots,[X^n::\mathbb{A}]$ by $[X^1,\ldots,X^n::\mathbb{A}]$ in order to conserve space and increase legibility.

Definition 7.2.3 (Raw $\Sigma\Lambda^{\to}$-formulae)
Raw $\Sigma\Lambda^{\to}$-formulae are well-typed $\Lambda^{\to}$-formulae, where λ-abstractions have the form $(\lambda X_{\mathbb{A}}.\mathbf{A})$ in $\Sigma^r\Lambda^{\to}$ and $(\lambda X_\alpha.\mathbf{A})$ in $\Sigma^d\Lambda^{\to}$. To make this definition formal, we extend the trivial injections $\sharp$ and $\flat$ (cf. 7.1.6) of sorts and types to formulae by

1. $\sharp(c) := \flat(c) := c$ for $c \in \Sigma$
2. $\sharp(X) := \flat(X) := X$ for $X \in \mathcal{V}$
3. $\sharp(\mathbf{AB}) := \sharp(\mathbf{A})\sharp(\mathbf{B})$ and $\flat(\mathbf{AB}) := \flat(\mathbf{A})\flat(\mathbf{B})$

4^r $\sharp(\lambda X_\alpha.\mathbf{A}) := \lambda X_{\sharp(\alpha)}.\sharp(\mathbf{A})$ and $\flat(\lambda X_{\mathbb{A}}.\mathbf{A}) := \lambda X_{\flat(\mathbb{A})}.\flat(\mathbf{A})$

4^d $\sharp(\lambda X_\alpha.\mathbf{A}) := \lambda X_\alpha.\sharp(\mathbf{A})$ and $\flat(\lambda X_\alpha.\mathbf{A}) := \lambda X_\alpha.\flat(\mathbf{A})$

These homomorphisms are syntactically[34] so trivial (they only add or delete sort information in abstractions), that we will often keep them implicit and directly consider $\Sigma\Lambda^{\rightarrow}$-formulae as $\Lambda^{\rightarrow}$-formulae. EdNote(34) *term declaration* *signature*

$\mathcal{T}$	$::=$	$\iota \mid o \mid \mathcal{T} \rightarrow \mathcal{T}'$	(Types: $\alpha, \beta, \ldots$)
$\mathcal{S}$	$::=$	$\mid \mathcal{S} \rightarrow \mathcal{S}' \mid \mathcal{S} \cap \mathcal{S}'$	(Sorts $\mathbb{A}, \mathbb{B}, \ldots$)
$\mathbf{M}$	$::=$	$x \mid c \mid \mathbf{MN} \mid \lambda X.\mathbf{M}$	(Formulae $\mathbf{A}, \mathbf{B}, \ldots$)
			(variables : $\mathrm{X}, \mathrm{Y}, \mathrm{Z}, \ldots$)
			(constants : $\mathrm{c}, \Pi^{\alpha}, \wedge, \neg, \ldots$)
Σ	$::=$	$\emptyset \mid \Sigma, [\mathbf{M}::\mathcal{S}]$	(Signature)
Γ	$::=$	$\emptyset \mid \Gamma, [X::\mathcal{S}]$	(Environment)

Figure 7.1: Syntax of $\Sigma\Lambda^{\rightarrow}$

Definition 7.2.4 (Term Declaration)
We call a pair $[\mathbf{A}::\mathbb{A}]$ consisting of a raw formula $\mathbf{A}$ and a sort $\mathbb{A}$ a **term declaration** and a finite set of term declarations a **signature**.

The idea of term declarations consists in the intuition, that there can be additional sort information within the structure of a formula, as the following example shows. Consider, for instance, the addition function, which we (semantically) would like to have the sort $\mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$, where $\mathbb{N}$ is the sort of natural numbers. If we also have a sort $\mathbb{E}$ for the evens, then we might want to specify that the expression $a+a$ is an even number, even if a is not. This information can be formalized by declaring the formula $X_{\mathbb{N}} + X$, to be of sort $\mathbb{E}$ (an even number) using a term declaration or equivalently that the identity $\lambda X.X$ has sort $\mathbb{N} \rightarrow \mathbb{E}$. Since it is simpler to deal with closed term declarations, we will use the second form in $\Sigma\Lambda^{\rightarrow}$; see the discussion in Remark 7.3.7 to see that we do not restrict the generality by this move.

Another kind of information, we might also want to specify in term declarations is that the addition function yields a even number when applied to two evens. Since there are two intuitions about functions encoded in the systems $\Sigma^{d}\Lambda^{\rightarrow}$ and $\Sigma^{r}\Lambda^{\rightarrow}$, there are two ways of encoding this information: in $\Sigma^{d}\Lambda^{\rightarrow}$, we can just add the term declaration $[+::\mathbb{E} \times \mathbb{E} \rightarrow \mathbb{E}]$. In $\Sigma^{r}\Lambda^{\rightarrow}$, we cannot do this, since formulae have unique domain sorts there. There we can use the expressivity of term declarations to explicitly declare that the restriction of the addition function to the even numbers has codomain in $\mathbb{E}$ by the declaration $[\lambda X_{\mathbb{E}}.\lambda Y_{\mathbb{E}}. + XY::\mathbb{E} \rightarrow \mathbb{E} \rightarrow \mathbb{E}]$.

Definition 7.2.5 (Sorted α-Conversion)
In $\Sigma^{d}\Lambda^{\rightarrow}$ we cannot simply take typed α-conversion, since this would not

[34] EdNote: Think about the semantics are they trivial as well? If they are not, integrate into semantics discussion.

*$\mathbb{A}$ **Rdom** $\mathbb{B}$ $\mathbb{A}$ and $\mathbb{B}$ have equal domains*
equal domains
domains!equal
$\mathbb{A} \parallel \mathbb{B}$ compatible sorts
compatible sorts
$\vdash_{sig} \Sigma$ Σ is a valid signature
$\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ in Σ formula $\mathbf{A}$ has sort $\mathbb{A}$ assuming Γ
$\Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B}$ in Σ $\mathbf{B}$ can be obtained from $\mathbf{A}$ by sorted $\beta\eta$-conversion assuming Γ
$wsf_{\mathbb{A}}(\Sigma, \Gamma)$ set of well-sorted formulae of sort $\mathbb{A}$

conserve well-sortedness. Consider, for instance, the formulae $\lambda X_{\mathbb{A}}.\mathbf{A}$ and $\lambda Y_{\mathbb{B}}.[Y/X]\mathbf{A}$, which are typed α-variants, if $\tau(\mathbb{A}) = \tau(\mathbb{B})$. They do not have the same sorts in $\Sigma\Lambda^{\rightarrow}$, thus a formula containing the first as a subformula would become ill-sorted, if it were to be replaced by the second. For a sorted α-conversion relation we define raw formulae $\lambda X_{\mathbb{A}}.\mathbf{A}$ and $\lambda Y_{\mathbb{A}}.[Y/X]\mathbf{A}$ to be alphabetic variants. Since bound variables are simply typed in $\Sigma^d\Lambda^{\rightarrow}$, we can use ordinary α-conversion there.

Just like in the system $\Lambda^{\rightarrow}$ in assumption 1.1.14, we consider sorted α-conversion as built into the system to avoid variable capture during instantiation. Thus we regard $\Sigma\Lambda^{\rightarrow}$-formulae as syntactically equal, iff they are sorted alphabetical variants. Note that this assumption can be justified with exactly the same argument as the one for assumption 1.1.14.

For $\Sigma^r\Lambda^{\rightarrow}$, it will be important that the signatures, over which our well-sorted formulae are built, "respect function domains", i.e. that for any formula $\mathbf{A}$ and any sorts $\mathbb{A}$ and $\mathbb{B}$ such that $\mathbf{A}$ has sort $\mathbb{A}$ and sort $\mathbb{B}$ at the same time, the domain sorts $\mathfrak{d}(\mathbb{A})$ and $\mathfrak{d}(\mathbb{B})$ are identical. The proof that signatures indeed satisfy this property (see theorem 7.5.1) depends on the fact that term declarations meet the sort condition of *ws:td* in definition 7.2.7 below. This sort condition is given in terms of the equivalence relation **Rdom** , which we now define.

Definition 7.2.6 (Compatibility of Sorts)
We say that sorts $\mathbb{A}$ and $\mathbb{B}$ have **equal domains** ($\mathbb{A}$ **Rdom** $\mathbb{B}$), if either $\mathbb{A}, \mathbb{B} \in \mathcal{S}^{nf}$ and $\tau(\mathbb{A}) = \tau(\mathbb{B})$, or $\mathfrak{r}(\mathbb{A})$ **Rdom** $\mathfrak{r}(\mathbb{B})$ and $\mathfrak{d}(\mathbb{A}) = \mathfrak{d}(\mathbb{B})$. Note that $\mathbb{A}$ **Rdom** $\mathbb{B}$, iff $\mathfrak{d}^i(\mathbb{A}) = \mathfrak{d}^i(\mathbb{B})$ for all $1 \leq i \leq k$ such that $\mathfrak{r}^k(\mathbb{A})$ and $\mathfrak{r}^k(\mathbb{B})$ are of the same base type. Thus **Rdom** is an equivalence relation that respects types, i.e. $\mathbb{A}$ **Rdom** $\mathbb{B}$ can only hold, if $\tau(\mathbb{A}) = \tau(\mathbb{B})$. For trivially sorted sort systems **Rdom** is just equality.

In $\Sigma^d\Lambda^{\rightarrow}$, the compatibility conditions are less strict, here we only have to insist that all sorts of a formula have the same type. In order to unify notation, we write $\mathbb{A} \parallel \mathbb{B}$ ($\mathbb{A}$ is **compatible**) with $\mathbb{B}$, iff $\mathbb{A}$ **Rdom** $\mathbb{B}$ for $\Sigma^r\Lambda^{\rightarrow}$ and iff $\tau(\mathbb{A}) = \tau(\mathbb{B})$ for $\Sigma^d\Lambda^{\rightarrow}$ and we will use this general compatibility relation in the definitions below.

$\vdash_{sig} \Sigma$	Σ is a valid signature
$\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$	in Σ formula $\mathbf{A}$ has sort $\mathbb{A}$ assuming Γ
$\Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B}$	$\mathbf{B}$ can be obtained from $\mathbf{A}$ by sorted $\beta\eta$-conversion in Σ assuming Γ

Figure 7.2: Judgments in $\Sigma\Lambda^{\rightarrow}$

well-sorted formula
formula!well-sorted
valid!signature
signature!valid
of sort
sort!of
well-sorted
ill-sorted

Definition 7.2.7 (Inference System for $\Sigma\Lambda^{\rightarrow}$)
We define **well-sorted formulae** by the inference system $\Sigma\Lambda^{\rightarrow}$ for the judgments in Figure 7.2 given by the inference rules in Figures 7.3 to 7.5.

We say that a signature Σ is **valid**, iff there is a $\Sigma\Lambda^{\rightarrow}$-derivation of $\vdash_{sig} \Sigma$. For a fixed signature Σ and a context Γ we say that a formula $\mathbf{A}$ is **of sort** $\mathbb{A}$, iff $\Gamma \vdash_{\Sigma} \mathbf{A}{::}\mathbb{A}$, and that $\mathbf{A}$ is **well-sorted**, iff there is a sort $\mathbb{A}$ such that $\mathbf{A}$ has sort $\mathbb{A}$, otherwise we call $\mathbf{A}$ **ill-sorted** in Γ and Σ. We fix the notation $wsf_{\mathbb{A}}(\Sigma, \Gamma)$ for the set of formulae of sort $\mathbb{A}$.

$$\frac{\vdash_{sig} \Sigma \quad \vdash_{ctx} \Gamma \quad \Gamma(X) = \mathbb{A}}{\Gamma \vdash_{\Sigma} X{::}\mathbb{A}}\, ws{:}var \qquad \frac{[\mathbf{A}{::}\mathbb{A}] \in \Sigma \quad \vdash_{sig} \Sigma \quad \vdash_{ctx} \Gamma}{\Gamma \vdash_{\Sigma} \mathbf{A}{::}\mathbb{A}}\, ws{:}td$$

$$\frac{\Gamma \vdash_{\Sigma} \mathbf{A}{::}\mathbb{A} \quad \Gamma \vdash_{\Sigma} \mathbf{B}{::}\mathfrak{d}(\mathbb{A})}{\Gamma \vdash_{\Sigma} (\mathbf{A}\mathbf{B}){::}\mathfrak{r}(\mathbb{A})}\, ws{:}app$$

$$\frac{\Gamma, [X{::}\mathbb{B}] \vdash_{\Sigma} \mathbf{A}{::}\mathbb{A}}{\Gamma \vdash_{\Sigma} (\lambda X_{\mathbb{B}}.\mathbf{A}){::}\mathbb{B} \to \mathbb{A}}\, ws^{r}{:}abs \qquad \frac{\Gamma, [X{::}\mathbb{B}] \vdash_{\Sigma} \mathbf{A}{::}\mathbb{A} \quad \tau(\mathbb{B}) = \beta}{\Gamma \vdash_{\Sigma} (\lambda X_{\beta}.\mathbf{A}){::}\mathbb{B} \to \mathbb{A}}\, ws^{d}{:}abs$$

$$\frac{\Gamma \vdash_{\Sigma} \mathbf{A}{::}\mathbb{A} \quad \Gamma \vdash_{\Sigma} \mathbf{A} =_{\beta\eta} \mathbf{B}}{\Gamma \vdash_{\Sigma} \mathbf{B}{::}\mathbb{A}}\, ws{:}\beta\eta$$

Figure 7.3: Well-sorted Formulae

Let us now discuss the sorting rules in turn. The main set of rules is that for the sort judgment in figure Figure 7.3. The rules for variables, application, and abstraction are the obvious generalizations of the corresponding rules for $\Lambda^{\rightarrow}$ in Figure 1.8 on page lvii. In the setting with term declarations we do not need a separate rule for constants, since all constants have to be declared in term declarations. The rules for abstractions are the only point, where the two systems $\Sigma^{r}\Lambda^{\rightarrow}$ and $\Sigma^{d}\Lambda^{\rightarrow}$ differ ($\Sigma^{r}\Lambda^{\rightarrow}$ uses $ws^{r}{:}abs$ while $\Sigma^{d}\Lambda^{\rightarrow}$ needs $ws^{d}{:}abs$), but the rules show the fundamentally different paradigms behind the two systems $ws^{r}{:}abs$ prescribes (by explicitly marking the bound variables) the domain sort of a function, while $ws^{d}{:}abs$ infers all possible sorts of the bound variable, thus determining all possible domain sorts of the function. In effect $\Sigma^{r}\Lambda^{\rightarrow}$ has the underlying semantic assumption that functions carry their domain around with them (a view supported by the set-theoretic definition of a function as a particular kind of relation (cf. 1.2.12)). $\Sigma^{d}\Lambda^{\rightarrow}$ views functions more like programs that can be applied, wherever they yield meaningful results (as a consequence $\Sigma^{d}\Lambda^{\rightarrow}$ uses sorts to describe the input-output behavior of λ-defined functions).

Despite their differing motivations, the rules are quite similar in structure, both discharging a variable from the context to the binder, therefore we will often be able to conduct arguments about both rules simultaneously, when we are not interested in the sort of the bound variable. We will then use the rule specifier *ws:abs*.

The rules for well-formed signatures are given in Figure 7.4; new declarations can be added to signatures for new constants (*sig:const*), or by term declarations (*sig:td*) specifying additional sort information $\mathbb{B}$ for a well-sorted formulae $\mathbf{B}$, if the new sort is compatible with the old sort information.

$$\frac{\vdash_{sig} \Sigma \quad c \notin \Sigma \quad \mathbb{A} \in \mathcal{S}_\alpha}{\vdash_{sig} \Sigma, [c_\alpha :: \mathbb{A}]}\ sig{:}const$$

$$\frac{}{\vdash_{sig} \emptyset}\ sig{:}empty \qquad \frac{\vdash_\Sigma \mathbf{A}::\mathbb{A} \quad \mathbb{A} \parallel \mathbb{B}}{\vdash_{sig} \Sigma, [\mathbf{A}::\mathbb{B}]}\ sig{:}td$$

Figure 7.4: Signature Rules

Note that in this rule we do not have to require $\vdash_{sig} \Sigma$, since $\mathbf{A}$ can only be proven to be well-sorted, if Σ is valid. The previous definition needs the judgment of well-sorted formulae, which we define with the inference rules from Figure 7.3.

Now we define the judgment $\Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B}$ for sorted $\beta\eta$-equality with the inference rules in Figure 7.5.

The rules in Figure 7.5 are sorted variants of the typed rules from Figures 1.10 and 1.9. We need them, since we view $\beta\eta$-conversion as basic to our system, therefore we do not want $\beta\eta$-conversion to increase the sort of a subformula, and thereby possibly convert a well-sorted formula to an ill-sorted one.

In the definition of sorted η-reduction we have taken care to identify the (unique) supporting sort $\mathfrak{d}(\mathbb{A})$ of $\mathbf{A}$ for $\Sigma^r\Lambda^\rightarrow$, since the formula $\lambda X_{\mathbb{B}}.\mathbf{A}X$ denotes the restriction of the function $\mathbf{A}$ to sort $\mathbb{B}$, if $\mathbb{B}$ is a subsort of $\mathfrak{d}(\mathbb{A})$. Since we need the sort information there, we had to extend all rules by appropriate sort judgments. The multistep rules differ slightly from those in Figure 1.11 for technical convenience, but the typed variants of the sorted rules are equivalent to those in Figure 1.11.

The following lemma is useful for carrying out proofs by induction over the structure of $\Sigma\Lambda^\rightarrow$-derivations.

Lemma 7.2.8 *If $\mathcal{A}{:}\,\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ ends in a ws:td-node, then either*

1. *$\mathbf{A}$ is a constant or*
2. *there is a signature $\Sigma' \subseteq \Sigma$ and a strict subderivation $\mathcal{B}{:}\vdash_{\Sigma'} \mathbf{A}::\mathbb{B}$ of $\mathcal{A}$ for some $\mathbb{B}$ with $\mathbb{B} \parallel \mathbb{A}$.*

Proof: Since $\mathcal{A}$ ends in *ws:td*, we know that there is a term declaration $[\mathbf{A}::\mathbb{A}] \in \Sigma$ and a $\Sigma\Lambda^\rightarrow$-derivation $\mathcal{D}{:}\vdash_{sig} \Sigma$. We show the assertion by induction on the structure of $\mathcal{D}$. Σ is nonempty, so *sig:empty* cannot apply. If $\mathcal{A}$ ends in *sig:const*, then $\mathbf{A}$ is a constant and we have obtained the first case. If $\mathcal{D}$ ends in *sig:td*, then we have the following situation

$$\frac{\dfrac{\mathcal{D}'}{\vdash_{\Sigma'} \mathbf{B}::\mathbb{C}} \quad \mathbb{C} \parallel \mathbb{B}}{\vdash_{sig} \Sigma', [\mathbf{B}::\mathbb{B}]}\ \mathit{sig{:}td}$$

Thus we have to consider two cases: if $\mathbf{A} = \mathbf{B}$, and $\mathbb{A} = \mathbb{B}$, then $\mathcal{D}': \vdash_\Sigma \mathbf{A}::\mathbb{C}$ and $\mathbb{C} \parallel \mathbb{B} = \mathbb{A}$. If this is not the case, we obtain the assertion by inductive hypothesis for $\mathcal{D}'$. □

The next lemma convinces us that the judgments defined above respect well-formedness, i.e. that the information described by $\Sigma\Lambda^{\rightarrow}$ merely refines the type information.

Lemma 7.2.9 *Let Σ be a valid signature.*

1. *If $\Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B}$, then $\flat(\mathbf{A}) =_{\beta\eta} \flat(\mathbf{B})$.*
2. *If $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$, then $\tau(\mathbf{A}) = \tau(\mathbb{A})$.*
3. *If $\vdash_{sig} \Sigma$ and $[\mathbf{A}::\mathbb{A}] \in \Sigma$, then $\tau(\mathbf{A}) = \tau(\mathbb{A})$.*

Proof: We prove the assertions by a simultaneous induction over the structure of the $\Sigma\Lambda^{\rightarrow}$-proofs for the judgments involved. The only interesting cases for the first assertion are *ws:β:top*, ws^r*:η:top* and ws^d*:η:top*, where we can read off the assertions from the inference rules.

For the second assertion we consider the cases for the last step in the $\Sigma\Lambda^{\rightarrow}$-derivation $\mathcal{D}: \Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$. If $\mathcal{D}$ ends in *ws:var*, then $\mathbf{A}$ is a variable of some type α and we have $\tau(\mathbf{A}) = \alpha$ by definition. In the cases where $\mathcal{D}$ ends in *ws:td* or *ws:βη* we obtain the assertions by 3. and 1. If $\mathcal{D}$ ends in *ws:app*, then $\mathbf{A} \equiv \mathbf{C}\mathbf{D}$, and we have $\Sigma\Lambda^{\rightarrow}$-proofs for $\Gamma \vdash_\Sigma \mathbf{C}::\mathbb{C}$ and $\Gamma \vdash_\Sigma \mathbf{D}::\mathfrak{d}(\mathbb{C})$, where $\mathbb{A} = \mathfrak{r}(\mathbb{C})$. By inductive hypothesis we have $\tau(\mathbf{C}) = \tau(\mathbb{C})$ and $\tau(\mathbf{D}) = \tau(\mathfrak{d}(\mathbb{C}))$, and therefore $\tau(\mathbf{A}) = \tau(\mathfrak{r}(\mathbb{C}))$, since $\mathfrak{r}(\tau(\mathbb{C})) = \tau(\mathfrak{r}(\mathbb{C}))$ and $\tau(\mathfrak{d}(\mathbb{C})) = \mathfrak{d}(\tau(\mathbb{C}))$. Finally, we get the assertion for the remaining case, where $\mathcal{D}$ ends in ws^r*:abs* or ws^d*:abs*, with a similar application of the inductive hypothesis.

The only interesting cases for the third assertion are the inductive rules *sig:const* and *sig:td*. While we obtain the assertion for the former by construction, the well-typedness for the latter relies on the fact that $\parallel$ is a typed binary relation. □

Remark 7.2.10 (Non-Structural)
Inspection of the inference rules above shows that the inference system

[$\mathbf{A}$|:$\mathbb{A}$]*term declaration for restriction information function!restriction restriction*

$\Sigma\Lambda^{\rightarrow}$ for sort inference is non-structural (cf. 1.5.7), since the succedent of *ws*:$\beta\eta$ is not restricted to any structural category and furthermore, sorted $\beta\eta$-conversion can dramatically change the structure of a formula. Thus it is not obvious how to construct a sort inference algorithm from this inference system. We will later recover some structural properties (cf. 8.1.4) of $\Sigma\Lambda^{\rightarrow}$-derivations and use these for sort computation in theorem (8.3.4).

Remark 7.2.11 At first sight the restriction of $\mathbb{A}$ **Rdom** $\mathbb{B}$ in *sig*:*td* appears to be a grave restriction on the expressiveness of term declarations in $\Sigma^r\Lambda^{\rightarrow}$, since it severely restricts the overloading of function constants. In particular, it is impossible to combine declarations like $[+::\mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}]$, *and* $[+::\mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}]$ declaring the addition function to be a function of naturals and of reals in one signature, since one of them would have to be added to the signature with the *sig*:*td* rule and we cannot have $\mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$ **Rdom** $\mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}$, since $\mathbb{R} \neq \mathbb{N}$.

However, the term declarations can be used to formalize the fact that the restriction of the addition function to the naturals ranges over the naturals in $[+::\mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}], [\lambda X_{\mathbb{N}}.\lambda Y_{\mathbb{N}}. + XY::\mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}]$, which is legal in $\Sigma^r\Lambda^{\rightarrow}$ (if $\lambda X.X::\mathbb{R} \rightarrow \mathbb{N}$, or in other words $\mathbb{N}$ is a subsort of $\mathbb{R}$, which we assume to make this example meaningful). This way the formula $+$ is of sort $\mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}$ with domain in the reals whereas the formula $[\lambda X_{\mathbb{N}}, Y_{\mathbb{N}}. + XY]$ is of sort $\mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$ and has domain in the naturals.

The notion of a unique supporting sort corresponds to the intuition in mathematics that functions come with a unique (maximal) domain and have to be distinguished from restrictions to subdomains.

Notation 7.2.12 (Term Declarations for Restriction)
To conserve space and increase legibility we abbreviate term declarations of the form $[\lambda\overline{X^n}.\mathbf{A}\overline{X^n}::\overline{\mathbb{A}^n} \rightarrow \mathbb{B}]$ with $[\mathbf{A}|:\mathbb{A}^1 \rightarrow \cdots \rightarrow \mathbb{A}^n \rightarrow \mathbb{B}]$ in $\Sigma^r\Lambda^{\rightarrow}$, if $X^1, \ldots, X^n \notin \mathbf{Free}(\mathbf{A})$. Such a declaration specifies that the denotation of the functional formula $\mathbf{A}$ when restricted to $\mathcal{D}_{\mathbb{A}^1} \times \cdots \times \mathcal{D}_{\mathbb{A}^n}$ has values in $\mathcal{D}_{\mathbb{B}}$.

With this convention the declaration in remark 7.2.11 would look like $[+::\mathbb{R} \rightarrow \mathbb{R} \rightarrow \mathbb{R}]$ and $[+|:\mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}]$, which is much easier to read.

The following lemma allows to enlarge signatures and variable contexts in judgments. We will often use it in the proofs to follow without explicitly stating it.

Lemma 7.2.13 (Monotonicity)
Let $\Sigma \subset \Sigma'$ be valid signatures, and let $\Gamma \subseteq \Gamma'$ be variable contexts, then

1. $\Gamma \vdash_{\Sigma} \mathbf{A}::\mathbb{A}$ *entails* $\Gamma \vdash_{\Sigma'} \mathbf{A}::\mathbb{A}$ *and* $\Gamma' \vdash_{\Sigma} \mathbf{A}::\mathbb{A}$

2. $\Gamma \vdash_{\Sigma} \mathbf{A}=_{\beta\eta}\mathbf{B}$ *entails* $\Gamma \vdash_{\Sigma'} \mathbf{A}=_{\beta\eta}\mathbf{B}$ *and* $\Gamma' \vdash_{\Sigma} \mathbf{A}=_{\beta\eta}\mathbf{B}$

Moreover, the derivations involved only differ in the variable contexts they mention and the subderivations for the $\vdash_{sig}$ judgment.

frugal
$\mathbf{wsSub}(\Sigma, \Delta \to \Gamma), WsSuballran\Delta$*sets of Σ-substitutions*
wsub:start, wsub:extinference rules for Σ-substitutions
$\Gamma \vdash_\Sigma \sigma{::}\Delta\sigma$*-is a Σ-substitution*
substitution!Σ
renaming!Σ

Proof: The assertions can be proven by a simple joint induction on the $\Sigma\Lambda^{\to}$-derivations involved. □

We will call a $\Sigma\Lambda^{\to}$-judgment **frugal**, iff the variable context involved only mentions the free variables of the formulae involved. The next Lemma says that we can always obtain frugality in $\Sigma\Lambda^{\to}$.

Lemma 7.2.14 (Frugal)
Let $\mathcal{A}\colon \Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ *and* $\mathcal{E}\colon \Gamma \vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{B}$, *and* $\Gamma' := \Gamma|_{\mathbf{Free}(\mathbf{A})\cup\mathbf{Free}(\mathbf{B})}$, *then there are* $\Sigma\Lambda^{\to}$*-derivations* $\mathcal{A}'\colon \Gamma' \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ *and* $\mathcal{E}'\colon \Gamma' \vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{B}$ *that only differ from* $\mathcal{A}$ *and* $\mathbf{E}$ *in the variable contexts they mention.*

Proof: We prove the assertion by joint induction on the structure of $\mathcal{A}$ and $\mathcal{E}$ starting with $\mathcal{A}$. The *ws:var* and *ws:td* cases are trivial by construction.

For *ws:app* we have $\mathbf{A} = \mathbf{CD}$ and subderivations $\mathcal{C}\colon \Gamma \vdash_\Sigma \mathbf{C}{::}\mathbb{C}$ and $\mathcal{D}\colon \Gamma \vdash_\Sigma \mathbf{D}{::}\mathfrak{d}(\mathbb{C})$, where $\mathbb{A} = \mathfrak{r}(\mathbb{D})$. By inductive hypothesis we have $\Sigma\Lambda^{\to}$-derivations $\mathcal{C}'\colon \Delta \vdash_\Sigma \mathbf{C}{::}\mathbb{C}$ and $\mathcal{D}'\colon \Delta' \vdash_\Sigma \mathbf{D}{::}\mathfrak{d}(\mathbb{C})$, and see that $\Gamma' = \Delta \cup \Delta'$ (since $\mathbf{Dom}(\Gamma') = \mathbf{Dom}(\Delta) \cup \mathbf{Dom}(\Delta') = \mathbf{Free}(\mathbf{C}) \cup \mathbf{Free}(\mathbf{D}) = \mathbf{Free}(\mathbf{A})$). By monotonicity, we also have $\mathcal{C}''\colon \Gamma' \vdash_\Sigma \mathbf{C}{::}\mathbb{C}$ and $\mathcal{D}''\colon \Gamma' \vdash_\Sigma \mathbf{D}{::}\mathfrak{d}(\mathbb{C})$, and obtain $\mathcal{A}'$ by a single application of *ws:app*.

For *ws:abs*, we have $\mathbf{A} = \lambda X.\mathbf{C}$ and $\mathcal{C}\colon \Gamma, [X{::}\mathbb{D}] \vdash_\Sigma \mathbf{C}{::}\mathbb{C}$, where $\mathbb{A} = \mathbb{C} \to \mathbb{D}$. If $X \in \mathbf{Free}(\mathbf{C})$, then let $\Gamma'' := \Gamma', [X{::}\mathbb{A}]$, else $\Gamma'' := \Gamma'$. In both cases $\Gamma'' = \Gamma|_{\mathbf{Free}(\mathbf{C})}$, so by inductive hypothesis we have $\Gamma'' \vdash_\Sigma \mathbf{C}{::}\mathbb{C}$. By applying monotonicity if $X \notin \mathbf{Free}(\mathbf{C})$, we obtain $\Gamma', [X{::}\mathbb{D}] \vdash_\Sigma \mathbf{C}{::}\mathbb{C}$ and thus $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ by *ws:abs*.

The *ws:βη* case is a simple consequence of the two inductive hypotheses, so we only have to show the assertion for the $\beta\eta$-equality judgment. These are largely trivial consequences of the inductive hypotheses, since they do not change the contexts involved. The only one is *ws:β:top*, where we can argue just as in the *ws:abs* case. □

7.3 Σ-Substitutions

In this section we will study well-sorted instantiation and substitutions.

Definition 7.3.1 (Σ-Substitution)
Let Γ and Δ be variable contexts, then we call a substitution σ a Σ-**substitution** with domain context Δ and codomain context Γ, iff the judgment $\Gamma \vdash_\Sigma \sigma{::}\Delta$ is derivable in the inference system given in Figure 7.6. The set of Σ-substitutions with domain context Δ and codomain context Γ is denoted by $\mathbf{wsSub}(\Sigma, \Delta \to \Gamma)$. A Σ-substitution $\sigma \in \mathbf{wsSub}(\Sigma, \Delta \to \Gamma)$ is called a Σ-**renaming**, if it is a sort-preserving renaming substitution, that

is, if $\sigma(X) = Y$ such that $\Delta(X) = \Gamma(Y)$. Sometimes we want to speak about the set of all substitutions that have domain context Δ, therefore we fix the notation $\mathbf{wsSub}(\Sigma,\Delta) := \bigcup_{\vdash_{ctx}\Gamma} \mathbf{wsSub}(\Sigma,\Delta\to\Gamma)$.

Lemma 7.3.2 *Let $\mathcal{A}\colon\Gamma\vdash_\Sigma \sigma{::}\Delta$, then we have*

1. $\mathbf{Dom}(\Delta) = \mathbf{Dom}(\sigma)$, $\mathbf{Intro}(\sigma) \subseteq \mathbf{Dom}(\Gamma)$, *and* $\mathbf{Dom}(\Delta)\cap\mathbf{Dom}(\Gamma) = \emptyset$, *so in particular, σ is idempotent (cf. 1.1.11).*

2. *If $\sigma = \sigma', [\mathbf{A}/X]$, then $\Delta = \Delta', [X{::}\mathbb{A}]$, and there are context $\Gamma', \Gamma'' \subseteq \Gamma$ and proper sub-derivations $\mathcal{A}'\colon\Gamma'\vdash_\Sigma \sigma'{::}\Delta'$ and $\mathcal{A}''\colon\Gamma''\vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ of $\mathcal{A}$.*

Proof: Immediate from the definitions. □

We now want to show that Σ-instantiation, i.e. application of Σ-substitutions, preserves sorts. Before we tackle the main theorem we will establish an intermediate result.

Lemma 7.3.3 *Let $\Delta\vdash_\Sigma \mathbf{B}{::}\mathbb{B}$ and $\Gamma||\Delta$, then we have*

1. *If $\Gamma,[X{::}\mathbb{B}]\vdash_\Sigma \mathbf{A}{::}\mathbb{A}$, then $\Gamma,\Delta\vdash_\Sigma [\mathbf{B}/X]\mathbf{A}{::}\mathbb{A}$.*

2. *If $\Gamma,[X{::}\mathbb{B}]\vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{C}$, then $\Gamma,\Delta\vdash_\Sigma [\mathbf{B}/X]\mathbf{A}{=}_{\beta\eta}[\mathbf{B}/X]\mathbf{C}$.*

Proof: We prove the assertions by a joint induction over the depth n of the $\Sigma\Lambda^{\to}$ derivations $\mathcal{A}\colon\Gamma,[X{::}\mathbb{B}]\vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ and $\mathcal{C}\colon\Gamma,[X{::}\mathbb{B}]\vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{C}$.

The case $n = 0$ only applies to the first assertion, since sorted $\beta\eta$-equality cannot be verified with the empty derivation since it always contains sort preconditions. If $\mathcal{D}$ ends in *ws:td*, then $\mathbf{A}$ is closed, so $[\mathbf{B}/X]\mathbf{A} = \mathbf{A}$, and $\Gamma,\Delta\vdash_\Sigma [\mathbf{B}/X]\mathbf{A}{::}\mathbb{A}$ by monotonicity (7.2.13). In the *ws:var* case we have $\mathbb{A} = \mathbb{B}$ by construction and assume that $\mathbf{A} = X$, since otherwise $[\mathbf{B}/X]\mathbf{A} = \mathbf{A}$ and the argument above applies. So we have $\Delta\vdash_\Sigma \mathbf{B} = [\mathbf{B}/X]\mathbf{A}{::}\mathbb{B} = \mathbb{A}$ by assumption and $\Gamma,\Delta\vdash_\Sigma [\mathbf{B}/X]\mathbf{A}{::}\mathbb{A}$ by monotonicity. This completes the base case for $n = 0$.

Now let $n > 0$ and the assertions hold for all $m < n$. We proceed by analyzing $\mathcal{A}$ for the first assertion. In the *ws:abs* cases we have $\mathbb{A} = \mathbb{C}\to\mathbb{D}$ and $\Gamma,[X{::}\mathbb{B}],[Y{::}\mathbb{C}]\vdash_\Sigma \mathbf{D}{::}\mathbb{D}$ by construction and thus $\Gamma,\Delta,[Y{::}\mathbb{C}]\vdash_\Sigma [\mathbf{B}/X]\mathbf{D}{::}\mathbb{D}$ by inductive hypothesis. Thus we obtain the assertion by definition of substitution application and *ws:abs*. The *ws:app* case is proven similarly by a direct appeal to the inductive hypothesis.

In the *ws:$\beta\eta$* case we have $\Gamma,[X{::}\mathbb{B}]\vdash_\Sigma \mathbf{D}{::}\mathbb{A}$ and $\Gamma,[X{::}\mathbb{B}]\vdash_\Sigma \mathbf{D}{=}_{\beta\eta}\mathbf{C}$ for some $\mathbf{D}\in \mathit{wff}(\Sigma)$ by construction. Thus we can appeal to both inductive hypotheses to obtain $\Gamma,\Delta\vdash_\Sigma [\mathbf{B}/X]\mathbf{D}{::}\mathbb{A}$. $\Gamma,\Delta\vdash_\Sigma [\mathbf{B}/X]\mathbf{D}{=}_{\beta\eta}[\mathbf{B}/X]\mathbf{C}$, which gives the first assertion by a single application of *ws:$\beta\eta$*.

For the second assertion we will analyze the structure of $\mathcal{C}$. If $\mathcal{C}$ ends in *ws:β:top*, then we have

$$\frac{\Gamma, [X::\mathbb{B}], [Y::\mathbb{D}] \vdash_\Sigma \mathbf{C}::\mathbb{C} \quad \Gamma, [X::\mathbb{B}] \vdash_\Sigma \mathbf{D}::\mathbb{D}}{\Gamma \vdash_\Sigma (\lambda Y.\mathbf{C})\mathbf{D} \rightarrow^t_\beta [\mathbf{D}/Y]\mathbf{C}}$$

By applying the inductive hypotheses we obtain $\Gamma, \Delta, [Y::\mathbb{D}] \vdash_\Sigma [\mathbf{B}/X]\mathbf{C}::\mathbb{C}$ and $\Gamma, \Delta \vdash_\Sigma [\mathbf{B}/X]\mathbf{D}::\mathbb{D}$, which yields the assertion. The *ws:η:top* rules are shown similarly, likewise the other cases of $\mathcal{C}$. □

Theorem 7.3.4 *If $\Xi, \Delta \vdash_\Sigma \mathbf{A}::\mathbb{A}$ and $\Gamma \vdash_\Sigma \sigma::\Delta$, then $\Xi, \Gamma \vdash_\Sigma \sigma(\mathbf{A})::\mathbb{A}$.*

Proof: We prove the assertion by induction on $\mathcal{D}: \Gamma \vdash_\Sigma \sigma::\Delta$ that $\Xi, \Gamma \vdash_\Sigma \sigma(\mathbf{A})::\mathbb{A}$. If $\mathcal{D}$ ends in *wsub:start*, then σ is the empty substitution and the assertion is trivial. If $\mathcal{D}$ ends in *wsub:ext*, then $\sigma = \sigma', [\mathbf{B}/X]$ and $\Delta = \Delta', [X::\mathbb{B}]$. Furthermore, we have $\Gamma' \vdash_\Sigma \mathbf{B}::\mathbb{B}$ and $\mathcal{D}': \Gamma' \vdash_\Sigma \sigma'::\Delta'$ for contexts $\Gamma', \Gamma'' \subseteq \Gamma$ by 7.3.2.2. By the previous theorem 7.3.3 we have $\Xi, \Gamma \vdash_\Sigma [\mathbf{B}/X]\mathbf{A}::\mathbb{A}$. Now the inductive hypothesis gives us the assertion, since $\sigma(\mathbf{A}) = \sigma'([\mathbf{B}/X]\mathbf{A})$, as σ is a Σ-substitution and consequently idempotent. □

Corollary 7.3.5 *The following assertions are direct consequences of 7.3.4.*

1. *If $\sigma \in \mathbf{wsSub}(\Sigma, \Delta \rightarrow \Gamma)$, then $\sigma(\mathit{wsf}_{\mathbb{A}}(\Sigma; \Xi, \Delta)) \subseteq \mathit{wsf}_{\mathbb{A}}(\Sigma; \Xi, \Gamma)$.*
2. *The composition of two Σ-substitutions is again a Σ-substitution when defined.*
3. *$\Xi, \Delta \vdash_\Sigma \mathbf{C}{=}_{\beta\eta}\mathbf{D}$ and $\Gamma \vdash_\Sigma \sigma::\Delta$ imply $\Xi, \Gamma \vdash_\Sigma \sigma(\mathbf{C}){=}_{\beta\eta}\sigma(\mathbf{D})$.*

These considerations can also be used to give new, convenient inference rules, that are admissible in $\Sigma\Lambda^{\rightarrow}$. Thus we will freely use these rules in $\Sigma\Lambda^{\rightarrow}$-derivations, without changing our system. In particular, we only have to consider the original inference system from 7.2.7 in our meta-logical considerations.

Definition 7.3.6 (Σ-Instantiation)
We define two derived inference rules that will become convenient in the following.

$$\frac{\Delta, [X::\mathbb{B}] \vdash_\Sigma \mathbf{A}::\mathbb{A} \quad \Gamma \vdash_\Sigma \mathbf{B}::\mathbb{B} \quad \Gamma || \Delta \quad X \notin \mathbf{Dom}(\Gamma)}{\Delta, \Gamma \vdash_\Sigma [\mathbf{B}/X]\mathbf{A}::\mathbb{A}}\ \mathit{ws{:}inst}$$

$$\frac{\Delta, \Xi \vdash_\Sigma \mathbf{A}::\mathbb{A} \quad \Gamma \vdash_\Sigma \sigma::\Xi \quad \Gamma || \Delta}{\Delta, \Gamma \vdash_\Sigma \sigma(\mathbf{A})::\mathbb{A}}\ \mathit{ws{:}subst}$$

Note that *ws:inst* is only a special case of *ws:subst*.

Remark 7.3.7 (Open Term declarations) It is easy to see that $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ iff $\vdash_\Sigma \lambda\overline{X^n}.\mathbf{A}::\overline{\mathbb{B}_n} \to \mathbb{A}$, where $\Gamma = [X^1::\mathbb{A}_1], \ldots, [X^n::\mathbb{A}_n]$. Indeed a proof of the former judgment can be transformed into one for the latter by n applications of *ws:abs*, and vice versa by n applications to variables X^i and a subsequent application of *ws:$\beta\eta$*. Thus for any term declaration $[\lambda\overline{X^n}.\mathbf{A}::\overline{\mathbb{B}_n} \to \mathbb{A}] \in \Sigma$, we could have used a term declaration $[\forall\Gamma.\mathbf{A}::\mathbb{A}]$, where $\Gamma = [X^1::\mathbb{A}_1], \ldots, [X^n::\mathbb{A}_n]$ together with the following alternative rule term declarations

$$\frac{[\forall\Gamma.\mathbf{A}::\mathbb{A}] \quad \Gamma \subseteq \Gamma'}{\Gamma' \vdash_\Sigma \mathbf{A}::\mathbb{A}}$$

It is easy to show that if Σ' a signature, such that all term declarations in Σ' are written this form, then $\mathit{wff}_\alpha(\Sigma;\Gamma) = \mathit{wff}_\alpha(\Sigma';\Gamma)$. Therefore we do not lose generality by assuming our term declarations to be closed.

On first sight the idea of open declarations is closer to the intuition that term declarations specify schematic sort information and in particular, it would seem that *ws:inst* inference rule is more appropriate instead of the much more powerful (and problematic) *ws:$\beta\eta$*-rule. Consider, for instance, the following signature:

$$\Sigma := \{[+::\mathbb{N} \to \mathbb{N} \to \mathbb{N}], [\forall[X::\mathbb{N}]. + XX::\mathbb{E}], [1::\mathbb{N}]\}$$

Using the *ws:inst* rule alone, we can derive $\Gamma \vdash_\Sigma (+11)::\mathbb{E}$, whereas in the signature that uses the closed declaration $[(\lambda X_\mathbb{N}. + XX)::\mathbb{N} \to \mathbb{E}]$ can only derive $\Gamma \vdash_\Sigma (\lambda X_\mathbb{N}. + XX)1::\mathbb{E}$ but not $\Gamma \vdash_\Sigma (+11)::\mathbb{E}$ (with severe consequences for sorted reduction). But in the signature

$$\Sigma := \{[f::\mathbb{A} \to \mathbb{C}], [a::\mathbb{A}], [\forall G_{\mathbb{A}\to\mathbb{A}}.f(Ga)::\mathbb{D}]\}$$

we have $\Gamma \vdash_\Sigma f((\lambda X_\mathbb{A}.X)a)::\mathbb{D}$, but only $\Gamma \vdash_\Sigma fa::\mathbb{C}$.

7.4 Closure Properties of $\Sigma\Lambda^\to$

In this section we will discuss important technical invariants of the system $\Sigma\Lambda^\to$ which amount to an important plausibility check of our system.

The first one (theorem 7.4.2) is that sorted $\beta\eta$-conversion conserves well-sortedness. The second one (theorem 7.4.3) is that subformulae of well-sorted formulae should again be well-sorted. Together they ensure that all syntactical operations are well-defined.

Theorem 7.4.1 *If $\mathcal{E}: \Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B}$, then there is are sorts $\mathbb{A}$ and $\mathbb{B}$ and $\Sigma\Lambda^\to$-derivations $\mathcal{A}: \mathbf{A}::\mathbb{A}$ and $\mathcal{B}: \mathbf{B}::\mathbb{B}$ with $\mathbf{dp}(\mathcal{A}), \mathbf{dp}(\mathcal{B}) < \mathbf{dp}(\mathcal{E})$.*

Proof: We prove the assertion by induction on $\mathbf{dp}(\mathcal{E})$, starting with the base cases for *ws:β:top* and *ws:η:top*. In the first case we have $\mathbf{A} = (\lambda X.\mathbf{C})\mathbf{D}$ and $\mathbf{B} = [\mathbf{D}/X]\mathbf{C}$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{C}}{\Gamma, [X::\mathbb{D}] \vdash_\Sigma \mathbf{C}::\mathbb{C}} \quad \dfrac{\mathcal{D}}{\Gamma \vdash_\Sigma \mathbf{D}::\mathbb{D}}}{\Gamma \vdash_\Sigma \mathbf{A} \to^t_{\beta\eta} \mathbf{B}}\ \mathit{ws{:}\beta{:}top}}{\Gamma \vdash_\Sigma \mathbf{A} \to^1_{\beta\eta} \mathbf{B}}\ \mathit{tr{:}start}}{\Gamma \vdash_\Sigma \mathbf{A} \to_{\beta\eta} \mathbf{B}}\ \mathit{ms{:}start}}{\Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B}}\ \mathit{eq{:}start}$$

So we can get $\mathcal{B}$ by a single application of the *ws:inst* rule to $\mathcal{C}$ and $\mathcal{D}$ ($\mathbf{dp}(\mathcal{B}) = \mathbf{dp}(\mathcal{E}) - 3$) and $\mathcal{A}$ ($\mathbf{dp}(\mathcal{B}) = \mathbf{dp}(\mathcal{E}) - 2$) as

$$\dfrac{\dfrac{\dfrac{\mathcal{C}}{\Gamma, [X::\mathbb{D}] \vdash_\Sigma \mathbf{C}::\mathbb{C}}}{\Gamma \vdash_\Sigma (\lambda X.\mathbf{C})::\mathbb{D} \to \mathbb{C}}\ \mathit{ws{:}abs} \quad \dfrac{\mathcal{D}}{\Gamma \vdash_\Sigma \mathbf{D}::\mathbb{D}}}{\Gamma \vdash_\Sigma (\lambda X.\mathbf{C})\mathbf{D}::\mathbb{C}}\ \mathit{ws{:}app}$$

For the case of *ws:η:top*, we have $\mathbf{A} \equiv \lambda X.\mathbf{B}X$ and the top of $\mathcal{E}$ is of the form

$$\dfrac{\dfrac{\mathcal{B}}{\Gamma \vdash_\Sigma \mathbf{B}::\mathbb{B}} \quad X \notin \mathbf{Free}(\mathbf{A})}{\Gamma \vdash_\Sigma (\lambda X.\mathbf{B}X) \to^t_\eta \mathbf{B}}\ \mathit{ws^r{:}\eta{:}top}$$

So we get the assertion ($\mathbf{dp}(\mathcal{B}) = \mathbf{dp}(\mathcal{E}) - 4$ and $\mathbf{dp}(\mathcal{A}) = \mathbf{dp}(\mathcal{E}) - 2$) by choosing $\mathcal{A}$ to be

$$\dfrac{\dfrac{\dfrac{\mathcal{B}'}{\Gamma, [X::\mathfrak{d}(\mathbb{B})] \vdash_\Sigma \mathbf{B}::\mathbb{B}} \quad \dfrac{}{\Gamma, [X::\mathfrak{d}(\mathbb{B})] \vdash_\Sigma X::\mathfrak{d}(\mathbb{B})}\ \mathit{ws{:}var}}{\Gamma, [X::\mathfrak{d}(\mathbb{B})] \vdash_\Sigma \mathbf{B}X::\mathfrak{r}(\mathbb{B})}\ \mathit{ws{:}app}}{\Gamma \vdash_\Sigma (\lambda X.\mathbf{B}X)::\mathfrak{d}(\mathbb{B}) \to \mathfrak{r}(\mathbb{B})}\ \mathit{ws{:}abs}$$

where $\mathcal{B}'$ is obtained from $\mathcal{B}$ by monotonicity. In the case of *tr:abs*, we have

~~subterm-closed~~
closed!subterm

$\mathbf{A} \equiv \lambda X.\mathbf{C}$ and

$$\cfrac{\cfrac{\cfrac{\cfrac{\mathcal{E}'}{\Gamma, [X :: \mathbb{D}] \vdash_\Sigma \mathbf{C} \to_{\beta\eta} \mathbf{C}'}}{\Gamma \vdash_\Sigma \mathbf{A} \to_{\beta\eta} \mathbf{B}}\ tr{:}abs}{\Gamma \vdash_\Sigma \mathbf{A} \to_{\beta\eta} \mathbf{B}}\ ms{:}start}{\Gamma \vdash_\Sigma \mathbf{A} {=_{\beta\eta}} \mathbf{B}}\ eq{:}start$$

By inductive hypothesis for $\mathcal{E}'$ we get a $\Sigma\Lambda^{\to}$-derivation $\mathcal{A}'\colon \Gamma, [X :: \mathbb{D}] \vdash_\Sigma \mathbf{C} :: \mathbb{C}$ for some sort $\mathbb{C}$, which we can extend to a $\Sigma\Lambda^{\to}$-derivation $\mathcal{A}\colon \mathbf{A} :: \mathbb{D} \to \mathbb{C}$ by *ws:abs*. The case for $\mathbb{B}$ is analogous, in both cases we have $\mathbf{dp}(\mathcal{A}) < \mathbf{dp}(\mathcal{E}) - 2$. In the *tr:app* cases, we can find the necessary sort derivations to invoke *ws:app* on for $\mathcal{A}$ in the preconditions of the rules. For $\mathcal{B}$ we in invoke the inductive hypothesis. The case of *ms:ref* again yields the derivation $\mathcal{A}$ directly as a sub-derivation of $\mathcal{E}$ and finally *ms:trans* and *eq:sym* work by inductive hypothesis. □

Corollary 7.4.2 *If* $\Gamma \vdash_\Sigma \mathbf{A} {=_{\beta\eta}} \mathbf{B}$*, then* $\mathbf{A}, \mathbf{B} \in wsf_{\mathbb{A}}(\Sigma, \Gamma)$ *for some sort* $\mathbb{A}$.

Proof: The proof relies on a relatively simple observation: By Theorem 7.4.1 $\mathbf{A}$ is well-sorted, so we can give $\mathcal{B}$ the same sort by a simple application of *ws:$\beta\eta$*. □

We will now turn to the preservation of well-sortedness under subterms. We generally call a logical system like $\Sigma\Lambda^{\to}$ **subterm-closed**, iff for any well-sorted formula $\mathbf{A}$ with $\Gamma \vdash_\Sigma \mathbf{A} :: \mathbb{A}$ and $\mathbf{D}$ of $\mathbf{A}$ there is a sort $\mathbb{D}$ and a context $\Gamma \subseteq \Delta$, such that $\Delta \vdash_\Sigma \mathbf{D} :: \mathbb{D}$. In other words, if each subformula of a well-sorted formula is again well-sorted. Note that in other sorted systems with term declarations (e.g. that of Manfred Schmidt-Schauß in [SS89]) subterm-closedness was not guaranteed, but had to be assumed. In $\Sigma\Lambda^{\to}$ this property is a consequence of the particular way we have set up the mutually recursive definition of well-formed signatures and well-sorted formulae in Definition 7.2.7.

Theorem 7.4.3 ($\Sigma\Lambda^{\to}$ is Subterm-Closed)
For any well-sorted formula $\mathbf{A}$ *with* $\mathcal{A}\colon \Gamma \vdash_\Sigma \mathbf{A} :: \mathbb{A}$ *and any sub-formula* $\mathbf{D}$ *of* $\mathbf{A}$ *there is a sort* $\mathbb{D}$*, a context* $\Delta \parallel \Gamma$*, and a* $\Sigma\Lambda^{\to}$*-derivation* $\mathcal{D}\colon \Delta \vdash_\Sigma \mathbf{D} :: \mathbb{D}$ *with* $\mathbf{dp}(\mathcal{D}) \leq \mathbf{dp}(\mathcal{A})$*. If* $\mathbf{D}$ *is a proper sub-formula of* $\mathbf{A}$*, then* $\mathbf{dp}(\mathcal{D}) < \mathbf{dp}(\mathcal{A})$.

Proof: We show the assertion by induction over the depth of $\mathcal{A}$.

ws:var In this case $\mathbf{A} = \mathbf{D} = X \in \mathcal{V}$, so we can choose $\Delta := [X :: \Gamma(X)]$ and $\mathbb{D} = \mathbb{A}$. The strong depth condition does not apply.

supporting sort
sort!supporting

ws:*td* By 7.2.8 we have two cases: either $\mathbf{A}$ is a constant and we can argue as in the variable case with $\Delta = \emptyset$, or there is a strict sub-derivation $\mathcal{A}'{:}\vdash_{\Sigma'} \mathbf{A}{::}\mathbb{A}'$ of $\mathcal{A}$ for some sort $\mathbb{A}' \parallel \mathbb{A}$. Now we apply the inductive hypothesis to $\mathcal{A}'$ and obtain a $\Sigma\Lambda^\to$-derivation $\mathcal{D}{:}\,\Delta \vdash_\Sigma \mathbf{D}{::}\mathbb{D}$. As we can rename the bound variables in $\mathbf{A}$, we can assume that $\Delta \parallel \Gamma$. The depth conditions are a consequence of the fact that $\mathcal{A}'$ is a sub-derivation of $\mathcal{D}$.

ws:*app* In this case $\mathbf{A} \equiv \mathbf{BC}$ and we have $\mathcal{B}{:}\,\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{B}$ and $\mathcal{C}{:}\,\Gamma \vdash_\Sigma \mathbf{C}{::}\mathfrak{d}(\mathbb{B})$. We can assume $\mathbf{D}$ is a subformula of $\mathbf{B}$ or $\mathbf{C}$, since otherwise the assertion is trivial. In the first case the inductive hypothesis for $\mathcal{B}$ yields a derivation $\mathcal{D}{:}\,\Delta \vdash_\Sigma \mathbf{D}{::}\mathbb{D}$ with $\mathbf{dp}(\mathcal{D}) \leq \mathbf{dp}(\mathcal{B}) < \mathbf{dp}(\mathcal{A})$. The case where $\mathbf{D}$ is a sub-formula of $\mathbf{C}$ is analogous.

ws:*abs* Here we have $\mathbb{A} = \mathbb{C} \to \mathbb{B}$ and $\mathbf{A} \equiv \lambda X.\mathbf{B}$ with $\Gamma, [X{::}\mathbb{C}] \vdash_\Sigma \mathbf{B}{::}\mathbb{B}$; again we assume $\mathbf{D}$ is a sub-formula of $\mathbf{C}$, so the inductive hypothesis yields a derivation $\mathcal{D}{:}\,\Delta, [X{::}\mathbb{C}] \vdash_\Sigma \mathbf{D}{::}\mathbb{D}$ of $\mathcal{D}$ with $\Delta \parallel \Gamma, [X{::}\mathbb{C}]$ and $\mathbf{dp}(\mathcal{D}) \leq \mathbf{dp}(\mathcal{C}) < \mathbf{dp}(\mathcal{A})$.

ws:*βη* we have $\Sigma\Lambda^\to$-derivations $\mathcal{B}{:}\,\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{A}$ and $\mathcal{E}{:}\,\Gamma \vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{B}$ by construction. By Theorem 7.4.1, there is a derivation $\mathcal{C}{:}\,\mathbf{A}{::}\mathbb{C}$ for some sort $\mathbb{C}$. As $\mathbf{dp}(\mathcal{C}) \leq \mathbf{dp}(\mathcal{E}) < \mathbf{dp}(\mathcal{A})$ we can apply the inductive hypothesis to $\mathcal{C}$ and obtain the assertion. □

As a consequence, any valid signature is subterm-closed. The property of subterm-closedness is natural in the context of mathematics, since it does not make sense to allow ill-formed subexpressions in well-formed expressions. This situation may, for instance, be different in the field of natural language processing. Non subterm-closed signatures would also cause technical problems, for example, structural induction would not be possible.

Remark 7.4.4 For a fixed, valid signature Σ we can simplify the inference system by dropping the premise $\vdash_{sig} \Sigma$ from the rule *ws*:*td*, since the proofs in the original system can be obtained from those in the simplified system by copying the validity proofs for Σ into the *ws*:*td*-nodes.

7.5 Domain Sorts in $\Sigma^r\Lambda^\to$

We now prove that valid signatures respect function domains in the sense that for every formula $\mathbf{A}$ of functional sort and any sorts $\mathbb{A}, \mathbb{B}$ of $\mathbf{A}$, we must have $\mathbb{A}$ $\mathbf{Rdom}$ $\mathbb{B}$, and therefore $\mathfrak{d}(\mathbb{A}) = \mathfrak{d}(\mathbb{B})$, which we call the **supporting sort** of $\mathbf{A}$. An important corollary of this is that we have inversion principles for application and λ-abstraction.

We will show that the restrictions imposed on the validity of signatures indeed guarantee that all functional well-sorted formulae have unique supporting sorts, which exactly capture the intuition of mathematical practice, where functions have unique domains associated with them. This fact a posteriori justifies our definition of sorted η-conversion, which is a weak form of the extensionality principle, and therefore relies on the existence of unique domains for functions.

Theorem 7.5.1 *If* $\Gamma \vdash^r_\Sigma \mathbf{A}{::}\mathbb{A}$ *and* $\Gamma \vdash^r_\Sigma \mathbf{A}{::}\mathbb{B}$*, then* $\mathbb{A}\ \mathbf{Rdom}\ \mathbb{B}$.

Proof: To make the induction go through we prove the stronger assertion that $\mathcal{A}{:}\,\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$, $\mathcal{B}{:}\,\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{B}$, and $\mathcal{E}{:}\,\Gamma \vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{B}$ together entail $\mathbb{A}\ \mathbf{Rdom}\ \mathbb{B}$ by induction on $\mu(\mathcal{A},\mathcal{B},\mathcal{E})$ with respect to $\prec$. We take $\mu(\mathcal{A},\mathcal{B},\mathcal{E}) := (\mu_1(\mathcal{A},\mathcal{B}),\mathcal{E})$, where $\mu_1(\mathcal{A},\mathcal{B}) := (\mathbf{dp}(\mathcal{A}),\mathbf{dp}(\mathcal{B}))$. Correspondingly, we take $\prec$ to be the strict lexicographic ordering on pairs for the multiset ordering on pairs of natural numbers and the structural ordering on $\Sigma\Lambda^{\rightarrow}$-derivations.

$\mathcal{A}$ or $\mathcal{B}$ ends in *ws:$\beta\eta$***.** We have the following situation (assuming $\mathcal{A}$ does):

$$\frac{\dfrac{\mathcal{A}'}{\Gamma \vdash_\Sigma \mathbf{C}{::}\mathbb{A}} \quad \dfrac{\mathcal{A}''}{\Gamma \vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{C}}}{\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}}\ \mathit{ws{:}\beta\eta}$$

Now let $\mathcal{E}'$ be the $\Sigma\Lambda^{\rightarrow}$-derivation constructed from $\mathcal{E}{:}\,\Gamma \vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{B}$ and $\mathcal{A}''$ by a single application of *ms:trans*. We have $\mu(\mathcal{A}',\mathcal{B},\mathcal{E}') \prec \mu(\mathcal{A},\mathcal{B},\mathcal{E})$, and thus by inductive hypothesis $\mathbb{A}\ \mathbf{Rdom}\ \mathbb{B}$.

Note that for this argument we have only used the fact that $\mathcal{A}'$ is a subderivation of $\mathcal{A}$. This makes this argument widely applicable in this proof. Clearly the cases where $\mathcal{B}$ ends in *ws:$\beta\eta$* is analogous, since sorted $\beta\eta$-equality is symmetric.

$\mathcal{A}$ or $\mathcal{B}$ ends in *ws:td***.** There is a term declaration $[\mathbf{A}{::}\mathbb{A}] \in \Sigma$, a signature $\Sigma' \subseteq \Sigma$, and a $\Sigma\Lambda^{\rightarrow}$-subderivation $\mathcal{A}'{:}\vdash_{\Sigma'} \mathbf{A}{::}\mathbb{C}$ of $\mathcal{A}$, for some sort $\mathbb{C}\ \mathbf{Rdom}\ \mathbb{A}$. Now $\mathbf{dp}(\mathcal{A}') < \mathbf{dp}(\mathcal{A})$ implies $\mu_1(\mathcal{A}',\mathcal{B}) < \mu_1(\mathcal{A},\mathcal{B})$, so by inductive hypothesis we have $\mathbb{C}\ \mathbf{Rdom}\ \mathbb{B}$ and thus $\mathbb{A}\ \mathbf{Rdom}\ \mathbb{B}$ by transitivity of $\mathbf{Rdom}$. Again the argumentation is analogous for the case where $\mathcal{B}$ ends in *ws:td*.

Now we proceed by analyzing the cases for $\mathcal{E}$. Since we have treated all unstructural cases for $\mathcal{A}$ and $\mathcal{B}$, we can now make use of structural arguments.

$\mathcal{E}$ ends in *ms:ref***.** Note that this is technically incorrect. (*ms:ref* can never prove a judgment of the form $\Gamma \vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{A}$, only $\Gamma \vdash_\Sigma \mathbf{A} \rightarrow_{\beta\eta} \mathbf{A}$), but we use it suggestively for: $\mathcal{A}$ ends in

$$\dfrac{\dfrac{\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}}{\Gamma \vdash_\Sigma \mathbf{B} \rightarrow_{\beta\eta} \mathbf{B}}\,\mathit{ms{:}ref}}{\Gamma \vdash_\Sigma \mathbf{A} {=}_{\beta\eta} \mathbf{B}}\,\mathit{eq{:}start}$$

This extra derivation tail does not change the situation or the measures involved, so we will continue the suggestive labeling below. We have $\mathbf{A} \equiv \mathbf{B}$. So if $\mathcal{A}$ ends in $\mathit{ws{:}var}$, then we can assume that $\mathcal{B}$ does too, since $\mathit{ws{:}app}$ and $\mathit{ws^r{:}abs}$ do not apply and we have already treated $\mathit{ws{:}\beta\eta}$ and $\mathit{ws{:}td}$. In the remaining case we have $\mathbb{A} = \mathbb{B}$, which clearly entails $\mathbb{A}$ **Rdom** $\mathbb{B}$. If $\mathcal{A}$ and $\mathcal{B}$ both end in $\mathit{ws{:}app}$, then $\mathbf{A} \equiv \mathbf{CD} \equiv \mathbf{B}$, and there are subderivations $\mathcal{A}'{:}\,\Gamma \vdash_\Sigma \mathbf{C}{::}\mathbb{C}$ and $\mathcal{B}'{:}\,\Gamma \vdash_\Sigma \mathbf{C}{::}\mathbb{D}$ of $\mathcal{A}$ and $\mathcal{B}$ that satisfy the inductive hypothesis, so we have $\mathbb{C}$ **Rdom** $\mathbb{D}$ and thus $\mathbb{A} = \mathfrak{r}(\mathbb{C})$ **Rdom** $\mathfrak{r}(\mathbb{D}) = \mathbb{B}$. Finally, if $\mathcal{A}$ and $\mathcal{B}$ both end in $\mathit{ws^r{:}abs}$, then $\mathbf{A} = \lambda X_{\mathbb{C}}.\mathbf{D} = \mathbf{B}$ and there are subderivations $\mathcal{A}'{:}\,\Gamma, [X{::}\mathbb{C}] \vdash_\Sigma \mathbf{D}{::}\mathbb{D}$ and $\mathcal{B}'{:}\,\Gamma, [X{::}\mathbb{C}] \vdash_\Sigma \mathbf{D}{::}\mathbb{E}$, so by inductive hypothesis $\mathbb{D}$ **Rdom** $\mathbb{E}$, and thus $\mathbb{A} = (\mathbb{C} \rightarrow \mathbb{D})$ **Rdom** $(\mathbb{C} \rightarrow \mathbb{E}) = \mathbb{B}$. This completes the case, where $\mathcal{E}$ ends in $\mathit{ms{:}ref}$.

$\mathcal{E}$ **ends in** $\mathit{ws{:}\beta{:}top}$ then $\mathbf{A} \equiv (\lambda X_{\mathbb{D}}.\mathbf{C})\mathbf{D}$ and $\mathbf{B} \equiv [\mathbf{D}/X]\mathbf{C}$. Since we have treated the cases $\mathit{ws{:}\beta\eta}$ and $\mathit{ws{:}td}$, we can assume that $\mathcal{A}$ ends in $\mathit{ws{:}app}$ and we have the following situation

$$\dfrac{\dfrac{\mathcal{A}'}{\Gamma \vdash_\Sigma \lambda X_{\mathbb{D}}.\mathbf{C}{::}\mathbb{E}} \quad \dfrac{\mathcal{A}''}{\Gamma \vdash_\Sigma \mathbf{D}{::}\mathfrak{d}(\mathbb{E})}}{\Gamma \vdash_\Sigma (\lambda X_{\mathbb{D}}.\mathbf{C})\mathbf{D}{::}\mathfrak{r}(\mathbb{E}) = \mathbb{A}}\,\mathit{ws{:}app}$$

with an argument exactly as in the case for $\mathit{ws{:}\beta\eta}$ for $\mathcal{A}$ we treat the cases, where $\mathcal{A}'$ ends in $\mathit{ws{:}\beta\eta}$ or $\mathit{ws{:}td}$, so we can assume that $\mathcal{A}'$ ends in $\mathit{ws^r{:}abs}$ and we have the following situation:

$$\dfrac{\dfrac{\dfrac{\mathcal{C}}{\Gamma, [X{::}\mathbb{D}] \vdash_\Sigma \mathbf{C}{::}\mathbb{A}}}{\Gamma \vdash_\Sigma \lambda X_{\mathbb{D}}.\mathbf{C}{::}\mathbb{D} \rightarrow \mathbb{A}}\,\mathit{ws^r{:}abs} \quad \dfrac{\mathcal{A}''}{\Gamma \vdash_\Sigma \mathbf{D}{::}\mathbb{D}}}{\Gamma \vdash_\Sigma (\lambda X_{\mathbb{D}}.\mathbf{C})\mathbf{D}{::}\mathbb{A}}\,\mathit{ws{:}app}$$

If $\mathbf{C} \equiv X$, then $\mathbf{B} \equiv \mathbf{D}$ and $\mathbb{D}$ **Rdom** $\mathbb{A}$, since we can invoke the inductive hypothesis for $\mathcal{C}$, $\mathit{ws{:}var}$, and $\mathit{ms{:}ref}$. Thus we obtain $\mathbb{A}$ **Rdom** $\mathbb{B}$ by inductive hypothesis for $\mathcal{A}''$, $\mathcal{B}$, and $\mathcal{E}$, since **Rdom** is transitive. With this argument we can for the rest of this case assume that

$\mathbf{C} \not\equiv X$. If $\mathcal{B}$ ends in *ws*:*var*, then $\mathbf{B} \equiv \mathbf{C} \in \mathbf{Dom}(\Gamma)$, since $\mathbf{C} \not\equiv X$. Thus we obtain the assertion by inductive hypothesis for $\mathcal{C}$, $\mathcal{B}$, and $\mathcal{E}$. If $\mathcal{B}$ ends in *ws*:*app*, then $\mathbf{C} \equiv \mathbf{EF}$, $\mathbf{B} = [\mathbf{D}/X](\mathbf{EF})$, and $\mathcal{B}$ is of the form

$$\cfrac{\cfrac{\mathcal{B}'}{\Gamma \vdash_\Sigma [\mathbf{D}/X]\mathbf{E}::\mathbb{E}} \quad \Gamma \vdash_\Sigma [\mathbf{D}/X]\mathbf{F}::\mathfrak{d}(\mathbb{E})}{\Gamma \vdash_\Sigma ([\mathbf{D}/X]\mathbf{E})([\mathbf{D}/X]\mathbf{F})::\mathfrak{r}(\mathbb{E}) = \mathbb{B}}\ ws{:}app$$

by an argument as above we can treat the cases, where $\mathcal{C}$ ends in *ws*:$\beta\eta$ or *ws*:*td* and $\mathcal{C}$ must have the form

$$\cfrac{\cfrac{\mathcal{C}'}{\Gamma, [X::\mathbb{A}] \vdash_\Sigma \mathbf{E}::\mathbb{F}} \quad \Gamma, [X::\mathbb{A}] \vdash_\Sigma \mathbf{F}::\mathfrak{d}(\mathbb{F})}{\Gamma, [X::\mathbb{A}] \vdash_\Sigma \mathbf{EF}::\mathfrak{r}(\mathbb{F}) = \mathbb{A}}\ ws{:}app$$

so we can construct a $\Sigma\Lambda^{\rightarrow}$-derivation $\mathcal{C}''$

$$\cfrac{\cfrac{\cfrac{\mathcal{C}'}{\Gamma, [X::\mathbb{A}] \vdash_\Sigma \mathbf{E}::\mathbb{F}}}{\Gamma \vdash_\Sigma (\lambda X_{\mathbb{D}}.\mathbf{E})::\mathbb{D} \to \mathbb{F}}\ ws^r{:}abs \quad \cfrac{\mathcal{A}''}{\Gamma \vdash_\Sigma \mathbf{D}::\mathbb{D}}}{\Gamma \vdash_\Sigma (\lambda X_{\mathbb{D}}.\mathbf{E})\mathbf{D}::\mathbb{F}}\ ws{:}app$$

Clearly we have $\mathbf{dp}(\mathcal{C}'') \leq \mathbf{dp}(\mathcal{A})$, so $\mu(\mathcal{C}'', \mathcal{B}') < \mu(\mathcal{A}, \mathcal{B})$, and therefore the inductive hypothesis guarantees that $\mathbb{E}\ \mathbf{Rdom}\ \mathbb{F}$, so $\mathbb{A} = \mathfrak{r}(\mathbb{F})\ \mathbf{Rdom}\ \mathfrak{r}(\mathbb{E}) = \mathbb{B}$. The case where $\mathcal{B}$ ends in *ws*^{r}:*abs* can be treated with analogous methods and completes the analysis for the case where $\mathcal{E}$ ends in *ws*:β:*top*.

$\mathcal{E}$ ends in *ws*^{r}:η:*top* then $\mathbf{A} \equiv (\lambda X_{\mathbb{D}}.\mathbf{B}X)$ and with an argument exactly as in the case for *ws*:$\beta\eta$ for $\mathcal{A}$ we treat the cases, where $\mathcal{A}'$ ends in *ws*:$\beta\eta$ or *ws*:*td*, so we can assume that $\mathcal{A}$ ends in *ws*:*abs* and we have the following situation:

$$\cfrac{\cfrac{\cfrac{\mathcal{C}}{\Gamma \vdash_\Sigma \mathbf{B}::\mathbb{A}'} \quad \cfrac{}{[X::\mathbb{D}] \vdash_\Sigma X::\mathbb{D}}\ ws{:}var}{\Gamma \vdash_\Sigma \mathbf{B}X::\mathfrak{r}(\mathbb{A})}\ ws{:}app}{\Gamma \vdash_\Sigma (\lambda X_{\mathbb{D}}.\mathbf{B}X)::\mathbb{A}}\ ws^r{:}abs$$

where $\mathfrak{d}(\mathbb{A})' = \mathbb{D}$ and $\mathfrak{r}(\mathbb{A})' = \mathfrak{r}(\mathbb{A})$. By inductive hypothesis we for $\mathcal{C}$, $\mathcal{B}$, and the the $\Sigma\Lambda^{\rightarrow}$-derivation consisting only of a single *ms*:*ref* step we obtain $\mathbb{A}'\ \mathbf{Rdom}\ \mathbb{A}$, which yields the assertion, $\mathbb{A}\ \mathbf{Rdom}\ \mathbb{A}'$ by construction.

Now it only remains to check the inductive cases for $\mathcal{E}$. If $\mathcal{E}$ ends in *tr*:*app*, then $\mathbf{A}$ and $\mathbf{B}$ must be applications, so we can assume that $\mathcal{A}$ and $\mathcal{B}$ end in *ws*:*app*. Thus we have $\Sigma\Lambda^{\rightarrow}$-subderivations $\mathcal{A}'$, $\mathcal{B}'$, and $\mathcal{E}'$ that meet the assumptions of the inductive hypothesis, which then yields $\mathbb{A}$ **Rdom** $\mathbb{B}$. It can easily be seen that the cases *tr*:*abs*, *ms*:*trans*, and *eq*:*sym*, can be handled with related methods. Thus we have finally completed our analysis of all possible cases for $\mathcal{E}$ and thus proven the assertion. □

Corollary 7.5.2 (Inversion for Abstractions)
If $\mathcal{A}\colon\Gamma\vdash_{\Sigma}(\lambda X_{\mathbb{B}}.\mathbf{C})::\mathbb{A}$*, then there is a* $\Sigma\Lambda^{\rightarrow}$*-derivation* $\mathcal{C}\colon\Gamma,[X::\mathbb{B}]\vdash_{\Sigma}\mathbf{B}::\mathbb{C}$ *with* $\mathbf{dp}(\mathcal{C})<\mathbf{dp}(\mathcal{A})$ *for some sort* $\mathbb{C}$ *with* $\mathbb{A}\parallel\mathbb{B}\rightarrow\mathbb{C}$*.*

Proof: Theorem 7.4.3 gives us a suitable $\mathbb{C}$, Δ and $\mathcal{C}\Delta\vdash_{\Sigma}\mathbf{C}::\mathbb{C}$. For the compatibility condition for $\Sigma^r\Lambda^{\rightarrow}$ note that by *ws*^{r}:*abs* we have $\Gamma\vdash_{\Sigma}(\lambda X_{\mathbb{B}}.\mathbf{A})::\mathbb{B}\rightarrow\mathbb{C}$, and $\mathbb{A}\parallel\mathbb{B}\rightarrow\mathbb{C}$ by Theorem 7.5.1. □

Remark 7.5.3 The previous corollary allows us to infer the sorts of bound variables in well-sorted formulae. This enables us to simplify the notation by dropping the sort in λ-abstractions, if we specify the sort. For example, if $\mathbf{B}\equiv(\lambda\overline{X^k_{\mathbb{B}^k}}.\mathbf{A})$ and $\Gamma\vdash_{\Sigma}\mathbf{B}::\mathbb{A}$, then $\mathbb{B}^i=\mathfrak{d}^i(\mathbb{A})$. Therefore we often write $\mathbf{B}$ as $\lambda\overline{X^k}.\mathbf{A}$.

A similar inversion principle can be shown for applications.

Lemma 7.5.4 (Inversion for Applications)
If $\Gamma\vdash_{\Sigma}\mathbf{AB}::\mathbb{C}$*, then there is a sort* $\mathbb{A}$*, such that* $\Gamma\vdash_{\Sigma}\mathbf{A}::\mathbb{A}$ *and* $\Gamma\vdash_{\Sigma}\mathbf{B}::\mathfrak{d}(\mathbb{A})$ *and* $\mathfrak{r}(\mathbb{A})\parallel\mathbb{C}$*. Moreover, we can even pick* $\mathbb{A}=\mathfrak{d}(\mathbb{A})\rightarrow\mathfrak{r}(\mathbb{A})$*.*

Proof: We prove the assertion by an induction over the structure of the derivation $\mathcal{A}\colon\Gamma\vdash_{\Sigma}\mathbf{AB}::\mathbb{C}$. Since $\mathbf{AB}$ is an application, the *ws*:*var* and *ws*:*abs* cases cannot occur.

If $\mathcal{A}$ ends in *ws*:*td*, then by lemma 7.2.8 and monotonicity, there is a proper subderivation $\mathcal{A}'\colon\vdash_{\Sigma}\mathbf{AB}::\mathbb{D}$ of $\mathcal{A}$ for some sort $\mathbb{D}$ with $\mathbb{D}\parallel\mathbb{C}$. So the inductive hypothesis together with transitivity of $\parallel$ yields the assertion.

If $\mathcal{A}$ ends in *ws*:*app*, it has the following form

$$\cfrac{\cfrac{\mathcal{A}'}{\Gamma\vdash_{\Sigma}\mathbf{A}::\mathbb{D}}\qquad\Gamma\vdash_{\Sigma}\mathbf{B}::\mathfrak{d}(\mathbb{D})}{\Gamma\vdash_{\Sigma}\mathbf{AB}::\mathfrak{r}(\mathbb{D})}\;ws{:}app$$

Note that $\mathfrak{r}(\mathbb{D})=\mathbb{C}$ and therefore $\mathfrak{r}(\mathbb{D})\parallel\mathbb{C}$ by construction, which yields the first part of the assertion by monotonicity. The stronger assertion can

$\mathbf{supp}^i(\mathbf{A})$ *i^{th} argument sort of* $\mathbf{A}$
argument sort!i^{th}
supporting sort

be obtained by the following derivation

$$\cfrac{\cfrac{\mathcal{A}'}{\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{D}} \quad \Gamma \vdash_\Sigma \mathbf{A}{=}_{\beta\eta}(\lambda X.\mathbf{A}X)}{\Gamma \vdash_\Sigma \mathbf{A}{::}\mathfrak{d}(\mathbb{D}) \to \mathfrak{r}(\mathbb{D})}\ ws{:}\beta\eta$$

The only remaining case is that where, $\mathcal{A}$ ends in $ws{:}\beta\eta$ and has the form

$$\cfrac{\cfrac{\mathcal{A}'}{\Gamma \vdash_\Sigma \mathbf{C}{::}\mathbb{C}} \quad \Gamma \vdash_\Sigma \mathbf{AB}{=}_{\beta\eta}\mathbf{C}}{\Gamma \vdash_\Sigma \mathbf{AB}{::}\mathbb{C}}\ ws{:}\beta\eta$$

By Theorem 7.4.1, we have a $\Sigma\Lambda^{\to}$-derivation $\mathcal{A}'{:}\,\Gamma \vdash_\Sigma \mathbf{AB}{::}\mathbb{C}'$ of depth smaller than $\mathcal{A}$. So by inductive hypothesis for $\mathcal{A}'$ we have $\mathfrak{r}(\mathbb{A}) \parallel \mathbb{C}'$, but $\mathbb{C}' \parallel \mathbb{C}$ by Theorem 7.5.1, so we obtain the assertion by transitivity of $\parallel$. ☐

Definition 7.5.5 Let $\mathbf{A}$ be a well-sorted formula in $\Sigma^r\Lambda^{\to}$, then we call the unique sort $\mathbb{B}^i$ such that $\mathbb{B}^i = \mathfrak{d}^i(\mathbb{A})$ for all sorts $\mathbb{A}$ with $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ the i^{th}**argument sort** of $\mathbf{A}$ ($\mathbf{supp}^i(\mathbf{A})$). We also call the first argument sort of $\mathbf{A}$ the **supporting sort** of $\mathbf{A}$ and denote it by $\mathbf{supp}(\mathbf{A})$.

Remark 7.5.6 The set of argument sorts is effectively computable. If $\mathbf{A}$ is an abstraction of the form $\lambda X_{\mathbb{A}}.\mathbf{B}$, then $\mathbf{supp}(\mathbf{A}) = \mathbb{A}$ and $\mathbf{supp}^{i+1}(\mathbf{A}) = \mathbf{supp}^i(\mathbf{B})$, if $\mathbf{A} \equiv \mathbf{BC}$, then $\mathbf{supp}^i(\mathbf{A}) = \mathbf{supp}^{i+1}(\mathbf{B})$. The supporting sorts of constants and variables can be read off their declarations.

Remark 7.5.7 The theorems above are the reason for requiring the **Rdom** proviso for any term declaration to be added to a signature by the rule *sig:td* in $\Sigma^r\Lambda^{\to}$. We do not need such a proviso for the *sig:const* rule, since for new constants there is no sort information present that would have to be respected.

Most mathematicians would agree that functional extensionality relies heavily on the notion of explicitly specified domains of functions. Unique supporting sorts are intended to syntactically capture this intuition in $\Sigma\Lambda^{\to}$. Indeed in mathematics, functions are assumed to have unique (explicitly specified) domains, and must therefore be distinguished from restrictions to subdomains. For example, the addition function on the reals must be distinguished from the addition function on the natural numbers, and in general functions f and g should only be considered the same, if $fa = ga$ for all a in the common (explicitly specified) domain of f and g. Observing these distinctions is necessary for a correct treatment of extensional higher-order calculi, and they must be reflected in the syntax of any such calculus. This fact is taken into account in our definition of Σ-structures by requiring that all functions in $\mathcal{D}_{\mathbb{A}}$ are total on $\mathcal{D}_{\mathfrak{d}(\mathbb{A})}$.

7.6 Sorted Reduction

It is important to the program of this book that the fundamental operations of the calculus do not allow the formation of ill-sorted terms from well-sorted ones. This will ensure that our calculus never has to handle ill-sorted terms, even intermediately. We have seen in 7.3.4 that Σ-instantiation conserves sorts. In this section we will use this to show that, if $\Gamma \vdash_\Sigma \mathbf{A}{=_{\beta\eta}}\mathbf{B}$, then $\mathbf{A}$ and $\mathbf{B}$ have the same sorts. This fact implies that sorted reduction is strongly normalizing and all results carry over from the typed case. For the confluence result in $\Sigma^r\Lambda^\rightarrow$ we need that well-sorted formulae of functional type have unique domain sorts.

Lemma 7.6.1 *If* $\Gamma \vdash_\Sigma \mathbf{A}{=_{\beta\eta}}\mathbf{B}$*, then* $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$*, iff* $\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{A}$*.*

Proof: This is immediate from the *ws:*$\beta\eta$ inference rule and the symmetry of $=_{\beta\eta}$. □

Theorem 7.6.2 $\Gamma \vdash_\Sigma \mathbf{A} \rightarrow_{\beta\eta} \mathbf{B}$*, iff* $\Sigma \vdash \flat\mathbf{A} \rightarrow_{\beta\eta} \flat\mathbf{B}$ *and* $\mathbf{A}$ *is well-sorted.*

Proof: Sufficiency is a direct consequence of lemma 7.2.9(1) and Lemma 7.4.1; we prove the converse direction by induction on the depths of the $\Lambda^\rightarrow$-derivation $\mathcal{E}\colon \Sigma \vdash \flat\mathbf{A} \rightarrow_{\beta\eta} \mathbf{B}$ and the $\Sigma\Lambda^\rightarrow$-derivation $\mathcal{A}\colon \Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$.

wff:β*:top* In this case $\mathbf{A} \equiv (\lambda X.\mathbf{C})\mathbf{D}$ and $\mathbf{B} \equiv [\mathbf{D}/X]\mathbf{C}$. Lemma 7.5.4 applied to $\mathcal{A}$ yields a sort $\mathbb{C}$ and $\Sigma\Lambda^\rightarrow$-derivations $\mathcal{C}\colon \Gamma \vdash_\Sigma \lambda X.\mathbf{C}{::}\mathbb{C}$ and $\mathcal{D}\colon \Gamma \vdash_\Sigma \mathbf{D}{::}\mathfrak{d}(\mathbb{C})$. Inversion for abstractions (7.5.2) yields $\Gamma, [X{::}\mathfrak{d}(\mathbb{C})] \vdash_\Sigma \mathbf{C}{::}\mathfrak{r}(\mathbb{C})$. This gives us the needed well-sortedness conditions for *ws:*β*:top*, so we the following $\Sigma\Lambda^\rightarrow$-derivation concludes the proof.

$$\frac{\dfrac{\mathcal{B}}{\Gamma, [X{::}\mathfrak{d}(\mathbb{C})] \vdash_\Sigma \mathbf{C}{::}\mathfrak{r}(\mathbb{C})} \qquad \dfrac{\mathcal{D}}{\Gamma \vdash_\Sigma \mathbf{D}{::}\mathfrak{d}(\mathbb{C})}}{\Gamma \vdash_\Sigma (\lambda X.\mathbf{C})\mathbf{D} \rightarrow_\beta [\mathbf{D}/X]\mathbf{C}}\ ws{:}\beta{:}top$$

wff:η*:top* In this case we have $\mathbf{A} \equiv (\lambda X.\mathbf{B}X)$ inversion for applications and abstractions gives us a $\Sigma\Lambda^\rightarrow$-derivation $\mathcal{B}\colon \Gamma, [X{::}\mathbb{B}] \vdash_\Sigma \mathbf{B}{::}\mathbb{B}$ and in fact (as $X \notin \mathbf{Free}(\mathbf{B})$) even $\mathcal{B}'\colon \Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{B}$ by Lemma 7.2.14. Thus we conclude this case by *ws:*η*:top*

$$\frac{\dfrac{\mathcal{B}'}{\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{B}} \qquad X \notin \mathbf{Free}(\mathbf{B})}{\Gamma \vdash_\Sigma (\lambda X.\mathbf{B}X) \rightarrow_\eta \mathbf{B}}\ ws^r{:}\eta{:}top$$

sorted β-normal form
normal form!sorted β

tr:app:fn Here we have $\flat\mathbf{A} \equiv \flat\mathbf{C}\flat\mathbf{D}$, $\flat\mathbf{B} \equiv \flat\mathbf{C}'\flat\mathbf{D}'$, and $\mathcal{E}'\colon \flat\mathbf{C} \to_{\beta\eta} \flat\mathbf{C}'$, so we also have $\mathbf{A} \equiv \mathbf{CD}$, $\mathbf{B} \equiv \mathbf{C}'\mathbf{D}'$. By applying inversion for applications (7.5.4) to $\mathcal{A}$ we obtain a sort $\mathbb{C}$, where $\mathcal{C}\colon \Gamma \vdash_\Sigma \mathbf{C}{::}\mathbb{C}$ and $\mathcal{D}\colon \Gamma \vdash_\Sigma \mathbf{D}{::}\mathfrak{d}(\mathbb{C})$. By the inductive hypothesis applied to $\mathcal{C}$ and $\mathcal{E}'$ we have a $\Sigma\Lambda^{\to}$-derivation $\mathcal{C}'\colon \Gamma \vdash_\Sigma \mathbf{C} \to_{\beta\eta} \mathbf{C}'$ and therefore we can conclude this case with

$$\frac{\dfrac{\mathcal{C}'}{\Gamma \vdash_\Sigma \mathbf{C} \to_{\beta\eta} \mathbf{C}'} \quad \dfrac{\mathcal{C}}{\Gamma \vdash_\Sigma \mathbf{C}{::}\mathbb{C}} \quad \dfrac{\mathcal{D}}{\Gamma \vdash_\Sigma \mathbf{D}{::}\mathfrak{d}(\mathbb{C})}}{\Gamma \vdash_\Sigma \mathbf{CD} \to_{\beta\eta} \mathbf{C}'\mathbf{D}}\ \mathit{tr{:}app}$$

tr:app:arg is analogous to *tr:app:fn*.

tr:abs Here $\mathbf{A} \equiv (\lambda X.\mathbf{C})$ and $\mathbf{B} \equiv (\lambda X.\mathbf{C}')$ and $\mathcal{E}\colon \Sigma \vdash \lambda X.\flat\mathbf{C} \to_{\beta\eta} \lambda X.\flat\mathbf{C}'$. As above, well-sortedness of $\lambda X.\mathbf{C}$ together with inversion of abstraction (Theorem 7.5.2) provides a derivation $\mathcal{C}\colon \Gamma, [X{::}\mathbb{D}] \vdash_\Sigma \mathbf{C}{::}\mathbb{C}$, which allows to use the inductive hypothesis to obtain a derivation $\mathcal{A}'\colon \Gamma, [X{::}\mathbb{D}] \vdash_\Sigma \mathbf{C} \to_{\beta\eta} \mathbf{C}'$ and to conclude this case with

$$\frac{\dfrac{\mathcal{A}'}{\Gamma, [X{::}\mathfrak{d}(\mathbb{A})] \vdash_\Sigma \mathbf{C} \to_{\beta\eta} \mathbf{C}'}}{\Gamma \vdash_\Sigma (\lambda X.\mathbf{C}) \to_{\beta\eta} (\lambda X.\mathbf{C}')}\ \mathit{tr{:}abs}$$

The remaining cases are trivial. *ms:trans* is a direct consequence of the inductive hypothesis, and for *ms:ref* we have $\mathbf{A} \equiv \mathbf{B}$ and $\mathcal{A}\colon \Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$, so we we can directly use the well-sorted *ms:ref* rule. ☐

Corollary 7.6.3 *Sorted $\beta\eta$-reduction on wsf(Σ, Γ) is terminating and confluent.*

Proof: For termination we note that any sequence of sorted $\beta\eta$-reductions is also a sequence of (unsorted) $\beta\eta$-reductions and those always terminate by 1.3.14.

For confluence let $\Gamma \vdash_\Sigma \mathbf{A} \to_{\beta\eta} \mathbf{B}$ and $\Gamma \vdash_\Sigma \mathbf{A} \to_{\beta\eta} \mathbf{C}$, then for any $\flat\mathbf{A} \to_{\beta\eta} \flat\mathbf{B}$ and $\flat\mathbf{A} \to_{\beta\eta} \flat\mathbf{C}$, by 1.3.14, there is a formula $\flat\mathbf{D}$, such that $\flat\mathbf{B} \to^*_{\beta\eta} \flat\mathbf{D}$ and $\flat\mathbf{C} \to^*_{\beta\eta} \flat\mathbf{D}$.

$\mathcal{A}$, $\mathbf{B}$, and $\mathcal{C}$ are well-sorted by Theorem 7.4.2, so by the converse direction of 7.6.2 we have $\Gamma \vdash_\Sigma \mathbf{B} \to^*_{\beta\eta} \mathbf{D}$ and $\Gamma \vdash_\Sigma \mathbf{C} \to^*_{\beta\eta} \mathbf{D}$. ☐

Remark 7.6.4 In light of the previous theorem it makes sense to speak of *the* **sorted β-normal form** and *the* **sorted (long) $\beta\eta$-normal form**. Since these normal forms are also unsorted normal forms all results and definitions from the typed case (cf. 1.3) carry over to $\Sigma\Lambda^{\rightarrow}$. Furthermore, lemma 7.6.1 implies that both notions of sorted normalization conserve the sets of sorts of formulae.

Lemma 7.6.5 *Let Σ be a set of term declarations, $\mathcal{D} = [\mathbf{B}::\mathbb{B}] \in \Sigma$, and let $\mathcal{D}' = [\mathbf{B}'::\mathbb{B}]$, where $\mathbf{B}'$ is the $\beta\eta$-normal form of $\mathbf{B}$. Furthermore, let $\Sigma' = \{\mathcal{D}' \mid \mathcal{D} \in \Sigma\}$, then*

1. $\vdash_{sig} \Sigma$ *implies* $\vdash_{sig} \Sigma'$
2. $\Gamma \vdash_{\Sigma} \mathbf{A}::\mathbb{A}$, *iff* $\Gamma \vdash_{\Sigma'} \mathbf{A}::\mathbb{A}$
3. $\Gamma \vdash_{\Sigma} \mathbf{A}=_{\beta\eta}\mathbf{C}$, *iff* $\Gamma \vdash_{\Sigma'} \mathbf{A}=_{\beta\eta}\mathbf{C}$

Proof: We prove the assertions by simultaneous induction on the structure of the $\Sigma\Lambda^{\rightarrow}$-derivations involved.

1. If $\mathcal{E}:\vdash_{sig} \Sigma$ ends in *sig:td*, then $\Sigma = \Delta, [\mathbf{B}::\mathbb{B}]$ and $\vdash_{\Delta} \mathbf{B}::\mathbb{C}$ such that $\mathbb{B} \parallel \mathbb{C}$. Let $\Delta' := \{\mathcal{D}' \mid \mathcal{D} \in \Delta\}$, then we have $\vdash_{sig} \Delta'$ by inductive hypothesis 1, and $\vdash_{\Delta'} \mathbf{B}::\mathbb{C}$ by inductive hypothesis 2 (the proof of $\vdash_{\Delta} \mathbf{B}::\mathbb{C}$ is a subderivation of $\mathcal{E}$). By 7.6.1 we have $\vdash_{\Delta'} \mathbf{B}'::\mathbb{C}$, and therefore we obtain $\vdash_{sig} \Sigma'$ by a single application of *sig:td*. The *sig:const* case is trivial, since constants are already in $\beta\eta$-normal form.

2. Let $\mathcal{E}:\Gamma \vdash_{\Sigma} \mathbf{A}::\mathbb{A}$, we only show that $\Gamma \vdash_{\Sigma'} \mathbf{A}::\mathbb{A}$, since the other direction is trivial. The only interesting cases are those, where $\mathcal{E}$ ends in *ws:td* or *ws:$\beta\eta$*, since *ws:var* is trivial, and *ws:app* and *ws:abs* are direct consequences of the inductive hypothesis. For the *ws:td* case let $\mathcal{E}$ end in
$$\frac{[\mathbf{A}::\mathbb{A}] \in \Sigma}{\Gamma \vdash_{\Sigma} \mathbf{A}::\mathbb{A}}\ \textit{ws:td}$$
and let $\mathbf{A}'$ be the $\beta\eta$-normal form of $\mathbf{A}$. Then by the construction we must have $[\mathbf{A}'::\mathbb{A}] \in \Sigma'$, so we can get $\vdash_{\Sigma'} \mathbf{A}'::\mathbb{A}$ by *ws:td*, and moreover $\vdash_{\Sigma'} \mathbf{A}=_{\beta\eta}\mathbf{A}'$ by 3. Finally, we obtain the assertion ($\vdash_{\Sigma'} \mathbf{A}::\mathbb{A}$) with 7.6.1. The case where $\mathcal{E}$ ends in *ws:$\beta\eta$* is similarly obtained with the inductive hypothesis and 3.

3. The assertion is a direct consequence of the inductive hypothesis for the inductive rules for congruence judgments. Furthermore, the preconditions of *ws:β:top*, *ws^r:η:top* and *ws^d:η:top* only consist of well-sortedness conditions, so the assertion is a simple consequence of 2.

Note that the first assertion is not true for long $\beta\eta$-normal forms, since we cannot add η-expanded forms of constants without declaring constants with a *sig:const* rule first. $\square$

$\Sigma \vdash \mathbb{A} \sqsubseteq \mathbb{B}\mathbb{A}$ *is a semantic subsort of* $\mathbb{B}$ *in* Σ
subsort

General Assumption 7.6.6 In the following we only consider signatures Σ such that in all term declarations $[\mathbf{A}::\mathbb{A}] \in \Sigma$ the formula $\mathbf{A}$ is a $\beta\eta$-normal form. As the previous lemma shows, this is only a syntactic restriction, since for any valid, unrestricted signature Σ we can give a valid, restricted signature Σ' that yields the same well-sorted formulae. Therefore this restriction do not amount to a limitation of the expressive power of $\Sigma\Lambda^{\rightarrow}$.

7.7 Sort Inclusion

We can make the following observation: if $[X::\mathbb{A}] \vdash_\Sigma X::\mathbb{B}$, then $wsf_{\mathbb{A}}(\Sigma, \Gamma) \subseteq wsf_{\mathbb{B}}(\Sigma, \Gamma)$. This is just the situation that is captured with the notion of sort inclusion in sorted logics without term declarations. In such systems the subsort relation is the smallest partial ordering that contains a set of subsort declarations. The subsort relation plays such a central role in these systems that they are collectively called "order-sorted". For specifying mathematics in $\Sigma\Lambda^{\rightarrow}$ the notion of subsorting also plays an important role, since it allows to specify taxonomic hierarchies of sorts, which, for instance, occur in the definitional hierarchies of mathematics and help to guide mathematical intuition. Since subsorting in $\Sigma\Lambda^{\rightarrow}$, where we have term declarations, is a derived relation, we do not have to treat it in our meta-logical development. On the object level (and for computation) however, it is a useful notion to employ.

Definition 7.7.1 (Subsort Relation)
We say that $\mathbb{A}$ is a **subsort** of $\mathbb{B}$ in Σ, and write $\Sigma \vdash \mathbb{A} \sqsubseteq \mathbb{B}$, iff $[X::\mathbb{A}] \vdash_\Sigma X::\mathbb{B}$.

In contrast to the first-order systems, the subsort relation in $\Sigma\Lambda^{\rightarrow}$ is not finite, even when we assume a finite set of base sorts. Thus the relation cannot be pre-computed in advance. Furthermore, it is not clear whether the sort-checking problem is decidable (see the discussion in section 8.3), which is another reason for the limited practical usefulness of the full subsorting relation. One way out of this situation is to define a computable partial ordering relation that approximates the full subsort relation by closing a set of subsort declarations under certain inductive principles that induce subsort relations on higher types from such for lower types.

For our approach it is only essential that the subsort declarations are correct, i.e. that $[X::\mathbb{A}] \vdash_\Sigma X::\mathbb{B}$, whenever $\mathbb{A} \leq \mathbb{B}$ is a subsort declaration. Completeness, i.e. that the subsort relation defined from the declarations captures *all* semantical subset relation, only plays a role for the effectiveness of actual computational algorithms, since the term declaration mechanism assures completeness of the calculi, even if we chose an empty set of term declarations.

approximation
$[\mathbb{A} \leq \mathbb{B}]$*subsort declaration*
$\mathcal{SD}(\Sigma)$*Set of subsort declarations*
subsort declarations
declaration!subsort
proper subsort declaration
partial subsort declaration

Definition 7.7.2 (Approximation of the Subsort Relation)
Let $\mathcal{R}$ be a typed binary relation on sorts such that $\mathbb{A} \sqsubseteq \mathbb{B}$ whenever $\mathcal{R}(\mathbb{A},\mathbb{B})$, then we call $\mathcal{R}$ an **approximation** of $\sqsubseteq$ in Σ.

We now identify special term declarations that we use to build a concrete approximation of $\sqsubseteq$.

Definition 7.7.3 (Subsort Declaration)
We call term declarations of the form $[\lambda X.X)::\mathbb{A}]$ **subsort declarations** and abbreviate them by $[\mathfrak{d}(\mathbb{A}) \leq \mathfrak{r}(\mathbb{A})]$. If $\mathbf{ln}(\mathbb{A}) > 0$, then we call $[\lambda X.X)::\mathbb{A}]$ a **proper subsort declaration** else a **partial one**. We denote the set of proper subsort declarations in Σ by $\mathcal{SD}(\Sigma)$.

The motivation for the division into proper and improper subsort declarations is that while all of them induce valid subsort judgments, only the proper ones can be eliminated from the signature in favor of a treatment with a specialized subsort inference system.

Lemma 7.7.4 *If* $[\mathbb{A} \leq \mathbb{B}] \in \Sigma$*, then* $\Sigma \vdash \mathbb{A} \sqsubseteq \mathbb{B}$*.*

Proof: Let $[\mathbb{A} \leq \mathbb{B}] := [(\lambda\overline{Y^k}.Y^i)::\mathbb{C}] \in \Sigma$, with $\mathfrak{r}^k(\mathbb{C}) = \mathbb{B}$ and $\mathfrak{d}^i(\mathbb{C}) = \mathbb{A}$. Furthermore, let $\Gamma = \overline{[X^k::\mathfrak{d}^k(\mathbb{C})]}$, then we have a $\Sigma\Lambda^{\rightarrow}$-derivation of the form

$$\cfrac{\cfrac{\cfrac{}{\Gamma \vdash_\Sigma (\lambda\overline{Y^k}.Y^i)::\mathbb{C}}\;ws{:}td}{\Gamma \vdash_\Sigma (\lambda\overline{Y^k}.Y^i)\overline{X^k}::\mathfrak{r}^k(\mathbb{C})}\;ws{:}app^k}{\Gamma \vdash_\Sigma X^i::\mathfrak{r}^k(\mathbb{C}) = \mathbb{B}}\;{}^{*}\,ws{:}\beta\eta$$

This yields the assertion, since $\Gamma(X^i) = \mathfrak{d}^i(\mathbb{C}) = \mathbb{A}$. □

Remark 7.7.5 (Variant Subsort Declarations)
We could also have defined a subsort relation $\Gamma \vdash_\Sigma \mathbb{A} \sqsubseteq \mathbb{B}$ by $\Gamma,[X::\mathbb{A}] \vdash_\Sigma X::\mathbb{B}$, and extended the set of subsort declarations to term declarations of the form $[(\lambda\overline{Y^k}.Y^i)::\mathbb{A}]$ for $[\mathfrak{d}^i(\mathbb{A}) \leq \mathfrak{r}^k(\mathbb{A})]$. The corresponding notion of proper subsort declaration is given by $k \leq \mathbf{ln}(\mathbb{A})$.

This would also have led to correct subsort declarations: If $[\mathbb{A} \leq \mathbb{B}] =: [(\lambda\overline{Y^k}.Y^i)::\mathbb{C}] \in \Sigma$, with $\mathfrak{r}^k(\mathbb{C}) = \mathbb{B}$ and $\mathfrak{d}^i(\mathbb{C}) = \mathbb{A}$ and $\Gamma = \overline{[X^k::\mathfrak{d}^k(\mathbb{C})]}$, then

$$\cfrac{\cfrac{\cfrac{}{\Gamma \vdash_\Sigma (\lambda\overline{Y^k}.Y^i)::\mathbb{C}}\;ws{:}td \qquad \cfrac{}{\Gamma \vdash_\Sigma X^j::\mathfrak{d}(j)\mathbb{C}}\;j \leq k}{\Gamma \vdash_\Sigma (\lambda\overline{Y^k}.Y^i)\overline{X^k}::\mathfrak{r}^k(\mathbb{C})}\;ws{:}app^k}{\Gamma \vdash_\Sigma X^i::\mathfrak{r}^k(\mathbb{C}) = \mathbb{B}}\;{}^{*}\,ws{:}\beta\eta$$

$\Sigma \vdash \mathbb{A} \leq_{\mathcal{R}} \mathbb{B}$*subsorting judgment for the set* $\mathcal{R}$ *of subsort declarations*
subsort inference system
inference system!subsort
contravariance rule
sort ordering

yields correctness, since $\Gamma(X^i) = \mathfrak{d}^i(\mathbb{C}) = \mathbb{A}$.

However, this version of subsorting is equivalent to the one we define only if all sorts are non-empty (in the derivation above, we could have used constants $c^j_{\mathfrak{d}^j(\mathbb{C})}$ to obtain $\Sigma \vdash \mathbb{A} \leq \mathbb{B}$). The meaning of $\Gamma \vdash_\Sigma \mathbb{A} \leq \mathbb{B}$, where $\Gamma = \overline{[X^n :: \mathbb{A}^n]}$ can be glossed as "if all sorts $\mathbb{A}$ are non-empty, then $\mathbb{A}$ is a subsort of $\mathbb{B}$".

Definition 7.7.6 (Sort Inclusion)
Let $\mathcal{R}$ be an approximation of $\sqsubseteq$, then the inference system in Figure 7.7 is called the $\Sigma\Lambda^{\rightarrow}$ **subsort inference system** for $\mathcal{R}$. For $\Sigma^r\Lambda^{\rightarrow}$ this system only has the inference rules over the line, $\Sigma^d\Lambda^{\rightarrow}$, it is augmented by the **contravariance rule** below the line. We call the relation $\mathcal{R}^{\leq}$, defined by $\mathcal{R}^{\leq}(\mathbb{A}, \mathbb{B})$, iff $\Sigma \vdash \mathbb{A} \leq_{\mathcal{R}} \mathbb{B}$, the **sort ordering** relation for $\mathcal{R}$.

Theorem 7.7.7 *If $\mathcal{R}$ is an approximation of the subsort relation of Σ, then the relation $\mathcal{R}^{\leq}$ is also an approximation. Moreover $\mathcal{R}^{\leq}$ is a quasi-ordering.*

Proof: We prove that $[X :: \mathbb{A}] \vdash_\Sigma X :: \mathbb{B}$ by induction on the structure of $\mathcal{D} : \Sigma \vdash \mathbb{A} \leq_{\mathcal{R}} \mathbb{B}$. Let us first consider the base cases. If $\mathcal{D}$ ends in *ior:start*, we obtain the assertion from the hypothesis, that $\mathcal{R}$ is an approximation, and the *ior:ref* case corresponds to the *ws:var* rule. For the *ior:nat* rule consider the following $\Sigma\Lambda^{\rightarrow}$-derivation,

$$\dfrac{\dfrac{\dfrac{}{[X :: \mathbb{A}] \vdash_\Sigma X :: \mathbb{A}}\,ws{:}var \quad \dfrac{}{[Y :: \mathfrak{d}(\mathbb{A})] \vdash_\Sigma Y :: \mathfrak{d}(\mathbb{A})}\,ws{:}var}{[X :: \mathbb{A}], [Y :: \mathfrak{d}(\mathbb{A})] \vdash_\Sigma XY :: \mathfrak{r}(\mathbb{A})}\,ws{:}app}{[X :: \mathbb{A}] \vdash_\Sigma (\lambda Y . XY) :: \mathfrak{d}(\mathbb{A}) \rightarrow \mathfrak{r}(\mathbb{A})}\,ws{:}abs$$

which yields $[X :: \mathbb{A}] \vdash_\Sigma X :: \mathfrak{d}(\mathbb{A}) \rightarrow \mathfrak{r}(\mathbb{A})$ by a single application of lemma 7.6.1.

For the inductive cases we see that the *ior:trans* rule can be recovered with the *ws:inst* rule: By inductive hypothesis we have $\mathcal{A} : \Gamma, [X :: \mathbb{A}] \vdash_\Sigma X :: \mathbb{B}$ and $\mathcal{B} : \Delta, [Y :: \mathbb{B}] \vdash_\Sigma Y :: \mathbb{C}$, so the following $\Sigma\Lambda^{\rightarrow}$-derivation yields the assertion.

$$\dfrac{\dfrac{\mathcal{A}}{[X :: \mathbb{A}] \vdash_\Sigma X :: \mathbb{B}} \quad \dfrac{\mathcal{B}}{[Y :: \mathbb{B}] \vdash_\Sigma Y :: \mathbb{C}}}{[X :: \mathbb{A}] \vdash_\Sigma X :: \mathbb{C}}\,ws{:}inst$$

For *ior:cov* let $\mathcal{A} : \Sigma \vdash \mathbb{A} \leq_{\mathcal{R}} \mathbb{B}$, then by inductive hypothesis we have a $\Sigma\Lambda^{\rightarrow}$-derivation $\mathcal{A}' : [Z :: \mathbb{A}] \vdash_\Sigma Z :: \mathbb{B}$. Furthermore, we have $[X :: \mathbb{C} \rightarrow \mathbb{A}], [Y :: \mathbb{C}] \vdash_\Sigma$

$XY::\mathbb{A}$ and thus

$$\dfrac{\dfrac{[X::\mathbb{C}\to\mathbb{A}],[Y::\mathbb{C}]\vdash_\Sigma XY::\mathbb{A}\quad \dfrac{\mathcal{A}'}{[Z::\mathbb{A}]\vdash_\Sigma Z::\mathbb{B}}}{[X::\mathbb{C}\to\mathbb{A}],[Y::\mathbb{C}]\vdash_\Sigma XY::\mathbb{B}}\,ws{:}inst}{[X::\mathbb{C}\to\mathbb{A}]\vdash_\Sigma (\lambda Y_{\bullet}XY)::\mathbb{C}\to\mathbb{B}}\,ws{:}abs$$

Just as above we can conclude $[X::\mathbb{A}\to\mathbb{C}]\vdash_\Sigma X::\mathbb{C}\to\mathbb{B}$ with lemma 7.6.1.

Finally, for ior^d:*contra* let $\mathcal{A}\colon\Sigma\vdash\mathbb{A}\leq_{\mathcal{R}}\mathbb{B}$, then by inductive hypothesis we have a $\Sigma\Lambda^{\to}$-derivation $\mathcal{A}'\colon[Z::\mathbb{A}]\vdash_\Sigma Z::\mathbb{B}$ and thus

$$\dfrac{\dfrac{\dfrac{\overline{[X::\mathbb{B}\to\mathbb{C}]\vdash_\Sigma X::\mathbb{B}\to\mathbb{C}}\quad \dfrac{\mathcal{A}'}{\Gamma,[Z::\mathbb{A}]\vdash_\Sigma Z::\mathbb{B}}}{\Gamma,[X::\mathbb{B}\to\mathbb{C}],[Z::\mathbb{A}]\vdash_\Sigma XZ::\mathbb{C}}\,ws{:}app}{\Gamma,[X::\mathbb{B}\to\mathbb{C}]\vdash_\Sigma(\lambda Z_{\alpha\bullet}XZ)::\mathbb{A}\to\mathbb{C}}\,ws^d{:}abs\quad *}{\Gamma,[X::\mathbb{B}\to\mathbb{C}]\vdash_\Sigma X::\mathbb{A}\to\mathbb{C}}\,ws{:}\beta\eta$$

which concludes the argument, since $[X::\mathbb{B}\to\mathbb{C}]\vdash_\Sigma(\lambda Z_{\alpha\bullet}XZ)\to_\eta X$ in $\Sigma^d\Lambda^{\to}$. Note that in $\Sigma^r\Lambda^{\to}$, we can only prove $[X::\mathbb{B}\to\mathbb{C}]\vdash_\Sigma(\lambda Z_{\mathbb{A}\bullet}XZ)\to_\eta X$, iff $[X::\mathbb{B}\to\mathbb{C}]\vdash_\Sigma X::\mathbb{A}\to\mathbb{C}$, which is just what we wanted to prove in the first place; thus we see that the contravariance rule is peculiar to $\Sigma^d\Lambda^{\to}$.

The claim that $\mathcal{R}^{\leq}$ is a quasi-ordering is an obvious consequence of the admissibility of *ior:trans* and *ior:ref*. □

Remark 7.7.8 (Variant Subsort Declarations (Continued))
For the variant definition of the subsort relation from Remark 7.7.5 we obtain an additional inference rule

$$\dfrac{\Sigma\vdash\mathbb{C}\to\mathbb{A}\leq_{\mathcal{R}}\mathbb{B}}{\Sigma\vdash\mathbb{A}\leq_{\mathcal{R}}\mathfrak{r}(\mathbb{B})}\,ior{:}cov^{-1}$$

that is admissible: let $\mathcal{A}\colon\Sigma\vdash(\mathbb{C}\to\mathbb{A})\leq_{\mathcal{R}}\mathbb{B}$. By inductive hypothesis, we have $\mathcal{A}'\colon\Gamma,[Z::\mathbb{C}\to\mathbb{A}]\vdash_\Sigma Z::\mathbb{B}$ and thus the following $\Sigma\Lambda^{\to}$-derivation $\mathcal{B}$

$$\dfrac{\dfrac{\overline{[X::\mathbb{C}],[Y::\mathbb{A}]\vdash_\Sigma Y::\mathbb{A}}\,ws{:}var}{[Y::\mathbb{A}]\vdash_\Sigma\lambda X_{\bullet}Y::\mathbb{C}\to\mathbb{A}}\,ws{:}abs\quad\dfrac{\mathcal{A}'}{\Gamma,[Z::\mathbb{C}\to\mathbb{A}]\vdash_\Sigma Z::\mathbb{B}}}{\Gamma,[Y::\mathbb{A}]\vdash_\Sigma\lambda X_{\bullet}Y::\mathbb{B}}\,ws{:}inst$$

$\Sigma \vdash \mathbb{A} \leq \mathbb{B}$ *concrete approximation of the subsort relation*
weakening rule

Thus the following $\Sigma\Lambda^{\rightarrow}$-derivation yields the assertion

$$\cfrac{\cfrac{\cfrac{\mathcal{B}}{\Gamma, [Y :: \mathbb{A}] \vdash_\Sigma (\lambda X.Y) :: \mathbb{B}} \qquad \cfrac{}{[W :: \mathfrak{d}(\mathbb{B})] \vdash_\Sigma W :: \mathfrak{d}(\mathbb{B})}\, ws{:}var}{\Gamma, [Y :: \mathbb{A}], [W :: \mathfrak{d}(\mathbb{B})] \vdash_\Sigma (\lambda X.Y) W :: \mathfrak{r}(\mathbb{B})}\, ws{:}app}{\Gamma, [Y :: \mathbb{A}], [W :: \mathfrak{d}(\mathbb{B})] \vdash_\Sigma Y :: \mathfrak{r}(\mathbb{B})}\, {}^{*}\, ws{:}\beta\eta$$

Note that the rule *ior*:*cov*$^{-1}$ cannot be derived for the original definition of subsorting, since we cannot get rid of the W^i in the context. Furthermore *ior*:*cov*$^{-1}$ is only semantically sound (as inclusion on the sets of well-formed formulae, if $\mathfrak{d}(\mathbb{B})$ is non-empty).

Notation 7.7.9 As a consequence of the previous theorem, the subsort relation is a partial ordering and closed under *ior*:*cov* and *ior*:*nat*, since it also is an approximation of $\sqsubseteq$. We have seen in 7.7.4 the set of subsort declarations for an approximation of $\sqsubseteq$. Thus the relation given by the judgment $\Sigma \vdash \mathbb{A} \leq_{\mathcal{SD}(\Sigma)} \mathbb{B}$ is an approximation of $\sqsubseteq$, which we simply denote by $\Sigma \vdash \mathbb{A} \leq_\Sigma \mathbb{B}$.

Definition 7.7.10 (Weakening)
Let $\mathcal{R}$ be an approximation of the subsort relation in Σ, then the following inference rule is called the **weakening rule** for $\mathcal{R}$:

$$\boxed{\cfrac{\Gamma \vdash_\Sigma \mathbf{A} :: \mathbb{A} \quad \Gamma \vdash_\Sigma \mathbf{A} :: \mathbb{B}}{\Sigma \vdash \mathbb{A} \leq_{\mathcal{R}} \mathbb{B}}\, ws{:}weaken(\mathcal{R})}$$

As above we denote the inference rule *ws*:*weaken*$(\mathcal{SD}(\Sigma))$ just by *ws*:*weaken*.

Lemma 7.7.11 *Let $\mathcal{R}$ be an approximation of the subsort relation, then*

1. *ws*:*weaken*$(\mathcal{R})$ *is admissible in* $\Sigma\Lambda^{\rightarrow}$.
2. *A formula* $\mathbf{A}$ *is well-sorted, iff the set of sorts of* $\mathbf{A}$ *is a nonempty upper segment for* $\mathcal{R}^{\leq}$.

Proof: By *ws*:*inst* and *ws*:*weaken*$(\mathcal{R})$ and the relevant definitions. □

Definition 7.7.12 ($\Sigma\Lambda^{\rightarrow}$ Augmented with Subsorts)
Let Σ be a signature and $\mathcal{SD}(\Sigma)$ a set of subsort declarations. Then we can get a sorted λ-calculus with subsorts by considering the judgment $\Gamma \vdash_\Sigma^\leq \mathbf{A} :: \mathbb{A}$ and the inference system of of $\Sigma\Lambda^{\rightarrow}$ augmented with *ws*:*weaken* and the subsort inference system for $\mathcal{SD}(\Sigma)$.

In the next theorem we show that this augmented system is equivalent to $\Sigma\Lambda^{\rightarrow}$, in the sense that we can simulate all judgments in $\Sigma\Lambda^{\rightarrow}$.

Theorem 7.7.13 (Subsort Theorem)
Let $\Sigma = \Sigma' \cup \mathcal{SD}(\Sigma)$, then we have

1. $\vdash_{sig} \Sigma$, *iff* $\vdash^a_{sig} \Sigma'$
2. $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$, *iff* $\Gamma \vdash^a_{\Sigma'} \mathbf{A}{::}\mathbb{A}$.
3. $\Gamma \vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{B}$, *iff* $\Gamma \vdash^a_{\Sigma'} \mathbf{A}{=}_{\beta\eta}\mathbf{B}$.

Proof: We will prove the assertions by simultaneous induction on the derivations involved.

For the first assertion, we only have to show that $\vdash_{sig} \Sigma$ entails $\vdash^a_{sig} \Sigma'$, since the other direction is trivially met by construction. The only interesting case is the *sig:td* case, where

$$\dfrac{\dfrac{\mathcal{A}}{\vdash_{sig} \Sigma} \quad \dfrac{\mathcal{B}}{\vdash_\Sigma \mathbf{A}{::}\mathbb{A}} \quad \mathbb{A} \parallel \mathbb{B}}{\vdash_{sig} \Sigma, [\mathbf{A}{::}\mathbb{B}]}\ sig{:}td$$

By inductive hypothesis for $\mathcal{A}$ we have $\mathcal{A}'{:}\vdash^a_{sig} \Sigma'$ and $\mathcal{B}'{:}\vdash^a_{\Sigma'} \mathbf{A}{::}\mathbb{A}$. If $[\mathbf{A}{::}\mathbb{B}] \in \mathcal{SD}(\Sigma)$, then $\mathcal{A}'{:}\vdash^a_{sig} \Sigma'$ is just the assertion, otherwise we can obtain it by a simple application of *sig:td* to $\mathbf{A}$ and $\mathbf{B}$.

For the second assertion let $\mathcal{A}{:}\,\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$. Here the only interesting case in the inductive proof of $\Gamma \vdash^a_{\Sigma'} \mathbf{A}{::}\mathbb{A}$ is the *ws:td* case, where $[\mathbf{A}{::}\mathbb{A}] \in \Sigma$. Now, if $[\mathbf{A}{::}\mathbb{A}]$ is not a subsort declaration, then it is in Σ' and there is nothing to show, otherwise it is of the form $[\lambda X.X{::}\mathbb{A}]$, where $\mathbb{A} = \mathbb{B} \to \mathbb{C}$, since the declaration is proper by definition. We can prove the assertion by

$$\dfrac{\dfrac{\dfrac{}{[X{::}\mathbb{B}] \vdash^a_{\Sigma'} X{::}\mathbb{C}}\ ws{:}var \quad \Sigma' \vdash \mathbb{B} \le \mathbb{C}}{\Gamma \vdash^a_{\Sigma'} X{::}\mathfrak{r}^k(\mathbb{C})}\ ws{:}weaken}{\vdash^a_{\Sigma'} \lambda\overline{Y^k}.Y^i{::}\mathbb{A}}\ ws{:}abs$$

For the converse direction is a direct consequence of the admissibility of *ws:weaken* and Lemma 7.7.4.

Finally, the third assertion can be proven by a trivial induction. □

Example 7.7.14 (Improper Subsort Declarations)
Note that the result of the subsort theorem is sharp, since there is no way of eliminating improper subsort declarations in favor of the subsort judgment. For instance let $\mathbb{C}$ be a functional base sort with $\mathfrak{d}(\mathbb{C}) = \mathbb{A}$ and $\mathfrak{r}(\mathbb{C}) = \mathbb{B}$, then the declaration $[\lambda X.X{::}\mathbb{C}]$ is stronger than the subsort declaration $[\mathbb{A} \le \mathbb{B}]$. In particular we have $[F{::}\mathbb{C} \to \mathbb{A}] \vdash_\Sigma F(\lambda X.X)$, but the judgment $[F{::}\mathbb{C} \to \mathbb{A}] \vdash^a_{\Sigma'} F(\lambda X.X)$ is not derivable.

This result tells us that we can use either of the formulations of $\Sigma\Lambda^{\rightarrow}$ interchangeably. In particular it allows us to keep the simple and mathematically elegant system $\Sigma\Lambda^{\rightarrow}$ for deriving results about the logical system $\Sigma\Lambda^{\rightarrow}$, while resorting to the more intuitive system $\Sigma\Lambda^{\rightarrow a}$ for actual computation.

Theorem 7.7.15 *Let $\mathcal{R}$ be an approximation of the subsort relation in Σ, then $\mathbb{A} \parallel \mathbb{B}$ whenever $\Sigma \vdash \mathbb{A} \leq_{\mathcal{R}} \mathbb{B}$.*

Proof: The assertion is a consequence of well-typedness for $\Sigma^d\Lambda^{\rightarrow}$, so let us only consider $\Sigma^r\Lambda^{\rightarrow}$: By theorem 7.7.7 we have $\Gamma, [X{::}\mathbb{A}] \vdash_\Sigma X{::}\mathbb{B}$, but we also have $[X{::}\mathbb{A}] \vdash_\Sigma X{::}\mathbb{A}$ by *ws:var*, therefore $\mathbb{A}\ \mathbf{Rdom}\ \mathbb{B}$ by 7.5.1. □

Remark 7.7.16 As a consequence the sets $\mathcal{S}_\alpha$ are mutually incomparable, i.e. if $\mathbb{A} \in \mathcal{S}_\alpha$, $\mathbb{B} \in \mathcal{S}_\beta$, and $\alpha \neq \beta$, then we can never have $\Gamma \vdash_\Sigma \mathbb{A} \sqsubseteq \mathbb{B}$. The most important consequence of this is that we can only have finite ascending and descending chains of sorts with respect to $\sqsubseteq$.

Lemma 7.7.17 (Decidability)
If $\mathcal{R}$ is a decidable approximation of the subsort relation in Σ, then $\leq_{\mathcal{R}}$ is decidable.

Proof: First note that $\leq$ is a typed relation, thus we can assume that $\alpha = \tau(\mathbb{A}) = \tau(\mathbb{B})$ if we want to decide $\mathbb{A} \leq_{\mathcal{R}} \mathbb{B}$. We prove the assertion by an induction over the structure of type α.

If α is a base type, then $\leq_{\mathcal{R}}$ is induced from $\mathcal{R}$ by *ior:ref*, *ior:trans*, which is decidable since the set of sorts of type α is finite. If $\alpha = \beta \rightarrow \gamma$, then by inductive hypothesis, $\leq_{\mathcal{R}}$ is decidable on sorts of types β and γ, so we can compute the contribution of *ior:cov* (and *ior^d:contra*). $\leq_{\mathcal{R}}$ is just the transitive-reflexive closure of that, which is computable, since the set of sorts of type α is finite. □

In particular, if $\mathcal{R}$ is the relation generated by the set of subsort declarations (which we can decide by the form of the term declarations), then we obtain:

Corollary 7.7.18 $\leq_\Sigma$ *is decidable.*

General Assumption 7.7.19 Let Σ be a sorted higher-order signature, and let $\sim$ be the equivalence relation induced by $\leq_\Sigma$, that is, $\Sigma \vdash \mathbb{A} \sim \mathbb{B}$, iff $\Sigma \vdash \mathbb{A} \sqsubseteq \mathbb{B}$ and $\Sigma \vdash \mathbb{B} \sqsubseteq \mathbb{A}$. Observe that for $\Sigma \vdash \mathbb{A} \sim \mathbb{B}$ we can derive $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ whenever we can derive $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{B}$ by application of the *ws:weaken* inference rule.

If we pass to the quotient signature Σ' with respect to $\sim$, that is, for any equivalence class in $\mathcal{S}$ we pick a representative and replace sorts by their representative, then we get a signature Σ', where $\sqsubseteq'$ is a partial ordering,

contravariance in the domain sort
top sort
bottom sort

which entails that $\sim'$ is trivial. Furthermore, Σ' is valid whenever Σ is, since the inference system for valid signatures is only concerned about sorts in compatibility of term declarations, and we have $\mathbb{A} \parallel \mathbb{B}$ whenever $\mathbb{A} \sim \mathbb{B}$ by 7.7.15.

If we always take care to pick a representative of maximal length, then we can never have $\mathfrak{d}(\mathbb{C}) \to \mathfrak{r}(\mathbb{C}) \sqsubset \mathbb{C}$ for any base sort $\mathbb{C} \in \mathcal{BS}^f$, since we always have the converse by *ior:nat*. Thus we can assume that $\sqsubseteq$ is a partial ordering with $\mathbb{C} \sqsubset \mathfrak{d}(\mathbb{C}) \to \mathfrak{r}(\mathbb{C})$ for all functional base sorts $\mathbb{C} \in \mathcal{BS}^f$. While we will not need this assumption in the following, other signatures would in some sense be redundant and thus non-optimal in practice.

Remark 7.7.20 (Contravariance and Function Restriction)
In $\Sigma^r\Lambda^{\to}$, the rule for **contravariance in the domain sort** which corresponds to function restriction cannot be admissible, since it contradicts 7.7.15. This defect in symmetry is intended, since we want functions to have unique supporting sorts (cf. 7.5.7) in $\Sigma^r\Lambda^{\to}$. The natural notion of semantics for Church-style λ-calculi for mathematics with covariance seems to be a total function semantics, where a declaration of the form $[f{::}\mathbb{A} \to \mathbb{B}]$ has the intended semantics that the denotation of f is a total (on some predefined universe) function that, when restricted to some subdomain $\mathcal{D}_{\mathbb{A}}$, yields values in the subset $\mathcal{D}_{\mathbb{B}}$ of the universe. In $\Sigma^r\Lambda^{\to}$ we do not have to treat function restriction as a built-in implicit notion, since we can make it explicit: for any formula $\mathbf{A}$ of function sort $\mathbb{A}$, the function $\mathcal{I}_\varphi(\lambda X_{\mathbb{B}}.\mathbf{A}X)$ is the restriction of $\mathcal{I}_\varphi(\mathbf{A})$ to the subset $\mathcal{D}_{\mathbb{B}} \subseteq \mathcal{D}_{\mathfrak{d}(\mathbb{A})}$, if $\mathbb{B}$ is a subsort of $\mathfrak{d}(\mathbb{A})$.

Remark 7.7.21 (Top Sorts)
[35] In $\Sigma^d\Lambda^{\to}$ we can pick for each base type $\alpha \in \mathcal{BT}$ a **top sort** $\mathbb{T}\mathsf{op}_\alpha$ and a **bottom sort** $\mathbb{B}\mathsf{ot}_\alpha$. To specify the intended behavior we would need subsort declarations $[\mathbb{B}\mathsf{ot}_\alpha \leq \mathbb{A}]$ and $[\mathbb{A} \leq \mathbb{T}\mathsf{op}_\alpha]$ in the signature for any non-functional base sort $\mathbb{A}$. Note that this set of declarations is finite, since the set of base sorts is. EdNote(35)

In the case of functional sorts, we will use $\mathbb{T}\mathsf{op}_{\alpha\to\beta}$ as an abbreviation for $\mathbb{B}\mathsf{ot}_\alpha \to \mathbb{T}\mathsf{op}_\alpha$ and $\mathbb{B}\mathsf{ot}_{\alpha\to\beta}$ as one for $\mathbb{T}\mathsf{op}_\alpha \to \mathbb{B}\mathsf{ot}_\alpha$. With *ior:cov* and *ior^d:contra* it is easy to see that they have the intended behavior, and the following rules are admissible in $\Sigma^d\Lambda^{\to}$.

$$\boxed{\frac{\tau(\mathbb{A}) = \alpha}{\Sigma \vdash \mathbb{A} \leq_{\mathcal{R}} \mathbb{T}\mathsf{op}_\alpha}\,ior{:}top \qquad \frac{\tau(\mathbb{A}) = \alpha}{\Sigma \vdash \mathbb{B}\mathsf{ot}_\alpha \leq_{\mathcal{R}} \mathbb{A}}\,ior{:}bot}$$

will sometimes use this setup.
Since in many examples, it is convenient to have top and bottom sorts, we
Note that in $\Sigma^r\Lambda^{\to}$, it is not possible to define global top sorts for functional types, since by the remark above (7.7.20) the subsort relation respects function domains, and we would get one top sort for any domain sort.

[35] EdNote: totally rework wrt. PER sorts

non-negative
function!monomial
monomial function

We will conclude this section with an example from mathematics and discuss it in more details.

Example 7.7.22 (Elementary Calculus)
Suppose we want to use the sorts to model the world of elementary analysis. This means we want to include the sets of real numbers ($\mathbb{R}$), the non-negative real numbers ($\mathbb{R}^+$) with $\tau(\mathbb{R}) = \tau(\mathbb{R}^+) = \iota$.

Furthermore, let $\mathbb{M}, \mathbb{P}, \mathbb{D}, \mathbb{C}$ be sorts of type $\iota \to \iota$ such that all have domain and codomain sorts $\mathbb{R}$. These have the intended meanings of monomials, polynomials, differentiable and continuous functions on the reals. Finally, let Σ be the signature with the following term declarations (for $\Sigma^r\Lambda^{\to}$ add the appropriate domain sorts to the bound variables):

$$
\begin{array}{l}
[+::\mathbb{R} \to \mathbb{R} \to \mathbb{R}], [+::\mathbb{R}^+ \to \mathbb{R}^+ \to \mathbb{R}^+],\\
[*::\mathbb{R} \to \mathbb{R} \to \mathbb{R}], [*::\mathbb{R}^+ \to \mathbb{R}^+ \to \mathbb{R}^+], [\lambda X\textbf{.} * XX::\mathbb{R} \to \mathbb{R}^+],\\
[\mathbb{R}^+ \leq \mathbb{R}], [\mathbb{M} \leq \mathbb{P}], [\mathbb{P} \leq \mathbb{D}], [\mathbb{D} \leq \mathbb{C}]\\
[\lambda X\textbf{.}X::\mathbb{M}], [\lambda X, Y\textbf{.}X::\mathbb{R} \to \mathbb{M}], [\lambda F, G\textbf{.}\lambda X\textbf{.} * (FX)(GX)::\mathbb{M} \to \mathbb{M} \to \mathbb{M}],\\
[\lambda F, G\textbf{.}\lambda X\textbf{.} + (FX)(GX)::\mathbb{P} \to \mathbb{P} \to \mathbb{P}],\\
[\partial::\mathbb{D} \to \mathbb{C}], [\partial::\mathbb{P} \to \mathbb{P}], [\partial::\mathbb{M} \to \mathbb{M}].
\end{array}
$$

We claim that Σ formalizes a small fragment of elementary calculus; The first four term declarations give information about elementary behavior of addition and multiplication on non-negative real numbers. We could have added a sort $\mathbb{N}$ of non-positive reals, and could have added more declarations, but this will suffice for our examples. The fifth declaration is more complex, it says that squares are always non-negative.

The subsort declarations in line three specify the well-known inclusions in elementary calculus: non-negative reals are reals, monomials are polynomials, which are differentiable and (as such) continuous functions.

Line four specifies the set of monomial function to include the identity and the constant functions (apply $\lambda X, Y\textbf{.}X$ to a real number r to get a monomial $\lambda X\textbf{.}r$) and states that the sort of monomials is closed under pointwise multiplication. Similarly the declaration on line five states that the sort of polynomials is closed under pointwise addition (it contains the monomials by the subsort declaration $\mathbb{M} \leq \mathbb{P}$). Note that with this signature the sort of monomials and polynomials could contain more functions ($\Sigma\Lambda^{\to}$ cannot restrict the elements of sorts), but in our intended application they contain exactly the functions generated by these declarations.

The last line of the signature contains information about the differentiation operator, which usually maps differentiable functions to continuous ones. If we know the argument to be a polynomial, or even a monomial, the result will also be one. Note that we have made use of $\Sigma^d\Lambda^{\to}$ for the formalization here. Following 7.2.12 we could have used $\Sigma^r\Lambda^{\to}$ with the term declarations $[\partial::\mathbb{D} \to \mathbb{C}], [\partial{:}\!{:}\mathbb{P} \to \mathbb{P}], [\partial{:}\!{:}\mathbb{M} \to \mathbb{M}]$ instead.

To sum up, we have seen that the mechanism of term declarations allows to encode a great deal of (positive) knowledge about the specific domain of formalization into the signature. In the next section, we will see how this knowledge can be made use of in the inference process (higher-order unification), to arrive at more guided inference processes.

$$\frac{\Gamma, [X::\mathbb{B}] \vdash_\Sigma \mathbf{A}::\mathbb{A} \quad \Gamma \vdash_\Sigma \mathbf{B}::\mathbb{B}}{\Gamma \vdash_\Sigma (\lambda X.\mathbf{A})\mathbf{B} \rightarrow^t_\beta [\mathbf{B}/X]\mathbf{A}}\ \mathit{ws{:}\beta{:}top}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A} \quad X \notin \mathbf{Free}(\mathbf{A})}{\Gamma \vdash_\Sigma (\lambda X_{\mathfrak{d}(\mathbb{A})}.\mathbf{A}X) \rightarrow^t_\eta \mathbf{A}}\ \mathit{ws^r{:}\eta{:}top}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A} \quad X \notin \mathbf{Free}(\mathbf{A}) \quad \alpha = \tau(\mathfrak{d}(\mathbb{A}))}{\Gamma \vdash_\Sigma (\lambda X_\alpha.\mathbf{A}X) \rightarrow^t_\eta \mathbf{A}}\ \mathit{ws^d{:}\eta{:}top}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow^t_{\beta\eta} \mathbf{B}}{\Gamma \vdash_\Sigma \Gamma \vdash_\Sigma \mathbf{A} \rightarrow^1_{\beta\eta} \mathbf{B}}\ \mathit{tr{:}start} \qquad \frac{\Gamma, [X::\mathbb{A}] \vdash_\Sigma \mathbf{A} \rightarrow_{\beta\eta} \mathbf{B}}{\Gamma \vdash_\Sigma \lambda X.\mathbf{A} \rightarrow_{\beta\eta} \lambda X.\mathbf{B}}\ \mathit{tr{:}abs}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow^1_{\beta\eta} \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A} \quad \Gamma \vdash_\Sigma \mathbf{C}::\mathfrak{d}(\mathbb{A})}{\Gamma \vdash_\Sigma \mathbf{AC} \rightarrow^1_{\beta\eta} \mathbf{BC}}\ \mathit{tr{:}app{:}fn}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{C} \rightarrow_{\beta\eta} \mathbf{D} \quad \Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A} \quad \Gamma \vdash_\Sigma \mathbf{C}::\mathfrak{d}(\mathbb{A})}{\Gamma \vdash_\Sigma \mathbf{AC} \rightarrow^1_{\beta\eta} \mathbf{AD}}\ \mathit{tr{:}app{:}arg}$$

$$\frac{\Sigma \vdash \mathbf{A} \rightarrow^1_{\beta\eta}}{\Sigma \vdash \mathbf{A} \rightarrow_{\beta\eta} \mathbf{B}}\ \mathit{ms{:}start} \qquad \frac{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}}{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow_{\beta\eta} \mathbf{A}}\ \mathit{ms{:}ref}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow_{\beta\eta} \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{B} \rightarrow_{\beta\eta} \mathbf{C}}{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow_{\beta\eta} \mathbf{C}}\ \mathit{ms{:}trans}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \rightarrow_{\beta\eta} \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B}}\ \mathit{eq{:}start} \qquad \frac{\Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{B} =_{\beta\eta} \mathbf{A}}\ \mathit{eq{:}sym}$$

Figure 7.5: Sorted $\beta\eta$-Reduction

$$\frac{}{\emptyset \vdash_\Sigma \emptyset::\emptyset}\ \mathit{wsub{:}start}$$

$$\frac{\Gamma \vdash_\Sigma \sigma::\Delta \quad \Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A} \quad X \notin \mathbf{Dom}(\Gamma)}{\Gamma \vdash_\Sigma \sigma, [\mathbf{A}/X]::\Delta, [X::\mathbb{A}]}\ \mathit{wsub{:}ext}$$

Figure 7.6: Well-Sorted Substitutions

$$\frac{\mathcal{R}(\mathbb{A},\mathbb{B}) \quad \vdash_{sig} \Sigma}{\Sigma \vdash \mathbb{A} \leq_{\mathcal{R}} \mathbb{B}}\,\mathit{ior{:}start} \qquad \frac{\vdash_{sig} \Sigma}{\Sigma \vdash \mathbb{A} \leq_{\mathcal{R}} \mathbb{A}}\,\mathit{ior{:}ref}$$

$$\frac{\vdash_{sig} \Sigma \quad \mathbb{A} \in \mathcal{BS}^f}{\Sigma \vdash \mathbb{A} \leq_{\mathcal{R}} \mathfrak{d}(\mathbb{A}) \to \mathfrak{r}(\mathbb{A})}\,\mathit{ior{:}nat} \qquad \frac{\Sigma \vdash \mathbb{A} \leq_{\mathcal{R}} \mathbb{B} \quad \Sigma \vdash \mathbb{B} \leq_{\mathcal{R}} \mathbb{C}}{\Sigma \vdash \mathbb{A} \leq_{\mathcal{R}} \mathbb{C}}\,\mathit{ior{:}trans}$$

$$\frac{\Sigma \vdash \mathbb{A} \leq_{\mathcal{R}} \mathbb{B}}{\Sigma \vdash \mathbb{C} \to \mathbb{A} \leq_{\mathcal{R}} \mathbb{C} \to \mathbb{B}}\,\mathit{ior{:}cov} \qquad \frac{\Sigma \vdash \mathbb{A} \leq \mathbb{B}}{\Sigma \vdash \mathbb{B} \to \mathbb{C} \leq \mathbb{A} \to \mathbb{C}}\,\mathit{ior}^d\mathit{{:}contra}$$

Figure 7.7: $\Sigma\Lambda^{\to}$ subsort inference system for $\mathcal{R}$

Chapter 8

Computational Aspects of $\Sigma\Lambda^{\rightarrow}$

In this chapter we investigate computational properties and algorithms for the judgments defined in Chapter 7 and take a look at the Σ-unification and Σ-matching problems extending the methods from Chapter 3. The main difference to the unsorted case will be the form of the general bindings, which turn out to be more complex than in the simply typed case. There, the general bindings follow the normal form theorem, which for long $\beta\eta$-normal forms gives a tight correspondence between the types of the formula and its head and the structure. In $\Sigma\Lambda^{\rightarrow}$, due to the presence of functional base sorts and term declarations, this correspondence is less direct. In particular, the sort of a formula is no longer tied to the sort of the head, but may depend on the whole structure of the formula. To make the relation of the sort and the structure of a formula explicit, we prove a *structure theorem*, which guarantees typing derivations of a special form, from which we can derive an appropriate concept of general binding.

We have seen that subsorting can be treated as a defined notion in $\Sigma\Lambda^{\rightarrow}$, but also (see the subsort theorem 7.7.13) that certain term declarations can be dropped from the signature, in favor of subsort reasoning. Similarly, the sorted higher-order unification algorithms presented in this chapter can be extended by subsort reasoning, allowing to drop the proper subsort declarations from the signature (and thus restricting the set of general bindings). To keep the presentation simple, we will not pursue this here and refer the reader to [Kle96].

For the purposes of this chapter, we will specialize the inference rules for valid signatures presented in Figure 7.4 on page ccxvi by the rules in Figure 8.1. The motivation of these rules is to force the existence of type derivations for $\mathbf{A}{::}\mathfrak{d}(\mathbb{A}) \to \mathfrak{r}(\mathbb{A})$ for any new term declaration $[\mathbf{A}{::}\mathbb{A}]$ added the signature, where $\mathbb{A}$ has a functional base sort as target sort. The original *sig:td* rule is split into two cases appropriate to the target sort. The constant rule is augmented so that only sorts with non-functional target can be specified for initial constant declarations (other sorts can be used in term declarations with *sig:td:f*).

Obviously, the new rules are special cases of the rules in Figure 7.4, so that every signature that is valid with these rules is valid under the old rules, so the theory in chapter 7 remains valid.

$$\frac{}{\vdash_{sig} \emptyset}\, sig{:}empty$$

$$\frac{\vdash_{sig} \Sigma \quad c \notin \Sigma \quad \mathbb{A} \in \mathcal{S} \quad \mathbf{ln}(\tau(\mathbb{A})) = \mathbf{ln}(\mathbb{A})}{\vdash_{sig} \Sigma, [c{::}\mathbb{A}]}\, sig{:}const'$$

$$\frac{\vdash_{\Sigma} \mathbf{A}{::}\overline{\mathbb{D}} \to \mathbb{B} \quad \tau(\mathbb{A}) = \tau(\mathbb{B}) \in \mathcal{BT}}{\vdash_{sig} \Sigma, [\mathbf{A}{::}\overline{\mathbb{D}} \to \mathbb{A}]}\, sig{:}td{:}nf$$

$$\frac{\vdash_{\Sigma} \mathbf{A}{::}\overline{\mathbb{D}} \to \mathfrak{d}(\mathbb{A}) \to \mathfrak{r}(\mathbb{A}) \quad \mathbb{A} \in \mathcal{BS}^f}{\vdash_{sig} \Sigma, [\mathbf{A}{::}\overline{\mathbb{D}} \to \mathbb{A}]}\, sig{:}td{:}f$$

Figure 8.1: Special Signature Rules

8.1 Structure Theorem

The key tool for the investigation of well-sorted formulae is the structure theorem, which we prove in this section. The principal difficulty of $\Sigma\Lambda^{\rightarrow}$ is that the property of well-sortedness is highly non-structural, which makes the classical deduction methods, such as unification, that analyze the structure of formulae difficult. The structure theorem recovers structural properties of well-sorted formulae by linking the sort information (the existence of certain term declarations) with structural information about normal forms.

$$\frac{\dfrac{\dfrac{\mathcal{H}}{\Gamma, \Xi^l \vdash_{\Sigma} \mathbf{H}{::}\mathbb{H}} \quad \dfrac{\mathcal{D}^i}{\Gamma, \Xi^l \vdash_{\Sigma} \mathbf{D}^i{::}\mathfrak{d}^i(\mathbb{H})}\, i \leq m}{\Gamma, \Xi^l \vdash_{\Sigma} \mathbf{H}\overline{\mathbf{D}^m}{::}\mathfrak{r}^l(\mathbb{A})}\, ws{:}app^m}{\Gamma \vdash_{\Sigma} \lambda \overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}{::}\mathbb{A}}\, ws{:}abs^l$$

Figure 8.2: Structural $\Sigma\Lambda^{\rightarrow}$-Derivation

structural
form!structural
principal subderivation
principal declaration
declaration!principal
semi-structural
form!semi-structural
structure formula
formula!structure-

Definition 8.1.1 (Structural $\Sigma\Lambda^{\rightarrow}$-Derivation)
We say that a $\Sigma\Lambda^{\rightarrow}$-derivation $\mathcal{A}\colon\Gamma\vdash_{\Sigma}\mathbf{A}::\mathbb{A}$ is in **structural** form, iff $\mathcal{A}$ is of the form shown in Figure 8.2 where

1. $l=\mathbf{ln}(\mathbb{A})$ and $\mathfrak{r}^{m}(\mathbb{H})=\mathfrak{r}^{l}(\mathbb{A})$
2. $\mathcal{H}$ either consists of either
   - a single *ws:var* step and $m=l+\mathbf{ln}(\tau(\mathbb{H}))-\mathbf{ln}(\tau(\mathbb{A}))\geq\mathbf{ln}(\mathbb{H})$.
   - or *ws:td* step and $m=\mathbf{ln}(\mathbb{H})$.
3. Ξ^{j} is the variable context $[X^{1}::\mathfrak{d}(\mathbb{A})],\ldots,[X^{j}::\mathfrak{d}^{j}(\mathbb{A})]$,

The subderivations $\mathcal{D}^{i}$ are called the **principal subderivation**s of $\mathcal{A}$ and the declaration $[\mathbf{H}\colon\mathbb{H}]$ the **principal declaration** of $\mathcal{A}$.

Note that not all sort judgments have a structural $\Sigma\Lambda^{\rightarrow}$-derivation

Example 8.1.2 Let $\Sigma=[c::\mathbb{A}],[\lambda Z.Z::\mathbb{H}]$, where $\mathbb{H}$ is a functional base sort with $\mathfrak{d}(\mathbb{B})=\mathbb{A}\in\mathcal{BS}^{nf}$ and $\mathfrak{r}(\mathbb{B})=\mathbb{C}\in\mathcal{BS}^{nf}$. Then the judgment $\vdash_{\Sigma}c::\mathbb{C}$ cannot have structural a $\Sigma\Lambda^{\rightarrow}$-derivation, since $l=m=0$ and therefore $[c::\mathbb{A}]$ is the only possible principal declaration, but this does not lead to the intended judgment. However, the judgment is derivable

$$\dfrac{\dfrac{\dfrac{}{\vdash_{\Sigma}\lambda Z.Z::\mathbb{H}}\,ws{:}td\qquad\dfrac{}{\vdash_{\Sigma}c::\mathbb{A}}\,ws{:}td}{\vdash_{\Sigma}(\lambda Z.Z)c::\mathbb{C}}\,ws{:}app\qquad\vdash_{\Sigma}(\lambda Z.Z)c{=_{\beta\eta}}c}{\vdash_{\Sigma}c::\mathbb{C}}\,ws{:}\beta\eta$$

This derivation is not structural, since we needed to apply $\beta\eta$-equality after a structural $\Sigma\Lambda^{\rightarrow}$-part.

We will show that this is indeed the worst that can happen, and formalize it in the concept of a semi-structural $\Sigma\Lambda^{\rightarrow}$-derivation.

Definition 8.1.3 (Semi-Structural $\Sigma\Lambda^{\rightarrow}$-Derivation)
We call a $\Sigma\Lambda^{\rightarrow}$-derivation $\mathcal{A}\colon\Gamma\vdash_{\Sigma}\mathbf{A}::\mathbb{A}$ in **semi-structural** form, iff $\mathcal{A}$ is of the form

$$\boxed{\dfrac{\dfrac{\mathcal{B}}{\Gamma\vdash_{\Sigma}\lambda\overline{X^{l}}.\mathbf{H}\overline{\mathbf{D}^{m}}::\mathbb{A}}\qquad\Gamma\vdash_{\Sigma}\lambda\overline{X^{l}}.\mathbf{H}\overline{\mathbf{D}^{m}}{=_{\beta\eta}}\mathbf{A}}{\Gamma\vdash_{\Sigma}\mathbf{A}::\mathbb{A}}\,ws{:}\beta\eta}$$

sub-derivations are again semi-structural. We call $\lambda\overline{X^{l}}.\mathbf{H}\overline{\mathbf{D}^{m}}$ the **structure formula** of $\mathbf{A}$. where $\mathcal{B}\colon\Gamma\vdash_{\Sigma}\lambda\overline{X^{l}}.\mathbf{H}\overline{\mathbf{D}^{m}}::\mathbb{A}$ is a structural $\Sigma\Lambda^{\rightarrow}$-derivation, and its principal

Structure Theorem
trivially sorted
sorted!trivially
signature!semi-structural
non-reducing
variable!non-reducing

In other words, a derivation is semi-structural, if it has a structural core, and the non-structural parts are limited to $\beta\eta$-conversion afterwards. This definition allows us to state the structure theorem.

Theorem 8.1.4 (Structure Theorem)
If $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$*, then there is a semi-structural* $\Sigma\Lambda^{\rightarrow}$*-derivation of* $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$*.*

Proof sketch: The course of the proof (which will take up the rest of this section) will be to define a relation $\mathcal{SR}$, which captures the content of the structure theorem, and a relation $\mathcal{LR}$, that transports the structure information from the base sorts to the arrow sorts. These relations coincide on the base sorts, and we show in 8.1.20 that $\mathcal{LR} \subseteq \mathcal{SR}$ and in 8.1.21 that all well-sorted formulae are in $\mathcal{LR}$, which together entail the assertion of the structure theorem. □

Before we embark on the proof let us look at a special case

Lemma 8.1.5 *If* $\vdash_{sig} \Sigma$ *can be proven only with the rules sig:empty and sig:const′, then every valid judgment* $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ *has a structural* $\Sigma\Lambda^{\rightarrow}$*-derivation.*

Proof: We first note that Σ has at most one term declaration per constant. Furthermore, there are no declarations involving functional base sorts. Thus Σ is trivially sorted (cf. 7.1.6) and we have structural $\Sigma\Lambda^{\rightarrow}$ derivations since we have them for $\Lambda^{\rightarrow}$. □

Remark 8.1.6 As a consequence we can assume in the proof of the structure theorem that every signature is semi-structural, i.e. that all the $\Sigma\Lambda^{\rightarrow}$-derivations in *sig:td:f* and *sig:td:nf* are semi-structural.

Proof: If the structure theorem holds with this assumption, then we can use an induction (using Lemma 8.1.5 in the base case and the structure theorem in the step case) on the number of applications of *sig:td:f* and *sig:td:nf* in the proof of $\vdash_{sig} \Sigma$ to establish the result for general signatures. □

Definition 8.1.7 (Non-Reducing)
We will call a variable Z **non-reducing** in a semi-structural $\Sigma\Lambda^{\rightarrow}$-derivation $\mathcal{D}$, iff there is no principal declaration $[Z{::}\mathbb{B}]$ in $\mathcal{D}$.

The significance of this concept is that substitutions for non-reducing variables conserve the existence of semi-structural derivations. The intuition behind this is that the substitutions will only take place in argument positions, where they do not result in new β-redexes.

Lemma 8.1.8 *Let* $\mathcal{B}{:}\, \Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{B}$ *and* $\mathcal{A}{:}\, \Delta, [Z{::}\mathbb{B}] \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ *be semi-structural* $\Sigma\Lambda^{\rightarrow}$*-derivations, such that* Z *is non-reducing in* $\mathcal{A}$*, then there is a semi-structural* $\Sigma\Lambda^{\rightarrow}$*-derivation for* $\Delta, \Gamma \vdash_\Sigma [\mathbf{B}/Z]\mathbf{A}{::}\mathbb{A}$*.*

Proof: We prove the assertion by induction on the size of $\mathcal{A}$. If $\mathcal{A}$ only consists of a single *ws:td* application, then $[\mathbf{B}/Z]\mathbf{A} = \mathbf{A}$, since $\mathbf{A}$ is closed. For *ws:var* we either have $[\mathbf{B}/Z]\mathbf{A} = \mathbf{A}$ or $[\mathbf{B}/Z]\mathbf{A} = \mathbf{B}$. Thus in all three base cases the assertion is a trivial consequence of the assumptions.

restricted semi-structural form
form!semi-structural!restricted
$\mathcal{SR}$structure relation
$\mathcal{RSR}$restricted structure relation

For the inductive case let $\Delta' = \Delta, [Z :: \mathbb{B}]$ and $\mathcal{A}$ be of the form

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{H}}{\Xi^l, \Delta' \vdash_\Sigma \mathbf{H} :: \mathbb{H}} \quad \dfrac{\mathcal{D}^i}{\Xi^l, \Delta' \vdash_\Sigma \mathbf{D}^i :: \mathfrak{d}^i(\mathbb{H})}\, i \leq m}{\Xi^l, \Delta' \vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^m} :: \mathfrak{r}^l(\mathbb{A})}\, ws{:}app^m}{\Delta' \vdash_\Sigma \lambda \overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m} :: \mathbb{A}}\, ws{:}abs^l \qquad \Delta' \vdash_\Sigma \lambda \overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m} =_{\beta\eta} \mathbf{A}}{\Delta' \vdash_\Sigma \mathbf{A} :: \mathbb{A}}\, ws{:}\beta\eta$$

Without loss of generality we assume that $\mathbf{Free}(\mathbf{B}) \cap \mathbf{Dom}(\Xi^l) = \emptyset$ and set $\mathbf{E}^i := [\mathbf{B}/Z]\mathbf{D}^i$. By inductive hypothesis we have semi-structural derivations $\mathcal{E}^i \colon \Delta, \Gamma \vdash_\Sigma \mathbf{E}^i :: \mathfrak{d}^i(\mathbb{H})$, so we have

$$\dfrac{\dfrac{\dfrac{\mathcal{H}}{\Xi^l, \Delta, \Gamma \vdash_\Sigma \mathbf{H} :: \mathbb{H}} \quad \dfrac{\mathcal{E}^i}{\Xi^l, \Delta, \Gamma \vdash_\Sigma \mathbf{E}^i :: \mathfrak{d}^i(\mathbb{H})}\, i \leq m}{\Xi^l, \Delta, \Gamma \vdash_\Sigma \mathbf{H}\overline{\mathbf{E}^m} :: \mathfrak{r}^l(\mathbb{A})}\, ws{:}app^m}{\Delta, \Gamma \vdash_\Sigma \lambda \overline{X^l}.\mathbf{H}\overline{\mathbf{E}^m} :: \mathbb{A}}\, ws{:}abs^l$$

which gives the assertion: we have $\Delta, \Gamma \vdash_\Sigma \lambda \overline{X^l}.\mathbf{H}\overline{[\mathbf{B}/Z]\mathbf{D}^m} =_{\beta\eta} [\mathbf{B}/Z]\mathbf{A}$ since $[\mathbf{B}/Z]\mathbf{H} = \mathbf{H}$ as in the base cases. □

With this notion, we can single out a restricted class of semi-structural $\Sigma\Lambda^{\rightarrow}$-derivations that contains the base cases and is closed under application to semi-structural formulae.

Definition 8.1.9 (Restricted Semi-Structural Form)
Let $\mathcal{A} \colon \Gamma \vdash_\Sigma \mathbf{A} :: \mathbb{A}$ be a semi-structural $\Sigma\Lambda^{\rightarrow}$-derivation, then we say that $\mathcal{A}$ is in **restricted semi-structural form**, iff $\lambda \overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}$ is the structure formula for $\mathbf{A}$ in $\mathcal{A}$ and all X^i are non-reducing in $\mathcal{A}$.

Note that not all valid sort judgments have $\Sigma\Lambda^{\rightarrow}$-derivations in restricted semi-structural form: For instance, in the empty signature, the judgment $\vdash_\emptyset \lambda X.X :: \mathbb{A} \rightarrow \mathbb{A}$ is valid, but the only $\Sigma\Lambda^{\rightarrow}$-proof of it is not in restricted form, since it uses $[X :: \mathbb{A}]$ as the principal declaration.

It will turn out that the property of having a $\Sigma\Lambda^{\rightarrow}$-derivation in restricted semi-structural form is just the generalization (closed under application to structural formulae) of the *ws:var* and *ws:td* declaration cases that we need to make the inductions in the logical relations proofs go through.

structure relation
relation!structural
restricted structure relation

Definition 8.1.10 (Structure Relation)
We say the **structure relation** $\mathcal{SR}$ holds on $\mathbf{A}$ and $\mathbb{A}$ (we write $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{A}; \mathbb{A})$), iff there is a semi-structural $\Sigma\Lambda^{\rightarrow}$-derivation $\mathcal{A}: \Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ such that for all principal sub-derivations $\mathcal{D}^i: \Gamma, \Xi^l \vdash_\Sigma \mathbf{D}^i::\mathfrak{d}^i(\mathbb{H})$ of $\mathcal{A}$ and every substitution σ with $\Delta \vdash_\Sigma \mathcal{SR}(\sigma; \Gamma, \Xi^l)$ (see below) we have $\Delta, \Xi^l \vdash_\Sigma \mathcal{SR}(\sigma(\mathbf{D}^i); \mathfrak{d}^i(\mathbb{H}))$.[36]

EdNote(36)

Similarly we define the **restricted structure relation** $\mathcal{RSR}$ by the existence of a $\Sigma\Lambda^{\rightarrow}$-derivation in restricted semi-structural form.

For a substitution σ with $\Gamma \vdash_\Sigma \sigma::\Delta$ we write $\Gamma \vdash_\Sigma \mathcal{SR}(\sigma; \Delta)$, iff for all $X \in \mathbf{Dom}(\sigma)$ we have $\Gamma \vdash_\Sigma \mathcal{SR}(\sigma(X); \Delta(X))$. Note that for the identity substitution $\emptyset$ we can vacuously conclude $\Gamma \vdash_\Sigma \mathcal{SR}(\emptyset; \emptyset)$.

For the rest of this section we will assume $\Delta \vdash_\Sigma \mathcal{SR}(\sigma; \Gamma, \Xi^l)$ for all occurrences of σ.

Remark 8.1.11 (Monotonicity)
Note that $\mathcal{SR}$ as well as $\mathcal{RSR}$ are monotonic, i.e. $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{A}; \mathbb{A})$, if $\Gamma' \vdash_\Sigma \mathcal{SR}(\mathbf{A}; \mathbb{A})$ and $\Gamma' \subseteq \Gamma$.

We now prove some technical lemmata we need in the proofs later on.

Lemma 8.1.12 *$[X::\mathbb{A}] \vdash_\Sigma \mathcal{RSR}(X; \mathbb{A})$ and $\vdash_\Sigma \mathcal{RSR}(\mathbf{H}; \mathbb{H})$ if $[\mathbf{H}::\mathbb{H}] \in \Sigma$.*

Proof: We prove the assertion by an induction over the structure of $\mathbb{A}$. If $\mathbb{A} \in \mathcal{BS}$, then $l = 0$ and the $\Sigma\Lambda^{\rightarrow}$-derivation that consists of a single *ws:var* or *ws:td* step yields the assertion.

If $\mathbb{A} \notin \mathcal{BS}$, then the following $\Sigma\Lambda^{\rightarrow}$-proof $[Y::\mathbb{A}] \vdash_\Sigma Y::\mathbb{A}$ is in restricted semi-structural form.

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{H}}{\Xi^l, [Y::\mathbb{A}] \vdash_\Sigma Y::\mathbb{A}} \quad \dfrac{\mathcal{D}^i}{\Xi^l, [Y::\mathbb{A}] \vdash_\Sigma X^i::\mathfrak{d}^i(\mathbb{A})}\ i \leq l}{\Xi^l, [Y::\mathbb{A}] \vdash_\Sigma Y\overline{X^l}::\mathfrak{r}^l(\mathbb{A})}\ ws{:}app^l}{[Y::\mathbb{A}] \vdash_\Sigma \lambda\overline{X^l}.Y\overline{X^l}::\mathbb{A}}\ ws{:}abs^l \quad [Y::\mathbb{A}] \vdash_\Sigma \lambda\overline{X^l}.Y\overline{X^l} =_\eta Y}{[Y::\mathbb{A}] \vdash_\Sigma Y::\mathbb{A}}\ ws{:}\beta\eta$$

To show that it yields the assertion, we have to show that for any principal sub-derivation $\mathcal{D}^i: [X^i::\mathfrak{d}^i(\mathbb{A})] \vdash_\Sigma X^i::\mathfrak{d}^i(\mathbb{A})$ and substitution σ with $\Gamma \vdash_\Sigma \mathcal{SR}(\sigma; \Xi^l, [Y::\mathbb{A}])$ we have $\mathcal{D}^i: [X^i::\mathfrak{d}^i(\mathbb{A})] \vdash_\Sigma \mathcal{SR}(\sigma(X^i); \mathfrak{d}^i(\mathbb{A}))$. For this we have to consider two cases: If $X^i \in \mathbf{Dom}(\sigma)$, then the assertion is a trivial consequence of the assumption about σ, if not, then we obtain suitable $\Sigma\Lambda^{\rightarrow}$-derivations by inductive hypothesis and the fact that $\mathcal{RSR} \subseteq \mathcal{SR}$ and by monotonicity. □

[36] EdNote: Chad, please check this definition carefully

Lemma 8.1.13 *Let* $\Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B}$*, then* $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{B}; \mathbb{A})$ *iff* $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{A}; \mathbb{A})$.

Proof: Let $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{A}; \mathbb{A})$, so we have a semi-structural $\Sigma\Lambda^{\rightarrow}$-derivation of the form

$$\dfrac{\dfrac{\mathcal{D}}{\Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}::\mathbb{A}} \quad \dfrac{\mathcal{C}}{\Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}}}{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}}\,ws{:}\beta\eta$$

We can obtain a $\Sigma\Lambda^{\rightarrow}$-derivation $\mathcal{C}'' \colon \Gamma \vdash_\Sigma \sigma(\mathbf{B}) =_{\beta\eta} \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}$ with *ms:trans* and *eq:sym* and conclude that

$$\dfrac{\dfrac{\mathcal{D}}{\Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}::\mathbb{A}} \quad \dfrac{\mathcal{C}''}{\Gamma \vdash_\Sigma \mathbf{B} =_{\beta\eta} \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}}}{\Gamma \vdash_\Sigma \mathbf{B}::\mathbb{A}}\,ws{:}\beta\eta$$

is a semi-structural $\Sigma\Lambda^{\rightarrow}$-derivation that verifies the claim. □

We will often simplify the displayed $\Sigma\Lambda^{\rightarrow}$-derivations by omitting the final *ws:βη* step and instantiation with σ, where it does not change from source to target.

Lemma 8.1.14 *Let* $\Gamma \vdash_\Sigma \mathcal{RSR}(\mathbf{A}; \mathbb{A})$ *and* $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{C}; \mathfrak{d}(\mathbb{A}))$*, then* $\Gamma \vdash_\Sigma \mathcal{RSR}(\mathbf{AC}; \mathfrak{d}(\mathbb{A}))$.

Proof: We have the following $\Sigma\Lambda^{\rightarrow}$-derivation $\mathcal{A}$ in restricted semi-structural form, where $\lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}$ is the structure formula of $\mathbf{A}$:

$$\dfrac{\dfrac{\dfrac{\mathcal{H}}{\Gamma, \Xi^l \vdash_\Sigma \mathbf{H}::\mathbb{H}} \quad \dfrac{\mathcal{D}^i}{\Gamma, \Xi^l \vdash_\Sigma \mathbf{D}^i::\mathfrak{d}^i(\mathbb{H})}\,i \leq m}{\Gamma, \Xi^l \vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^m}::\mathfrak{r}^l(\mathbb{A})}\,ws{:}app^m}{\Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}::\mathbb{A}}\,ws{:}abs^l$$

We prove the assertion in two steps, we show the special case where $\mathbb{A} = \mathbb{C} \rightarrow \mathbb{D}$ and then convince ourselves that $\Gamma \vdash_\Sigma \mathcal{RSR}(\mathbf{A}; \mathbb{A})$ entails $\Gamma \vdash_\Sigma \mathcal{RSR}(\mathbf{A}; \mathfrak{d}(\mathbb{A}) \rightarrow \mathfrak{r}(\mathbb{A}))$. The case for general $\mathbb{A}$ is just a combination of these two claims.

For the first claim note that $\mathbf{AC}{=_{\beta\eta}}\lambda X^2\ldots X^l.\mathbf{H}\overline{[\mathbf{C}/X^1]\mathbf{D}^m}$, since X^1 is non-reducing in $\mathcal{A}$. Now let $\mathbf{E}^i := [\mathbf{C}/X^1]\mathbf{D}^i$ and $\Xi^l = [X::\mathbb{C}], \Xi^l_2$, then

$$\dfrac{\dfrac{\dfrac{\mathcal{H}}{\Gamma,\Xi^l_2 \vdash_\Sigma \mathbf{H}::\mathbb{H}} \quad \dfrac{\mathcal{E}^i}{\Gamma,\Xi^l_2 \vdash_\Sigma \mathbf{E}^i::\mathfrak{d}^i(\mathbb{H})}\; i\le m}{\Gamma,\Xi^l_2 \vdash_\Sigma \mathbf{H}\overline{\mathbf{E}^m}::\mathfrak{r}^l(\mathbb{C}\to\mathbb{D}) = \mathfrak{r}^{l-1}(\mathbb{D})}\; ws{:}app^m}{\Gamma \vdash_\Sigma \lambda\overline{X^2\ldots X^l}.\mathbf{H}\overline{\mathbf{E}^m}::\mathbb{D}}\; ws{:}abs^{l-1}$$

is in restricted semi-structural form and we obtain the assertion, since $\mathcal{E}^i\colon \Gamma,\Xi^l \vdash_\Sigma \mathcal{SR}(\mathbf{E}^i;\mathfrak{d}^i(\mathbb{H}))$ by Lemma 8.1.8.

For the second claim we can assume without loss of generality that $\mathbb{A} \in \mathcal{BS}^f$ (and as a consequence $l = 0$ and $\mathfrak{r}^m(\mathbb{H}) = \mathbb{A}$), since otherwise the assertion is trivial. To ease the load of notation we set $\mathbb{B} = \overline{\mathfrak{d}(\mathbb{A})} \to \mathfrak{r}(\mathbb{A})$, and $\mathbb{C} := \mathfrak{r}^n(\mathbb{B})$, where $n := \mathbf{ln}(\mathbb{B})$, so we have $\mathbb{C} \in \mathcal{BS}$ and $\mathbb{B} = \overline{\mathfrak{d}^n(\mathbb{B})} \to \mathbb{C}$.

If $\mathcal{H}$ consists of a single *ws:var* step, then the following $\Sigma\Lambda^{\to}$-derivation gives the assertion:

$$\dfrac{\dfrac{\dfrac{\mathcal{H}}{\Gamma,\Upsilon^n \vdash_\Sigma \mathbf{H}::\mathbb{H}} \quad \dfrac{\mathcal{D}^i}{\Gamma,\Upsilon^n \vdash_\Sigma \mathbf{D}^i::\mathfrak{d}^i(\mathbb{H})}\; i\le m \quad \dfrac{Y^j}{\Gamma,\Upsilon^n \vdash_\Sigma Y^j::\mathfrak{d}^j(\mathbb{H})}\; j\le n}{\Gamma,\Upsilon^n \vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^m Y^n}::\mathfrak{r}^n(\mathbb{A})}\; ws{:}app^m}{\Gamma \vdash_\Sigma \lambda\overline{Y^n}.\mathbf{H}\overline{\mathbf{D}^m Y^n}::\mathbb{B}}\; ws{:}abs^n$$

Clearly this is in restricted semi-structural form, since the Y^j do not occur in $\mathbf{H}$ or the $\mathbf{D}^i$, thus they are non-reducing. Note that this is the best we can do, for the variable case; in particular, we cannot obtain $m = \mathbf{ln}(\mathbb{H})$ for the variable declaration case.

In the *ws:td* case of the second assertion we have to take a closer look at the derivation $\mathcal{H}$, taking into account the $\Sigma\Lambda^{\to}$-derivations for the precondition $\vdash_{sig} \Sigma$ in the *ws:td* rule. We have disregarded these derivations in most the derivations so far, but in this proof we need it, since we cannot prove the claim by juggling around with the material from the inductive hypothesis; we need a new principal declaration (for a discussion see Example 8.1.15).

Since $\mathbb{A}$ is a functional base sort, there must be a sub-derivation of $\mathcal{H}$ of the form

$$\dfrac{\vdash_{sig} \Sigma' \quad \dfrac{\mathcal{H}'}{\vdash_{\Sigma'} \mathbf{H}::\mathbb{H}'}}{\vdash_{sig} \Sigma', [\mathbf{H}::\mathbb{H}]}\; sig{:}td{:}f$$

where $\mathbb{H}' = \overline{\mathfrak{d}^m(\mathbb{H})} \to \mathbb{B}$ (only *sig:td:f* introduces term declarations for functional base sorts). Using Remark 8.1.6 and the assumption that $l = 0$, we see that $\mathcal{H}'$ must be of the form

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\mathcal{K}}{\Upsilon^m, \Theta^n \vdash_{\Sigma'} \mathbf{K}{::}\mathbb{K}} \quad \cfrac{\mathcal{E}^i}{\Upsilon^m, \Theta^n \vdash_{\Sigma'} \mathbf{E}^i{::}\mathfrak{d}^i(\mathbb{K})}}{\Upsilon^m, \Theta^n \vdash_{\Sigma'} \mathbf{K}\overline{\mathbf{E}^{m'}}{::}\mathbb{C}}\, ws{:}app^{m'}}{\Upsilon^m \vdash_{\Sigma'} \lambda Z^n.\mathbf{K}\overline{\mathbf{E}^{m'}}{::}\mathbb{B}}\, ws{:}abs^n}{\vdash_{\Sigma'} \lambda Y^m.\lambda Z^n.\mathbf{K}\overline{\mathbf{E}^{m'}}{::}\mathbb{H}'}\, ws{:}abs^m \qquad \vdash_{\Sigma'} \mathbf{H}{=_{\beta\eta}}\lambda Y^m.\lambda Z^n.\mathbf{K}\overline{\mathbf{E}^{m'}}}{\vdash_{\Sigma'} \mathbf{H}{::}\mathbb{H}'}\, ws{:}\beta\eta$$

Now let $\mathbf{F}^i := \overline{[\mathbf{D}^m/Y^m]}\mathbf{E}^i$. As the $\mathcal{D}^i$ are the principal subderiva- BegNP(37)
tions of $\mathcal{A}$ we have $\Gamma \vdash_\Sigma \mathcal{SR}(\mathcal{D}^i; \mathfrak{d}^i(\mathbb{H}))$ (taking $\sigma = \emptyset$) and thus $\Gamma \vdash_\Sigma \mathcal{SR}(\overline{[\mathbf{D}^m/Y^m]}; \Upsilon^m)$ by definition. With our assumption 8.1.6 we have $\mathcal{H}'{:}\vdash_{\Sigma'} \mathcal{SR}(\mathbf{H}; \mathbb{H}')$ with principal sub-derivations $\overline{\mathcal{E}^i}$ and thus $\mathcal{F}^i{:}\,\Theta^n, \Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{F}^i; \mathfrak{d}^i(\mathbb{K}))$ by definition of $\mathcal{SR}$ (taking $\sigma = \overline{[\mathbf{D}^m/Y^m]}$).

Thus we have the following $\Sigma\Lambda^{\rightarrow}$-derivation $\mathcal{A}'$ (note that $\Sigma\Lambda^{\rightarrow}$ is monotonic in the signature):

$$\cfrac{\cfrac{\cfrac{\mathcal{K}}{\Theta^n, \Gamma \vdash_\Sigma \mathbf{K}{::}\mathbb{K}} \quad \cfrac{\mathcal{F}^i}{\Theta^n, \Gamma \vdash_\Sigma \mathbf{F}^i{::}\mathfrak{d}^i(\mathbb{K})}}{\Theta^n, \Gamma \vdash_\Sigma \mathbf{K}\overline{\mathbf{F}^{m'}}{::}\mathbb{C}}\, ws{:}app^{m'}}{\Gamma \vdash_\Sigma \lambda Z^n.\mathbf{K}\overline{\mathbf{F}^{m'}}{::}\mathbb{B}}\, ws{:}abs^n$$

Clearly, extending $\mathcal{A}'$ with a $ws{:}\beta\eta$ step gives a $\Sigma\Lambda^{\rightarrow}$-derivation for $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ in restricted semi-structural form, since

$$\mathbf{A}{=_{\beta\eta}}\mathbf{H}\overline{\mathbf{D}^m}{=_{\beta\eta}}(\lambda Y^m.\lambda Z^n.\mathbf{K}\overline{\mathbf{E}^{m'}})\overline{\mathbf{D}^m}{=_{\beta\eta}}\lambda Z^n.\mathbf{K}\overline{\mathbf{F}^{m'}}$$

To obtain $\Gamma \vdash_\Sigma \mathcal{RSR}(\mathbf{A}; \mathbb{A})$, we only have to prove $\Theta^n, \Delta \vdash_\Sigma \mathcal{SR}(\sigma(\mathbf{F}^i); \mathfrak{d}^i(\mathbb{K}))$. This is can be obtained by extending the $\mathcal{SR}$-substitution $\overline{[\mathbf{D}^m/Y^m]}$ to $\sigma \circ \overline{[\mathbf{D}^m/Y^m]}$ in the argumentation above. □ EndNP(37)

To gain a better understanding of the proof let us exemplify the second claim in a familiar example.

Example 8.1.15 (Specializing to Elementary Calculus)
If we take Σ to be the signature from example 7.7.22, there are two semi-structural derivations for $\Gamma \vdash_\Sigma \lambda X.FX + GX{::}\mathbb{R} \rightarrow \mathbb{R}$, where $\mathbf{H} =$

[37] New Part:

$\lambda F, G.\lambda X.FX + GX$, and $\Gamma = [F::\mathbb{P}], [G::\mathbb{G}]$ and $\Gamma' = \Gamma, [X::\mathbb{R}]$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash_{\Sigma'} \mathbf{H}::\mathbb{P}\to\mathbb{P}\to\mathbb{R}\to\mathbb{R}}{\vdash_{sig}\Sigma}\,\textit{sig:td:f}}{\Gamma'\vdash_\Sigma \mathbf{H}::\mathbb{P}\to\mathbb{P}\to\mathbb{P}}\,\textit{ws:td} \qquad \Gamma'\vdash_\Sigma F,G::\mathbb{P} \quad \Gamma'\vdash_\Sigma X::\mathbb{R}}{\Gamma'\vdash_\Sigma \mathbf{H}FGX::\mathbb{R}}\,\textit{ws:app}^3}{\Gamma'\vdash_\Sigma \lambda X.\mathbf{H}FGX::\mathbb{R}\to\mathbb{R}}\,\textit{ws:abs} \qquad *}{\Gamma\vdash_\Sigma \lambda X.FX+GX::\mathbb{R}\to\mathbb{R}}\,\textit{ws:}\beta\eta$$

and (where $\mathcal{F}$ and $\mathcal{B}$ are the obvious semi-structural $\Sigma\Lambda^{\to}$-derivations)

$$\dfrac{\dfrac{\dfrac{}{\Gamma'\vdash_\Sigma +::\mathbb{R}\to\mathbb{R}\to\mathbb{R}}\,\textit{ws:td} \quad \dfrac{\mathcal{F}}{\Gamma'\vdash_\Sigma FX::\mathbb{R}} \quad \dfrac{\mathcal{G}}{\Gamma'\vdash_\Sigma GX::\mathbb{R}}}{\Gamma'\vdash_\Sigma FX+GX::\mathbb{R}\to\mathbb{R}}\,\textit{ws:app}^2}{\Gamma\vdash_\Sigma \lambda X.FX+GX::\mathbb{R}\to\mathbb{R}}\,\textit{ws:abs}$$

Only the second one is in strong semi-structural form ($m = 2 = \mathbf{ln}(\mathbb{R}\to\mathbb{R}\to\mathbb{R})$), since in the first one has $m = 3 > 2 = \mathbf{ln}(\mathbb{P}\to\mathbb{P}\to\mathbb{P})$. The cut-elimination element in the second claim constructs the second one from the first by coming up with a new principal declaration from the well-formedness proof of the signature.

Lemma 8.1.16 *If* $\Gamma, [Y::\mathbb{C}] \vdash_\Sigma \mathcal{SR}(\mathbf{A};\mathbb{D})$, *then* $\Gamma\vdash_\Sigma \mathcal{SR}(\lambda Y.\mathbf{A};\mathbb{C}\to\mathbb{D})$.

Proof: Let $\Gamma' = \Gamma, [Y::\mathbb{C}]$ and $\mathcal{A}\colon \Gamma'\vdash_\Sigma \mathbf{A}::\mathbb{D}$ be of the form

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{H}}{\Gamma',\Xi^l\vdash_\Sigma \mathbf{H}::\mathbb{H}} \quad \dfrac{\mathcal{D}^i}{\Gamma',\Xi^l\vdash_\Sigma \mathbf{D}^i::\mathfrak{d}^i(\mathbb{H})}\, i\leq m}{\Gamma',\Xi^l\vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^m}::\mathfrak{r}^l(\mathbb{D})}\,\textit{ws:app}^m}{\Gamma'\vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}::\mathbb{D}}\,\textit{ws:abs}^l \qquad \mathcal{C}}{\Gamma'\vdash_\Sigma \mathbf{A}::\mathbb{D}}\,\textit{ws:}\beta\eta$$

where $\mathcal{C}\colon\Gamma' \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}{=_{\beta\eta}}\mathbf{A}$. We have $\mathcal{C}'\colon\Gamma \vdash_\Sigma \lambda Y.\mathbf{A}{=_{\beta\eta}}\lambda Y\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}$ by *tr*:*abs*, thus the following $\Sigma\Lambda^{\rightarrow}$-derivation $\mathcal{A}'$ is in semi-structural form

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{H}}{\Gamma',\Xi^l \vdash_\Sigma \mathbf{H}{::}\mathbb{H}} \quad \dfrac{\mathcal{D}^i}{\Gamma',\Xi^l \vdash_\Sigma \mathbf{D}^i{::}\mathfrak{d}^i(\mathbb{H})}\; i \leq m}{\Gamma',\Xi^l \vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^m}{::}\mathfrak{r}^{l+1}(\mathbb{C} \rightarrow \mathbb{D})}\; ws{:}app^m}{\Gamma \vdash_\Sigma \lambda Y\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}{::}\mathbb{C} \rightarrow \mathbb{D}}\; ws{:}abs^{l+1}}{} \quad \mathcal{C}'}{\Gamma \vdash_\Sigma \lambda Y.\mathbf{A}{::}\mathbb{C} \rightarrow \mathbb{D}}\; ws{:}\beta\eta$$

The length conditions for m still hold, since we have not changed its value.

$\square$

Remark 8.1.17 (Depth Condition)
If we look closely at the proof above we can see that $\mathbf{dp}(\mathcal{A}') = \mathbf{dp}(\mathcal{A}) + 1$, since $\mathbf{dp}(\mathcal{B}') = \mathbf{dp}(\mathcal{B}) + 1$, $\mathbf{dp}(\mathcal{C}') = \mathbf{dp}(\mathcal{C}) + 1$, and there is an additional *ws*:*abs* step in the left branch of $\mathcal{A}'$.

We now want to define a second relation $\mathcal{LR}$ for the structure theorem and investigate its relation to $\mathcal{SR}$.

Definition 8.1.18 (Logical Relation)
The **logical structure relation** $\mathcal{LR}$ is defined by an induction over the type of $\mathbf{A}$: We write $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A}; \mathbb{A})$, iff

1. $\mathbb{A} \in \mathcal{BT}$ and $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{A}; \mathbb{A})$
2. If $\mathbb{A}$ is functional, $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$, and $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{AC}; \mathfrak{r}(\mathbb{A}))$ for all $\mathbf{C}$ with $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{C}; \mathfrak{d}(\mathbb{A}))$

For a Σ-substitution θ with $\Gamma \vdash_\Sigma \sigma{::}\Delta$ we define $\mathcal{LR}(\theta; \Delta)$ to hold, iff $\Gamma \vdash_\Sigma \mathcal{LR}(\theta(X); \Delta(X))$ holds for all $X \in \mathbf{Dom}(\theta)$. Note that for the identity substitution $\emptyset$ we can vacuously conclude $\Gamma \vdash_\Sigma \mathcal{LR}(\emptyset; \emptyset)$.

This definition is analogous to that of a logical relation in $\Lambda^{\rightarrow}$ (cf. 1.3.5).

The course of the proof of the structure theorem is to show that for a given signature Σ and context Γ the relation $\mathcal{LR}$ subsumes the relation $\mathcal{SR}$. Then we will prove that for all well-sorted formulae $\mathbf{A}$ of sort $\mathbb{A}$ we have $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A}; \mathbb{A})$ and therefore $\mathcal{SR}(\mathbf{A}; \mathbb{A})$, which is just the assertion of the structure theorem. The logical Relation technique is due to Tait and Statman [Tai67, Sta85], the concrete style of the proof is heavily influenced by a presentation of the argument by Frank Pfenning in [Pfe93a].

Lemma 8.1.19 (Closure under $\beta\eta$-Equality)
If $\Gamma \vdash_\Sigma \mathbf{A}{=_{\beta\eta}}\mathbf{B}$, then we have $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A}; \mathbb{A})$, iff $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{B}; \mathbb{A})$ holds.

Proof: We prove the assertion by induction over $\tau(\mathbb{A})$. For $\tau(\mathbb{A}) \in \mathcal{BT}$ we obtain the assertion with 8.1.13.

If $\mathbb{A}$ is functional, let $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{C}; \mathfrak{d}(\mathbb{A}))$; then $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{AC}; \mathfrak{r}(\mathbb{A}))$ and $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{BC}; \mathfrak{r}(\mathbb{A}))$, since $\Gamma \vdash_\Sigma \mathbf{AC} =_{\beta\eta} \mathbf{BC}$ by *tr:app* and the inductive hypothesis, therefore we obtain $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{B}; \mathbb{A})$ by definition of $\mathcal{LR}$. □

Now, we establish a central result on the relation of $\mathcal{SR}$ and $\mathcal{LR}$. We will use the second assertion to reduce the structure theorem from a claim about $\mathcal{SR}$ to one about $\mathcal{LR}$ in 8.1.21. The first assertion is only needed to make the induction work for the second one.

Lemma 8.1.20 (Roller Coaster Lemma)
We have the following dependencies between $\mathcal{SR}$, $\mathcal{RSR}$, and $\mathcal{LR}$:

1. *If $\Gamma \vdash_\Sigma \mathcal{RSR}(\mathbf{A}; \mathbb{A})$, then $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A}; \mathbb{A})$.*

2. *If $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A}; \mathbb{A})$, then $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{A}; \mathbb{A})$.*

Proof: We prove the two assertions by joint induction on $\tau(\mathbb{A})$. If $\tau(\mathbb{A}) \in \mathcal{BT}$, then the claim is trivial, since $\mathcal{RSR}$ is a sub-relation of $\mathcal{SR}$ and the relations $\mathcal{LR}$ and $\mathcal{SR}$ coincide for non-functional sorts.

Now let $\mathbb{A}$ be functional. For the first assertion let $\Gamma \vdash_\Sigma \mathcal{RSR}(\mathbf{A}; \mathbb{A})$ and $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{C}; \mathfrak{d}(\mathbb{A}))$. Then by the second inductive hypothesis we have $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{C}; \mathfrak{d}(\mathbb{A}))$ and thus $\Gamma \vdash_\Sigma \mathcal{RSR}(\mathbf{AC}; \mathfrak{d}(\mathbb{A}))$ by 8.1.14, so by the first inductive hypothesis we have $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{AC}; \mathfrak{r}(\mathbb{A}))$, which gives $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A}; \mathbb{A})$ by definition of $\mathcal{LR}$.

For the second assertion let $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A}; \mathbb{A})$ and $Y \notin \mathbf{Dom}(\Gamma)$ be a variable, by Lemma 8.1.12 we have $[Y{::}\mathfrak{d}(\mathbb{A})] \vdash_\Sigma \mathcal{RSR}(Y; \mathfrak{d}(\mathbb{A}))$ and $\Gamma, [Y{::}\mathfrak{d}(\mathbb{A})] \vdash_\Sigma \mathcal{LR}(Y; \mathfrak{d}(\mathbb{A}))$ by the first inductive hypothesis and Remark 8.1.11. Thus by definition of $\mathcal{LR}$ we see that $\Gamma, [Y{::}\mathfrak{d}(\mathbb{A})] \vdash_\Sigma \mathcal{LR}(\mathbf{A}Y; \mathfrak{r}(\mathbb{A}))$ and get $\Gamma, [Y{::}\mathfrak{d}(\mathbb{A})] \vdash_\Sigma \mathcal{SR}(\mathbf{A}Y; \mathfrak{r}(\mathbb{A}))$ by the second inductive hypothesis. By lemma 8.1.16 we have $\Gamma \vdash_\Sigma \mathcal{SR}(\lambda Y.\mathbf{A}Y; \mathbb{A})$, thus we have completed the second assertion with 8.1.13, since $Y \notin \mathbf{Free}(\mathbf{A})$. □

Lemma 8.1.21 *If $\Xi, \Delta \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ and $\Gamma \vdash_\Sigma \mathcal{LR}(\theta; \Delta)$, then $\Xi, \Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{A}); \mathbb{A})$.*

Proof: By induction over the $\Sigma\Lambda^{\to}$-derivation $\mathcal{D}{:}\, \Xi, \Delta \vdash_\Sigma \mathbf{A}{:}\, \mathbb{A}$:

ws:var Here $\mathbf{A}$ is a variable X and $\Delta(X) = \mathbb{A}$ or $\Xi(X) = \mathbb{A}$. In the first case we have $\Xi, \Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{A}); \mathbb{A})$ by monotonicity (8.1.11), since we have required that $\Gamma \vdash_\Sigma \mathcal{LR}(\theta; \Delta)$. In the second case $\theta(X) = X$ and $[X{::}\mathbb{A}] \vdash_\Sigma \mathcal{RSR}(\mathbf{A}; \mathbb{A})$, by Lemma 8.1.12. Thus we obtain the assertion by 8.1.20.1 and 8.1.11.

ws:td Since we have assumed that term declarations are closed, we have $\theta(\mathbf{A}) = \mathbf{A}$, so we have $\vdash_\Sigma \mathcal{RSR}(\theta(\mathbf{A}); \mathbb{A})$ by Lemma 8.1.12; again we are done by Lemma 8.1.20.1. and 8.1.11.

ws:abs In this case $\mathbf{A} = (\lambda X.\mathbf{C})$, $\mathbb{A} = \mathbb{B} \to \mathbb{C}$ and $\mathcal{D}$ ends in

$$\frac{\Xi, \Delta, [X{::}\mathbb{B}] \vdash_\Sigma \mathbf{C}{::}\mathbb{C}}{\Xi, \Delta \vdash_\Sigma \lambda X.\mathbf{C}{::}\mathbb{B} \to \mathbb{C}}\ ws{:}abs$$

In order to show that $\Xi, \Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{A}); \mathbb{B} \to \mathbb{C})$ we consider an arbitrary formula $\mathbf{B}$, such that $\Xi, \Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{B}; \mathbb{B})$, and show that $\Xi, \Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{A})\mathbf{B}; \mathbb{C})$. Then the assertion is a direct consequence of the definition of $\mathcal{LR}$.

Now let $\theta' := \theta, [\mathbf{B}/X]$, and $\Delta' := \Delta, [X{::}\mathbb{B}]$ then $\Gamma \vdash_\Sigma \mathcal{LR}(\theta'; \Delta')$ by definition and $\Xi, \Gamma, [X{::}\mathbb{B}] \vdash_\Sigma \mathcal{LR}(\theta'(\mathbf{C}); \mathbb{C})$ by inductive hypothesis. Now

$$\begin{array}{rcl} \theta(\mathbf{A})\mathbf{B} \equiv \theta(\lambda X.\mathbf{C})\mathbf{B} & \equiv & (\lambda X.\theta_{-X}(\mathbf{C}))\mathbf{B} \\ & \to_\beta^h & \theta_{-X}([\mathbf{B}/X]\mathbf{C}) \equiv \theta'(\mathbf{C}) \end{array}$$

together with Lemma 8.1.19 finally gives us $\Xi, \Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{A}); \mathbb{A})$.

ws:app In this case $\mathbf{A} \equiv \mathbf{BC}$, $\mathbb{A} = \mathfrak{r}(\mathbb{B})$ for some sort $\mathbb{B}$ and $\mathcal{D}$ ends in

$$\frac{\Xi, \Delta \vdash_\Sigma \mathbf{B}{::}\mathbb{B} \quad \Xi, \Delta \vdash_\Sigma \mathbf{C}{::}\mathfrak{d}(\mathbb{B})}{\Xi, \Delta \vdash_\Sigma \mathbf{BC}{::}\mathfrak{r}(\mathbb{B})}\ ws{:}app$$

By inductive hypothesis we have $\Xi, \Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{B}); \mathbb{B})$ and $\Xi, \Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{C}); \mathfrak{d}(\mathbb{B}))$. Thus we obtain $\Xi, \Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{BC}); \mathfrak{r}(\mathbb{B}))$ by definition of $\mathcal{LR}$.

ws:βη Here, we have the following situation:

$$\frac{\dfrac{\mathcal{A}}{\Xi, \Delta \vdash_\Sigma \mathbf{B}{::}\mathbb{A}} \quad \Xi, \Delta \vdash_\Sigma \mathbf{B} =_{\beta\eta} \mathbf{A}}{\Xi, \Delta \vdash_\Sigma \mathbf{A}{::}\mathbb{A}}\ ws{:}\beta\eta$$

By inductive hypothesis for $\mathcal{A}$ we have that $\Xi, \Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{B}); \mathbb{A})$, so with 8.1.19 we obtain $\Xi, \Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{A}); \mathbb{A})$, since $\Xi, \Gamma \vdash_\Sigma \theta(\mathbf{A}) =_{\beta\eta} \theta(\mathbf{B})$ by substitutivity of $\beta\eta$-reduction (1.3.4), which can be applied by Theorem 7.6.2.

□

Finally, we have obtained all the partial results and only need to assemble the proof of the structure theorem.

Corollary 8.1.22 (Structure Theorem)
If $\mathbf{A} \in wff_\alpha(\Sigma)$, *then* $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{A}; \mathbb{A})$.

Proof: Let $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$, by Lemma 8.1.21 where we take θ to be the identity substitution we see that $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A};\mathbb{A})$, and with lemma 8.1.20.2 we obtain the assertion. $\square$

Note that the structure theorem does not make any claim about the uniqueness of the semi-structural $\Sigma\Lambda^{\to}$-derivation it guarantees. In fact, as the following example shows, there may be several, which even use different principal declarations.

Example 8.1.23 Let $\Sigma := \{[\lambda X_{\bullet}X::\mathbb{C} \to \mathbb{A}], [\lambda X_{\bullet}X::\mathbb{B} \to \mathbb{A}], [a::\mathbb{B}], [a::\mathbb{C}]\}$, then

$$\dfrac{\dfrac{\dfrac{[a::\mathbb{B}] \in \Sigma}{\vdash_\Sigma a::\mathbb{B}}\,\textit{ws:td} \quad \dfrac{[\lambda X_{\bullet}X::\mathbb{B} \to \mathbb{A}] \in \Sigma}{\vdash_\Sigma \lambda X_{\bullet}X::\mathbb{B} \to \mathbb{A}}\,\textit{ws:td}}{\vdash_\Sigma \lambda X_{\bullet}Xa::\mathbb{A}}\,\textit{ws:app}}{\vdash_\Sigma a::\mathbb{A}}\,{*}\,\textit{ws:}\beta\eta$$

and

$$\dfrac{\dfrac{\dfrac{[a::\mathbb{C}] \in \Sigma}{\vdash_\Sigma a::\mathbb{C}}\,\textit{ws:td} \quad \dfrac{[\lambda X_{\bullet}X::\mathbb{C} \to \mathbb{A}] \in \Sigma}{\vdash_\Sigma \lambda X_{\bullet}X::\mathbb{C} \to \mathbb{A}}\,\textit{ws:td}}{\vdash_\Sigma \lambda X_{\bullet}Xa::\mathbb{A}}\,\textit{ws:app}}{\vdash_\Sigma a::\mathbb{A}}\,{*}\,\textit{ws:}\beta\eta$$

are both semi-structural $\Sigma\Lambda^{\to}$-derivations for $\Gamma \vdash_\Sigma a::\mathbb{A}$, they differ in the principal declarations and the principal subderivations (Note that these are in restricted semi-structural form.).

8.2 General Bindings

One of the key steps in sort computation and Σ-unification consists in solving the following problem: given a sort $\mathbb{A}$ and a symbol h, find the most general well-sorted formula of sort $\mathbb{A}$ that has head h. Such formulae are called general bindings (cf. 8.2.2) of sort $\mathbb{A}$ for the head h. In contrast to unsorted general bindings, they are not essentially unique (up to names of the introduced variables), but every term declaration can result in a general binding, if the head and sort satisfy some eligibility conditions. Furthermore, we need a third category of general binding (additionally to the projections and imitations of the unsorted case), to account for term declarations with a subsort component (cf. section 7.7).

The central result, which will allow us to transport the computational methods from the unsorted case is the “General Binding Theorem” 8.2.7, which guarantees a most general representative in the class of formulae of given sort and head.

$\mathcal{G}^h_{\mathbb{A}}(\Sigma,\Gamma)$*set of general bindings of sort* $\mathbb{A}$ *and head* h
$\mathcal{G}^j_{\mathbb{A}}(\Sigma,\Gamma)$*set of* j*-projection bindings of sort* $\mathbb{A}$
$\mathcal{C}(\mathbf{G})$*variable context introduced for* $\mathbf{G}$
general binding
general binding
binding!weakening
imitation binding
binding!imitation
projection binding
binding!projection
approximate
$\mathcal{A}^h_{\mathbb{A}}(\Sigma,\Gamma)$*set of general bindings of sort* $\mathbb{A}$ *that approximate the head* h

Example 8.2.1 (Multiple General Bindings)
Consider the following signature,

$$\Sigma := \{[a{::}\mathbb{A}], [b{::}\mathbb{B}], [f{::}(\mathbb{B} \to \mathbb{B} \to \mathbb{B})], [\lambda X\mathbf{.} faX{::}\mathbb{B} \to \mathbb{A}], [\lambda X\mathbf{.} fXb{::}\mathbb{B} \to \mathbb{A}]\}$$

If Γ be a context with $\Gamma(Z) = \mathbb{B} \to \mathbb{B}$, $\Gamma(X) = \Gamma(Y) = \mathbb{B}$, then the most general formula with the head f and sort $\mathbb{B}$ is fXY, those of sort $\mathbb{A}$ are faX and fXb, and finally those of sort $\mathbb{B} \to \mathbb{A}$ are $\lambda X_{\mathbb{B}}\mathbf{.} fa(ZX)$ and $\lambda X_{\mathbb{B}}\mathbf{.} f(ZX)b$. Note that both pairs of solutions are incomparable by the instantiation ordering $\preceq$.

Definition 8.2.2 (General Binding)
Let Γ be a variable context, $\mathbb{A}$ and $\mathbb{H}$ sorts and $l = \mathbf{ln}(\mathbb{A})$. Then $\mathbf{G} := \lambda X^1 \ldots X^l\mathbf{.HV}^1 \ldots \mathbf{V}^m$ is called a **general binding** of sort $\mathbb{A}$, if $\mathfrak{r}^m(\mathbb{H}) = \mathfrak{r}^l(\mathbb{A})$ and $\mathbf{V}^i = (H^i X^1 \ldots X^l)$ and one of the following holds

1. $[\mathbf{H}{::}\mathbb{H}] \in \Sigma$ and $m = \mathbf{ln}(\mathbb{H})$ or
2. $\mathbf{H}$ is a variable: either $Y = X^j$ and $\mathbb{H} = \mathfrak{d}^j(\mathbb{A})$ or $Y \in \mathbf{Dom}(\Gamma)$ and $\Gamma(Y) = \mathbb{H}$. In both cases $m = l + \mathbf{ln}(\tau(\mathbb{H})) - \mathbf{ln}(\tau(\mathbb{A})) \geq \mathbf{ln}(\mathbb{H})$.

We call $\mathcal{C}(\mathbf{G}) := [H^1{::}\overline{\mathfrak{d}^l(\mathbb{A})} \to \mathfrak{d}^1(\mathbb{H})], \ldots, [H^m{::}\overline{\mathfrak{d}^l(\mathbb{A})} \to \mathfrak{d}^m(\mathbb{H})]$, where $H^i \notin \mathbf{Dom}(\Gamma)$ the **context of variables introduced for G**.

Definition 8.2.3 (Imitation, Projection and Weakening Bindings)

We characterize general bindings $\mathbf{G}$ by the possible cases for $\mathbf{head}(\mathbf{G})$.

- If $\mathbf{G}$ is flexible and $\mathbf{head}(\mathbf{G}) \neq \mathbf{head}(\mathbf{A})$, then we call $\mathbf{G}$ a **weakening binding**.
- If $\mathbf{head}(\mathbf{G})$ is a constant, then we call $\mathbf{G}$ an **imitation binding**.
- If $\mathbf{head}(\mathbf{G}) = j$ (recall our convention in 1.3.2), then we call $\mathbf{G}$ a j**-projection binding**.

For a given head h and sort $\mathbb{A}$, we denote the set of all weakening bindings of sort $\mathbb{A}$, together with those of head h by $\mathcal{G}^h_{\mathbb{A}}(\Sigma,\Gamma)$. Note that since valid signatures are finite, these sets of bindings are also finite for a given sort and head.

Finally, we need the set $\mathcal{A}^h_{\mathbb{A}}(\Sigma,\Delta)$ of general bindings of sort $\mathbb{A}$ that **approximate** a given head h by projection, imitation, or weakening. We define this set by $\mathcal{A}^h_{\mathbb{A}}(\Sigma,\Delta) := \mathcal{G}^h_{\mathbb{A}}(\Sigma,\Delta) \cup \mathcal{G}^j_{\mathbb{A}}(\Sigma,\Delta)$.

Example 8.2.4 (General Bindings for Elementary Calculus)
Let Σ be the signature from Example 7.7.22, then

1. We have $Z * Z \in \mathcal{G}^*_{\mathbb{R}+}(\Sigma, [Z::\mathbb{R}])$ and $Z * W \in \mathcal{G}^*_{\mathbb{R}+}(\Sigma, [Z::\mathbb{R}^+], [W::\mathbb{R}^+])$.

2. $\lambda X.(PX) + (QX) \in \mathcal{G}^+_{\mathbb{P}}(\Sigma; \emptyset)$ and $\mathcal{C}(\lambda X.(PX) + (QX)) = [P, Q::\mathbb{P}])$ using the term declaration $[\lambda FGX. + (FX)(GX))::\mathbb{P} \to \mathbb{P} \to \mathbb{P}]$ with $l = 0$ and $m = 2$.

The definition of general bindings closely corresponds to the definition of semi-structural $\Sigma\Lambda^{\rightarrow}$-derivations. In particular, we have two possible forms for general bindings, the first (classical) one obtains the sort information from the head variable, whereas the second one obtains the sort information from a term declaration. In $\Lambda^{\rightarrow}$ we only have one term declaration per constant (a constant declaration). Consequently, general bindings are unique up to the choice of new variables and consist only of the head and of variables. In $\Sigma\Lambda^{\rightarrow}$ each term declaration, that has the appropriate head and meets certain conditions contributes a general binding.

Lemma 8.2.5 *If $\mathcal{A}: \Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ is a semi-structural $\Sigma\Lambda^{\rightarrow}$-derivation with principal declaration $[\mathbf{H}::\mathbb{H}]$, then the heads of $\mathbf{H}$ and that of $\mathbf{A}$ are related, in particular, we can have three cases:*

1. $\mathbf{head}(\mathbf{A}) = \mathbf{head}(\mathbf{H})$
2. $\mathbf{head}(\mathbf{A}) = j$ *and* $\mathbf{H} = X^j$
3. $\mathbf{head}(\mathbf{H}) = j$

Proof: By a trivial case analysis. Note that if $\mathcal{H}$ consists of *ws:var*, then $\mathbf{H}$ is a variable with $\Gamma, \Xi^l(\mathbf{H}) = \mathbb{H}$, so we have two cases,

- if $\mathbf{H} \in \mathbf{Dom}(\Gamma)$, then $\mathbf{H} = \mathbf{head}(\mathbf{H}) = \mathbf{head}(\mathbf{A})$, since $\Xi^l, \Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m} =_{\beta\eta} \mathbf{A}$, and we obtain the first case of the assertion

- otherwise, $\mathbf{H} = X^j \in \mathbf{Dom}(\Xi^l)$ for some $j \leq l$ and $\mathbf{head}(\mathbf{A}) = j$ and we obtain the second one.

If $\mathcal{H}$ consists of *ws:td*, then there is a term declaration $[\mathbf{H}::\mathbb{H}] \in \Sigma$. In this case we have the following two possibilities for the head of $\mathbf{H}$:

- $\mathbf{head}(\mathbf{H}) = j$, which constitutes the third case of the assertion.

- $\mathbf{head}(\mathbf{H}) \neq j$, then $\Xi^l, \Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m} =_{\beta\eta} \mathbf{A}$ entails that $\mathbf{head}(\mathbf{H}) = \mathbf{head}(\mathbf{A})$ and therefore the first case of the assertion. □

Lemma 8.2.6 *If $\mathbf{G} \in \mathcal{G}^h_{\mathbb{A}}(\Sigma, \Gamma)$ for a sorts $\mathbb{A}$, then $\Gamma, \mathcal{C}(\mathbf{G}) \vdash_\Sigma \mathbf{G}::\mathbb{A}$ and $\mathbf{head}(\mathbf{G}) = h$.*

Proof: We use the notation of definition 8.2.2. Let $\mathbf{G} = (\lambda X^1 \ldots X^l.\mathbf{H}\mathbf{V}^1 \ldots \mathbf{V}^m)$ and $\Xi := [X^1::\mathfrak{d}^1(\mathbb{A})], \ldots, [X^l::\mathfrak{d}^l(\mathbb{A})]$, then we have

$$\dfrac{\dfrac{\dfrac{\mathcal{H}}{\Xi, \mathcal{C}(\mathbf{G}), \Gamma \vdash_\Sigma \mathbf{H}::\mathbb{H}} \qquad \dfrac{\mathcal{C}(\mathbf{G}) \vdash_\Sigma H^i::\overline{\mathfrak{d}^l(\mathbb{A})} \to \mathfrak{d}^i(\mathbb{H}) \quad \Xi \vdash_\Sigma X^j::\mathfrak{d}^j(\mathbb{A})}{\Xi, \mathcal{C}(\mathbf{G}) \vdash_\Sigma \mathbf{V}^i::\mathfrak{d}^i(\mathbb{H})}\,\mathit{ws{:}app}^l}{\Xi, \mathcal{C}(\mathbf{G}), \Gamma \vdash_\Sigma \mathbf{H}\overline{\mathbf{V}^m}::\mathfrak{r}^m(\mathbb{H}) = \mathfrak{r}^l(\mathbb{A})}\,\mathit{ws{:}app}^m}{\Gamma, \mathcal{C}(\mathbf{G}) \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{V}^m}::\mathbb{A}}\,\mathit{ws{:}abs}^l$$

where $\mathcal{H}$ is one of the following $\Sigma\Lambda^{\to}$-derivations (corresponding to the cases for $[\mathbf{H}::\mathbb{H}]$ in the definition of general bindings).

$$\dfrac{[\mathbf{B}::\mathbb{H}] \in \Sigma}{\Xi, \mathcal{C}(\mathbf{G}), \Gamma \vdash_\Sigma \mathbf{B}::\mathbb{H}}\,\mathit{ws{:}td}$$

$$\dfrac{\mathcal{C}(\mathbf{G}), \Gamma(\mathbf{H}) = \mathbb{H}}{\Xi, \mathcal{C}(\mathbf{G}), \Gamma \vdash_\Sigma \mathbf{H}::\mathbb{H}}\,\mathit{ws{:}var} \qquad \dfrac{\Xi(X^j) = \mathfrak{d}^j(\mathbb{A})}{\Xi\mathcal{C}(\mathbf{G}), \Gamma \vdash_\Sigma X^j::\mathfrak{d}^j(\mathbb{A}) = \mathbb{H}}\,\mathit{ws{:}var}$$

Thus we have verified the assertion. □

We will now present a sorted version of the general binding theorem (cf. 3.4.2 for the simply typed analogue).

Theorem 8.2.7 (Sorted General Binding Theorem)
Let $\mathbf{A}$ be a well-sorted formula with $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ and $\mathbf{head}(\mathbf{A}) = h$, then there is a general binding $\mathbf{G} \in \mathcal{G}^h_{\mathbb{A}}(\Sigma, \Gamma)$ and a Σ-substitution ρ such that $\Gamma \vdash_\Sigma \rho::\mathcal{C}(\mathbf{G})$ and $\mathcal{C}(\mathbf{G}), \Gamma \vdash_\Sigma \rho(\mathbf{G}) =_{\beta\eta} \mathbf{A}$.

Proof: Let $\mathcal{A}{:}\,\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ be a semi-structural $\Sigma\Lambda^{\to}$-derivation, then $\mathcal{A}$ is of the form

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{H}}{\Gamma, \Xi^l \vdash_\Sigma \mathbf{H}::\mathbb{H}} \quad \dfrac{\mathcal{D}^i}{\Gamma, \Xi^l \vdash_\Sigma \mathbf{D}^i::\mathfrak{d}^i(\mathbb{H})}\, i \leq m}{\Gamma, \Xi^l \vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^m}::\mathfrak{r}^m(\mathbb{H}) = \mathfrak{r}^l(\mathbb{A})}\,\mathit{ws{:}app}^m}{\Gamma \vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}::\mathbb{A}}\,\mathit{ws{:}abs}^l}{\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}}\,{*}\,\mathit{ws{:}\beta\eta}$$

Now let $\mathbf{G} = (\lambda X^1 \ldots X^l.\mathbf{H}\mathbf{V}^1 \ldots \mathbf{V}^m) \in \mathcal{G}^h_{\mathbb{A}}(\Sigma, \Gamma)$ be a general binding as defined in 8.2.2, and $\mathcal{C}(\mathbf{G})$ the variable context introduced for $\mathbf{G}$. Now, we define the substitution ρ by

$$\rho := [\lambda\overline{X^l}.\mathbf{D}^1/H^1], \ldots, [\lambda\overline{X^l}.\mathbf{D}^m/H^m]$$

so we see that $\mathbf{A}$ is really a $\beta\eta$-instance of $\mathbf{G}$ by

$$\Gamma \vdash_{\Sigma} \rho(\mathbf{G}) \equiv \lambda\overline{X^l}.\rho(\mathbf{H})\overline{\rho(\mathbf{V}^m)} =_{\beta\eta} \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m} =_{\beta\eta} \mathbf{A}$$

since $\rho(\mathbf{H}) = \mathbf{H}$ as $\mathbf{H}$ is a term declaration and therefore closed or $\mathbf{H}$ is a variable but $\mathbf{H} \neq H^i$ for all i. Furthermore, by Lemma 8.2.5 we can see that either $\mathbf{head}(\mathbf{G}) = \mathbf{head}(\mathbf{A})$ (by the first two possibilities) or $\mathbf{head}(\mathbf{H}) = j$ and therefore $\mathbf{head}(\mathbf{G}) = \mathbf{head}(\mathbf{V}^j) = H^j$ and thus $\mathbf{G}$ is a weakening binding in this case. Thus we only have to consider general bindings in $\mathcal{G}^h_{\mathbb{A}}(\Sigma, \Gamma)$. □

Lemma 8.2.8 (Depth Conditions for General Bindings)
Let $\mathcal{A}: \Gamma \vdash_{\Sigma} \mathbf{A}::\mathbb{A}$ be the semi-structural $\Sigma\Lambda^{\to}$-derivation used in the proof of the Theorem above, then there are semi-structural $\Sigma\Lambda^{\to}$-derivations $\mathcal{R}^i$ that witness $\Gamma \vdash_{\Sigma} \mathcal{SR}(\rho; \mathcal{C})(\mathbf{G})$ with $\mathbf{dp}(\mathcal{R}^i) < \mathbf{dp}(\mathcal{A})$.

Proof: We apply lemma 8.1.16 l times to $\mathcal{D}^i$ in the proof above, we obtain semi-structural $\Sigma\Lambda^{\to}$-derivations $\widetilde{\mathcal{D}}^i: \Gamma \vdash_{\Sigma} \lambda\overline{X^l}.\mathbf{D}^i::\overline{\mathfrak{d}^l(\mathbb{A})} \to \mathfrak{d}^i(\mathbb{H})$. By Remark 8.1.17 we have $\mathbf{dp}(\widetilde{\mathcal{D}}^i) = \mathbf{dp}(\mathcal{D}^i) + l < \mathbf{dp}(\mathcal{A})$, since there are l *ws:abs* nodes in $\mathcal{A}$ below $\mathcal{D}^i$. Collecting the $\mathcal{D}^i$ gives us the semi-structural $\Sigma\Lambda^{\to}$-derivations $\mathcal{R}^i$ with the appropriate depth conditions. □

Remark 8.2.9 (General Bindings in One-Sorted Signatures)
Let Σ be a one-sorted signature, then $\Sigma = \{[c::\sharp(\alpha)] \mid c \in \overline{\Sigma}_{\alpha}\}$, so general bindings of head h and sort $\sharp(\alpha)$ have the form $\mathbf{G} := \lambda\overline{X}.h(H^1\overline{X^n})\ldots(H^m\overline{X^n})$ if $h \in \mathcal{V}_{\beta} \cup \overline{\Sigma}_{\beta}$ where $\mathbf{ln}(\flat(\beta)) = m$ and the H^i are new variables of sort $\overline{\mathfrak{d}^n(\sharp(\alpha))} \to \mathfrak{d}^i(\sharp(\beta))$. Note that this is just (up to η-equality) Huet's definition of a partial binding (cf. [Sny91]).

Let us now try to restrict the term declarations in the signature so that the fragment of higher-order patterns is closed under the relevant operations of the calculus. Dale Miller shows in [Mil92] that higher-order unification for this fragment is decidable, a result that was later refined to linear time and space unification by Zhenyu Qian in [Qia96]. This computational behavior will make certain aspects of $\Sigma\Lambda^{\to}$ tractable.

Higher-order patterns are well-typed formulae where any occurrence of a free variable F in $\mathbf{A}$ is in a subformula $\mathbf{B}$ of $\mathbf{A}$ of the form $FX^1\ldots X^n$, where the X^i are bound in $\mathbf{A}$. In other words, all free variables of a higher-order pattern occur at the leaves, or applied to a list of distinct bound variables; thus all closed formulae and first-order terms are higher-order patterns. For closure under the calculus operation, it is not sufficient to require that all term declarations are higher-order patterns, since this is satisfied trivially, as they are closed. The following definition will at least give general bindings that are higher-order patterns.

Definition 8.2.10 (Pattern Signature)
We call a term declaration $[\mathbf{A}::\mathbb{A}]$ a **pattern declaration**, iff for distinct variables X^i, where $n = \mathbf{ln}(\mathbb{A})$, the formula $\mathbf{A}\overline{X^n}$ is a higher-order pattern. Intuitively the first $\mathbf{ln}(\mathbb{A})$ bound variables in $\mathbf{A}{=}_{\beta\eta}\lambda\overline{X^l}\mathbf{B}.$ may only occur applied to distinct variables that are bound in $\mathbf{B}$. We will call the X^i (and their occurrences in $\mathbf{B}$) **essentially free**.

pattern declaration *declaration!pattern* *essentially free* *occurrence!essentially free* *free!essentially* *pattern signature* *signature!pattern*

We call a signature Σ a **pattern signature**, iff all term declarations in Σ are pattern declarations.

Lemma 8.2.11 *If Σ is a pattern signature and $h \in \overline{\Sigma} \cup \mathbb{N}$, then $\mathcal{G}^h_{\mathbb{A}}(\Sigma; \Gamma)$ is a set of higher-order patterns.*

Proof: A general binding $\mathbf{G} \in \mathcal{G}^h_{\mathbb{A}}(\Sigma; \Gamma)$ has the form $\lambda X^1 \ldots X^l.\mathbf{H}\mathbf{V}^1 \ldots \mathbf{V}^m$ where $\mathbf{V}^i = H\overline{X^l}$ and $H = X^j$ for some $j \leq l$ or $[\mathbf{H}::\mathbb{A}] \in \Sigma$. In the first case, $\mathbf{G}$ is a higher-order pattern since the H^i are the only free variables. Otherwise $\mathbf{H}\overline{Y^m}$ is a higher-order pattern for variables Y^i, since Σ is a pattern signature, i.e. the Y^i may only occur applied to distinct variables $Z^i_{k_i}$ that are bound in $\mathbf{H}$. So if we consider $\mathbf{H}\overline{\mathbf{V}^m}$ instead of $\mathbf{H}\overline{Y^m}$, the variables H^i can only occur applied to some X^i (bound in $\mathbf{G}$) and $Z^i_{k_i}$ (bound in $\mathbf{H}$ and therefore in $\mathbf{G}$); thus $\mathbf{G}$ is a higher-order pattern, since the X^i are distinct from the $Z^i_{k_i}$. □

Example 8.2.12 (Patterns in Elementary Calculus)
The Signature for elementary calculus from our running example 7.7.22, is a pattern signature, in particular, the declaration $[\lambda F, G, X. + (FX)(GX))::\mathbb{P} \to \mathbb{P} \to \mathbb{P}]$ is a pattern declaration, since $\mathbf{ln}(\mathbb{P} \to \mathbb{P} \to \mathbb{P}) = 2$ and $\lambda F, G, X. + (FX)(GX)$ is a higher-order pattern.

As we have seen in Example 8.2.4.2, $\lambda X.(PX)+(QX) \in \mathcal{G}^+_{\mathbb{P}}(\Sigma; \emptyset)$, which is a higher-order pattern is a strong general binding, but $(PX) + (QX)$, which is not fails to be a strong general binding (see Example 8.1.15), is not a higher-order pattern (the X are not bound).

Generally, it seems that most mathematical theories give rise to pattern signatures, which would account for tractability the in mathematical practice.

8.3 Sort Computation

In this section we analyze the sort computation problem. It will turn out, that the sort computation problem for arbitrary signatures and the higher-order matching problem are inter-reducible. Unfortunately, the latter is only known to be decidable in subcases [Hue76, Dow92, Wol93], and the issue is still open for the general case.

Definition 8.3.1 Let $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma)$ be a formula in β-normal form, then we define the sort system $\mathcal{S}^{\mathbf{A}}$ by $\mathcal{BS}^{\mathbf{A}} := \sharp(\mathcal{BT}) \cup \{\mathbb{A}\}$ where $\tau(\mathbb{A}) := \alpha$ and, if α is functional, $\mathfrak{d}(\mathbb{A}) := \mathfrak{d}(\sharp(\alpha))$ where $\mathfrak{r}(\mathbb{A}) := \mathfrak{r}(\sharp(\alpha))$. Here, $\sharp$ is the sort injection defined in 7.1.6. Note that we have $\mathbb{A} \parallel \sharp(\alpha)$, since $\mathfrak{d}(\mathbb{A}) = \mathfrak{d}(\sharp(\alpha))$ and $\mathfrak{r}(\mathbb{A}) = \mathfrak{r}(\sharp(\alpha))$. We now define a signature $\Sigma^{\mathbf{A}}$ by choosing $\overline{\Sigma^{\mathbf{A}}} := \Sigma$ and $\Sigma := \Sigma^\sharp, [(\lambda \overline{X^n}.\sharp(\mathbf{A}))::\overline{\mathbb{B}^n} \to \mathbb{A}]$, where $\overline{X^n} = \mathbf{Free}(\mathbf{A})$ and $\Sigma^\sharp = \{[\sharp(c)::\sharp(\alpha)] \mid c \in \Sigma_\alpha, \alpha \in \mathcal{T}\}$. Obviously, $\Sigma^\sharp$ is a valid signature (consecutively add new declarations by *ws:const*) and $\mathbf{Free}(\mathbf{A})^\sharp \vdash_{\Sigma^\sharp} \sharp(\mathbf{A})::\sharp(\alpha)$, so $\Sigma^{\mathbf{A}}$ is a valid signature by *sig:td*.

Now, we prove a Lemma that relates well-sortedness on $\Sigma^{\mathbf{A}}$ with higher-order matching.

Lemma 8.3.2 *Let $\mathbf{B} \in \mathit{wff}_\alpha(\Sigma)$ be an arbitrary formula, then the following two assertions are equivalent:*

1. *The judgment $\mathbf{Free}(\mathbf{B})^\sharp \vdash_{\Sigma^{\mathbf{A}}} \mathbf{B}::\mathbb{A}$ is provable in $\Sigma\Lambda^{\rightarrow}$, but $\mathbf{Free}(\mathbf{B})^\sharp \vdash_{\Sigma^\sharp} \mathbf{B}::\mathbb{A}$ is not.*
2. *There is a well-typed substitution $\rho \in \mathbf{SUB}(\Sigma; \Delta \to \Gamma)$ such that $\rho(\mathbf{B}) =_{\beta\eta} \mathbf{A}$ in $\Lambda^{\rightarrow}$.*

Proof: Let assertion 1. be valid, then we observe that there is only one general binding of head $\mathbf{head}(\mathbf{B})$ and sort $\mathbb{A}$, namely, $\sharp(\mathbf{A})$ itself, since we have chosen $\mathbb{A}$ to be a base sort, which has length zero. Furthermore, there cannot be weakening bindings apart from (possibly) $\sharp(\mathbf{A})$. Thus we have $\{\mathbf{B}\} = \mathcal{G}^h_{\mathbb{A}}(\Sigma, \Gamma)$ and get the sufficiency direction by the general binding lemma 8.2.7.

The sets $\mathit{wff}_\alpha(\overline{\Sigma}, W)$ are isomorphic to the $\mathit{wsf}_{\sharp(\alpha)}(\Sigma^\sharp, W^\sharp)$ and even to $\mathit{wsf}_{\sharp(\alpha)}(\Sigma^{\mathbf{A}}, W^\sharp)$, since the new term declaration cannot result in any new judgments $\Gamma \vdash_{\Sigma^{\mathbf{A}}} \mathbf{B}::\mathbb{H}$, where $\mathbb{B}$ does not contain $\mathbb{A}$. So the substitution $\sharp(\rho)$ is also a $\Sigma^{\mathbf{A}}$-substitution, and therefore we obtain the necessitation direction by *ws:inst*. □

The general binding theorem 8.2.7 can be read as an algorithm for sort computation using higher-order matching.

Theorem 8.3.3 (Sort Computation)
Sort computation in $\Sigma\Lambda^{\rightarrow}$ can be reduced to higher-order matching.

Proof: We first note that for a well-typed formula $\mathbf{A}_\alpha$ there is a $\Sigma\Lambda^{\rightarrow}$-proof of $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$, iff $\mathbf{A}$ is the Σ-instance of a general binding of sort $\mathbb{A}$. Indeed if $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$, then by the general binding lemma 8.2.7, there is a general binding $\mathbf{G} \in \mathcal{G}^h_{\mathbb{A}}(\Sigma, \Gamma)$ and a substitution ρ such that $\Gamma \vdash_\Sigma \rho::\mathcal{C}(\mathbf{G})$ and $\Gamma \vdash_\Sigma \rho(\mathbf{G}) =_{\beta\eta} \mathbf{A}$. On the other hand, if there is a well-sorted formula $\mathbf{G}$ of sort $\mathbb{A}$ (the general binding) and a Σ-substitution ρ, then $\rho(\mathbf{G})$ is also of sort $\mathbb{A}$ by 8.2.7.

Now let $\mathbf{Mat}(\flat(\mathbf{A}), \flat(\mathbf{B}))$ be a complete set of higher-order matchers (cf. definition 3.2.4) of well-typed formulae $\flat(\mathbf{A})$ and $\flat(\mathbf{B})$, then the following inference rule together with those for Σ-substitutions (cf. 7.3.1) give an alternative inference system for well-sortedness:

$$\boxed{\dfrac{\Gamma \vdash_\Sigma \sharp(\rho){::}\mathcal{C}(\mathbf{G}) \quad \rho \in \mathbf{Mat}(\flat(\mathbf{A}), \flat(\mathbf{G})) \quad \mathbf{G} \in \mathcal{G}_{\mathbb{A}}(\Sigma, \Gamma)}{\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}}\ ws{:}alt}$$

We now can convince ourselves that this inference system is terminating modulo higher-order matching. Let $\mathcal{A}{:}\,\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ be in semi-structural form, then the general binding theorem (8.2.7) also gives us semi-structural $\Sigma\Lambda^{\rightarrow}$-derivations $\mathcal{A}^i{:}\,\Gamma \vdash_\Sigma \rho(X^i){::}\mathcal{C}(\mathbf{G})(X^i)$ for all $X^i \in \mathbf{Dom}(\rho)$ such that $\mathbf{dp}(\mathcal{A}^i) < \mathbf{dp}(\mathcal{A})$ (see Lemma 8.2.8). Thus, if we consider the set $\mathcal{N}$ of depths of semi-structural $\Sigma\Lambda^{\rightarrow}$-derivations associated with the subgoal judgments $\Gamma \vdash_\Sigma \mathbf{A}^i{::}\mathbb{A}^i$, then each backward application of *ws:alt* with subsequent decomposition of $\Gamma \vdash_\Sigma \sharp(\rho){::}\mathcal{C}(\mathbf{G})$ by *wsub:ext* yields a multiset $\mathcal{N}'$ with $\mathcal{N}' < \mathcal{N}$ where $<$ is the multiset ordering on natural numbers. $\Box$

Corollary 8.3.4 *The problem of sort computation in $\Sigma\Lambda^{\rightarrow}$ and the problem of higher-order matching in wff(Σ) are inter-reducible and therefore equivalent.*

It is not clear whether the higher-order matching problem in typed λ-calculi is decidable in general. However, there are large subclasses where matching is known to be decidable, for instance third-order logic [Hue76, Dow92].

Remark 8.3.5 (Tractability and Order)
Practice in programming languages has shown that humans have great difficulties thinking in terms of functions of order greater than three. Therefore it would probably — for all practical (theorem proving) purposes — not be a significant restriction to restrict term declarations to third order logic. This would make the reconstruction of sorts for $\Sigma\Lambda^{\rightarrow}$ a computable problem, since third-order matchers are computable.

Another tractable fragment, is the class of higher-order patterns [Mil92] Restricting term declarations to these or other such classes yield instances of $\Sigma\Lambda^{\rightarrow}$, where the sort computation problem is decidable, which is of course desirable for all practical applications. We conjecture that all term declarations that come up when a user codes a mathematical theory into a sort structure will be simple and the usual matching and unification procedures will terminate on them.

Corollary 8.3.6 (Sort Computation in Pattern Signatures)
Sort computation in pattern signatures is decidable

Proof: Higher-order matching against general bindings is the special case of higher-order pattern unification, where the right hand side is closed, and thus a higher-order pattern. Thus theorem 8.3.4 gives the assertion. □

Of course, these considerations about tractability carry over to the sorted unification and matching problems.

8.4 General Σ-Unification ($\Sigma\mathcal{UT}$)

For sorted higher-order unification, we will follow the methodology developed in Chapter 3. It will turn out that with the work invested into general bindings, we can almost directly apply the results from above. Therefore, we will only point out the differences to the case of unsorted higher-order unification. As before, Σ-unification problems are formulae of the form $\exists\Gamma.\forall\Upsilon.\mathcal{E}$, where $\mathcal{E}$ is a set (represented by a conjunction) of pairs. The only difference is that the formulae in $\mathcal{E}$ are well-sorted, and that Γ is a sorted variable context. Again the context Υ is used for collecting the contexts of bound variables; thus it has to be a sorted variable context in the case of $\Sigma^r\Lambda^{\rightarrow}$ and a typed one for $\Sigma^d\Lambda^{\rightarrow}$. Finally, we have seen that sort computation may be undecidable, so for an effective algorithm, we cannot use an elimination rule like $\mathcal{SIM}(elim)$ to eliminate pairs of the form $F\overline{X^k} =^? \mathbf{A}$ by instantiating the problem with $[(\lambda\overline{X^k}.\mathbf{A})/F_{\mathbb{A}}]$, which would require to verify that $\Gamma \vdash_\Sigma \lambda\overline{X^k}.\mathbf{A}::\mathbb{A}$. Fortunately, we can solve this problem by restricting variable elimination to pairs that we know are well-sorted, namely the pairs $X_{\mathbb{A}} =^? \mathbf{G}$ newly introduced by $\mathcal{U}\Lambda(flex/rig)$, where $\mathbf{G}$ is a general binding, of sort $\mathbb{A}$. In this case we know that the substitution $[\mathbf{G}/X]$ is well-sorted by 8.2.6. Since we want to collect the partial substitutions as solved pairs, we mark such pairs with $\mathbf{A} =^?_s \mathbf{B}$. This gives us a very simple notion of solved form (all pairs have to be marked as solved), but we have to modify the unification transformations, so that they guarantee that a the substitution corresponding to a solved unification problem is indeed it's unique most general unifier (cf. Lemma 3.2.8).

Note that Σ-unifiability does not entail that both formulae of a pair have identical sets of sorts, since these sets may grow as more term declarations become applicable with instantiation. For instance, consider the unification problem $\exists\Gamma.F =^? G$ where $\Gamma(F) = \mathbb{A}$, $\Gamma(G) = \mathbb{B}$, and $\Sigma \vdash \mathbb{A} \sqsubseteq \mathbb{B}$. Nevertheless, Σ-unifiable pairs must have the same types, and moreover the sorts must be compatible.

Lemma 8.4.1 *If $\Delta \vdash_\Sigma \mathbf{A}::\mathbb{A}$ and $\Delta \vdash_\Sigma \mathbf{B}::\mathbb{B}$, and $\mathbf{A}$ and $\mathbf{B}$ are Σ-unifiable, then $\mathbb{A} \parallel \mathbb{B}$.*

Proof: The assertion is only non-trivial for $\Sigma^r\Lambda^{\rightarrow}$, where we have to show that $\mathbb{A}$ **Rdom** $\mathbb{B}$. Let $\theta \in \mathbf{wsSub}(\Sigma, \Delta \rightarrow \Gamma)$ be a Σ-unifier for $\exists\Gamma.\forall\Upsilon.\mathbf{A} =^? \mathbf{B}$, then $\Gamma \vdash_\Sigma \theta(\mathbf{A}) =_{\beta\eta} \theta(\mathbf{B})$, and we have $\Gamma \vdash_\Sigma \theta(\mathbf{A})::\mathbb{A}$ and $\Gamma \vdash_\Sigma \theta(\mathbf{B})::\mathbb{B}$

by *ws:inst* and $\Gamma \vdash_\Sigma \theta(\mathbf{A})::\mathbb{B}$ by 7.6.1. Now we get the assertion with 7.5.1. □

Σ-solved in
solved in!Σ
Σ-solved form
solved form!Σ
$\Sigma\mathcal{UT}$ transformations for Σ-unification

General Assumption 8.4.2 Since we are only interested in Σ-unification problems, where Σ-unification does not fail trivially, we assume for all Σ-unification problems $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathbf{A}^1 =^? \mathbf{B}^1 \wedge \ldots \wedge \mathbf{A}^n =^? \mathbf{B}^n$ that $\Gamma \vdash_\Sigma \mathbf{A}^i::\mathbb{A}^i$, $\Gamma \vdash_\Sigma \mathbf{B}^i::\mathbb{B}^i$, and $\mathbb{A}^i \parallel \mathbb{B}^i$ for all $i \leq n$. Note that we can decide whether a well-sorted unification problem meets this condition even in $\Sigma^r\Lambda^\rightarrow$, because we can compute the argument sorts (see 7.5.6). Furthermore, we assume all $\mathbf{A}^i$ and $\mathbf{B}^i$ to be in β-normal form.

BegNP(38)

We will now adapt the notion of solved form for the sorted case.

Definition 8.4.3 (Solved Form)
Let $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{F}$ be a unification problem with $\Gamma(X) = \mathbb{A}$. Then we call a pair $X =^? \mathbf{A}$ **Σ-solved in** $\mathcal{E}$, iff $\mathbf{A} \in \mathit{wsf}_{\mathbb{A}}(\Sigma, \Gamma)$ and X is not free in $\mathbf{A}$ or elsewhere in $\mathcal{F}$, and a unification problem $\mathcal{E}$ in **Σ-solved form**, if all of its pairs are Σ-solved in $\mathcal{E}$.

Lemma 8.4.4 *Let $\mathcal{E}$ be a unification problem in Σ-solved form, then $\mathcal{E}_\sigma$ is a most general Σ-unifier for $\mathcal{E}$.*

Proof: The proof is exactly the same as that for Lemma 3.2.8, only that we have to check that $\mathcal{E}_\sigma$ is well-sorted. As we have $\Gamma \vdash_\Sigma \sigma_\sigma(X_{\mathbb{A}})::\mathbb{A}$ for all pairs $X_{\mathbb{A}} \in \mathbf{Dom}(\sigma)$, we can combine them to a $\Sigma\Lambda^\rightarrow$-derivation $\Gamma \vdash_\Sigma \sigma::\Delta$, where $\Delta(X_{\mathbb{A}}) := \mathbb{A}$. □

EndNP(38)

Definition 8.4.5 ($\Sigma\mathcal{UT}$: Transformations for Σ-Unification)
The transformation rules for Σ-simplification are those for unsorted simplification (cf. 3.3.1), except for $\mathcal{SIM}(\mathit{elim})$, which is restricted to solved pairs, since an effective rule for variable elimination would need decidable sort computation. The transformations $\Sigma\mathcal{UT}$ consist in $\mathcal{SIM}$ together with the rules for flex/rigid and flex/flex pairs given in Figure 8.3.

$$\frac{\exists\Gamma.\forall\Upsilon.F\overline{\mathbf{U}} =^? h\overline{\mathbf{V}} \wedge \mathcal{E} \quad \Gamma(F) = \mathbb{A} \quad \mathbf{G} \in \mathcal{A}^h_{\mathbb{A}}(\Sigma, \Delta)}{\exists\Gamma, \mathcal{C}(\mathbf{G}).\forall\Upsilon.F =^?_s \mathbf{G} \wedge F\overline{\mathbf{U}} =^? h\overline{\mathbf{V}} \wedge \mathcal{E}}\ \Sigma\mathcal{UT}(\mathit{flex/rig})$$

$$\frac{\exists\Gamma.\forall\Upsilon.F\overline{\mathbf{U}} =^?_s H\overline{\mathbf{V}} \wedge \mathcal{E} \quad \Gamma(F) = \mathbb{A} \quad \mathbf{G} \in \mathcal{A}^h_{\mathbb{A}}(\Sigma, \Delta)}{\exists\Gamma, \mathcal{C}(\mathbf{G}).\forall\Upsilon.F =^? \mathbf{G} \wedge F\overline{\mathbf{U}} =^? H\overline{\mathbf{V}} \wedge \mathcal{E}}\ \Sigma\mathcal{UT}(\mathit{guess})$$

Figure 8.3: Transformations for sorted HOU

[38] New Part:

EdNote(39)

Definition 8.4.6 For signatures, where sort computation is decidable (see Remarks 8.3.6 and 8.3.5) the inference rule $\Sigma\mathcal{UT}(elim)$ in figure Figure 8.4 is effective, and can be added to $\mathcal{SIM}$.[39] In contrast to the substitution

$$\frac{\exists\Gamma.\forall\Upsilon.F\overline{X^n}=^{?}\mathbf{A}\wedge\mathcal{E}\quad X^i\in\mathbf{Dom}(\Upsilon)\quad \Gamma\vdash_{\Sigma}\lambda\overline{X}.\mathbf{A}::\Gamma(F)}{\exists\Gamma.\forall\Upsilon.F=^{?}_{s}(\lambda\overline{X^n}.\mathbf{A})\wedge\mathcal{E}}\ \Sigma\mathcal{UT}(solv)$$

Figure 8.4: An effective $\mathcal{SIM}$ rule (for tractable Signatures)

in $\Sigma\mathcal{UT}(\mathit{flex/rig})$ and $\Sigma\mathcal{UT}(guess)$, where well-sortedness of the added pair is guaranteed by 8.2.6, we have to check for well-sortedness before eliminating the variable F. The following example shows that this gives practical improvements.

Example 8.4.7 Let $\Sigma := \{[a::\mathbb{C}\to\mathbb{B}],[c::\mathbb{C}]\}$ and $\Gamma := [F::\mathbb{A}\to\mathbb{B}],[G::\mathbb{A}\to\mathbb{C}]$, then

$$\cfrac{\cfrac{\cfrac{\exists[F::\mathbb{A}\to\mathbb{B}].\forall[Y::\mathbb{A}].FY=^{?}ac}{\exists\Gamma.\forall[Y::\mathbb{A}].a(GY)=^{?}ac\wedge F=^{?}_{s}\lambda Z_{\mathbb{A}}.a(GZ)}\ \Sigma\mathcal{UT}(\mathit{flex/rig})}{\exists\Gamma.\forall[Y::\mathbb{A}].GY=^{?}c\wedge F=^{?}_{s}\lambda Z_{\mathbb{A}}.a(GZ)}\ \mathcal{SIM}(dec)}{\exists\Gamma.\forall[Y::\mathbb{A}].c=^{?}c\wedge F=^{?}_{s}\lambda Z_{\mathbb{A}}.ac\wedge G=^{?}_{s}\lambda W_{\mathbb{A}}.c}\ \Sigma\mathcal{UT}(\mathit{flex/rig})$$

The inference rule $\Sigma\mathcal{UT}(elim)$ immediately computes the Σ-unifier $[\lambda X.ac/F]$ and therefore replaces two applications of $\Sigma\mathcal{UT}(\mathit{flex/rig})$ and one of $\mathcal{SIM}(dec)$. Generally, we can see that the sort computation in the $\Sigma\mathcal{UT}(solv)$ rule is mimicked in computations like the above, but the higher-order matching is a more efficient implementation. Thus for an implementation of general higher-order sorted unification (where decidability of sort computation is open) it would be advantageous to use an $\Sigma\mathcal{UT}(elim)$ rule that residuates the higher-order matching problems and interleaves the matching computation with the search for sorted unifiers.

The soundness proofs for $\Sigma\mathcal{UT}$ are identical to those presented in Chapter 3. The only additional work needed for sorts is to verify that the sort conditions in the inference rules enforce the preservation of well-sortedness of the Σ-unifiers.

Lemma 8.4.8 *Let $\mathcal{E}\vdash_{\Sigma\mathcal{UT}}\mathcal{E}'$ by a single application of $\Sigma\mathcal{UT}(dec)$ to a pair $H\overline{\mathbf{U}^n}=^{?}H\overline{\mathbf{V}^n}$, then for any substitution θ we have*

1. *If $H\in\mathbf{supp}(\theta)$, then $\theta\in\mathbf{wsU}(\Sigma,\mathcal{E}')$ implies that $\theta\in\mathbf{wsU}(\Sigma,\mathcal{E})$.*

[39] EdNote: compare with Chad's comments in unif.tex

2. *If $H \notin \mathbf{supp}(\theta)$, then $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E})$, iff $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E}')$.*

If $\mathcal{E} \vdash_{\Sigma\mathcal{UT}} \mathcal{E}'$ by a $\Sigma\mathcal{UT}$-derivation only containing applications of the rules $\Sigma\mathcal{UT}$(flex/rig) and $\Sigma\mathcal{UT}$(guess), then $\mathbf{wsU}(\Sigma, \mathcal{E}') \subseteq \mathbf{wsU}(\Sigma, \mathcal{E})$.

For the completeness result, we also follow the methodology from chapter 3. We have to prove an analogue of the semi-termination Lemma 3.5.9 from the general binding theorem. However, since the sorted general binding theorem 8.2.7 only yields a depth condition on the size of certain semi-structural derivations, we have to adapt the measure used in the semi-termination Lemma. For a unification problem $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{E}'$ and a substitution $\theta \in \mathbf{wsSub}(\Sigma, \Delta \to \Gamma)$ we also have $\mu(\mathcal{E}, \theta) := (\mu_1(\mathcal{E}, \theta), \mu_2(\mathcal{E}))$. However for $\mu_1(\mathcal{E}, \theta)$ we take multiset of depths of semi-structural $\Sigma\Lambda^{\to}$-derivations $\mathcal{A}_X : \Gamma \vdash_\Sigma \theta(X) :: \Delta(X)$, where $X \in \mathbf{Dom}(\theta)$ is unsolved in $\mathcal{E}$. Note that since semi-structural derivations are not unique (see e.g. 8.1.23), there is a choice here, which affects well-definedness of the measure. We can make μ well-defined technically by well-ordering the set of $\Sigma\Lambda^{\to}$-derivations (it is countable) and always taking the minimal semi-structural derivations that are available. As in the unsorted case $\mu_2(\mathcal{E})$ is the multiset of depths of formulae in $\mathcal{E}$ and $\prec$ be the strict lexicographic ordering for the obvious component orderings. With this definition, the the proofs of the semi-termination lemma goes through like in the unsorted case, only that we have to use the sorted general binding theorem here.

Lemma 8.4.9 (Semi-Termination)
Let $\mathcal{E}$ be an unsolved unification problem in $\mathcal{SIM}$-normal form $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E})$, then there exists a unification problem $\mathcal{E}'$, an $\mathcal{E}'$-substitution θ' such that

$$\mathcal{E} \vdash_{\mathcal{U}\Lambda} \mathcal{E}' \qquad \theta \equiv \theta'[\mathcal{E}] \qquad \theta' \in \mathbf{U}(\mathcal{E}') \qquad \mu(\mathcal{E}', \theta') \prec \mu(\mathcal{E}, \theta)$$

Proof: Let $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{F}$ and $\mathbf{A} =^? \mathbf{B}$ be an unsolved pair in $\mathcal{F}$. Furthermore let $\theta \in \mathbf{wsSub}(\Sigma, \Gamma \to \Xi)$ be a Σ-unifier of $\mathcal{E}$. As $\mathcal{E}$ is in $\mathcal{SIM}$-normal form we have $\mathbf{A} \equiv \mathbf{M}\overline{\mathbf{U}}$ and $\mathbf{B} \equiv \mathbf{N}\overline{\mathbf{V}}$, where $\mathbf{M}$ and $\mathbf{N}$ are symbols. Note that $\mathbf{M}$ and $\mathbf{N}$ cannot be equal constants, and moreover we cannot have $\Gamma \vdash_\Sigma \theta(\mathbf{A}) =_{\beta\eta} \theta(\mathbf{B})$.

If $\mathbf{M} = \mathbf{N}$ is a variable not in $\mathbf{supp}(\theta)$, then $\mathcal{U}\Lambda(dec)$ applies. By Lemma 8.4.8 we have $\theta \in \mathbf{U}(\mathcal{E}')$ and $\mu(\mathcal{E}', \theta) \prec \mu(\mathcal{E}, \theta)$, since $\mu_1(\mathcal{E}', \theta) \preceq \mu_1(\mathcal{E}, \theta)$ and $\mu_2(\mathcal{E}') \prec \mu(\mathcal{E})$.

Otherwise either $\mathbf{M} \neq \mathbf{N}$ or $\mathbf{M} = \mathbf{N} \in \mathbf{supp}(\theta)$. In both cases, since $\mathcal{E}$ is unifiable, either $\mathbf{M}$ or $\mathbf{N}$ is an unsolved variable $F \in \mathbf{supp}(\theta)$ with $\Gamma(F) = \mathbb{A}$. Without loss of generality we assume that $\mathbf{M} = F$ and set $h := \mathbf{head}(\theta(F))$. By the general binding theorem 8.2.7 there is a general binding $\mathbf{G} \in \mathcal{A}^h_{\mathbb{A}}(\Sigma; \Xi, \Gamma)$ and a substitution ρ, such that $\Gamma \vdash_\Sigma \rho :: \mathcal{C}(\mathbf{G})$, where $\mathcal{C}(\mathbf{G}), \Gamma \vdash_\Sigma \rho(\mathbf{G}) =_{\beta\eta} \theta(F)$. Now,

monomial
polynomial
degree
non-negative

- if $\mathbf{head}(\mathbf{N}) \notin \mathbf{supp}(\theta)$, then $\mathcal{UA}(\mathit{flex/rig})$ applies.
- if $\mathbf{head}(\mathbf{N}) \in \mathbf{supp}(\theta)$ then $\mathcal{UA}(\mathit{guess})$ applies.

In all these cases we set $\theta' := \theta \cup \rho$ and have $\theta = \theta'[\mathcal{E}]$, since $\mathbf{supp}(\rho) \cap \mathbf{Dom}(\Gamma) = \mathbf{Dom}(\mathcal{C}(\mathbf{G})) \cap \mathbf{Dom}(\Gamma) = \emptyset$ and $\theta' \in \mathbf{U}(\mathcal{E}')$ by 3.5.7.

If $\mathcal{A}: \Xi, \Gamma \vdash_\Sigma \theta(F)::\mathbb{A}$ is the semi-structural $\Sigma\Lambda^{\rightarrow}$-derivation that contributes to $\mu_1(\mathcal{E},\theta)$, then the general binding theorem guarantees the existence of semi-structural $\Sigma\Lambda^{\rightarrow}$-derivations $\mathcal{R}^i: \Xi, \Gamma \vdash_\Sigma \rho(X)::\mathcal{C}(\mathbf{G})(X)$ for all $X \in \mathbf{Dom}(\theta)$. Since $\mathcal{UA}(\mathit{guess})$ and $\mathcal{UA}(\mathit{flex/rig})$ remove the variable F from the set of variables in $\mathbf{supp}(\theta)$ that are not solved in $\mathcal{E}$, and since they replace it with the set $\mathbf{Dom}(\mathcal{C}(\mathbf{G})) = \mathbf{supp}(\rho)$, we have $\mu_1(\mathcal{E}',\theta') \prec \mu_1(\mathcal{E},\theta)$. Thus we have $\mu(\mathcal{E}',\theta') \prec \mu(\mathcal{E},\theta)$. □

Building on this result, we obtain the completeness result for $\Sigma\mathcal{UT}$ with exactly the same proof as in the unsorted case.

Theorem 8.4.10 (Completeness Theorem for $\Sigma\mathcal{UT}$)
For any Σ-unification problem $\mathcal{E}$ and any Σ-substitution $\theta \in \mathbf{wsU}(\Sigma,\mathcal{E})$, there is a $\Sigma\mathcal{UT}$-derivation $\mathcal{E} \vdash_{\Sigma\mathcal{UT}} \mathcal{F}$ such that $\mathcal{F}$ is in Σ-solved form and $\Gamma \vdash_\Sigma \sigma_{\mathcal{F}} \preceq_{\beta\eta} \theta[\mathcal{E}]$.

If we combine the soundness results with the completeness result from theorem 8.4.10, we can characterize the set of solutions found by the algorithm $\Sigma\mathcal{UT}$ by the following corollary.

Corollary 8.4.11 *For any Σ-unification problem $\mathcal{E}$ the set*

$$\Sigma\mathcal{UT}(\mathcal{E}) := \{\sigma_{\mathcal{F}} \mid \mathcal{E} \vdash_{\Sigma\mathcal{UT}} \mathcal{F} \text{ and } \mathcal{F} \text{ is in } \Sigma\text{-solved form}\}$$

is a complete set of Σ-unifiers for $\mathcal{E}$.

Example 8.4.12 (Non-Negative Differentiable Functions)
Let Σ be the signature for elementary calculus from Example 7.7.22. In this setting we can answer the question whether there are differentiable functions that are non-negative by solving the following unification problem:

$$\exists[F::\mathbb{D}],[G::\mathbb{R} \to \mathbb{R}^+].F =^? G$$

This is a flex/flex problem, so we can use $\Sigma\mathcal{UT}(\mathit{guess})$ with the general imitation binding $\mathbf{G}^*_{\mathbb{R}+} = \lambda X.*(H^1X)(H^1X)$ induced by the term declaration $[\lambda X.*XX::\mathbb{R} \to \mathbb{R}^+] \in \Sigma$, thus we obtain

$$\exists\Gamma.F =^?_s \lambda X.*(H^1X)(H^1X) \wedge G =^? \lambda X.*(H^1X)(H^1X)$$

where $\Gamma = [F::\mathbb{D}],[G::\mathbb{R} \to \mathbb{R}^+],[H^1::\mathbb{R} \to \mathbb{R}]$. Now, we can apply $\mathcal{SIM}(\alpha)$ to the unsolved pair and obtain

$$\exists\Gamma.\forall[Y::\mathbb{R}].F =^?_s \lambda X.*(H^1X)(H^1X) \wedge GY =^? *(H^1Y)(H^1Y)$$

To make the $\Sigma\mathcal{UT}$-derivation more legible, we drop the declaration and delete solved pairs in Σ-unification problems in this example. To keep the sort information complete we indicate the sorts of variables in the subscript. With the weakening bindings $G =^?_s H^2_{\mathbb{P}}$ and $H^2 =^?_s H^3_{\mathbb{M}}$ from the subsort declarations $[\mathbb{P} \leq \mathbb{D}], [\mathbb{M} \leq \mathbb{P}] \in \Sigma$ we have

$$H^3_{\mathbb{M}} Y =^? * (H^1 Y)(H^1 Y)$$

We continue our $\Sigma\mathcal{UT}$-derivation by applying $\Sigma\mathcal{UT}(\mathit{flex/rig})$ with the binding $H^3_{\mathbb{M}} =^?_s \lambda X.\, * (H^4_{\mathbb{M}} X)(H^5_{\mathbb{M}} X)$ from the declaration $[\lambda F, G.\lambda X.\, * (FX)(GX) :: \mathbb{M} \to \mathbb{M} \to \mathbb{M}] \in \Sigma$ and obtain

$$H^4_{\mathbb{M}} Y =^? H^1_{\mathbb{R}\to\mathbb{R}} Y \wedge H^5_{\mathbb{M}} Y =^? H^1_{\mathbb{R}\to\mathbb{R}} Y$$

The first pair can be solved by $\Sigma\mathcal{UT}(\mathit{guess})$ using the imitation binding $H^1_{\mathbb{R}\to\mathbb{R}} =^?_s \lambda X. H^4_{\mathbb{M}}(H^6_{\mathbb{R}\to\mathbb{R}} X)$ and the subsequent projection using $H^6_{\mathbb{R}\to\mathbb{R}} =^?_s (\lambda X. X)$. This leaves us with the problem

$$H^5_{\mathbb{M}} Y =^? H^4_{\mathbb{M}} Y$$

which can be solved by $\Sigma\mathcal{UT}(\mathit{guess})$ with the bindings $H^4_{\mathbb{M}} =^?_s H^5_{\mathbb{M}} =^?_s \lambda X. X$. Note that these general binding come from the term declaration $[\lambda X. X :: \mathbb{M}] \in \Sigma$.

Collecting all partial solutions obtained in this $\Sigma\mathcal{UT}$-derivation gives us the Σ-unifier $\sigma := [\mathbf{D}/F], [\mathbf{D}/G]$ where $\mathbf{D} = (\lambda X. X * X)$, which is just square function, which is differentiable (since it is a polynomial) and non-negative everywhere.

Remark 8.4.13 (Enhanced Signatures)
Note that if we had a term declaration $[\lambda FGZ. F(GZ) :: \mathbb{M} \to \mathbb{M} \to \mathbb{M}]$ in the signature (composition of monomials yields monomials), then we would have more general bindings that yield more Σ-unifiers. In this example, we could have continued with $H^5_{\mathbb{M}} =^?_s (\lambda X. H^4_{\mathbb{M}}(H^7_{\mathbb{M}} X)$ (from the new term declaration) and $H^7_{\mathbb{M}} =^?_s (\lambda X. X)$. This gives us the Σ-unifier $\sigma := [\mathbf{D}/F], [\mathbf{D}/G]$ where $\mathbf{D} = (\lambda X. * (H^4_{\mathbb{M}} X)(H^4_{\mathbb{M}} X))$, which is just the most general expression for a monomial with even degree.

Generally, more term declarations (that is more knowledge about the domains represented by the sorts) give more power to the unification procedure, possibly resulting in better witnesses for certain properties. In this case, we could not have obtained the same unifier without the declarations, even though the information about the composition of monomials is derivable from the information in the signature (just not by sort reasoning).

problem!empty sort
empty sort problem

Chapter 9

Sort Intersection and Unification Refinements

In this chapter we will look at refinements of the sorted higher-order unification unification problem discussed in the last chapter. In particular, we will look at the treatment of the notoriously explosive $\Sigma\mathcal{UT}(guess)$ rule for flex/flex pairs and present a generalization of Huet's pre-unification procedure to $\Sigma\Lambda^{\rightarrow}$.

Unfortunately, pre-unification for $\Sigma\Lambda^{\rightarrow}$ only makes sense for signatures where all formulae have unique least sorts, as the example 9.2.1 shows. These so-called regular signatures are very desirable for practical purposes, since in an implementation they allow to annotate a term $\mathbf{A}$ only with their least sort, instead of with the — in general rather large — set of all sorts. Unfortunately, it will also turn out that the regularity is an undecidable property for signatures in general (cf. Theorem 9.4.4). Therefore we will treat regular signatures in the context of intersection sorts (where signatures are trivially regular, since for any two compatible sorts $\mathbb{A}$ and $\mathbb{B}$ their intersection $\mathbb{A} \sqcap \mathbb{B} \leq \mathbb{A}, \mathbb{B}$ is present). We will introduce the necessary machinery in the next section. In section 9.2, we present pre-unification in $\Sigma\Lambda^{\rightarrow\sqcap}$ and in section 9.3 we will look at fragments of $\Sigma\Lambda^{\rightarrow\sqcap}$, where higher-order (pattern) unification is decidable.

Finally, in section 9.4 we will look at the empty sort problem for $\Sigma\Lambda^{\rightarrow\sqcap}$. The unification algorithms presented in this and the last chapter do not alone suffice for sorted variants of refutation calculi such as those presented in part **??** of this book. These calculi use free variables to represent sets of ground terms as sets of counterexamples in the refutation. If a unifier introduces a variable ranging over an empty sort, then it does not represent a counterexample needed in the refutation. Moreover, we cannot make an non-emptiness assumption for intersection sorts like that in 1.4.14, since this would make any two compatible sorts non-disjoint (defeating our goal of a plausible and expressive sort system).

$\mu\mathcal{S}_\Sigma(\mathbf{A})$least sort of $\mathbf{A}$
regular
signatureregular
least sort
sort!least
intersection sort
sort!intersection
$\mathbb{A}\sqcap\mathbb{B}$*intersection sort*
simple sort, context, signature

9.1 Intersection Sorts

In this section we will present an extension to the sort methods developed so far, that makes signatures regular by definition.

Definition 9.1.1 (Regular Signature)
A signature Σ is called **regular**, iff each well-sorted formula has a unique least sort with respect to $\leq_\Sigma$. In regular signatures we denote the unique **least sort** of $\mathbf{A}$ by $\mu\mathcal{S}_\Sigma(\mathbf{A})$.

Not all signatures are regular, in fact even very simple ones are not.

Example 9.1.2 Let $\Sigma := \{[a{::}\mathbb{A}], [a{::}\mathbb{B}]\}$ be a valid signature, then $\mu\mathcal{S}_\Sigma(a) = \{\mathbb{A}, \mathbb{B}\}$, and therefore Σ is not regular.

We will now specify an extension $\Sigma\Lambda^{\to\sqcap}$ of the sorted λ-calculus $\Sigma\Lambda^{\to}$ developed in Chapter 7 by extending the definitions there.

Definition 9.1.3 (Intersection Sorts)
In the definition 7.1.3 of a sort system, the set of sorts is augmented by **intersection sort**s of the form $\mathbb{A}\sqcap\mathbb{B}$, provided that $\mathbb{A} \parallel \mathbb{A}$. We call all sorts that are not intersection sorts **simple** (i.e. the base sorts and the functional ones), furthermore we will call variable contexts and signature simple, iff all of their sorts are.

We extend the length function on sorts to $\Sigma\Lambda^{\to\sqcap}$ by setting $\mathbf{ln}(\mathbb{A}\sqcap\mathbb{B}) := 0$.

The semantic motivation for sort intersections of the form $\mathbb{A}\sqcap\mathbb{B}$ is to characterize the intersection of $\mathcal{D}_\mathbb{A}\cap\mathcal{D}_\mathbb{B}$, so a formula that has sorts $\mathbb{A}$ and $\mathbb{B}$ will also have the sort $\mathbb{A}\sqcap\mathbb{B}$. Naturally, this intended semantics entails that $\mathbb{A} \leq \mathbb{A}\sqcap\mathbb{B}$ for any sort $\mathbb{B}$ and that the $\sqcap$ operator is associative, commutative, and idempotent. In the rest of this section we will make these intuitions more formal. Note that intersection sorts make more sense in the descriptive than in the restrictive sorts paradigm, as the improvement of descriptive power is weaker in the latter, since they are essentially limited to base sorts there. The sort inference system $\Sigma\Lambda^{\to}$ is augmented with the

$$\frac{\Gamma\vdash_\Sigma \mathbf{A}{::}\mathbb{A} \quad \Gamma\vdash_\Sigma \mathbf{A}{::}\mathbb{B}}{\Gamma\vdash_\Sigma \mathbf{A}{::}\mathbb{A}\sqcap\mathbb{B}}\,ws{:}\sqcap$$

$$\frac{\Gamma\vdash_\Sigma \mathbf{A}{::}\mathbb{A}\sqcap\mathbb{B}}{\Gamma\vdash_\Sigma \mathbf{A}{::}\mathbb{A}}\,ws{:}\sqcap l \qquad \frac{\Gamma\vdash_\Sigma \mathbf{A}{::}\mathbb{A}\sqcap\mathbb{B}}{\Gamma\vdash_\Sigma \mathbf{A}{::}\mathbb{B}}\,ws{:}\sqcap r$$

Figure 9.1: Sort Inference in $\Sigma\Lambda^{\to\sqcap}$

rules in figure Figure 9.1. A careful rereading of all the proof of all theorems will reveal that the cases in the inductions introduced by the inference rules above are always direct consequences of the inductive hypotheses. For the subsort inference system, we obtain two new inference rules in Figure 9.2, which are direct consequences of $ws{:}\sqcap l$ and $ws{:}\sqcap r$. In fact in the presence of $ws{:}weaken$, they are equivalent to these, i.e. the former can be dropped in favor of the latter. We will now convince ourselves of some further syntactic properties of $\sqcap$ operator.

$$\dfrac{}{\Sigma \vdash \mathbb{A} \sqcap \mathbb{B} \leq_{\mathcal{R}} \mathbb{A}}\,\mathit{ior}{:}\sqcap l \qquad \dfrac{}{\Sigma \vdash \mathbb{A} \sqcap \mathbb{B} \leq_{\mathcal{R}} \mathbb{B}}\,\mathit{ior}{:}\sqcap r$$

Figure 9.2: Subsorting in $\Sigma\Lambda^{\to\sqcap}$

Lemma 9.1.4 (Type Identities in $\Sigma\Lambda^{\to\sqcap}$)
$\sqcap$ is associative, commutative and idempotent, and $\mathbb{A} \leq \mathbb{B}$ implies $\mathbb{A}\sqcap\mathbb{B} = \mathbb{B}$ and $\mathbb{A} \sqcap \mathbb{C} \leq \mathbb{B} \sqcap \mathbb{C}$.

Proof: We have assumed that $\leq_\Sigma$ is a partial ordering (cf. 7.7.19), and in particular that $\mathbb{A} = \mathbb{B}$, iff $\mathbb{A} \leq_\Sigma \mathbb{B}$ and $\mathbb{B} \leq_\Sigma \mathbb{A}$, so we only have to prove the subsort judgments. For commutativity, consider

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash_\Sigma X{::}\mathbb{A} \sqcap \mathbb{B}}\,\mathit{ws}{:}\mathit{var}}{\Gamma \vdash_\Sigma X{::}\mathbb{B}}\,\mathit{ws}{:}\sqcap r \quad \dfrac{\dfrac{}{\Gamma \vdash_\Sigma X{::}\mathbb{A} \sqcap \mathbb{B}}\,\mathit{ws}{:}\mathit{var}}{g \vdash_\Sigma X{::}\mathbb{A}}\,\mathit{ws}{:}\sqcap l}{\Gamma \vdash_\Sigma X{::}\mathbb{B} \sqcap \mathbb{A}}\,\mathit{ws}{:}\sqcap$$

where $\Gamma = [X{::}\mathbb{A} \sqcap \mathbb{B}]$, the other direction can be established by symmetry. For associativity, we do a similar construction with three branches. Finally, idempotence is a direct consequence of the presence of $ws{:}\sqcap$ and $ws{:}\sqcap l$.

Now let $\mathbb{A} \leq \mathbb{B}$, for the first type identity we note that that since $\mathbb{A}\sqcap\mathbb{B} \leq \mathbb{A}$, by $ior{:}\sqcap l$ we only have to show $\mathbb{A} \leq \mathbb{A} \sqcap \mathbb{B}$. So let $\Gamma := [X{::}\mathbb{A}]$, then we have

$$\dfrac{\dfrac{}{\Gamma \vdash_\Sigma X{::}\mathbb{A}}\,\mathit{ws}{:}\mathit{var} \quad \dfrac{\dfrac{}{\Gamma \vdash_\Sigma X{::}\mathbb{A}}\,\mathit{ws}{:}\mathit{var} \qquad \mathbb{A} \leq \mathbb{B}}{\Gamma \vdash_\Sigma X{::}\mathbb{B}}\,\mathit{ws}{:}\mathit{weaken}}{\Gamma \vdash_\Sigma X{::}\mathbb{A} \sqcap \mathbb{B}}\,\mathit{ws}{:}\sqcap$$

If we set $\Gamma := [X{::}\mathbb{A}\sqcap\mathbb{C}]$, then the the type inequality follows from

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma\vdash_\Sigma X{::}\mathbb{A}\sqcap\mathbb{C}}\,\mathit{ws{:}var}}{\Gamma\vdash_\Sigma X{::}\mathbb{A}}\,\mathit{ws}{:}\sqcap l \qquad \mathbb{A}\leq\mathbb{B}}{\Gamma\vdash_\Sigma X{::}\mathbb{B}}\,\mathit{ws{:}weaken} \quad \dfrac{\dfrac{}{\Gamma\vdash_\Sigma X{::}\mathbb{A}\sqcap\mathbb{C}}\,\mathit{ws{:}var}}{\Gamma\vdash_\Sigma X{::}\mathbb{C}}\,\mathit{ws}{:}\sqcap r}{\Gamma\vdash_\Sigma X{::}\mathbb{B}\sqcap\mathbb{C}}\,\mathit{ws}{:}\sqcap$$

□

The associativity and commutativity of $\sqcap$ allows us to delete brackets from intersection sorts, and write an intersection sort $\mathbb{A}$ as $\mathbb{A}_1\sqcap\mathbb{A}_2\sqcap\ldots\sqcap\mathbb{A}_n$ or even $\bigcap_{i=1}^{k}\mathbb{A}_i$. We will give $\sqcap$ precedence over $\to$, so $\mathbb{A}\to\mathbb{B}\sqcap\mathbb{C}$ abbreviates $\mathbb{A}\to(\mathbb{B}\sqcap\mathbb{C})$. Furthermore we will sometimes abbreviate $k-1$ applications of the $\mathit{ws}{:}\sqcap$ rule by

$$\dfrac{\Gamma\vdash_\Sigma \mathbf{A}{::}\mathbb{A}_1 \quad \ldots \quad \Gamma\vdash_\Sigma \mathbf{A}{::}\mathbb{A}_k}{\Gamma\vdash_\Sigma \mathbf{A}{::}\mathbb{A}_1\sqcap\ldots\sqcap\mathbb{A}_k}\,\mathit{ws}{:}\sqcap$$

BegNP(40)

Corollary 9.1.5 (Finiteness and Sorts)
For any type α, the set $\mathcal{S}_\alpha$ is finite.

Proof: By Remark 7.1.2, we know that the set $\mathcal{S}_\alpha^s$ of simple sorts of type α is finite. As all of the sorts in an intersection sort must be compatible, they must be of the same type. Since $\sqcap$ is associative, commutative and idempotent, $\mathcal{S}_\alpha$ is essentially the power set of $\mathcal{S}_\alpha^s$, which is finite as $\mathcal{S}_\alpha^s$ is. □

EndNP(40)

Finally, we have one more systematic sort identity for intersection sorts

Lemma 9.1.6 (Interaction of $\sqcap$ and $\to$)
$(\mathbb{A}\to\mathbb{B})\sqcap(\mathbb{A}\to\mathbb{C}) = \mathbb{A}\to\mathbb{B}\sqcap\mathbb{C}$

Proof: To show equality of sorts, we have to show the corresponding subsort judgments. Let $\Gamma = [F{::}(\mathbb{A}\to\mathbb{B})\sqcap(\mathbb{A}\to\mathbb{C})]$, then obtain the $\Sigma\Lambda^{\to}$-derivation $\mathcal{A}$

$$\dfrac{\dfrac{\dfrac{}{\Gamma\vdash_\Sigma F{::}(\mathbb{A}\to\mathbb{B})\sqcap(\mathbb{A}\to\mathbb{C})}}{\Gamma\vdash_\Sigma F{::}\mathbb{A}\to\mathbb{B}}\,\mathit{ws}{:}\sqcap l \qquad [X{::}\mathbb{A}]\vdash_\Sigma X{::}\mathbb{A}}{\Gamma,[X{::}\mathbb{A}]\vdash_\Sigma FX{::}\mathbb{B}}\,\mathit{ws{:}app}$$

[40] New Part:

sort intersection
semi-structural

and analogously a $\Sigma\Lambda^{\to}$-derivation $\mathcal{B}\colon \Gamma, [X{::}\mathbb{A}] \vdash_\Sigma FX{::}\mathbb{C}$, which we can use to construct a $\Sigma\Lambda^{\to}$-derivation for the first judgment:

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{A}}{\Gamma,[X{::}\mathbb{A}]\vdash_\Sigma FX{::}\mathbb{B}} \qquad \dfrac{\mathcal{B}}{\Gamma,[X{::}\mathbb{A}]\vdash_\Sigma FX{::}\mathbb{C}}}{\Gamma,[X{::}\mathbb{A}]\vdash_\Sigma FX{::}\mathbb{B}\sqcap\mathbb{C}}\,ws{:}\sqcap}{\Gamma\vdash_\Sigma \lambda X.FX{::}\mathbb{A}\to\mathbb{B}\sqcap\mathbb{C}}\,ws{:}abs}{\Gamma\vdash_\Sigma F{::}\mathbb{A}\to\mathbb{B}\sqcap\mathbb{C}}\,{*}\,ws{:}\beta\eta$$

For the converse direction let $\Gamma := [F{::}\mathbb{A} \to \mathbb{B}\sqcap\mathbb{C}]$, and $\mathcal{A}$ the $\Sigma\Lambda^{\to}$-derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma\vdash_\Sigma F{::}\mathbb{A}\to\mathbb{B}\sqcap\mathbb{C} \qquad [X{::}\mathbb{A}]\vdash_\Sigma X{::}\mathbb{A}}{\Gamma,[X{::}\mathbb{A}]\vdash_\Sigma FX{::}\mathbb{B}\sqcap\mathbb{C}}\,ws{:}app}{\Gamma,[X{::}\mathbb{A}]\vdash_\Sigma FX{::}\mathbb{B}}\,ws{:}\sqcap l}{\Gamma\vdash_\Sigma \lambda X.FX{::}\mathbb{A}\to\mathbb{B}}\,ws{:}abs}{\Gamma\vdash_\Sigma F{::}\mathbb{A}\to\mathbb{B}}\,{*}\,ws{:}\beta\eta$$

In the same way, we can obtain a $\Sigma\Lambda^{\to}$-proof for the judgment $\Gamma \vdash_\Sigma F{::}\mathbb{A} \to \mathbb{C}$, which we can then combine

$$\dfrac{\dfrac{\mathcal{A}}{\Gamma\vdash_\Sigma F{::}\mathbb{A}\to\mathbb{B}} \qquad \dfrac{\mathcal{B}}{\Gamma\vdash_\Sigma F{::}\mathbb{A}\to\mathbb{C}}}{\Gamma\vdash_\Sigma F{::}(\mathbb{A}\to\mathbb{B})\sqcap(\mathbb{A}\to\mathbb{C})}\,ws{:}\sqcap$$

to complete the proof of second part the assertion. □

Let us now turn to the the structure theorem to the case with intersection sorts. We will follow exactly the course of the previous proof, and take sort intersections into account as we go along. First of all, we have to augment the notion of a semi-structural derivation. So let $\mathbb{A} = \mathbb{A}_1 \sqcap \ldots \sqcap \mathbb{A}_n$ be an intersection sort, such that the $\mathbb{A}_i$ are simple sorts, then we call a $\Sigma\Lambda^{\to}$-derivation $\mathcal{A}\colon \mathbf{A}{::}\mathbb{A}$ semi-structural, iff it is of the form

$$\dfrac{\dfrac{\mathcal{A}_1}{\Gamma\vdash_\Sigma \mathbf{A}{::}\mathbb{A}_1} \quad \ldots \quad \dfrac{\mathcal{A}_n}{\Gamma\vdash_\Sigma \mathbf{A}{::}\mathbb{A}_n}}{\Gamma\vdash_\Sigma \mathbf{A}{::}\mathbb{A}}\,ws{:}\sqcap$$

where the derivations $\mathcal{A}_1$ are semi-structural without intersection sorts, i.e. of the form of definition 8.1.3, where the principal subderivations are semi-structural with intersections.

restricted semi-structural logical relation

Theorem 9.1.7 (Structure Theorem)
If $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$*, then* $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{A};\mathbb{A})$.

Proof: The definition of restricted semi-structural derivations is left unchanged, in particular, it is only defined, if $\mathbb{A}$ is already simple. The proof of Lemma 8.1.13 is independent of the sort $\mathbb{A}$, and in that of Lemma 8.1.14 the sort $\mathbb{A}$ must be simple, since $\mathcal{RSR}$ is only defined for simple sorts, thus the proof need not be augmented. For the $\Sigma\Lambda^{\to\sqcap}$-variant of Lemma 8.1.16 let $\mathbb{D} = \bigcap_{i\leq n}\mathbb{D}_i$ and $\Gamma,[X{::}\mathbb{C}] \vdash_\Sigma \mathcal{SR}(\mathbf{A};\mathbb{D})$. Then by our $\sqcap$-enhanced definition of a semi-structural $\Sigma\Lambda^{\to}$-derivation, we have $\mathcal{SR}(\mathbf{A};\mathbb{D}_i)$ and by the original Lemma 8.1.16 we have $\mathcal{SR}(\lambda X\mathbf{.A};\mathbb{C}\to\mathbb{D}_i)$. This gives us $\mathcal{SR}(\lambda X\mathbf{.A};\bigcap_{i\leq n}\mathbb{C}\to\mathbb{D}_i)$, which entails the assertion since $\bigcap_{i\leq n}\mathbb{C}\to\mathbb{D}_i = \mathbb{C}\to\bigcap_{i\leq n}\mathbb{D}_i = \mathbb{C}\to\mathbb{D}$ by Lemmata 9.1.4 and 9.1.6. Note that the depth conditions of Remark 8.1.17 survive the changes in the proof.

We extend the definition of logical relations by a clause saying that $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A};\mathbb{A}_1\sqcap\ldots\sqcap\mathbb{A}_n)$, iff $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A};\mathbb{A}_i)$ for all $\mathbb{A}_i$.

Now we have to consider the case $\mathbb{A} = \mathbb{A}_1\sqcap\ldots\sqcap\mathbb{A}_n$ in the inductions over sorts. In the proof of closure under $\beta\eta$-equality (8.1.19) we have $\mathcal{LR}(\mathbf{A};\mathbb{A})$ and want to show $\mathcal{LR}(\mathbf{B};\mathbb{A})$ for $\mathbf{A}{=}_{\beta\eta}\mathbf{B}$; by definition of $\mathcal{LR}$ we have $\mathcal{LR}(\mathbf{A};\mathbb{A}_i)$ for all i and thus $\mathcal{LR}(\mathbf{B};\mathbb{A}_i)$ by inductive hypothesis. This yields the assertion by definition of $\mathcal{LR}$.

In Lemma 8.1.20, we only have to consider the second assertion, since in the first one, $\Gamma \vdash_\Sigma \mathcal{RSR}(\mathbf{A};\mathbb{A})$ implies that $\mathbb{A}$ must be simple, so this case cannot appear. So we assume $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A};\mathbb{A})$ to show $\Gamma \vdash_\Sigma \mathcal{SR}(\mathbf{A};\mathbb{A})$. By definition of $\mathcal{LR}$ we obtain $\mathcal{LR}(\mathbf{A};\mathbb{A}_i)$ and by inductive hypothesis $\mathcal{SR}(\mathbf{A};\mathbb{A}_i)$. Since the $\mathbb{A}_i$ are simple, the corresponding semi-structural derivations can be assembled to a semi-structural derivation of $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ by *ws*:$\sqcap$. Thus we have $\mathcal{SR}(\mathbf{A};\mathbb{A})$.

Finally in 8.1.21 we need a case for *ws*:$\sqcap$. We want to show $\Xi,\Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{A});\mathbb{A})$ from $\Gamma \vdash_\Sigma \mathcal{LR}(\theta;\Delta)$ and $\Xi,\Delta \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ and are in a situation where

$$\frac{\dfrac{\mathcal{A}_1}{\Xi,\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}_1} \quad \ldots \quad \dfrac{\mathcal{A}_n}{\Xi,\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}_n}}{\Xi,\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}}\ ws{:}\sqcap$$

By inductive hypothesis we have $\Xi,\Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{A});\mathbb{A}_i)$, so by the intersection sort clause in the definition of $\mathcal{LR}$ we obtain $\Xi,\Gamma \vdash_\Sigma \mathcal{LR}(\theta(\mathbf{A});\mathbb{A})$.

Thus we have augmented all the technical Lemmata for intersection sorts and can conclude the proof of the theorem as above: Let $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$, by lemma 8.1.21 where we take θ to be the identity substitution we see that $\Gamma \vdash_\Sigma \mathcal{LR}(\mathbf{A};\mathbb{A})$, and with lemma 8.1.20 we obtain the assertion. $\square$

Now we can extend the notion of general bindings to the case of intersection sorts. Intuitively, for every simple conjunct of a sort $\mathbb{A}$, there will be a general binding.

Theorem 9.1.8 (General Binding Theorem for $\Sigma\Lambda^{\to\sqcap}$)
Let $\mathbf{A} \in wsf_{\mathbb{A}}(\Sigma, \Gamma)$ with $\mathbf{head}(\mathbf{A}) = h$, where $\mathbb{A} = \mathbb{A}_1 \sqcap \ldots \sqcap \mathbb{A}_n$, then there are n general bindings $\mathbf{G}^i \in \mathcal{G}^h_{\mathbb{A}_i}(\Sigma, \Gamma)$, a Σ-substitution ρ with $\Gamma \vdash_\Sigma \rho{::}\mathcal{C}$ and $\mathcal{C}(\mathbf{G}_i), \Gamma \vdash_\Sigma \rho(\mathbf{G}_i) {=}_{\beta\eta} \mathbf{A}$, where $\mathcal{C} := \mathcal{C}(\mathbf{G}_1), \ldots, \mathcal{C}(\mathbf{G}_n)$.

If $\mathcal{A}{:}\, \Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ is semi-structural, then there are semi-structural $\Sigma\Lambda^{\to}$-derivations $\mathcal{R}^i$ that witness $\Gamma \vdash_\Sigma \mathcal{SR}(\rho; \mathcal{C})$ with $\mathbf{dp}(\mathcal{R}^i) < \mathbf{dp}(\mathcal{A})$.

Proof: Let $\mathcal{A}{:}\, \Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$ be a semi-structural $\Sigma\Lambda^{\to}$-derivation, then $\mathcal{A}$ has the form

$$\dfrac{\dfrac{\mathcal{A}_1}{\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}_1} \quad \cdots \quad \dfrac{\mathcal{A}_n}{\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}_n}}{\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}}\, ws{:}\sqcap$$

With Theorem 8.2.7, for each $i = 1, \ldots, n$, there are general bindings $\mathbf{G}^i \in \mathcal{G}^h_{\mathbb{A}_i}(\Sigma, \Gamma)$ and Σ-substitutions ρ_i, such that $\Gamma \vdash_\Sigma \rho_i{::}\mathcal{C}_i$ and $\mathcal{C}(\mathbf{G}_i), \Gamma \vdash_\Sigma \rho(\mathbf{G}_i) {=}_{\beta\eta} \mathbf{A}$. We take $\rho := \rho_1 \cup \ldots \cup \rho_n$. As we can assume that the $\mathbf{Dom}(\rho_i) = \mathcal{C}(\mathbf{G}_i)$ are disjoint, ρ is a well-defined Σ-substitution and $\Gamma \vdash_\Sigma \rho{::}\mathcal{C}$. The assertion on the depths is a direct consequence of Lemma 8.2.7. □

This immediately gives us the analogue to the sort computation theorem 8.3.3 for $\Sigma\Lambda^{\to\sqcap}$.

Corollary 9.1.9 (Sort Computation for $\Sigma\Lambda^{\to\sqcap}$)
Sort computation in $\Sigma\Lambda^{\to\sqcap}$ can be reduced to higher-order matching.

For higher-order unification, we have to change the rules slightly: if the sort $\mathbb{A} = \mathbb{A}_1 \sqcap \ldots \sqcap \mathbb{A}_n$ of the head variable F is not simple, we have to accommodate multiple general bindings $\mathbf{G}_1, \ldots, \mathbf{G}_n$.

Definition 9.1.10 ($\Sigma\mathcal{UTI}$ Higher-Order Σ-Unification for $\Sigma\Lambda^{\to\sqcap}$)
Let $\Sigma\mathcal{UTI}$ be the higher-order unification transformations from Definition 8.4.5 where the rule $\Sigma\mathcal{UT}(\mathit{flex/rig})$ and $\Sigma\mathcal{UT}(\mathit{guess})$ rules have been replaced by the ones in Figure 9.3. In these rules we use the convention that $F =^?_s \mathbf{G}_1 {=^?} \cdots {=^?} \mathbf{G}_n$ stands for $F =^?_s \mathbf{G}_1 \wedge \mathbf{G}_1 {=^?} \mathbf{G}_2 \wedge \mathbf{G}_2 {=^?} \cdots \wedge \mathbf{G}_{n-1} {=^?} \mathbf{G}_n$.

Theorem 9.1.11 *$\Sigma\mathcal{UTI}$ is a sound and complete sorted higher-order unification algorithm for $\Sigma\Lambda^{\to\sqcap}$. In particular, for any Σ-unification problem $\mathcal{E}$ in $\Sigma\Lambda^{\to\sqcap}$, the set*

$$\Sigma\mathcal{UTI}(\mathcal{E}) := \{\sigma_{\mathcal{F}} \mid \mathcal{E} \vdash_{\Sigma\mathcal{UTI}} \mathcal{F} \text{ and } \mathcal{F} \text{ is in } \Sigma\text{-solved form}\}$$

is a complete set of Σ-unifiers for $\mathcal{E}$.

$\mathbb{A} = \mathbb{A}_1 \sqcap \ldots \sqcap \mathbb{A}_n$ and $\mathcal{C} = \mathcal{C}(\mathbf{G}_1), \ldots, \mathcal{C}(\mathbf{G}_n)$

$$\frac{\exists\Gamma.\forall\Upsilon.F\overline{\mathbf{U}}{=}^{?}h\overline{\mathbf{V}} \wedge \mathcal{E} \quad \Gamma(F) = \mathbb{A} \quad \mathbf{G}_i \in \mathcal{A}^h_{\mathbb{A}_i}(\Sigma, \Delta)}{\exists\Gamma, \mathcal{C}.\forall\Upsilon.F =^?_s \mathbf{G}_1 {=}^? \cdots {=}^? \mathbf{G}_n \wedge F\overline{\mathbf{U}}{=}^{?}h\overline{\mathbf{V}} \wedge \mathcal{E}}$$

$$\frac{\exists\Gamma.\forall\Upsilon.F\overline{\mathbf{U}} =^?_s H\overline{\mathbf{V}} \wedge \mathcal{E} \quad \Gamma(F) = \mathbb{A} \quad \mathbf{G}_i \in \mathcal{A}^h_{\mathbb{A}_i}(\Sigma, \Delta)}{\exists\Gamma, \mathcal{C}.\forall\Upsilon.F{=}^?\mathbf{G}_1 {=}^? \cdots {=}^? \mathbf{G}_n \wedge F\overline{\mathbf{U}}{=}^{?}H\overline{\mathbf{V}} \wedge \mathcal{E}}$$

Figure 9.3: Transformations for sorted HOU

Proof: For the completeness proof in the we only have to consider the semi-termination result 8.4.9. In the proof we have to reexamine the case, where either $\mathbf{M}$ or $\mathbf{N}$ is an unsolved variable $F \in \mathbf{supp}(\theta)$ at the head with $\Gamma(F) = \mathbb{A} = \bigcap_{i=1}^n \mathbb{A}_i$. Again, we can assume without loss of generality that $\mathbf{M} = F$. By the general binding theorem 9.1.8 there are general bindings $\mathbf{G}_i \in \mathcal{A}^h_{\mathbb{A}_i}(\Sigma; \Xi, \Gamma)$ and a substitution ρ, such that $\Gamma \vdash_\Sigma \rho{::}\mathcal{C}$, where $\mathcal{C}(\mathbf{G}_i), \Gamma \vdash_\Sigma \rho(\mathbf{G}_i){=}_{\beta\eta}\theta(F)$. Thus we can continue the proof as in above arguing about the principal subderivations of the semi-structural $\Sigma\Lambda^{\to}$-derivations. □

Practical implementations of the $\Sigma\mathcal{UTI}$ algorithm would benefit a couple of simple optimizations.

Remark 9.1.12 (Pre-Computing Simultaneous Unifiers)
The inference rules in Figure 9.3 are highly branching: if $\mathbb{A} = \mathbb{A}_1 \sqcap \ldots \sqcap \mathbb{A}_n$, then the flex/rigid rule can generate m^2 possible resulting unification problems, where m is the number general bindings of sort $\mathbb{A}$ and head h. Note that these general bindings have to be unified simultaneously (as specified in the sub-problem $\mathcal{E}^h_{\mathbb{A}} := \mathbf{G}_1 =^? \mathbf{G}_2 \wedge \mathbf{G}_2 =^? \cdots =^? \mathbf{G}_n$). Furthermore, the unifiers of $\mathcal{E}^h_{\mathbb{A}}$ only depend information in Σ, so the the solutions to the $\mathcal{E}^h_{\mathbb{A}}$ can be pre-computed for all $\mathbb{A}$ and h and stored in a table. Solutions for higher-type $\mathcal{E}^h_{\mathbb{A}}$ can be computed from lower-type ones; therefore the table is finite. This results in a significant practical speedup, in particular in cases where $\mathcal{E}^h_{\mathbb{A}}$ is unsolvable.

Remark 9.1.13 (Sort Constraints)
Note that if $\mathbb{A} = \bigcap_{i=1}^n \mathbb{C}_{1j} \to \ldots \to \mathbb{C}_{lj} \to \mathbb{D}_j$, then the general bindings $\mathbf{G}$ induced by a particular term declaration $[\mathbf{H}{::}\mathbb{H}] \in \Sigma$ only differ in the sorts of the variables introduced for $\mathbf{G}$ (they only depend on the sorts $\mathbb{H}$ and $\mathbb{A}$). Thus we can write this set of general bindings more compactly by introducing sort variables and specifying sort constraints involving $\mathbb{A}$ and $\mathbb{H}$. We could then alter the unification algorithm by augmenting the unification problems to include sort constraints, introducing constraint simplification rules into

the algorithm. [KP93] explores this for signatures that only contain constant declarations. If we for instance have the signature (we will use this again in example 10.1.2)

$$[\wedge{::}\mathbb{T} \to \mathbb{T} \to \mathbb{T}], [\wedge{::}\mathbb{T} \to \mathbb{F} \to \mathbb{F}], [\wedge{::}\mathbb{F} \to \mathbb{T} \to \mathbb{F}], [\wedge{::}\mathbb{F} \to \mathbb{F} \to \mathbb{F}]$$

then we have

$$\begin{array}{rcl} \mathbf{G}^{\wedge}_{\mathbb{T}\to\mathbb{T}\sqcap\mathbb{F}\to\mathbb{T}} & = & \lambda X. \wedge (H^1X)(H^2X) \\ \mathcal{C}(\mathbf{G}) & = & [H^1{::}\mathbb{T} \to \mathbb{A}^1_1 \sqcap \mathbb{F} \to \mathbb{A}^1_2], [H^2{::}\mathbb{T} \to \mathbb{A}^2_1 \sqcap \mathbb{F} \to \mathbb{A}^2_2] \end{array}$$

where the sort variables $\mathbb{A}^i_j$ are solutions to the following sort constraint on the sorts of the new variables H^i

$$\begin{array}{r} \mathbb{A}^1_1 \to \mathbb{A}^1_2 \to \mathbb{F} \\ \sqcap\mathbb{A}^2_1 \to \mathbb{A}^2_2 \to \mathbb{T} \end{array} =^? \begin{array}{r} \mathbb{T} \to \mathbb{T} \to \mathbb{T} \sqcap \mathbb{T} \to \mathbb{F} \to \mathbb{F} \\ \sqcap\mathbb{F} \to \mathbb{T} \to \mathbb{F} \sqcap \mathbb{F} \to \mathbb{F} \to \mathbb{F} \end{array}$$

which has the three solutions:

$$\begin{array}{ll} H^1{::}\mathbb{F} \to \mathbb{T} \sqcap \mathbb{T} \to \mathbb{F} & H^2{::}\mathbb{F} \to \mathbb{T} \sqcap \mathbb{T} \to \mathbb{T} \\ H^1{::}\mathbb{F} \to \mathbb{T} \sqcap \mathbb{T} \to \mathbb{F} & H^2{::}\mathbb{F} \to \mathbb{T} \sqcap \mathbb{T} \to \mathbb{F} \\ H^1{::}\mathbb{F} \to \mathbb{T} \sqcap \mathbb{T} \to \mathbb{T} & H^2{::}\mathbb{F} \to \mathbb{T} \sqcap \mathbb{T} \to \mathbb{F} \end{array}$$

9.2 Pre-Σ-Unification ($\Sigma\mathcal{PT}$)

Let us now consider pre-unification for $\Sigma\Lambda^{\to\sqcap}$. In $\Lambda^{\to}$, pre-unification builds on the observation that flex/flex pairs are always solvable by instantiating constant functions for the head variables. Basically the constant functions absorb their arguments. In $\Sigma\Lambda^{\to}$ the notion of pre-unification does not really make sense: We can see how this gets us into problems if we naively try to define pre-unification as in the unsorted case (see section 3.6).

Example 9.2.1 (Non-Regular Pre-Σ-Unification)
The Σ-substitution $[c/X], [c/Y]$ is the only Σ-unifier of the unification problem $\exists[X{::}\mathbb{A}], [Y{::}\mathbb{B}].X =^? Y$, but it can only be found by applying some kind of $\Sigma\mathcal{UT}(guess)$ transformation.

Even in $\Sigma\Lambda^{\to\sqcap}$ the solution to the flex/flex problem is not as simple as in the unsorted case, since heads of flex/flex pairs can include variables of functional base sort $\mathbb{A}$. In this case flex/flex pairs are not solvable independently of their arguments, since in general the constant functions needed for absorbing the arguments are not of sort $\mathbb{A}$. Our solution to this problem is to modify the definition of pre-solved pairs and to keep the guess rule, but to restrict its application to cases where it is unavoidable.

contains constant functions
functions!constant
constant functions!contains
$\mathbf{wsPU}(\Sigma,\mathcal{E})$ *pre-unifiers for* $\mathcal{E}$
$=^p$ *pre-Σ-equality*
pre-equal
pre-Σ-unifier
$\mathcal{E}_p$ *subsystem of pre-Σ-solved pairs in* $\mathcal{E}$
pre-Σ-solved form
form!pre-Σ-solved
$\Sigma\mathcal{PT}$ *pre-Σ-unification*
$\Sigma\mathcal{PT}$

Definition 9.2.2 (Sorts with Constant Functions)
We say that a sort $\mathbb{A}$ **contains constant functions** with target sort $\mathbb{B}$, iff $\mathfrak{r}^i(\mathbb{A}) = \mathbb{B}$ for some i and the judgment $\Gamma, [Y::\mathbb{B}] \vdash_\Sigma \lambda\overline{X^i}.Y::\mathbb{A}$ is provable in $\Sigma\Lambda^{\to}$ (note that this is only non-trivial, if $\mathbf{ln}(\mathbb{A}) < i$)

Example 9.2.3 Examples of classes of functions that can be modeled by functional base sorts that contain constant functions are (multi-variable) polynomials, continuous, and differentiable functions (see example 7.7.22); counterexamples are the classes of surjections, bijections or strictly monotonic functions.

Definition 9.2.4 (Pre-Equality, Pre-Unifier)
Let $\Gamma \vdash_\Sigma \mathbf{A}{=^p}_{\beta\eta}\mathbf{B}$ (we call $\mathbf{A}$ and $\mathbf{B}$ **pre-equal** in Γ), iff $\Gamma \vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{B}$ or $\mathbf{A} = F\overline{\mathbf{U}^n}$, and $\mathbf{B} = G\overline{\mathbf{V}^k}$, here $\Gamma(F)$ and $\Gamma(G)$ contain constant functions of target sort $\mathfrak{r}^k(\Gamma(F)) = \mathfrak{r}^m(\Gamma(G))$.

Let $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{E}'$ be a Σ-unification problem, then we call an Σ-substitution σ a **pre-Σ-unifier** of the pair $\mathbf{A} =^? \mathbf{B} \in \mathcal{E}'$, iff $\Delta, \Gamma \vdash_\Sigma \sigma(\mathbf{A}){=^p}_{\beta\eta}\sigma(\mathbf{B})$. We denote the set of pre-Σ-unifiers by $\mathbf{wsPU}(\Sigma,\mathcal{E})$.

Definition 9.2.5 (Pre-Σ-Solved Form)
Let $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{E}'$ be a Σ-unification problem, then $\mathcal{E}'$ is in **pre-Σ-solved form**, iff all pairs in $\mathcal{E}'$ are Σ-solved or pre-Σ-solved. If $\mathcal{E}$ is a Σ-unification problem in pre-Σ-solved form, then we can write $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{E}_\sigma \wedge \mathcal{E}_p$, where the pairs in $\mathcal{E}_p$ are pre-Σ-equal in Γ, but Σ-unsolved.

Remark 9.2.6 Let $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{F}$ be a Σ-unification problem in pre-Σ-solved form such that $\mathcal{P} := F\overline{\mathbf{U}^n} =^? G\overline{\mathbf{V}^m}$ is a pre-Σ-equal pair in $\mathcal{F}$, and let H be a variable not in $\mathbf{Dom}(\Gamma)$, then $[H::\mathfrak{r}^n(\Gamma(F))] \vdash_\Sigma \lambda\overline{X^n}.H::\mathbb{A}$ and $[H::\mathfrak{r}^n(\Gamma(F))] \vdash_\Sigma \lambda\overline{X^m}.H::\mathbb{B}$ as $\mathbb{A}$ and $\mathbb{B}$ contain constant functions of target sort $\mathfrak{r}^n(\Gamma(F)) = \mathfrak{r}^m(\Gamma(G))$. Thus

$$\sigma_{\mathcal{P}} := [\lambda\overline{X^n}.H/F], [\lambda\overline{X^m}.H/G]$$

is a Σ-unifier for $\mathcal{E}$, since $[H::\mathfrak{r}^n(\Gamma(F))] \vdash_\Sigma \sigma_{\mathcal{F}}(F\overline{\mathbf{U}^n}){=}_{\beta\eta}H{=}_{\beta\eta}\sigma_{\mathcal{F}}(G\overline{\mathbf{V}^m})$. Thus pre-Σ-unifiers can always be extended to Σ-unifiers by finding trivial Σ-unifiers for the pre-Σ-solved pairs.

Let $\mathcal{E} = \exists\Gamma.\forall\Upsilon.\mathcal{E}_\sigma \wedge \mathcal{E}_p$ be a Σ-unification problem in pre-Σ-solved form, then we can construct a Σ-unifier θ for $\mathcal{E}_p$ as above, and see that $\theta \cup \sigma$ is a Σ-unifier of $\mathcal{E}$. Therefore a Σ-unification problem $\mathcal{E}$ is pre-Σ-unifiable, iff it is Σ-unifiable.

Definition 9.2.7 ($\Sigma\mathcal{PT}$: Transformations for Pre-Σ-Unification)
We define the set $\Sigma\mathcal{PT}$ of **transformations for pre-Σ-unification** by

modifying $\Sigma\mathcal{UT}$ (8.4.5): the inference rules of $\Sigma\mathcal{PT}$ are obtained from their $\Sigma\mathcal{UT}$ counterparts by requiring that they may not be performed on a pre-Σ-solved pair.

With this definition we immediately obtain the soundness of $\Sigma\mathcal{PT}$, since all $\Sigma\mathcal{PT}$-transformations are special cases of the $\Sigma\mathcal{UT}$-transformations.

Theorem 9.2.8 (Completeness Theorem for $\Sigma\mathcal{PT}$)
For any Σ-unification problem $\mathcal{E}$ and any $\theta \in \mathbf{wsPU}(\Sigma, \mathcal{E})$, there is a $\Sigma\mathcal{PT}$-derivation $\mathcal{E} \vdash_{\Sigma\mathcal{PT}} \mathcal{F}$ such that $\mathcal{F}$ is in pre-Σ-solved form and $\sigma_{\mathcal{F}} \preceq^{p}_{\beta\eta} \theta[\mathcal{E}]$.

Proof sketch: The proof for the completeness of $\Sigma\mathcal{PT}$ is analogous to that of 8.4.10, using an analogue of the semi-termination lemma 8.4.9. For the proof of this we observe that the restriction of the $\Sigma\mathcal{UT}$ inference rules to $\Sigma\mathcal{PT}$ inference rules by forbidding applications of $\Sigma\mathcal{UT}$(*flex*/*rig*) to flex/flex pairs does not affect the result, since the forbidden applications to unsolved flex/flex pairs can be simulated by a $\Sigma\mathcal{PT}$(*guess*). $\Box$

Corollary 9.2.9 *$\Sigma\mathcal{PT}$ is a complete pre-Σ-unification procedure. Moreover, if $\mathcal{E}$ is a Σ-unification problem, then the set $\Sigma\mathcal{PT}(\mathcal{E}) := \{\sigma \mid \mathcal{E} \vdash_{\Sigma\mathcal{PT}} \mathcal{E}_\sigma\}$ is a complete set of pre-Σ-unifiers for $\mathcal{E}$.*

BegNP(41)

Example 9.2.10 Let us reconsider the example 8.4.12, where we discussed $\Sigma\mathcal{UT}$ unification for $\exists[F::\mathbb{D}], [G::\mathbb{R} \to \mathbb{R}^+].F =^? G$. Since $\mathbb{M}$ and $\mathbb{R}^+$ are base sorts, and therefore have length 0, we have to execute the $\Sigma\mathcal{UT}$(*guess*) rule and proceed as in 8.4.12. The last two $\Sigma\mathcal{UT}$(*guess*) steps are unnecessary since both $H^4_{\mathbb{M}}Y =^? H^1_{\mathbb{R}\to\mathbb{R}}Y$ and $H^5_{\mathbb{M}}Y =^? H^1_{\mathbb{R}\to\mathbb{R}}Y$ (see page cclxxii) are pre-Σ-equal and $\mathbb{M}$ contains constant functions.

EndNP(41)

9.3 Sorted Higher-Order Patterns

Now we are set up to consider a generalization of the higher-order pattern unification to the sorted setting. The observation in Lemma 8.2.11 will allow us to extend the higher-order pattern unification algorithm from section 3.8 to the sorted case. Note that we will not have decidability of unification even for the case of pattern signatures, since it does not even hold in the first-order case [SS89]. However, we will be able to extend the decidability results for first-order sorted logics(cf. [SS89, Uri92, Wei96b, Wei96a]) to the higher-order pattern setting.

[41] New Part:

closed under cross-sections
constant functions

The construction of an algorithm for unification of sorted higher-order patterns is straightforward using the ideas from the unsorted case. Therefore, we will not make all of the details explicit, but refer the reader to section 3.8. However, just like in the case of sorted pre-unification we have to ensure that the methods from the unsorted case do not violate sort conditions. In contrast to the flex/flex case where we had to retain the $\Sigma\mathcal{UT}(guess)$ rule in pre-unification for pairs where one of the head variables was not closed under cross-sections, in the pattern case, we have no method to fall back onto without leaving the fragment. Therefore, we restrict our analysis to signatures, where all functional base sorts have good closure properties.

Definition 9.3.1 (Closed under Cross-Sections)
We say that a sort $\mathbb{A}$ is **closed under cross-sections**, iff for any partial permutation φ from k to $n = \mathbf{ln}(\tau(\mathbb{A}))$, there is a sort $\mathbb{A}^\varphi$, such that $\mathbb{A}^\varphi \parallel \overline{\mathfrak{d}^{\varphi(k)}(\mathbb{A})} \to \mathfrak{r}^n(\mathbb{A})$ and $[H{::}\mathbb{A}^\varphi] \vdash_\Sigma \lambda\overline{X^n}.H\overline{X^{\varphi(k)}}{::}\mathbb{A}$.

Remark 9.3.2 (Constant Functions)
Note that $\lambda\overline{X^n}.H\overline{X^{\varphi(k)}}{=}_\eta\lambda\overline{X_{\psi(n-k)}}.H$ for some partial permutation ψ from $n-k$ to n, such that $\Im(\varphi) \cup \Im(\psi) = \{1,\ldots,n\}$. If $\mathbb{A}$ is closed under cross-sections, we can choose $\varphi(i) := i$ and $\psi(j) := k+j$ for $1 \leq i \leq k$ and $1 \leq j \leq n-k$, so in particular, $\mathbb{A}$ contains arbitrary constant functions.

Example 9.3.3 Examples of classes of functions that can be modeled by functional base sorts that contain constant functions are (multi-variable) polynomials, continuous, and differentiable functions (see example 7.7.22); counterexamples are the classes of surjections, bijections or strictly monotonic functions.

General Assumption 9.3.4 For this section, we assume that all functional base sorts are closed under cross sections.

This assumption need not pose a grave restriction on applications, such as sorted variants of λ-PROLOG or sorted theorem provers, since it only means that sets of functions that are not closed under cross-sections should not be formalized as sorts, but rather as unary predicates.

Definition 9.3.5 (Sorted Pattern Unification)
The inference rules for $\mathcal{U}$Pat are those of $\Sigma\mathcal{UT}$, where the $\Sigma\mathcal{UT}(guess)$ rule is is replaced by the flex/flex rules in Figure 9.4. In the first rule ρ picks out all arguments, where φ and ψ coincide. Note that the new binding for F is well-sorted, since we have assumed all functional sorts to be closed under cross-sections. In the $\mathcal{U}$Pat(*diff*) the new bindings are again well-sorted by construction, since we have declared a suitable intersection sort for the variable H.

$\Upsilon = [X^1 :: \mathbb{A}_1], \ldots, [X^n :: \mathbb{A}_n]$, $\Gamma(F) = \mathbb{B}$, $\Gamma(G) = \mathbb{C}$
φ, ψ are partial permutations from k to n; χ from l to n.

$$\frac{\exists\Gamma.\forall\Upsilon.F\overline{X^{\varphi(k)}} =^? F\overline{X^{\psi(k)}} \wedge \mathcal{E}}{\exists\Gamma, [H :: \mathbb{B}^{\rho}].\forall\Upsilon.\mathcal{E} \wedge \begin{array}{l} F\overline{X^{\varphi(k)}} =^? F\overline{X^{\psi(k)}} \wedge \\ F =^?_s \lambda\overline{Y^k_{\mathbb{A}_{\varphi(k)}}}.H\overline{Y^{\rho(l)}} \wedge \mathcal{E} \end{array}} \;\mathcal{U}\mathrm{Pat}(\mathit{same})$$

where ρ is a partial permutation from k to l, such that $\rho(i) = \varphi(j)$, iff $\varphi(j) = \psi(j)$.

$$\frac{\exists\Gamma.\forall\Upsilon.F\overline{X^{\varphi(k)}} =^? G\overline{X^{\chi(l)}} \wedge \mathcal{E}}{\exists\Gamma, [H :: \mathbb{B}^{\varphi'} \sqcap \mathbb{C}^{\chi'}].\forall\Upsilon.\mathcal{E} \wedge \begin{array}{l} F\overline{X^{\varphi(k)}} =^? G\overline{X^{\chi(l)}} \wedge \\ F =^?_s \lambda\overline{Y^k_{\mathbb{A}_{\varphi(k)}}}.H\overline{Y^{\varphi'(m)}} \wedge \\ G =^?_s \lambda\overline{Y^l_{\mathbb{A}_{\chi(l)}}}.H\overline{Y^{\chi(m)}} \end{array}} \;\mathcal{U}\mathrm{Pat}(\mathit{diff})$$

where φ' and χ are partial permutations from m into k, l, such that $\varphi'(m) = i$ and $\chi(m) = j$, iff $\varphi(i) = \chi(j)$.

Figure 9.4: flex/flex rules for sorted pattern unification

Soundness of these rules is a consequence of the soundness of the $\Sigma\mathcal{UT}$ rules they specialize. Pattern conservation of the simplification rules is straightforward as in the unsorted case. In the flex/rigid case conservation is a consequence of the fact that rigid approximating bindings are higher-order patterns (see Lemma 8.2.11), and the flex/flex case, where $\Sigma\mathcal{UT}$ would have used flexible bindings is treated by $\mathcal{U}\mathrm{Pat}(\mathit{same})$ and $\mathcal{U}\mathrm{Pat}(\mathit{diff})$, which conserves higher-order patterns by construction.

For completeness, we note that the flex/rigid case is no longer deterministic (as in the unsorted case), since in general there are several approximating bindings for the same head and type. Thus we cannot hope to have a unitary unification problem. However the flex/flex case, which does not involve any considerations about the head can be solved exactly as in the unsorted case, making $\mathcal{U}\mathrm{Pat}(\mathit{same})$ and $\mathcal{U}\mathrm{Pat}(\mathit{diff})$ complete unification transformations.

Theorem 9.3.6 (Completeness for $\mathcal{U}\mathrm{Pat}$)
If Σ is a pattern signature, then $\mathcal{U}\mathrm{Pat}$ is a sound and complete for sorted higher-order patterns.

In order to achieve decidability, we have to restrict the signatures further. Here, we take the decidability results for the first-order case as guiding

BegNP(42) elementary term declaration!elementary declaration!elementary **elementary** signature!elementary declaration!subsort subsort declaration constant declaration declaration!constant

intuition.

Definition 9.3.7 (Elementary Signature)
Let Σ be a pattern signature (cf. Definition 8.2.10), then we will call $\mathcal{D} := [\mathbf{H}::\mathbb{H}] \in \Sigma$ **elementary** iff $\mathbf{H} =_{\beta\eta} \lambda\overline{Z^l}.\lambda\overline{W^n}.h\mathbf{V}^1 \ldots \mathbf{V}^m$, where

1. h is a constant or $h = Z^p$ for some $p \leq l = \mathbf{ln}(\mathbb{H})$
2. $n \leq \mathbf{ln}(\tau(\mathbb{H})) - l$ and $m = \mathbf{ln}(\tau(h)) - \mathbf{ln}(\tau(\mathbb{H})) + l + n$,
3. $\mathbf{V}^i$ is $Z^{j_i}\overline{W^n}$ or W^{k_i}, for some $j_i \leq l$ and $k_i \leq n$, such that $j_p \neq j_q$ and $k_p \neq k_p$ for $p \neq q$.

We will call a signature **elementary**, iff all of its term declarations are.

EdNote(43) **Lemma 9.3.8** *Subsort declarations and constant declarations*[43] *are elementary.*

Proof: Subsort declarations $\mathbb{A} \leq \mathbb{B}$ are term declarations of the form $[\lambda X.X::\mathbb{H}]$, where $\mathfrak{d}(\mathbb{H}) = \mathbb{A}$ and $\mathfrak{r}(\mathbb{H}) = \mathbb{B}$. If we η-expand, we have $\lambda X\overline{Z^{l-1}}.X^1\overline{W^{l-1}}$ for $l = \mathbf{ln}(\mathbb{H})$, which is of the required form: for $n = 0$ we have $m = \mathbf{ln}(\tau(X)) - \mathbf{ln}(\tau(\mathbb{H})) + l = l - 1$, since $\mathbf{ln}(\tau(()X)) = \mathbf{ln}(\tau(()\mathbb{H} - 1))$.

Analoguously, we can η-expand a constant declaration of the form $[c::\mathbb{C}]$ to $\lambda X\overline{Z^l}.c\overline{Z^l}$ for $l = \mathbf{ln}(\mathbb{C})$. □

Example 9.3.9 (Elementary Calculus) The only term declaration that is not elementary in the signature in Example 7.7.22 is $[\lambda X.{*}XX::\mathbb{R} \to \mathbb{R}^+]$: it breaks the distinctness condition (3) in Definition 9.3.7.

The significance of elementary signatures is that the general bindings for elementary term declarations directly correspond to the unsorted general bindings, i.e. for any $\mathbf{G} \in \mathcal{G}^h_{\mathbb{A}}$ we have $\flat(\mathbf{G}) = \mathcal{G}^h_{\flat(\mathbb{A})}$. As a consequence, for simple Σ, any $\Sigma\mathcal{UT}$-derivation directly corresponds to a $\mathcal{U}\Lambda$-derivation and similarly for $\Sigma\mathcal{PT}$ and $\mathcal{U}$Pat.

Lemma 9.3.10 *If Σ is an elementary sorted pattern signature and $\mathcal{D}: \mathcal{E} \vdash \mathcal{F}$ is a sorted higher-order pattern unification derivation, then there is an unsorted higher-order pattern derivation $\flat\mathcal{D}: \flat\mathcal{E} \vdash \flat\mathcal{F}$.*

Proof: Without loss of generality we assume that $\mathcal{D}$ consists of exctly one step (the full assertion by induction on the length of $\mathcal{D}$) and proceed by a case distinction on the $\mathcal{U}$Pat inference rules. The only interesting cases are
EdNote(44) the flex/rigid rules and the flex/flex rules from Figure 9.4.[44]

[42] New Part: re-check everything carefully.
[43] EdNote: introduced before?
[44] EdNote: and the elimination rule, but that should not be a problem, since it is more fine-grained in the sorted case.

So if $\mathcal{D}$ is $\Sigma\mathcal{UT}(\mathit{flex}/\mathit{rig})$, then we have the following situation: *partial permutation*

$$\cfrac{\cfrac{\cfrac{\mathcal{E} \wedge F_{\mathbb{A}}\overline{\mathbf{A}^p} =^? h\overline{\mathbf{B}^q}}{\mathcal{E}' \wedge \mathbf{G}_{\mathbb{A}}\overline{\mathbf{A}^p} =^? h\overline{\mathbf{B}^q}}}{\mathcal{E}' \wedge h\overline{\mathbf{C}^p} =^? h\overline{\mathbf{B}^q}\overline{\mathbf{Z}^{p-q}}}}{\mathcal{E}' \wedge \overline{\mathbf{C}^q =^? \mathbf{B}^q} \wedge \overline{\mathbf{C}^{q+1} =^? \mathbf{Z}^{p-q}}}$$

first $F_{\mathbb{A}}$ is bound to a rigid[45] general binding, and then the remaining λ-binder is eliminated by simplification. By Definition 8.2.2 we have $\mathbf{G}_{\mathbb{A}} := \lambda X^1 \ldots X^n\mathbf{.H}(H^1\overline{X^n})\ldots(H^l\overline{X^n})$ where $n = \mathbf{ln}(\mathbb{A})$ and $[\mathbf{H}{::}\mathbb{H}]$ is a term declaration in Σ, or $\mathbf{H} = X^j$[46]. In the first case $[\mathbf{H}{::}\mathbb{H}]$ is elementary, so EdNote(45) EdNote(46)

$$\mathbf{H} = \lambda\overline{Z^l}\mathbf{.}\lambda\overline{W^k}\mathbf{.}h(Z^{\varphi(1)}\overline{W^k})\ldots(Z^{\varphi(m)}\overline{W^k})$$

where $l = \mathbf{ln}(\mathbb{H})$, $k = \mathbf{ln}(\tau(\mathbb{H})) - \mathbf{ln}(\mathbb{H})$, φ is a partial permutation from $m = \mathbf{ln}(\tau(h))$ to $l \geq m$, and h is a constant. Thus

$$\mathbf{G} =_\beta \lambda\overline{X^n}\mathbf{.}\lambda\overline{W^k}\mathbf{.}h(H^{\varphi(1)}\overline{X^n}\,\overline{W^k})\ldots(H^{\varphi(m)}\overline{X^n}\,\overline{W^k})$$

and the sort erasure of $\mathbf{G}$ is the imitation binding $\mathbf{G}^h_{\tau(\mathbb{A})}$ since

$$\mathbf{ln}(\tau(\mathbb{A})) = n + \mathbf{ln}(\tau(\mathfrak{r}^n(\mathbb{A}))) = n + \mathbf{ln}(\tau(\mathfrak{r}^l(\mathbb{H}))) = n + \mathbf{ln}(\tau(\mathbb{H})) - \mathbf{ln}(\mathbb{H}) = n + k$$

(which is a higher-order pattern by Definition 3.8.1). This gives us the assertion in this subcase.

If $\mathbf{H} = X^j$, then $\mathbf{G} := \lambda\overline{X^n}\mathbf{.}X^j(H^1\overline{X^n})\ldots(H^l\overline{X^n})$. Let $\mathcal{D}'$ be an $\mathcal{U}\Lambda(\mathit{flex}/\mathit{rig})$ step with the j-projection binding

$$\mathbf{G}^j_{\tau(\mathbb{A})} = \lambda X^n\mathbf{.}\lambda W^k\mathbf{.}X^j(H^1\overline{X^n}\,\overline{W^k})\ldots(H^l\overline{X^n}\,\overline{W^k})(K^1\overline{X^n}\,\overline{W^k})\ldots(K^k\overline{X^n}\,\overline{W^k}))$$

followed by k applications of $\mathcal{U}\Lambda(\mathit{flex}/\mathit{rig})$ to the the K^i $i+n$-projections for[47] EdNote(47)

[48] □ EdNote(48)

Note that the presence of weakening bindings is one difference to the first-order case (cf. [Uri92, Wei96a]), where term declarations are not assumed to be closed, and where subsort declarations are therefore included in (almost) elementary signatures.

---

[45] EdNote: what about the non-rigid ones?

[46] EdNote: any more cases here?

[47] EdNote: we cannot do this, since the K^i are not at the heads, must delay or argue with the inductive hypothesis. Maybe do something about the termination order directly

[48] EdNote: say something about the case where $Y \in \mathbf{Dom}(\Gamma)$ and $\Gamma(Y) = \mathbb{H}$. In both cases $m = l + \mathbf{ln}(\tau(\mathbb{H})) - \mathbf{ln}(\tau(\mathbb{A})) \geq \mathbf{ln}(\mathbb{H})$.

refine

Corollary 9.3.11 (Termination of elementary $\mathcal{U}$Pat) *There are no infinite $\mathcal{U}$Pat derivations.*

Corollary 9.3.12 *Sorted pattern unification is decidable and finitary for elementary signatures.*

EdNote(49) [49]

Now we will extend the decidability result above to a more general case: that of linear pattern signatures. We will do this by constructing from a linear pattern signature an elementary one that has essentially the same well-sorted formulae, and thus an equivalent Σ-unification problem.

Definition 9.3.13 (Signature Refinement) Let Σ and Σ' be signatures and $\mathcal{S}(\Sigma') = \mathcal{S}(\Sigma) \cup \mathcal{S}'$, then we say that Σ' **refine**s Σ with new sorts, iff for all sorts $\mathbb{A} \in \mathcal{S}(\Sigma)$ and all variable contexts Γ with $\Im(\Gamma) \subseteq \mathcal{S}(\Sigma)$ we have

1. $wsf_{\mathbb{A}}(\Sigma;\Gamma) \subseteq wsf_{\mathbb{A}}(\Sigma';\Gamma)$ and
2. for all $\mathbf{A} \in wsf_{\mathbb{A}}(\Sigma';\Gamma,\Gamma')$ where $\Im(\Gamma') \subseteq \mathcal{S}'$, there is a substitution σ with $\Gamma \vdash_{\Sigma'} \sigma{::}\Gamma'$ such that $\sigma(\mathbf{A}) \in wsf_{\mathbb{A}}(\Sigma,\Gamma)$.

EdNote(50) [50]

We will now convince ourselves that signature refinemets really have the properties claimed above.

Lemma 9.3.14 *If Σ' refines Σ with new sorts and $\Im(\Gamma) \subseteq \mathcal{S}(\Sigma)$, then $wsf_{\mathbb{A}}(\Sigma,\Gamma) = wsf_{\mathbb{A}}(\Sigma';\Gamma)$ for all $\mathbb{A} \in \mathcal{S}(\Sigma)$.*

Proof: As $\Im(\Gamma) \subseteq \mathcal{S}(\Sigma)$, we have $\Gamma' = \emptyset$ in 9.3.13(2) above. So σ can only be the empty substitution and for all $\mathbf{A} \in wsf_{\mathbb{A}}(\Sigma';\Gamma)$ we have $\sigma(\mathbf{A}) = \mathbf{A} \in wsf_{\mathbb{A}}(\Sigma,\Gamma)$. Thus we have $wsf_{\mathbb{A}}(\Sigma';\Gamma) \subseteq wsf_{\mathbb{A}}(\Sigma,\Gamma)$ for all $\mathbb{A} \in \mathcal{S}(\Sigma)$, which gives the assertion together with 9.3.13(1). □

Lemma 9.3.15 *If Σ' refines Σ with new sorts, and $\mathcal{E}$ is a Σ-unification problem, then $\mathcal{E}$ is Σ-unifiable, iff it is Σ'-unifiable.*

Proof: Let $\mathcal{E} := \exists\Gamma.\forall\Upsilon.\mathcal{F}$ and θ be a Σ-unifier of $\mathcal{E}$, then we have $\Delta \vdash_{\Sigma} \theta{::}\Gamma$ for some variable context Δ with $\Im(\Delta) \subseteq \mathcal{S}(\Sigma)$ and thus $\theta(X) \in wsf_{\Gamma(X)}(\Sigma;\Delta) = wsf_{\Gamma(X)}(\Sigma';\Delta)$ by Lemma 9.3.14, since $\Gamma(X) \in \mathcal{S}(\Sigma)$. Thus θ is also a Σ'-unifier of $\mathcal{E}$.

For the converse direction let θ be a Σ'-unifier of $\mathcal{E}$ and $X \in \mathbf{supp}(\theta)$. Furthermore, let $\mathbf{A} := \theta(X)$ and $\mathbb{A} := \Gamma(X)$, then we have $\mathbf{A} \in$

[49] EdNote: do some example here. Maybe some variant of 8.4.12. There the upat-diff rule will generate an intersection sort, and we have the problem of empty sorts,

[50] EdNote: need some kind of uniformity condition here, so that we can do the union on these.

$\mathit{wsf}_{\mathbb{A}}(\Sigma';\Delta,\Delta')$ for some variable contexts Δ and Δ' with $\Im(\Delta) \subseteq \mathcal{S}(\Sigma)$ and $\Im(\Delta') \subseteq \mathcal{S}'$. Now let σ_X be the substitution guaranteed by 9.3.13(2) above (we have $\Delta \vdash_{\Sigma'} \sigma_X{::}\Delta'$ and $\sigma_X(\theta(X)) \in \mathit{wsf}_{\mathbb{A}}(\Sigma,\Delta)$.) Thus $\sigma \circ \theta$ is a Σ-unifier of $\mathcal{E}$, if we set $\sigma := \bigcup_{X \in \mathbf{supp}(\theta)} \sigma_X$.[51] $\square$ EdNote(51)

Lemma 9.3.16 *For any linear pattern signature Σ there is an elementary pattern signature Σ' that refines Σ with new sorts.*

Proof: We prove the assertion by a Noetherian induction on the derivation $\mathcal{C}{:}\vdash_{sig} \Sigma$. Without loss of generality we can assume that $\mathcal{C}$ only contains semi-structural $\Sigma\Lambda^{\to}$-derivations in *sig*:*td* steps (by the structure theorem), moreover we assume that these are of minimal depth among semi-structural derivations. Now let $\mu(\mathcal{C})$ be the multiset of depths of these derivations.

For the base case of our induction over $\mu(\mathcal{C})$ we note that if $\mathcal{C}$ only consists of *sig*:*empty* and *sig*:*const*$'$ steps, then Σ is elementary by definition. So we only have to concern ourselves with the term declaration cases. So let $\mathcal{C}$ end in *sig*:*td*:*nf*, then we are in the following situation:

$$\dfrac{\dfrac{\mathcal{A}}{\vdash_\Sigma \mathbf{A}{::}\overline{\mathbb{C}^l} \to \mathbb{A}} \qquad \tau(\mathbb{A}) = \tau(\mathbb{B}) \in \mathcal{BT}}{\vdash_{sig} \Sigma, [\mathbf{A}{::}\overline{\mathbb{C}^l} \to \mathbb{B}]}\ \mathit{sig{:}td{:}nf}$$

Note that $[\mathbf{A}{::}\overline{\mathbb{C}^l} \to \mathbb{A}]$ is non-elementary, iff $[\mathbf{A}{::}\overline{\mathbb{C}^l} \to \mathbb{B}]$ is, since $\mathbb{A}$ and $\mathbb{B}$ are of base type. So let us assume that $[\mathbf{A}{::}\overline{\mathbb{C}^l} \to \mathbb{A}]$ isn't, since otherwise we can obtain the assertion by a simple appeal to the inductive hypothesis for Σ. We have assumed that $\mathcal{A}$ is semi-structural, so it has the form

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{H}}{\Gamma,\Xi^l \vdash_\Sigma \mathbf{H}{::}\mathbb{H}} \quad \dfrac{\mathcal{D}^i}{\Gamma,\Xi^l \vdash_\Sigma \mathbf{D}^i{::}\mathfrak{d}^i(\mathbb{H})}\ i\le m}{\Gamma,\Xi^l\vdash_\Sigma \mathbf{H}\overline{\mathbf{D}^m}{::}\mathbb{A}}\ \mathit{ws{:}app}^m}{\Gamma\vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m}{::}\overline{\mathbb{C}^l}\to\mathbb{A}}\ \mathit{ws{:}abs}^l \qquad \Gamma\vdash_\Sigma \lambda\overline{X^l}.\mathbf{H}\overline{\mathbf{D}^m} =_{\beta\eta}\mathbf{A}}{\Gamma\vdash_\Sigma \mathbf{A}{::}\overline{\mathbb{C}^l}\to\mathbb{A}}\ \mathit{ws{:}\beta\eta}$$

We will abbreviate $\mathfrak{d}^i(\mathbb{H})$ with $\mathbb{D}_i$ and set $l_i := \mathbf{ln}(\mathbb{D}^i)$ for the construction. Note that $\mathbb{D}^i = \overline{\mathfrak{d}^{l_i}(\mathbb{D}_i)} \to \mathfrak{r}^{l_i}(\mathbb{D}_i)$ where $\mathfrak{r}^{l_i}(\mathbb{D}_i) \in \mathcal{BS}$. Now let $\mathbb{E}_i := \overline{\mathfrak{d}^{l_i}(\mathbb{D}_i)} \to \mathbb{F}^i$, where the $\mathbb{F}^i$ are new base sort symbols with $\mathbb{F}_i \parallel \mathfrak{r}^{l_i}(\mathbb{D}_i)$

[51] EdNote: This union need not be defined, we probably have to extend condition 9.3.13(2) to a formulation with substitutions, to get a simultaneous σ in one go.

for $1 \leq i \leq m$. Furthermore let $\widetilde{\mathbb{H}} := \overline{\mathbb{E}^m} \to \mathfrak{r}^m(\mathbb{H})$ and $\widetilde{\mathbf{H}} := \lambda Z^m_{\mathbb{E}_m}.\mathbf{H}\overline{Z^m}$, and finally

$$\widetilde{\Sigma} := \Sigma, [\widetilde{\mathbf{H}}{::}\widetilde{\mathbb{H}}], [\mathbb{E}_i \leq \mathfrak{r}^{l_i}(\mathbb{D}_i)], [\lambda X^l.\mathcal{D}^i{::}\overline{\mathbb{C}^l} \to \mathbb{E}_i]$$

EdNote(52) EdNote(53) EdNote(54)

[52][53] Note that we have $\vdash_{sig} \widetilde{\Sigma}$, since we have chosen the sorts compatibly[54]. Thus we have to show that $\widetilde{\Sigma}$ refines $\Sigma, [\mathbf{A}{::}\overline{\mathbb{D}_m} \to \mathbb{A}]$ with new sorts, we will show the conditions in Definition 9.3.13. The first condition is trivial, since $\widetilde{\Sigma}$ extends Σ. For the second condition let Γ be a variable context with $\Im(\Gamma) \subseteq \mathcal{S}(\Sigma)$ and Γ' one with $\Im(\Gamma') \subseteq \mathcal{S}' := \mathcal{S}(\Sigma') \setminus \mathcal{S}(\Sigma)$. By the structure theorem any $\mathbf{A} \in wsf_{\mathbb{A}}(\Sigma'; \Gamma, \Gamma')$ has a a semi-structural Σ'-derivation. As $\mathbb{A} \in \mathcal{S}(\Sigma)$ and $\Im(\Gamma) \subseteq \mathcal{S}(\Sigma)$, the only interesting case is where the principal declaration is one of the subsort declarations $[\mathbb{E}_i \leq \mathfrak{r}^{l_i}(\mathbb{D}_i)]$, so $\Gamma \vdash_\Sigma \lambda\overline{X^l}.(\lambda X.X)\overline{\mathbf{E}^m} =_{\beta\eta} \mathbf{A}$ and the derivation has the following structural core:

$$\dfrac{\dfrac{\dfrac{}{\vdash_\Sigma \lambda X.X{::}\mathbb{E}_i \to \mathfrak{r}^{l_i}(\mathbb{D}_i)} \quad \dfrac{\mathcal{E}}{\Gamma, \Xi^l \vdash_\Sigma \mathbf{E}{::}\mathbb{E}^i} \quad \dfrac{\mathcal{E}^i}{\Gamma, \Xi^l \vdash_\Sigma \mathbf{E}^j{::}\mathfrak{d}^j(\mathfrak{r}^{l_i}(\mathbb{D}_i))}\, j \leq m-1}{\Gamma, \Xi^l \vdash_\Sigma (\lambda X.X)\overline{\mathbf{E}^m}{::}\mathfrak{r}^l(\mathbb{A})}\, ws{:}app^m}{\Gamma \vdash_\Sigma \lambda\overline{X^l}.(\lambda X.X)\overline{\mathbf{E}^m}{::}\mathbb{A}}\, ws{:}abs^l$$

Now, the only way to derive $\Gamma, \Xi^l \vdash_\Sigma \mathbf{E}{::}\mathbb{E}^i$ is via the principal declaration $[\lambda X^l.\mathcal{D}^i{::}\overline{\mathbb{C}^l} \to \mathbb{E}_i]$ (note that $\Im(\Gamma) \subseteq \mathcal{S}(\Sigma)$). As $\mathbb{E}^i$ is a base sort, $\mathcal{E}$ must have the form

$$\dfrac{\dfrac{\dfrac{\mathcal{F}}{\vdash_\Sigma \lambda X^l.\mathbf{D}^i{::}\overline{\mathbb{C}^l} \to \mathbb{E}^i} \quad \dfrac{\mathcal{F}_j}{\Gamma, \Xi^l \vdash_\Sigma \mathbf{F}^j{::}\mathbb{C}^j}\, j \leq l}{\Gamma, \Xi^l \vdash_\Sigma (\lambda X^l.\mathbf{D}^i)\overline{\mathbf{F}^l}{::}\mathbb{E}^i}\, *}{\Gamma, \Xi^l \vdash_\Sigma \mathbf{E}{::}\mathbb{E}^i}$$

Let us now put the pieces we have identified together to a Σ-derivation for $\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$. If we continue $\mathcal{D}^i$ from the construction above with l *ws:abs* steps we arrive at a Σ-derivation $\mathcal{F}'{:}\,\Gamma \vdash_\Sigma \lambda X^l.\mathbf{D}^i{::}\overline{\mathbb{C}^l} \to \mathbb{D}^i$, which we can plug into $\mathcal{E}$ to get a Σ-derivation $\mathcal{E}'{:}\,\Gamma, \Xi^l \vdash_\Sigma \mathbf{E}{::}\mathbb{D}_i$. This can be combined with the derivations $\mathcal{E}^i$ to obtain XXXXXXXXXXX

Furthermore, we have $\mu(\widetilde{\Sigma}) < \mu(\Sigma, [\mathbf{A}{::}\overline{\mathbb{D}_m} \to \mathbb{A}])$, so we can apply the inductive hypothesis, to obtain the assertion. □

[52]EdNote: In the draft we assumed that $[\mathbf{H}{::}\mathbb{H}]$ was elementary and that $[\widetilde{\mathbf{H}}{::}\widetilde{\mathbb{H}}]$ is as well, since $\mathbf{ln}(\mathbb{H}) = \mathbf{ln}(\widetilde{\mathbb{H}}) = m$ we have to make some changes in the induction to be able to assume this.

[53]EdNote: are the $\lambda X^l.\mathcal{D}^i$ closed? do we need to abstract more?

[54]EdNote: expand this

Theorem 9.3.17 *Let Σ be a linear pattern signature, then sorted pattern unification in Σ is decidable.*

empty sort
sort!empty
witness formula
formula!witness

Proof: We can apply Lemma 9.3.15: To decide Σ-unification it is sufficient to decide Σ'-unification, which is possible by Theorem 9.3.12. □ EndNP(42)

9.4 Empty sorts

The unification algorithms presented in the last sections do not alone suffice for sorted variants of refutation calculi such as those presented in part **??** of this book. In $\Sigma\Lambda^{\rightarrow}$, we have not made the assumption that all sorts are nonempty, i.e. that for every sort $\mathbb{A} \in \mathcal{S}$ there is at least one closed formula $\mathbf{A}$ with $\vdash_{\Sigma} \mathbf{A}{::}\mathbb{A}$, which implies that the set $\mathcal{D}_{\mathbb{A}}$ is non-empty (cf. 1.4.14).

Definition 9.4.1 (Empty Sorts)
We call a sort $\mathbb{A}$ **empty** with respect to a signature Σ, iff $\mathit{wsf}_{\mathbb{A}}(\Sigma;\emptyset) = \emptyset$ and a closed formula $\mathbf{A} \in \mathit{wsf}_{\mathbb{A}}(\Sigma;\emptyset)$ a **witness formula** for $\mathbb{A}$.

As we have already seen in the typed case, empty sorts problematic for refutation calculi, since if $\mathcal{D}_{\mathbb{A}}$ is empty, the formula $\mathbf{A} := \forall X_{\alpha}.(pX) \wedge \neg(pX)$ satisfiable, since vacuous universal quantifications are true, but a naive resolution calculus would be able to obtain the empty clause from $\mathbf{CNF}(\mathbf{A}) = \{(pY_{\mathbb{A}})^{\mathtt{T}}, (pZ_{\mathbb{A}})^{\mathtt{F}}\}$ by resolving upon the two unit clauses with the unifier $[Y_{\mathbb{A}}/Z_{\mathbb{A}}]$. Therefore, most first-order sorted logics restrict their attention to the case, where a closed formula for each sort. In the presence of intersection sorts, this is not viable, since this would mean that we cannot have disjoint sorts $\mathbb{E}$ and $\mathbb{O}$ (say for the even and odd natural numbers), since $\mathbb{E} \sqcap \mathbb{O}$ would have to be non-empty. Thus we will now develop an algorithm that tests for emptiness of sorts. It is easy to see that in the presence of intersection sorts, the emptiness of sorts is undecidable (it is already undecidable for the first-order case [SS89]), so the algorithm can only be a semi-decision procedure. BegNP(55)

Lemma 9.4.2 *For each signature Σ and each Σ-unification problem $\mathcal{E}$, there is a signature Σ' and a sort $\mathbb{A}(\mathcal{E})$, such that $\mathcal{E}$ is Σ-unifiable, iff $\mathbb{A}(\mathcal{E})$ is non-empty in Σ'.*

Proof: We will first do the construction for $\Sigma^d\Lambda^{\rightarrow}$, since this is simpler, and adapt it to $\Sigma^r\Lambda^{\rightarrow}$ later. So let $\exists\Gamma.\forall\Upsilon.\mathbf{A} =^? \mathbf{B}$ be a unification problem with $\Gamma, \Upsilon \vdash_{\Sigma} \mathbf{A}{::}\mathbb{A}$ and $\Gamma, \Upsilon \vdash_{\Sigma} \mathbf{B}{::}\mathbb{B}$ and $\Gamma = [X^1{::}\mathbb{E}_1], \ldots, [X^n{::}\mathbb{E}_n]$ (we will only consider unification problems of length 1 since the construction for larger problems is analogous). Now let h be a new constant symbol and $\mathbb{A}', \mathbb{B}'$, and

[55] New Part:

$\mathbb{D}$ be new base sorts of base type α, and

$$\Sigma' := \Sigma \cup \{[h{::}\mathbb{A} \to \mathbb{D}], [h{::}\mathbb{B} \to \mathbb{D}], [\lambda\overline{X^n}\textbf{.}h\mathbf{A}{::}\overline{\mathbb{E}_n} \to \mathbb{A}'], [\lambda\overline{X^n}\textbf{.}h\mathbf{B}{::}\overline{\mathbb{E}_n} \to \mathbb{B}']\}$$

Clearly Σ' is a valid signature, since $\mathbb{A}', \mathbb{B}', \mathbb{D}$ are of base type and therefore $\mathbb{A} \parallel \mathbb{B}$ and $\mathbb{B} \parallel \mathbb{D}$.

If there is a Σ-unifier θ of $\exists\Gamma\textbf{.}\forall\Upsilon\textbf{.}\mathbf{A} =^? \mathbf{B}$, then $\Delta \vdash_\Sigma \theta{::}\Gamma$ and we have $\mathcal{A}^i{:}\,\theta(X^i){::}\Gamma(X^i) = \mathbb{E}^i$, thus we have

$$\cfrac{\cfrac{\cfrac{}{\Delta,\Upsilon \vdash_{\Sigma'} (\lambda\overline{X^n}\textbf{.}h\mathbf{A}){::}\overline{\mathbb{E}^n} \to \mathbb{A}'}\,\textit{ws:td} \qquad \cfrac{\mathcal{A}^i}{\Delta,\Upsilon \vdash_{\Sigma'} \theta(X^i){::}\mathbb{E}^i}}{\Delta,\Upsilon \vdash_{\Sigma'} (\lambda\overline{X^n}\textbf{.}h\mathbf{A})\overline{\theta(X^n)}{::}\mathbb{A}' \quad *}\,\textit{ws:app}^n}{\Delta,\Upsilon \vdash_{\Sigma'} h\theta(\mathbf{A}){::}\mathbb{A}'}\,\textit{ws:}\beta\eta$$

and $\Delta,\Upsilon \vdash_{\Sigma'} h\theta(\mathbf{B}){::}\mathbb{B}'$ by the same construction (The $\Sigma\Lambda^{\to\sqcap}$-derivations $\mathcal{A}^i$ come from the derivation of well-sortedness of θ). Furthermore we have $\Delta,\Upsilon \vdash_{\Sigma'} h\theta(\mathbf{A}){::}\mathbb{B}'$ by 7.6.1, since $\Delta,\Upsilon \vdash_\Sigma \theta(\mathbf{A}){=_{\beta\eta}}\theta(\mathbf{B})$, and therefore
EdNote(56) $\Delta,\Upsilon \vdash_{\Sigma'} h\theta(\mathbf{A}){=_{\beta\eta}}h\theta(\mathbf{B})$ by *tr:app*. In particular, $\Delta,\Upsilon \vdash_{\Sigma'} h\theta(\mathbf{A}){::}\mathbb{A}'{\sqcap}\mathbb{B}'$.[56] If $\mathbf{A}$ and $\mathbf{B}$ are not unifiable, then $\mathbb{A}' \sqcap \mathbb{B}'$ is empty, since $\mathbb{A}'$ and $\mathbb{B}'$ are new to Σ and thus the only way, we can arrive at a formula of sort $\mathbb{A}' \sqcap \mathbb{B}'$ is via the last two term declarations in Σ'.

Thus $\mathbb{A}' \sqcap \mathbb{B}'$ is non-empty, iff $\mathbf{A}$ and $\mathbf{B}$ are Σ-unifiable. Since $\mathbf{A}$ and $\mathbf{B}$
EdNote(57) were arbitrary, we have proven the assertion.[57]

Note that the construction only works for $\Sigma^d\Lambda^{\to}$, since $\mathbb{A} \to \mathbb{D} \nparallel \mathbb{B} \to \mathbb{D}$ in $\Sigma^r\Lambda^{\to}$. So we will use term declarations

$$[h{::}\overline{\mathfrak{d}^n(\mathbb{A})} \to \mathbb{K}], [\mathbb{K} \le \mathfrak{r}^n(\mathbb{A})], [\mathbb{K} \le \mathfrak{r}^n(\mathbb{B})], [\lambda X_{\mathbb{A}}\textbf{.}hX{::}\mathbb{A} \to \mathbb{D}], [\lambda X_{\mathbb{B}}\textbf{.}hX{::}\mathbb{B} \to \mathbb{D}]$$

instead of the declarations for h in Σ'. To see that Σ' is well-sorted in $\Sigma^r\Lambda^{\to}$, note that $\mathbb{A} \parallel \mathbb{B}$ by Lemma 8.4.1 so $\overline{\mathfrak{d}^n(\mathbb{A})} \to \mathbb{K}$ is a joint supersort of $\mathbb{A}$ and $\mathbb{B}$ by construction, thus the restriction declarations for h lead to a well-sorted signature Σ'. The rest of the construction goes through as above. □

Corollary 9.4.3 *In $\Sigma\Lambda^{\to\sqcap}$, it is undecidable whether a sort $\mathbb{A}$ is empty with respect to an arbitrary signature Σ.*

Proof: We prove the assertion by reducing the Σ-unification problem to the Σ'-emptyness problem, where Σ' is the signature guaranteed by Lemma 9.4.2. Thus if the latter were decidable for arbitrary signatures, Σ-unification would be, contradicting the undecidability results from [Gol81, Far91a, SS89]. □

---

[56] EdNote: This assumes that $\Delta = \emptyset$, since otherwise the non-emptyness does not hold, look what Schmidt-Schauss and Weidenbach do.

[57] EdNote: And the the way around?

$\mathcal{I}_0$ *intuitionistic propositional logic*
intuitionistic propositional logic
Curry/Howard-isomorphism
Curry
Howard
pure
formula pure
EndNP(55)

Corollary 9.4.4 *It is undecidable whether a given signature* Σ *for* $\Sigma\Lambda^{\to\sqcap}$ *is regular.*

Proof: We can use the same construction as in Lemma 9.4.2; since $\Delta, \Upsilon \vdash_\Sigma h\theta(\mathbf{A}){::}\mathbb{A}'$ and $\Delta, \Upsilon \vdash_\Sigma h\theta(\mathbf{A}){::}\mathbb{B}'$, and $\mathbb{A}'$ and $\mathbb{B}'$ are incomparable, Σ' is regular, iff $\mathbf{A}$ and $\mathbf{B}$ are unifiable. $\square$

Remark 9.4.5 Note that the previous undecidability results are independent of the decidability of the sort computation problem.

We will base our treatment of empty sorts in this section on the observation above that emptiness of intersection sorts is equivalent to sorted unification (see 9.4.3) and on the well-known Curry/Howard Isomorphism for $\Lambda^\to$ that links non-emptiness of types to provability in intuitionistic propositional logic.

Definition 9.4.6 (Intuitionistic Propositional Logic)
Let $\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots \in \mathit{wff}_o([\Rightarrow{:}\, o \to o \to o]; [A{:}\, o], [B{:}\, o], \ldots)$ (i.e. they are nested implications of propositional variables), then the inference system $\mathcal{I}_0$ for intuitionistic propositional logic consists of the inference rules in Figure 9.5. We say that $\mathbf{A}$ is intuitionistically valid, iff $\Vdash_{\mathcal{I}_0} \mathbf{A}$ is derivable in $\mathcal{I}_0$.

$$\frac{}{\mathcal{H}, \mathbf{A} \Vdash_{\mathcal{I}_0} \mathbf{A}}\,\mathcal{I}_0(ax)$$

$$\frac{\mathcal{H} \Vdash_{\mathcal{I}_0} \mathbf{A} \Rightarrow \mathbf{B} \quad \mathcal{K} \Vdash_{\mathcal{I}_0} \mathbf{A}}{\mathcal{H} \Vdash_{\mathcal{I}_0} \mathbf{B}}\,\mathcal{I}_0(\Rightarrow E) \qquad \frac{\mathcal{H}, \mathbf{A} \Vdash_{\mathcal{I}_0} \mathbf{B}}{\mathcal{H} \Vdash_{\mathcal{I}_0} \mathbf{A} \Rightarrow \mathbf{B}}\,\mathcal{I}_0(\Rightarrow I)$$

Figure 9.5: Minimal Intuitionistic Propositional Logic

The Curry/Howard-isomorphism is based on the observation, that these rules are very similar to the type inference rules for $\Lambda^\to$ (cf. section 1.5), if we equate a formula $\mathbf{A} \Rightarrow \mathbf{B}$ with a type $\alpha \to \beta$. We will now briefly explore this idea as far as it helps give an understanding to the methods needed here. For a thorough development see [HS86, Tho91b, GLT89].

Let us write $\tau(\mathbf{A} \Rightarrow \mathbf{B}) = \tau(\mathbf{A}) \to \tau(\mathbf{B})$ and for propositional variables $\tau(A) = \alpha$, $\tau(B) = \beta, \ldots$. In particular, the rules $\mathcal{I}_0(ax)$, $\mathcal{I}_0(\Rightarrow E)$, and $\mathcal{I}_0(\Rightarrow I)$ can be obtained from the rules *wff*:*var*, *wff*:*app*, and *wff*:*abs* by simply deleting the formula parts (obtaining a set $\mathcal{H}$ of propositions corresponding to $\mathbf{Dom}(\Gamma)$ from the context Γ). Since we do not have a rule corresponding to *wff*:*const*, let us for the moment only explore the isomorphism for so-called **pure** formulae, i.e. where the signature Σ is empty. Since the type inference system for $\Lambda^\to$ is structural, the structure of formulae corresponds to the structure of the type-derivation. Therefore, a $\Lambda^\to$-formula

$\mathbf{M} \in \mathit{wff}_\alpha(\emptyset;\Gamma)$ can be seen as a representation of $\mathcal{I}_0$-derivation, of $\tau^{-1}(\alpha)$ from the hypotheses $\tau^{-1}(\mathbf{Dom}(\Gamma))$. In particular, the implication introduction rule $\mathcal{I}_0(\Rightarrow I)$ suggests that a proof of $\mathbf{A} \Rightarrow \mathbf{B}$ can be seen as a function (after all, it corresponds to a λ-abstraction), that transforms proofs of $\mathbf{A}$ into proofs of $\mathbf{B}$.

Example 9.4.7 In particular, the identity function $\lambda X_\alpha. X$ is of type $\alpha \to \alpha$ and corresponds to the proof of the propositional implication $A \Rightarrow A$.

$$\dfrac{\dfrac{}{[X\colon\alpha] \vdash_\emptyset X\colon\alpha}\,\mathit{wff}{:}\mathit{var}}{\vdash_\emptyset \lambda X_\alpha. X\colon\alpha \to \alpha}\,\mathit{wff}{:}\mathit{abs} \qquad \dfrac{\dfrac{}{A \Vdash_{\mathcal{I}_0} A}\,\mathcal{I}_0(ax)}{\Vdash_{\mathcal{I}_0} A \Rightarrow A}\,\mathcal{I}_0(\Rightarrow I)$$

The consequence that is interesting for our application, is that in the pure simply typed λ-calculus, a type α is non-empty iff $\tau^{-1}(\alpha)$ is intuitionistically valid. Now we will extend this idea to our sorted setting, where it will give us a decidability result for emptiness of simple sorts. We will account for non-empty signatures and functional base sorts by introducing additional hypotheses. To make the exposition more legible, we will drop the isomorphism τ (and τ^{-1}) for sorts. Whether a formula $\mathbb{A} \to \mathbb{B}$ is meant as a sort or a propositional formula $\mathbf{A}_o \Rightarrow \mathbf{B}_o$ can always be determined from the context.

Definition 9.4.8 For a given sort system $\mathcal{S}$, a signature Σ, and a variable context Γ let us consider a formulation of $\mathcal{I}_0$, where the set of propositional variable is $\mathcal{BS}$. Furthermore let

$$\begin{array}{rcl}
\mathcal{H}_\Sigma(\Gamma) & := & \mathcal{H}_\mathcal{S} \cup \mathcal{H}_\Gamma \cup \mathcal{H}_\Sigma \\
\mathcal{H}_\mathcal{S} & := & \{\mathbb{C} \to \mathfrak{d}(\mathbb{C}) \to \mathfrak{r}(\mathbb{C}) \mid \mathbb{C} \in \mathcal{BS}^f\} \\
\mathcal{H}_\Sigma & := & \{\mathbb{A} \mid [\mathbf{A}{::}\mathbb{A}] \in \Sigma\} \\
\mathcal{H}_\Gamma & := & \{\mathbb{A} \mid [X{::}\mathbb{A}] \in \Gamma\}
\end{array}$$

Note that we again assume term declarations in Σ to be closed.

Lemma 9.4.9 *If $\mathbb{A}$ is a simple sort, then $\mathit{wsf}_\mathbb{A}(\Sigma,\Gamma) \neq \emptyset$, iff $\mathcal{H}_\Sigma(\Gamma) \Vdash_{\mathcal{I}_0} \mathbb{A}$.*

Proof: We prove the assertion extending the Curry/Howard isomorphism to $\Sigma\Lambda^\to$. For the τ-direction let $\mathbf{A} \in \mathit{wsf}_\mathbb{A}(\Sigma,\emptyset)$ be a witness for non-emptyness of $\mathbb{A}$. We construct τ by induction over the derivation $\mathcal{A}\colon\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}$. The *ws:var* and *ws:td* cases correspond to $\mathcal{I}_0(ax)$, acting on the appropriate hypothesis from $\mathcal{H}_\Gamma$ (for *ws:var*) or $\mathcal{H}_\Sigma$ (for *ws:td*). The interesting case is that of *ws:app*, where $\mathbf{A} \equiv \mathbf{BC}$ and we have the following situation.

$$\dfrac{\dfrac{\mathcal{B}}{\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{B}} \quad \dfrac{\mathcal{C}}{\Gamma \vdash_\Sigma \mathbf{C}{::}\mathfrak{d}(\mathbb{B})}}{\Gamma \vdash_\Sigma \mathbf{BC}{::}\mathfrak{r}(\mathbb{B})}\,\mathit{ws}{:}\mathit{app}$$

By inductive hypothesis we have derivations $\mathcal{B}'{:}\,\mathcal{H}_\Sigma(\Gamma) \Vdash_{\mathcal{I}_0} \mathbb{B}$ and $\mathcal{C}'{:}\,\mathcal{H}_\Sigma(\Gamma) \Vdash_{\mathcal{I}_0} \mathfrak{d}(\mathbb{B})$. If $\mathbb{B} \in \mathcal{BS}^f$, then $\mathbb{B} \to \mathfrak{d}(\mathbb{A}) \to \mathfrak{r}(\mathbb{A}) \in \mathcal{H}_\Sigma \subseteq \mathcal{H}_\Sigma(\Gamma)$, and $\mathbb{A} = \mathfrak{r}(\mathbb{B})$, and we have the following $\mathcal{I}_0$-derivation. *decidable*

$$\dfrac{\dfrac{\dfrac{}{\mathcal{H}_\Sigma(\Gamma) \Vdash_{\mathcal{I}_0} \mathbb{B} \to \mathfrak{d}(\mathbb{B}) \to \mathfrak{r}(\mathbb{B})} \quad \dfrac{\mathcal{B}'}{\mathcal{H}_\Sigma(\Gamma) \Vdash_{\mathcal{I}_0} \mathbb{B}}}{\mathcal{H}_\Sigma(\Gamma) \Vdash_{\mathcal{I}_0} \mathfrak{d}(\mathbb{B}) \to \mathfrak{r}(\mathbb{B})} \qquad \dfrac{\mathcal{C}'}{\mathcal{H}_\Sigma(\Gamma) \Vdash_{\mathcal{I}_0} \mathfrak{d}(\mathbb{B})}}{\mathcal{H}_\Sigma(\Gamma) \Vdash_{\mathcal{I}_0} \mathfrak{r}(\mathbb{B})}$$

If $\mathbb{B}$ is already of the form $\mathbb{C} \to \mathbb{A}$, then we can directly use $\mathcal{I}_0(\Rightarrow E)$ to combine $\mathcal{B}'$ and $\mathcal{C}'$. The *ws:abs* case directly corresponds to $\mathcal{I}_0(\Rightarrow I)$, and *ws:*$\beta\eta$ is a direct consequence of the inductive hypothesis, since we have only asserted the non-emptiness of $\mathbb{A}$.

$$\dfrac{\dfrac{\mathcal{A}'}{\Gamma \vdash_\Sigma \mathbf{B}{::}\mathbb{A}}\,ws{:}abs \qquad \Gamma \vdash_\Sigma \mathbf{A}{=}_{\beta\eta}\mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{A}}\,ws{:}\beta\eta$$

In particular, the actual formula $\mathbf{B}$ instead of $\mathbf{A}$ is irrelevant.

For τ^{-1} we proceed by induction on the derivation $\mathcal{D}{:}\,\mathcal{H}_\Sigma(\Gamma) \Vdash_{\mathcal{I}_0} \mathbb{A}$. The only case that is interesting is that of $\mathcal{I}_0(ax)$ for $\mathcal{H}_\mathcal{S}(\Sigma)$ (we have the direct correspondence of the other rules to $\Sigma\Lambda^\to$ rules above). However, in this case $\mathcal{I}_0(ax)$ corresponds to an application of *ws:weaken* with the admissible subsort judgment $\mathbb{A} \leq \mathfrak{d}(\mathbb{A}) \to \mathfrak{r}(\mathbb{A})$. □

Corollary 9.4.10 *The emptiness problem is decidable in* $\Sigma\Lambda^\to$.

Proof: It is well-known [Kle52, Sza69, Dyc92], that intuitionistic propositional logic is decidable, just like its classical counterpart. □

Up to now, we have only considered the implicational fragment of intuitionistic propositional logic, which corresponds to simple sorts. Intersection sorts correspond to intuitionistic propositional logic with implications and conjunctions. For this $\mathcal{I}_0$ has to be augmented with inference rules for conjunctions

$$\dfrac{\mathcal{H} \Vdash_{\mathcal{I}_0} \mathbf{A} \wedge \mathbf{B}}{\mathcal{H} \Vdash_{\mathcal{I}_0} \mathbf{A} \quad \mathcal{H} \Vdash_{\mathcal{I}_0} \mathbf{B}}\,\mathcal{I}_0(\wedge E) \qquad \dfrac{\mathcal{H} \Vdash_{\mathcal{I}_0} \mathbf{A} \quad \mathcal{H} \Vdash_{\mathcal{I}_0} \mathbf{B}}{\mathcal{H} \Vdash_{\mathcal{I}_0} \mathbf{A} \wedge \mathbf{B}}\,\mathcal{I}_0(\wedge I)$$

However, for a type, the Curry/Howard Isomorphism for sorts breaks down, since the $\mathcal{I}_0(\wedge I)$ rule allows two separate witnesses for the non-emptiness of a conjunction, whereas in the *ws:*$\sqcap$ rule, the formula in question has to have both sorts, i.e. the *ws:*$\sqcap$ rule calls for a single witness of the non-emptiness of a conjunction.

Lemma 9.4.11 *The set* $\mathit{wsf}_{\mathbb{A}}(\Sigma, \Gamma)$ *is nonempty for* $\mathbb{A} = \bigcap_{i=1}^{n} \mathbb{A}_i$*, iff*

1. *there are general bindings* $\mathbf{G}_i \in \mathcal{G}^h_{\mathbb{A}_i}(\Sigma, \Gamma)$ *for some head* h *and*

2. *there is a a unifier* $\theta \in \mathbf{wsU}(\Sigma, \exists\Gamma, \mathcal{C}.\mathbf{G}_1 =^? \ldots =^? \mathbf{G}_n)$ *where* $\mathcal{C} = \mathcal{C}(\mathbf{G}_1), \ldots, \mathcal{C}(\mathbf{G}_n)$*, and*

3. *all of the sorts of variables in* $\Im(\mathcal{C})$ *are non-empty.*

Proof: If $\mathbb{A}$ is non-empty, then there is a (closed) witness formula $\mathbf{A}$. We will prove the assertion by an induction on the structure of a semi-structural $\Sigma\Lambda^{\to\sqcap}$-derivation $\mathcal{A}: \vdash_\Sigma \mathbf{A}::\mathbb{A}$.

By the general binding theorem for $\Sigma\Lambda^{\to\sqcap}$ (cf. 9.1.8), there are general general bindings $\mathbf{G}_i$ and the corresponding substitution ρ is a unifier of $\mathcal{E}$ by construction. As $\mathbf{A}$ is closed, $\rho(X)$ must be as well for all $X \in \mathbf{Dom}(\mathcal{C})$, so we have witness formulae $\rho(X)$ for all sorts in $\Im(\mathcal{C})$

For the converse direction let $\mathcal{C} = [X^1::\mathbb{B}_1], \ldots, [X^n::\mathbb{B}_n]$ and $\mathbf{G}_i$ be general bindings of sort $\mathbb{A}_i$, and $\theta \in \mathbf{wsU}(\Sigma, \mathcal{E})$. If the sorts $\mathbb{B}_i$ in $\mathcal{C}$ are non-empty, then we can construct a Σ-substitution $\rho = \overline{[\mathbf{B}^n/X^n]}$ by selecting a closed formula $\mathbf{B}^n \in \mathit{wsf}_{\mathbb{B}_i}(\Sigma, \emptyset)$ for each $i \leq n$. Thus $\theta' := \rho\circ\theta \in \mathbf{wsU}(\Sigma, \mathcal{E})$ and $\theta'(\mathbf{G}_i)::\mathbb{A}_i$. Thus $\theta'(\mathbf{G}_1) \in \mathit{wsf}_{\mathbb{A}}(\Sigma; \emptyset)$ and $\mathbb{A}$ is non-empty. □

$$\frac{}{\mathcal{H}, A \Vdash_{\mathcal{DY}} A}\,\mathcal{DY}(ax) \qquad \frac{\mathcal{H}, A \Vdash_{\mathcal{DY}} B}{\mathcal{H} \Vdash_{\mathcal{DY}} A \Rightarrow B}\,\mathcal{DY}(\Rightarrow R)$$

$$\frac{\mathcal{H}, A, B \Vdash_{\mathcal{DY}} C \quad B \text{ atomic}}{\mathcal{H}, B \Rightarrow A, \mathbf{B} \Vdash_{\mathcal{DY}} C}\,\mathcal{DY}(\Rightarrow L^1)$$

$$\frac{C \Rightarrow D \Rightarrow A \Vdash_{\mathcal{DY}} E}{(C \wedge D) \Rightarrow A \Vdash_{\mathcal{DY}} E}\,\mathcal{DY}(\Rightarrow L^2)$$

$$\frac{\mathcal{H}, A \Rightarrow B \Vdash_{\mathcal{DY}} C \Rightarrow A \quad \mathcal{H}, B \Vdash_{\mathcal{DY}} E}{\mathcal{H}, (C \Rightarrow A) \Rightarrow B \Vdash_{\mathcal{DY}} E}\,\mathcal{DY}(\Rightarrow L^3)$$

$$\frac{\mathcal{H}, A, B \Vdash_{\mathcal{DY}} C}{\mathcal{H}, A \wedge B \Vdash_{\mathcal{DY}} C}\,\mathcal{DY}(\wedge L) \qquad \frac{\mathcal{H} \Vdash_{\mathcal{DY}} A \quad \mathcal{H} \Vdash_{\mathcal{DY}} B}{\mathcal{H} \Vdash_{\mathcal{DY}} A \wedge B}\,\mathcal{DY}(\wedge R)$$

Figure 9.6: Dyckhoff's Calculus for $\mathcal{I}_0$

This result suggests an inference rule for deriving the non-emptiness of sorts in the presence of intersection sorts, which reduces the problem to higher-order sorted unification. We will use this rule to extend Dyckhoff's algorithm from [Dyc92] to $\Sigma\Lambda^{\to\sqcap}$. Dyckhoff presents a decision procedure for intuitionistic propositional logic by replacing $\mathcal{I}_0$ by the equivalent (contraction-free) sequent calculus $\mathcal{DY}$, (see Figure 9.6) for which termination can be shown. BegNP(58)

Theorem 9.4.12 (Termination of $\mathcal{DY}$) *The calculus $\mathcal{DY}$ is terminating.*

Proof: Let the weight of a formula be the sum of the number of implications and twice the number of conjunctions occurring in it. Now let let the weight $\mu(\mathcal{H} \Vdash_{\mathcal{DY}} \mathbf{A})$ of a sequent be the multiset of the weights of the formulae in $\mathcal{H}$ together with that of $\mathbf{A}$. To see that the set of rules is terminating, we consider the multiset ordering $\prec_{mul}$ on natural numbers and verify that for all the rules in $\mathcal{DY}$ the sequents in the antecedent have a weight that is strictly smaller with respecti to $\prec_{mul}$ than those in the succedent. Thus $\mathcal{DY}$ is terminating, since $\prec_{mul}$ is noetherian. $\square$ EndNP(58)

Clearly, we can directly use all of the rules in Figure 9.6 except $\mathcal{DY}(\wedge R)$ with our version of the Curry/Howard Isomorphism and obtain a decision procedure for the emptiness problem in $\Sigma\Lambda^{\rightarrow}$, replicating the decidability result from Corollary9.4.10. BegNP(59)

For the full problem, we replace $\mathcal{DY}(\wedge R)$ with a rule $\emptyset(\sqcap)R$ which is a direct implementation of Lemma 9.4.11.

Definition 9.4.13 (Detection of Empty Sorts in $\Sigma\Lambda^{\rightarrow\sqcap}$) The Calculus $\Sigma\emptyset$ is the inference system in Figure 9.7.

$$\frac{}{\mathcal{H}, \mathbb{A} \Vdash_{\emptyset} \mathbb{A}}\,\emptyset(ax) \qquad \frac{\mathcal{H}, \mathbb{A} \Vdash_{\emptyset} \mathbb{B}}{\mathcal{H} \Vdash_{\emptyset} \mathbb{A} \rightarrow \mathbb{B}}\,\emptyset(\Rightarrow R)$$

$$\frac{\mathcal{H}, \mathbb{A}, \mathbb{B} \Vdash_{\emptyset} \mathbb{C} \quad \mathbb{B} \in \mathcal{BS}}{\mathcal{H}, \mathbb{B} \rightarrow \mathbb{A}, \mathbb{B} \Vdash_{\emptyset} \mathbb{C}}\,\emptyset(\Rightarrow L^{1}) \qquad \frac{\mathbb{C} \rightarrow \mathbb{D} \rightarrow \mathbb{A} \Vdash_{\emptyset} \mathbb{E}}{(\mathbb{C} \sqcap \mathbb{D}) \rightarrow \mathbb{A} \Vdash_{\emptyset} \mathbb{E}}\,\emptyset(\Rightarrow L^{2})$$

$$\frac{\mathcal{H}, \mathbb{A} \rightarrow \mathbb{B} \Vdash_{\emptyset} \mathbb{C} \rightarrow \mathbb{A} \quad \mathcal{H}, \mathbb{B} \Vdash_{\emptyset} \mathbb{E}}{\mathcal{H}, (\mathbb{C} \rightarrow \mathbb{A}) \rightarrow \mathbb{B} \Vdash_{\emptyset} \mathbb{E}}\,\emptyset(\Rightarrow L^{3}) \qquad \frac{\mathcal{H}, \mathbb{A}, \mathbb{B} \Vdash_{\emptyset} \mathbb{C}}{\mathcal{H}, \mathbb{A} \sqcap \mathbb{B} \Vdash_{\emptyset} \mathbb{C}}\,\emptyset(\sqcap)L$$

$$\frac{\mathcal{H} \Vdash_{\emptyset} \mathbb{C}_i \quad \begin{array}{l} \mathbf{G}^{\mathbb{A}} \in \mathcal{A}_{\mathbb{A}}(\Sigma;\emptyset), \mathbf{G}^{\mathbb{B}} \in \mathcal{A}_{\mathbb{B}}(\Sigma;\emptyset), \\ \theta \in \mathbf{wsU}(\Sigma, \exists\mathcal{C}(\mathbf{G}^{\mathbb{A}}), \mathcal{C}(\mathbf{G}^{\mathbb{B}}).\mathbf{G}^{\mathbb{A}} =^{?} \mathbf{G}^{\mathbb{B}}) \\ \mathbf{Intro}(\theta) = \{X^1_{\mathbb{C}_1}, \ldots, X^n_{\mathbb{C}_n}\} \end{array}}{\mathcal{H} \Vdash_{\emptyset} \mathbb{A} \sqcap \mathbb{B}}\,\emptyset(\sqcap)R$$

Figure 9.7: Detection of Empty Sorts

Note that the calculus $\Sigma\emptyset$ cannot be terminating in general, since it involves higher-order unification. However in fragments of $\Sigma\Lambda^{\rightarrow}$, where unification

[58]New Part:
[59]New Part:

is terminating and finitary we can pre-compute simultaneous unifiers for intersection sorts (see Remark 9.1.12) and obtain a decision procedure.

Theorem 9.4.14 (Deciding Emptyness)
EdNote(60) *If Σ is an elementary*[60]*, then the calculus $\emptyset$ decides whether a sort $\mathbb{A}$ is empty in Σ.*

EdNote(61) **Proof:** By Theorem 9.3.12, we know that[61] □
EndNP(59)

[60] EdNote: adapt, if we have the result for linear signatures
[61] EdNote: This is not so trivial, since we have to break cycles, and all that.

$\mathbb{O}$*sort of truth values*
individuals
truth values!sort of
sort!of truth values
sentence
proposition

Chapter 10

Sorted Higher-Order Logic and Automated Theorem Proving

In this chapter we will present a sorted formulation $\Sigma\mathcal{HOL}$ of higher-order logic by giving the sorted λ-calculus $\Sigma\Lambda^{\rightarrow}$ developed so far a logical interpretation. Just as in the typed case we specialize the types in $\Sigma\Lambda^{\rightarrow}$ by restricting to those of $\mathcal{HOL}$, i.e. $\mathcal{BT} = \{o, \iota\}$. Closely tied to the type o we use a sort $\mathbb{O}$ to be able to speak about truth values in our sort system. To complete the logical system $\Sigma\mathcal{HOL}$, we add compulsory logical constants and term declarations $[q^{\mathbb{A}}{::}\mathbb{A} \rightarrow \mathbb{A} \rightarrow \mathbb{O}]$ for equality, $[\neg{::}\mathbb{O} \rightarrow \mathbb{O}]$ and $[\wedge{::}\mathbb{O} \rightarrow \mathbb{O} \rightarrow \mathbb{O}]$ for the connectives, and, finally, $[\Pi^{\mathbb{A}}{::}(\mathbb{A} \rightarrow \mathbb{O}) \rightarrow \mathbb{O}]$ for quantification.

We will explore the effect of introducing sorts to the calculi presented in chapter **??** by using pre-Σ-unification instead of unsorted, higher-order pre-unification. We will first extend the tableau and resolution calculi from sections 5 and 6, and then generalize these methods to a setting with conditional term declarations. Instead of redoing the completeness proofs, which are almost exactly the same, exept for using the sorted results instead of the unsorted ones, we will discuss the practical differences of the sorted formulation in examples.

10.1 $\Sigma\mathcal{HOL}$, A sorted Higher-Order Logic

The sorted higher-order logic $\Sigma\mathcal{HOL}$ is a common generalization of $\mathcal{HOL}$ and $\Sigma\Lambda^{\rightarrow}$, so the set $\mathcal{BT}$ of base types is $\{o, \iota\}$. We do not need any other types, since we can model all other type distinctions in our sort system. $\Sigma\mathcal{HOL}$ has a sort $\mathbb{O}$ with $\tau(\mathbb{O}) = o$. We need this sort of truth values, since types are not first-class objects of $\Sigma\mathcal{HOL}$, and we want to be able to characterize sentences and and propositions by their sort. Finally, we assume that $\mathbb{O}$ is

top sort
sort!top
$q^{\mathbb{A}}, \neg, \wedge, \Pi^{\mathbb{A}}$ *logical constants*
$\Sigma^{\Sigma\mathcal{HOL}}$, *sig:empty′ signatures with logical constants*
logical constant
constant!logical
connective
quantifier
equality constant

the top sort of type o, i.e. $\mathbb{A} \in \mathcal{BS}_o$, then $[\mathbb{A} \leq \mathbb{O}] \in \Sigma$. Otherwise it would be possible to specify signatures that severely distort the intended semantics of the universe of truth values.

We could have achieved the same goal by prohibiting any other sorts of type o, but as we will see our more general solution provides a powerful means for specifying the semantics of predicates (10.1.2), and even allows alternate, specialized formalizations of logical calculi (10.1.3).

In our sorted setting we have a problem, when naively a priori assuming the existence of the logical constants, since we need infinitely many declarations of the form $[p^{\mathbb{A}}{::}\mathbb{A} \to \mathbb{A} \to \mathbb{O}]$ for logical constants $q^{\mathbb{A}}_{\alpha\to\alpha\to o}$, because there are infinitely many sorts $\mathbb{A}$.

As a solution we extend the definition of valid signature to encompass an infinite sub-signature of logical constants. This will not lead us into problems, since the type of formulae is always effectively computable and the set of sorts of a given type is always finite by 7.1.2.

Definition 10.1.1 (Logical Constants)

$$\Sigma^{\Sigma\mathcal{HOL}} := \left\{ \begin{array}{l} [\neg{::}\mathbb{O} \to \mathbb{O}], [\wedge{::}\mathbb{O} \to \mathbb{O} \to \mathbb{O}], \\ [q^{\mathbb{A}}{::}\mathbb{A} \to \mathbb{A} \to \mathbb{O}], [\Pi^{\mathbb{A}}{::}(\mathbb{A} \to \mathbb{O}) \to \mathbb{O}] \end{array} \middle| \, \mathbb{A} \in \mathcal{S} \right\}$$

Note that every finite subset of $\Sigma^{\Sigma\mathcal{HOL}}$ is a valid signature, since all these constants are distinct and can therefore be added with the rule *sig:const*. As we have argued above, we can also assume all of $\Sigma^{\Sigma\mathcal{HOL}}$ to be well-sorted. We will implement this by replacing the rule *sig:empty* with

$$\frac{}{\vdash_{sig} \Sigma^{\Sigma\mathcal{HOL}}} \; sig{:}empty'$$

We will assume that all signatures in $\Sigma\mathcal{HOL}$ are supersets of $\Sigma^{\Sigma\mathcal{HOL}}$, and refrain from mentioning $\Sigma^{\Sigma\mathcal{HOL}}$.

Just as in $\mathcal{HOL}$, we call the constants $q^{\mathbb{A}}, \neg, \wedge, \Pi^{\mathbb{A}}$ **logical constant**s. More specifically $\neg, \wedge$ are called **connective**s and $\Pi^{\mathbb{A}}$ a **quantifier** in order to distinguish them from the **equality constant** $q^{\mathbb{A}}$.

Example 10.1.2 (Subsorts of $\mathbb{O}$)
Let $\mathbb{T}$ and $\mathbb{F}$, be sorts of type o and

$$\Sigma^{\mathbb{F},\mathbb{T}} := \left\{ \begin{array}{l} [\mathbb{T} \leq \mathbb{O}], [\mathbb{F} \leq \mathbb{O}], [\neg{::}\mathbb{T} \to \mathbb{F}], [\neg{::}\mathbb{F} \to \mathbb{T}], \\ [\wedge{::}\mathbb{T} \to \mathbb{T} \to \mathbb{T}], [\wedge{::}\mathbb{T} \to \mathbb{F} \to \mathbb{F}], \\ [\wedge{::}\mathbb{F} \to \mathbb{T} \to \mathbb{F}], [\wedge{::}\mathbb{F} \to \mathbb{F} \to \mathbb{F}] \end{array} \right\}$$

then $\Sigma^{\mathbb{F},\mathbb{T}}$ is a valid signature in $\Sigma^d\Lambda^{\to}$. For $\Sigma^r\Lambda^{\to}$, we would need a variant of the declarations, where e.g. $[(\lambda X_{\mathbb{T}}.\neg X){::}\mathbb{T} \to \mathbb{F}]$; we will do the examples in this section for $\Sigma^r\Lambda^{\rightarrow}$, since the notation is simpler, they also work for $\Sigma^r\Lambda^{\rightarrow}$ with the same trick.

Note that with the same argumentation about finiteness as in Definition 10.1.1 we can even extend $\Sigma^{\mathbb{F},\mathbb{T}}$ with the declarations $[\Pi^{\mathbb{A}}{::}(\mathbb{A} \to \mathbb{T}) \to \mathbb{T}]$ for all $\mathbb{A} \in \mathcal{S}$.

Clearly $\Sigma^{\mathbb{F},\mathbb{T}}$ gives complete semantic information for the connectives and partial information for the quantifier $\Pi^{\mathbb{A}}$, if we interpret $\mathbb{T}$ as the subset $\mathcal{D}_{\mathbb{T}} = \{\mathtt{T}\} \subset \mathcal{D}_{\mathbb{O}}$ and $\mathbb{F}$ as $\mathcal{D}_{\mathbb{F}} = \{\mathtt{F}\} \subset \mathcal{D}_{\mathbb{O}}$.

Let us now use this in the context of our running example from elementary calculus. Recall that we had sorts $\mathbb{R}$ for the real numbers and $\mathbb{R}^+$ for the non-negative reals. So let us augment $\Sigma^{\mathbb{T},\mathbb{F}}$ with sorts $\mathbb{R}^-$, $\mathbb{R}_0^+$, and $\mathbb{R}_0^-$ for the non-positive, positive, and negative reals, and with the term declarations

$$\begin{array}{l}
[\mathbb{R}^+ \leq \mathbb{R}], [\mathbb{R}^+ \leq \mathbb{R}_0^+], [\mathbb{R}^- \leq \mathbb{R}], [\mathbb{R}^- \leq \mathbb{R}_0^-],\\
[1{::}\mathbb{R}_0^+], [3{::}\mathbb{R}_0^+], [-5{::}\mathbb{R}_0^-], [0{::}\mathbb{R}^+ \sqcap \mathbb{R}^-],\\
[\leq{::}\mathbb{R} \to \mathbb{R} \to \mathbb{O}], [\lambda X_{\bullet} X \leq X{::}\mathbb{R} \to \mathbb{T}],\\
[\leq{::}\mathbb{R}_0^+ \to \mathbb{N} \to \mathbb{F}], [\leq{::}\mathbb{N} \to \mathbb{P} \to \mathbb{T}], [\leq{::}\mathbb{P} \to \mathbb{Z} \to \mathbb{F}],\\
[+{::}\mathbb{R} \to \mathbb{R} \to \mathbb{R}], [+{::}\mathbb{P} \to \mathbb{P} \to \mathbb{P}], [+{::}\mathbb{N} \to \mathbb{N} \to \mathbb{N}], [+{::}\mathbb{Z} \to \mathbb{P} \to \mathbb{P}] \ldots
\end{array}$$

then formulae like $-5+0 \leq 3$ and $\forall X_{\mathbb{P}\bullet} -5 \leq 3+X$ can be seen to be of sort $\mathbb{T}$, while $\neg 0 \leq 0$ and $0+1 \leq 0$ have sort $\mathbb{F}$. Moreover we have $\vdash_\Sigma \forall P_{\mathbb{T}\bullet} P \Rightarrow P$ and $\vdash_\Sigma \forall P_{\mathbb{F}\bullet} P \Rightarrow P$. But we cannot infer $\vdash_\Sigma \forall P_{\mathbb{O}\bullet} P \Rightarrow P$ in $\Sigma\mathcal{HOL}$, since the sort mechanism of $\Sigma\mathcal{HOL}$ cannot manipulate sort information that formalizes that $\mathbb{O}$ is the union of $\mathbb{T}$ and $\mathbb{F}$.[62] EdNote(62)

We will now see that the subsorts of $\mathbb{O}$ can also be used to embed some of the sort information into the calculus.

Example 10.1.3 ($\Sigma\mathfrak{T}$ with Sorts)
We can give an alternative formalization of the calculus $\Sigma\mathfrak{T}$ using term declarations: if we augment the signature from 10.1.2 with the declarations

$$\begin{array}{l}
[\lambda P_{\bullet}(P \vee P) \Rightarrow P{::}\mathbb{O} \to \mathbb{T}], [\lambda P_{\bullet} P \Rightarrow (P \Rightarrow P){::}\mathbb{O} \to \mathbb{T}],\\
[\lambda P, Q_{\bullet}(P \vee Q) \Rightarrow (Q \Rightarrow P){::}\mathbb{O} \to \mathbb{T}],\\
[\lambda P, Q, R_{\bullet}(P \vee Q) \Rightarrow_{\bullet}(R \vee P) \Rightarrow (R \vee Q){::}\mathbb{O} \to \mathbb{T}],\\
[\lambda F, X_{\bullet}\Pi^{\mathfrak{d}(\mathbb{A})} F \Rightarrow FX{::}(\mathbb{A} \to \mathbb{O}) \to \mathbb{A} \to \mathbb{T}],\\
[\lambda P, F_{\bullet}\forall X{::}\mathbb{A}(P \vee FX) \Rightarrow_{\bullet} P \vee \Pi^{\mathbb{A}} F{::}\mathbb{O} \to (\mathbb{A} \to \mathbb{O}) \to \mathbb{T}],
\end{array}$$

and augment $\Sigma\mathfrak{T}$ with the inference rule

$$\frac{\Gamma \vdash_\Sigma \mathbf{A}{::}\mathbb{T}}{\Gamma \Vdash^{\iota}_{\Sigma} \mathbf{A}}$$

[62] EdNote: Check, work out the sorts.

sorted collection
$\overline{\mathcal{A}}$pre-$\overline{\Sigma}$-structure corresponding to $\mathcal{A}$
pre-Σ-structure
partial pre-$\overline{\Sigma}$-structure

then we can drop all of the propositional axioms of $\Sigma\mathfrak{T}$. The gain of this measure is greater than apparent at first glance. As we have seen in example 10.1.2, a whole class of tautologies (which can be arbitrarily extended by providing further term declarations) can be shown to be of sort $\mathbb{T}$, and thus is specially treated by the calculus. Thus the term declaration mechanism offers a tool to adapt certain calculi to specific needs.

BegNP(63) Now we turn to the semantics of $\Sigma\mathcal{HOL}$.

10.2 A Set-Theoretic Semantics for $\Sigma^r\Lambda^{\rightarrow}$

With the previously defined concept of valid signatures we can now lay the framework of Σ-structures, which serves as an algebraic basis for our development of $\Sigma\Lambda^{\rightarrow}$.

Note that the definition does not take the information given in the term declarations into account, but only concentrates on the sort structure. This is natural for pre-Σ-structures, since the underlying pre-$\overline{\Sigma}$-structures do not assume anything about comprehension-closedness, and therefore do not guarantee denotations for the formulae in term declarations. Hence it is not possible to give a definition that takes the term declarations into account either. This situation will be better in Σ-structures, which we are about to define. Since we have taken care to define all the relevant notions of $\Lambda^{\rightarrow}$ in a partial-function setting, we can now simply use the value function defined there. This is in par with the intuition, that sorts are only a refinement constraint on the λ-calculus.

Definition 10.2.1 (Sorted Collection) In analogy to the typed case we define **sorted collection**s of sets, functions, and relations by substituting sorts for types in the definitions. But we do not insist that $\mathcal{D}_{\mathbb{A}}$ and $\mathcal{D}_{\mathbb{B}}$ be disjoint in a sorted collection $\mathcal{D}_{\mathcal{S}}$ of sets for distinct sorts $\mathbb{A}$ and $\mathbb{B}$, since it is intended e.g. for well-sorted formulae to have multiple sorts. Note that sorted collections are also typed collections, since sorts have types: Let $\mathcal{D}_{\mathcal{S}}$ be a sorted collection, then $\overline{\mathcal{D}}_{\mathcal{T}}$ defined by $\overline{\mathcal{D}}_{\alpha} := \bigcup_{\tau(\mathbb{A})=\alpha} \mathcal{D}_{\mathbb{A}}$ is a typed collection. We call $\overline{\mathcal{D}} := \overline{\mathcal{D}}_{\mathcal{T}}$ the typed collection corresponding to $\mathcal{D} = \mathcal{D}_{\mathcal{S}}$.

Definition 10.2.2 (Pre-Σ-Structures) Let Σ be a valid signature and $\mathcal{D}_{\mathcal{S}}$ a sorted collection of sets, then we call the triple $(\mathcal{D}_{\mathcal{S}}, @, \mathcal{I})$ a **pre-Σ-structure**, if

1. $\mathcal{A} := (\overline{\mathcal{D}}_{\mathcal{T}}, @, \mathcal{I})$ is a **partial pre-$\overline{\Sigma}$-structure** (cf. 1.4.2), where $\overline{\mathcal{D}}_{\alpha} := \bigcup_{\tau(\mathbb{A})=\alpha} \mathcal{D}_{\mathbb{A}}$,
2. For all functional sorts $\mathbb{A} \in \mathcal{S}_{\alpha\rightarrow\beta}$, we have $\mathcal{D}_{\mathbb{A}} @ \mathcal{D}_{\mathfrak{d}(\mathbb{A})} \subseteq \mathcal{D}_{\mathfrak{r}(\mathbb{A})}$ and

[63] New Part:

- $\mathbf{Dom}(@\mathcal{D}_{\mathbb{A}}) = \mathcal{D}_{\mathfrak{d}(\mathbb{A})}$ for $\Sigma^r\Lambda^{\rightarrow}$
- $\mathbf{Dom}(@) \supseteq \mathcal{D}_{\mathbb{A}} \times \mathcal{D}_{\mathfrak{d}(\mathbb{A})}$ for $\Sigma^d\Lambda^{\rightarrow}$.

universe of sort $\mathbb{A}$
comprehension-closed
algebra!comprehension-closed/functional
functional
homomorphism
monomorphism
epimorphism
isomorphism
descriptive sorts
restrictive sorts
assignment!Γ
structureΣ

Now we can adapt the nomenclature from definition 1.4.2 to the sorted case. In particular, the set $\mathcal{D}_{\mathbb{A}}$ is called the **universe of sort** $\mathbb{A}$. Note that in contrast to the pre-Σ-structures we require pre-Σ-structures to be total on the sorted universes $\mathcal{D}_{\mathbb{A}}$. For a pre-Σ-structure $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ we call the partial pre-$\overline{\Sigma}$-structure $\overline{\mathcal{A}} = (\overline{\mathcal{D}}, @, \mathcal{I})$ the **corresponding pre-$\overline{\Sigma}$-structure**. We call $\mathcal{A}$ **comprehension-closed** (**functional**), iff $\overline{\mathcal{A}}$ is.

Let $\mathcal{A} = (\mathcal{D}, @^{\mathcal{A}}, \mathcal{I})$ and $\mathcal{B} = (\mathcal{E}, @^{\mathcal{B}}, \mathcal{J})$ be pre-Σ-structures, then a $\overline{\Sigma}$-homomorphism $\kappa : \overline{\mathcal{A}} \longrightarrow \overline{\mathcal{B}}$ is called a Σ-**homomorphism**, iff $\kappa(\mathcal{D}_{\mathbb{A}}) \subset \mathcal{E}_{\mathbb{A}}$ for all sorts $\mathbb{A} \in \mathcal{S}$. κ is called Σ-**monomorphism**, if it is injective, and a Σ-**epimorphism**, if it is surjective and moreover $\kappa(\mathcal{D}_{\mathbb{A}}) = \kappa(\mathcal{E}_{\mathbb{A}})$ for all $\mathbb{A} \in \mathcal{S}$. We call κ a Σ-**isomorphism**, iff it is an injective Σ-epimorphism.

This definition makes a small, but essential difference between the restrictive and descriptive sort disciplines by insisting that the domain of functional objects is exactly that specified by the domain sort for the restrictive sorts. For $\Sigma^d\Lambda^{\rightarrow}$ we are more liberal by only requiring that it is defined everywhere on the specified domain.

Definition 10.2.3 (Σ-structure) Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a pre-Σ-structure and Γ a variable context, then we call a function $\varphi : \mathbf{Dom}(\Gamma) \longrightarrow \mathcal{D}_{\mathcal{S}}$ a Γ-**assignment** into $\mathcal{A}$, iff $\varphi(X) \in \mathcal{D}_{\mathbb{A}}$ for every $X \in \mathbf{Dom}(\Gamma)$ with $\Gamma(X) = \mathbb{A}$. We call a functional pre-Σ-structure $\mathcal{A}$ a Σ-**structure**, iff it is comprehension-closed and for all term declarations $[\mathbf{A} :: \mathbb{A}] \in \Sigma$ we have $\mathcal{I}(\mathbf{A}) \in \mathcal{D}_{\mathbb{A}}$.[64] EdNote(64)

Now we state some lemmata, which are direct consequences of their counterparts in $\Lambda^{\rightarrow}$. Here we take advantage of the fact that we have invested some extra work for $\Lambda^{\rightarrow}$ by generalizing all notions to partial functions.

Lemma 10.2.4 *Sorted $\beta\eta$-conversion is sound, i.e. if $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ is a Σ-structure and $\Gamma \vdash_{\Sigma} \mathbf{A} =_{\beta\eta} \mathbf{B}$, then $\mathcal{I}_{\varphi}(\mathbf{A}) = \mathcal{I}_{\varphi}(\mathbf{B})$ for any Γ-assignment φ.*

Proof: This result is an immediate consequence of the unsorted result for $\overline{\Sigma}$-structures (see lemma 1.4.24). □

Lemma 10.2.5 *Let $\Gamma \vdash_{\Sigma} \mathbf{A} :: \mathbb{A}$ and $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a Σ-structure, then for any Γ-assignment φ into $\mathcal{A}$ we have $\mathcal{I}_{\varphi}(\mathbf{A}) \in \mathcal{D}_{\mathbb{A}}$. As a consequence $\mathcal{I}_{\varphi}$ is a Σ-homomorphism.*

[64] EdNote: Rework, check the semantics.

congruence!Σ
functional

Proof: We prove the assertion by an induction on the structure of $\mathcal{D}\colon \Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$. For the ground cases we remark that, if $\mathcal{D}$ ends in *ws:var* or *ws:td*, the assertion holds, since φ is a Γ-assignment, and $\mathcal{A}$ is a Σ-structure (where the assertion holds for term declarations by definition). In the cases where $\mathcal{D}$ ends in *ws:app* or *ws:abs* we obtain the assertion from the inductive hypothesis and the definition of $\mathcal{I}_\varphi$. For *ws:$\beta\eta$* note that $\Gamma \vdash_\Sigma \mathbf{A} =_{\beta\eta} \mathbf{B}$ implies $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$ (10.2.4). □

In particular, we have the same tight correspondence between substitutions and variable assignments as in $\Lambda^{\rightarrow}$.

Theorem 10.2.6 (Substitution Value Theorem) *Let $\mathcal{A} = (\mathcal{D}, @, \mathcal{I})$ be a Σ-structure, $\mathbf{A} \in wsf(\Sigma; \Gamma, [X::\mathbb{A}])$, $\mathbf{B} \in wsf(\Sigma; \Delta)$, and φ be a Γ, Δ-assignment, then $\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A}) = \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{A})$.*

EdNote(65) **Proof:** By 4[65]. □

Definition 10.2.7 Let $\mathcal{A}$ be a pre-Σ-structure, then a congruence $\sim$ on $\overline{\mathcal{A}}$ is called a Σ-**congruence** on $\mathcal{A}$, iff $f \in \mathcal{D}_\mathbb{B}$ and $g \sim f$ imply $g \in \mathcal{D}_\mathbb{B}$. Here we have adapted the definition of a $\overline{\Sigma}$-congruence (cf. 1.4.6) by requiring a totality condition for domain sorts, that is analogous to that in the definition of Σ-structures.

A Σ-congruence $\sim$ is called **functional**, iff for all functional sorts $\mathbb{A} \in \mathcal{S}^f$ and all $f, g \in \mathcal{D}_\mathbb{A}$ the fact that $f@a \sim g@a$ for all $a \in \mathcal{D}_{\mathfrak{d}(\mathbb{A})}$ implies $f \sim g$. Note that, since $\sim$ is a congruence, we also have the other direction, so
EdNote(66) $f@a \sim g@a$ for all $a \in \mathcal{D}_\beta$, iff $f \sim g$.[66]

Lemma 10.2.8 *If $\sim$ is a Σ-congruence on a pre-Σ-structure $\mathcal{A}$, then $\mathcal{A}/_\sim$ is a pre-Σ-structure as well, and $\pi_\sim$ is a Σ-homomorphism. Furthermore,*
EdNote(67) *$\mathbf{A}/_\sim$ is comprehension-closed, iff $\mathcal{A}$ is, and functional, if $\sim$ is*[67].

Proof: In the light of 1.4.2 and 10.2.2 it only remains to show that $\mathcal{D}^\sim_\mathbb{A} @^\sim \mathcal{D}^\sim_{\mathfrak{d}(\mathbb{A})} \subseteq \mathcal{D}^\sim_{\mathfrak{r}(\mathbb{A})}$ and that $@^\sim$ is a total function on $\mathcal{D}^\sim_\mathbb{A} \times \mathcal{D}_{f\mathfrak{d}(\mathbb{A})}$. So let $f \in \mathcal{D}_\mathbb{A}$ and $a \in \mathcal{D}_{\mathfrak{d}(\mathbb{A})}$, then $[\![f]\!] \in \mathcal{D}^\sim_\mathbb{A}$ and $[\![a]\!] \in \mathcal{D}^\sim_{\mathfrak{d}(\mathbb{A})}$ and $[\![f]\!] @^\sim [\![a]\!] = [\![f@a]\!] \in \mathcal{D}^\sim_{\mathfrak{r}(\mathbb{A})}$, since $f@a \in \mathcal{D}_{\mathfrak{r}(\mathbb{A})}$. □

Lemma 10.2.9 *Let $W := \{X^1_{\alpha_1}, \ldots, X^n_{\alpha_n}\}$ and $\overline{\Sigma} := \{c^1_{\beta_1}, \ldots, c^m_{\beta_m}\}$ be typed sets of variables and constants. Furthermore, let*

$$\begin{array}{rcl} W^\sharp & := & \{[X^1::\sharp(\alpha_1)], \ldots, [X^n::\sharp(\alpha_n)]\} \\ \overline{\Sigma}^\sharp & := & \{[c^1::\sharp(\beta_1)], \ldots, [c^m::\sharp(\beta_m)]\} \end{array}$$

for some context Γ with $\mathbf{Dom}(\Gamma) = W$, then

[65] EdNote: Adapt all this to the semantic notions in the first part.
[66] EdNote: do we have to make a difference between $\Sigma^r\Lambda^{\rightarrow}$ and $\Sigma^d\Lambda^{\rightarrow}$ here, I think that we have to, compare with η.
[67] EdNote: Redo for the systems

trivially sorted
pre-Σ-algebra
algebra!(pre)-Σ

1. $\overline{\Sigma}^\sharp$ *is a valid signature.*

2. $\sharp\colon \mathit{wff}(\overline{\Sigma}, W) \longrightarrow \mathit{wsf}(\Sigma, \Gamma)$ *is a* $\overline{\Sigma}$*-monomorphism and* $\flat\colon \mathit{wsf}(\Sigma,\Gamma) \longrightarrow \mathit{wff}(\overline{\Sigma}, W)$ *is a* $\overline{\Sigma}$*-epimorphism.*

3. *For* $\mathbf{A} \in \mathit{wff}(\overline{\Sigma}, W)$ *we have* $\Sigma \vdash \mathbf{A}\colon \alpha$*, iff* $\Gamma \vdash_{\overline{\Sigma}^\sharp} \sharp(\mathbf{A})::\sharp(\alpha)$.

4. *For* $\mathbf{A} \in \mathit{wsf}(\Sigma, \Gamma)$ *we have* $\Gamma \vdash_{\overline{\Sigma}^\sharp} \mathbf{A}::\mathbb{A}$*, iff* $\overline{\Sigma} \vdash \flat(\mathbf{A})\colon \flat(\mathbb{A})$.

5. $\flat(\sharp(\mathbf{A})) = \mathbf{A}$ *for al* $\mathbf{A} \in \mathit{wff}(\overline{\Sigma}, W)$*, thus* $\flat$ *is an inverse for* $\sharp$.

6. $\sharp$ *and* $\flat$ *are* $\overline{\Sigma}$*-isomorphisms, if* $\overline{\Sigma}^\sharp$ *is trivially sorted.*

Proof: Immediate from the definitions. □

Remark 10.2.10 $\Sigma\Lambda^{\rightarrow}$ is a generalization of $\Lambda^{\rightarrow}$.

Proof: If Σ is trivially sorted, then Σ contains exactly one term declaration $[c::\mathbb{B}]$ with $\tau(\mathbb{B}) = \alpha$ for each constant $c_\alpha \in \overline{\Sigma}_\alpha$, since $\|$ is just the equality relation and we have assumed declarations for all constants in $\overline{\Sigma}$. Thus trivially sorted signatures are isomorphic (by τ) to the signatures of 1.5.6. It is easy to see that sorted $\beta\eta$-conversion is just unsorted $\beta\eta$-conversion, which is sort-preserving even without the rule *ws:*$\beta\eta$, therefore *ws:*$\beta\eta$ becomes redundant in the trivially sorted case. Thus the judgment $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ coincides with the judgment $\overline{\Sigma} \vdash \mathbf{A}\colon \tau(\mathbb{A})$, and therefore well-sortedness just reduces to well-typedness, and the functions $\sharp$ and $\flat$ are isomorphisms of pre-$\overline{\Sigma}$-structures. □

Remark 10.2.11 If Σ is a valid signature and Γ is a context, then $\mathit{wsf}(\Sigma, \Gamma)$ is a sorted pre-Σ-structure[68], if $\mathbf{A}@\mathbf{B} = (\mathbf{AB})$, since *ws:app* ensures the totality condition 10.2.2(2): This is nearly trivial for the more liberal conditional for $\Sigma^d\Lambda^{\rightarrow}$. For $\Sigma^r\Lambda^{\rightarrow}$ Let $\mathbf{A}$ be a formula of sort $\mathbb{A} \in \mathcal{S}^f$, then $\mathbf{AC}$ is well-sorted entails that $\mathbf{C}$ has sort $\mathfrak{d}(\mathbb{A})$[69]. EdNote(68) EdNote(69)

Example 10.2.12 (Valid Signatures and Well-Sorted Formulae) [70] EdNote(70)

Definition 10.2.13 (Σ-Algebra) Let Σ be a valid signature, then a **pre-Σ-algebra** $\mathcal{A} = (\mathcal{D}, \mathcal{I})$ is a pre-Σ-structure such that

1. for all $\mathbb{A} \in \mathcal{S}^f$ we have $\mathcal{D}_\mathbb{A} \subset \mathcal{F}(\mathcal{D}_{\mathfrak{d}(\mathbb{A})}; \mathcal{D}_{\mathfrak{r}(\mathbb{A})})$,

2. $f@a = f(a)$.

[68] EdNote: This still has to be defined, maybe do a whole chapter on the semantics later.
[69] EdNote: ref, complete the argument
[70] EdNote: coming up

standard
$\mathcal{TS}(\Sigma,\Gamma)$term structure for Σ and Γ
term structure
structure!term

Like in the unsorted case we call a pre-Σ-algebra **standard**, iff $\mathcal{D}_{\mathbb{A}\to\mathbb{B}} = \mathcal{F}(\mathcal{D}_{\mathbb{A}};\mathcal{D}_{\mathbb{B}})$.

Note that $\overline{\mathcal{A}} = (\overline{\mathcal{D}}_\alpha, \mathcal{I})$ forms a pre-$\overline{\Sigma}$-algebra, therefore pre-Σ-algebras are functional. We call a pre-Σ-algebra $\mathcal{A}$ a **Σ-algebra**, iff it is a Σ-structure.

Example 10.2.14 (Rudimentary Calculus (Semantics)) Let Σ be the signature from example 7.7.22, i.e.

$$\Sigma = \{[\pi::\mathbb{P}], [a::(\mathbb{R}\to\mathbb{P})], [i::\mathbb{C}], [d::(\mathbb{C}\to\mathbb{R}\to\mathbb{R})], [\forall[X::\mathbb{P}].X::\mathbb{R}]\}$$

Then the definitions $\mathcal{D}_{\mathbb{R}} := \mathbb{R}$ and $\mathcal{D}_{\mathbb{C}} := \mathcal{C}^1(\mathbb{R};\mathbb{R})$, together with the convention that the interpretation of π, i, a, and d are the number π (3.1415...), the identity function, absolute value function and $\mathcal{I}(d)$ the differentiation operator respectively specify a standard pre-Σ-algebra, whereas the setting $\mathcal{D}_{\mathbb{R}\to\mathbb{R}} := \mathcal{C}^0(\mathbb{R};\mathbb{R})$ defines a class of non-standard pre-Σ-algebras.

Now we use the results of this section to define suitable notions of term algebras.

Definition 10.2.15 (Term Σ-Structure) Let $\mathcal{D}_{\mathbb{A}}$ be the set of well-sorted formulae $\mathbf{A}$ in sorted $\beta\eta$-normal form such that $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$. Furthermore, let $\mathbf{A}@\mathbf{B}$ be the sorted $\beta\eta$-normal form of $(\mathbf{AB})$ and $\mathcal{I} := \mathrm{Id}_{\overline{\Sigma}}$, then we call $\mathcal{TS}(\Sigma,\Gamma) := (\mathcal{D}_{\mathcal{S}}, @, \mathcal{I})$ the **term structure** for Σ and Γ.

This definition is justified by the following lemma.

Lemma 10.2.16 $\mathcal{TS}(\Sigma,\Gamma)$ *is a Σ-structure with* $\mathcal{I}_\varphi(\mathbf{A}) = \varphi(\mathbf{A})\!\downarrow$.

Proof: By 1.4.21 $\overline{\mathcal{TS}(\Sigma,\Gamma)} = \mathcal{TS}(\Sigma)$ is a pre-$\overline{\Sigma}$-structure. Let Δ be a finite variable context such that $\mathbf{Dom}(\Delta) \cap \mathbf{Dom}(\Gamma) = \emptyset$ and $\Delta \vdash_\Sigma \mathbf{A}::\mathbb{A}$ and φ be a Δ-assignment into $\mathcal{TS}(\Sigma,\Gamma)$, then φ is a Σ-substitution, since it is a Δ-assignment. We can convince ourselves that $\mathcal{I}_\varphi(\mathbf{A}) = \sigma(\mathbf{A})\!\downarrow$ by a simple induction on the structure of formulae using

$$\sigma(\lambda X.\mathbf{A})@\mathbf{B} = (\lambda X.\sigma_{-X}\mathbf{A})@\mathbf{B} = \sigma, [\mathbf{B}/X]\mathbf{A} = \mathcal{I}_{\varphi,[\mathbf{B}/X]}(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{A})@\mathbf{B}$$

Thus $\mathcal{I}_\varphi(\mathbf{A}) = \sigma(\mathbf{A})\!\downarrow \in \mathcal{TS}_{\mathbb{A}}(\Sigma,\Gamma)$ by 7.6.1 and 7.3.4. In other words, $\mathcal{TS}(\Sigma,\Gamma)$ is comprehension-closed. Finally, for any term declaration $[\mathbf{A}::\mathbb{A}] \in \Sigma$ we have $\Gamma \vdash_\Sigma \mathbf{A}::\mathbb{A}$ by *ws:td*, and thus we can verify $\mathcal{I}_\varphi \in \mathcal{TS}_{\mathbb{A}}(\Sigma,\Gamma)$ with the same arguments. $\square$

We now convince ourselves that we need only consider term declarations of a certain syntactically restricted form, since all other term declarations can be replaced by term declarations in this form without changing the set of well-formed formulae.

EdNote(71) **Remark 10.2.17** Considering the semantics for $\Sigma\Lambda^{\rightarrow}$, we see that a more general version of the *ior:cov* inference rule[71]

$$\frac{\Sigma \vdash \mathfrak{d}(\mathbb{A}) = \mathfrak{d}(\mathbb{B}) \quad \Sigma \vdash \mathfrak{r}(\mathbb{A}) \leq \mathfrak{r}(\mathbb{B})}{\Sigma \vdash \mathbb{A} \leq \mathbb{B}}\, ior{:}cov'$$

would not be correct. Let $\mathcal{D}_{\mathbb{A}}$ and $\mathcal{D}_{\mathbb{B}}$ be the classes of surjective functions in $\mathcal{F}(\mathcal{D}_{\mathfrak{d}(\mathbb{A})}; \mathcal{D}_{\mathfrak{r}(\mathbb{A})})$ and $\mathcal{F}(\mathcal{D}_{\mathfrak{d}(\mathbb{B})}; \mathcal{D}_{\mathfrak{r}(\mathbb{B})})$. If $\Sigma \vdash \mathfrak{r}(\mathbb{A}) \leq \mathfrak{r}(\mathbb{B})$, then $\mathcal{D}_{\mathfrak{r}(\mathbb{A})} \subset \mathcal{D}_{\mathfrak{r}(\mathbb{B})}$ in any Σ-structure $(\mathcal{D}, @, \mathcal{I})$ but functions from $\mathcal{D}_{\mathbb{A}}$ would in general not be surjective in the larger codomain $\mathcal{D}_{\mathfrak{r}(\mathbb{B})}$ and therefore not in $\mathcal{D}_{\mathbb{B}}$. For similar reasons an inference rule like

$$\frac{\mathbb{F} \leq \mathbb{G}}{\mathfrak{r}(\mathbb{F}) \leq \mathfrak{r}(\mathbb{G})}\, ior{:}nat^{-1}$$

cannot be correct. Let $\mathcal{D}_{\mathbb{A}} := \{a\}$, $\mathcal{D}_{\mathbb{B}} := \{b, d\}$, $\mathcal{D}_{\mathbb{C}} := \{b, c\}$, $\mathfrak{r}(\mathbb{F}) = \mathbb{B}$, $\mathfrak{d}(\mathbb{F}) = \mathbb{A}$, and $\mathbb{G} := \mathbb{A} \rightarrow \mathbb{C}$. Then the definitions $\mathcal{D}_{\mathbb{F}} := \{\{(a, c)\}\}$ and $\mathcal{D}_{\mathbb{G}} := \{\{(a, b)\}, \{(a, c)\}\}$ satisfy the preconditions ($\mathcal{D}_{\mathbb{F}} \subseteq \mathcal{D}_{\mathbb{G}}$), but clearly we do not have $\mathcal{D}_{\mathbb{B}} \subseteq \mathcal{D}_{\mathbb{C}}$. In other words, the sort semantics for codomains is less tight than that for domains in the restrictive sort discipline.

Remark 10.2.18 We have assumed that $\mathbb{O}$ is the top sort of type o, thus for all $\mathbb{A}$ of type o we have $\Sigma \vdash \mathbb{A} \sqsubseteq \mathbb{B}$ and thus $\mathcal{D}_{\mathbb{A}} \subseteq \mathcal{D}_{\mathbb{O}}$. Together with the fact that $\mathcal{D}_o$ has two elements (cf. **??**), this guarantees that $\overline{\mathcal{D}}_o = \bigcup_{\mathbb{A} \in \mathcal{BS}} \mathcal{D}_{\mathbb{A}} = \mathcal{D}_{\mathbb{O}}$.[72] EdNote(72)

10.3 A Categorical Semantics for $\Sigma^d\Lambda^{\rightarrow}$ (Partial Equivalence Relations)

partial equivalence relation
equivalence relation!partial
domain
function PER

In this section we will sketch a semantics for the descriptive sorted λ-calculus. It is not a set-theoretic semantics, indeed it is impossible to give one for $\Sigma^d\Lambda^{\rightarrow}$[73] EdNote(73)

Definition 10.3.1 (Partial Eqivalence Relations) Let U be a set, then we call a relation $R \subseteq U \times U$ a **partial equivalence relation**, iff R is transitive and symmetric (and not neccessary reflexive). If R is a PER, then we say that $\mathbf{Dom}(R) := \{[a]_R \mid aRa\}$ is the **domain** of R.

For two PER P and Q, we define the **function PER** $P \longrightarrow Q$ by $\mathsf{f}(P \longrightarrow Q)\mathsf{g}$, iff $\mathsf{f}(\mathsf{p})Q\mathsf{g}(\mathsf{q})$ for all $\mathsf{p}P\mathsf{q}$. If $\mathsf{f} \in \mathbf{Dom}(P \longrightarrow Q)$ and $\mathsf{p} \in \mathbf{Dom}(P)$, then we call $\mathsf{f}@\mathsf{p} := \{f(p) \mid f \in \mathsf{f}, p \in \mathsf{p}\}$ the application of f to p.

[71] EdNote: adapt to the descriptive case, what is the semantics there anyway? For all $\mathsf{f} \in \mathcal{D}_{\mathbb{A}\rightarrow\mathbb{B}}$, $\mathbf{Dom}(\mathsf{f}) \supseteq \mathcal{D}_{\mathbb{A}}$ and $\mathsf{f}(\mathcal{D}_{\mathbb{A}}) \subseteq \mathcal{D}_{\mathbb{B}}$.
[72] EdNote: Put somewhere else.
[73] EdNote: Check, discuss some more

sub-PER

The definition of the application operator @ for PER is justified by the following observation.

Lemma 10.3.2 *If P and Q are partial equivalence relations,* $\mathsf{f} \in \mathbf{Dom}(P \longrightarrow Q)$*, and* $\mathsf{p} \in \mathbf{Dom}(P)$*, then* $f(p)Qf'(p')$ *iff* $f, f' \in \mathsf{f}$ *and* $p, p' \in \mathsf{p}\}$.

Proof: Let $f, f' \in \mathsf{f}$, then $\mathbf{Dom}(P \longrightarrow Q)$ iff $f(P \longrightarrow Q)f'$ by definition of $\mathbf{Dom}(P \longrightarrow Q)$, iff $f(p)Qf'(p')$ by definition of $(P \longrightarrow Q)$. □

Lemma 10.3.3 *If P and Q are partial equivalence relations, then $P \longrightarrow Q$ is one.*

Proof: We first show transitivity: let $\mathsf{f}(P \longrightarrow Q)\mathsf{g}$ and $\mathsf{g}(P \longrightarrow Q)\mathsf{h}$ and $\mathsf{p}P\mathsf{q}$, then $\mathsf{f}(\mathsf{p})Q\mathsf{g}(\mathsf{q})$ and $\mathsf{g}(\mathsf{q})Q\mathsf{h}(\mathsf{q})$. So we have $\mathsf{f}(\mathsf{p})Q\mathsf{h}(\mathsf{q})$ by transitivity of Q.

For symmetry let $\mathsf{f}(P \longrightarrow Q)\mathsf{g}$ and $\mathsf{p}P\mathsf{q}$, then $\mathsf{q}P\mathsf{p}$ by symmetry of P and $\mathsf{f}(\mathsf{q})Q\mathsf{g}(\mathsf{p})$ by definition of $P \longrightarrow Q$. Thus we have $\mathsf{g}(\mathsf{p})Q\mathsf{f}(\mathsf{q})$ by symmetry of Q. □

Lemma 10.3.4 $\mathsf{f} \in \mathbf{Dom}(P \longrightarrow Q)$*, iff* $\mathsf{f}@\mathsf{p} \in \mathbf{Dom}(Q)$ *for all* $\mathsf{p} \in \mathbf{Dom}(P)$.

Proof: Let $f \in \mathsf{f}$ and $p \in \mathsf{p}$, then $\mathsf{f}@\mathsf{p} = [f(p)]_Q$ by definition. We have $f(P \longrightarrow Q)f$ since $\mathsf{f} \in \mathbf{Dom}(P \longrightarrow Q)$, thus $f(p)Qf(p')$ for all pPp' by definition of $P \longrightarrow Q$ and in particular $f(p)Qf(p)$ since pPp as $\mathsf{p} \in \mathbf{Dom}(P)$.[74] □

EdNote(74)

Remark 10.3.5 Note that PER is a cartesian closed category, so it is a model of the lambda calculus.[75]

EdNote(75)

Definition 10.3.6 (Sub-PER) Let P and Q be partial equivalence relations, then we call P a **sub-PER** of Q, iff $P \subseteq Q$ as relations on U.

Lemma 10.3.7 (PER is contravariant) *Let $P \subseteq Q$, then $Q \longrightarrow R \subseteq P \longrightarrow R$.*

Proof: Let $f(Q \longrightarrow R)g$ and pPp', then $f(q)Rg(q')$ for all qQq' by definition of $Q \longrightarrow R$. We have pQp', since $P \subseteq Q$ and thus $f(P \longrightarrow R)g$ by definition of $P \longrightarrow R$. □

[74] EdNote: Show the converse Direction

[75] EdNote: Explain

Definition 10.3.8 (PER-Model for $\Sigma^d\Lambda^{\rightarrow}$) Let $\vdash_{sig}\ \Sigma$ in $\Sigma^d\Lambda^{\rightarrow}$ and $\mathcal{S} := (\mathcal{D}, \mathcal{I})$ be Σ-structure. Furthermore, let $\mathcal{D}_{\mathbb{A}}$ be a PER on $\mathcal{D}_\alpha$ for all sorts $\mathbb{A}$ of type α. We call $\mathcal{D}_{\mathcal{S}}$ a **sorted frame** for $\Sigma^d\Lambda^{\rightarrow}$ over $\mathcal{S}$, iff $\mathsf{f}@\mathsf{a} \neq \emptyset$ for all functional base sorts $\mathbb{A}$ and all $\mathsf{a} \in \mathcal{D}_{\mathfrak{d}(\mathbb{A})}$ and $\mathsf{f}\mathcal{D}_{\mathbb{A}}$. For a variable context Γ, we say that $\varphi \colon \mathcal{V} \longrightarrow \mathcal{D}_{\mathcal{T}}$ is a Γ-assignment, iff $\varphi(X) \in \mathbf{Dom}(\mathcal{D}_{\mathbb{A}})$ for all $[X :: \mathbb{A}] \in \Gamma$. *sorted frame*

Let $\mathcal{D}_{\mathcal{S}}$ be a sorted frame over $\mathcal{S}$, then we call $\mathcal{S}$ a sorted PER model, iff $\mathcal{I}(\mathbf{A}) \in \mathbf{Dom}(\mathcal{D}_{\mathbb{A}})$ for all declarations $[\mathbf{A} :: \mathbb{A}] \in \Sigma$.

Lemma 10.3.9 *Let $(\mathcal{D}, \mathcal{I})$ be a Σ-PER structure and $\Gamma \vdash_\Sigma \mathbf{A} :: \mathbb{A}$ in $\Sigma^d\Lambda^{\rightarrow}$, then $\mathcal{I}_\varphi(\mathbf{A}) \in \mathcal{D}_{\mathbb{A}}$ for all Γ-assignments φ.*

Proof: We prove the assertion by induction over the structure of $\mathcal{A} \colon \Gamma \vdash_\Sigma \mathbf{A} :: \mathbb{A}$. The *ws:var* and *ws:td* cases are simple consequences of the assumptions that $\mathcal{S}$ is a Σ-PER structure and φ is a Γ-assigment.

For the *ws:app* case we have the following situation

$$\frac{\Gamma \vdash_\Sigma \mathbf{C} :: \mathbb{C} \quad \Gamma \vdash_\Sigma \mathbf{D} :: \mathfrak{d}(\mathbb{C})}{\Gamma \vdash_\Sigma \mathbf{CD} :: \mathfrak{r}(\mathbb{D})}$$

So by inductive hypothesis, we have $\mathcal{I}_\varphi(\mathbf{A}) \in \mathbf{Dom}(\mathcal{D}_{\mathbb{C}}) \subset \mathcal{D}_{\mathfrak{d}(\mathbb{C})} \longrightarrow \mathcal{D}_{\mathfrak{r}(\mathbb{D})}$[76] EdNote(76) and $\mathcal{I}_\varphi(\mathbf{D}) \in \mathbf{Dom}(\mathcal{D}_{\mathfrak{d}(\mathbb{D})})$. So $\mathcal{I}_\varphi(\mathbf{CD}) = \mathcal{I}_\varphi(\mathbf{C})@\mathcal{I}_\varphi(\mathbf{D}) \in \mathbf{Dom}(\mathcal{D}_{\mathfrak{r}(\mathbb{D})})$.

For the *ws^d:abs* case we have $\mathbf{A} = \lambda X.\mathbf{B}$ and $\mathcal{A}$ has the form

$$\frac{\Gamma, [X :: \mathbb{A}] \vdash_\Sigma \mathbf{B} :: \mathbb{B}}{\Gamma \vdash_\Sigma \lambda X.\mathbf{B} :: \mathbb{A} \to \mathbb{B}}$$

By inductive hypothesis we have $\mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{B}) \in \mathbf{Dom}(\mathcal{D}_{\mathbb{B}})$, so $\mathcal{I}_\varphi(\mathbf{A})@\mathsf{a} \in \mathbf{Dom}(\mathcal{D}_{\mathbb{A}})$ for all $\mathsf{a} \in \mathbf{Dom}(\mathcal{D}_{\mathbb{A}})$, so $\mathcal{I}_\varphi(\mathbf{A}) \in \mathbf{Dom}(\mathcal{D}_{\mathbb{A}\to\mathbb{B}})$ by Lemma 10.3.4.

The *ws:$\beta\eta$* case case is a simple consequence of the inductive hypothesis and the correctness of $\beta\eta$-equality in Σ-PER-structures[77]. □ EdNote(77)

EndNP(63)

10.4 Sorted Tableaux

[78] EdNote(78)

Example 10.4.1 (Smooth Non-Negative Functions) Now let us use the signature for elementary calculus from Example 7.7.22[79] and augment EdNote(79)

[76] EdNote: Is this better than the definition above?
[77] EdNote: Formulate this somewhere
[78] EdNote: state the calculus
[79] EdNote: Check reference

real numbers
numbers!real

it with the predicates smooth for arbitrary differentiability on the real numbers and $>$. Say we want to prove that there are smooth non-negative functions from the knowledge that polynomials are smooth, then we have the tableau in Figure 10.1. In contrast to example 8.4.12 the solution is a

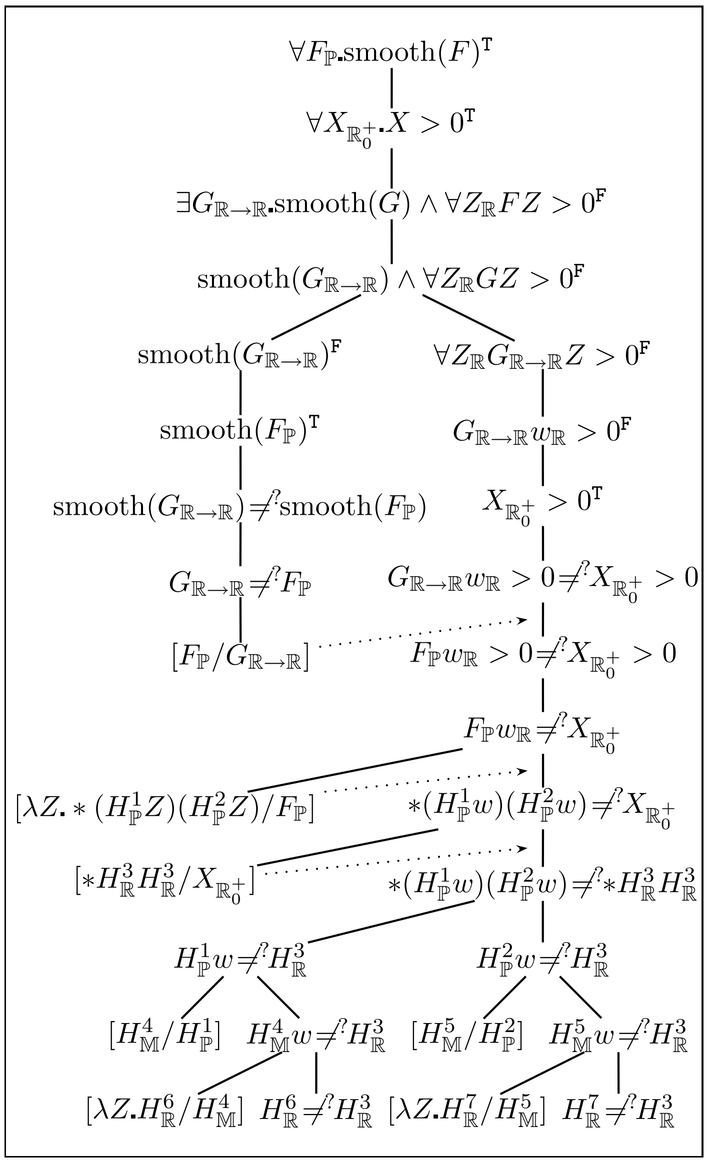


Figure 10.1: Smooth Non-negative Functions

$[\lambda Z. H^7 * H^7]$, which is a constant function with non-negative value (it is a square).

monomial *polynomial* *degree* *non-negative*

The completeness and soundness proofs in the sorted setting are trivial generalizations of the unsorted ones. The only only place, where there is a difference to the unsorted setting is in the proof of the lifting lemma for $\mathcal{RC}$. Since general bindings in $\Sigma\mathcal{HOL}$ (coming from term declarations with logical constants) can have more than one occurrence of logical constants, the proof of the lifting lemma cannot directly follow the structure of θ as it was possible in the unsorted case[80]. EdNote(80)

10.5 $\Sigma\mathcal{HR}$: Resolution for $\Sigma\mathcal{HOL}$

Corollary 10.5.1 (Completeness of $\Sigma\mathcal{HR}$)
Let Φ be a finite set of well-sorted sentences. If Φ is unsatisfiable in the class of Σ-model structures, then $\Phi \vdash_{\Sigma\mathcal{HR}} \Box$.

Proof: Let $\nabla_\Sigma(\Gamma) := \{\Psi \subset \mathit{wsf}_{\mathbb{O}}(\Sigma, \Gamma) \mid \Psi \not\vdash_{\Sigma\mathcal{HR}} \Box\}$. If Φ were not $\Sigma\mathcal{HR}$-refutable, then $\Phi \in \Gamma_\Sigma(\emptyset)$ by construction. Furthermore, by 6.5.5 Γ_Σ is a saturated abstract consistency class and therefore by **??** there is a Σ-model structure $\mathcal{M} \models \Phi$, which contradicts the assumption. $\Box$

We conclude our exposition of $\Sigma\mathcal{HR}$ with an example.

Example 10.5.2 Let Σ be the signature of 8.4.12 augmented by the sort $\mathbb{N}$ of natural numbers and the declarations

$$[0{::}\mathbf{N}], [s{::}\mathbb{N} \to \mathbb{N}], [+{::}\mathbb{N} \to \mathbb{N} \to \mathbb{N}], [>{::}\mathbb{R} \to \mathbb{R} \to \mathbb{O}],$$
$$[\deg{::}\mathbb{M} \to \mathbb{N}], [p{::}(\mathbb{R} \to \mathbb{R}) \to \mathbb{O}], [nc{::}(\mathbb{R} \to \mathbb{R}) \to \mathbb{O}]$$

We consider the following axiomatization of the degree deg of monomials

D1 $\forall Y_{\mathbb{R}}.\, \deg(\lambda X_{\mathbb{R}}.Y) = 0$

D2 $\deg(\lambda X_{\mathbb{R}}.X) = s(0)$

D3 $\forall F_{\mathbb{M}}, G_{\mathbb{M}}.\, \deg(\lambda X_{\mathbb{R}}.\, * (FX)(GX)) = + \deg(F) \deg(G))$

and the following basic facts about $>$, non-constant (nc), and positive functions (p):

$>$ $\forall X_{\mathbb{N}}.s(X) > 0$

N $\forall F_{\mathbb{M}}.(\deg(F) > 0) \Rightarrow nc(F)$

P $\forall F_{\mathbb{R}\to\mathbb{R}^+}.p(F)$

$+$ $\forall X_{\mathbb{N}}, Y_{\mathbb{N}}.(X > 0 \wedge Y > 0) \Rightarrow X + Y > 0$

[80] EdNote: check, continue, do an example

From these we want to prove the assertion that there is a differentiable function that is positive, but not constant:

$$\exists Y_{\mathbb{D}}.p(Y) \wedge nc(Y)$$

The clause normal form of its negation is

T $\langle [Y :: \mathbb{D}] : \emptyset \rangle . p(Y)^{\mathtt{F}} \vee nc(Y)^{\mathtt{F}}$

from which we can obtain

1=R(T,N) $\langle [Z :: \mathbb{M}], [Y :: \mathbb{D}] : \emptyset \rangle . p(Y)^{\mathtt{F}} \vee (\deg(Z) > 0)^{\mathtt{F}} \vee Y \neq^? Z$
2=R(1,P) $\langle [Z :: \mathbb{M}], [W :: \mathbb{R} \to \mathbb{R}^+] : \emptyset \rangle . (\deg(Z) > 0)^{\mathtt{F}} \vee Z \neq^? W \vee Y \neq^? Z$

by $\Sigma\mathcal{HR}(Res)$. Since all variable conditions in this example are empty, we drop the declarations from the clauses, and indicate the sort of variables by indices. With the $\Sigma\mathcal{PT}$-derivation (cf 9.2.10) of 8.4.12 and $\Sigma\mathcal{HR}(Solv)$ we obtain

3=U(2) $(\deg(Z_{\mathbb{M}}) > 0)^{\mathtt{F}} \vee Z \neq^? (\lambda X_{\mathbb{R}} . * (H^5_{\mathbb{M}} X)(H^5_{\mathbb{M}} X))$
4=S(3) $(\deg(\lambda X_{\mathbb{R}} . * (H^5_{\mathbb{M}} X)(H^5_{\mathbb{M}} X)) > 0)^{\mathtt{F}}$

If we instantiate Leibniz definition 1.2.6 for $=^{\mathbb{N}}$ in $D3$, then the clause normal form is

$$P_{\mathbb{N}\to\mathbb{O}}(\deg(\lambda X_{\mathbb{R}} . * (F_{\mathbb{M}} X)(G_{\mathbb{M}} X)))^{\mathtt{T}} \vee P(\deg(F) + \deg(G))^{\mathtt{F}}$$

thus we have

5=R(4,G3) $P_{\mathbb{N}\to\mathbb{O}}(\deg(F_{\mathbb{M}}) + \deg(G_{\mathbb{M}}))^{\mathtt{F}} \vee$
$(\deg(\lambda X_{\mathbb{R}} . * (H^5_{\mathbb{M}} X)(H^5_{\mathbb{M}} X)) > 0)^{\mathtt{F}} \neq^? P(\deg(\lambda X_{\mathbb{R}} . * (F_{\mathbb{M}} X)(G_{\mathbb{M}} X)))$

With the general bindings

$$\begin{array}{lll} P_{\mathbb{N}\to\mathbb{O}} & =^? & \lambda X_{\mathbb{N}} . > (H^1_{\mathbb{N}\to\mathbb{N}} X)(H^2_{\mathbb{N}\to\mathbb{N}} X) \\ H^1_{\mathbb{N}\to\mathbb{N}} & =^? & (\lambda X_{\mathbb{N}} . X) \\ H^2_{\mathbb{N}\to\mathbb{N}} & =^? & (\lambda X_{\mathbb{N}} . 0) \end{array}$$

and reduction to $\mathcal{SIM}$-normal form we obtain

6=U(5) $(\deg(F_{\mathbb{M}}) + \deg(F_{\mathbb{M}}) > 0)^{\mathtt{F}} \vee H^5_{\mathbb{M}} =^? F_{\mathbb{M}} \vee H^5_{\mathbb{M}} =^? G_{\mathbb{M}}$
7=S(6) $(\deg(H^5_{\mathbb{M}}) + \deg(H^5_{\mathbb{M}}))^{\mathtt{F}}$
8=R(7,+) $\deg(H^5_{\mathbb{M}}) > 0^{\mathtt{F}}$

On the other hand the clause normal form of $D2$ is

D2 $Q_{\mathbb{N}\to\mathbb{O}}(\deg(\lambda X_{\mathbb{R}} . X))^{\mathtt{F}} \vee Q(s(0))^{\mathtt{T}}$

thus by primitive substitution (for the head $\neg$), $\mathcal{HR}(E)$ and resolution we have

9=P(D2) $Q_{\mathbb{N}\to\mathbb{O}}(\deg(\lambda X_{\mathbb{R}} . X))^{\mathtt{F}} \vee Q(s(0))^{\mathtt{T}} \vee Q \neq^? (\lambda X_{\mathbb{N}} . \neg(P_{\mathbb{N}\to\mathbb{O}} X))$
10=S(9) $P_{\mathbb{N}\to\mathbb{O}}(\deg(\lambda X_{\mathbb{R}} . X))^{\mathtt{T}} \vee P(s(0))^{\mathtt{F}}$
11=R(>,10) $Q_{\mathbb{N}\to\mathbb{O}}(\deg(\lambda X_{\mathbb{R}} . X))^{\mathtt{T}} \vee Q(s(0)) \neq^? s(X_{\mathbb{N}}) > 0$

With $X =^? 0$ and general bindings for P analogous to those above we obtain

description functions
$\iota_{(\alpha\to o)\to\alpha}$ *description function*

12=U(11) $\deg(\lambda X_{\mathbb{R}}.X) > 0^{\mathrm{T}}$
13=R(12,8) $H^5 = (\lambda X_{\mathbb{R}}.X)$

which is an empty clause, since the remaining pair is solved as the identity function is a monomial. □

The example for a differentiable function that is not constant (the instance for $Z_{\mathbb{D}}$) found in the proof is the square function $\lambda X_{\mathbb{R}}. * XX$, which is indeed the simplest such function in our simple taxonomy of functions.

10.5.1 Resolution with dynamic Sorts

In this section, we will generalize[81] EdNote(81)

10.5.2 Sorted Logics Theorem Proving: An A-Posteriori View

To get a better intuition of the improvements that $\Sigma\mathcal{HOL}$ has to offer for specifying mathematical theorems and proving them, let us have a closer look at the alternative of using an unsorted higher-order logic. With the relativization technique well-known from first-order logic a well-sorted formulae $\mathbf{A}$ can be coded into an unsorted formula $\mathbf{Rel}(\mathbf{A})$. For instance, a sorted quantification $\forall X_{\mathbb{A}}.\mathbf{A}$ is transformed into $\forall X_{\tau(\mathbb{A})}.(\mathbf{P}_{\mathbb{A}} X) \Rightarrow \mathbf{Rel}(\mathbf{A})$ for some new predicate $\mathbf{P}_{\mathbb{A}}$ that captures the sort information. These new predicates obtain their meaning from an axiomatization $\mathbf{Rel}(\Sigma)$ of the sort information present in a valid signature Σ, which is provided by the relativization operation. For most sorted first-order logics it is a theorem (see [Wal87, SS89, Wei91] for examples) that a well-sorted formula $\mathbf{A}$ is satisfiable in the class of sorted models, iff $\mathbf{Rel}(\mathbf{A})$ is satisfiable in the class of unsorted models that satisfy the signature axioms $\mathbf{Rel}(\Sigma)$. Thus from a theoretical point of view these sorted first-order logics are not more expressive than unsorted ones. Indeed we conjecture that some kind of sort theorem also holds for $\Sigma\mathcal{HOL}$. This would entail that in theory all theorems of $\Sigma\mathcal{HOL}$ can be proven by coding them into simple type theory and then proving them by simply typed constrained resolution.

Unfortunately, in $\Sigma\mathcal{HOL}$ the situation is not as simple as in the first-order case, where the only binding constructs are quantifications that can be transformed as shown above. In the presence of λ-abstractions we need some form of conditionals in the target system for coding functional formulae, such as $\lambda X_{\mathbb{A}}.\mathbf{A}$, which have to be relativized as partial functions. Conditionals can be realized by description functions (see 2.1.29) as in [Chu40]. Such a

[81] EdNote: continue a

relativization
Rel*relativization operator*
signature axioms

system can be obtained from $\Sigma\mathcal{HOL}$ by adding a logical constant $\iota_{(\alpha\to o)\to\alpha}$ for each $\alpha \in \mathcal{T}$ and the inference rule

$$\frac{\Gamma \vdash_{\Sigma} \mathbf{Q}{::}\mathbb{A} \to \mathbb{O} \quad \Gamma \Vdash_{\Sigma}^{\iota} \forall X_{\mathbb{A}}.\mathbf{Q}X \Rightarrow .\forall Y_{\mathbb{A}}.(\mathbf{Q}Y \Rightarrow X = Y)}{\Gamma \Vdash_{\Sigma}^{\iota} \mathbf{Q}(\iota\mathbf{Q})}\ \Sigma\mathfrak{T}(\iota)$$

to $\Sigma\mathfrak{T}_{\eta\mathfrak{b}}$. Moreover, in the definition of general Σ-models we have to specify that the value of ι is the function that maps singleton sets to their unique member. In this setting we can define a conditional $\mathbf{w}_{\alpha\to o\to\alpha}$ as implication, if $\alpha = o$, and otherwise as $(\lambda X_\alpha, P_o.\iota_{(\alpha\to o)\to\alpha}(\lambda Y_\alpha.P \wedge Y = X))$. It is easy to see that for any general Σ-model $\mathcal{M} = (\mathcal{D}, \mathcal{I})$ we have $\mathcal{I}_\varphi(\mathbf{w})(\mathsf{a}, \mathtt{T}) = a$.

With these extensions we can now define a relativization operator **Rel**, and use it to compare the relativization of an example with the sorted version. This comparison will give us a feeling for the advantages of sorts in higher-order deduction.

Definition 10.5.3 (Relativization)
Rel is a typed mapping $\mathbf{Rel}\colon \mathcal{S} \to \mathit{wff}(\overline{\Sigma} \cup \mathbf{P}_{\mathcal{S}})$, where $\mathbf{P}_{\mathcal{S}} := \{\mathbf{P}_{\mathbb{A}} \mid \mathbb{A} \in \mathcal{S}\}$ is a set of new predicates of type $\tau(\mathbb{A}) \to o$. We now inductively define **Rel** by setting

1. $\mathbf{Rel}(\mathbb{A}) := \mathbf{P}_{\mathbb{A}}$ for $\mathbb{A} \in \mathcal{S}^{nf} \cap \mathcal{BS}$,
2. $\mathbf{Rel}(\mathbb{A} \to \mathbb{B}) := (\lambda F_{\tau(\mathbb{A})\to\tau(\mathbb{B})}.\forall X_{\tau(\mathbb{A})}.(\mathbf{Rel}(\mathbb{A})X) \Rightarrow .\mathbf{Rel}(\mathbb{B})(FX))$,
3. $\mathbf{Rel}(\mathbb{A}) := \lambda X_{\tau(\mathbb{A})}.\mathbf{P}_{\mathbb{A}} \wedge \mathbf{Rel}(\mathfrak{d}(\mathbb{A}) \to \mathfrak{r}(\mathbb{A}))X$ for $\mathbb{A} \in \mathcal{S}_0^f$,

This relativization of sorts allows us to define the full relativization operator on well-sorted formulae, which is a typed function $\mathbf{Rel}\colon \mathit{wsf}(\Sigma, \Gamma) \longrightarrow \mathit{wff}(\overline{\Sigma} \cup \mathbf{P}_{\mathcal{S}})$ such that

4. $\mathbf{Rel}(\mathbf{A}) := \mathbf{A}$, if $\mathbf{A}$ is a constant or variable,
5. $\mathbf{Rel}(\mathbf{AB}) := \mathbf{Rel}(\mathbf{A})\mathbf{Rel}(\mathbf{B})$,
6. $\mathbf{Rel}(\lambda X_{\mathbb{B}}.\mathbf{A}) := (\lambda \overline{X}.\mathbf{w}\mathbf{Rel}(\mathbb{B})\mathbf{A})$.

Finally, we can map valid signatures Σ into sets $\mathbf{Rel}(\Sigma)$ of sentences, called **signature axioms**, by defining

7. $\mathbf{Rel}(\Sigma, [\mathbf{A}{::}\mathbb{A}]) := \mathbf{Rel}(\Sigma) \wedge (\mathbf{Rel}(\mathbb{A})\mathbf{Rel}(\mathbf{A}))$

For instance, in the case of a subsort declaration $[\mathbb{A} \leq \mathbb{B}] \in \Sigma$ we have

$$\begin{aligned}\mathbf{Rel}([\mathbb{A} \leq \mathbb{B}]) &= \mathbf{Rel}([\lambda X.X{::}\mathbb{A} \to \mathbb{B}]) \\ &= \forall X_{\tau(\mathbb{A})}.(\mathbf{Rel}(\mathbb{A})X) \Rightarrow .\mathbf{Rel}(\mathbb{B})X\end{aligned}$$

as we would have expected from the analogy with first-order sorted logics. In particular, if $\mathbb{A}$ and $\mathbb{B}$ are non-functional sorts, then the sentence $\forall X.(\mathbf{P}_{\mathbb{A}} X) \Rightarrow .\mathbf{P}_{\mathbb{B}} X$ just amounts to the subset definition. To enhance our intuition on relativizations of signatures, let us compare the sorted formulation of the signature in example 8.4.12 with its relativization in the following signature axioms:

$$\begin{array}{rcl}
\mathbf{Rel}([\partial{::}\mathbb{D} \to \mathbb{C}]) & = & \forall F.(\mathbf{P}_{\mathbb{D}} F \wedge \forall Y.\mathbf{P}_{\mathbb{R}} Y \Rightarrow \mathbf{P}_{\mathbb{R}}(FY)) \Rightarrow .\mathbf{P}_{\mathbb{C}}(\partial F) \\
& & \quad \wedge(\forall Y.\mathbf{P}_{\mathbb{R}} Y \Rightarrow .\mathbf{P}_{\mathbb{R}}(\partial F)Y) \\
\mathbf{Rel}([\partial{::}\mathbb{P} \to \mathbb{P}]) & = & \forall F.(\mathbf{P}_{\mathbb{P}} F \wedge \forall Y.\mathbf{P}_{\mathbb{R}} Y \Rightarrow \mathbf{P}_{\mathbb{R}}(FY))) \Rightarrow .(\mathbf{P}_{\mathbb{P}}(\partial F)) \\
& & \quad \wedge(\forall Y.\mathbf{P}_{\mathbb{R}} Y \Rightarrow .\mathbf{P}_{\mathbb{R}}(\partial F)Y) \\
\mathbf{Rel}([\lambda X_{\mathbb{R}}.X{::}\mathbb{P}]) & = & \mathbf{P}_{\mathbb{P}}(\lambda X.\mathbf{w}(\mathbf{P}_{\mathbb{R}} XX)) \wedge \forall Y.(\mathbf{P}_{\mathbb{R}} Y) \Rightarrow (\mathbf{P}_{\mathbb{R}}(\mathbf{w}(\mathbf{P}_{\mathbb{R}} X)Y)) \\
\mathbf{Rel}([\lambda Z, X.Z{::}\mathbb{R} \to \mathbb{R} \to \mathbb{P}]) & = & \forall Z.(\mathbf{P}_{\mathbb{R}} Z \Rightarrow \mathbf{P}_{\mathbb{P}}(\lambda X.\mathbf{w}(\mathbf{P}_{\mathbb{R}} XZ))\forall Y.(\mathbf{P}_{\mathbb{R}} Y)) \Rightarrow (\mathbf{P}_{\mathbb{R}}(\mathbf{w}(\mathbf{P}_{\mathbb{R}} X)Z)))
\end{array}$$

$$\begin{array}{l}
\mathbf{Rel}([\lambda F, G.\lambda X. + (FY)(GY){::}\mathbb{P} \to \mathbb{P} \to \mathbb{P}]) \\
\quad = \forall F, G.(\mathbf{P}_{\mathbb{P}} F) \Rightarrow .(\mathbf{P}_{\mathbb{P}} G) \Rightarrow \\
\qquad \mathbf{P}_{\mathbb{P}}(\lambda X.\mathbf{w}(\mathbf{P}_{\mathbb{R}} X(+(FX)(GX))))\forall Y.(\mathbf{P}_{\mathbb{R}} Y) \Rightarrow (\mathbf{P}_{\mathbb{R}}(\mathbf{w}(\mathbf{P}_{\mathbb{R}} Y) + (FY)(GY))) \\
\mathbf{Rel}([\lambda F, G.\lambda X. * (FY)(GY){::}\mathbb{P} \to \mathbb{P} \to \mathbb{P}]) \\
\quad = \forall F, G.(\mathbf{P}_{\mathbb{P}} F) \Rightarrow .(\mathbf{P}_{\mathbb{P}} G) \Rightarrow \\
\qquad \mathbf{P}_{\mathbb{P}}(\lambda X.\mathbf{w}(\mathbf{P}_{\mathbb{R}} X(*(FX)(GX))))\forall Y.(\mathbf{P}_{\mathbb{R}} Y) \Rightarrow (\mathbf{P}_{\mathbb{R}}(\mathbf{w}(\mathbf{P}_{\mathbb{R}} Y) * (FY)(GY)))
\end{array}$$

This set of axioms has to be added to the relativization of any $\Sigma\mathcal{HOL}$-sentence that we want to prove in the relativized form. A further effect, which we have not illustrated for lack of space, is that the unification in the sorted setting finds out conflicting taxonomic information for proof objects, and prevent any inference that would yield ill-sorted objects. These objects arise naturally in the relativized setting, but due to $\mathbf{Rel}(\Sigma)$ they can never contribute to any proof. Thus resolution in $\Sigma\mathcal{HOL}$ gives us the further advantage of cutting off enormous redundant branches in the search space. As a consequence the search spaces are so much smaller that the sorted calculus is clearly practically superior.

In an a posteriori view we can see the generalization of resolution to a sorted setting as the process of building certain classes of axioms, namely, those that correspond to term declarations, into the unification. This process takes axioms of the form

$$(\forall \overline{X^k}.p^1 X^1 \Rightarrow \ldots \Rightarrow p^k X^k \Rightarrow (q\mathbf{A}))$$

where the p^i and q are unary predicate constants and X^i are the free variables of $\mathbf{A}$ out of the search, and treats them algorithmically in the unification.

Part IV

Representing Mathematical Knowledge

In this part we propose a methodology for developing representation formalisms for mathematical knowledge bases. We propose to equip knowledge bases with a hierarchy of logical systems that are linked by *logic morphisms*. These mappings relativize formulae and proofs and thus support translation of the knowledge to the various formats currently in use in automated deduction. On the other hand they define higher language features from simpler ones and ultimately serve as a means to found the whole knowledge base in axiomatic set theory.

The viability of this approach is proven by developing a sorted record-λ-calculus with dependent sorts and labeled abstraction that is well-suited both for formalizing mathematical practice and supporting efficient inference services. This "mathematical vernacular" is an extension of the sorted λ-calculus from [Koh94] by records, dependent record sorts and selection sorts.

Mathematical Knowledge Bases

Around 1995, an anonymous group of authors put forward the “QED Manifesto” [QED95], which advocates building up a mathematical knowledge base (and supporting software systems) as a kind of “human genome project” for the deduction community (see `http://www.mcs.anl.gov/qed` for an overview of the QED activities). Unfortunately, the vision has failed to catch on in spite of a wave of initial interest. In my view this is due to the lack of supporting software[1], as well as well as to the ensuing debate on the “right” logical formalism. We will not concentrate on the motivation or structure of a knowledge base system[2], but we will try to argue how we can circumnavigate the language problems of QED.

Set theorists argue that axiomatic set theory is established mathematical practice [Qua92, Pau93], while type theorists claim that the various type systems model informal structuring and notational principles used by mathematicians. In this book, we argue that we should take this debate seriously, try to accommodate for all these notational and logical styles. We propose to establish a framework for interoperability based on *relativization* and *conservative extension*, maintaining the necessary structuring mechanisms necessary for efficient mechanical treatment of mathematics.

Concretely, we propose to use a hierarchy of representation languages that are linked by *logic morphisms* for formalizing mathematics. These mappings relativize formulae and proofs and thus support translation of the knowledge to the various formats currently in use in automated deduction. On the other hand they define higher-level level language features from simpler ones by “pulling back” the semantics, and ultimately serve as a means to found the whole knowledge base in axiomatic set theory (see [Ebb77] for an introduction).

This approach closely follows mathematical theory that views axiomatic set theory as a foundation of all mathematics and resolves all foundational issues by relativization to this system. On the other hand, for practical work mathematicians use highly specialized mathematical languages and structure the available knowledge into subfields, areas and theories; all this representational machinery is built as a series of abbreviations and definitions on the foundational set theory. In the rest of this chapter, we will proceed in the same way. First, we will present the methodology of logic morphisms, and then sketch how this can be used to build up the simply typed λ-calculus. Then, we will use sections 12.1 to 12.1.4 to build a logical system MV, that we claim is well-suited for formalizing mathematical

---

[1]The software issue is outside the scope of this book; we are currently working on a distributed, Internet-based mathematical knowledge base system that is based on the logical issues discussed here.

[2]This has been more thoroughly and more elaborately in the QED Manifesto [QED95], which we presuppose as a background reference to the discussion.

practice.

relativization
Logical System
logical language
calculus
derivation
assertion
hypotheses
proof
theorem
logic morphism
language morphism
calculus morphism

Chapter 11

Logic Morphisms and Relativization

We will use a variant of the the theory interpretation approach proposed in [Far93] for **relativization** mappings, that can be used to transport meanings and proofs between logical formalisms. The fundamental logical notions for this are logical systems and logic morphisms. For the purposes of this book, we will call a pair $\mathcal{S} = (\mathcal{L}, \mathcal{C})$ **Logical System**. if $\mathcal{L}$ is a **logical language** (set of well-formed formulae) and $\mathcal{C}$ is a **calculus** (a set of inference rules defined in usual way as n-ary relations over well-formed formulae).

Given a logical system $\mathcal{S} = (\mathcal{L}, \mathcal{C})$, we define an $\mathcal{S}$-**derivation** $\mathcal{D}$ of an **assertion** $\mathbf{A}$ from a set $\mathcal{H}$ of **hypotheses**s (written $\mathcal{D}{:}\, \mathcal{H} \vdash_{\mathcal{S}} \mathbf{A}$) as a tree $\mathcal{D}$ (or a directed acyclic graph), where the leaves of $\mathcal{D}$ are labelled with the formulae from $\mathcal{H}$ and the root is labeled with $\mathbf{A}$. Furthermore, all nodes of $\mathcal{D}$ are labeled by assertions $\mathbf{C}$ and inference rules $\mathcal{R} \in \mathcal{C}$, such that the for the labels $\mathbf{H}_1, \ldots, \mathbf{H}_n$ of the daughters of a node we have $\mathcal{R}(\mathbf{H}_1, \ldots, \mathbf{H}_n, \mathbf{C})$.

We will use the terms like $\mathcal{S}$-**proof** (for a derivation of an assertion $\mathbf{A}$ from the empty set of hypotheses) and $\mathcal{S}$-**theorem** (for an assertion for which there is a $\mathcal{S}$-proof) in the usual way.

A **logic morphism** $\mathcal{F}{:}\, \mathcal{S} \longrightarrow \mathcal{S}'$, is a pair $(\mathcal{F}^{\mathcal{L}}, \mathcal{F}^{\mathcal{C}})$ of mappings, such that $\mathcal{F}^{\mathcal{L}}{:}\, \mathcal{L} \longrightarrow \mathcal{L}'$ and $\mathcal{F}^{\mathcal{C}}$ maps $\mathcal{S}$-derivations to $\mathcal{S}'$-derivations, such that for any $\mathcal{S}$-derivation $\mathcal{D}{:}\, \mathcal{H} \vdash_{\mathcal{S}} \mathbf{A}$, we have $\mathcal{F}^{\mathcal{C}}(\mathcal{D}){:}\, \mathcal{F}^{\mathcal{L}}(\mathcal{H}) \vdash_{\mathcal{S}'} \mathcal{F}^{\mathcal{L}}(\mathbf{A})$. We call $\mathcal{F}^{\mathcal{L}}$ the **language morphism** and $\mathcal{F}^{\mathcal{C}}$ the **calculus morphism** of $\mathcal{F}$.

Note that we do not insist that a logical morphism is a homomorphism of the term algebras or proof structures, even if it will be in most examples covered below. Since we have purposely left the term "set of well-formed formulae" vague, the concept of a homomorphism would be difficult to define and would restrict the generality of the approach. The defining intuition for logic morphisms is that

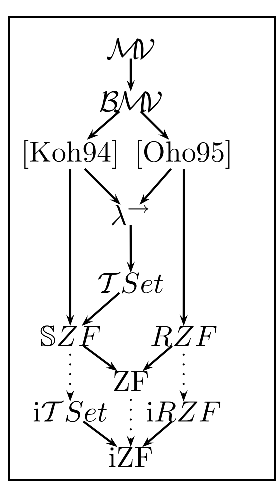


Figure 11.1: Hierarchy

redundant
MBase@MBase
MBase@MBase
MBase@MBase

Logic Morphisms Transport Proofs: Let $\mathcal{F}: \mathcal{S} \longrightarrow \mathcal{S}'$ be a logic morphism and $\mathbf{A}$ an $\mathcal{S}$-theorem, then $\mathcal{F}(\mathbf{A})$ is an $\mathcal{S}'$-theorem.

This already suggests the logical structure of a mathematical knowledge base: Orthogonal to the usual theory hierarchy (induced by theory interpretation morphisms; we will not go into in this book, see [Far93]), there is a hierarchy of logical system induced by logic morphisms. In Figure 11.1, we have specified some of the logical systems we will discuss in this article.

Mathematical knowledge can be specified in any of the logical systems; it can be queried and retrieved in any logical system that is downward accessible from this one. Furthermore, communication of mathematical software systems is possible by way of the "least common denominator logic". This may seem as a severe restriction of applicability of the approach, but it is not since the set of logical systems and morphisms in the hierarchy is not necessarily fixed:

- A new logical system can be incorporated by specifying a logic morphism to any of the existing systems.
- A new logic morphism can be added, if it is consistent with the information already present in the structure, i.e. if it is *redundant*.

Of course these hierarchy extensions generate proof obligations (determining the logic morphism property and redundancy), which will have to be supported in a system like MBase. We leave a discussion of this to another article.

The practical usefulness of a language hierarchy will depend very much on the existence of such redundant morphisms. In particular for the "least-common-denominator" problem between languages $\mathcal{L}$ and $\mathcal{L}'$ we can have two kinds of situations:

- If there is a good and well-understood way to translate formulae from language $\mathcal{L}$ to $\mathcal{L}'$, then we can implement this as a redundant logic morphism in MBase bypassing the need of an intermediate "communication logic". Moreover, making the logic morphism available in MBase will allow other users to use it.
- If there is no such translation, or if it is very domain-specific, then (of course) logic morphisms will not help (only further research into the semantic relation between the logics and possible translations will).

In the rest of this section, we will make the relativization approach concrete. We will first look at the elimination of sorts from sorted first-order logic by relativization. Based on the this guiding example, we we discuss the logical foundations and the relation to set-theoretic semantics in section **??**. We will conclude this section by a discussion of the relativization of higher-order logic into first-order logic, in order to complete the lower half of the diagram in figure 11.1, before turning to the upper half in section 12.1.

11.1 Relativization

In the rest of this section, we will look at the example of the relativization from sorted first-order logic to classical first-order logic. To make the intuitions clear. We will use the simply typed λ-calculus [HS86] as a metalogical framework for representing the logical systems, since it gives us substitution, replacement and the treatment of bound variables for free. This is only a notational convenience and of no fundamental importance. In particular, this does not make any prerequisites on part of the logical systems like first-order logic presented in this chapter.

11.1.1 Example: Relativizing Sorted First-Order Logic

The first logic morphism we want to consider is that from first-order logic FOL to sorted first-order logic $SFOL$. $FOL = (\mathcal{L}^{FOL}, ND(FOL))$ is the

FOL	$= (\mathcal{L}^{FOL}, ND(FOL))$		
Signature	ι	Type	Individuals
	o	Type	Truth Values
	$\wedge$	$o \rightarrow o \rightarrow o$	Conjunction
	$\neg$	$o \rightarrow o$	Negation
	Π^{ι}	$(\iota \rightarrow o) \rightarrow o$	Universal Quantification
$\mathcal{L}^{FOL}$	= well-typed formulae of type o		
ND(FOL)	$\dfrac{\begin{matrix}\vdots\\ \Vdash^{\iota}_{\Sigma} \Pi^{\iota}\mathbf{B}\end{matrix}}{\Vdash^{\iota}_{\Sigma} \mathbf{B}\mathbf{A}} \Vdash^{\iota}_{\Sigma} \Pi^{\iota}E$	$\dfrac{\begin{matrix}\vdots\\ \Vdash^{\iota}_{\Sigma} \mathbf{A}X\end{matrix}}{\Vdash^{\iota}_{\Sigma} \Pi^{\iota}\mathbf{A}} \Vdash^{\iota}_{\Sigma} \Pi^{\iota}I$	

Figure 11.2: First-Order Logic

logical system, where the logical part of the signature consists of the type constants o and ι (for truth values and individuals) and the term constants $\wedge, \neg$, and Π (see Figure 11.2, all other connectives can be defined from $\neg$ and $\wedge$, by De Morgan rules, and quantification can be regained by treating $\forall X.\mathbf{A}$ as an abbreviation of $\Pi(\lambda X.\mathbf{A})$). The signature of first-order logic parameter

can contain further non-logical constants (called **parameter**s) that model mathematical structures. In the following, all arguments and constructions will be parametric in the choice of parameters in the signature, and we will use the more precise $FOL(\Sigma)$ for the instance of FOL that contains the parameters declared in the signature Σ.

$\mathcal{C} = ND(FOL)$ is the well-known calculus of natural deduction introduced by Gerhard Gentzen in [Gen35]. We will use $\Vdash^{\iota}_{\Sigma}$ to abbreviate $\vdash_{ND(FOL)}$ (in Figure 11.2, we have only depicted the quantifier rules, since they will be the only interesting ones for the discussion in this chapter).

$\mathbb{S}$FOL	$= (\mathcal{L}^{\mathbb{S}}\text{FOL}, ND(\mathbb{S}\text{FOL})) = FOL+$		
Signature Σ	$\mathbb{A}, \mathbb{B}, \ldots$	$\iota \to o$	Sorts $\widehat{=}$ Subsets of individuals
	$[\mathbb{A} \leq \mathbb{B}]$		Subsort declarations
	$[c{::}\mathbb{A}]$		Constant declarations
	Π^*	$(\iota \to o)^2 \to o$	Sorted Universal Quantification
$\mathcal{L}^{\mathbb{S}}$FOL	Well-formed = well-typed		
Subsorting	$\dfrac{[\mathbb{A} \leq \mathbb{B}] \in \Sigma}{\vdash^s_\Sigma \mathbb{A} \leq \mathbb{B}}$	$\dfrac{}{\vdash^s_\Sigma \mathbb{A} \leq \mathbb{A}}$	$\dfrac{\vdash^s_\Sigma \mathbb{A} \leq \mathbb{B} \quad \vdash^s_\Sigma \mathbb{B} \leq \mathbb{C}}{\vdash^s_\Sigma \mathbb{A} \leq \mathbb{C}}$
Well-sorted	$\dfrac{[c{::}\mathbb{A}] \in \Sigma}{\Gamma \vdash^s_\Sigma c{::}\mathbb{A}}$	$\dfrac{}{\Gamma, [X{::}\mathbb{A}] \vdash^s_\Sigma X{::}\mathbb{A}}$	$\dfrac{\Gamma \vdash^s_\Sigma \mathbf{A}{::}\mathbb{B} \to \mathbb{C} \quad \Gamma \vdash^s_\Sigma \mathbf{B}{::}\mathbb{B}}{\Gamma \vdash^s_\Sigma \mathbf{AB}{::}\mathbb{C}}$
$ND(\mathbb{S}\text{FOL})$	$\ldots$	$\dfrac{\Gamma \Vdash^{\iota}_\Sigma \Pi^*\mathbb{A}\mathbf{B} \quad \Gamma \vdash^s_\Sigma \mathbf{A}{::}\mathbb{A}}{\Gamma \Vdash^{\iota}_\Sigma \mathbf{BA}}$	$\dfrac{\Gamma, [X{::}\mathbb{A}] \Vdash^{\iota}_\Sigma \mathbf{A}X}{\Gamma \Vdash^{\iota}_\Sigma \Pi^*\mathbb{A}\mathbf{A}}$

Figure 11.3: Sorted First-Order Logic

The logical system $\mathbb{S}$FOL (see Figure 11.3) is an extension of FOL, where the signature is extended by an order-sorted set $\mathbb{S}$ of sorts, a sorted quantifier Π^* and a set of constant- and subsort declarations (again, we will make use of higher-order abstract syntax here and write the traditional $\forall X_{\mathbb{A}}.\mathbf{B}$ as $\Pi^*\mathbb{A}(\lambda X.t\mathbf{B})$.). The language $\mathcal{L}^{\mathbb{S}}$FOL is the set of well-sorted formulae, i.e. formulae, where for all applications $f(a)$ the argument a has a sort that is an argument sort of the function f. We specify this by the sort judgment $\Gamma \vdash^s_\Sigma \mathbf{A}{::}\mathbb{A}$ ($\mathbf{A}$ has sort $\mathbb{A}$ under the sort assumptions for the variables in $\mathbf{A}$ given in the variable context Γ). The subsort relation and the property of being well-sorted are given by the judgments $\vdash^s_\Sigma \mathbb{A} \leq \mathbb{B}$ and $\Gamma \vdash^s_\Sigma \mathbf{A}{::}\mathbb{A}$, which are proven by the subsorting and well-sortedness sub-calculi of $ND(\mathbb{S}\text{FOL})$. We will use $\Vdash^s_\Sigma$ for the propositional part of $ND(\mathbb{S}\text{FOL})$.

The logical morphism $\mathcal{R}_{\mathcal{S}}$ from $\mathbb{S}$FOL to FOL interprets the sorts in $\mathcal{S}$ as unary predicates (parameters of type $\iota \to o$) in FOL. Note that with the definitions in Figure 11.4, the universal $\forall X_{\mathbb{A}}.\mathbf{A}$ in $\mathcal{L}^{\mathbb{S}}\text{FOL}(\Sigma)$ is relativized to the $FOL(\overline{\Sigma})$-formula $\forall X.\mathbb{A}(X) \Rightarrow \mathbf{A}$ (if $\mathbb{A}$ is a base sort). This is just the well-known relativization morphism for sorted first-order

signature axiom
$\mathcal{R}_{\mathcal{S}}(\Sigma)$*signature axioms*
sort assumption
$\mathcal{R}_{\mathcal{S}}(\Gamma)$*sort assumptions*

logics. Function sorts are relativized into first-order assertions about the domains and ranges of functions. The second part of Figure 11.4 defines the **signature axiom**s generated by a declaration in a sorted signature Σ. We will denote the set of all signature axioms by $\mathcal{R}_{\mathcal{S}}(\Sigma)$. Similarly, we can define the set $\mathcal{R}_{\mathcal{S}}(\Gamma)$ of **sort assumption**s generated by a sorted context Γ by setting $\mathcal{R}_{\mathcal{S}}([X::\mathbb{A}]) := \mathcal{R}_{\mathcal{S}}(\mathbb{A})(X)$ for a declaration $[X::\mathbb{A}]$, we will use $\mathcal{R}_{\mathcal{S}}(\Gamma,\Sigma)$ for $\mathcal{R}_{\mathcal{S}}(\Gamma) \cup \mathcal{R}_{\mathcal{S}}(\Sigma)$.

$\mathcal{R}^l_{\mathcal{S}}\colon \mathcal{L}^{\mathbb{S}}\mathrm{FOL} \longrightarrow \mathcal{L}^{FOL}$	
Signature Σ	$\mathcal{R}_{\mathcal{S}}$-image
$a, f, g, \ldots$	$a, f, g, \ldots \in \overline{\Sigma}$
$\mathbb{A}, \mathbb{B}, \ldots$	$\mathcal{P}_{\mathbb{A}}, \mathcal{P}_{\mathbb{B}} \ldots \in \overline{\Sigma}$
$\mathbb{A} \to \mathbb{B}$	$\lambda F_{\iota\to\iota}. (\forall X. \mathcal{R}_{\mathcal{S}}(\mathbb{A})(X) \Rightarrow \mathcal{R}_{\mathcal{S}}(\mathbb{B})(FX))$
Π^*	$\lambda S_{\iota\to o}. \lambda T_{\iota\to o}. \forall X_\iota. SX \Rightarrow TX$
Signature Σ	Signature Axioms $\mathcal{R}_{\mathcal{S}}(\Sigma)$
$[\mathbb{A} \leq \mathbb{B}]$	$\mathcal{P}_{\mathbb{A}} \subseteq \mathcal{P}_{\mathbb{B}}$
$[c::\mathbb{A}]$	$\mathcal{R}^l_{\mathcal{S}}(\mathbb{A})(c)$

Figure 11.4: Formula Relativization from $\mathbb{S}$FOL to FOL: $\mathcal{R}^l_{\mathcal{S}}$

Their significance is that they encode all the information of the sorted signature in first-order logic, so that we have the following theorem:

Theorem 11.1.1 (Sort Relativization Theorem)
If $\Gamma \Vdash^s_{\Sigma} \mathbf{A}$*, then* $\mathcal{R}_{\mathcal{S}}(\Sigma,\Gamma) \Vdash^s_{\overline{\Sigma}} \mathcal{R}_{\mathcal{S}}(\mathbf{A})$

The proof is a direct consequence of the definition of $\mathcal{R}^l_{\mathcal{S}}$, defined in Figure 11.5: Let $\mathcal{D}\colon \Gamma \Vdash^s_{\Sigma} \mathbf{A}$, then $\mathcal{D}_{\mathcal{S}} orts^{ND(\mathbb{S}\mathrm{FOL})}\colon \mathcal{R}_{\mathcal{S}}(\Sigma,\Gamma) \Vdash^s_{\overline{\Sigma}} \mathcal{R}_{\mathcal{S}}(\mathbf{A})$, since $\mathcal{R}^l_{\mathcal{S}}$ is a calculus morphism from $\mathbb{S}$FOL to FOL.

All the discussion so far has been purely syntactic, we will come to semantic questions in the next section.

11.1.2 Relativizations and Semantics

Let us now see how the ideas of linking logics relates to semantics. In this example, we take the semantics of first-order logic as given. It is just the classicl Tarski-Style semantics: A model is a pair $(\mathcal{D}_\iota, \mathcal{I})$, where $\mathcal{D}_\iota$ is an arbitrary set of individuals and $\mathcal{I}$ is a function that maps individual constants in Σ to members of $\mathcal{D}_\iota$, funcitons in Σ to funcions/relations on $\mathcal{D}_\iota$ (of appropriate arities). Variables are evaluated by a variable assignment φ, so that the value funciton $\mathcal{I}_\varphi$ is just the homomorphism determined by $\mathcal{I}$ and φ[1].

[1]This semantics is absolutely consistent with our choice to take the simply typed λ-calculus as a meta-logic: the choice of the universe $\mathcal{D}_\iota$ determines the standard model $\mathcal{D} =$

$\dfrac{[\mathbb{A} \leq \mathbb{B}] \in \Sigma}{\models^s_\Sigma \mathbb{A} \leq \mathbb{B}}$	$\dfrac{}{\mathcal{R}_\mathcal{S}(\Sigma, \Gamma) \Vdash^\iota_\Sigma \mathcal{P}_\mathbb{A} \subseteq \mathcal{P}_\mathbb{B}}$ since $(\mathcal{P}_\mathbb{A} \subseteq \mathcal{P}_\mathbb{B}) \in \mathcal{T}_\Sigma$
$\dfrac{}{\models^s_\Sigma \mathbb{A} \leq \mathbb{A}}$	$\dfrac{\dfrac{\dfrac{}{\mathcal{P}_\mathbb{A} X \Vdash^\iota_\Sigma \mathcal{P}_\mathbb{A} X}}{\Vdash^\iota_\Sigma \mathcal{P}_\mathbb{A} X \Rightarrow \mathcal{P}_\mathbb{A} X}}{\Vdash^\iota_\Sigma \forall X.\mathcal{P}_\mathbb{A} X \Rightarrow \mathcal{P}_\mathbb{A} X}$
$\dfrac{\models^s_\Sigma \mathbb{A} \leq \mathbb{B} \quad \models^s_\Sigma \mathbb{B} \leq \mathbb{C}}{\models^s_\Sigma \mathbb{A} \leq \mathbb{C}}$	transitivity of $\Rightarrow$
$\dfrac{[c{::}\mathbb{A}] \in \Sigma}{\Gamma \models^s_\Sigma c{::}\mathbb{A}}$	$\dfrac{}{\mathcal{R}_\mathcal{S}(\Sigma) \Vdash^\iota_\Sigma \mathcal{P}_\mathbb{A} c}$ since $\mathcal{P}_\mathbb{A} c \in \mathcal{R}_\mathcal{S}(\Sigma)$
$\dfrac{}{\Gamma, [X{::}\mathbb{A}] \models^s_\Sigma X{::}\mathbb{A}}$	$\dfrac{}{\mathcal{R}(\Gamma), \mathcal{P}_\mathbb{A}(X) \Vdash^\iota_\Sigma \mathcal{P}_\mathbb{A} X}$
$\dfrac{\Gamma \models^s_\Sigma \mathbf{A}{::}\mathbb{B} \to \mathbb{C} \quad \Gamma \models^s_\Sigma \mathbf{B}{::}\mathbb{B}}{\Gamma \models^s_\Sigma \mathbf{AB}{::}\mathbb{C}}$	$\dfrac{\dfrac{\mathcal{R}_\mathcal{S}(\Sigma, \Gamma) \Vdash^\iota_\Sigma \forall X.\mathcal{P}_\mathbb{B} X \Rightarrow \mathcal{P}_\mathbb{C}(\mathcal{R}_\mathcal{S}(\mathbf{A}X))}{\mathcal{R}_\mathcal{S}(\Sigma, \Gamma) \Vdash^\iota_\Sigma \mathcal{P}_\mathbb{B}(\mathcal{R}_\mathcal{S}(\mathbf{B})) \Rightarrow \mathcal{P}_\mathbb{C}(\mathcal{R}_\mathcal{S}(\mathbf{AB}))} \quad \mathcal{R}_\mathcal{S}(\Sigma, \Gamma) \Vdash^\iota_\Sigma \mathcal{P}_\mathbb{B}(\mathcal{R}_\mathcal{S}(\mathbf{B}))}{\mathcal{R}_\mathcal{S}(\Sigma, \Gamma) \Vdash^\iota_\Sigma \mathcal{P}_\mathbb{C}(\mathcal{R}_\mathcal{S}(\mathbf{AB}))}$
$\dfrac{\Gamma \models^s_\Sigma \Pi^* \mathbb{A}\mathbf{B} \quad \Gamma \models^s_\Sigma \mathbf{A}{::}\mathbb{A}}{\Gamma \models^s_\Sigma \mathbf{BA}}$	$\dfrac{\dfrac{\mathcal{R}_\mathcal{S}(\Sigma, \Gamma) \Vdash^\iota_\Sigma \Pi(\lambda X.(\mathbb{A}X) \Rightarrow (\mathbf{B}X))}{\mathcal{R}_\mathcal{S}(\Sigma, \Gamma) \Vdash^\iota_\Sigma \mathbb{A}\mathbf{A} \Rightarrow \mathbf{BA}} \quad \mathcal{R}_\mathcal{S}(\Sigma, \Gamma) \Vdash^\iota_\Sigma \mathbb{A}\mathbf{A}}{\mathbf{BA}}$
$\dfrac{\Gamma, [X{::}\mathbb{A}] \models^s_\Sigma \mathbf{A}X}{\Gamma \models^s_\Sigma \Pi^* \mathbb{A}\mathbf{A}}$	$\dfrac{\dfrac{\mathcal{R}_\mathcal{S}(\Sigma, \Gamma), \mathcal{P}_\mathbb{A} X \Vdash^\iota_\Sigma \mathbf{A}X}{\mathcal{R}_\mathcal{S}(\Sigma, \Gamma) \Vdash^\iota_\Sigma \mathbb{A}X \Rightarrow \mathbf{A}X}}{\mathcal{R}_\mathcal{S}(\Sigma, \Gamma) \Vdash^\iota_\Sigma \Pi(\lambda X.\mathbb{A}X \Rightarrow \mathbf{A}X)}$

Figure 11.5: Proof Relativization from $\mathbb{S}$FOL to FOL: $\mathcal{R}^c_\mathcal{S}$.

The semantics of $\mathbb{S}$FOL is similar and well-known from the literature. Let us for the moment forget this and see whether we can *define* the semantics of the logical system $\mathbb{S}$FOL by $\mathcal{R}_\mathcal{S}$. For this we intuitively work the

$\{\mathcal{D}_\alpha \mid \alpha \in \mathcal{T}$ if we take $\mathcal{D}_o := \{\mathtt{T}, \mathtt{F}\}$. n-ary functions are then objects of type $\iota \to \ldots \to \iota \to \iota$, and predicates of type $\iota \to \ldots \to \iota \to \iota$. The semantics of universal quantification (see footnote **??**) is regained by setting $\mathcal{I}(\Pi^\iota)$ to be the predicate that evaluates to $\mathtt{T}$, iff its argument is $\mathtt{T}$ on all inputs: $\mathcal{I}_\varphi(\forall X.\mathbf{A}) = \mathcal{I}_\varphi(\Pi^\alpha(\lambda X.\mathbf{A})) = \mathcal{I}(\Pi^\alpha)(\mathcal{I}_\varphi(\lambda X.\mathbf{A})) = \mathtt{T}$, iff $\mathcal{I}_\varphi(\lambda X.\mathbf{A})\mathsf{a} = \mathtt{T}$ for all $\mathsf{a} \in \mathcal{D}_\alpha$. This is the case, iff $\mathcal{I}_{\varphi,[\mathsf{a}/X]}\mathbf{A} = \mathtt{T}$, by definition of the value of λ-terms in the simply typed λ-calculus.

induced semantics
semantics!induce
fragment
Otter@OTTER

relativization mapping backwards.

We start out with the sorts. These are members of the signature, so they should be reflected directly in the structure of the model. Since they are relativized to unary predicates, a sort $\mathbb{A}$ must correspond to a subset $\mathcal{D}_{\mathbb{A}} = \{\mathsf{a} \in \mathcal{D}_\iota \mid \mathcal{I}(\mathbb{A}) = \mathtt{T}\} \subseteq \mathcal{D}_\iota$ of the universe $\mathcal{D}_\iota$. Now, the signature axioms tell us that if $[\mathbb{A} \leq \mathbb{B}] \in \Sigma$, then $\mathcal{D}_{\mathbb{A}} \subseteq \mathcal{D}_{\mathbb{B}}$ and if $[c{::}\mathbb{A}] \in \Sigma$, then $\mathcal{I}(c) \in \mathcal{D}_{\mathbb{A}}$. In particular, the signature axiom for functional sorts insists on the right input-output behavior of functions. For a variable context Γ, the context assumptions specify that the context is well-sorted.

Note that this is a (a posteriori) verification of the semantics of sorted logics from the literature. Also note that this account does not entail the fact that sorts are non-empty (a fact that is often assumed in sorted logics). We only know this if there is a constant declaration for each base sort in the signature.

We will say that the semantics we have constructed by looking at the relativization was **induced** by $\mathcal{R}_{\mathcal{S}}$ from the the semantics of FOL. Now, the relativization theorem gives us a conservative extension result: If $ND(FOL)$ is sound for first-order semantics, then $ND(\mathbb{S}\mathrm{FOL})$ is for the induced semantics. Furthermore, the logical system $\mathbb{S}$FOL is not more expressive than FOL.

In the special case of $\mathcal{R}\colon \mathbb{S}\mathrm{FOL} \longrightarrow FOL$ we also have the converse result, ($\mathbb{S}$FOL and FOL are equally expressive), since there is a partial inverse $\mathcal{R}_{\mathbb{T}\mathsf{op}}$ to $\mathcal{R}_{\mathcal{S}}$ ($\mathcal{R}_{\mathcal{S}} \circ \mathcal{R}_{\mathbb{T}\mathsf{op}} = \mathrm{Id}_{FOL}$), which embeds FOL as a **fragment** into $\mathbb{S}$FOL. $\mathcal{R}_{\mathbb{T}\mathsf{op}}(\Sigma)$ contains only one (base) sort $\mathbb{T}\mathsf{op}_\iota$ and one declaration $[c{::}\mathbb{T}\mathsf{op}_\alpha]$, for each constant of type α in Σ (here we use the convention that $\mathbb{T}\mathsf{op}_{\alpha\to\alpha} = \mathbb{T}\mathsf{op}_\alpha \to \mathbb{T}\mathsf{op}_\beta$). The language and calculus morphism are the identity. Clearly, the semantics induced from the semantics of $\mathbb{S}$FOL by $\mathcal{R}_{\mathbb{T}\mathsf{op}}$ is again the semantics of FOL.

11.2 Relativizing Type Theory into Set Theory

The goal of the next section will be to construct a hierarchy of representation languages culminating in a high-level logical system $\mathcal{MV}$ (see section 12.1) for formalizing mathematics. $\mathcal{BMV}$ is a joint generalization of Ohori's record λ-calculus [Oho95] and the sorted λ-calculus from [Koh94]. $\mathcal{MV}$ extends $\mathcal{BMV}$ by specialized sort machinery to formalize mathematical structures like groups. Before we undertake that, let us briefly complete the discussion of the lower half of Figure 11.1. The method of relativizations can be used to build up the simply typed λ-calculus ($\lambda^{\rightarrow}$) from axiomatic set theories like ZF [Fra22], and we will spend the rest of this section exploring this possibility to ground the hierarchy of representation languages in set theories. Since the logical side of this is rather standard and well-understood (see e.g. [Ebb77]) and has been formalized in several deduction systems, e.g.in OTTER [Qua92]

*Isabelle@*ISABELLE
typed set theory

or ISABELLE [Pau93], we will only briefly sketch the process.

Axiomatic set theories like ZF only come with a basic type γ of "set" and with the logical relation constant $\in$ for element-hood. The axiomatic method is used to restrict set comprehension to get around paradoxical sets like Russell's set of all sets that do not contain themselves: the theories contain specific axioms for set comprehension; for instance there is an axiom stating that for any sets $\mathbb{A}$ and $\mathbb{B}$, the Cartesian product $\mathbb{A}\times\mathbb{B}$ is again a set. (Partial) functions are construed as univocal relations (a relation $\mathcal{F}\subset\mathbb{A}\times\mathbb{B}$ is a function, iff for all $(x,y),(x,z)\in\mathcal{F}$ we have $y=z$) and function application is represented as projection to the second argument ($\mathsf{f}[\mathsf{a}]$ is the (unique) b, such that $(\mathsf{a},\mathsf{b})\in\mathsf{f}$.)

We start out by relativizing the simply typed λ-calculus to **typed set theory** $\mathcal{T}Set$, i.e. a simply typed higher-order predicate logic HOL together with a formulation of the ZF axioms, interpreting sets as predicates and element-hood as predication (i.e. $\mathbf{A}\in\mathbf{S}$ stands for $\mathbf{S}(\mathbf{A})$).

HOL is a variant of Andrews' system $\mathcal{Q}$ with comprehension axioms instead of β-conversion; the types make $\mathcal{T}Set$ consistent (see [And86] for a deduction-related introduction of higher-order logic and the simply typed model theory). Using the techniques from [Ebb77, Pau93], we use the selection axiom from ZF to construct a Λ-operator, i.e. a $\mathcal{T}Set$-formula that behaves like the λ-abstraction operator. Thus we can construct a language morphism from the simply typed λ-calculus to $\mathcal{T}Set$ by mapping λ-abstractions in $\lambda^{\rightarrow}$ to HOL-formulae using Λ. The calculus morphism is constructed by mapping the β-axiom scheme of $\lambda^{\rightarrow}$ to the proof of the validity β in $\mathcal{T}Set$.

The next step is to relativize $\mathcal{T}Set$ (higher-order logic) to sorted first-order logic. For this, we can either use a technique developed by Manfred Kerber [Ker91] or we can directly use the definition of functions as univocal relations in ZF to build a logic morphism from $\mathcal{T}Set$ to sorted first-order logic. Finally, the techniques detailed in section 11.1.1 get us to classical ZF. Note that we have to take care to relativize the ZF axioms in the source system to a form in which they are equivalent to the ZF axioms native to the target system.

Note also that the logic morphisms presented in this section can always be used to transform any proof in the source system into one of the target system (this is the reason for the definition of logic morphism used in this book), in other words, from a purely theoretical point of view, the expressive type-theoretic representation formalisms in the $\mathcal{MV}$ hierarchy can be viewed as being only syntactic sugar to enhance legibility. However, from a practical point of view, the expressive formalisms allow for more efficient inference systems that allow the knowledge base system to give added value services, that would be impractical on the level of set theory.

We believe that while axiomatic set theories address foundational issues of formalizing mathematics – in the old *representational tradition* of classical logic, where there is a quest for the minimal logical system that is expressive enough to encode all relevant problems – logical systems like the simply typed λ-calculus are more adequate to address *computational* needs of doing mathematics.

Orthogonal to the debate about set theory vs. type theory, there is a discussion, whether or not formalized mathematics should be constructive or not. We do not make any assertion about this, but note, that it is simple to extend the hierarchy of representation languages by providing a logical morphism to intuitionistic set theory that basically introduces oracles for the law of the excluded middle; see e.g. [How96]. In Figure 11.1, we have marked the intuitionistic logical systems with an i and the oracle-morphisms with dotted lines.In the next section, we will continue to develop higher-level representation formalisms for mathematics by the logical morphism method discussed in this section.

Chapter 12

Mathematical Vernacular

12.1 Basic Mathematical Vernacular

In this chapter, we develop the basic concepts for a representation language for formalizing mathematics; we will extend it to the system MV in section **??** and show some mathematical examples in section 12.1.5.

$\mathcal{BMV}$ is a sorted record-λ-calculus (see Figure 12.1), i.e. an extension of the simply typed λ-calculus by records. The systems $\mathcal{BMV}$ and $\mathcal{MV}$ are intended to be used as a high-level representation and reasoning language for the mathematical knowledge base MBase[FK00]. Such a logic must be flexible, easy of use, and last but not least, it must support the rich, structured inference machinery mathematicians have a their disposal. In short, it should be oriented after the natural language of everyday mathematics, that is sometimes called "mathematical vernacular[1]"

In contrast to other authors, we contend that this language *can* be modelled in a formal language, and that the system MV is a good first approximation. We will develop the syntax and operational semantics of $\mathcal{MV}$, and show that it can be grounded in simpler logical systems (and ultimately in axiomatic set theory) by the technique of logical morphisms. This also gives us a way of relativizing all inference mechanisms, such as sort computation, sorted higher-order matching and unification into less expressive logics, where they can (if wanted) be verified.

Note that the relativizations give us a form of set-theoretic semantics (by mapping formulae to set theory), which can be shown to be equivalent to the standard Tarski-style semantics for $\Lambda^{\rightarrow}$ (see chapter 1 and 7).

To get a better intuition about the language, we will develop $\mathcal{MV}$ in three steps. To introduce the basic setup of the language we start out with a language $\mathcal{BMV}$, which extends the simply typed λ-calculus by sorts and records in section **??**. Then we successively enhance the practical expressive power

[1]This term is taken from N.G. de Bruin in [dB94], where he proposes logical system with similar intentions.

record type
selection
signature
context
term sorting

of the language by introducing label-selective application and abstractions, and dependent sorts as additional language constructs using semigroups as the motivating example. As we will see in section 12.1.4, this does not enhance the expressivity in principle, but (as we will see in the mathematical examples in section 12.1.5) it has practical advantages both for conciseness of representation and in enabling inference procedures. $\mathcal{BMV}$ is a sorted

$$
\begin{array}{lll}
\mathcal{T} & ::= \iota \mid o \mid \mathcal{T} \to \mathcal{T}' \mid \{\!\{\ell^1\colon \mathcal{T}^1, \ldots, \ell^n\colon \mathcal{T}^n\}\!\} & \text{(Types: } \alpha, \beta, \ldots) \\
\mathcal{S} & ::= \mathbb{T}\mathsf{op}_{\mathcal{T}} \mid \mathcal{S} \to \mathcal{S}' \mid \mathcal{S} \sqcap \mathcal{S}' \mid \{\!\{\ell_1 :: \mathcal{S}^1, \ldots, \ell_n :: \mathcal{S}^n\}\!\} & \text{(Sorts } \mathbb{A}, \mathbb{B}, \ldots) \\
\mathbf{M} & ::= X \mid c \mid \mathbf{MN} \mid \lambda x.\mathbf{M} & \\
 & \quad \mid \{\!\{\ell^1 = \mathbf{M}_1, \ldots, \ell^n = \mathbf{M}_n\}\!\} \mid \mathbf{M}.\ell & \text{(Formulae } \mathbf{A}, \mathbf{B}, \ldots) \\
 & \quad \text{variables}: \quad X, Y, Z; \quad \text{constants}: \quad c, \Pi^\alpha, \wedge, \neg & \\
\Sigma & ::= \emptyset \mid \Sigma, [\mathbf{M} :: \mathcal{S}] \mid \Sigma, [\mathcal{S} \triangleright \mathcal{T}] \mid \Sigma, [\mathcal{S} \leq \mathcal{S}'] & \text{(Signature)} \\
\Gamma & ::= \emptyset \mid \Gamma, [X :: \mathcal{S}] & \text{(Environment)}
\end{array}
$$

Figure 12.1: Syntax of $\mathcal{BMV}$

record-λ-calculus (see Figure 7.1), i.e. an extension of the simply typed λ-calculus by records. We will use the type o for the truth values and the type ι for individuals. As a consequence terms and formulae can be distinguished by their type: the equivalents of (first-order) formulae are λ-terms of type o, whereas terms are λ-terms of type ι. We will call a type a **record type**, iff it is of the form $\{\!\{\ell_1\colon \alpha_1, \ldots, \ell_n\colon \alpha_n\}\!\}$, and we will use the standard record **selection** operator "." with the assumption that it is only applied to record types.

Furthermore, the type system is augmented with a typed sort system, that can be used to specify domains and ranges of functions and thus enables the system to compute most of the definedness preconditions that are ubiquitous in mathematics fully autonomously. From an abstract point of view, sorts enable us to constrain the set of models and restrict the inference procedures to this set of models. It is important for the soundness of the system that sorts are also typed (see Figure **??** for an inference system that computes the type of a given sort).

The set of judgments (see Figure 12.2) that are needed for the formal development of the calculus comprises the typing judgments for terms $\Gamma \vdash_\Sigma \mathbf{A}\colon \alpha$ and sorts $\models^s_\Sigma \mathbb{A} \triangleright \alpha$, the sorting judgment ($\Gamma \models^s_\Sigma \mathbf{A} :: \mathbb{A}$).

All of these judgments are relative to a set of global type/sort assumptions in the **signature** Σ and the judgments for terms (sorts do not contain variables) are also relative to a set of (local) type and sort assumptions Γ (the **context**) for the variables.

The most important judgment for well-formedness of $\mathcal{MV}$ expressions is the **term sorting** judgment (see Figure 12.3), which classifies terms by their sorts. The first set of rules comes from the ordinary sorted λ-calculus (see [Koh94] for an introduction), the second is an obvious adaptation of

$\Gamma \Vdash^s_\Sigma \mathbf{M}$	$\mathbf{M}$ is provable from assumptions Γ
$\Vdash^s_\Sigma \mathbb{A} \leq \mathbb{B}$	Sort $\mathbb{A} \in \mathcal{S}$ is a subsort of $\mathbb{B} \in \mathcal{S}$
$\vdash_\Sigma \mathbb{A} \rhd \alpha$	Sort $\mathbb{A} \in \mathcal{S}$ has type $\alpha \in \mathcal{T}$ (at most one per sort $\mathbb{A}$)
$\Gamma \Vdash^s_\Sigma \mathbf{A}{::}\mathbb{A}$	Term $\mathbf{A}$ has sort $\mathbb{A}$ assuming Γ and Σ
$\Gamma \vdash_\Sigma \mathbf{A}{:}\,\alpha$	Term $\mathbf{A}$ has type $\alpha \in \mathcal{T}_0$ assuming Γ and Σ

Figure 12.2: Judgments

Ohori's rules for record typing [Oho95], and the third set of rules is that for intersection sorts from [KP93]. The most important rule in the sorted calculus is the first one in Figure 12.3, the term declaration rule. In contrast to other systems it allows to declare and use sort information for term schemata like $[X_\mathbb{R} * X{::}\mathbb{P}]$ (doubling a real number produces an positive real), $[(\lambda X.X), (\lambda X.Y_\mathbb{R}){::}\mathbb{M}]$ (the identity and the constant function are monomials), and even $[(\lambda F, G, X.FX * GX){::}\mathbb{M}^2 \to \mathbb{M}]$ (the set of monoids is closed under pointwise multiplication). Note that the latter give a full theory of monoid functions on the reals. The term typing judgment – which guar-

$$\frac{\Gamma \Vdash^s_\Sigma \mathbf{A}{::}\mathbb{A}}{\Gamma \Vdash^s_\Sigma \mathbf{A}.\ell{::}\mathbb{A}.\ell} \qquad \frac{\Gamma \Vdash^s_\Sigma \mathbf{A}_1{::}\mathbb{A}_1 \quad \ldots \quad \Gamma \Vdash^s_\Sigma \mathbf{A}_n{::}\mathbb{A}_n}{\Gamma \Vdash^s_\Sigma \{\!\{\ell_1 = \mathbf{A}_1, \ldots, \ell_n = \mathbf{A}_n\}\!\}{::}\{\!\{\ell_1{::}\mathbb{A}_1, \ldots, \ell_n{::}\mathbb{A}_n\}\!\}}$$

$$\frac{\mathbb{A}' \leq \mathbb{A} \quad \mathbb{B} \leq \mathbb{B}'}{\mathbb{A} \to \mathbb{B} \leq \mathbb{A}' \to \mathbb{B}'}$$

$$\frac{}{\Gamma \Vdash^s_\Sigma \{\!\{\ell = \mathbf{A}, \ldots\}\!\}.\ell \to_\rho \mathbf{A}} \qquad \frac{}{\Gamma \Vdash^s_\Sigma \{\!\{\ell_1 = \mathbf{A}.\ell_1, \ldots, \ell_n = \mathbf{A}.\ell_n\}\!\} \to_\eta \mathbf{A}}$$

Figure 12.3: Extensions for $\mathcal{BMV}$

antees consistency and termination of $\beta\eta$-reduction – is defined in terms of it: if $\Gamma \Vdash^s_\Sigma \mathbf{A}{::}\mathbb{A}$ and $\vdash_\Sigma \mathbb{A} \rhd \alpha$, then $\Gamma \vdash_\Sigma \mathbf{A}{:}\,\alpha$. Note that the typed system is just Ohori's record calculus [Oho95], which is a conservative extension of the simply typed λ-calculus since sorts are typed[82]. [83] EdNote(82)

Now, we come to the $\mathcal{BMV}$ calculus for validity: a variant of Gentzen's calculus of natural deduction. We will use alphabetic renaming and permutation for records, record types and record sorts without reference. Furthermore, $\mathcal{MV}$ knows sorted variants of $\beta\eta$ reduction like the one in [Koh94] and furthermore ρ-reduction for record constructors (see Figure 12.4). Finally, we have the introduction and elimination rules for the sorted quantifier $\Pi^\mathbb{A}$. EdNote(83)

[82] EdNote: reference to discussion
[83] EdNote: Adapt the discussion

keyword arguments
labeled application
application!labeled
labeled λ-abstraction
abstraction!labeled

The $\Pi^{\mathbb{A}}$ ($\mathbb{A} \in \mathcal{S}$) are logical constants of sort $(\mathbb{A} \to \mathbb{O}) \to \mathbb{O}$ for $\mathcal{BMV}$, we use the usual higher-order abstract syntax, where $\forall X_{\mathbb{A}}.\mathbf{A}$ stands for $\Pi^{\mathbb{A}}(\lambda X.\mathbf{A})$.

$$\frac{\Gamma \Vdash^s_\Sigma \Pi^{\mathbb{A}}\mathbf{B} \quad \Gamma \Vdash^s_\Sigma \mathbf{A}::\mathbb{A}}{\Gamma \Vdash^s_\Sigma \mathbf{B}\mathbf{A}} \qquad \frac{\Gamma, [X::\mathbb{A}] \Vdash^s_\Sigma \mathbf{A}X}{\Gamma \Vdash^s_\Sigma \Pi^{\mathbb{A}}\mathbf{A}} \qquad \frac{\Gamma \Vdash^s_\Sigma \mathbf{A} =_{\beta\eta\rho} \mathbf{B} \quad \Gamma \Vdash^s_\Sigma \mathbf{A}}{\Gamma \Vdash^s_\Sigma \mathbf{B}}$$

Figure 12.4: Natural Deduction for $\mathcal{BMV}$.

12.1.1 Label-Selective Abstraction and Application

When formalizing larger bodies of mathematics or reusing already existing theories it often becomes problematic to remember argument order of functions. For this, programming languages like Common Lisp – where the situation is similar – have developed the so-called **keyword arguments**, i.e. a variant of function application and abstraction, where the mapping of arguments to formal parameters is not based on argument order, but on identification by so-called keywords. This idea has been formalized by Ait Kaci and Garrigue in the so-called *label-selective λ-calculus* [AKG93], which extends the simply typed λ-calculus by label-selective application and abstraction.

In the following, we will briefly sketch how to extend $\mathcal{MV}$ analogously. Formally, we need an additional type schema: $\alpha \xrightarrow{\ell} \beta$, a corresponding sort schema $\mathbb{A} \xrightarrow{\ell} \mathbb{B}$ and two new term constructors: $[\mathbf{A}@_\ell\mathbf{B}]$ for **labeled application** and $\lambda^\ell X.\mathbf{A}$ for **labeled λ-abstraction**. We will reuse the record labels as selection labels, since they serve a similar purpose (ℓ and k correspond to the keywords in Lisp). Finally, we will use the n-ary notation $[\mathbf{A}@_\ell\mathbf{B}@_k\mathbf{C}]$ as an abbreviation for $[[\mathbf{A}@_\ell\mathbf{B}]@_k\mathbf{C}]$.

The extensions to the respective inference systems can be found in Figure 12.5. In particular, we consider labeled application/abstraction to be commutative (they are associative by construction, since types are left-associative). With this extension, to $\mathcal{MV}$ we can for instance have a constant div for integer division and express the term $5\mathsf{div}2$ as $[\mathsf{div}@_{\mathsf{dividend}}5@_{\mathsf{divisor}}2]$ or $[\mathsf{div}@_{\mathsf{divisor}}2@_{\mathsf{dividend}}5]$

12.1.2 Dependent (Record) Sorts

Label-selectivity gives us another advantage, we can extend it to a system with dependent sorts, if we allow terms and labels of type $\beta \to o$ to appear as base sorts locally. Conceptually, in $\mathcal{BMV}$, sorts are unary predicate *constants*, so the generalization is not as large as it seems at first. Let us look at the following formalization of associativity:

$$\mathsf{assoc} := \lambda^{\mathsf{Set}} S.\lambda^{\mathsf{Op}} F.\forall X_S.\forall Y_S.\forall Z_S.FX(FYZ) = F(FXY)Z \qquad (12.1)$$

$$\frac{\Gamma \models^s_\Sigma \mathbf{A}::\mathbb{C} \xrightarrow{\ell} \mathbb{A} \quad \Gamma \models^s_\Sigma \mathbf{C}::\mathbb{C}}{\Gamma \models^s_\Sigma [\mathbf{A}@_\ell\mathbf{C}]::\mathbb{A}} \qquad \frac{\Gamma, X::\mathbb{B} \models^s_\Sigma \mathbf{A}::\mathbb{A}}{\Gamma \models^s_\Sigma \lambda^\ell X.\mathbf{A}::\mathbb{B} \xrightarrow{\ell} \mathbb{A}}$$

$$\frac{\Gamma, X::\mathbb{B} \models^s_\Sigma \mathbf{A}::\mathbb{A} \quad \Gamma \models^s_\Sigma \mathbf{B}::\mathbb{B}}{[(\lambda^\ell X.\mathbf{A})@_\ell\mathbf{B}] \to_\beta [\mathbf{B}/X]\mathbf{A}}$$

$$\frac{X \notin \mathbf{Free}(\mathbf{B})}{[(\lambda^\ell X.\mathbf{A})@_k\mathbf{B}] \to_\eta (\lambda^\ell X.[\mathbf{A}@_k\mathbf{B}])} \qquad \frac{\Gamma \vdash_\Sigma \mathbf{A}\colon \alpha \xrightarrow{\ell} \beta \quad X \notin \mathbf{Free}(\mathbf{A})}{(\lambda^\ell X.[\mathbf{A}@_\ell X]) \to_\eta \mathbf{A}}$$

$$\frac{}{\mathbb{A}_1 \xrightarrow{\ell_1} \mathbb{A}_2 \xrightarrow{\ell_2} \mathbb{B} =_\rho \mathbb{A}_2 \xrightarrow{\ell_2} \mathbb{A}_1 \xrightarrow{\ell_1} \mathbb{B}}$$

$$\frac{}{(\lambda^\ell X.\lambda^k Y.\mathbf{A}) =_\rho (\lambda^k Y.\lambda^\ell X.\mathbf{A})} \qquad \frac{}{[\mathbf{A}@_\ell\mathbf{B}@_k\mathbf{C}] =_\rho [\mathbf{A}@_k\mathbf{C}@_\ell\mathbf{B}]}$$

Figure 12.5: A label-selective extension to $\mathcal{MV}$.

In $\mathcal{BMV}$, this would have the sort $\mathbb{T}\mathsf{op}_{\alpha\to o} \to (\mathbb{A} \to \mathbb{A} \to \mathbb{A}) \to \mathbb{O}$ for some a priori given sort $\mathbb{A}$. We would however have $[\mathsf{assoc}@_{\mathsf{Set}}\mathbf{S}]$ (associativity on a given set $\mathbf{S}$) to have sort $(\mathbf{S} \to \mathbf{S} \to \mathbf{S}) \to \mathbb{O}$, i.e. to be a predicate on binary functions on $\mathbf{S}$.

If we extend the set of base sorts by variables of type $\alpha \to o$ (and of course make the sort typing judgment dependent on a context Γ, by extending all rules in Figure **??** with contexts in the obvious way) and add the first two rules in Figure 12.6 to $\mathcal{MV}$, then we obtain the following sort derivation, which gives us a sort that *does* show the dependence missing above.

$$\dfrac{\dfrac{\dfrac{\vdots}{[S::\mathbb{T}\mathsf{op}_{\alpha\to o}], [F::S \to S \to S] \models^s_\Sigma (\forall X_S, Y_S.\forall Z_S.FX(FYZ) = F(FXY)Z)::\mathbb{O}}}{[S::\mathbb{T}\mathsf{op}_{\alpha\to o}] \vdash_\Sigma \lambda^{\mathsf{Op}}F.\forall X_S, Y_S.\forall Z_S.FX(FYZ) = F(FXY)Z::(S \to S \to S) \xrightarrow{\mathsf{Op}} \mathbb{O}}}{\models^s_\Sigma \mathsf{assoc}::\mathbb{T}\mathsf{op}_{\alpha\to o} \xrightarrow{\mathsf{Set}} (\mathsf{Set}^3) \xrightarrow{\mathsf{Op}} \mathbb{O}}$$

Here (and in the following) we use $\mathbb{A}^3$ as an abbreviation for the sort $\mathbb{A} \to \mathbb{A} \to \mathbb{A}$.

Unfortunately, the sorts discussed so far are not yet expressive enough for a direct representation of common mathematical structures such as semigroups. A semigroup is a pair $(S, \circ)$, were S is an arbitrary set and $\circ\colon S \times S \longrightarrow S$ is an associative binary function on S. Just as in the case of associativity discussed in section 12.1.2, we would like to represent S as a sort $\mathbb{S}$ and $\circ$ as a function of sort $\mathbb{S} \to \mathbb{S} \to \mathbb{S}$ in a record of type

very dependent record sorts
merge

$\{\!\!\{\mathsf{Set}\colon \alpha \to o, \mathsf{Op}\colon \alpha \to \alpha \to \alpha\}\!\!\}$. However, in the system developed so far, we cannot express a record sort like

$$\mathbb{S}\mathsf{op} := \{\!\!\{\mathsf{Set}{::}\mathbb{T}\mathsf{op}_{\alpha\to o}, \mathsf{Op}{::}\mathsf{Set} \to \mathsf{Set} \to \mathsf{Set}\}\!\!\}$$

The second two rules in Figure 12.6 extend $\mathcal{MV}$ by **very dependent record sorts**. This name is chosen to resemble Jason Hickey's "very dependent record types" [Hic96] and serve the same purpose, even if the formalization on the level of sorts is much more unproblematic, since there are no consistency problems involved: Well-typedness is preserved at the level of (simple) record types.

To make the records dependent, we have to serialize the record construction rule from Figure 12.3. Technically, we will (ab)use the context to store the necessary assumptions about the feature values and use the standard record **merge** operator $\otimes$ to write down the rules in Figure 12.6. Let $\mathbb{N}$

$$\frac{\Gamma, [X{::}\mathbb{B}] \models^s_\Sigma \mathbf{A}{::}\mathbb{A} \quad \vdash_\Sigma \mathbb{B} \rhd \beta \to o}{\Gamma \models^s_\Sigma \lambda^\ell X.\mathbf{A}{::}\mathbb{B} \xrightarrow{\ell} [\ell/X]\mathbb{A}}$$

$$\frac{\Gamma \models^s_\Sigma \mathbf{A}{::}\mathbb{C} \xrightarrow{\ell} \mathbb{A} \quad \Gamma \models^s_\Sigma \mathbf{C}{::}\mathbb{C}}{\Gamma \models^s_\Sigma \mathbf{AC}{::}[\mathbf{C}/\ell]\mathbb{A}}$$

$$\frac{\Gamma, [\ell = \mathbf{A}] \models^s_\Sigma \mathbf{B}{::}\mathbb{B} \quad \Gamma \models^s_\Sigma \mathbf{A}{::}\mathbb{A} \quad \Gamma \vdash_\Sigma \mathbb{A} \rhd \alpha \to o}{\Gamma \models^s_\Sigma \{\!\!\{\ell = \mathbf{A}\}\!\!\} \otimes \mathbf{B}{::}\{\!\!\{\ell{::}\mathbb{A}\}\!\!\} \otimes \mathbb{B}}$$

$$\frac{\Gamma \models^s_\Sigma [\mathbf{B}/\ell]\mathbf{A}{::}[\mathbf{B}/\ell]\mathbb{A}}{\Gamma, [\ell = \mathbf{B}] \models^s_\Sigma \mathbf{A}{::}\mathbb{A}}$$

Figure 12.6: Extending $\mathcal{MV}$ by dependent sorts

be the set of natural numbers and $+{::}\mathbb{N}^3$ the addition function on natural numbers, then we have the following sort derivation in $\mathcal{MV}$.

$$\frac{\dfrac{\dfrac{\models^s_\Sigma +{::}\mathbb{N}^3}{[\mathsf{Set} = \mathbb{N}] \models^s_\Sigma +{::}\mathsf{Set}^3}}{[\mathsf{Set} = \mathbb{N}] \models^s_\Sigma \{\!\!\{\mathsf{Op} = +\}\!\!\}{::}\{\!\!\{\mathsf{Op}{::}\mathsf{Set}^3\}\!\!\}} \quad \models^s_\Sigma \mathbb{T}\mathsf{op}_{\iota\to o} \rhd \iota \to o}{\models^s_\Sigma \{\!\!\{\mathsf{Set} = \mathbb{N}, \mathsf{Op} = +\}\!\!\}{::}\mathbb{S}\mathsf{op}}$$

12.1.3 Selection Sorts and Semigroups

In this section, we will fortify our intuition about $\mathcal{MV}$ by considering an example from elementary algebra: semigroups. To be able to handle them *selection sort*

naturally, we will need to upgrade the system by **selection sort**s. Concretely, we use a new sort constructor $\{\cdot|\cdot\}$ that yields a new (base) sort $\{\mathbb{A}|\mathbf{P}\}$ for a given sort $\mathbb{A}$ with $\vdash_\Sigma \mathbb{A} \rhd \alpha$ and a closed term $\mathbf{P}$ of type $\alpha \to o$. Intuitively, this sort corresponds to the set of all objects of sort $\mathbb{A}$, on which $\mathbf{P}$ holds. Consider for instance the set of continuous real functions, that we

$$\frac{}{\models^s_\Sigma \{\mathbb{A}|\mathbf{P}\} \leq \mathbb{A}} \qquad \frac{\Gamma \models^s_\Sigma \mathbf{A} :: \mathbb{A} \quad \Gamma \Vdash^s_\Sigma \mathbf{P}\mathbf{A}}{\Gamma \models^s_\Sigma \mathbf{A} :: \{\mathbb{A}|\mathbf{P}\}}$$

$$\frac{\vdash_\Sigma \mathbf{P} \colon \alpha \to o \quad \vdash_\Sigma \mathbb{A} \rhd \alpha}{\vdash_\Sigma \{\mathbb{A}|\mathbf{P}\} \colon \alpha} \qquad \frac{}{\Gamma, [X :: \{\mathbb{A}|\mathbf{P}\}] \Vdash^s_\Sigma \mathbf{P}X}$$

Figure 12.7: Augmenting $\mathcal{MV}$ by Selection Sorts

can model as the sort $\mathbb{C} := \{(\mathbb{R} \to \mathbb{R})|(\lambda X \forall \epsilon. \exists \delta. \ldots)\}$. Now we can represent the theorem that the sum of two continuous functions is again continuous by $\forall F_\mathbb{C} G_\mathbb{C}. \mathbb{C}(\lambda X_\mathbb{R}. + (FX)(GX))$ If we want to prove this lemma, we have to be able to expand the definitions of the sort $\mathbb{C}$, which explains the necessity of the last axiom in Figure 12.7. Note that once we have proven this theorem, we can interpret it as a term declaration, add it to the signature, and directly use it for further sort computations.

But let us come back to the problem of modeling semigroups. We have seen in section 12.1.2 that we can represent the structure consisting of a set and an operation on this set by the sort $\mathbb{S}\mathsf{op} = \{\!\{\mathsf{Set} :: \mathbb{T}\mathsf{op}_{\alpha \to o}, \mathsf{Op} :: \mathsf{Set}^3\}\!\}$, thus we can represent the sort of all semigroups by

$$\mathbb{S}\mathsf{emigroup} := \{\mathbb{S}\mathsf{op}|\mathbf{A}\} \quad \text{where} \quad \mathbf{A} := (\lambda X. [\mathsf{assoc} @_{\mathsf{Set}} (X.\mathsf{Set}) @_{\mathsf{Op}} (X.\mathsf{Op})])$$

As we have seen above, we have $\models^s_\Sigma \mathsf{assoc} :: \mathbb{A} := (\mathbb{T}\mathsf{op}_{\alpha \to o}) \xrightarrow{\mathsf{Set}} (\mathsf{Set}^3) \xrightarrow{\mathsf{Op}} \mathbb{O}$ and therefore we can show that $\mathbb{S}\mathsf{emigroup}$ is a well-typed sort.

$$\cfrac{\cfrac{\cfrac{\cfrac{[X :: \mathbb{S}\mathsf{op}] \models^s_\Sigma \mathsf{assoc} :: \mathbb{A} \quad [X :: \mathbb{S}\mathsf{op}] \models^s_\Sigma X.\mathsf{Set} :: \mathbb{T}\mathsf{op}_{\alpha \to o}}{[X :: \mathbb{S}\mathsf{op}] \models^s_\Sigma [\mathsf{assoc} @_{\mathsf{Set}} X.\mathsf{Set}] :: (X.\mathsf{Set})^3 \xrightarrow{\mathsf{Op}} \mathbb{O}} \quad \cfrac{\vdots}{[X :: \mathbb{S}\mathsf{op}] \models^s_\Sigma X.\mathsf{Op} :: (X.\mathsf{Set})^3}}{[X :: \mathbb{S}\mathsf{op}] \models^s_\Sigma [\mathsf{assoc} @_{\mathsf{Set}} (X.\mathsf{Set}) @_{\mathsf{Op}} (X.\mathsf{Op})] :: \mathbb{O}}}{\models^s_\Sigma \lambda X [\mathsf{assoc} @_{\mathsf{Set}} (X.\mathsf{Set}) @_{\mathsf{Op}} (X.\mathsf{Op})] :: \mathbb{S}\mathsf{op} \to \mathbb{O}}}{\vdash_\Sigma \{\mathbb{S}\mathsf{op}|\lambda X [\mathsf{assoc} @_{\mathsf{Set}} (X.\mathsf{Set}) @_{\mathsf{Op}} (X.\mathsf{Op})]\} \rhd \{\!\{\mathsf{Set} \colon \alpha \to o, \mathsf{Op} \colon \alpha^3\}\!\}}$$

To see how we can use the selection sorts, let us now prove that the operation of a semigroup is associative on its set, i.e. we want to prove the formula $\Pi^{\mathbb{S}\mathsf{emigroup}}\mathbf{A}$, then – using $\mathbb{S}\mathsf{emigroup} = \{\mathbb{S}\mathsf{op}|\mathbf{A}\}$ – we have $[X :: \mathbb{S}\mathsf{emigroup}] \Vdash^s_\Sigma \mathbf{A}X$ by the axiom in Figure 12.6 and thus $\Vdash^s_\Sigma \Pi^{\mathbb{S}\mathsf{emigroup}}\mathbf{A}$ by the sorted quantifier introduction rule from Figure 12.4.

12.1.4 Relativization for extended $\mathcal{MV}$

In this section, we will present two relativization morphisms that show that the records and selection sorts in $\mathcal{MV}$ can be eliminated and that therefore $\mathcal{MV}$ is a conservative extension of the simply typed λ-calculus.

For constructing an elimination morphism for label-selective applications and abstractions, we will make use of the record calculus in $\mathcal{BMV}$. Intuitively, the translation works like this: maximal chains of labeled abstractions are represented as single abstractions over records with the same labels. Similarly, maximal chains of labeled applications as applications to single records. In our example involving integer division we would translate:

$$\begin{array}{lcr} [[\mathsf{div}@_{\mathsf{dividend}}5]@_{\mathsf{divisor}}2] & \text{to} & \mathsf{div}\{\!\{\mathsf{dividend}=5,\mathsf{divisor}=2\}\!\} \\ \mathsf{div}=_\eta\lambda^{\mathsf{dividend}}X.\lambda^{\mathsf{divisor}}Y.\mathbf{A} & \text{to} & \lambda Z_{\{\!\{\mathsf{dividend}:\iota,\mathsf{divisor}:\iota\}\!\}}.\mathbf{A}' \end{array}$$

The language morphism $\overline{\cdot}$ is given by the three equations below.

$$\begin{array}{rcl} \overline{\mathbb{A}_1 \xrightarrow{\ell_1} \ldots \xrightarrow{\ell_{n-1}} \mathbb{A}_n \xrightarrow{\ell_n} \mathbb{B}} & = & \{\!\{\ell_1{::}\overline{\mathbb{A}_1}\}\!\} \otimes [\mathbb{A}_1/\ell_1](\{\!\{\ell_2{::}\overline{\mathbb{A}_2}\}\!\} \otimes [\mathbb{A}_2/\ell_2](\ldots \\ & & \quad \otimes[\mathbb{A}_{n-1}/\ell_{n-1}]\{\!\{\ell_n{::}\overline{\mathbb{A}_n}\}\!\})\ldots) \to \overline{\mathbb{B}} \\ \overline{\lambda^{\ell_1}X_1.\ldots\lambda^{\ell_n}X_n.\mathbf{A}} & = & \lambda Z.\overline{[Z.\ell_1/X_1],\ldots,[Z.\ell_n/X_n]\mathbf{A}} \\ \overline{[\mathbf{B}@_{\ell_1}\mathbf{C}_1@\ldots@_{\ell_n}\mathbf{C}_n]} & = & \overline{\mathbf{A}}\{\!\{\ell_1=\overline{\mathbf{C}_1},\ldots,\ell_n=\overline{\mathbf{C}_n}\}\!\} \end{array}$$

Here, $\mathbf{A}$ and $\mathbb{B}$ must be of base type and $\mathbf{B}$ may not be an application, so that we always transform maximal sequences of arguments and bound variables in one step. Note that this is not a restriction of generality, since we can always η-expand. This translation uses the fact that entries in a record do not have a fixed order to obtain the order-independence of labeled abstraction:

$$\begin{array}{rcl} \overline{[\mathbf{B}@_{\ell_1}\mathbf{C}_1@_{\ell_2}\mathbf{C}_2]} & = & \overline{\mathbf{B}}\{\!\{\ell_1=\overline{\mathbf{A}_1},\ell_2=\overline{\mathbf{A}_2}\}\!\} \\ & = & \overline{\mathbf{B}}\{\!\{\ell_2=\overline{\mathbf{A}_2},\ell_1=\overline{\mathbf{A}_1}\}\!\} = \overline{[\mathbf{B}@_{\ell_2}\mathbf{C}_2@_{\ell_1}\mathbf{C}_1]} \\ \overline{\lambda^{\ell_1}X_1.\lambda^{\ell_2}X_2.\mathbf{A}} & = & \lambda Z.\overline{[Z.\ell_1/X_1],[Z.\ell_2/X_2]\mathbf{A}} \\ & = & \lambda Z.\overline{[Z.\ell_2/X_2],[Z.\ell_1/X_1]\mathbf{A}} = \overline{\lambda^{\ell_2}X_2.\lambda^{\ell_1}X_1.\mathbf{A}} \end{array}$$

The argumentation for the types and sorts is analogous to the case for applications. A tedious but simple calculation with ND proofs shows that this language morphism can be extended to a calculus morphism. In particular, the inference rules in Figure 12.5 turn into trivial ND proofs about records and their sets of labels.

We will not go in to details for relativizing away record sorts and types. This can be achieved by using one of two standard techniques. By introducing a new type ρ for record objects and modeling all record labels as partial functions from records to values. Thus a record $\mathcal{BMV}$ of type $\{\!\{\ell\colon\alpha,k\colon\beta\}\!\}$ would receive type ρ and we would extend the signature by functions $f_\ell\colon \rho \to \alpha$ and $f_k\colon \rho \to \beta$. Alternatively, one can fix an ordering on record labels and map records to n-tuples.

The logic morphism for eliminating selection sorts uses the fact that we can define selection sorts by relativization using $\mathcal{P}_{\{\mathbb{A}|\mathbf{P}\}} := \lambda X. \mathcal{P}_{\mathbb{A}} X \wedge \mathbf{P}X$, just like we did for ordinary sort relativization in section 11.1.1. Thus the language morphism relativizes all occurrences of formulae of the form $\Pi^{\{\mathbb{A}|\mathbf{P}\}}\mathbf{A}$ to $\Pi^{\mathbb{A}}(\lambda X.\mathbf{P}X \Rightarrow \mathbf{A})$. The calculus morphism is given in Figure 12.8, it shows the relativization rules for $\Pi^{\{\mathbb{A}|\mathbf{P}\}}$ elimination and introduction. Finally, the relativization of the axiom in Figure 12.6 is a trivial. Note that this allows any proof $\Gamma, [X{::}\{\mathbb{A}|\mathbf{P}\}] \Vdash^{s}_{\Sigma} \mathbf{A}$ to be transformed into one of the form for $\Gamma, [X{::}\mathbb{A}] \Vdash^{\ell}_{\Sigma} \mathbf{A}$ under the assumption of $\Gamma, [X{::}\mathbb{A}] \Vdash^{\ell}_{\Sigma} \mathbf{P}X$. This justifies the implication introduction step in the transformation of the second derivation in Figure 12.8 (we only have to make sure that the first rule is eliminated first during the transformation).

$$\dfrac{\dfrac{\Gamma \Vdash^{\ell}_{\Sigma} \Pi^*\mathbb{A}(\lambda X.\mathbf{P}X \Rightarrow \mathbf{B}X) \quad \Gamma \Vdash^{s}_{\Sigma} \mathbf{A}{::}\mathbb{A}}{\Gamma \Vdash^{\ell}_{\Sigma} \mathbf{PA} \Rightarrow \mathbf{BA}} \quad \Gamma \Vdash^{\ell}_{\Sigma} \mathbf{PA}}{\Gamma \Vdash^{\ell}_{\Sigma} \mathbf{BA}} \qquad \dfrac{\dfrac{\dfrac{\begin{matrix}[\Gamma, [X{::}\mathbb{A}] \Vdash^{\ell}_{\Sigma} \mathbf{P}X]^1 \\ \Gamma, [X{::}\mathbb{A}] \Vdash^{\ell}_{\Sigma} \mathbf{A}X\end{matrix}}{\Gamma, [X{::}\mathbb{A}] \Vdash^{\ell}_{\Sigma} \mathbf{P}X \Rightarrow \mathbf{A}X}\,1}{\Gamma \Vdash^{\ell}_{\Sigma} (\lambda Y.\mathbf{P}Y \Rightarrow \mathbf{A}Y)X}}{\Gamma \Vdash^{\ell}_{\Sigma} \Pi^*\mathbb{A}(\lambda X.\mathbf{P}X \Rightarrow \mathbf{A}X)}$$

Figure 12.8: Relativizing Selection Sorts (from the Calculus Morphism)

12.1.5 Mathematical Example

In this section, we will fortify our intuition about $\mathcal{MV}$ by considering an example from elementary algebra: semigroups. We have seen in section **??**, that we can represent the structure consisting of a set and an operation on this set by

$$\mathbb{S}\mathsf{op} := \{\!\{\mathsf{Set}{::}\mathbb{T}\mathsf{op}_{\alpha\to o}, \mathsf{Op}{::}\mathsf{Set}^3\}\!\}$$

Thus we can represent the sort of all semigroups by

$$\mathbb{S}\mathsf{emigroup} := \{\mathbb{S}\mathsf{op}|(\lambda X.[\mathsf{assoc}@_{\mathsf{Set}}(X.\mathsf{Set})@_{\mathsf{Op}}(X.\mathsf{Op})])\}$$

using the definition of associativity (12.1). Now let us first see that this is a well-typed sort, i.e. that we can derive $\vdash_\Sigma \mathbb{S}\mathsf{emigroup} \rhd \alpha$ for some type α. We will start by looking at the type of the sort $\mathbb{S}$.

$$\dfrac{\dfrac{\vdash_\Sigma \mathbb{T}\mathsf{op}_{\alpha\to o} \rhd \alpha \to o}{\vdash_\Sigma \{\!\{\mathsf{Set}{::}\mathbb{T}\mathsf{op}_{\alpha\to o}\}\!\} \rhd \{\!\{\mathsf{Set}\colon \alpha \to o\}\!\}} \quad \mathsf{Set}^3 \rhd \alpha^3 \in \{\!\{\mathsf{Set}{::}\mathbb{T}\mathsf{op}_{\alpha\to o}\}\!\}}{\vdash_\Sigma \mathbb{S} \rhd \{\!\{\mathsf{Set}\colon \alpha \to o, \mathsf{Op}\colon \alpha^3\}\!\}}$$

As we have seen above, we have the sort judgement $\vdash_\Sigma \mathsf{assoc}::\mathbb{A}$, where

$$\mathbb{A} := (\mathbb{T}\mathsf{op}_{\alpha\to o}) \xrightarrow{\mathsf{Set}} (\mathsf{Set}^3) \xrightarrow{\mathsf{Op}} \mathbb{O}$$

for associativity, and therefore we have

$$\dfrac{\dfrac{[X::\mathbb{S}] \vdash_\Sigma \mathsf{assoc}::\mathbb{A} \quad [X::\mathbb{S}] \vdash_\Sigma X.\mathsf{Set}::\mathbb{T}\mathsf{op}_{\alpha\to o}}{[X::\mathbb{S}] \vdash_\Sigma [\mathsf{assoc}@_{\mathsf{Set}}X.\mathsf{Set}]::(X.\mathsf{Set})^3 \to \mathbb{O}} \quad \dfrac{\vdots}{[X::\mathbb{S}] \vdash_\Sigma X.\mathsf{Op}::(X.\mathsf{Set})^3}}{[X::\mathbb{S}] \vdash_\Sigma [\mathsf{assoc}@_{\mathsf{Set}}(X.\mathsf{Set})@_{\mathsf{Op}}(X.\mathsf{Op})]::\mathbb{O}}$$

With this, we can show that $\mathbb{S}\mathsf{emigroup}$ is a well-typed sort

$$\dfrac{\vdash_\Sigma (\lambda X\mathbf{.}[\mathsf{assoc}@_{\mathsf{Set}}(X.\mathsf{Set})@_{\mathsf{Op}}(X.\mathsf{Op})]::)\mathbb{S} \to \mathbb{O}}{\vdash_\Sigma \{\mathbb{S}|(\lambda X\mathbf{.}[\mathsf{assoc}@_{\mathsf{Set}}(X.\mathsf{Set})@_{\mathsf{Op}}(X.\mathsf{Op})])\} \gtrdot \{\!\{\mathsf{Set}\colon \alpha \to o, \mathsf{Op}\colon \alpha^3\}\!\}}$$

To see how we can use the restriction sorts, let us now prove that the operation of a semigroup is associative on its set, i.e. we want to prove the formula

$$\Pi^{\mathbb{S}\mathsf{emigroup}}(\lambda X[\mathsf{assoc}@_{\mathsf{Set}}(X.\mathsf{Set})@_{\mathsf{Op}}(X.\mathsf{Op})])$$

Let $\mathbf{A} := (\lambda X[\mathsf{assoc}@_{\mathsf{Set}}(X.\mathsf{Set})@_{\mathsf{Op}}(X.\mathsf{Op})])$, then — using $\mathbb{S}\mathsf{emigroup} = \{\mathbb{S}\mathsf{op}|\mathbf{A}\}$ — we obtain the following ND proof.

$$\dfrac{\dfrac{}{X::\mathbb{S}\mathsf{emigroup} \Vdash_\Sigma^{\prime} \mathbf{A}X}}{\Vdash_\Sigma^{\prime} \Pi^{*}\mathbb{S}\mathsf{emigroup}\mathbf{A}}$$

12.2 Conclusion and Future Work

In this chapter, we have presented a methodology for building a hierarchy of representation languages for a mathematical knowledge base. We have shown that using logic morphisms allow to define high-level language features, such as dependent sorts step-by-step from lower (and more standard) ones, and ultimately from axiomatic set theory. The intended meaning of the more expressive logical systems is induced via the logical morphisms from the simpler logical systems, thus a knowledge base that is built up using the method proposed in this chapter is truly grounded in set theory. An implementation of the logic morphisms in a knowledge base system, such as the MBase system under development in Saarbrücken, will give constructive evidence to the old belief of working mathematicians that all of mathematics can be relativized (and thus grounded) in set theory.

We have instantiated this methodology by sketching the development of a sorted λ-calculus that we claim is well-suited for formalizing mathematical practice. It is an extension of the sorted λ-calculus from [Koh94] by dependent function–, record–, and selection sorts. We have sketched the relativizations needed to integrate it into the MBase system. The advantage of a sorted formulation over a classical type-theoretic one (e.g. LF dependent type discipline or Jason Hickeys "very dependent record types" [Hic96]) is that consistency is a consequence of the relativization, since the sorts are typed. This makes all objects simply typed, and hence important metatheorems like strong normalization of the built-in reductions are relatively easy to prove. *MBase*@MBase *MBase*@MBase

The next step will be to develop inference procedures like higher-order matching that are needed for answering high-level queries in MBase. We conjecture that this will be possible by adapting the methods (in particular, the structure theorem) from [Koh94]. In fact, one key motivation to extend known representation languages for mathematics by the additional structure developed in this chapter was to use the additional structure for inference purposes. We conjecture that the availability of such inference procedures will decide on the usefulness, and thus ultimately on the success of a mathematical knowledge base system.

Chapter 13

Conclusion

This book has been motivated essentially by practical practical considerations, i.e. the shortcomings of first-order theorem proving and the lack of expressive power in higher-order logics due to the absence of sorts. However, as it turned out the content of this book is rather theoretical in nature. To bridge this gap and to see whether this theoretical system can fulfill its practical expectations, let us have a look at some applications of $\Sigma\mathcal{HOL}$.

We claim that introducing sorts to higher-order logic results in considerably more expressivity, and hence ultimately a practical language to express mathematical facts naturally. In fact, $\Sigma\mathcal{HOL}$ is the logical basis for the higher-order logic $\mathcal{POST}$ (**P**artial **O**rder-**S**orted **T**ype theory) of the Ω-MKRP deduction system [BCF+97], currently under development at the Universität des Saarlandes. The goal of the Ω-MKRP project is to develop an interactive proof development environment which can be used to prove the total contents of a typical mathematical textbook. The extensive experience of the Ω-MKRP group, gained by axiomatizing mathematical theories for automated theorem provers, and the critique of existing input languages, which are considered too weak, have been a major motivation for the work reported in this book. The experiments with this new system have verified that sorted higher-order logics indeed offer a sufficiently rich language for naturally specifying a non-trivial fragment of mathematics. These experiments also show that, while term declarations are a desirable feature of a specification language for mathematics, pattern signatures have so far been sufficient for all practical applications. Thus the problems with decidability of sort computation (see the discussion in **??**) are mainly a theoretical concern.

The resolution calculus $\Sigma\mathcal{HR}$ finds its concretization in the $\mathcal{LEO}$ (**L**ogic **E**ngine for Ω-MKRP) theorem prover also currently under development in Saarbrücken. This implementation is currently been used to test the practical applicability of our calculus and for the development of search strategies specialized to higher-order logic. Naturally unification in $\Sigma\mathcal{HOL}$ is considerably more complex than in the simply typed λ-calculus, but this complexity is more than compensated by the restriction of search spaces associated with resolution theorem proving.

These applications in higher-order deduction for mathematics are not the only conceivable ones. For instance, $\Sigma\mathcal{HOL}$ can be seen as a logical basis for sorted logic programming languages, such as extensions to λ-Prolog [MN87, Mil89] in the spirit of TEL [Smo89] or Godel [HL94]. Because of undecidability issues, term declarations would have to be severely restricted, e.g. to higher-order patterns, in order to make sort computation decidable. Even with this restriction Σ-unification is still undecidable and infinitary (because the first-order subcase is), but it seems probable that there are decidable subcases analogous to those for first-order systems (cf. [Soc93, Uri92]).

Finally, our work can be seen as a guide for adding sort information to other typed λ-calculi. In this respect our work has applications in the field of meta-logical frameworks (λ-calculi with strong type systems that are used to formalize logical systems), since the added expressivity makes practical formalizations of logic systems much more convenient. It is still unclear, whether our focus on an extensional partial function semantics is advantageous for logical frameworks. The features of $\Sigma\mathcal{HOL}$ like functional base sorts or term declarations can, however, be adapted to existing sort systems for logical frameworks [Pfe93b, KP93] as well.

Bibliography

[ABI$^+$96a] Peter B. Andrews, Matthew Bishop, Sunil Issar, Dan Nesmith, Frank Pfenning, and Hongwei Xi. TPS: A theorem proving system for classical type theory. *Journal of Automated Reasoning*, 16(3):321–353, 1996.

[ABI$^+$96b] Peter B. Andrews, Matthew Bishop, Sunil Issar, Dan Nesmith, Frank Pfenning, and Hongwei Xi. TPS: A theorem-proving system for classical type theory. *Journal of Automated Reasoning*, 16:321–353, 1996.

[AKG93] Hassan Aït-Kaci and Jacques Garrigue. Label-selective lambda-calculus: Syntax and confluence. In *Proceedings of the 13th International Conference on Foundations of Software Technology and Theoretical Computer Science*, Vol.761 of *LNCS*, Bombay, India, 1993.

[AKR00] Alessandro Armando, Michael Kohlhase, and Silvio Ranise. Communication protocols for mathematical services based on KQML and OMRS. In Manfred Kerber and Michael Kohlhase, eds., *Proceedings of the 8th Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (Calculemus-2000)*, St. Andrews, Scotland, 2000. AKPeters. forthcoming.

[ALCMP84] Peter B. Andrews, Eve Longini-Cohen, Dale Miller, and Frank Pfenning. Automating higher order logics. *Contemp. Math*, 29:169–192, 1984.

[And71] Peter B. Andrews. Resolution in type theory. *Journal of Symbolic Logic*, 36(3):414–432, 1971.

[And72a] Peter B. Andrews. General models and extensionality. *Journal of Symbolic Logic*, 37(2):395–397, 1972.

[And72b] Peter B. Andrews. General models descriptions and choice in type theory. *Journal of Symbolic Logic*, 37(2):385–394, 1972.

[And73] Peter B. Andrews, 1973. letter to Roger Hindley dated January 22, 1973.

[And81] Peter B. Andrews. Theorem proving via general matings. *Journal of the Association for Computing Machinery*, 28(2):193–214, April 1981.

[And86] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Academic Press, 1986.

[And89] Peter B. Andrews. On Connections and Higher Order Logic. *Journal of Automated Reasoning*, 5:257–291, 1989.

[Bar84] H. P. Barendregt. *The Lambda Calculus*. North Holland, 1984.

[Bax78] L. D. Baxter. The undecidability of the third order dyadic unification problem. *Information and Control*, 38(2), 1978.

[BCF+97] C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, E. Melis, A. Meier, W. Schaarschmidt, J. Siekmann, and V. Sorge. ΩMEGA: Towards a mathematical assistant. In William McCune, ed., *Proceedings of the 14th Conference on Automated Deduction*, no.1249 in LNAI, pp.252–255, Townsville, Australia, 1997. Springer Verlag.

[BEG+64] J. H. Bennet, W. B. Easton, J. R. Guard, D. B. Loveman, and T. H. Mott. Semi-automated mathematics: SAM IV. Scientific Report 64–827, Air Force Cambridge Research Laboratories, October 1964.

[Ben97] Christoph Benzmüller. A calculus and a system architecture for extensional higher-order resolution. Research Report 97-198, Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh,USA, June 1997.

[Ber41] Paul Bernays. A system of axiomatic set-theory. *Journal of Symbolic Logic*, 6:1–17, 1941.

[Bet55] E. W. Beth. Semantic entailmentand formla derivability. *Medelingen von de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde*, 18(13):309–342, 1955.

[Bet69] E. W. Beth. Semantic entailmentand formla derivability. In J. Hintikka, ed., *The Philosophy of Mathematics*, pp.9–49. Oxford University Press, 1969. First published in [Bet55].

[BG98] Leo Bachmair and Harald Ganzinger. Equational reasoning in saturation-based theorem proving. In Bibel and Schmitt [BS98], chap. 11, pp.353–397.

[BHP+92] Christoph Beierle, U. Hedtstück, U. Pletat, P. Schmitt, and J. Siekmann. An order sorted logic for knowlege representation. *Journal of Artificial Intelligence*, 55:149–191, 1992.

[Bib82] Wolfgang Bibel. *Automated Theorem Proving*. Vieweg, Braunschweig, 1982.

[Bib83] Wolfgang Bibel. Matings in matrices. *Communications of the ACM*, 26:844–852, 1983.

[BK97a] Christoph Benzmüller and Michael Kohlhase. Model existence for higher-order logic. SEKI-Report SR-97-09, Universität des Saarlandes, 1997. submitted to JSL.

[BK97b] Christoph Benzmüller and Michael Kohlhase. Resolution for Henkin models. SEKI-Report SR-97-10, Universität des Saarlandes, 1997.

[BK98] Christoph Benzmüller and Michael Kohlhase. LEO – a higher order theorem prover. In Claude Kirchner and Hélène Kirchner, eds., *Proceedings of the 15th Conference on Automated Deduction*, no.1421 in LNAI, pp.139–144. Springer Verlag, 1998.

[BL90] Kim B. Bruce and Giuseppe Longo. A modest model of records, inheritance and bounded quantification. *Information and Computation*, 87:196–240, 1990.

[Ble77] W. W. Bledsoe. Set variables. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI)*, pp.501–509. Morgan Kaufmann, San Mateo, California, USA, 1977.

[Ble79] W. W. Bledsoe. A maximal method for set variables in automatic theorem proving. *Machine Intelligence*, 9:53–99, 1979.

[BLM+86] Robert Boyer, Ewing Lusk, William McCune, Ross Overbeek, Mark Stickel, and Lawrence Wos. Set theory for first-order logic: Clauses for Gödel's axioms. *Journal of Automated Reasoning*, 2:287–327, 1986.

[BS94] Franz Baader and Jörg Siekmann. Unification theory. In Dov Gabbay, ed., *Logic in Artificial Intelligence and Logic Programming*. Oxford University Press, 1994.

[BS98] Wolfgang Bibel and Peter Schmitt, eds.. *Automated Deduction – A Basis for Applications*. Kluwer, 1998.

[BS00] Franz Baader and Wayne Snyder. Unification theory. In J.A. Robinson and A. Voronkov, eds., *Handbook of Automated Reasoning*. Elsevier Science Publishers, 2000. forthcoming.

[Bun94] Alan Bundy, ed.. *Proceedings of the 12th Conference on Automated Deduction*, no.814 in LNAI, Nancy, France, 1994. Springer Verlag.

[CAB+86] Robert L. Constable, S. Allen, H. Bromly, W. Cleaveland, J. Cremer, R. Harper, D. Howe, T. Knoblock, N. Mendler, P. Panangaden, J. Sasaki, and S. Smith. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[Car84] Luca Cardelli. A semantics of multiple inheritance. In G. Kahn and G. Plotkin D.G. MacQueen, eds., *Semantics of Data Types*, no.173 in LNCS. Springer Verlag, 1984.

[Car88] Luca Cardelli. A semantics of multiple inheritance. *Information and Computation*, 76:138–164, 1988.

[CF58] H. B. Curry and R. Feys. *Combinatory Logic, Volume 1*. North Holland, 1958.

[CF92] Anthony G. Cohn and Alan M. Frisch. An abstract view of sorted unification. In Kapur [Kap92], pp.178–192.

[CG91] Pierre-Louis Curien and Giorgio Ghelli. Subtyping + extensionality: Confluence of $\beta\eta$-top reduction in $f_{\leq}$. In T. Ito and A. R. Meyer, eds., *Theoretical Aspects of Computer Science*, no.526 in LNCS. Springer Verlag, 1991.

[CH85] Thierry Coquand and Gérard Huet. A theory of constructions. In *Semantics of Data Types*. Springer Verlag, 1985.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[Coh87] Anthony G. Cohn. A more expressive forumlation of many sorted logic. *Journal of Autmated Reasoning*, 3:113–200, 1987.

[Coh89] Anthony G. Cohn. Taxonomic reasoning with many-sorted logics. *Artificial Intelligence Review*, 3:89–128, 1989.

[Coh92] Anthony G. Cohn. A many sorted logic with possibly empty sorts. In Kapur [Kap92], pp.633–647.

[CQ94] Régis Curien and Zhenyu Qian. Modular second-order E-matchning for regular theories, 1994. draft.

[Cur93] Régis Curien. Second order E-matching as a tool for automated theorem proving. In Miguel Filgueiras and Luís Damas, eds., *Progress in Artificial Intelligence, 6th Protuguese Conference on AI, EPIA '93*, no.727 in Lecture Notes in Artificial Intelligence, pp.242–257, Porto, Portugal, October 1993. Springer Verlag.

[Dar68] J. L. Darlington. Automatic theorem porving with equality substitutions and mathematical induction. *Machine Intelligence*, 3:113–130, 1968.

[Dar71] J. L. Darlington. A partial mechanization of second order logic. *Machine Intelligence*, 6:91–100, 1971.

[dB72] Nicolaas Govert de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with an application to the Church-Rosser theorem. *Indagationes Mathematicae*, 34(5):381–392, 1972.

[dB80] Nicolaas Govert de Bruijn. A survey of the project AUTOMATH. In R. Hindley and J. Seldin, eds., *To H.B. Curry: Essays in Combinator Logic, Lambda Calculus and Formalisms*, pp.579–606. Academic Press, 1980.

[dB94] N. G. de Bruijn. The mathematical vernacular, a language for mathematics with typed sets. In R. P Nederpelt, J. H. Geuvers, and R. C. de Vrijer, eds., *Selected Papers on Automath*, Vol.133 of *Studies in Logic and the Foundations of Mathematics*, pp.865 − 935. Elsevier, 1994.

[DGHP99] Marcello D'Agostino, Dov Gabbay, Reiner Hähnle, and Joachim Posegga. *Handbook of Tableau Methods*. Kluwer, 1999.

[DHK95] Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Higher-order unification via explicit substitutions. In Dexter Kozen, ed., *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science (LICS-10)*, pp.366–374, San Diego, California, June 1995. IEEE Computer Society Press.

[DHKP96] Gilles Dowek, Thérèse Hardin, Claude Kirchner, and Frank Pfenning. Unification via explicit substitutions: The case of higher-order patterns. In M. Maher, ed., *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pp.259–273, Bonn, Germany, 1996. MIT Press.

[Dig79] Vincent J. Digricoli. Resolution by unificatoin and equality. In William H. Joyner, ed., *Proceedings of the 4th Workshop on Automated Deduction*, Austin, Texas, USA, 1979.

[DJ92] Daniel Dougherty and Patricia Johann. A combinatory logic approach to higher-order E-unification. In Kapur [Kap92], pp.79–93.

[Dou93] Daniel Dougherty. Higher-order unification using combinators. *Theoretical Computer Science B*, 114(2):273–298, 1993.

[Dow92] Gilles Dowek. Third order matching is decidable. In *Proceedings of the 7th Annual IEEE Symposium on Logic in Computer Science (LICS-7)*, pp.2–10, Santa Cruz, Ca, USA, 1992. IEEE Computer Society Press.

[DSP91a] Mary Dalrymple, Stuart Shieber, and Fernando Pereira. Ellipsis and higher-order-unification. *Linguistics and Philosophy*, 14:399–452, 1991.

[DSP91b] Mary Dalrymple, Stuart Shieber, and Fernando Pereira. Ellipsis and higher-order unification. *Linguistics & Philosophy*, 14:399–452, 1991.

[Dyc92] Roy Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *The Journal of Symbolic Logic*, 57(3):795–807, 1992.

[Ebb77] H.D. Ebbinghaus. *Einführung in die Mengenlehre*. Wissenschaftliche Buchgesellschaft, 1977.

[Ern71] G. W. Ernst. A matching procedure for type theory. Tech. rep., Case Western Reserve University, 1971.

[Far91a] William M. Farmer. Simple second-order languages for which unification is undecidable. *Theoretical Computer Science*, 87(251):25–41, 1991.

[Far91b] William M. Farmer. A simple type theory with partial functions and sub-types. Technical report, MITRE Corporation, Bedford, MA01730 USA, 1991.

[Far93] William M. Farmer. Theory interpretation in simple type theory. In *HOA'93, an International Workshop on Higher-order Algebra, Logic and Term Rewriting*, Vol.816 of *LNCS*, Amsterdam, The Netherlands, 1993. Springer Verlag.

[FGT93] William M. Farmer, Joshua D. Guttman, and F. Javier Thayer. IMPS: An Interactive Mathematical Proof System. *Journal of Automated Reasoning*, 11(2):213–248, October 1993.

[FHJ+99] Andreas Franke, Stephan M. Hess, Christoph G. Jung, Michael Kohlhase, and Volker Sorge. Agent-oriented integration of distributed mathematical services. *Journal of Universal Computer Science*, 5:156–187, 1999.

[Fit90] Melvin Fitting. *First-Order Logic and Automated Threorem Proving*. Springer Verlag, 1990.

[FK99] Andreas Franke and Michael Kohlhase. System description: MathWeb, an agent-based communication layer for distributed automated theorem proving. In Harald Ganzinger, ed., *Automated Deduction — CADE-16*, no.1632 in LNAI, pp.217–221. Springer Verlag, 1999.

[FK00] Andreas Franke and Michael Kohlhase. System description: MBase, an open mathematical knowledge base. In David McAllester, ed., *Automated Deduction – CADE-17*, no.1831 in LNAI, pp.455–459. Springer Verlag, 2000.

[Fra22] Adolf Abraham Fraenkel. Zu den Grundlagen der Cantor-Zermeloschen Mengenlehre. *Mathematische Annalen*, 86:230–237, 1922.

[Fra28] Adolf Abraham Fraenkel. Zusatz zu vorstehendem Aufsatz Herrn v. Neumanns. *Mathematische Annalen*, 99:392–393, 1928.

[Fri90] Alan M. Frisch. The substitutional framework for sorted deduction: Fundamental results on hybrid reasoning. *Artificial Intelligence*, 49:161–198, 1990.

[Gar97] Claire Gardent. Sloppy identity. In Christian Retoré, ed., *Logical Aspects of Computational Linguistics*, pp.188–207. Springer, 1997.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen I & II. *Mathematische Zeitschrift*, 39:176–210, 572–595, 1935.

[GK96a] Claire Gardent and Michael Kohlhase. Focus and higher–order unification. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, 1996.

[GK96b] Claire Gardent and Michael Kohlhase. Higher–order coloured unification and natural language semantics. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz, 1996. ACL.

[GK97] Claire Gardent and Michael Kohlhase. Computing parallelism in discourse. In Martha E. Pollack, ed., *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, pp.1016–1021, Nagoya, Japan, 1997. Morgan Kaufmann.

[GKK99] Claire Gardent, Michael Kohlhase, and Karsten Konrad. Higher–order coloured unification: a linguistic application. *Techniques Sciences Informatiques*, pp.1–28, 1999.

[GKvL96] Claire Gardent, Michael Kohlhase, and Noor van Leusen. Corrections and Higher-Order Unification. In *Proceedings of KONVENS'96*, pp.268–279, Bielefeld, Germany, 1996. De Gruyter.

[GLT89] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. No. 7 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1989.

[GM93] M. J. C. Gordon and T. F. Melham. *Introduction to HOL – A theorem proving environment for higher order logic*. Cambridge University Press, 1993.

[GOBS69] J. R. Guard, F.C. Oglesby, J.H. Bennet, and L. G. Settle. Semi-automated mathematics. *Journal of the Association of Computing Machinery*, 16(1):49–62, 1969.

[Göd30] Kurt Gödel. Die Vollständigkeit der Axiome des logischen Funktionenkalküls. *Monatshefte für Mathematik und Physik*, 37:349–360, 1930. English Version in [vH67].

[Göd31] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte der Mathematischen Physik*, 38:173–198, 1931. English Version in [vH67].

[Göd40] Kurt Gödel. *The Consistency of the Axiom of Choice and of the Generalized Continuum-Hypothesis with the Axioms of Set Theory*, Vol. 3 of *Annals of Mathematics Studies*. Princeton University Press, Princeton, New Jersey; eighth printing 1970, 1940.

[Gol81] Warren D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.

[Gor85] Mike Gordon. HOL: a machine oriented formulation of higher-order logic. Tech. rep. 68, University of Cambridge, Computer Laboratory, July 1985.

[Gou66] William Eben Gould. A matching procedure for ω-order logic. Tech. rep., Applied Logic Corporation, One Palmer Square, Princeton, NJ, 1966.

[Hen50] Leon Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15(2):81–91, 1950.

[Hen96] Leon Henkin. The discovery of my completeness proofs. *The Bulletin of Symbolic Logic*, 2(2):127–158, 1996.

[Her30] Jaques Herbrand. *Recherches sur la théorie de la démonstration*. Ph.D. thesis, Université de Paris, 1930. Englisch translation in [vH67].

[Hib73] Günter Hibsch. Ansatz für ein mechanisches Beweisverfahren für die Prädikatenlogik zweiter Stufe mit Anwendungen auf die Zahlentheorie. Master Thesis, TU-München, 1973.

[Hic96] Jason J. Hickey. Formal objects in type theory using very dependent types. In *Foundations of of Object Oriented Languages 3*, 1996.

[Hil04] David Hilbert. Über die Grundlagen der Logik und der Arithmetik. In *Verhandlungen des Dritten Internationalen Mathematiker-Kongress in Heidelberg*, pp.174–185. Teubner, Leibzig, 1904.

[Hin55] K. J. J. Hintikka. Form and content in quantification theory. *Acta Philosophica Fennica*, 8:7–55, 1955.

[HK00] Dieter Hutter and Michael Kohlhase. Managing structural information by higher-order colored unification. *Journal of Automated Reasoning*, 25(2):123–164, 2000.

[HL94] Patricia Hill and John Lloyd. *The Goedel Programming Language*. Logic Programming series. MIT Press, 1994.

[How96] Douglas Howe. Semantic foundations for embedding hol in nuprl. In Martin Wirsing and Maurice Nivat, eds., *Algebraic Methodolgy and Software Technology*, Vol.1101 of *LNCS*, pp.85–101. Springer Verlag, 1996.

[HP91] Gérard Huet and Gordon Plotkin, eds.. *Logical Frameworks*. Cambridge University Press, 1991.

[HRS90] Maritta Heisel, Wolfgang Reif, and Werner Stephan. Tactical Theorem Proving in Program Verification. In Mark Stickel, ed., *Proceedings of the 10th Conference on Automated Deduction*, no.449 in LNCS, pp.115–131, Kaiserslautern, Germany, 1990. Springer Verlag.

[HRS91] Maritta Heisel, Wolfgang Reif, and Werner Stephan. *Automating Software Design*, chap. 21, pp.547–574. AAAI Press, 1991.

[HS86] J. Hindley and J. Seldin. *Introduction to Combinators and Lambda Calculus*. Cambridge University Press, 1986.

[Hue72] Gérard P. Huet. *Constrained Resolution: A Complete Method for Higher Order Logic*. Ph.D. thesis, Case Western Reserve University, 1972.

[Hue73a] Gérard P. Huet. A mechanization of type theory. In Walker and Norton [WN73], pp.139–146.

[Hue73b] Gérard P. Huet. The undecidability of unification in third order logic. *Information and Control*, 22(3):257–267, 1973.

[Hue75] Gérard P. Huet. An unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1:27–57, 1975.

[Hue76] Gérard P. Huet. *Résolution d'Équations dans des Langages d'ordre 1,2,...,w*. Thèse d'État, Université de Paris VII, 1976.

[JK93] Patricia Johann and Michael Kohlhase. Unification in an extensional lambda calculus with ordered function sorts and constant overloading. SEKI-Report SR-93-14, Universität des Saarlandes, 1993.

[JK94] Patricia Johann and Michael Kohlhase. Unification in an extensional lambda calculus with ordered function sorts and constant overloading. In Bundy [Bun94], pp.620–634.

[Joh91] Patricia Johann. *Complete Sets of Transformations for Unification Problems*. Ph.D. thesis, Wesleyan University, 1991.

[Joh93] Patricia Johann. A combinator-based order-sorted higher-order unification algorithm. SEKI-Report SR-93-16, Universität des Saarlandes, 1993.

[JP73] D. Jensen and T. Pietrzykowski. Mechanizing ω-order type theory through unification. Internal Report CS-73-13, Department of Applied Analysis and Computation, University of Waterloo, 1973.

[JP76] D. C. Jensen and T. Pietrzykowski. Mechanizing ω-order type theory through unification. *Theoretical Computer Science*, 3:123–171, 1976.

[Jut79] L.S. van Benthem Jutting. *Checking Landau's "Grundlagen" in the AUTOMATH System*, Vol. 83 of *Mathematical Centre Tracts*. Mathematisch Centrum, Amsterdam, Netherlands, 1979.

[Kap92] D. Kapur, ed.. *Proceedings of the 11th Conference on Automated Deduction*, Vol.607 of *LNCS*, Saratoga Spings, NY, USA, 1992. Springer Verlag.

[KB70] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In J. Leech, ed., *Computational Problems in Abstract Algebra*, pp.263–297. Pergamon Press, 1970.

[Ker91] Manfred Kerber. How to prove higher order theorems in first order logic. In John Mylopoulos and Ray Reiter, eds., *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI)*, pp.137–142, Sydney, Australia, 1991. Morgan Kaufmann, San Mateo, CA.

[KF01] Michael Kohlhase and Andreas Franke. MBase: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Computation; Special Issue on the Integration of Computer algebra and Deduction Systems*, 32(4):365–402, September 2001.

[Kir90] Claude Kirchner, ed.. *Unification*. Academic Press, London, 1990.

[Kle52] Stephen C. Kleene. *Introduction to Meta-Mathematics*. North Holland, 1952.

[Kle96] Gerald Klein. Unifikation in einem sortiertem λ-Kalkül mit Termdeklarationen und Schnittsorten. Master Thesis, FB Informatik, Universität des Saarlandes, 1996.

[Koh93] Michael Kohlhase. Higher-order resolution with combinators. In J. Avenhaus and J. Denzinger, eds., *Informal Proceedings fo the Annual Meeting of "GI-Fachgruppe 'Deduktlionssysteme'" in Kaiserslautern, 1993*, no.SR-93-11 (SFB) in SEKI-Report, p. 15, 1993.

[Koh94] Michael Kohlhase. *A Mechanization of Sorted Higher-Order Logic Based on the Resolution Principle*. Ph.D. thesis, Universität des Saarlandes, 1994.

[Koh95] Michael Kohlhase. Higher-Order Tableaux. In P. Baumgartner, R. Hähnle, and J. Posegga, eds., *Theorem Proving with Analytic Tableaux and Related Methods*, Vol.918 of *LNAI*, pp.294–309, 1995.

[Koh98] Michael Kohlhase. Higher-order automated theorem proving. In Bibel and Schmitt [BS98], pp.431–462.

[KP93] Michael Kohlhase and Frank Pfenning. Unification in a λ-calculus with intersection types. In Dale Miller, ed., *Proceedings of the International Logic Programming Sympsion. ILPS'93*, pp.488–505. MIT Press, 1993.

[Lan30] Edmund Landau. *Grundlagen der Analysis*. Wissenschaftliche Buchgesellschaft, Darmstadt, Germany; second edition, 1930. Reprint of the edition, Leipzig, 1970.

[Luc72] Claudio. L. Lucchesi. The undecidability of the unification problem for third order languages. Report CSRR 2059, University of Waterloo, Waterloo, Canada, 1972.

[Mak77] G. S. Makanin. The problem of solvabiliy of equations in a free semigroup. *Math. USSR Sbornik*, 32(2):129–198, 1977.

[McC97] William McCune. Solution of the robbins problem. *Journal of Automated Reasoning*, 19(3):263–276, 1997.

[Mil83] Dale Miller. *Proofs in Higher-Order Logic*. Ph.D. thesis, Carnegie-Mellon University, 1983.

[Mil87] Dale Miller. A compact representation of proofs. *Studia Logica*, 46(4):345–368, 1987.

[Mil89] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. In Peter Schroeder-Heister, ed., *Extensions of Logic Programming: International Workshop, Tübingen FRG, December 1989*, pp.253–281. Springer-Verlag LNCS 475, 1989.

[Mil91] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *Journal of Logic and Computation*, 4(1):497–536, 1991.

[Mil92] Dale Miller. Unification under a mixed prefix. *Journal of Symbolic Computation*, 14:321–358, 1992.

[ML94] Per Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1994.

[MM73] A. Martinelli and U. Montanari. An efficient unification algorithm. In Walker and Norton [WN73].

[MN87] Dale Miller and Gopalan Nadathur. A logic programming approach to manipulating formulas and programs. In *IEEE Symposium on Logic Programming*, Salt Lake City, 1987.

[Mon74] R. Montague. The proper treatment of quantification in ordinary english. In R. Montague, ed., *Formal Philosophy. Selected Papers*. Yale University Press, New Haven, 1974.

[Mor69] James B. Morris. E-resolution. In Donald E. Walker and Lewis Norton, eds., *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, pp.287–294, 1969.

[Mül93] Olaf Müller. Optimierung der modularen E-Unifikation höherer Stufe. Master Thesis, Universität Karlsruhe, Germany, May 1993.

[MW94] Olaf Müller and Franz Weber. Theory and practice of minimal modular higher-order E-unification. In Bundy [Bun94], pp.650–677.

[Neu28] John von Neumann. Die Axiomatisierung der Mengenlehre. *Mathematische Zeitschrift*, 27:669–752, 1928.

[Nip91] Tobias Nipkow. Higher-order critical pairs. In *Proceedings of the 6th Annual IEEE Symposium on Logic in Computer Science (LICS-6)*, pp.342–349, Amsterdam, NL, 1991. IEEE Computer Society Press.

[NM94] Gopalan Nadathur and Dale Miller. Higher-order logic programming. Technical Report CS-1994-38, Department of Computer Science, Duke University, 1994. To appear in Volume 5 of the Handbook of Logic in AI and Logic Programming.

[NQ91] Tobias Nipkow and Zhenyu Qian. Modular higher-order E-unification. In Ronald V. Book, ed., *Proceedings of the 4^{th} International Conference on Rewriting Techniques and Applications*, no.488 in LNCS, pp.200–214. Springer Verlag, 1991.

[NQ92] Tobias Nipkow and Zhenyu Qian. Reduction and unification in lambda calculi with subtypes. In Kapur [Kap92], pp.66–78.

[Obe62] Arnold Oberschelp. Untersuchungen zur mehrsortigen Quantorenlogik. *Mathematische Annalen*, 145:297–333, 1962.

[Oho95] Atsushi Ohori. A polymorphic record calculus and its compilation. *ACM Transactions on Programming Languages and Systems*, 17(6):844–895, 1995.

[ORS92] S. Owre, J. M. Rushby, and N. Shankar. PVS: a prototype verification system. In Kapur [Kap92], pp.748–752.

[OS89] Hans Jürgen Ohlbach and Jörg Siekmann. The Markgraf Karl Refutation Procedure. In *Computational Logic – Essays in Honor of Alan Robinson*, pp.41–112. MIT Press, Cambridge, 1989.

[Pad96] V. Padovani. *Filtrage d'order supérieur*. Thése de doctorat, Université Paris VII, 1996.

[Pau93] Lawrence C. Paulson. Set theory for verification: I. from foundations to functions. *Journal of Automated Reasoning*, 11:353–389, 1993.

[Pau94] Lawrence C. Paulson. *Isabelle: A Generic Theorem Prover*. LNCS. Springer Verlag, 1994.

[Pfe87] F. Pfenning. *Proof Transformations in Higher-Order Logic*. Ph.D. thesis, Carnegie-Mellon University, Pittsburgh, PA, 1987.

[Pfe91] Frank Pfenning. Logic programming in the LF logical framework. In Gérard P. Huet and Gordon D. Plotkin, eds., *Logical Frameworks*. Cambridge University Press, 1991.

[Pfe92] Frank Pfenning. Intersection types for a logical framework. POP-Report 92–106, Carnegie Mellon University, 1992.

[Pfe93a] Frank Pfenning. Automated theorem proving. Notes for the course Computation and Deduction, School of Computer Science, Carnegie Mellon University, Spring 1993.

[Pfe93b] Frank Pfenning. Refinement types for logical frameworks. In Herman Geuvers, ed., *Informal Proceedings of the 1993 Workshop on Types for Proofs and Programs*, pp.285–301, Nijmegen, The Netherlands, May 1993. University of Nijmegen.

[Pie73] Thomasz Pietrzykowski. A complete mechanization of second-order type theory. *Journal of the Association for Computing Machinery*, 20:333–364, 1973.

[Pie91] Bencamin C. Pierce. *Programming with Intersection Types and Bounded Polymorphism*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University, December 1991.

[Plo72] G. Plotkin. Building in equational theories. *Machine Intelligence*, 7:73–90, 1972.

[PN90] Lawrence C. Paulson and Tobias Nipkow. Isabelle tutorial and user's manual. Tech. rep.189, Computer Laboratory, University of Cambridge, January 1990.

[Pra60] Dag Prawitz. An improved proof procedure. *Theoria*, 26:102–139, 1960.

[Pre94a] Christian Prehofer. Decidable higher-order unification problems. In Bundy [Bun94], pp.635–649.

[Pre94b] Christian Prehofer. Solving higher-order equations. In *Proceedings of the 9th Annual IEEE Symposium on Logic in Computer Science (LICS-9)*, pp.507–516, Paris, France, 1994. IEEE Computer Society Press.

[Pul97] Steve Pulman. Higher order unification and the interpretation of focus. *Linguistics & Philosophy*, 20:73–115, 1997.

[QED95] The QED manifesto. Internet Report `http://www.rbjones.com/rbjpub/logic/qedres00.htm`, 1995.

[Qia91] Zhenyu Qian. *Extensions of Order-Sorted Algebraic Specifications: Parameterization, Higher-Functions and Polymorphism.* Ph.D. thesis, Universität Bremen, März 1991.

[Qia93] Zhenyu Qian. Linear unification of higher-order patterns. In J.-P. Jouannaud M.-C. Gaudel, ed., *Proceedings of TAPSOFT(CAAP)'93*, no.668 in LNCS, pp.391–405. Springer Verlag, 1993.

[Qia96] Zhenyu Qian. Unification of higher-order patterns in linear time and space. *Journal of Logic and Computation*, 6(3):315–341, 1996.

[Qua92] Art Quaife. Automated deduction in von Neumann-Bernays-Gödel set theory. *Journal of Automated Reasoning*, 8(1):91–148, 1992.

[QW94] Zhenyu Qian and Kang Wang. Modular AC unification of higher-order patterns, 1994. draft.

[Ree87] S. Reeves. Semantic tableaux as a framework for automated theorem-proving. In J. Hallam and C. Mellish, eds., *Advances in Artificial Intelligence, AISB-87*, pp.125–139. Wiley, 1987.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12(1):23–41, 1965.

[Rob68] J. A. Robinson. New directions in theorem proving. In *Proceedings of IFIP Congress in Information Processing*, Vol. 68, pp.63–67. North Holland, Amsterdam, 1968.

[Rob69a] J. A. Robinson. Mechanizing higher order logic. *Machine Intelligence*, 4:151–170, 1969.

[Rob69b] J. A. Robinson. A note on mechanizing higher order logic. *Machine Intelligence*, 5:121–134, 1969.

[Rus08] Bertrand Russell. Mathematical logic as based on the theory of types. *American Journal of Mathematics*, XXX:222–262, 1908.

[RW69] Arthur Robinson and Larry Wos. Paramodulation and TP in first order theories with equality. *Machine Intelligence*, 4:135–150, 1969.

[Sch24] Moses Schönfinkel. Über die Bausteine der mathematischen Logik. *Mathematische Annalen*, 92:305–316, 1924. Englisch Version entitled: On the building blocks of mathematical logic in [vH67].

[Sch38] A. Schmidt. Über deduktive Theorien mit mehreren Sorten von Grunddingen. *Mathematische Annalen*, 115, 1938.

[Sch51] A. Schmidt. Die Zulässigkeit der Behandlung mehrsortiger Theorien mittels der üblichen einsortigen prädikatenlogik. *Mathematische Annalen*, 123, 1951.

[Sch60] Kurt Schütte. Semantical and syntactical properties of simple type theory. *Journal of Symbolic Logic*, 25:305–326, 1960.

[Sko19] Albert Thoralf Skolem. Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit oder Beweisbarkeit mathematischer Sätze. *Videnskapkasselskapets Skrifter*, I:1–36, 1919. English translation in [vH67].

[Smo89] Gert Smolka. *Logic Programming over Polymorphically Order-Sorted Types*. Ph.D. thesis, Universität Kaiserslautern, 1989.

[Smu63] Raymond M. Smullyan. A unifying principle for quantification theory. *Proc. Nat. Acad Sciences*, 49:828–832, 1963.

[Smu68] Raymond M. Smullyan. *First-Order Logic*. Springer Verlag, 1968.

[SNGM87] Gert Smolka, Werner Nutt, Joseph A. Goguen, and José Meseguer. Order-sorted equational computation. SEKI-Report SR-87-14, Universität Kaiserslautern, 1987.

[Sny90] Wayne Snyder. Higher order E-unification. In Mark Stickel, ed., *Proceedings of the 10th Conference on Automated Deduction*, no.449 in LNCS, pp.573–578, Kaiserslautern, Germany, 1990. Springer Verlag.

[Sny91] Wayne Snyder. *A Proof Theory for General Unification*. Progress in Computer Science and Applied Logic. Birkhäuser, 1991.

[Soc93] Rolf Socher. Unification in order-sorted logic with term declarations. In Andrei Voronkov, ed., *Proceedings of the International Conference on Logic Programming and Automated Reasoning LPAR'93*, Vol.698 of *LNAI*, pp.301–308, St. Petersburg, Russia, 1993. Springer Verlag.

[SS86] Manfred Schmidt-Schauß. Unification in many-sorted equational theories. In J. Siekmann, ed., *Proceedings of the 8th Conference on Automated Deduction*, no.230 in LNCS, pp.538–552, Oxford, England, 1986. Springer Verlag.

[SS87] Manfred Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations*. Ph.D. thesis, University of Kaiserslautern, 1987. Also [SS89].

[SS89] Manfred Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations*, Vol.395 of *LNAI*. Springer Verlag, 1989.

[Sta85] R. Statman. Logical relations and the typed lambda calculus. *Information and Computation*, 65, 1985.

[Sut00] Geoff Sutcliffe. The cade-16 atp system competition. *Journal of Automated Reasoning*, 24(3):371–396, 2000.

[Sza69] M.E. Szabo, ed.. *The Collected Papers of Gerhard Gentzen*. North-Holland, 1969.

[Tai67] W. Tait. Intensional interpretation of functionals of finite type I. *Information and Computation*, 32:198–212, 1967.

[Tak53] Gaisi Takeuti. On a generalized logic calculus. *Japan Journal of Mathematics*, 23:39 f., 1953.

[Tak68] Moto-o Takahashi. Cut-elimination in simple type theory with extensionality. *Journal of the Mathematical Society of Japan*, 19, 1968.

[Tak87] Gaisi Takeuti. *Proof Theory*. North Holland, 1987.

[Tho91a] Simon Thompson. *Type Theory and Functional Programming*. International Computer Science Series. Addison-Wesley, 1991.

[Tho91b] Simon Thompson. *Type Theory and Functional Programming*. International Computer Science Series. Addison Wesley, 1991.

[Uri92] T. E. Uribe. Sorted unification using set constraints. In Kapur [Kap92], pp.163–177.

[vH67] Jean van Heijenoort, ed.. *From Frege to Gödel A Source Book in Mathematical Logic, 1879-1931*. Source Books in the History of the Sciences. Harvard University Press, 1967.

[Wal83] Christoph Walther. A many-sorted calculus based on resolution and paramodulation. In Alan Bundy, ed., *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pp.882–891, Los Altos, California, USA, August 1983. William Kaufmann.

[Wal84] Christoph Walther. Unification in many-sorted theories. In T. O'Shea, ed., *Proceedings of of the 7th European Conference on Artifical Intelligence*, pp.593–602, Pisa, Italy, 1984. Pitman Publishing, London, England.

[Wal85] Christoph Walther. A mechanical solution of Schubert's steamroller by many-sorted resolution. *Artificial Intelligence*, 26(2):217–224, 1985.

[Wal87] Christoph Walther. *A Many-Sorted Calculus Based on Resolution and Paramodulation*. Pitman, London. Morgan Kaufman Publishers, Inc, 1987.

[Wal88] Christoph Walther. Many-sorted unification. *Journal of the Accociation for Computing Machinery*, 35(1):1–17, January 1988.

[Wan52] Hao Wang. Logic of many-sorted theories. *Journal of Symbolic Logic*, 17, 1952.

[Web93] Franz Weber. *Softwareentwicklung mit Logik höherer Stufe*. Ph.D. thesis, Universität Karlsruhe, Germany, July 1993.

[Wei91] Christoph Weidenbach. A sorted logic using dynamic sorts. Technical Report MPI-I-91-218, Max-Planck-Institut für Informatik, Saarbrücken, Germany, 1991.

[Wei95] Christoph Weidenbach. First-order tableaux with sorts. *Journal of the Interest Group in Pure and Applied Logics, IGPL*, 3(6):887–906, 1995.

[Wei96a] Christoph Weidenbach. *Computational Aspects of a First-Order Logic with Sorts*. Ph.D. thesis, Universtiät des Saarlandes, Germany, 1996.

[Wei96b] Christoph Weidenbach. Unification in pseudo-linear sort theories is decidable. In M.A. McRobbie and J.K. Slaney, eds., *Proceedings of the 13th Conference on Automated Deduction*, no.1104 in LNAI, pp.343–357, New Brunswick, NJ, USA, 1996. Springer Verlag.

[Wir98] Claus-Peter Wirth. Full first-order sequent and tableau calculi with preservation of solutions and the liberalized delta-rule but without skolemization. In R. Caferra and G. Salzer, eds., *Automated Deduction in Classical and Non-Classical Logics (FTP'98)*, LNAI1761, pp.283–298. Springer Verlag, 1998.

[Wir00] Claus-Peter Wirth. Descente infinie + deduction. Technical Report 737/2000, FB Informatik, Universität Dortmund, 7, 2000.

[WN73] Donald E. Walker and Lewis Norton, eds.. *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, 1973.

[WO90] Christoph Weidenbach and Hans Jürgen Ohlbach. A resolution calculus with dynamic sort structures and partial functions. In Luigia Carlucci Aiello, ed., *Proceedings of of the 9th European Conference on Artifical Intelligence*, pp.688–693, Stockholm, Sweden, 1990. Pitman Publishing, London, England.

[Wol93] David A. Wolfram. *The Clausal Theory of Types*. Cambridge University Press, 1993.

[Wol94] David A. Wolfram. A semantics for λ-PROLOG. *Theoretical Computer Science*, 136(1), 1994.

[WR10] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica*, Vol. I. Cambridge University Press, Cambridge, Great Britain; second edition, 1910.

[Wu00] Wen-Tsun Wu. *Mechanical Geometry Theorem-Proving, Mechanical Geometry Problem-Solving, and Polynomial Equations-Solving.* Kluwer, 2000.

[Zai87] M. Zaionc. Word operation definable in the typed λ-calculus. *Theoretical Computer Science*, 52:1–14, 1987.

[Zer08] Ernst Zermelo. Untersuchungen über die Grundlagen der Mengenlehre. I. *Mathematische Annalen*, 65:261–281, 1908.

Draft
30/6/2004

Index

Herbrand
Skolem

Author Index

Here is the preable text

Defined Symbols