Knowledge Representation for Mathematical/Technical Knowledge
Summer 2017

Provisional Lecture Notes

Michael Kohlhase

Professur für Wissensrepräsentation und -verarbeitung
Informatik, FAU Erlangen-Nürnberg
Michael.Kohlhase@FAU.de

July 27, 2017

Preface

Course Concept

Aims: To give students a solid foundation of the basic concepts and practices in representing mathematical/technical knowledge, so they can do (guided) research in the KWARC group.

Prerequisites: The course builds on the logic courses in the FAU Bachelor's program, in particular the course "Grundlagen der Logik in der Informatik"[1] EdN:1

Course Contents

Goal: To give students a solid foundation of the basic concepts and practices in representing mathematical/technical knowledge, so they can do (guided) research in the KWARC group.

This Document

This document contains the course notes for the course Artificial Intelligence held at at FAU Erlangen in the winter semester 2016/17[1]

Format: The document mixes the slides presented in class with comments of the instructor to give students a more complete background reference.

 Caveat  : This document is made available for the students of this course only. It is still very much a draft and will develop over the course of the current course and in coming academic years.

Knowledge Representation Experiment: This document is also an experiment in knowledge representation. Under the hood, it uses the sTEX package [Koh08, Koh16], a TEX/LATEX extension for semantic markup, which allows to export the contents into the eLearning platform PantaRhei.

Comments and extensions are always welcome, please send them to the author.

Acknowledgments

Materials: All course materials have bee restructured and semantically annotated in the sTEX format, so that we can base additional semantic services on them (see slide 7 for details).

KRMT Students: The following students have submitted corrections and suggestions to this and earlier versions of the notes:

[1] EdNote: figure out what happens with Lutz' Ontologies course

[1] They are based on a course held at Jacobs University in Fall 2005.

Recorded Syllabus for SS 2017

In this document, we record the progress of the course in the summer semester 2017 in the form of a "recorded syllabus", i.e. a syllabus that is created after the fact rather than before.

Recorded Syllabus Summer Semester 2017:

#	date	until	slide	page
1	4. May	overview, some admin, math search	??	??
2	8. May	framing, theory graphs,content/form	??	??
3	11. May	$\mathbb{N}, +$ in MMT		

Contents

Chapter 1

Administrativa

General Admin Disclaimer

- ▷ This is the first time this course runs (we have to figure out what works best)
- ▷ I am still quite new to FAU (different system than I am used to)
  - ▷ at a private university for 13 years: Jacobs University Bremen
  - ▷ at a US University for three years: Carnegie Mellon University
  - ▷ Saarland University Saarbrücken (but that is soooo long ago)
- ⚠: Do not trust me on admin issues (rather help me understand)
- ⋙ I will do my best to learn quickly.
- ▷ We should apply the Reasonable Person Principle both ways!

We will now go through the ground rules for the course. This is a kind of a social contract between the instructor and the students. Both have to keep their side of the deal to make learning as efficient and painless as possible.

Prerequisites

- ▷ the mandatory courses from Semester 1-4, in particular: (or equivalent)
  - ▷ course "Grundlagen der Logik in der Informatik" (GLOIN)
  - ▷ CS Math courses "Mathematik C1-4" (IngMath1-4) (our "domain")
  - ▷ algorithms and data structures
  - ▷ course "Künstliche Intelligenz I" (nice-to-have only)
- ▷ Motivation, Interest, Curiosity, hard work
  - ▷ You can do this course if you want! (and we will help you)

Now we come to a topic that is always interesting to the students: the grading scheme.

Grades

- ▷ Academic Assessment: two parts (Portfolio Assessment)
  - ▷ 20-min oral exam at the end of the semester (50%)
  - ▷ results of the KRMT lab (50%)

KRMT Lab (Dogfooding our own Techniques)

- ▷ (generally) we use the thursday slot to get our hands dirty with actual representations.
- ▷ Instructor: Dennis Müller (dennis.mueller@fau.de) Room: 11.138, Tel: 85-64053
- ▷ Goal: Reinforce what was taught in class and have some fun
- ▷ Homeworks: will be small individual problem/programming/proof assignments (but take time to solve) group submission if and only if explicitly permitted
- ▷ Admin: To keep things running smoothly
  - ▷ Homeworks will be posted on course forum (discussed in the lab)
  - ▷ No "submission", but open development on a git repos. (details follow)
- ▷ Homework Discipline:
  - ▷ start early! (many assignments need more than one evening's work)
  - ▷ Don't start by sitting at a blank screen
  - ▷ Humans will be trying to understand the text/code/math when grading it.

Textbook, Handouts and Information, Forums

- ▷ (No) Textbook: Course notes will be posted at http://kwarc.info/teaching/KRMT
  - ▷ I mostly prepare them as we go along (semantically preloaded $\rightsquigarrow$ research resource)
  - ▷ please e-mail me any errors/shortcomings you notice. (improve for the group)
- ▷ Announcements will be posted on the course forum
  - ▷ https://fsi.cs.fau.de/forum/150-Logikbasierte-Wissensrepraesentation

- ▷ Check the forum frequently for
  - ▷ announcements, homeworks, questions
  - ▷ discussion among your fellow students

Do I need to attend the lectures

- ▷ Attendance is not mandatory for the KRMT lecture (official version)
- ▷ There are two ways of learning: (both are OK, your mileage may vary)
  - ▷ Approach B: Read a book/papers
  - ▷ Approach I: come to the lectures, be involved, interrupt me whenever you have a question.

  The only advantage of I over B is that books/papers do not answer questions
- ▷ Approach S: come to the lectures and sleep does not work!
- ▷ The closer you get to research, the more we need to discuss!

Next we come to a special project that is going on in parallel to teaching the course. I am using the coures materials as a research object as well. This gives you an additional resource, but may affect the shape of the coures materials (which now server double purpose). Of course I can use all the help on the research project I can get, so please give me feedback, report errors and shortcomings, and suggest improvements.

Experiment: E-Learning with KWARC Technologies

- ▷ My research area: deep representation formats for (mathematical) knowledge
- ▷ Application: E-learning systems (represent knowledge to transport it)
- ▷ Experiment: Start with this course (Drink my own medicine)
  - ▷ Re-Represent the slide materials in $OMDoc$ (Open Math Documents)
  - ▷ Feed it into the PantaRhei system (http://panta.kwarc.info)
  - ▷ Try it on you all (to get feedback from you)
- ▷ Tasks (Unfortunately, I cannot pay you for this; maybe later)
  - ▷ help me complete the material on the slides (what is missing/would help?)
  - ▷ I need to remember "what I say", examples on the board. (take notes)
- ▷ Benefits for you (so why should you help?)
  - ▷ you will be mentioned in the acknowledgements (for all that is worth)
  - ▷ you will help build better course materials (think of next-year's students)

Chapter 2

Overview over the Course

Plot of this Course

- Today: Motivation, Admin, and find out what you already know
  - What is logic, knowledge representation
  - What is mathematical/technical knowledge
  - how can you get involved with research at KWARC

2.1 Introduction & Motivation

Knowledge-Representation and -Processing

- **Definition 2.1.1 (True and Justified Belief)** Knowledge is a body of facts, theories, and rules available to persons or groups that are so well justified that their validity/truth is assumed.
- **Definition 2.1.2** Knowledge representation formulates knowledge in a formal language so that new knowledge can be induced by inferred via rule systems (inference).
- **Definition 2.1.3** We call an information system knowledge-based, if a large part of its behaviour is based on inference on represented knowledge.
- **Definition 2.1.4** The field of knowledge processing studies knowledge-based systems, in particular
  - compilation and structuring of explicit/implicit knowledge (knowledge acquisition)
  - formalization and mapping to realization in computers (knowledge representation)
  - processing for problem solving (inference)
  - presentation of knowledge (informationvisualization)

▷ knowledge representation and processing are subfields of symbolic artificial intelligence

Mathematical Knowledge (Representation and -Processing)

▷ KWARC (my research group) develops foundations, methods, and applications for the representation and processing of mathematical knowledge

- ▷ Mathematics plays a fundamental role in Science and Technology(practice with maths, apply in STEM)
- ▷ mathematical knowledge is rich in content, sophisticated in structure, and explicitly represented . . .
- ▷ . . . , and we know exactly what we are talking about (in contrast to economics or love)

Working Definition: Everything we understand well is "mathematics" (e.g. CS, Physics, . . .)

⋙ There is a lot of mathematical knowledge

- ▷ 120,000 Articles are published in pure/applied mathematics (3.5 millions so far)
- ▷ 50 Millionen science articles in 2010 [Jin10] with a doubling time of 8-15 years [Lvl10]
- ▷ 1 M Technical Reports on http://ntrs.nasa.gov/ (e.g. the Apollo reports)
- ▷ a Boeing-Ingenieur tells of a similar collection (but in Word 3,4,5,. . .)

About Humans and Computers in Mathematics

▷ Computers and Humans have complementary strengths.

- ▷ Computers can handle large data and computations flawlessly at enormous speeds.
- ▷ Humans can sense the environment, react to unforeseen circumstances and use their intuitions to guide them through only partially understood situations.

In mathematics: we exploit this, we

▷
- ▷ let humans explore mathematical theories and come up with novel insights/proofs,
- ▷ delegate symbolic/numeric computation and typesetting of documents to computers.

▷ (sometimes) delegate proof checking and search for trivial proofs to computers

Overlooked Opportunity: management of existing mathematical knowledge

▷ ▷ cataloguing, retrieval, refactoring, plausibilization, change propagation and in some cases even application do not require (human) insights and intuition

▷ can even be automated in the near future given suitable representation formats and algorithms.

Math. Knowledge Management (MKM): is the discipline that studies this.

⋙ Application: Scaling Math beyond the One-Brain-Barrier

The One-Brain-Barrier

▷ **Observation 2.1.5** *More than 10^5 math articles published annually in Math.*

▷ **Observation 2.1.6** *The libraries of Mizar, Coq, Isabelle,... have $\sim 10^5$ statements+proofs each.* *(but are mutually incompatible)*

▷ Consequence: humans lack overview over – let alone working knowledge in – all of math/formalizations. (Leonardo da Vinci was said to be the last who had)

▷ Dire Consequences: duplication of work and missed opportunities for the application of mathematical/formal results.

▷ Problem: Math Information systems like arXiv.org, Zentralblatt Math, MathSciNet, etc. do not help (only make documents available)

▷ Fundamenal Problem: the One-Brain Barrier (OBB)

▷ To become productive, math must pass through a brain

▷ Human brains have limited capacity (compared to knowledge available online)

▷ Idea: enlist computers (large is what they are good at)

▷ Prerequisite: make math knowledge machine-actionable & foundation-independent (use MKM)

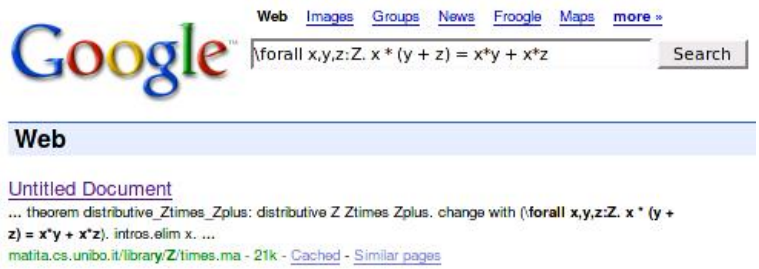
2.2 Mathematical Formula Search

More Mathematics on the Web

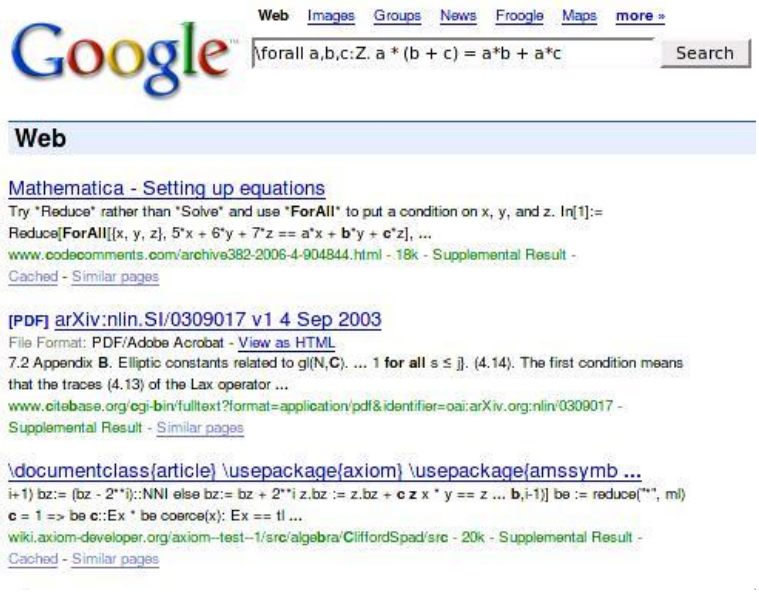
- ▷ The Connexions project (http://cnx.org)
- ▷ Wolfram Inc. (http://functions.wolfram.com)
- ▷ Eric Weisstein's MathWorld (http://mathworld.wolfram.com)
- ▷ Digital Library of Mathematical Functions (http://dlmf.nist.gov)
- ▷ Cornell ePrint arXiv (http://www.arxiv.org)
- ▷ Zentralblatt Math (http://www.zentralblatt-math.org)
- ▷ . . . Engineering Company Intranets, . . .
- ▷ Question: How will we find content that is relevant to our needs
- ▷ Idea: try Google (like we always do)
- ▷ Scenario: Try finding the distributivity property for $\mathbb{Z}$ ($\forall k,l,m \in \mathbb{Z}.k \cdot (l+m) = (k \cdot l) + (k+m)$)

Searching for Distributivity

Searching for Distributivity

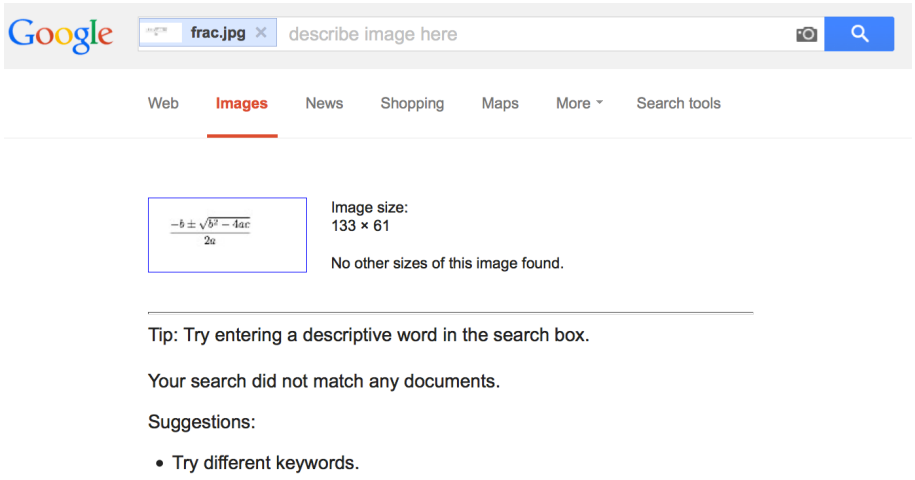


Searching for Distributivity



Does Image Search help?

▷ Math formulae are visual objects, after all (let's try it)



Of course Google cannot work out of the box

- ▷ Formulae are not words:
  - ▷ a, b, c, k, l, m, x, y, and z are (bound) variables. (do not behave like words/symbols)
  - ▷ where are the word boundaries for "bag-of-words" methods?
- ▷ Formulae are not images either: They have internal (recursive) structure and compositional meaning
- ▷ Idea: Need a special treatment for formulae (translate into "special words")
  Indeed this is done ([MY03, MM06, LM06, MG11])
  ... and works surprisingly well (using e.g. Lucene as an indexing engine)
- ▷ Idea: Use database techniques (extract metadata and index it)
  Indeed this is done for the `Coq/HELM` corpus ([AGC+06])
- ▷ Our Idea: Use Automated Reasoning Techniques (free term indexing from theorem prover jails)
- ▷ Demo: MathWebSearch on Zentralblatt Math, the arXiv Data Set

A running example: The Power of a Signal

- ▷ An engineer wants to compute the power of a given signal $s(t)$
- ▷ She remembers that it involves integrating the square of s.

▷ Problem: But how to compute the necessary integrals

▷ Idea: call up `MathWebSearch` with $\int_?^? s^2(t)dt$.

▷ `MathWebSearch` finds a document about Parseval's Theorem and $\frac{1}{T}\int_0^T s^2(t)dt = \Sigma_{k=-\infty}^{\infty}|c_k|^2$ where c_k are the Fourier coefficients of $s(t)$.

Some other Problems (Why do we need more?)

▷ Substitution Instances: search for $x^2 + y^2 = z^2$, find $3^2 + 4^2 = 5^2$

▷ Homonymy: $\binom{n}{k}$, ${}_nC^k$, C_k^n, and C_n^k all mean the same thing(binomial coeff.)

▷ Solution: use content-based representations (MathML, $OpenMath$)

▷ Mathematical Equivalence: e.g. $\int f(x)dx$ means the same as $\int f(y)dy$ (α-equivalence)

▷ Solution: build equivalence (e.g. α or ACI) into the search engine (or normalize first [Normann'06])

▷ Subterms: Retrieve formulae by specifying some sub-formulae

▷ Solution: record locations of all sub-formulae as well

`MathWebSearch`: Search Math. Formulae on the Web

▷ Idea 1: Crawl the Web for math. formulae (in $OpenMath$ or CMathML)

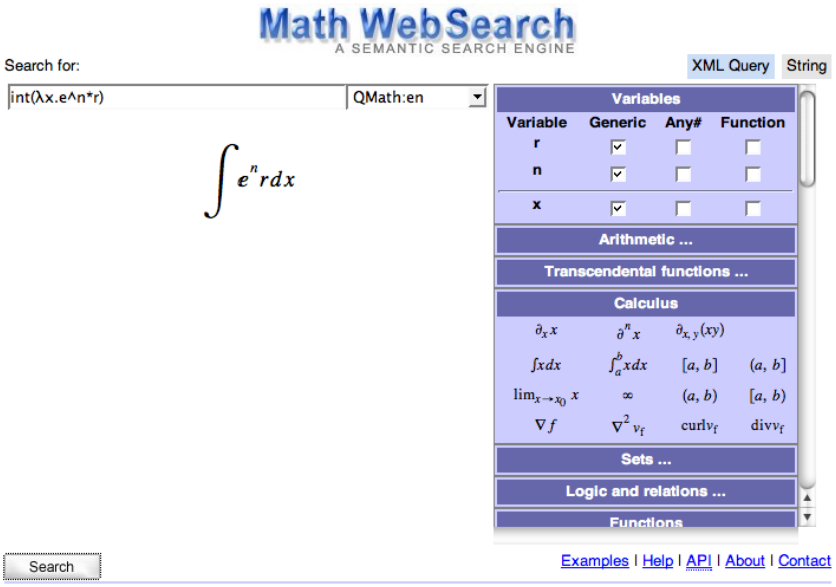
▷ Idea 2: Math. formulae can be represented as first order terms (see below)

▷ Idea 3: Index them in a substitution tree index (for efficient retrieval)

▷ Problem: Find a query language that is intuitive to learn

▷ Idea 4: Reuse the XML syntax of $OpenMath$ and CMathML, add variables

Statt einer Demo: Wir suchen ein Integral



Statt einer Demo: Suchresultate

Other integrals (5 formulas) (Source)

Other integrals (5 formulas)

Matched term:

$$\int \frac{e^{3z/4}}{\left(-2+e^{3z/4}\right)\sqrt{-2+e^{3z/4}+e^{3z/2}}}\,dz = \frac{2}{3}\left(\log\left(-2+e^{3z/4}\right)-\log\left(4\sqrt{-2+e^{3z/4}+e^{3z/2}}+5e^{3z/4}-2\right)\right)$$

Rank: 100%

XML Source

Used substitution:

$\mathbf{n} \to 3z4^{-1}$

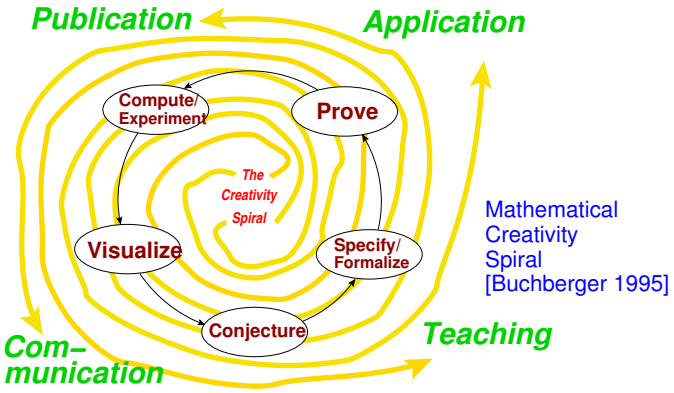
$\mathbf{r} \to \left(\left((-2)+e^{3z4^{-1}}\right)\left((-2)+e^{3z4^{-1}}+e^{3z2^{-1}}\right)^{1/2}\right)^{-1}$

$\mathbf{x} \to z$

2.3 The Mathematical Knowledge Space

The way we do math will change dramatically

▷ **Definition 2.3.1 (Doing Math)** Buchberger's Math creativity spiral



▷ Every step will be supported by mathematical software systems

▷ Towards an infrastructure for web-based mathematics!

Mathematical Literacy

▷ Note: the form and extent of knowledge representation for the components of "doing math" vary greatly. (e.g. publication vs. proving)

▷ **Observation 2.3.2 (Primitive Cognitive Actions)**
To "do mathematics", we need to

- ▷ *extract the relevant structures,*
- ▷ *reconcile them with the context of our existing knowledge*
- ▷ *recognize parts as already known*
- ▷ *identify parts that are new to us.*

During these processes mathematicians (are trained to)

- ▷ *abstract from syntactic differences, and*
- ▷ *employ interpretations via non-trivial, but meaning-preserving mappings*

▷ **Definition 2.3.3** We call the skillset that identifies mathematical training mathematical literacy (cf. Observation 2.3.2)

Introduction: Framing as a Mathematical Practice

▷ Understanding Mathematical Practices:

- ▷ To understand Math, we must understand what mathematicians do!
- ▷ The value of a math education is more in the skills than in the knowledge.
- ▷ Have been interested in this for a while (see [KK06])

▷ Framing: Understand new objects in terms of already understood structures. Make creative use of this perspective in problem solving.

▷ **Example 2.3.4** Understand point sets in 3-space as zeroes of polynomials. Derive insights by studying the algebraic properties of polynomials.

▷ **Definition 2.3.5** We are framing the point sets as algebraic varieties (sets of zeroes of polynomials).

▷ **Example 2.3.6 (Lie group)** Equipping a differentiable manifold with a (differentiable) group operation

▷ **Example 2.3.7 (Stone's representation theorem)** Interpreting a Boolean algebra as a field of sets.

▷ Claim: Framing is valuable, since it transports insights between fields.

▷ Claim: Many famous theorems earn their recognition *because* they establish profitable framings.

2.4 Modular Representation of mathematical Knowledge

Modular Representation of Math (Theory Graph)

▷ Idea: Follow mathematical practice of generalizing and framing

- ▹ framing: If we can view an object a as an instance of concept B, we can inherit all of B properties (almost for free.)
- ▹ state all assertions about properties as general as possible (to maximize inheritance)
- ▹ examples and applications are just special framings.

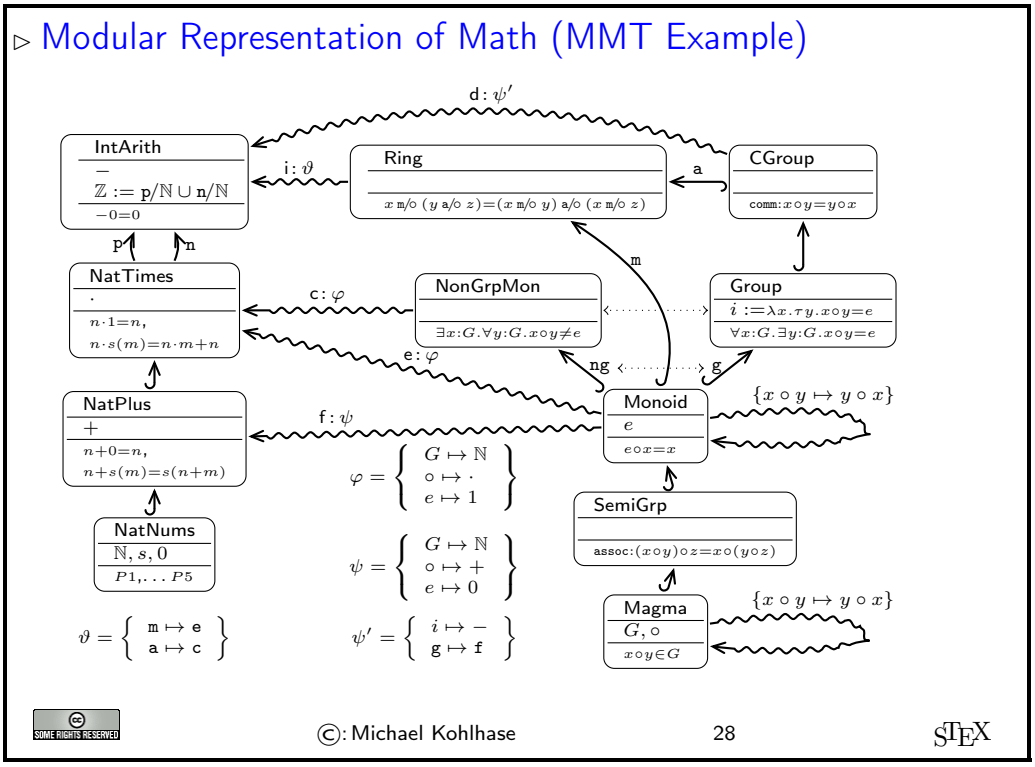
▷ Modern expositions of Mathematics follow this rule (radically e.g. in Bourbaki)

▷ formalized in the theory graph paradigm (little/tiny theory doctrine)

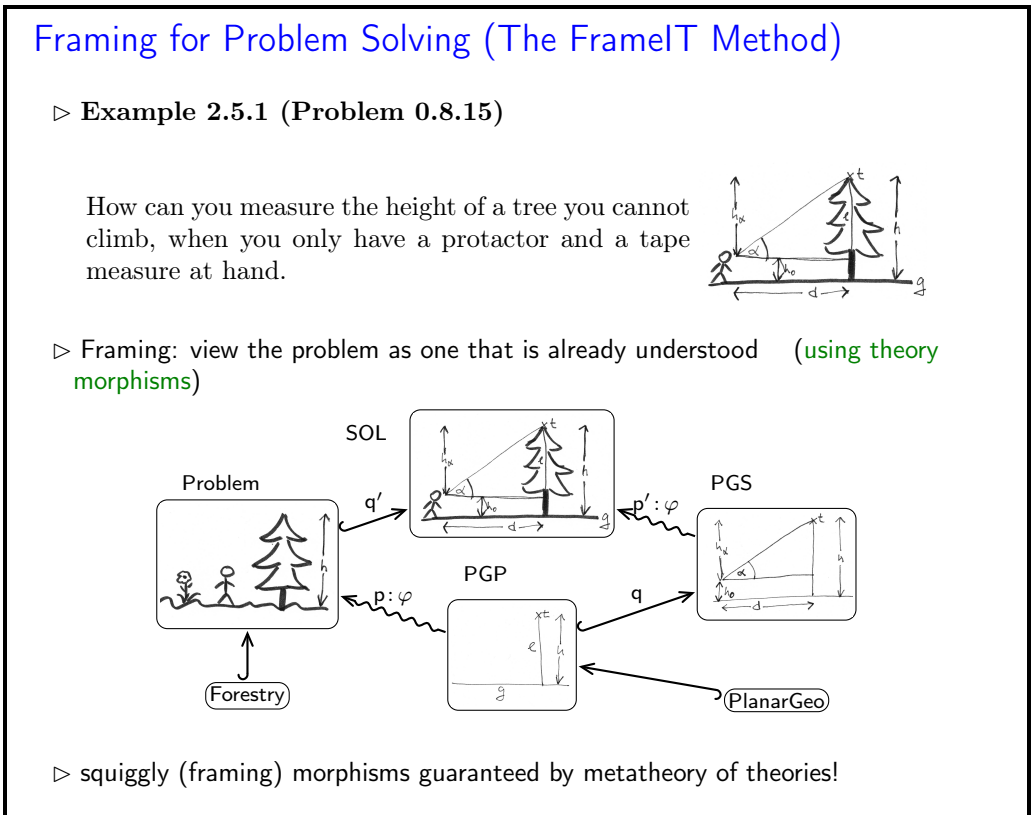
- ▹ theories as collections of symbol declarations and axioms (model assumptions)
- ▹ theory morphisms as mappings that translate axioms into theorems

▷ **Example 2.4.1 (MMT: Modular Mathematical Theories)** MMT is a foundation-indepent theory graph formalism with advanced theory morphisms.

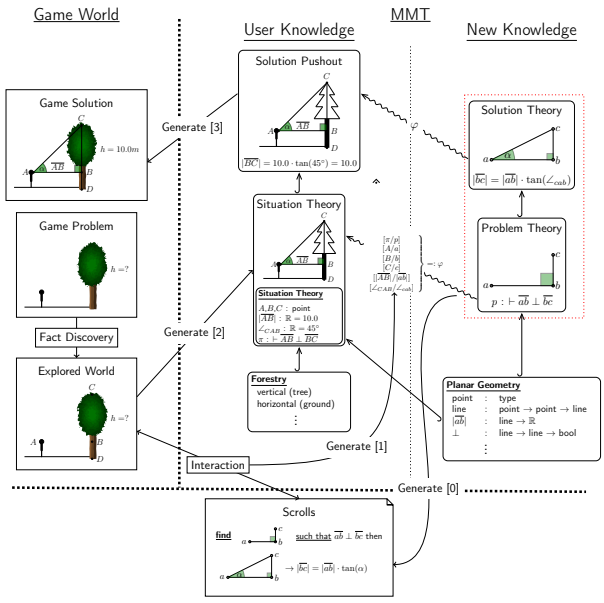
Problem: With a proliferation of abstract (tiny) theories readability and accessibility suffers (one reason why the Bourbaki books fell out of favor)



2.5 Application: Serious Games

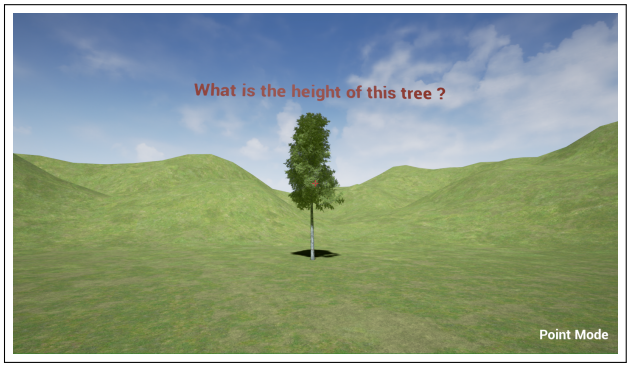


Example Learning Object Graph



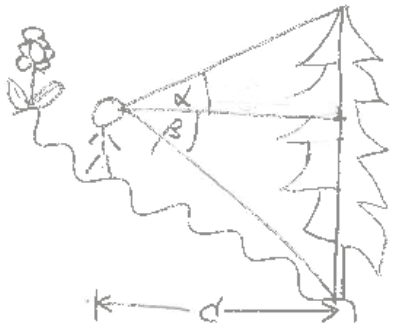
FrameIT: Problem Solving by Framing ⇝ Serious Games

▷ Implementation in custom web interface (Bachelor's thesis)



Currently: Minecraft mod (problem/solution pairs as objects that can be applied for automation)

▷ Combining Problem/Solution Pairs



▷ We can use the same mechanism for combining P/S pairs

▷ create more complex P/S pairs (e.g. for trees on slopes)

 ©: Michael Kohlhase 32 STEX

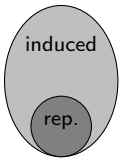
Another whole set of applications and game behaviors can come from the fact that LOGraphs give ways to combine problem/solution pairs to novel ones. Consider for instance the diagram on the right, where we can measure the height of a tree of a slope. It can be constructed by combining the theory SOL with a copy of SOL along a second morphism the inverts h to $-h$ (for the lower triangle with angle β) and identifies the base lines (the two occurrences of h_0 cancel out). Mastering the combination of problem/solution pairs further enhances the problem solving repertoire of the player.

2.6 Search in the Mathematical Knowledge Space

The Mathematical Knowledge Space

▷ **Observation 2.6.1** *The value of framing is that it induces new knowledge*

▷ **Definition 2.6.2** The mathematical knowledge space MKS is the structured space of represented and induced knowledge, mathematically literate have access to.



▷ Idea: make math systems mathematically literate by supporting the MKS

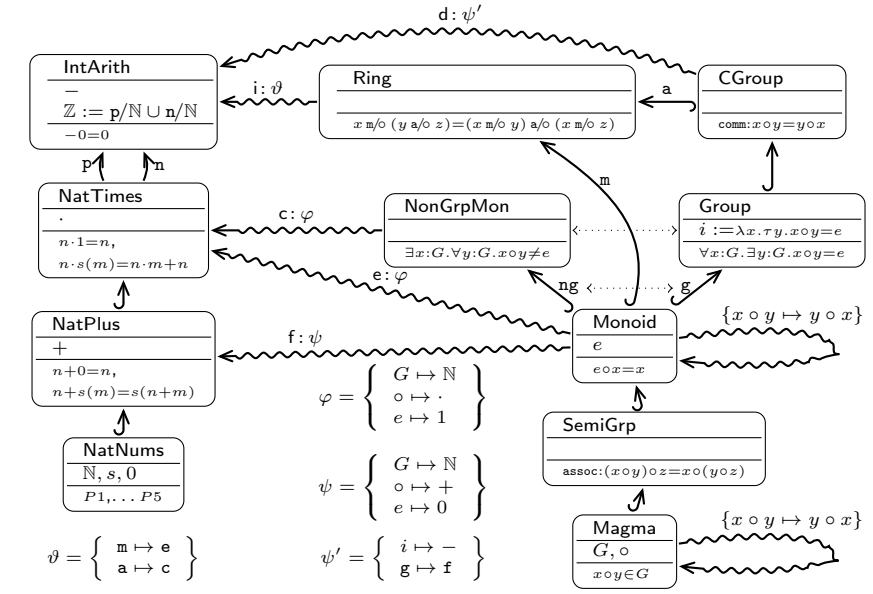
▷ In this talk: I will cover three aspects

- ▷ an approach for representing framing and the MKS (OMDoc/MMT)
- ▷ search modulo framing (MKS-literate search)
- ▷ a system for archiving the MKS (MathHub.info)

▷ Told from the Perspective of: searching the MKS

 ©: Michael Kohlhase 33 STEX

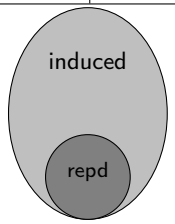
Modular Representation of Math (MMT Example)



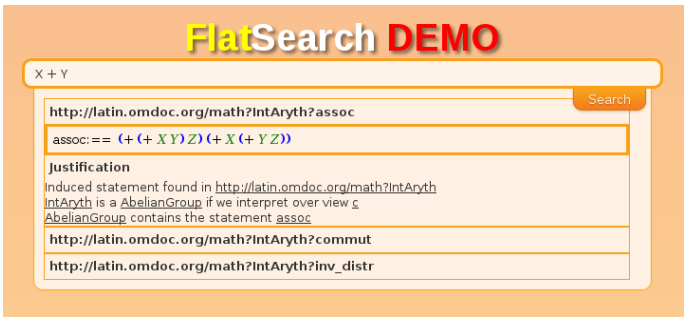
♭search on the LATIN Logic Atlas

▷ Flattening the LATIN Atlas (once):

type	modular	flat	factor
declarations	2310	58847	25.4
library size	23.9 MB	1.8 GB	14.8
math sub-library	2.3 MB	79 MB	34.3
MathWebSearch harvests	25.2 MB	539.0 MB	21.3



▷ simple ♭search frontend at http://cds.omdoc.org:8181/search.html



Overview: KWARC Research and Projects

Applications: eMath 3.0, Active Documents, Semantic Spreadsheets, Semantic CAD/CAM, Change Mangagement, Global Digital Math Library, Math Search Systems, SMGloM: Semantic Multilingual Math Glossary, Serious Games, ...

Foundations of Math:	**KM & Interaction**:	**Semantization**:
▷ MathML, $OpenMath$	▷ Semantic Interpretation (aka. Framing)	▷ LaTeXML: LaTeX → XML
▷ advanced Type Theories	▷ math-literate interaction	▷ sTeX: Semantic LaTeX
▷ MMT: Meta Meta Theory	▷ MathHub: math archives & active docs	▷ invasive editors
▷ Logic Morphisms/Atlas	▷ Semantic Alliance: embedded semantic services	▷ Context-Aware IDEs
▷ Theorem Prover/CAS Interoperability		▷ Mathematical Corpora
		▷ Linguistics of Math

Foundations: Computational Logic, Web Technologies, $OMDoc$/MMT

Take-Home Message

▷ Overall Goal: *Overcoming the "One-Brain-Barrier" in Mathematics* (by knowledge-based systems)

▷ Means: Mathematical Literacy by Knowledge Representation and Processing in theory graphs in Theoriegraphen. (Framing as mathematical practice)

Chapter 3

What is (Computational) Logic

What is (Computational) Logic?

- ▷ The field of logic studies representation languages, inference systems, and their relation to the world.
- ▷ It dates back and has its roots in Greek philosophy (Aristotle et al.)
- ▷ Logical calculi capture an important aspect of human thought, and make it amenable to investigation with mathematical rigour, e.g. in
  - ▷ foundation of mathematics (Hilbert, Russell and Whitehead)
  - ▷ foundations of syntax and semantics of language(Creswell, Montague, ...)
- ▷ Logics have many practical applications
  - ▷ logic/declarative programming (the third programming paradigm)
  - ▷ program verification: specify conditions in logic, prove program correctness
  - ▷ program synthesis: prove existence of answers constructively, extract program from proof
  - ▷ proof-carrying code: compiler proves safety conditions, user verifies before running.
  - ▷ deductive databases: facts + rules (get more out than you put in)
  - ▷ semantic web: the Web as a deductive database
- ▷ Computational Logic is the study of logic from a computational, proof-theoretic perspective. (model theory is mostly comprised under "mathematical logic".)

What is Logic?

- ▷ Logic $\widehat{=}$ formal languages, inference and their relation with the world
  - ▷ Formal language $\mathcal{FL}$: set of formulae $(2+3/7,\ \forall x.x+y=y+x)$
  - ▷ Formula: sequence/tree of symbols $(x, y, f, g, p, 1, \pi, \in, \neg, \wedge \forall, \exists)$

- ▷ Models: things we understand (e.g. number theory)
- ▷ Interpretation: maps formulae into models ($[\![\text{three plus five}]\!] = 8$)
- ▷ Validity: $\mathcal{M} \models \mathbf{A}$, iff $[\![\mathbf{A}]\!]^{\mathcal{M}} = \mathsf{T}$ (five greater three is valid)
- ▷ Entailment: $\mathbf{A} \models \mathbf{B}$, iff $\mathcal{M} \models \mathbf{B}$ for all $\mathcal{M} \models \mathbf{A}$. (generalize to $\mathcal{H} \models \mathbf{A}$)
- ▷ Inference rules to transform (sets of) formulae ($\mathbf{A}, \mathbf{A} \Rightarrow \mathbf{B} \vdash \mathbf{B}$)
- ▷ Syntax: formulae, inference (just a bunch of symbols)
- ▷ Semantics: models, interpr., validity, entailment (math. structures)

Important Question: relation between syntax and semantics?

So logic is the study of formal representations of objects in the real world, and the formal statements that are true about them. The insistence on a *formal language* for representation is actually something that simplifies life for us. Formal languages are something that is actually easier to understand than e.g. natural languages. For instance it is usually decidable, whether a string is a member of a formal language. For natural language this is much more difficult: there is still no program that can reliably say whether a sentence is a grammatical sentence of the English language.

We have already discussed the meaning mappings (under the monicker "semantics"). Meaning mappings can be used in two ways, they can be used to understand a formal language, when we use a mapping into "something we already understand", or they are the mapping that legitimize a representation in a formal language. We understand a formula (a member of a formal language) $\mathbf{A}$ to be a representation of an object $\mathcal{O}$, iff $[\![\mathbf{A}]\!] = \mathcal{O}$.

However, the game of representation only becomes really interesting, if we can do something with the representations. For this, we give ourselves a set of syntactic rules of how to manipulate the formulae to reach new representations or facts about the world.

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 now feasible for young children. What is the cause of this dramatic change? Of course the formalized reasoning procedures for arithmetic that we use nowadays. These *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 Wilhelm Schickard (1623), Blaise Pascal (1642), and Gottfried Wilhelm Leibniz (1671) was only a natural consequence.

We will see that it is not only possible to calculate with numbers, but also with representations of statements about the world (propositions). For this, we will use an extremely simple example; a fragment of propositional logic (we restrict ourselves to only one logical connective) and a small calculus that gives us a set of rules how to manipulate formulae.

3.1 A History of Ideas in Logic

Before starting with the discussion on particular logics and inference systems, we put things into perspective by previewing ideas in logic from a historical perspective. Even though the presentation (in particular syntax and semantics) may have changed over time, the underlying ideas are still pertinent in today's formal systems.

Many of the source texts of the ideas summarized in this Section can be found in [vH67].

▷ History of Ideas (abbreviated): Propositional Logic

▷ General Logic ([ancient Greece, e.g. Aristotle])

+ conceptual separation of syntax and semantics
+ system of inference rules ("Syllogisms")
– no formal language, no formal semantics

▷ Propositional Logic [Boole ~ 1850]

+ functional structure of formal language (propositions + connectives)
+ mathematical semantics ($\rightsquigarrow$ Boolean Algebra)
– abstraction from internal structure of propositions

History of Ideas (continued): Predicate Logic

▷ Frege's "Begriffsschrift" [Fre79]

+ functional structure of formal language (terms, atomic formulae, connectives, quantifiers)
– weird graphical syntax, no mathematical semantics
– paradoxes e.g. Russell's Paradox [R. 1901] (the set of sets that do not contain themselves)

▷ modern form of predicate logic [Peano $\sim$ 1889]

+ modern notation for predicate logic ($\vee, \wedge, \Rightarrow, \forall, \exists$)

History of Ideas (continued): First-Order Predicate Logic

▷ Types ([Russell 1908])

– restriction to well-types expression
+ paradoxes cannot be written in the system
+ Principia Mathematica ([Whitehead, Russell 1910])

▷ Identification of first-order Logic ([Skolem, Herbrand, Gödel $\sim$ 1920 – '30])

– quantification only over individual variables (cannot write down induction principle)
+ correct, complete calculi, semi-decidable
+ set-theoretic semantics ([Tarski 1936])

History of Ideas (continued): Foundations of Mathematics

▷ Hilbert's Program: find logical system and calculus, ([Hilbert ∼ 1930])

- ▷ that formalizes all of mathematics
- ▷ that admits sound and complete calculi
- ▷ whose consistence is provable in the system itself

▷ Hilbert's Program is impossible! ([Gödel 1931])

Let $\mathcal{L}$ be a logical system that formalizes arithmetics ($\langle NaturalNumbers, +, * \rangle$),

- ▷ then $\mathcal{L}$ is incomplete
- ▷ then the consistence of $\mathcal{L}$ cannot be proven in $\mathcal{L}$.

History of Ideas (continued): λ-calculus, set theory

▷ Simply typed λ-calculus ([Church 1940])

- + simplifies Russel's types, λ-operator for functions
- + comprehension as β-equality (can be mechanized)
- + simple type-driven semantics (standard semantics $\leadsto$ incompleteness)

▷ Axiomatic set theory

- +− type-less representation (all objects are sets)
- + first-order logic with axioms
- + restricted set comprehension (no set of sets)
- − functions and relations are derived objects

Part I

Formal Systems

To prepare the ground for the particular developments coming up, let us spend some time on recapitulating the basic concerns of formal systems.

Chapter 4

Logical Systems

The notion of a logical system is at the basis of the field of logic. In its most abstract form, a logical system consists of a formal language, a class of models, and a satisfaction relation between models and expressions of the formal lanugage. The satisfaction relation tells us when an expression is deemed true in this model.

Logical Systems

▷ **Definition 4.0.1** A logical system is a triple $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$, where $\mathcal{L}$ is a formal language, $\mathcal{K}$ is a set and $\models \subseteq \mathcal{K} \times \mathcal{L}$. Members of $\mathcal{L}$ are called formulae of $\mathcal{S}$, members of $\mathcal{K}$ models for $\mathcal{S}$, and $\models$ the satisfaction relation.

▷ **Definition 4.0.2** Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system, $\mathcal{M} \in \mathcal{K}$ be a model and $\mathbf{A} \in \mathcal{L}$ a formula, then we call $\mathbf{A}$

- ▷ satisfied by $\mathcal{M}$, iff $\mathcal{M} \models \mathbf{A}$
- ▷ falsified by $\mathcal{M}$, iff $\mathcal{M} \not\models \mathbf{A}$
- ▷ satisfiable in $\mathcal{K}$, iff $\mathcal{M} \models \mathbf{A}$ for some model $\mathcal{M} \in \mathcal{K}$.
- ▷ valid in $\mathcal{K}$ (write $\models \mathcal{M}$), iff $\mathcal{M} \models \mathbf{A}$ for all models $\mathcal{M} \in \mathcal{K}$
- ▷ falsifiable in $\mathcal{K}$, iff $\mathcal{M} \not\models \mathbf{A}$ for some $\mathcal{M} \in \mathcal{K}$.
- ▷ unsatisfiable in $\mathcal{K}$, iff $\mathcal{M} \not\models \mathbf{A}$ for all $\mathcal{M} \in \mathcal{K}$.

Entailment

▷ **Definition 4.0.3** Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system, then we define the entailment relation $\models \subseteq \mathcal{L}^* \times \mathcal{L}$. We say that a set $\mathcal{H} \subseteq \mathcal{L}$ of formulae entails $\mathbf{B}$ (written $\mathcal{H} \models \mathbf{B}$), iff we have $\mathcal{M} \models \mathbf{B}$ for all $\mathbf{A} \in \mathcal{H}$ and models $\mathcal{M} \in \mathcal{K}$ with $\mathcal{M} \models \mathbf{A}$.

▷ **Observation 4.0.4 (Entailment conserves Validity)** *If* $\mathbf{A} \models \mathbf{B}$ *and* $\mathcal{M} \models \mathbf{A}$, *then* $\mathcal{M} \models \mathbf{B}$.

▷ **Observation 4.0.5 (Entailment is monotonic)** *If* $\mathcal{H} \models \mathbf{B}$ *and* $\mathcal{H} \subseteq \mathcal{K}$, *then* $\mathcal{K} \models \mathbf{B}$.

Example 4.0.6 (First-Order Logic as a Logical System) Let $\mathcal{L} := \mathit{wff}_o(\Sigma)$, $\mathcal{K}$ be the class of first-order models, and $\mathcal{M} \models \mathbf{A} :\Leftrightarrow \mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$, then $\langle \mathcal{L}, \mathcal{K}, \models \rangle$ is a logical system in the sense of Definition 4.0.1.

Note that central notions like the entailment relation (which is central for understanding reasoning processes) can be defined independently of the concrete compositional setup we have used for first-order logic, and only need the general assumptions about logical systems.

Let us now turn to the syntactical counterpart of the entailment relation: derivability in a calculus. Again, we take care to define the concepts at the general level of logical systems.

Chapter 5

Calculi, Derivations, and Proofs

The intuition of a calculus is that it provides a set of syntactic rules that allow to reason by considering the form of propositions alone. Such rules are called inference rules, and they can be strung together to derivations — which can alternatively be viewed either as sequences of formulae where all formulae are justified by prior formulae or as trees of inference rule applications. But we can also define a calculus in the more general setting of logical systems as an arbitrary relation on formulae with some general properties. That allows us to abstract away from the homomorphic setup of logics and calculi and concentrate on the basics.

Derivation Systems and Inference Rules

▷ **Definition 5.0.1** Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system, then we call a relation $\vdash \subseteq \mathcal{P}(\mathcal{L}) \times \mathcal{L}$ a derivation relation for $\mathcal{S}$, if it

- ▷ is proof-reflexive, i.e. $\mathcal{H} \vdash \mathbf{A}$, if $\mathbf{A} \in \mathcal{H}$;
- ▷ is proof-transitive, i.e. if $\mathcal{H} \vdash \mathbf{A}$ and $\mathcal{H}' \cup \{\mathbf{A}\} \vdash \mathbf{B}$, then $\mathcal{H} \cup \mathcal{H}' \vdash \mathbf{B}$;
- ▷ monotonic (or admits weakening), i.e. $\mathcal{H} \vdash \mathbf{A}$ and $\mathcal{H} \subseteq \mathcal{H}'$ imply $\mathcal{H}' \vdash \mathbf{A}$.

▷ **Definition 5.0.2** We call $\langle \mathcal{L}, \mathcal{K}, \models, \vdash \rangle$ a formal system, iff $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ is a logical system, and $\vdash$ a derivation relation for $\mathcal{S}$.

▷ **Definition 5.0.3** Let $\mathcal{L}$ be a formal language, then an inference rule over $\mathcal{L}$

$$\frac{\mathbf{A}_1 \quad \cdots \quad \mathbf{A}_n}{\mathbf{C}}\mathcal{N}$$

where $\mathbf{A}_1, \ldots, \mathbf{A}_n$ and $\mathbf{C}$ are formula schemata for $\mathcal{L}$ and $\mathcal{N}$ is a name. The $\mathbf{A}_i$ are called assumptions, and $\mathbf{C}$ is called conclusion.

▷ **Definition 5.0.4** An inference rule without assumptions is called an axiom (schema).

▷ **Definition 5.0.5** Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system, then we call a set $\mathcal{C}$ of inference rules over $\mathcal{L}$ a calculus for $\mathcal{S}$.

With formula schemata we mean representations of sets of formulae, we use boldface uppercase letters as (meta)-variables for formulae, for instance the formula schema $\mathbf{A} \Rightarrow \mathbf{B}$ represents the set of formulae whose head is $\Rightarrow$.

Derivations and Proofs

▷ **Definition 5.0.6** Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system and $\mathcal{C}$ a calculus for $\mathcal{S}$, then a $\mathcal{C}$-derivation of a formula $\mathbf{C} \in \mathcal{L}$ from a set $\mathcal{H} \subseteq \mathcal{L}$ of hypotheses (write $\mathcal{H} \vdash_{\mathcal{C}} \mathbf{C}$) is a sequence $\mathbf{A}_1, \ldots, \mathbf{A}_m$ of $\mathcal{L}$-formulae, such that

- ▷ $\mathbf{A}_m = \mathbf{C}$, (derivation culminates in $\mathbf{C}$)
- ▷ for all $1 \leq i \leq m$, either $\mathbf{A}_i \in \mathcal{H}$, or (hypothesis)
- ▷ there is an inference rule $\dfrac{\mathbf{A}_{l_1} \quad \cdots \quad \mathbf{A}_{l_k}}{\mathbf{A}_i}$ in $\mathcal{C}$ with $l_j < i$ for all $j \leq k$. (rule application)

Observation: We can also see a derivation as a tree, where the $\mathbf{A}_{l_j}$ are the children of the node $\mathbf{A}_k$.

⋙ **Example 5.0.7** In the propositional Hilbert calculus $\mathcal{H}^0$ we have the derivation $P \vdash_{\mathcal{H}^0} Q \Rightarrow P$: the sequence is $P \Rightarrow Q \Rightarrow P, P, Q \Rightarrow P$ and the corresponding tree on the right.

$$\dfrac{\dfrac{}{P \Rightarrow Q \Rightarrow P}\,K \qquad P}{Q \Rightarrow P}\,MP$$

▷ **Observation 5.0.8** *Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system and $\mathcal{C}$ a calculus for $\mathcal{S}$, then the $\mathcal{C}$-derivation relation $\vdash_{\mathcal{D}}$ defined in Definition 5.0.6 is a derivation relation in the sense of Definition 5.0.1.*[2]

▷ **Definition 5.0.9** We call $\langle \mathcal{L}, \mathcal{K}, \models, \mathcal{C} \rangle$ a formal system, iff $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ is a logical system, and $\mathcal{C}$ a calculus for $\mathcal{S}$.

▷ **Definition 5.0.10** A derivation $\emptyset \vdash_{\mathcal{C}} \mathbf{A}$ is called a proof of $\mathbf{A}$ and if one exists (write $\vdash_{\mathcal{C}} \mathbf{A}$) then $\mathbf{A}$ is called a $\mathcal{C}$-theorem.

▷ **Definition 5.0.11** an inference rule $\mathcal{I}$ is called admissible in $\mathcal{C}$, if the extension of $\mathcal{C}$ by $\mathcal{I}$ does not yield new theorems.

[b] EDNOTE: MK: this should become a view!

Inference rules are relations on formulae represented by formula schemata (where boldface, uppercase letters are used as meta-variables for formulae). For instance, in Example 5.0.7 the inference rule $\dfrac{\mathbf{A} \Rightarrow \mathbf{B} \quad \mathbf{A}}{\mathbf{B}}$ was applied in a situation, where the meta-variables $\mathbf{A}$ and $\mathbf{B}$ were instantiated by the formulae P and $Q \Rightarrow P$.

As axioms do not have assumptions, they can be added to a derivation at any time. This is just what we did with the axioms in Example 5.0.7.

Chapter 6

Properties of Calculi

In general formulae can be used to represent facts about the world as propositions; they have a semantics that is a mapping of formulae into the real world (propositions are mapped to truth values.) We have seen two relations on formulae: the entailment relation and the deduction relation. The first one is defined purely in terms of the semantics, the second one is given by a calculus, i.e. purely syntactically. Is there any relation between these relations?

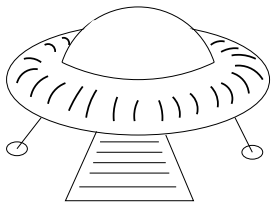
Soundness and Completeness

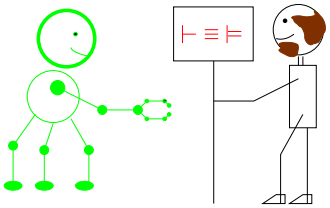
▷ **Definition 6.0.1** Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system, then we call a calculus $\mathcal{C}$ for $\mathcal{S}$

- ▷ sound (or correct), iff $\mathcal{H} \models \mathbf{A}$, whenever $\mathcal{H} \vdash_{\mathcal{C}} \mathbf{A}$, and
- ▷ complete, iff $\mathcal{H} \vdash_{\mathcal{C}} \mathbf{A}$, whenever $\mathcal{H} \models \mathbf{A}$.

▷ Goal: $\vdash \mathbf{A}$ iff $\models \mathbf{A}$ (provability and validity coincide)

- ▷ To TRUTH through PROOF (CALCULEMUS [Leibniz ∼1680])





Ideally, both relations would be the same, then the calculus would allow us to infer all facts that can be represented in the given formal language and that are true in the real world, and only those. In other words, our representation and inference is faithful to the world.

A consequence of this is that we can rely on purely syntactical means to make predictions about the world. Computers rely on formal representations of the world; if we want to solve a problem on our computer, we first represent it in the computer (as data structures, which can be seen as a formal language) and do syntactic manipulations on these structures (a form of calculus). Now, if the provability relation induced by the calculus and the validity relation coincide (this will be quite difficult to establish in general), then the solutions of the program will be correct, and we will find all possible ones.

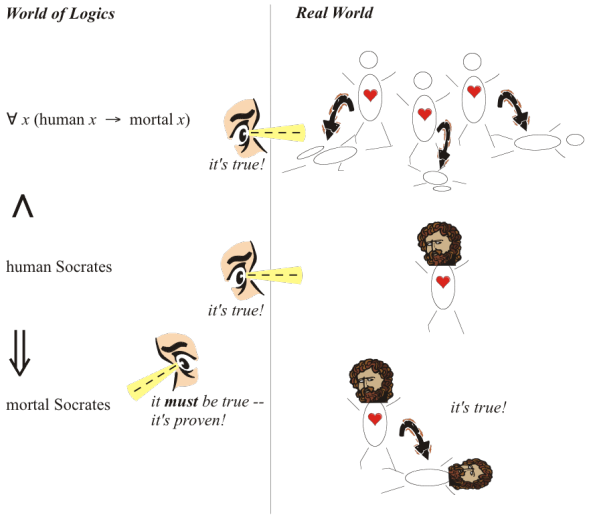
Of course, the logics we have studied so far are very simple, and not able to express interesting facts about the world, but we will study them as a simple example of the fundamental problem of Computer Science: How do the formal representations correlate with the real world.

Within the world of logics, one can derive new propositions (the *conclusions*, here: *Socrates is mortal*) from given ones (the *premises*, here: *Every human is mortal* and *Sokrates is human*). Such derivations are *proofs*.

In particular, logics can describe the internal structure of real-life facts; e.g. individual things, actions, properties. A famous example, which is in fact as old as it appears, is illustrated in the slide below.

The miracle of logics

▷ Purely formal derivations are true in the real world!



If a logic is correct, the conclusions one can prove are true (= hold in the real world) whenever the premises are true. This is a miraculous fact (think about it!)

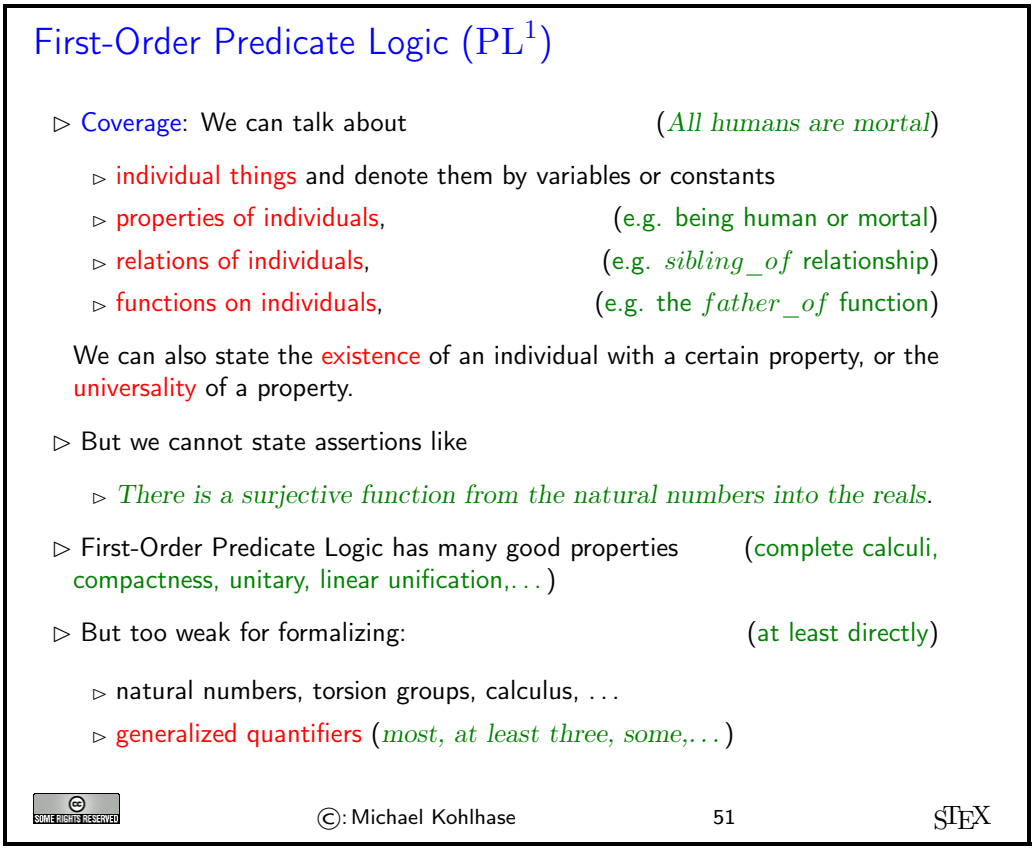
Part II

First-Order Logic and Inference

Chapter 7

First-Order Logic

First-order logic is the most widely used formal system for modelling knowledge and inference processes. It strikes a very good bargain in the trade-off between expressivity and conceptual and computational complexity. To many people first-order logic is "the logic", i.e. the only logic worth considering, its applications range from the foundations of mathematics to natural language semantics.



We will now introduce the syntax and semantics of first-order logic. This introduction differs from what we commonly see in undergraduate textbooks on logic in the treatment of substitutions in the presence of bound variables. These treatments are non-syntactic, in that they take the renaming of bound variables (α-equivalence) as a basic concept and directly introduce capture-avoiding substitutions based on this. But there is a conceptual and technical circularity in this approach, since a careful definition of α-equivalence needs substitutions.

In this Chapter we follow Peter Andrews' lead from [And02] and break the circularity by introducing syntactic substitutions, show a substitution value lemma with a substitutability condition, use that for a soundness proof of α-renaming, and only then introduce capture-avoiding substitutions on this basis. This can be done for any logic with bound variables, we go through the details for first-order logic here as an example.

7.1 First-Order Logic: Syntax and Semantics

The syntax and semantics of first-order logic is systematically organized in two distinct layers: one for truth values (like in propositional logic) and one for individuals (the new, distinctive feature of first-order logic).

The first step of defining a formal language is to specify the alphabet, here the first-order signatures and their components.

PL^1 Syntax (Signature and Variables)

▷ **Definition 7.1.1** First-order logic (PL^1), is a formal logical system extensively used in mathematics, philosophy, linguistics, and computer science. It combines propositional logic with the ability to quantify over individuals.

▷ PL^1 talks about two kinds of objects: (so we have two kinds of symbols)

- ▷ truth values; sometimes annotated by type o (like in PL^0)
- ▷ individuals; sometimes annotated by type ι (numbers, foxes, Pokémon,...)

▷ **Definition 7.1.2** A first-order signature consists of (all disjoint; $k \in \mathbb{N}$)

- ▷ connectives: $\Sigma^o = \{T, F, \neg, \vee, \wedge, \Rightarrow, \Leftrightarrow, \ldots\}$ (functions on truth values)
- ▷ function constants: $\Sigma^f_k = \{f, g, h, \ldots\}$ (functions on individuals)
- ▷ predicate constants: $\Sigma^p_k = \{p, q, r, \ldots\}$ (relations among inds.)
- ▷ (Skolem constants: $\Sigma^{sk}_k = \{f^k_1, f^k_2, \ldots\}$) (witness constructors; countably ∞)
- ▷ We take Σ_ι to be all of these together: $\Sigma_\iota := \Sigma^f \cup \Sigma^p \cup \Sigma^{sk}$, where $\Sigma^* := \bigcup_{k\in\mathbb{N}} \Sigma^*_k$ and define $\Sigma := \Sigma_\iota \cup \Sigma^o$.

▷ We assume a set of individual variables: $\mathcal{V}_\iota = \{X_\iota, Y_\iota, Z, {X^1}_\iota, X^2\}$ (countably ∞)

We make the deliberate, but non-standard design choice here to include Skolem constants into the signature from the start. These are used in inference systems to give names to objects and construct witnesses. Other than the fact that they are usually introduced by need, they work exactly like regular constants, which makes the inclusion rather painless. As we can never predict how many Skolem constants we are going to need, we give ourselves countably infinitely many for every arity. Our supply of individual variables is countably infinite for the same reason.

The formulae of first-order logic is built up from the signature and variables as terms (to represent individuals) and propositions (to represent propositions). The latter include the propositional connectives, but also quantifiers.

PL^1 Syntax (Formulae)

▷ **Definition 7.1.3** terms: $\mathbf{A} \in wff_\iota(\Sigma_\iota)$ (denote individuals: type ι)

- ▷ $\mathcal{V}_\iota \subseteq wff_\iota(\Sigma_\iota)$,
- ▷ if $f \in \Sigma^f_k$ and $\mathbf{A}^i \in wff_\iota(\Sigma_\iota)$ for $i \leq k$, then $f(\mathbf{A}^1, \ldots, \mathbf{A}^k) \in wff_\iota(\Sigma_\iota)$.

▷ **Definition 7.1.4** propositions: $\mathbf{A} \in wff_o(\Sigma)$(denote truth values: type o)

- ▷ if $p \in \Sigma^p_k$ and $\mathbf{A}^i \in wff_\iota(\Sigma_\iota)$ for $i \leq k$, then $p(\mathbf{A}^1, \ldots, \mathbf{A}^k) \in wff_o(\Sigma)$,
- ▷ if $\mathbf{A}, \mathbf{B} \in wff_o(\Sigma)$, then $T, \mathbf{A} \wedge \mathbf{B}, \neg\mathbf{A}, \forall X\,.\,\mathbf{A} \in wff_o(\Sigma)$.

▷ **Definition 7.1.5** We define the connectives $F, \vee, \Rightarrow, \Leftrightarrow$ via the abbreviations $\mathbf{A} \vee \mathbf{B} := \neg(\neg\mathbf{A} \wedge \neg\mathbf{B})$, $\mathbf{A} \Rightarrow \mathbf{B} := \neg\mathbf{A} \vee \mathbf{B}$, $\mathbf{A} \Leftrightarrow \mathbf{B} := (\mathbf{A} \Rightarrow \mathbf{B}) \wedge (\mathbf{B} \Rightarrow \mathbf{A})$, and $F := \neg T$. We will use them like the primary connectives $\wedge$ and $\neg$

▷ **Definition 7.1.6** We use $\exists X\,.\,\mathbf{A}$ as an abbreviation for $\neg(\forall X\,.\,\neg\mathbf{A})$.(existential quantifier)

▷ **Definition 7.1.7** Call formulae without connectives or quantifiers atomic else complex.

Note: that we only need e.g. conjunction, negation, and universal quantification, all other logical constants can be defined from them (as we will see when we have fixed their interpretations).

Alternative Notations for Quantifiers

Here	Elsewhere
$\forall x\,.\,\mathbf{A}$	$\bigwedge x\,.\,\mathbf{A}$ $(x)\,.\,\mathbf{A}$
$\exists x\,.\,\mathbf{A}$	$\bigvee x\,.\,\mathbf{A}$

The introduction of quantifiers to first-order logic brings a new phenomenon: variables that are under the scope of a quantifiers will behave very differently from the ones that are not. Therefore we build up a vocabulary that distinguishes the two.

Free and Bound Variables

▷ **Definition 7.1.8** We call an occurrence of a variable X bound in a formula $\mathbf{A}$, iff it occurs in a sub-formula $\forall X\,.\,\mathbf{B}$ of $\mathbf{A}$. We call a variable occurrence free otherwise.

For a formula $\mathbf{A}$, we will use $\mathrm{BVar}(\mathbf{A})$ (and $\mathrm{free}(\mathbf{A})$) for the set of bound (free) variables of $\mathbf{A}$, i.e. variables that have a free/bound occurrence in $\mathbf{A}$.

▷ **Definition 7.1.9** We define the set $\mathrm{free}(\mathbf{A})$ of free variables of a formula $\mathbf{A}$ inductively:

$$\begin{aligned}
\mathrm{free}(X) &:= \{X\}\\
\mathrm{free}(f(\mathbf{A}_1,\ldots,\mathbf{A}_n)) &:= \bigcup_{1\leq i\leq n}\mathrm{free}(\mathbf{A}_i)\\
\mathrm{free}(p(\mathbf{A}_1,\ldots,\mathbf{A}_n)) &:= \bigcup_{1\leq i\leq n}\mathrm{free}(\mathbf{A}_i)\\
\mathrm{free}(\neg\mathbf{A}) &:= \mathrm{free}(\mathbf{A})\\
\mathrm{free}(\mathbf{A}\wedge\mathbf{B}) &:= \mathrm{free}(\mathbf{A})\cup\mathrm{free}(\mathbf{B})\\
\mathrm{free}(\forall X\,.\,\mathbf{A}) &:= \mathrm{free}(\mathbf{A})\backslash\{X\}
\end{aligned}$$

▷ **Definition 7.1.10** We call a formula $\mathbf{A}$ closed or ground, iff $\mathrm{free}(\mathbf{A})=\emptyset$. We call a closed proposition a sentence, and denote the set of all ground terms with $cwff_\iota(\Sigma_\iota)$ and the set of sentences with $cwff_o(\Sigma_\iota)$.

We will be mainly interested in (sets of) sentences – i.e. closed propositions – as the representations of meaningful statements about individuals. Indeed, we will see below that free variables do not gives us expressivity, since they behave like constants and could be replaced by them in all situations, except the recursive definition of quantified formulae. Indeed in all situations where variables occur freely, they have the character of meta-variables, i.e. syntactic placeholders that can be instantiated with terms when needed in an inference calculus.

The semantics of first-order logic is a Tarski-style set-theoretic semantics where the atomic syntactic entities are interpreted by mapping them into a well-understood structure, a first-order universe that is just an arbitrary set.

Semantics of PL^1 (Models)

▷ We fix the Universe $\mathcal{D}_o=\{\mathsf{T},\mathsf{F}\}$ of truth values.

▷ We assume an arbitrary universe $\mathcal{D}_\iota\neq\emptyset$ of individuals (this choice is a parameter to the semantics)

▷ **Definition 7.1.11** An interpretation $\mathcal{I}$ assigns values to constants, e.g.

- ▷ $\mathcal{I}(\neg)\colon\mathcal{D}_o\to\mathcal{D}_o$ with $\mathsf{T}\mapsto\mathsf{F}$, $\mathsf{F}\mapsto\mathsf{T}$, and $\mathcal{I}(\wedge)=\ldots$ (as in PL^0)
- ▷ $\mathcal{I}\colon\Sigma^f_k\to\mathcal{F}({\mathcal{D}_\iota}^k;\mathcal{D}_\iota)$ (interpret function symbols as arbitrary functions)
- ▷ $\mathcal{I}\colon\Sigma^p_k\to\mathcal{P}({\mathcal{D}_\iota}^k)$ (interpret predicates as arbitrary relations)

▷ **Definition 7.1.12** A variable assignment $\varphi\colon\mathcal{V}_\iota\to\mathcal{D}_\iota$ maps variables into the universe.

▷ A first-order Model $\mathcal{M}=\langle\mathcal{D}_\iota,\mathcal{I}\rangle$ consists of a universe $\mathcal{D}_\iota$ and an interpretation$\mathcal{I}$.

We do not have to make the universe of truth values part of the model, since it is always the same; we determine the model by choosing a universe and an interpretation function.

Given a first-order model, we can define the evaluation function as a homomorphism over the construction of formulae.

Semantics of PL^1 (Evaluation)

▷ Given a model $\langle\mathcal{D},\mathcal{I}\rangle$, the value function $\mathcal{I}_\varphi$ is recursively defined:(two parts: terms & propositions)

▷ $\mathcal{I}_\varphi\colon \mathit{wff}_\iota(\Sigma_\iota) \to \mathcal{D}_\iota$ assigns values to terms.
  ▷ $\mathcal{I}_\varphi(X) := \varphi(X)$ and
  ▷ $\mathcal{I}_\varphi(f(\mathbf{A}_1,\ldots,\mathbf{A}_k)) := \mathcal{I}(f)(\mathcal{I}_\varphi(\mathbf{A}_1),\ldots,\mathcal{I}_\varphi(\mathbf{A}_k))$
▷ $\mathcal{I}_\varphi\colon \mathit{wff}_o(\Sigma) \to \mathcal{D}_o$ assigns values to formulae:
  ▷ $\mathcal{I}_\varphi(T) = \mathcal{I}(T) = \mathsf{T}$,
  ▷ $\mathcal{I}_\varphi(\neg\,\mathbf{A}) = \mathcal{I}(\neg)(\mathcal{I}_\varphi(\mathbf{A}))$
  ▷ $\mathcal{I}_\varphi(\mathbf{A}\wedge\mathbf{B}) = \mathcal{I}(\wedge)(\mathcal{I}_\varphi(\mathbf{A}),\mathcal{I}_\varphi(\mathbf{B}))$ (just as in PL^0)
  ▷ $\mathcal{I}_\varphi(p(\mathbf{A}^1,\ldots,\mathbf{A}^k)) := \mathsf{T}$, iff $\langle\mathcal{I}_\varphi(\mathbf{A}^1),\ldots,\mathcal{I}_\varphi(\mathbf{A}^k)\rangle \in \mathcal{I}(p)$
  ▷ $\mathcal{I}_\varphi(\forall X\,\mathbf{.}\,\mathbf{A}) := \mathsf{T}$, iff $\mathcal{I}_{\varphi,[a/X]}(\mathbf{A}) = \mathsf{T}$ for all $a \in \mathcal{D}_\iota$.

The only new (and interesting) case in this definition is the quantifier case, there we define the value of a quantified formula by the value of its scope – *but with an extended variable assignment*. Note that by passing to the scope $\mathbf{A}$ of $\forall x\,\mathbf{.}\,\mathbf{A}$, the occurrences of the variable x in $\mathbf{A}$ that were bound in $\forall x\,\mathbf{.}\,\mathbf{A}$ become free and are amenable to evaluation by the variable assignment $\psi := \varphi,[a/X]$. Note that as an extension of φ, the assignment ψ supplies exactly the right value for x in $\mathbf{A}$. This variability of the variable assignment in the definition value function justifies the somewhat complex setup of first-order evaluation, where we have the (static) interpretation function for the symbols from the signature and the (dynamic) variable assignment for the variables.

Note furthermore, that the value $\mathcal{I}_\varphi(\exists x\,\mathbf{.}\,\mathbf{A})$ of $\exists x\,\mathbf{.}\,\mathbf{A}$, which we have defined to be $\neg\,(\forall x\,\mathbf{.}\,\neg\,\mathbf{A})$ is true, iff it is not the case that $\mathcal{I}_\varphi(\forall x\,\mathbf{.}\,\neg\,\mathbf{A}) = \mathcal{I}_\psi(\neg\,\mathbf{A}) = \mathsf{F}$ for all $\mathsf{a} \in \mathcal{D}_\iota$ and $\psi := \varphi,[a/X]$. This is the case, iff $\mathcal{I}_\psi(\mathbf{A}) = \mathsf{T}$ for some $\mathsf{a} \in \mathcal{D}_\iota$. So our definition of the existential quantifier yields the appropriate semantics.

7.2 First-Order Substitutions

We will now turn our attention to substitutions, special formula-to-formula mappings that operationalize the intuition that (individual) variables stand for arbitrary terms.

Substitutions on Terms

▷ Intuition: If $\mathbf{B}$ is a term and X is a variable, then we denote the result of systematically replacing all occurrences of X in a term $\mathbf{A}$ by $\mathbf{B}$ with $[\mathbf{B}/X](\mathbf{A})$.

▷ Problem: What about $[Z/Y],[Y/X](X)$, is that Y or Z?

▷ Folklore: $[Z/Y],[Y/X](X) = Y$, but $[Z/Y]([Y/X](X)) = Z$ of course. (Parallel application)

▷ **Definition 7.2.1** We call $\sigma\colon \mathit{wff}_\iota(\Sigma_\iota) \to \mathit{wff}_\iota(\Sigma_\iota)$ a substitution, iff $\sigma(f(\mathbf{A}_1,\ldots,\mathbf{A}_n)) = f(\sigma(\mathbf{A}_1),\ldots,\sigma(\mathbf{A}_n))$ and the support $\mathbf{supp}(\sigma) := \{X \mid \sigma(X) \neq X\}$ of σ is finite.

▷ **Observation 7.2.2** *Note that a substitution σ is determined by its values on variables alone, thus we can write σ as $\sigma|_{\mathcal{V}_\iota} = \{[\sigma(X)/X] \mid X \in \mathbf{supp}(\sigma)\}$.*

▷ **Notation 7.2.3** We denote the substitution σ with $\mathbf{supp}(\sigma) = \{x^i \mid 1{\leq}i{\leq}n\}$ and $\sigma(x^i) = \mathbf{A}_i$ by $[\mathbf{A}_1/x^1],\ldots,[\mathbf{A}_n/x^n]$.

▷ **Example 7.2.4** $[a/x], [f(b)/y], [a/z]$ instantiates $g(x, y, h(z))$ to $g(a, f(b), h(a))$.

▷ **Definition 7.2.5** We call $\mathbf{intro}(\sigma) := \bigcup_{X \in \mathbf{supp}(\sigma)} \mathrm{free}(\sigma(X))$ the set of variables introduced by σ.

The extension of a substitution is an important operation, which you will run into from time to time. Given a substitution σ, a variable x, and an expression $\mathbf{A}$, $\sigma, [\mathbf{A}/x]$ extends σ with a new value for x. The intuition is that the values right of the comma overwrite the pairs in the substitution on the left, which already has a value for x, even though the representation of σ may not show it.

Substitution Extension

▷ **Notation 7.2.6 (Substitution Extension)** Let σ be a substitution, then we denote with $\sigma, [\mathbf{A}/X]$ the function $\{(Y, \mathbf{A}) \in \sigma \mid Y \neq X\} \cup \{(X, \mathbf{A})\}$.
($\sigma, [\mathbf{A}/X]$ coincides with σ of X, and gives the result $\mathbf{A}$ there.)

▷ Note: If σ is a substitution, then $\sigma, [\mathbf{A}/X]$ is also a substitution.

▷ **Definition 7.2.7** If σ is a substitution, then we call $\sigma, [\mathbf{A}/X]$ the extension of σ by $[\mathbf{A}/X]$.

▷ We also need the dual operation: removing a variable from the support

▷ **Definition 7.2.8** We can discharge a variable X from a substitution σ by $\sigma_{-X} := \sigma, [X/X]$.

Note that the use of the comma notation for substitutions defined in Notation 7.2.3 is consistent with substitution extension. We can view a substitution $[a/x], [f(b)/y]$ as the extension of the empty substitution (the identity function on variables) by $[f(b)/y]$ and then by $[a/x]$. Note furthermore, that substitution extension is not commutative in general.

For first-order substitutions we need to extend the substitutions defined on terms to act on propositions. This is technically more involved, since we have to take care of bound variables.

Substitutions on Propositions

▷ Problem: We want to extend substitutions to propositions, in particular to quantified formulae: What is $\sigma(\forall X.\mathbf{A})$?

▷ Idea: σ should not instantiate bound variables. ($[\mathbf{A}/X](\forall X.\mathbf{B}) = \forall \mathbf{A}.\mathbf{B}'$ ill-formed)

▷ **Definition 7.2.9** $\sigma(\forall X.\mathbf{A}) := (\forall X.\sigma_{-X}(\mathbf{A}))$.

▷ Problem: This can lead to variable capture: $[f(X)/Y](\forall X.p(X, Y))$ would evaluate to $\forall X.p(X, f(X))$, where the second occurrence of X is bound after instantiation, whereas it was free before.

▷ **Definition 7.2.10** Let $\mathbf{B} \in wff_\iota(\Sigma_\iota)$ and $\mathbf{A} \in wff_o(\Sigma)$, then we call $\mathbf{B}$ substitutable for X in $\mathbf{A}$, iff $\mathbf{A}$ has no occurrence of X in a subterm $\forall Y.\mathbf{C}$ with $Y \in \text{free}(\mathbf{B})$.

▷ Solution: Forbid substitution $[\mathbf{B}/X]\mathbf{A}$, when $\mathbf{B}$ is not substitutable for X in $\mathbf{A}$.

▷ Better Solution: Rename away the bound variable X in $\forall X.p(X,Y)$ before applying the substitution. (see alphabetic renaming later.)

Here we come to a conceptual problem of most introductions to first-order logic: they directly define substitutions to be capture-avoiding by stipulating that bound variables are renamed in the to ensure subsitutability. But at this time, we have not even defined alphabetic renaming yet, and cannot formally do that without having a notion of substitution. So we will refrain from introducing capture-avoiding substitutions until we have done our homework.

We now introduce a central tool for reasoning about the semantics of substitutions: the "substitution-value Lemma", which relates the process of instantiation to (semantic) evaluation. This result will be the motor of all soundness proofs on axioms and inference rules acting on variables via substitutions. In fact, any logic with variables and substitutions will have (to have) some form of a substitution-value Lemma to get the meta-theory going, so it is usually the first target in any development of such a logic.

We establish the substitution-value Lemma for first-order logic in two steps, first on terms, where it is very simple, and then on propositions, where we have to take special care of substitutability.

Substitution Value Lemma for Terms

▷ **Lemma 7.2.11** *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be terms, then* $\mathcal{I}_\varphi([\mathbf{B}/X]\mathbf{A}) = \mathcal{I}_\psi(\mathbf{A})$*, where* $\psi = \varphi, [\mathcal{I}_\varphi(\mathbf{B})/X]$.

▷ Proof: by induction on the depth of $\mathbf{A}$:

P.1.1 depth=0:

P.1.1.1 Then $\mathbf{A}$ is a variable (say Y), or constant, so we have three cases

P.1.1.1.1 $\mathbf{A} = Y = X$: then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](X)) = \mathcal{I}_\varphi(\mathbf{B}) = \psi(X) = \mathcal{I}_\psi(X) = \mathcal{I}_\psi(\mathbf{A})$.

P.1.1.1.2 $\mathbf{A} = Y \neq X$: then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](Y)) = \mathcal{I}_\varphi(Y) = \varphi(Y) = \psi(Y) = \mathcal{I}_\psi(Y) = \mathcal{I}_\psi(\mathbf{A})$.

P.1.1.1.3 $\mathbf{A}$ is a constant: analogous to the preceding case ($Y \neq X$)

P.1.1.2 This completes the base case (depth = 0). □

P.1.2 depth> 0: then $\mathbf{A} = f(\mathbf{A}_1, \ldots, \mathbf{A}_n)$ and we have

$$\begin{aligned}\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) &= \mathcal{I}(f)(\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}_1)), \ldots, \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}_n)))\\ &= \mathcal{I}(f)(\mathcal{I}_\psi(\mathbf{A}_1), \ldots, \mathcal{I}_\psi(\mathbf{A}_n))\\ &= \mathcal{I}_\psi(\mathbf{A}).\end{aligned}$$

by inductive hypothesis

P.1.2.2 This completes the inductive case, and we have proven the assertion
□
□

We now come to the case of propositions. Note that we have the additional assumption of substitutability here.

Substitution Value Lemma for Propositions

▷ **Lemma 7.2.12** *Let $\mathbf{B} \in wff_\iota(\Sigma_\iota)$ be substitutable for X in $\mathbf{A} \in wff_o(\Sigma)$, then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\psi(\mathbf{A})$, where $\psi = \varphi, [\mathcal{I}_\varphi(\mathbf{B})/X]$.*

▷ Proof: by induction on the number n of connectives and quantifiers in $\mathbf{A}$

P.1.1 $n = 0$: then $\mathbf{A}$ is an atomic proposition, and we can argue like in the inductive case of the substitution value lemma for terms.

P.1.2 $n>0$ and $\mathbf{A} = \neg\mathbf{B}$ or $\mathbf{A} = \mathbf{C} \circ \mathbf{D}$: Here we argue like in the inductive case of the term lemma as well.

P.1.3 $n>0$ and $\mathbf{A} = \forall X\,.\,\mathbf{C}$: then $\mathcal{I}_\psi(\mathbf{A}) = \mathcal{I}_\psi(\forall X\,.\,\mathbf{C}) = \mathsf{T}$, iff $\mathcal{I}_{\psi,[a/X]}(\mathbf{C}) = \mathcal{I}_{\varphi,[a/X]}(\mathbf{C}) = \mathsf{T}$, for all $a \in \mathcal{D}_\iota$, which is the case, iff $\mathcal{I}_\varphi(\forall X\,.\,\mathbf{C}) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathsf{T}$.

P.1.4 $n>0$ and $\mathbf{A} = \forall Y\,.\,\mathbf{C}$ where $X \neq Y$: then $\mathcal{I}_\psi(\mathbf{A}) = \mathcal{I}_\psi(\forall Y\,.\,\mathbf{C}) = \mathsf{T}$, iff $\mathcal{I}_{\psi,[a/Y]}(\mathbf{C}) = \mathcal{I}_{\varphi,[a/Y]}([\mathbf{B}/X](\mathbf{C})) = \mathsf{T}$, by inductive hypothesis. So $\mathcal{I}_\psi(\mathbf{A}) = \mathcal{I}_\varphi(\forall Y\,.\,[\mathbf{B}/X](\mathbf{C})) = \mathcal{I}_\varphi([\mathbf{B}/X](\forall Y\,.\,\mathbf{C})) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}))$
□

To understand the proof fully, you should look out where the substitutability is actually used.

Armed with the substitution value lemma, we can now define alphabetic renaming and show it to be sound with respect to the semantics we defined above. And this soundness result will justify the definition of capture-avoiding substitution we will use in the rest of the course.

7.3 Alpha-Renaming for First-Order Logic

Armed with the substitution value lemma we can now prove one of the main representational facts for first-order logic: the names of bound variables do not matter; they can be renamed at liberty without changing the meaning of a formula.

Alphabetic Renaming

▷ **Lemma 7.3.1** *Bound variables can be renamed: If Y is substitutable for X in $\mathbf{A}$, then $\mathcal{I}_\varphi(\forall X\,.\,\mathbf{A}) = \mathcal{I}_\varphi(\forall Y\,.\,[Y/X](\mathbf{A}))$*

▷ Proof: by the definitions:

P.1 $\mathcal{I}_\varphi(\forall X\,.\,\mathbf{A}) = \mathsf{T}$, iff

P.2 $\mathcal{I}_{\varphi,[a/X]}(\mathbf{A}) = \mathsf{T}$ for all $a \in \mathcal{D}_\iota$, iff

P.3 $\mathcal{I}_{\varphi,[a/Y]}([Y/X](\mathbf{A})) = \mathsf{T}$ for all $a \in \mathcal{D}_\iota$, iff (by substitution value lemma)
P.4 $\mathcal{I}_\varphi(\forall Y\,.\,[Y/X](\mathbf{A})) = \mathsf{T}$. □

▷ **Definition 7.3.2** We call two formulae $\mathbf{A}$ and $\mathbf{B}$ alphabetical variants (or α-equal; write $\mathbf{A} =_\alpha \mathbf{B}$), iff $\mathbf{A} = \forall X\,.\,\mathbf{C}$ and $\mathbf{B} = \forall Y\,.\,[Y/X](\mathbf{C})$ for some variables X and Y.

We have seen that naive substitutions can lead to variable capture. As a consequence, we always have to presuppose that all instantiations respect a substitutability condition, which is quite tedious. We will now come up with an improved definition of substitution application for first-order logic that does not have this problem.

Avoiding Variable Capture by Built-in α-renaming

▷ Idea: Given alphabetic renaming, we will consider alphabetical variants as identical

▷ So: Bound variable names in formulae are just a representational device (we rename bound variables wherever necessary)

▷ Formally: Take $cwff_o(\Sigma_\iota)$ (new) to be the quotient set of $cwff_o(\Sigma_\iota)$ (old) modulo $=_\alpha$. (formulae as syntactic representatives of equivalence classes)

▷ **Definition 7.3.3 (Capture-Avoiding Substitution Application)** Let σ be a substitution, $\mathbf{A}$ a formula, and $\mathbf{A}'$ an alphabetical variant of $\mathbf{A}$, such that $\mathbf{intro}(\sigma) \cap \mathrm{BVar}(\mathbf{A}) = \emptyset$. Then $[\mathbf{A}]_{=_\alpha} = [\mathbf{A}']_{=_\alpha}$ and we can define $\sigma([\mathbf{A}]_{=_\alpha}) := [\sigma(\mathbf{A}')]_{=_\alpha}$.

▷ **Notation 7.3.4** After we have understood the quotient construction, we will neglect making it explicit and write formulae and substitutions with the understanding that they act on quotients.

Chapter 8

Inference in First-Order Logic

In this Chapter we will introduce inference systems (calculi) for first-order logic and study their properties, in particular soundness and completeness.

8.1 First-Order Calculi

In this section we will introduce two reasoning calculi for first-order logic, both were invented by Gerhard Gentzen in the 1930's and are very much related. The "natural deduction" calculus was created in order to model the natural mode of reasoning e.g. in everyday mathematical practice. This calculus was intended as a counter-approach to the well-known Hilbert-style calculi, which were mainly used as theoretical devices for studying reasoning in principle, not for modeling particular reasoning styles.

The "sequent calculus" was a rationalized version and extension of the natural deduction calculus that makes certain meta-proofs simpler to push through[3]. EdN:3

Both calculi have a similar structure, which is motivated by the human-orientation: rather than using a minimal set of inference rules, they provide two inference rules for every connective and quantifier, one "introduction rule" (an inference rule that derives a formula with that symbol at the head) and one "elimination rule" (an inference rule that acts on a formula with this head and derives a set of subformulae).

This allows us to introduce the calculi in two stages, first for the propositional connectives and then extend this to a calculus for first-order logic by adding rules for the quantifiers.

To obtain a first-order calculus, we have to extend $\mathcal{ND}^0$ with (introduction and elimination) rules for the quantifiers.

First-Order Natural Deduction ($\mathcal{ND}^1$; Gentzen [Gen34])

▷ Rules for propositional connectives just as always

▷ **Definition 8.1.1 (New Quantifier Rules)** The first-order natural deduction calculus $\mathcal{ND}^1$ extends $\mathcal{ND}^0$ by the following four rules

[3] EdNote: say something about cut elimination/analytical calculi somewhere

$$\frac{\mathbf{A}}{\forall X.\mathbf{A}}\forall I^* \qquad \frac{\forall X.\mathbf{A}}{[\mathbf{B}/X](\mathbf{A})}\forall E$$

$$\frac{[\mathbf{B}/X](\mathbf{A})}{\exists X.\mathbf{A}}\exists I \qquad \frac{\exists X.\mathbf{A} \qquad \begin{array}{c}[[c/X](\mathbf{A})]^1\\ \vdots\\ \mathbf{C}\end{array}}{\mathbf{C}}\exists E^1$$

* means that $\mathbf{A}$ does not depend on any hypothesis in which X is free.

The intuition behind the rule $\forall I$ is that a formula $\mathbf{A}$ with a (free) variable X can be generalized to $\forall X.\mathbf{A}$, if X stands for an arbitrary object, i.e. there are no restricting assumptions about X. The $\forall E$ rule is just a substitution rule that allows to instantiate arbitrary terms $\mathbf{B}$ for X in $\mathbf{A}$. The $\exists I$ rule says if we have a witness $\mathbf{B}$ for X in $\mathbf{A}$ (i.e. a concrete term $\mathbf{B}$ that makes $\mathbf{A}$ true), then we can existentially close $\mathbf{A}$. The $\exists E$ rule corresponds to the common mathematical practice, where we give objects we know exist a new name c and continue the proof by reasoning about this concrete object c. Anything we can prove from the assumption $[c/X](\mathbf{A})$ we can prove outright if $\exists X.\mathbf{A}$ is known.

This is the classical formulation of the calculus of natural deduction. To prepare the things we want to do later (and to get around the somewhat un-licensed extension by hypothetical reasoning in the calculus), we will reformulate the calculus by lifting it to the "judgements level". Instead of postulating rules that make statements about the validity of propositions, we postulate rules that make state about derivability. This move allows us to make the respective local hypotheses in ND derivations into syntactic parts of the objects (we call them "sequents") manipulated by the inference rules.

Natural Deduction in Sequent Calculus Formulation

▷ Idea: Explicit representation of hypotheses (lift calculus to judgments)

▷ **Definition 8.1.2** A judgment is a meta-statement about the provability of propositions

▷ **Definition 8.1.3** A sequent is a judgment of the form $\mathcal{H} \vdash \mathbf{A}$ about the provability of the formula $\mathbf{A}$ from the set $\mathcal{H}$ of hypotheses.

▷ Idea: Reformulate ND rules so that they act on sequents

▷ **Example 8.1.4**

$$\dfrac{\dfrac{\dfrac{\mathbf{A}\wedge\mathbf{B}\vdash\mathbf{A}\wedge\mathbf{B}}{\mathbf{A}\wedge\mathbf{B}\vdash\mathbf{B}}\wedge E_r \quad \dfrac{\mathbf{A}\wedge\mathbf{B}\vdash\mathbf{A}\wedge\mathbf{B}}{\mathbf{A}\wedge\mathbf{B}\vdash\mathbf{A}}\wedge E_l}{\mathbf{A}\wedge\mathbf{B}\vdash\mathbf{B}\wedge\mathbf{A}}\wedge I}{\emptyset\vdash\mathbf{A}\wedge\mathbf{B}\Rightarrow\mathbf{B}\wedge\mathbf{A}}\Rightarrow I$$

▷ Note: Even though the antecedent of a sequent is written like a sequence, it is actually a set. In particular, we can permute and duplicate members at will.

Sequent-Style Rules for Natural Deduction

▷ **Definition 8.1.5** The following inference rules make up the sequent calculus

$$\frac{}{\Gamma, \mathbf{A} \vdash \mathbf{A}}\,\mathrm{Ax} \qquad \frac{\Gamma \vdash \mathbf{B}}{\Gamma, \mathbf{A} \vdash \mathbf{B}}\,\mathrm{weaken} \qquad \frac{}{\Gamma \vdash \mathbf{A} \vee \neg \mathbf{A}}\,\mathrm{TND}$$

$$\frac{\Gamma \vdash \mathbf{A} \quad \Gamma \vdash \mathbf{B}}{\Gamma \vdash \mathbf{A} \wedge \mathbf{B}} \wedge I \qquad \frac{\Gamma \vdash \mathbf{A} \wedge \mathbf{B}}{\Gamma \vdash \mathbf{A}} \wedge E_l \qquad \frac{\Gamma \vdash \mathbf{A} \wedge \mathbf{B}}{\Gamma \vdash \mathbf{B}} \wedge E_r$$

$$\frac{\Gamma \vdash \mathbf{A}}{\Gamma \vdash \mathbf{A} \vee \mathbf{B}} \vee I_l \qquad \frac{\Gamma \vdash \mathbf{B}}{\Gamma \vdash \mathbf{A} \vee \mathbf{B}} \vee I_r \qquad \frac{\Gamma \vdash \mathbf{A} \vee \mathbf{B} \quad \Gamma, \mathbf{A} \vdash \mathbf{C} \quad \Gamma, \mathbf{B} \vdash \mathbf{C}}{\Gamma \vdash \mathbf{C}} \vee E$$

$$\frac{\Gamma, \mathbf{A} \vdash \mathbf{B}}{\Gamma \vdash \mathbf{A} \Rightarrow \mathbf{B}} \Rightarrow I \qquad \frac{\Gamma \vdash \mathbf{A} \Rightarrow \mathbf{B} \quad \Gamma \vdash \mathbf{A}}{\Gamma \vdash \mathbf{B}} \Rightarrow E$$

$$\frac{\Gamma, \mathbf{A} \vdash F}{\Gamma \vdash \neg \mathbf{A}} \neg I \qquad \frac{\Gamma \vdash \neg \neg \mathbf{A}}{\mathbf{A}} \neg E$$

$$\frac{\Gamma \vdash \neg \mathbf{A} \quad \Gamma \vdash \mathbf{A}}{\Gamma \vdash F} FI \qquad \frac{\Gamma \vdash F}{\Gamma \vdash \mathbf{A}} FE$$

First-Order Natural Deduction in Sequent Formulation

▷ Rules for propositional connectives just as always

▷ **Definition 8.1.6 (New Quantifier Rules)**

$$\frac{\Gamma \vdash \mathbf{A} \quad X \notin \mathrm{free}(\Gamma)}{\Gamma \vdash \forall X.\mathbf{A}} \forall I \qquad \frac{\Gamma \vdash \forall X.\mathbf{A}}{\Gamma \vdash [\mathbf{B}/X](\mathbf{A})} \forall E$$

$$\frac{\Gamma \vdash [\mathbf{B}/X](\mathbf{A})}{\Gamma \vdash \exists X.\mathbf{A}} \exists I \qquad \frac{\Gamma \vdash \exists X.\mathbf{A} \quad \Gamma, [c/X](\mathbf{A}) \vdash \mathbf{C} \quad c \in \Sigma_0^{sk} \text{ new}}{\Gamma \vdash \mathbf{C}} \exists E$$

Natural Deduction with Equality

▷ **Definition 8.1.7 (First-Order Logic with Equality)** We extend PL^1 with a new logical symbol for equality $= \in \Sigma_2^p$ and fix its semantics to $\mathcal{I}(=) := \{(x, x) \mid x \in \mathcal{D}_\iota\}$. We call the extended logic first-order logic with equality ($\mathrm{PL}^1_=$)

▷ We now extend natural deduction as well.

▷ **Definition 8.1.8** For the calculus of natural deduction with equality $\mathcal{ND}^1_=$ we add the following two equality rules to $\mathcal{ND}^1$ to deal with equality:

$$\frac{}{\mathbf{A} = \mathbf{A}} =I \qquad \frac{\mathbf{A} = \mathbf{B} \quad \mathbf{C}\,[\mathbf{A}]_p}{[\mathbf{B}/p]\mathbf{C}} =E$$

where $\mathbf{C}[\mathbf{A}]_p$ if the formula $\mathbf{C}$ has a subterm $\mathbf{A}$ at position p and $[\mathbf{B}/p]\mathbf{C}$ is the result of replacing that subterm with $\mathbf{B}$.

▷ In many ways equivalence behaves like equality, so we will use the following derived rules in $\mathcal{ND}^1$:

$$\frac{}{\mathbf{A} \Leftrightarrow \mathbf{A}} \Leftrightarrow I \qquad \frac{\mathbf{A} \Leftrightarrow \mathbf{B} \quad \mathbf{C}[\mathbf{A}]_p}{[\mathbf{B}/p]\mathbf{C}} \Leftrightarrow =E$$

Again, we have two rules that follow the introduction/elimination pattern of natural deduction calculi.

$\mathcal{ND}^1_=$ Example: $\sqrt{2}$ is Irrational

▷ We can do real Maths with $\mathcal{ND}^1_=$:

▷ **Theorem 8.1.9** *$\sqrt{2}$ is irrational*

Proof: We prove the assertion by contradiction

P.1 Assume that $\sqrt{2}$ is rational.

P.2 Then there are numbers p and q such that $\sqrt{2} = p\,/\,q$.

P.3 So we know $2\ s^2 = r^2$.

P.4 But $2\ s^2$ has an odd number of prime factors while r^2 an even number.

P.5 This is a contradiction (since they are equal), so we have proven the assertion □

If we want to formalize this into $\mathcal{ND}^1$, we have to write down all the assertions in the proof steps in MathTalk and come up with justifications for them in terms of $\mathcal{ND}^1$ inference rules. Figure **??** shows such a proof, where we write $\prime n$ is prime, use $\#(n)$ for the number of prime factors of a number n, and write $\mathrm{irr}(r)$ if r is irrational. Each line in Figure **??** represents one "step" in the proof. It consists of line number (for referencing), a formula for the asserted property, a justification via a $\mathcal{ND}^1$ rules (and the lines this one is derived from), and finally a list of line numbers of proof steps that are local hypotheses in effect for the current line. Lines 6 and 9 have the pseudo-justification "local hyp" that indicates that they are local hypotheses for the proof (they only have an implicit counterpart in the inference rules as defined above). Finally we have abbreviated the arithmetic simplification of line 9 with the justification "arith" to avoid having to formalize elementary arithmetic.

We observe that the $\mathcal{ND}^1$ proof is much more detailed, and needs quite a few Lemmata about # to go through. Furthermore, we have added a MathTalk version of the definition of irrationality (and treat definitional equality via the equality rules). Apart from these artefacts of formalization, the two representations of proofs correspond to each other very directly.

$\mathcal{ND}^1_=$ Example: $\sqrt{2}$ is Irrational (the Proof)

#	hyp	formula	NDjust
1		$\forall n,m.\neg(2\ n+1)=(2\ m)$	lemma
2		$\forall n,m.\#(n^m)=m\ \#(n)$	lemma
3		$\forall n,p.\prime p\Rightarrow\#(p\ n)=\#(n)+1$	lemma
4		$\forall x.\mathrm{irr}(x)\Leftrightarrow(\neg(\exists p,q.x=p\,/\,q))$	definition
5		$\mathrm{irr}(\sqrt{2})\Leftrightarrow(\neg(\exists p,q.\sqrt{2}=p\,/\,q))$	$\forall E(4)$
6	6	$\neg\mathrm{irr}(\sqrt{2})$	local hyp
7	6	$\neg\neg(\exists p,q.\sqrt{2}=p\,/\,q)$	$\Leftrightarrow=E(6,5)$
8	6	$\exists p,q.\sqrt{2}=p\,/\,q$	$\neg E(7)$
9	6,9	$\sqrt{2}=r\,/\,s$	local hyp
10	6,9	$2\ s^2=r^2$	arith(9)
11	6,9	$\#(r^2)=2\ \#(r)$	$\forall E^2(2)$
12	6,9	$\prime 2\Rightarrow\#(2\ s^2)=\#(s^2)+1$	$\forall E^2(1)$
13		$\prime 2$	lemma
14	6,9	$\#(2\ s^2)=\#(s^2)+1$	$\Rightarrow E(13,12)$
15	6,9	$\#(s^2)=2\ \#(s)$	$\forall E^2(2)$
16	6,9	$\#(2\ s^2)=2\ \#(s)+1$	$=E(14,15)$
17		$\#(r^2)=\#(r^2)$	$=I$
18	6,9	$\#(2\ s^2)=\#(r^2)$	$=E(17,10)$
19	6.9	$2\ \#(s)+1=\#(r^2)$	$=E(18,16)$
20	6.9	$2\ \#(s)+1=2\ \#(r)$	$=E(19,11)$
21	6.9	$\neg(2\ \#(s)+1)=(2\ \#(r))$	$\forall E^2(1)$
22	6,9	F	$FI(20,21)$
23	6	F	$\exists E^6(22)$
24		$\neg\neg\mathrm{irr}(\sqrt{2})$	$\neg I^6(23)$
25		$\mathrm{irr}(\sqrt{2})$	$\neg E^2(23)$

We leave the soundness result for the first-order natural deduction calculus to the reader and turn to the complenesss result, which is much more involved and interesting.

8.2 Abstract Consistency and Model Existence

We will now come to an important tool in the theoretical study of reasoning calculi: the "abstract consistency"/"model existence" method. This method for analyzing calculi was developed by Jaako Hintikka, Raymond Smullyan, and Peter Andrews in 1950-1970 as an encapsulation of similar constructions that were used in completeness arguments in the decades before. The basis for this method is Smullyan's Observation [Smu63] that completeness proofs based on Hintikka sets only certain properties of consistency and that with little effort one can obtain a generalization "Smullyan's Unifying Principle".

The basic intuition for this method is the following: typically, a logical system $\mathcal{S}=\langle\mathcal{L},\mathcal{K},\models\rangle$ has multiple calculi, human-oriented ones like the natural deduction calculi and machine-oriented ones like the automated theorem proving calculi. All of these need to be analyzed for completeness (as a basic quality assurance measure).

A completeness proof for a calculus $\mathcal{C}$ for $\mathcal{S}$ typically comes in two parts: one analyzes $\mathcal{C}$-consistency (sets that cannot be refuted in $\mathcal{C}$), and the other construct $\mathcal{K}$-models for $\mathcal{C}$-consistent sets.

In this situtation the "abstract consistency"/"model existence" method encapsulates the model construction process into a meta-theorem: the "model existence" theorem. This provides a set of syntactic ("abstract consistency") conditions for calculi that are sufficient to construct models.

With the model existence theorem it suffices to show that $\mathcal{C}$-consistency is an abstract consistency property (a purely syntactic task that can be done by a $\mathcal{C}$-proof transformation argument) to obtain a completeness result for $\mathcal{C}$.

Model Existence (Overview)

▷ Definition: Abstract consistency

▷ Definition: Hintikka set (maximally abstract consistent)

▷ Theorem: Hintikka sets are satisfiable

▷ Theorem: If Φ is abstract consistent, then Φ can be extended to a Hintikka set.

▷ Corollary: If Φ is abstract consistent, then Φ is satisfiable

▷ Application: Let $\mathcal{C}$ be a calculus, if Φ is $\mathcal{C}$-consistent, then Φ is abstract consistent.

▷ Corollary: $\mathcal{C}$ is complete.

The proof of the model existence theorem goes via the notion of a Hintikka set, a set of formulae with very strong syntactic closure properties, which allow to read off models. Jaako Hintikka's original idea for completeness proofs was that for every complete calculus $\mathcal{C}$ and every $\mathcal{C}$-consistent set one can induce a Hintikka set, from which a model can be constructed. This can be considered as a first model existence theorem. However, the process of obtaining a Hintikka set for a set $\mathcal{C}$-consistent set Φ of sentences usually involves complicated calculus-dependent constructions.

In this situation, Raymond Smullyan was able to formulate the sufficient conditions for the existence of Hintikka sets in the form of "abstract consistency properties" by isolating the calculus-independent parts of the Hintikka set construction. His technique allows to reformulate Hintikka sets as maximal elements of abstract consistency classes and interpret the Hintikka set construction as a maximizing limit process.

To carry out the "model-existence"/"abstract consistency" method, we will first have to look at the notion of consistency.

Consistency and refutability are very important notions when studying the completeness for calculi; they form syntactic counterparts of satisfiability.

Consistency

▷ Let $\mathcal{C}$ be a calculus

▷ **Definition 8.2.1** Φ is called $\mathcal{C}$-refutable, if there is a formula $\mathbf{B}$, such that $\Phi \vdash_{\mathcal{C}} \mathbf{B}$ and $\Phi \vdash_{\mathcal{C}} \neg \mathbf{B}$.

▷ **Definition 8.2.2** We call a pair $\mathbf{A}$ and $\neg \mathbf{A}$ a contradiction.

▷ So a set Φ is $\mathcal{C}$-refutable, if $\mathcal{C}$ can derive a contradiction from it.

▷ **Definition 8.2.3** Φ is called $\mathcal{C}$-consistent, iff there is a formula $\mathbf{B}$, that is not derivable from Φ in $\mathcal{C}$.

▷ **Definition 8.2.4** We call a calculus $\mathcal{C}$ reasonable, iff implication elimination and conjunction introduction are admissible in $\mathcal{C}$ and $\mathbf{A} \wedge \neg \mathbf{A} \Rightarrow \mathbf{B}$ is a $\mathcal{C}$-theorem.

▷ **Theorem 8.2.5** *$\mathcal{C}$-inconsistency and $\mathcal{C}$-refutability coincide for reasonable calculi.*

It is very important to distinguish the syntactic $\mathcal{C}$-refutability and $\mathcal{C}$-consistency from satisfiability, which is a property of formulae that is at the heart of semantics. Note that the former specify the calculus (a syntactic device) while the latter does not. In fact we should actually say $\mathcal{S}$-satisfiability, where $\mathcal{S} = \langle \mathcal{L}, \mathcal{K}, \models \rangle$ is the current logical system.

Even the word "contradiction" has a syntactical flavor to it, it translates to "saying against each other" from its latin root.

The notion of an "abstract consistency class" provides the a calculus-independent notion of "consistency": A set Φ of sentences is considered "consistent in an abstract sense", iff it is a member of an abstract consistency class ∇.

Abstract Consistency

▷ **Definition 8.2.6** Let ∇ be a family of sets. We call ∇ closed under subsets, iff for each $\Phi \in \nabla$, all subsets $\Psi \subseteq \Phi$ are elements of ∇.

▷ **Notation 8.2.7** We will use $\Phi * \mathbf{A}$ for $\Phi \cup \{\mathbf{A}\}$.

▷ **Definition 8.2.8** A family $\nabla \subseteq \mathit{wff}_o(\Sigma)$ of sets of formulae is called a (first-order) abstract consistency class, iff it is closed under subsets, and for each $\Phi \in \nabla$

- ∇_c) $\mathbf{A} \notin \Phi$ or $\neg \mathbf{A} \notin \Phi$ for atomic $\mathbf{A} \in \mathit{wff}_o(\Sigma)$.
- $\nabla_\neg$) $\neg \neg \mathbf{A} \in \Phi$ implies $\Phi * \mathbf{A} \in \nabla$
- $\nabla_\wedge$) $(\mathbf{A} \wedge \mathbf{B}) \in \Phi$ implies $(\Phi \cup \{\mathbf{A}, \mathbf{B}\}) \in \nabla$
- $\nabla_\vee$) $\neg (\mathbf{A} \wedge \mathbf{B}) \in \Phi$ implies $\Phi * \neg \mathbf{A} \in \nabla$ or $\Phi * \neg \mathbf{B} \in \nabla$
- $\nabla_\forall$) If $(\forall X\,.\,\mathbf{A}) \in \Phi$, then $\Phi * [\mathbf{B}/X](\mathbf{A}) \in \nabla$ for each closed term $\mathbf{B}$.
- $\nabla_\exists$) If $\neg (\forall X\,.\,\mathbf{A}) \in \Phi$ and c is an individual constant that does not occur in Φ, then $\Phi * \neg [c/X](\mathbf{A}) \in \nabla$

The conditions are very natural: Take for instance ∇_c, it would be foolish to call a set Φ of sentences "consistent under a complete calculus", if it contains an elementary contradiction. The next condition $\nabla_\neg$ says that if a set Φ that contains a sentence $\neg \neg \mathbf{A}$ is "consistent", then we should be able to extend it by $\mathbf{A}$ without losing this property; in other words, a complete calculus should be able to recognize $\mathbf{A}$ and $\neg \neg \mathbf{A}$ to be equivalent.

We will carry out the proof here, since it gives us practice in dealing with the abstract consistency properties.

Actually we are after abstract consistency classes that have an even stronger property than just being closed under subsets. This will allow us to carry out a limit construction in the Hintikka set extension argument later.

Compact Collections

▷ **Definition 8.2.9** We call a collection ∇ of sets compact, iff for any set Φ we have
$\Phi \in \nabla$, iff $\Psi \in \nabla$ for every finite subset Ψ of Φ.

▷ **Lemma 8.2.10** *If ∇ is compact, then ∇ is closed under subsets.*

▷ Proof:

P.1 Suppose $S \subseteq T$ and $T \in \nabla$.

P.2 Every finite subset A of S is a finite subset of T.

P.3 As ∇ is compact, we know that $A \in \nabla$.

P.4 Thus $S \in \nabla$. □

The property of being closed under subsets is a "downwards-oriented" property: We go from large sets to small sets, compactness (the interesting direction anyways) is also an "upwards-oriented" property. We can go from small (finite) sets to large (infinite) sets. The main application for the compactness condition will be to show that infinite sets of formulae are in a family ∇ by testing all their finite subsets (which is much simpler).

The main result here is that abstract consistency classes can be extended to compact ones. The proof is quite tedious, but relatively straightforward. It allows us to assume that all abstract consistency classes are compact in the first place (otherwise we pass to the compact extension).

Compact Abstract Consistency Classes

▷ **Lemma 8.2.11** *Any first-order abstract consistency class can be extended to a compact one.*

▷ Proof:

P.1 We choose $\nabla' := \{\Phi \subseteq cwff_o(\Sigma_\iota) \,|\, \text{every finite subset of } \Phi \text{is in } \nabla\}$.

P.2 Now suppose that $\Phi \in \nabla$. ∇ is closed under subsets, so every finite subset of Φ is in ∇ and thus $\Phi \in \nabla'$. Hence $\nabla \subseteq \nabla'$.

P.3 Let us now show that each ∇' is compact.

P.3.1 Suppose $\Phi \in \nabla'$ and Ψ is an arbitrary finite subset of Φ.

P.3.2 By definition of ∇' all finite subsets of Φ are in ∇ and therefore $\Psi \in \nabla'$.

P.3.3 Thus all finite subsets of Φ are in ∇' whenever Φ is in ∇'.

P.3.4 On the other hand, suppose all finite subsets of Φ are in ∇'.

P.3.5 Then by the definition of ∇' the finite subsets of Φ are also in ∇, so $\Phi \in \nabla'$. Thus ∇' is compact.

P.4 Note that ∇' is closed under subsets by the Lemma above.

P.5 Next we show that if ∇ satisfies ∇_*, then ∇' satisfies ∇_*.

P.5.1 To show ∇_c, let $\Phi \in \nabla'$ and suppose there is an atom $\mathbf{A}$, such that $\{\mathbf{A}, \neg\mathbf{A}\} \subseteq \Phi$. Then $\{\mathbf{A}, \neg\mathbf{A}\} \in \nabla$ contradicting ∇_c.

P.5.2 To show $\nabla_\neg$, let $\Phi \in \nabla'$ and $\neg\neg\mathbf{A} \in \Phi$, then $\Phi * \mathbf{A} \in \nabla'$.

P.5.2.1 Let Ψ be any finite subset of $\Phi * \mathbf{A}$, and $\Theta := (\Psi \backslash \{\mathbf{A}\}) * \neg\neg\mathbf{A}$.

P.5.2.2 Θ is a finite subset of Φ, so $\Theta \in \nabla$.

P.5.2.3 Since ∇ is an abstract consistency class and $\neg\neg\mathbf{A} \in \Theta$, we get $\Theta * \mathbf{A} \in \nabla$ by $\nabla_\neg$.

P.5.2.4 We know that $\Psi \subseteq \Theta * \mathbf{A}$ and ∇ is closed under subsets, so $\Psi \in \nabla$.

P.5.2.5 Thus every finite subset Ψ of $\Phi * \mathbf{A}$ is in ∇ and therefore by definition $\Phi * \mathbf{A} \in \nabla'$.

P.5.3 the other cases are analogous to $\nabla_\neg$. □

Hintikka sets are sets of sentences with very strong analytic closure conditions. These are motivated as maximally consistent sets i.e. sets that already contain everything that can be consistently added to them.

∇-Hintikka Set

▷ **Definition 8.2.12** Let ∇ be an abstract consistency class, then we call a set $\mathcal{H} \in \nabla$ a ∇-Hintikka Set, iff $\mathcal{H}$ is maximal in ∇, i.e. for all $\mathbf{A}$ with $\mathcal{H} * \mathbf{A} \in \nabla$ we already have $\mathbf{A} \in \mathcal{H}$.

▷ **Theorem 8.2.13 (Hintikka Properties)** *Let ∇ be an abstract consistency class and $\mathcal{H}$ be a ∇-Hintikka set, then*

$\mathcal{H}_c$) For all $\mathbf{A} \in wff_o(\Sigma)$ we have $\mathbf{A} \notin \mathcal{H}$ or $\neg\mathbf{A} \notin \mathcal{H}$.

$\mathcal{H}_\neg$) If $\neg\neg\mathbf{A} \in \mathcal{H}$ then $\mathbf{A} \in \mathcal{H}$.

$\mathcal{H}_\wedge$) If $(\mathbf{A} \wedge \mathbf{B}) \in \mathcal{H}$ then $\mathbf{A}, \mathbf{B} \in \mathcal{H}$.

$\mathcal{H}_\vee$) If $\neg(\mathbf{A} \wedge \mathbf{B}) \in \mathcal{H}$ then $\neg\mathbf{A} \in \mathcal{H}$ or $\neg\mathbf{B} \in \mathcal{H}$.

$\mathcal{H}_\forall$) If $(\forall X.\mathbf{A}) \in \mathcal{H}$, then $[\mathbf{B}/X](\mathbf{A}) \in \mathcal{H}$ for each closed term $\mathbf{B}$.

$\mathcal{H}_\exists$) If $\neg(\forall X.\mathbf{A}) \in \mathcal{H}$ then $\neg[\mathbf{B}/X](\mathbf{A}) \in \mathcal{H}$ for some term closed term $\mathbf{B}$.

Proof:

▷ **P.1** We prove the properties in turn

$\mathcal{H}_c$ goes by induction on the structure of $\mathbf{A}$

P.2.1 $\mathbf{A}$ atomic: Then $\mathbf{A} \notin \mathcal{H}$ or $\neg\mathbf{A} \notin \mathcal{H}$ by ∇_c.

P.2.2 $\mathbf{A} = \neg\mathbf{B}$:

P.2.2.1 Let us assume that $\neg\mathbf{B} \in \mathcal{H}$ and $\neg\neg\mathbf{B} \in \mathcal{H}$,

P.2.2.2 then $\mathcal{H} * \mathbf{B} \in \nabla$ by $\nabla_\neg$, and therefore $\mathbf{B} \in \mathcal{H}$ by maximality.

P.2.2.3 So $\{\mathbf{B}, \neg\mathbf{B}\} \subseteq \mathcal{H}$, which contradicts the inductive hypothesis. □

P.2.3 $\mathbf{A} = \mathbf{B} \vee \mathbf{C}$: similar to the previous case

We prove $\mathcal{H}_\neg$ by maximality of $\mathcal{H}$ in ∇.

P.3.1 If $\neg\neg\mathbf{A} \in \mathcal{H}$, then $\mathcal{H} * \mathbf{A} \in \nabla$ by $\nabla_\neg$.

P.3.2 The maximality of $\mathcal{H}$ now gives us that $\mathbf{A} \in \mathcal{H}$.

The other $\mathcal{H}_*$ are similar □

The following theorem is one of the main results in the "abstract consistency"/"model existence" method. For any abstract consistent set Φ it allows us to construct a Hintikka set $\mathcal{H}$ with $\Phi \in \mathcal{H}$.

P.4 Extension Theorem

▷ **Theorem 8.2.14** *If ∇ is an abstract consistency class and $\Phi \in \nabla$ finite, then there is a ∇-Hintikka set $\mathcal{H}$ with $\Phi \subseteq \mathcal{H}$.*

▷ Proof: Wlog. assume that ∇ compact (else use compact extension)

P.1 Choose an enumeration $\mathbf{A}^1, \mathbf{A}^2, \ldots$ of $cwff_o(\Sigma_\iota)$ and $c^1, c^2, \ldots$ of Σ_0^{sk}.

P.2 and construct a sequence of sets H^i with $H^0 := \Phi$ and

$$H^{n+1} := \begin{cases} H^n & \text{if } H^n * \mathbf{A}^n \notin \nabla \\ H^n \cup \{\mathbf{A}^n, \neg[c^n/X](\mathbf{B})\} & \text{if } H^n * \mathbf{A}^n \in \nabla \text{ and } \mathbf{A}^n = \neg(\forall X.\mathbf{B}) \\ H^n * \mathbf{A}^n & \text{else} \end{cases}$$

P.3 Note that all $H^i \in \nabla$, choose $\mathcal{H} := \bigcup_{i\in\mathbb{N}} H^i$

P.4 $\Psi \subseteq \mathcal{H}$ finite implies there is a $j \in \mathbb{N}$ such that $\Psi \subseteq H^j$,

P.5 so $\Psi \in \nabla$ as ∇ closed under subsets and $\mathcal{H} \in \nabla$ as ∇ is compact.

P.6 Let $\mathcal{H} * \mathbf{B} \in \nabla$, then there is a $j \in \mathbb{N}$ with $\mathbf{B} = \mathbf{A}^j$, so that $\mathbf{B} \in H^{j+1}$ and $H^{j+1} \subseteq \mathcal{H}$

P.7 Thus $\mathcal{H}$ is ∇-maximal □

Note that the construction in the proof above is non-trivial in two respects. First, the limit construction for $\mathcal{H}$ is not executed in our original abstract consistency class ∇, but in a suitably extended one to make it compact — the original would not have contained $\mathcal{H}$ in general. Second, the set $\mathcal{H}$ is not unique for Φ, but depends on the choice of the enumeration of $cwff_o(\Sigma_\iota)$. If we pick a different enumeration, we will end up with a different $\mathcal{H}$. Say if $\mathbf{A}$ and $\neg\mathbf{A}$ are both ∇-consistent[4] with Φ, then depending on which one is first in the enumeration $\mathcal{H}$, will contain that one; with all the consequences for subsequent choices in the construction process.

EdN:4

Valuation

▷ **Definition 8.2.15** A function $\nu\colon cwff_o(\Sigma_\iota) \to \mathcal{D}_o$ is called a (first-order) valuation, iff

- ▷ $\nu(\neg\mathbf{A}) = \mathsf{T}$, iff $\nu(\mathbf{A}) = \mathsf{F}$
- ▷ $\nu(\mathbf{A} \wedge \mathbf{B}) = \mathsf{T}$, iff $\nu(\mathbf{A}) = \mathsf{T}$ and $\nu(\mathbf{B}) = \mathsf{T}$
- ▷ $\nu(\forall X.\mathbf{A}) = \mathsf{T}$, iff $\nu([\mathbf{B}/X](\mathbf{A})) = \mathsf{T}$ for all closed terms $\mathbf{B}$.

[4] EdNote: introduce this above

▷ **Lemma 8.2.16** *If $\varphi\colon \mathcal{V}_\iota \to \mathcal{D}$ is a variable assignment, then $\mathcal{I}_\varphi\colon cwff_o(\Sigma_\iota) \to \mathcal{D}_o$ is a valuation.*

▷ Proof Sketch: Immediate from the definitions □

Thus a valuation is a weaker notion of evaluation in first-order logic; the other direction is also true, even though the proof of this result is much more involved: The existence of a first-order valuation that makes a set of sentences true entails the existence of a model that satisfies it.[5] EdN:5

Valuation and Satisfiability

▷ **Lemma 8.2.17** *If $\nu\colon cwff_o(\Sigma_\iota) \to \mathcal{D}_o$ is a valuation and $\Phi \subseteq cwff_o(\Sigma_\iota)$ with $\nu(\Phi) = \{\mathsf{T}\}$, then Φ is satisfiable.*

▷ Proof: We construct a model for Φ.

P.1 Let $\mathcal{D}_\iota := cwff_\iota(\Sigma_\iota)$, and

- ▷ $\mathcal{I}(f)\colon {\mathcal{D}_\iota}^k \to \mathcal{D}_\iota; \langle \mathbf{A}_1, \ldots, \mathbf{A}_k\rangle \mapsto f(\mathbf{A}_1, \ldots, \mathbf{A}_k)$ for $f \in \Sigma^f$
- ▷ $\mathcal{I}(p)\colon {\mathcal{D}_\iota}^k \to \mathcal{D}_o; \langle \mathbf{A}_1, \ldots, \mathbf{A}_k\rangle \mapsto \nu(p(\mathbf{A}_1, \ldots, \mathbf{A}_n))$ for $p \in \Sigma^p$.

P.2 Then variable assignments into $\mathcal{D}_\iota$ are ground substitutions.

P.3 We show $\mathcal{I}_\varphi(\mathbf{A}) = \varphi(\mathbf{A})$ for $\mathbf{A} \in wff_\iota(\Sigma_\iota)$ by induction on $\mathbf{A}$

P.3.1 $\mathbf{A} = X$: then $\mathcal{I}_\varphi(\mathbf{A}) = \varphi(X)$ by definition.

P.3.2 $\mathbf{A} = f(\mathbf{A}_1, \ldots, \mathbf{A}_n)$: then $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}(f)(\mathcal{I}_\varphi(\mathbf{A}_1), \ldots, \mathcal{I}_\varphi(\mathbf{A}_n)) = \mathcal{I}(f)(\varphi(\mathbf{A}_1), \ldots, \varphi(\mathbf{A}_n)) = f(\varphi(\mathbf{A}_1), \ldots, \varphi(\mathbf{A}_n)) = \varphi(f(\mathbf{A}_1, \ldots, \mathbf{A}_n)) = \varphi(\mathbf{A})$

P.4 We show $\mathcal{I}_\varphi(\mathbf{A}) = \nu(\varphi(\mathbf{A}))$ for $\mathbf{A} \in wff_o(\Sigma)$ by induction on $\mathbf{A}$

P.4.1 $\mathbf{A} = p(\mathbf{A}_1, \ldots, \mathbf{A}_n)$: then $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}(p)(\mathcal{I}_\varphi(\mathbf{A}_1), \ldots, \mathcal{I}_\varphi(\mathbf{A}_n)) = \mathcal{I}(p)(\varphi(\mathbf{A}_1), \ldots, \varphi(\mathbf{A}_n)) = \nu(p(\varphi(\mathbf{A}_1), \ldots, \varphi(\mathbf{A}_n))) = \nu(\varphi(p(\mathbf{A}_1, \ldots, \mathbf{A}_n))) = \nu(\varphi(\mathbf{A}))$

P.4.2 $\mathbf{A} = \neg\mathbf{B}$: then $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$, iff $\mathcal{I}_\varphi(\mathbf{B}) = \nu(\varphi(\mathbf{B})) = \mathsf{F}$, iff $\nu(\varphi(\mathbf{A})) = \mathsf{T}$.

P.4.3 $\mathbf{A} = \mathbf{B} \wedge \mathbf{C}$: similar

P.4.4 $\mathbf{A} = \forall X.\mathbf{B}$: then $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$, iff $\mathcal{I}_\psi(\mathbf{B}) = \nu(\psi(\mathbf{B})) = \mathsf{T}$, for all $\mathbf{C} \in \mathcal{D}_\iota$, where $\psi = \varphi, [\mathbf{C}/X]$. This is the case, iff $\nu(\varphi(\mathbf{A})) = \mathsf{T}$.

P.5 Thus $\mathcal{I}_\varphi(\mathbf{A}) = \nu(\varphi(\mathbf{A})) = \nu(\mathbf{A}) = \mathsf{T}$ for all $\mathbf{A} \in \Phi$.

P.6 Hence $\mathcal{M} \models \mathbf{A}$ for $\mathcal{M} := \langle \mathcal{D}_\iota, \mathcal{I}\rangle$. □

Now, we only have to put the pieces together to obtain the model existence theorem we are after.

Model Existence

▷ **Theorem 8.2.18 (Hintikka-Lemma)** *If ∇ is an abstract consistency class*

[5] EdNote: I think that we only get a semivaluation, look it up in Andrews.

and $\mathcal{H}$ a ∇-Hintikka set, then $\mathcal{H}$ is satisfiable.

▷ Proof:

P.1 we define $\nu(\mathbf{A}) := \mathsf{T}$, iff $\mathbf{A} \in \mathcal{H}$,

P.2 then ν is a valuation by the Hintikka set properties.

P.3 We have $\nu(\mathcal{H}) = \{\mathsf{T}\}$, so $\mathcal{H}$ is satisfiable. □

▷ **Theorem 8.2.19 (Model Existence)** *If ∇ is an abstract consistency class and $\Phi \in \nabla$, then Φ is satisfiable.*

Proof:

▷ **P.1** There is a ∇-Hintikka set $\mathcal{H}$ with $\Phi \subseteq \mathcal{H}$ (Extension Theorem)

We know that $\mathcal{H}$ is satisfiable. (Hintikka-Lemma)

In particular, $\Phi \subseteq \mathcal{H}$ is satisfiable. □

8.3 A Completeness Proof for First-Order ND

With the model existence proof we have introduced in the last section, the completeness proof for first-order natural deduction is rather simple, we only have to check that ND-consistency is an abstract consistency property.

P.2 P.3 Consistency, Refutability and Abstract Consistency

▷ **Theorem 8.3.1 (Non-Refutability is an Abstract Consistency Property)** $\Gamma := \{\Phi \subseteq cwff_o(\Sigma_\iota) \mid \Phi \text{ not } \mathcal{ND}^1\text{−refutable}\}$ *is an abstract consistency class.*

▷ Proof: We check the properties of an ACC

P.1 If Φ is non-refutable, then any subset is as well, so Γ is closed under subsets.

P.2 We show the abstract consistency conditions ∇_* for $\Phi \in \Gamma$.

P.2.1 ∇_c: We have to show that $\mathbf{A} \notin \Phi$ or $\neg\mathbf{A} \notin \Phi$ for atomic $\mathbf{A} \in wff_o(\Sigma)$.

P.2.1.2 Equivalently, we show the contrapositive: If $\{\mathbf{A}, \neg\mathbf{A}\} \subseteq \Phi$, then $\Phi \notin \Gamma$.

P.2.1.3 So let $\{\mathbf{A}, \neg\mathbf{A}\} \subseteq \Phi$, then Φ is $\mathcal{ND}^1$-refutable by construction.

P.2.1.4 So $\Phi \notin \Gamma$. □

P.2.2 $\nabla_\neg$: We show the contrapositive again

P.2.2.2 Let $\neg\neg\mathbf{A} \in \Phi$ and $\Phi * \mathbf{A} \notin \Gamma$

P.2.2.3 Then we have a refutation $\mathcal{D}$: $\Phi * \mathbf{A} \vdash_{\mathcal{ND}^1} F$

P.2.2.4 By prepending an application of $\neg E$ for $\neg\neg\mathbf{A}$ to $\mathcal{D}$, we obtain a refutation $\mathcal{D}'$: $\Phi \vdash_{\mathcal{ND}^1} F$.

P.2.2.5 Thus $\Phi \notin \Gamma$. □

P.2.3 other ∇_* similar:

This directly yields two important results that we will use for the completeness analysis.

Henkin's Theorem

▷ **Corollary 8.3.2 (Henkin's Theorem)** *Every $\mathcal{ND}^1$-consistent set of sentences has a model.*

▷ Proof:

P.1 Let Φ be a $\mathcal{ND}^1$-consistent set of sentences.

P.2 The class of sets of $\mathcal{ND}^1$-consistent propositions constitute an abstract consistency class

P.3 Thus the model existence theorem guarantees a model for Φ. □

▷ **Corollary 8.3.3 (Löwenheim&Skolem Theorem)** *Satisfiable set Φ of first-order sentences has a countable model.*

▷ Proof Sketch: The model we constructed is countable, since the set of ground terms is. □

Now, the completeness result for first-order natural deduction is just a simple argument away. We also get a compactness theorem (almost) for free: logical systems with a complete calculus are always compact.

Completeness and Compactness

▷ **Theorem 8.3.4 (Completeness Theorem for $\mathcal{ND}^1$)** *If $\Phi \models \mathbf{A}$, then $\Phi \vdash_{\mathcal{ND}^1} \mathbf{A}$.*

▷ Proof: We prove the result by playing with negations.

P.1 If $\mathbf{A}$ is valid in all models of Φ, then $\Phi * \neg \mathbf{A}$ has no model

P.2 Thus $\Phi * \neg \mathbf{A}$ is inconsistent by (the contrapositive of) Henkins Theorem.

P.3 So $\Phi \vdash_{\mathcal{ND}^1} \neg\neg \mathbf{A}$ by $\neg I$ and thus $\Phi \vdash_{\mathcal{ND}^1} \mathbf{A}$ by $\neg E$. □

▷ **Theorem 8.3.5 (Compactness Theorem for first-order logic)** *If $\Phi \models \mathbf{A}$, then there is already a finite set $\Psi \subseteq \Phi$ with $\Psi \models \mathbf{A}$.*

Proof: This is a direct consequence of the completeness theorem

▷ **P.1** We have $\Phi \models \mathbf{A}$, iff $\Phi \vdash_{\mathcal{ND}^1} \mathbf{A}$.

As a proof is a finite object, only a finite subset $\Psi \subseteq \Phi$ can appear as leaves in the proof. □

8.4 Limits of First-Order Logic

EdN:6 We will now come to the limits of first-order Logic.[6]

P.2 Gödel's Incompleteness Theorem

▷ **Theorem 8.4.1** *No logical system that can represent Peano-Arithmetic* $(\mathbb{N}, s, 0, +, *)$ *admits complete calculi.*

▷ Proof: (Sketch)

P.1 Let $\mathcal{L} := \langle \mathcal{S}, \mathcal{C} \rangle$ be such a system. We show that there is a valid $\mathcal{S}$-sentence $\mathbf{A}_{\mathcal{C}}$, that is no $\mathcal{C}$-theorem.

P.2 Encode the syntax of $\mathcal{S}$ and the $\mathcal{C}$ in Peano-arithmetic

P.3 We can now talk about $\mathcal{S}$ and $\mathcal{C}$ in $\mathcal{S}$ itself.

P.4 E.g. there is a $\mathcal{S}$-sentence $\mathbf{B}$ with the meaning: $\mathbf{A}$ is a $\mathcal{C}$-theorem.

P.5 Choose $\mathbf{A}_{\mathcal{C}}$ as "$\mathbf{A}_{\mathcal{C}}$ is no $\mathcal{C}$-theorem" (cf. Russell's set)

P.6 Obviously: $\mathbf{A}_{\mathcal{C}}$ ist valid in all standard models.

P.7 So $\mathcal{C}$ is either not correct or cannot derive $\mathbf{A}_{\mathcal{C}}$. □

[6] EdNote: MK: also present the theorem (whose name I forgot) that show that FOL is the "strongest logic" for first-order models. Maybe also the interpolation theorem.

Part III

Axiomatic Set Theory (ZFC)

Sets are one of the most useful structures of mathematics. They can be used to form the basis for representing functions, ordering relations, groups, vector spaces, etc. In fact, they can be used as a foundation for all of mathematics as we know it. But sets are also among the most difficult structures to get right: we have already seen that "naive" conceptions of sets lead to inconsistencies that shake the foundations of mathematics.

There have been many attempts to resolve this unfortunate situation and come up a "foundation of mathematics": an inconsistency-free "foundational logic" and "foundational theory" on which all of mathematics can be built.

In this Part we will present the best-known such attempt – and an attempt it must remain as we will see – the axiomatic set theory by Zermelo and Fraenkel (ZFC), a set of axioms for first-order logic that carefully manage set comprehension to avoid introducing the "set of all sets" which leads us into the paradoxes.

Recommended Reading: The – historical and personal – background of the material covered in this Part is delightfully covered in [DPPDD09].

Chapter 9

Naive Set Theory

We will first recap "naive set theory" and try to formalize it in first-order logic to get a feeling for the problems involved and possible solutions.

(Naive) Set Theory [Can95, Can97]

▷ **Definition 9.0.1** A set is "everything that can form a unity in the face of God". (Georg Cantor (∗1845, †1918))

▷ **Example 9.0.2** (determination by elementhood relation $\in$)

- ▷ "the set that consists of the number 7 and the prime divisors of 510510"
- ▷ $\{7, c\}$, $\{1, 2, 3, 4, 5n, \ldots\}$, $\{x \mid x \text{ is an integer}\}$, $\{X \mid \mathbf{P}(X)\}$

Questions (extensional/intensional):

▷
- ▷ If $c = 7$, is $\{7, c\} = \{7\}$?
- ▷ Is $\{X \mid X \in \mathbb{N}, X \neq X\} = \{X \mid X \in \mathbb{N}, X^2 < 0\}$?
- ▷ yes $\rightsquigarrow$ *extensional*; no $\rightsquigarrow$ *intensional*;

Georg Cantor was the first to systematically develop a "set theory", introducing the notion of a "power set" and distinguishing finite from infinite sets – and the latter into denumerable and uncountable sets, basing notions of cardinality on bijections.

In doing so, he set a firm foundation for mathematics[1], even if that needed more work as was later discovered.

Now let us see whether we can write down the "theory of sets" as envisioned by Georg Cantor in first-order logic – which at the time Cantor published his seminal articles was just being invented by Gottlob Frege. The main idea here is to consider sets as individuals, and only introduce a single predicate – apart from equality which we consider given by the logic: the binary elementhood predicate.

(Naive) Set Theory: Formalization

[1] David Hilbert famously exclaimed "*No one shall expel us from the Paradise that Cantor has created*" in [Hil26, p. 170]

▷ Idea: Use first-order logic (with equality)

- ▹ Signature: (sets are individuals) $\Sigma := \{\in\}$
- ▹ Extensionality: $\forall M, N \,.\, M = N \Leftrightarrow (\forall X \,.\, (X \in M) \Leftrightarrow (X \in N))$
- ▹ Comprehension: (all sets that we can write down exist)
  $\exists M \,.\, \forall X \,.\, (X \in M) \Leftrightarrow \mathbf{E}$ (schematic in expression $\mathbf{E}$)

▷ Idea: Define set theoretic concepts from $\in$ as signature extensions

Union	$\cup \in \Sigma_2^f$	$\forall M, N, X \,.\, (X \in (M \cup N)) \Leftrightarrow (X \in M \vee X \in N)$
Intersection	$\cap \in \Sigma_2^f$	$\forall M, N, X \,.\, (X \in (M \cap N)) \Leftrightarrow (X \in M \wedge X \in N)$
Empty Set	$\emptyset \in \Sigma_0^f$	$\neg (\exists X \,.\, X \in \emptyset)$
and so on.	⋮	⋮

The central here is the comprehension axiom that states that any set we can describe by writing down a frist-order formula $\mathbf{E}$ – which usually contains the variable X – must exist. This is a direct implementation of Cantor's intuition that sets can be "*... everything that forms a unity ...*". The usual set-theoretic operators $\cup$, $\cap$, ... can be defined by suitable axioms.

This formalization will now allow to understand the problems of set theory: with great power comes great responsibility!

(Naive) Set Theory (Problems)

▷ **Example 9.0.3 (The set of all set and friends)**
$\{M \mid M \text{ set}\}$, $\{M \mid M \text{ set}, M \in M\}$, ...

▷ **Definition 9.0.4 (Problem)** Russell's Antinomy:

$$\mathcal{M} := \{M \mid M \text{ set}, M \notin M\}$$

the set $\mathcal{M}$ of all sets that do not contain themselves.

▷ Question: Is $\mathcal{M} \in \mathcal{M}$? Answer: $\mathcal{M} \in \mathcal{M}$ iff $\mathcal{M} \notin \mathcal{M}$.

▷ What happened?: We have written something down that makes problems

▷ Solutions: Define away the problems:

weaker comprehension	axiomatic set theory	now
weaker properties	higher-order logic	done
non-standard semantics	domain theory [Scott]	another time

The culprit for the paradox is the comprehension axiom that guarantees the existence of the "set of all sets" from which we can then separate out Russell's set. Multiple ways have been proposed to get around the paradoxes induced by the "set of all sets". We have already seen one: (typed) higher-order logic simply does not allow to write down MM which is higher-order (sets-as-predicates) way of representing set theory.

The way we are going to exploren now is to remove the general set comprehension axiom we had introduced above and replace it by more selective ones that only introduce sets that are known to be safe.

Chapter 10

ZFC Axioms

We will now introduce the set theory axioms due to Zermelo and Fraenkel.

We write down a first-order theory of sets by declaring axioms in first-order logic (with equality). The basic idea is that all individuals are sets, and we can therefore get by with a single binary predicate: $\in$ for elementhood.

Axiomatic Set Theory in First-Order Logic

▷ Idea: Avoid paradoxes by cautious (*axiomatic*) Comprehension. ([Zer08])

Ex	$\exists X.X = X$	There is a set
Ext	$\forall M, N.M = N \Leftrightarrow (\forall X.(X \in M) \Leftrightarrow (X \in N))$	Extensionality
Sep	$\forall N.\exists M.\forall Z.(Z \in M) \Leftrightarrow (Z \in N \wedge \mathbf{E})$ From a given set N we can separate all members described by expression $\mathbf{E}$.	

▷ **Theorem 10.0.1** $\forall M, N.(M \subseteq N) \wedge (N \subseteq M) \Rightarrow M = N$

▷ **Theorem 10.0.2** *M is uniquely determined in* **Sep**

▷ Proof Sketch: With **Ext** □

▷ **Notation 10.0.3** Write $\{X \in N \mid \mathbf{E}\}$ for the set M guaranteed by **Sep**.

Note that we do not have a general comprehension axiom, which allows the construction of sets from expressions, but the separation axiom **Sep**, which – given a set – allows to "separate out" a subset. As this axiom is insufficient to providing any sets at all, we guarantee that there is one in **Ex** to make the theory less boring.

Before we want to develop the theory further, let us fix the success criteria we have for our foundation.

Quality Control

▷ Question: Is ZFC good? (make this more precise under various views)

foundational: Is ZFC sufficient for mathematics?

adequate: is the ZFC notion of sets adequate?

formal: is ZFC consistent?

ambitious: Is ZFC complete?

pragmatic: Is the formalization convenient?

computational: does the formalization yield computation-guiding structure?

▷ Questions like these help us determine the quality of a foundational system or theory.

The question about consistency is the most important, so we will address it first. Note that the absence of paradoxes is a big question, which we cannot really answer now. But we *can* convince ourselves that the "set of all sets" cannot exist.

How about Russel's Antinomy?

▷ **Theorem 10.0.4** *There is no universal set*

▷ Proof:

P.1 For each set M, there is a set $M_R := \{X \in M \mid X \notin X\}$ by **Sep**.

P.2 show $\forall M.M_R \notin M$

P.3 If $M_R \in M$, then $M_R \notin M_R$, (also if $M_R \notin M$)

P.4 thus $M_R \notin M$ or $M_R \in M_R$. □

▷ to get the paradox we would have to separate from the universal set $\mathcal{A}$, to get $\mathcal{A}_R$.

▷ Great, then we can continue developing our set theory!

Somewhat surprisingly, we can just use Russell's construction to our advantage here. So back to the other questions.

Are there Interesting Sets at all?

▷ yes, e.g. the empty set

- ▷ let M be a set (there is one by **Ex**; we do not need to know what it is)
- ▷ define $\emptyset := \{X \in M \mid X \neq X\}$
- ▷ $\emptyset$ is empty and uniquely determined by **Ext**.

▷ **Definition 10.0.5** Intersections: $M \cap N := \{X \in M \mid X \in N\}$

Question: How about $M \cup N$? or $\mathbb{N}$?

⋙ Answer: we do not know they exist yet! (need more axioms)
Hint: consider $\mathcal{D}_\iota = \{\emptyset, \{\emptyset\}, \{\{\emptyset\}\}, \ldots\}$

So we have identified at least interesting set, the empty set. Unfortunately, the existence of the intersection operator is no big help, if we can only intersect with the empty set. In general, this is a consequence of the fact that **Sep** – in contrast to the comprehension axiom we have abolished – only allows to make sets "smaller". If we want to make sets "larger", we will need more axioms that guarantee these larger sets. The design contribution of axiomatic set theories is to find a balance between "too large" – and therefore paradoxical – and "not large enough" – and therefore inadequate.

Before we have a look at the remaining axioms of ZFC, we digress to a very influential experiment in developing mathematics based on set theory.

"Nicolas Bourbaki" is the collective pseudonym under which a group of (mainly French) 20th-century mathematicians, with the aim of reformulating mathematics on an extremely abstract and formal but self-contained basis, wrote a series of books beginning in 1935. With the goal of grounding all of mathematics on set theory, the group strove for rigour and generality.

Is Set theory enough? ⇝ Nicolas Bourbaki

▷ Is it possible to develop all of Mathematics from set theory?
⇝ N. Bourbaki: Éléments de Mathématiques (there is only one mathematics)

▷ Original Goal: A modern textbook on calculus.

▷ Result: 40 volumes in nine books from 1939 to 1968

Set Theory [Bou68]	Functions of one real variable	Commutative Algebra
Algebra [Bou74]	Integration	Lie Theory
Topology [Bou89]	Topological Vector Spaces	Spectral Theory

▷ Contents:

- ▷ starting from set theory all of the fields above are developed.
- ▷ All proofs are carried out, no references to other books.

Even though Bourbaki has dropped in favor in modern mathematics, the universality of axiomatic set theory is generally acknowledged in mathematics and their rigorous style of exposition has influenced modern branches of mathematics.

The first two axioms we add guarantee the unions of sets, either of finitely many – $\cup\mathbf{Ax}$ only guarantees the union of two sets – but can be iterated. And an axiom for unions of arbitrary families of sets, which gives us the infinite case. Note that once we have the ability to make finite sets, $\bigcup\mathbf{Ax}$ makes $\cup\mathbf{Ax}$ redundant, but minimality of the axiom system is not a concern for us currently.

The Axioms for Set Union

▷ **Axiom 10.0.6 (Small Union Axiom ($\cup\mathbf{Ax}$))** For any sets M and N there is a set W, that contains all elements of M and N.
$\forall M, N.\exists W.\forall X.(X \in M \vee X \in N) \Rightarrow X \in W$

▷ **Definition 10.0.7** $M \cup N := \{X \in W \mid X \in M \vee X \in N\}$ (exists by **Sep.**)

▷ **Axiom 10.0.8 (large Union Axiom ($\bigcup$Ax))** For each set M there is a set W, that contains the elements of all elements of M.
$\forall M.\exists W.\forall X, Y.Y \in M \Rightarrow X \in Y \Rightarrow X \in W$

▷ **Definition 10.0.9** $\bigcup(M) := \{X \mid \exists Y.Y \in M \wedge X \in Y\}$ (exists by **Sep.**)

▷ This also gives us intersections over families (without another axiom):

▷ **Definition 10.0.10**

$$\bigcap(M) := \{Z \in \bigcup(M) \mid \forall X.X \in M \Rightarrow Z \in X\}$$

In Definition 10.0.10 we note that **$\bigcup$Ax** also guarantees us intersection over families. Note that we could not have defined that in analogy to Definition 10.0.5 since we have no set to separate out of. Intuitively we could just choose one element N from M and define

$$\bigcap(M) := \{Z \in N \mid \forall X.X \in M \Rightarrow Z \in X\}$$

But for choice from an infinite set we need another axiom still.

The power set axiom is one of the most useful axioms in ZFC. It allows to construct finite sets.

The Power Set Axiom

▷ **Axiom 10.0.11 (Power Set Axiom)** For each set M there is a set W that contains all subsets of M: $\wp\,\text{Ax} := (\forall M.\exists W.\forall X.(X \subseteq M) \Rightarrow X \in W)$

▷ **Definition 10.0.12** Power Set: $\mathcal{P}(M) := \{X \mid X \subseteq M\}$ (Exists by **Sep.**)

▷ **Definition 10.0.13** singleton set: $\{X\} := \{Y \in \mathcal{P}(X) \mid X = Y\}$

▷ **Axiom 10.0.14 (Pair Set (Axiom))** (is often assumed instead of **$\cup$Ax**)

Given sets M and N there is a set W that contains exactly the elements M and N: $\forall M, N.\exists W.\forall X.(X \in W) \Leftrightarrow ((X = N) \vee (X = M))$

▷ Is derivable from $\wp$ Ax: $\{M, N\} := \{M\} \cup \{N\}$.

▷ **Definition 10.0.15 (Finite Sets)** $\{X, Y, Z\} := \{X, Y\} \cup \{Z\} \ldots$

▷ **Theorem 10.0.16** $\forall Z, X_1, \ldots, X_n.(Z \in \{X_1, \ldots, X_n\}) \Leftrightarrow (Z = X_1 \vee \ldots \vee Z = X_n)$

The Foundation Axiom

▷ **Axiom 10.0.17 (The foundation Axiom (Fund))** Every non-empty set has a $\in$-minimal element,.
$\forall X.X \neq \emptyset \Rightarrow (\exists Y.Y \in X \wedge \neg(\exists Z.Z \in X \wedge Z \in Y))$

▷ **Theorem 10.0.18** *There are no infinite descendig chains* $\ldots, X_2, X_1, X_0$ *and thus no cycles* $\ldots X_1, X_0, \ldots, X_2, X_1, X_0$.

▷ **Definition 10.0.19 Fund** guarantees a hierarchical structure (von Neumann Hierarchy) of the universe. 0. order: $\emptyset$, 1. order: $\{\emptyset\}$, 2. order: all subsets of 1. order, $\cdots$

▷ Note: In contrast to a Russel-style typing where sets of differernt type are distinct, this categorization is cummulative

The Infinity Axiom

▷ We already know a lot of sets

- ▹ z.B. $\emptyset$, $\{\emptyset\}$, $\{\{\emptyset\}\}$, ... (iterated singleton set)
- ▹ or $\emptyset$, $\{\emptyset\}$, $\{\emptyset, \{\emptyset\}\}$, ... (iterated pair set)

But: Does the set $\mathbb{N}$ of all members of these sequences?

▷ **Axiom 10.0.20 (Infinity Axiom (∞Ax))** There is a set that contains $\emptyset$ and with each X also $X \cup \{X\}$.
$\exists M . \emptyset \in M \wedge (\forall Z . Z \in M \Rightarrow (Z \cup \{Z\}) \in M)$.

▷ **Definition 10.0.21** M is inductive: $\mathbf{Ind}(M) := \emptyset \in M \wedge (\forall Z . Z \in M \Rightarrow (Z \cup \{Z\}) \in M)$.

▷ **Definition 10.0.22** Set of the Inductive Set: $\omega := \{Z \,|\, \forall W . \mathbf{Ind}(W) \Rightarrow Z \in W\}$

▷ **Theorem 10.0.23** ω *is inductive.*

The Replacement Axiom

▷ We have ω, $\wp(M)$, but not $\{\omega, \wp(\omega), \wp(\wp(\omega)), \ldots\}$.

▷ **Axiom 10.0.24 (The Replacement Axiom (Schema): Rep)** If for each X there is exactly one Y with property $\mathbf{P}(X, Y)$, then for each set U, that contains these X, there is a set V that contains the respective Y.
$(\forall X . \exists^1 Y . \mathbf{P}(X, Y)) \Rightarrow (\forall U . \exists V . \forall X, Y . X \in U \wedge \mathbf{P}(X, Y) \Rightarrow Y \in V)$

▷ Intuitively: A right-unique property $\mathbf{P}$ induces a replacement $\forall U . \exists V . V = \{F(X) \,|\, X \in U\}$.

▷ **Example 10.0.25** Let $U = \{1, \{2, 3\}\}$ and $\mathcal{P}(X \Leftrightarrow Y) \Leftrightarrow (\forall Z . Z \in Y \Rightarrow Z = X)$, then the induced function F maps each X to the set V that contains X, i.e. $V = \{\{X\} \,|\, X \in U = \{\{1\}, \{\{2, 3\}\}\}\}$.

Zermelo Fraenkel Set Theory

▷ **Definition 10.0.26 (Zermelo Fraenkel Set Theory)** We call the first-order theory given by the axioms below Zermelo/Fraenkel set theory and denote it by **ZF**.

Ex	$\exists X.X = X$
Ext	$\forall M, N.M = N \Leftrightarrow (\forall X.(X \in M) \Leftrightarrow (X \in N))$
Sep	$\forall N.\exists M.\forall Z.(Z \in M) \Leftrightarrow (Z \in N \wedge \mathbf{E})$
$\cup$**Ax**	$\forall M, N.\exists W.\forall X.(X \in M \vee X \in N) \Rightarrow X \in W$
$\bigcup$**Ax**	$\forall M.\exists W.\forall X, Y.Y \in M \Rightarrow X \in Y \Rightarrow X \in W$
$\wp$ Ax	$\forall M.\exists W.\forall X.(X \subseteq M) \Rightarrow X \in W$
∞**Ax**	$\exists M.\emptyset \in M \wedge (\forall Z.Z \in M \Rightarrow (Z \cup \{Z\}) \in M)$
Rep	$(\forall X.\exists^1 Y.\mathbf{P}(X, Y)) \Rightarrow (\forall U.\exists V.\forall X, Y.X \in U \wedge \mathbf{P}(X, Y) \Rightarrow Y \in V)$
Fund	$\forall X.X \neq \emptyset \Rightarrow (\exists Y.Y \in X \wedge \neg(\exists Z.Z \in X \wedge Z \in Y))$

The Axiom of Choice

▷ **Axiom 10.0.27 (The axiom of Choice :AC)** For each set X of non-empty, pairwise disjoint subsets there is a set that contains exactly one element of each element of X.
$\forall X, Y, Z.Y \in X \wedge Z \in X \Rightarrow Y \neq \emptyset \wedge (Y = Z \vee Y \cap Z = \emptyset) \Rightarrow \exists U.\forall V.V \in X \Rightarrow (\exists W.U \cap V = \{W\})$

▷ This axiom assumes the existence of a set of representatives, even if we cannot give a construction for it. $\rightsquigarrow$ we can "pick out" an arbitrary element.

▷ Reasons for **AC**:

- ▷ Neither **ZF** $\vdash$ **AC**, nor **ZF** $\vdash \neg$**AC**
- ▷ So it does not harm?

▷ **Definition 10.0.28 (Zermelo Fraenkel Set Theory with Choice)** The theory **ZF** together with **AC** is called ZFC with choice and denoted as **ZFC**.

Chapter 11

ZFC Applications

Limits of ZFC

▷ **Conjecture 11.0.1 (Cantor's Continuum Hypothesis (CH))** *There is no set whose cardinality is strictly between that of integers and real numbers.*

▷ **Theorem 11.0.2** *If* **ZFC** *is consistent, then neither* **CH** *nor* $\neg$ **CH** *can be derived.* (**CH** *is independent of* **ZFC**)

▷ The axiomatzation of **ZFC** does not suffice

▷ There are other examples like this.

Ordered Pairs

▷ Empirically: In **ZFC** we can define all mathematical concepts.

▷ For Instance: We would like a set that behaves like an odererd pair

▷ **Definition 11.0.3** Define $\langle X, Y\rangle := \{\{X\}, \{X, Y\}\}$

▷ **Lemma 11.0.4** $\langle X, Y\rangle = \langle U, V\rangle \Rightarrow X = U \wedge Y = V$

▷ **Lemma 11.0.5** $U \in X \wedge V \in Y \Rightarrow \langle U, V\rangle \in \mathcal{P}(\mathcal{P}(X \cup Y))$

▷ **Definition 11.0.6** left projection: $\pi_l(X) = \begin{cases} U & \text{if } \exists V.X = \langle U, V\rangle \\ \emptyset & \text{if } X \text{ is no pair} \end{cases}$

▷ **Definition 11.0.7** right projection π_r analogous.

Relations

▷ All mathematical objects are represented by sets in **ZFC**, in particular relations

▷ **Definition 11.0.8** The Cartesian produkt of X and Y
$X \times Y := \{Z \in \mathcal{P}(\mathcal{P}(X \cup Y)) \mid Z \text{ is ordered pair with } \pi_l(Z) \in X \wedge \pi_r(Z) \in Y\}$
A relation is a subset of a Cartesian product.

▷ **Definition 11.0.9** The domain and codomain of a function are defined as usual

$$\text{Dom}(X) = \begin{cases} \{\pi_l(Z) \mid Z \in X\} & \text{if if X is a relation;} \\ \emptyset & \text{else} \end{cases}$$

$$\text{coDom}(X) = \begin{cases} \{\pi_r(Z) \mid Z \in X\} & \text{if if } X \text{ is a relation;} \\ \emptyset & \text{else} \end{cases}$$

but they (as first-order functions) must be total, so we (arbitrarily) extend them by the empty set for non-relations

Functions

▷ **Definition 11.0.10** A function f from X to Y is a right-unique relation with $\text{Dom}(f) = X$ and $\text{coDom}(f) = Y$; write $f\colon X \to Y$.

▷ **Definition 11.0.11** function application: $f(X) = \begin{cases} Y & \text{if } f \text{ function and } \langle X, Y\rangle \in f \\ \emptyset & \text{else} \end{cases}$

Domain Language vs. Representation Language

▷ Note: Relations and functions are objects of set theory, $ZFC \in$ is a predicate of the representation language

▷ predicates and functions of the representation language can be expressed in the object language:

- ▷ $\forall A.\exists R.R = \{\langle U, V\rangle \mid U \in A \wedge V \in A \wedge p(U \wedge V)\}$ for all predicates p.
- ▷ $\forall A.\exists F.F = \{\langle X, f(X)\rangle \mid X \in A\}$ for all functions f.

▷ As the natural numbers can be epxressed in set theory, the logical calculus can be expressed by Gödelization.

Part IV

Higher-Order Logic and λ-Calculus

In this Part we set the stage for a deeper discussions of the logical foundations of mathematics by introducing a particular higher-order logic, which gets around the limitations of first-order logic — the restriction of quantification to individuals. This raises a couple of questions (paradoxes, comprehension, completeness) that have been very influential in the development of the logical systems we know today.

Therefore we use the discussion of higher-order logic as an introduction and motivation for the λ-calculus, which answers most of these questions in a term-level, computation-friendly system.

The formal development of the simply typed λ-calculus and the establishment of its (meta-logical) properties will be the body of work in this Part. Once we have that we can reconstruct a clean version of higher-order logic by adding special provisions for propositions.

Chapter 12

Higher-Order Predicate Logic

The main motivation for higher-order logic is to allow quantification over classes of objects that are not individuals — because we want to use them as functions or predicates, i.e. apply them to arguments in other parts of the formula.

Higher-Order Predicate Logic (PLΩ)

▷ Quantification over functions and Predicates: $\forall P.\exists F.P(a) \vee \neg P(F(a))$

▷ Comprehension: (Existence of Functions)
$\exists F.\forall X.FX = \mathbf{A}$ e.g. $f(x) = 3x^2 + 5x - 7$

▷ Extensionality: (Equality of functions and truth values)
$\forall F.\forall G.(\forall X.FX = GX) \Rightarrow F = G$
$\forall P.\forall Q.(P \Leftrightarrow Q) \Leftrightarrow P = Q$

▷ Leibniz Equality: (Indiscernability)
$\mathbf{A} = \mathbf{B}$ for $\forall P.P\mathbf{A} \Rightarrow P\mathbf{B}$

Indeed, if we just remove the restriction on quantification we can write down many things that are essential on everyday mathematics, but cannot be written down in first-order logic. But the naive logic we have created (BTW, this is essentially the logic of Frege [Fre79]) is much too expressive, it allows us to write down completely meaningless things as witnessed by Russell's paradox.

Problems with PLΩ

▷ Problem: Russell's Antinomy: $\forall Q.\mathcal{M}(Q) \Leftrightarrow (\neg Q(Q))$

- ▷ the set $\mathcal{M}$ of all sets that do not contain themselves
- ▷ Question: Is $\mathcal{M} \in \mathcal{M}$? Answer: $\mathcal{M} \in \mathcal{M}$ iff $\mathcal{M} \notin \mathcal{M}$.

▷ What has happened? the predicate Q has been applied to itself

▷ Solution for this course: Forbid self-applications by types!!

- ▷ ι, o (type of individuals, truth values), $\alpha \to \beta$ (function type)
- ▷ right associative bracketing: $\alpha \to \beta \to \gamma$ abbreviates $\alpha \to (\beta \to \gamma)$

▷ vector notation: $\overline{\alpha_n} \to \beta$ abbreviates $\alpha_1 \to \ldots \to \alpha_n \to \beta$

▷ Well-typed formulae (prohibits paradoxes like $\forall Q.\mathcal{M}(Q) \Leftrightarrow (\neg Q(Q))$)

▷ Other solution: Give it a non-standard semantics (Domain-Theory [Scott])

The solution to this problem turns out to be relatively simple with the benefit of hindsight: we just introduce a syntactic device that prevents us from writing down paradoxical formulae. This idea was first introduced by Russell and Whitehead in their Principia Mathematica [WR10].

Their system of "ramified types" was later radically simplified by Alonzo Church to the form we use here in [Chu40]. One of the simplifications is the restriction to unary functions that is made possible by the fact that we can re-interpret binary functions as unary ones using a technique called "Currying" after the Logician Haskell Brooks Curry (∗1900, †1982). Of course we can extend this to higher arities as well. So in theory we can consider n-ary functions as syntactic sugar for suitable higher-order functions. The vector notation for types defined above supports this intuition.

Types

▷ Types are semantic annotations for terms that prevent antinomies

▷ **Definition 12.0.1** Given a set $\mathcal{B}$ $\mathcal{T}$ of base types, construct function types: $\alpha \to \beta$ is the type of functions with domain type α and range type β. We call the closure $\mathcal{T}$ of $\mathcal{B}$ $\mathcal{T}$ under function types the set of types over $\mathcal{B}$ $\mathcal{T}$.

▷ **Definition 12.0.2** We will use ι for the type of individuals and o for the type of truth values.

▷ The type constructor is used as a right-associative operator, i.e. we use $\alpha \to \beta \to \gamma$ as an abbreviation for $\alpha \to (\beta \to \gamma)$

▷ We will use a kind of vector notation for function types, abbreviating $\alpha_1 \to \ldots \to \alpha_n \to \beta$ with $\overline{\alpha_n} \to \beta$.

Armed with a system of types, we can now define a typed higher-order logic, by insisting that all formulae of this logic be well-typed. One advantage of typed logics is that the natural classes of objects that have otherwise to be syntactically kept apart in the definition of the logic (e.g. the term and proposition levels in first-order logic), can now be distinguished by their type, leading to a much simpler exposition of the logic. Another advantage is that concepts like connectives that were at the language level e.g. in PL^0, can be formalized as constants in the signature, which again makes the exposition of the logic more flexible and regular. We only have to treat the quantifiers at the language level (for the moment).

Well-Typed Formulae (PLΩ)

▷ signature $\Sigma = \bigcup_{\alpha \in \mathcal{T}} \Sigma_\alpha$ with

▷ connectives: $\neg \in \Sigma_{o \to o}$ $\{\vee, \wedge, \Rightarrow, \Leftrightarrow \ldots\} \subseteq \Sigma_{o \to o \to o}$

▷ variables $\mathcal{V}_\mathcal{T} = \bigcup_{\alpha \in \mathcal{T}} \mathcal{V}_\alpha$, such that every $\mathcal{V}_\alpha$ countably infinite.

▷ well-typed formula e $wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ of type α

- ▷ $\mathcal{V}_\alpha \cup \Sigma_\alpha \subseteq wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$
- ▷ If $\mathbf{C} \in wff_{\alpha\to\beta}(\Sigma, \mathcal{V}_\mathcal{T})$ and $\mathbf{A} \in wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, then $(\mathbf{CA}) \in wff_\beta(\Sigma, \mathcal{V}_\mathcal{T})$
- ▷ If $\mathbf{A} \in wff_o(\Sigma, \mathcal{V}_\mathcal{T})$, then $(\forall X_\alpha.\mathbf{A}) \in wff_o(\Sigma, \mathcal{V}_\mathcal{T})$

▷ first-order terms have type ι, propositions the type o.

▷ there is no type annotation such that $\forall Q.\mathcal{M}(Q) \Leftrightarrow (\neg Q(Q))$ is well-typed. Q needs type α as well as $\alpha \to o$.

The semantics is similarly regular: We have universes for every type, and all functions are "typed functions", i.e. they respect the types of objects. Other than that, the setup is very similar to what we already know.

Standard Semantics for PLΩ

▷ **Definition 12.0.3** The universe of discourse (also carrier)

- ▷ arbitrary, non-empty set of individuals $\mathcal{D}_\iota$
- ▷ fixed set of truth values $\mathcal{D}_o = \{\mathsf{T}, \mathsf{F}\}$
- ▷ function universes $\mathcal{D}_{\alpha\to\beta} = \mathcal{F}(\mathcal{D}_\alpha; \mathcal{D}_\beta)$

▷ interpretation of constants: typed mapping $\mathcal{I}\colon \Sigma \to \mathcal{D}$ (i.e. $\mathcal{I}(\Sigma_\alpha) \subseteq \mathcal{D}_\alpha$)

▷ **Definition 12.0.4** We call a structure $\langle \mathcal{D}, \mathcal{I}\rangle$, where $\mathcal{D}$ is a universe and $\mathcal{I}$ an interpretation of constants a standard model of PLΩ.

▷ variable assignment: typed mapping $\varphi\colon \mathcal{V}_\mathcal{T} \to \mathcal{D}$

▷ **Definition 12.0.5** value function: typed mapping $\mathcal{I}_\varphi\colon wff_\mathcal{T}(\Sigma, \mathcal{V}_\mathcal{T}) \to \mathcal{D}$

- ▷ $\mathcal{I}_\varphi|_{\mathcal{V}_\mathcal{T}} = \varphi$ $\quad$ $\mathcal{I}_\varphi|_{\Sigma_\mathcal{T}} = \mathcal{I}$
- ▷ $\mathcal{I}_\varphi(\mathbf{AB}) = \mathcal{I}_\varphi(\mathbf{A})(\mathcal{I}_\varphi(\mathbf{B}))$
- ▷ $\mathcal{I}_\varphi(\forall X_\alpha.\mathbf{A}) = \mathsf{T}$, iff $\mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{A}) = \mathsf{T}$ for all $\mathsf{a} \in \mathcal{D}_\alpha$.

▷ $\mathbf{A}_o$ valid under φ, iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$.

We now go through a couple of examples of what we can express in PLΩ, and that works out very straightforwardly. For instance, we can express equality in PLΩ by Leibniz equality, and it has the right meaning.

Equality

▷ "Leibniz equality" (Indiscernability) $\mathbf{Q}^\alpha\mathbf{A}_\alpha\mathbf{B}_\alpha = \forall P_{\alpha\to o}.P\mathbf{A} \Leftrightarrow P\mathbf{B}$

▷ not that $\forall P_{\alpha\to o}.P\mathbf{A} \Rightarrow P\mathbf{B}$ (get the other direction by instantiating P with Q, where $QX \Leftrightarrow (\neg PX)$)

▷ **Theorem 12.0.6** *If $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}\rangle$ is a standard model, then $\mathcal{I}_\varphi(\mathbf{Q}^\alpha)$ is the*

identity relation on $\mathcal{D}_\alpha$.

▷ **Notation 12.0.7** We write $\mathbf{A} = \mathbf{B}$ for **QAB**($\mathbf{A}$ and $\mathbf{B}$ are equal, iff there is no property P that can tell them apart.)

▷ Proof:

P.1 $\mathcal{I}_\varphi(\mathbf{QAB}) = \mathcal{I}_\varphi(\forall P.P\mathbf{A} \Rightarrow P\mathbf{B}) = \mathsf{T}$, iff $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{A} \Rightarrow P\mathbf{B}) = \mathsf{T}$ for all $r \in \mathcal{D}_{\alpha\to o}$.

P.2 For $\mathbf{A} = \mathbf{B}$ we have $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{A}) = r(\mathcal{I}_\varphi(\mathbf{A})) = \mathsf{F}$ or $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{B}) = r(\mathcal{I}_\varphi(\mathbf{B})) = \mathsf{T}$.

P.3 Thus $\mathcal{I}_\varphi(\mathbf{QAB}) = \mathsf{T}$.

P.4 Let $\mathcal{I}_\varphi(\mathbf{A}) \neq \mathcal{I}_\varphi(\mathbf{B})$ and $r = \{\mathcal{I}_\varphi(\mathbf{A})\}$

P.5 so $r(\mathcal{I}_\varphi(\mathbf{A})) = \mathsf{T}$ and $r(\mathcal{I}_\varphi(\mathbf{B})) = \mathsf{F}$

P.6 $\mathcal{I}_\varphi(\mathbf{QAB}) = \mathsf{F}$, as $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{A} \Rightarrow P\mathbf{B}) = \mathsf{F}$, since $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{A}) = r(\mathcal{I}_\varphi(\mathbf{A})) = \mathsf{T}$ and $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{B}) = r(\mathcal{I}_\varphi(\mathbf{B})) = \mathsf{F}$. □

Another example are the Peano Axioms for the natural numbers, though we omit the proofs of adequacy of the axiomatization here.

Example: Peano Axioms for the Natural Numbers

▷ $\Sigma = \{[\mathbb{N} : \iota \to o], [0 : \iota], [s : \iota \to \iota]\}$

▷ $\mathbb{N}0$ (0 is a natural number)

▷ $\forall X_\iota.\mathbb{N}X \Rightarrow \mathbb{N}(sX)$ (the successor of a natural number is natural)

▷ $\neg(\exists X_\iota.\mathbb{N}X \wedge sX = 0)$ (0 has no predecessor)

▷ $\forall X_\iota.\forall Y_\iota.(sX = sY) \Rightarrow X = Y$ (the successor function is injective)

▷ $\forall P_{\iota\to o}.P0 \Rightarrow (\forall X_\iota.\mathbb{N}X \Rightarrow PX \Rightarrow P(sX)) \Rightarrow (\forall Y_\iota.\mathbb{N}Y \Rightarrow P(Y))$
induction axiom: all properties P, that hold of 0, and with every n for its successor $s(n)$, hold on all $\mathbb{N}$

Finally, we show the expressivity of PLΩ by formalizing a version of Cantor's theorem.

Expressive Formalism for Mathematics

▷ **Example 12.0.8 (Cantor's Theorem)** The cardinality of a set is smaller than that of its power set.

- ▷ smaller-card$(M, N) := \neg(\exists F.\text{surjective}(F, M, N))$
- ▷ surjective$(F, M, N) := (\forall X \in M.\exists Y \in N.FY = X)$

▷ **Example 12.0.9 (Simplified Formalization)** $\neg(\exists F_{\iota\to\iota\to\iota}.\forall G_{\iota\to\iota}.\exists J_\iota.FJ = G)$

▷ Standard-Benchmark for higher-order theorem provers

▷ can be proven by TPS and LEO (see below)

The simplified formulation of Cantor's theorem in Example 12.0.9 uses the universe of type ι for the set S and universe of type $\iota \to \iota$ for the power set rather than quantifying over S explicitly.

The next concern is to find a calculus for PLΩ.

We start out with the simplest one we can imagine, a Hilbert-style calculus that has been adapted to higher-order logic by letting the inference rules range over PLΩ formulae and insisting that substitutions are well-typed.

Hilbert-Calculus

▷ **Definition 12.0.10 ($\mathcal{H}_\Omega$ Axioms)** ▷ $\forall P_o, Q_o.P \Rightarrow Q \Rightarrow P$

▷ $\forall P_o, Q_o, R_o.(P \Rightarrow Q \Rightarrow R) \Rightarrow (P \Rightarrow Q) \Rightarrow P \Rightarrow R$

▷ $\forall P_o, Q_o.(\neg P \Rightarrow \neg Q) \Rightarrow P \Rightarrow Q$

▷ **Definition 12.0.11 ($\mathcal{H}_\Omega$ Inference rules)**

$$\frac{\mathbf{A}_o \Rightarrow \mathbf{B}_o \quad \mathbf{A}}{\mathbf{B}} \qquad \frac{\forall X_\alpha.\mathbf{A}}{[\mathbf{B}/X_\alpha](\mathbf{A})} \qquad \frac{\mathbf{A}}{\forall X_\alpha.\mathbf{A}} \qquad \frac{X \notin \text{free}(\mathbf{A}) \quad \forall X_\alpha.\mathbf{A} \wedge \mathbf{B}}{\mathbf{A} \wedge (\forall X_\alpha.\mathbf{B})}$$

▷ **Theorem 12.0.12** *Sound, wrt. standard semantics*

▷ Also Complete?

Not surprisingly, $\mathcal{H}_\Omega$ is sound, but it shows big problems with completeness. For instance, if we turn to a proof of Cantor's theorem via the well-known diagonal sequence argument, we will have to construct the diagonal sequence as a function of type $\iota \to \iota$, but up to now, we cannot in $\mathcal{H}_\Omega$. Unlike mathematical practice, which silently assumes that all functions we can write down in closed form exists, in logic, we have to have an axiom that guarantees (the existence of) such a function: the comprehension axioms.

Hilbert-Calculus $\mathcal{H}_\Omega$ (continued)

▷ valid sentences that are not $\mathcal{H}_\Omega$-theorems:

▷ Cantor's Theorem:
$\neg(\exists F_{\iota\to\iota\to\iota}.\forall G_{\iota\to\iota}.(\forall K_\iota.(\mathbb{N}K) \Rightarrow \mathbb{N}(GK)) \Rightarrow (\exists J_\iota.(\mathbb{N}J) \wedge FJ = G))$
(There is no surjective mapping from $\mathbb{N}$ into the set $\mathcal{F}(\mathbb{N};,)\mathbb{N}$ of natural number sequences)

▷ proof attempt fails at the subgoal $\exists G_{\iota\to\iota}.\forall X_\iota.GX = s(fXX)$

▷ Comprehension $\exists F_{\alpha\to\beta}.\forall X_\alpha.FX = \mathbf{A}_\beta$ (for every variable X_α and every term $\mathbf{A} \in \mathit{wff}_\beta(\Sigma, \mathcal{V}_\mathcal{T})$)

▷ extensionality
$\mathbf{Ext}^{\alpha\beta}$ $\forall F_{\alpha\to\beta}.\forall G_{\alpha\to\beta}.(\forall X_\alpha.FX = GX) \Rightarrow F = G$
$\mathbf{Ext}^{o}$ $\forall F_o.\forall G_o.(F \Leftrightarrow G) \Leftrightarrow F = G$

▷ correct! complete? cannot be!! [Göd31]

Actually it turns out that we need more axioms to prove elementary facts about mathematics: the extensionality axioms. But even with those, the calculus cannot be complete, even though empirically it proves all mathematical facts we are interested in.

Way Out: Henkin-Semantics

- ▷ Gödel's incompleteness theorem only holds for standard semantics
- ▷ find generalization that admits complete calculi:
- ▷ Idea: generalize so that the carrier only contains those functions that are requested by the comprehension axioms.
- ▷ **Theorem 12.0.13 (Henkin 1950)** $\mathcal{H}_\Omega$ *is complete wrt. this semantics.*
- ▷ Proof Sketch: more models $\rightsquigarrow$ less valid sentences (these are $\mathcal{H}_\Omega$-theorems) □
- ▷ Henkin-models induce sensible measure of completeness for higher-order logic.

Actually, there is another problem with PLΩ: The comprehension axioms are computationally very problematic. First, we observe that they are equality axioms, and thus are needed to show that two objects of PLΩ are equal. Second we observe that there are countably infinitely many of them (they are parametric in the term $\mathbf{A}$, the type α and the variable name), which makes dealing with them difficult in practice. Finally, axioms with both existential and universal quantifiers are always difficul to reason with.

Therefore we would like to have a formulation of higher-order logic without comprehension axioms. In the next slide we take a close look at the comprehension axioms and transform them into a form without quantifiers, which will turn out useful.

From Comprehension to β-Conversion

- ▷ $\exists F_{\alpha\to\beta}.\forall X_\alpha.FX = \mathbf{A}_\beta$ for arbitrary variable X_α and term $\mathbf{A} \in \mathit{wff}_\beta(\Sigma, \mathcal{V}_\mathcal{T})$ (for each term $\mathbf{A}$ and each variable X there is a function $f \in \mathcal{D}_{\alpha\to\beta}$, with $f(\varphi(X)) = \mathcal{I}_\varphi(\mathbf{A})$)
  - ▷ schematic in α, β, X_α and $\mathbf{A}_\beta$, very inconvenient for deduction
- ▷ Transformation in $\mathcal{H}_\Omega$
  - ▷ $\exists F_{\alpha\to\beta}.\forall X_\alpha.FX = \mathbf{A}_\beta$
  - ▷ $\forall X_\alpha.(\lambda X_\alpha.\mathbf{A})X = \mathbf{A}_\beta$ $(\exists E)$
    Call the function F whose existence is guaranteed "$(\lambda X_\alpha.\mathbf{A})$"
  - ▷ $(\lambda X_\alpha.\mathbf{A})\mathbf{B} = [\mathbf{B}/X]\mathbf{A}_\beta$ $(\forall E)$, in particular for $\mathbf{B} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$.
- ▷ **Definition 12.0.14** Axiom of β-equality: $(\lambda X_\alpha.\mathbf{A})\mathbf{B} = [\mathbf{B}/X](\mathbf{A}_\beta)$
- ▷ new formulae (λ-calculus [Church 1940])

In a similar way we can treat (functional) extensionality.

From Extensionality to η-Conversion

▷ **Definition 12.0.15** Extensionality Axiom: $\forall F_{\alpha\to\beta}.\forall G_{\alpha\to\beta}.(\forall X_\alpha.FX = GX) \Rightarrow F = G$

▷ Idea: Maybe we can get by with a simplified equality schema here as well.

▷ **Definition 12.0.16** We say that $\mathbf{A}$ and $\lambda X_\alpha.\mathbf{A}X$ are η-equal, (write $\mathbf{A}_{\alpha\to\beta} =_\eta (\lambda X_\alpha.\mathbf{A}X)$, if), iff $X \notin \text{free}(\mathbf{A})$.

▷ **Theorem 12.0.17** *η-equality and Extensionality are equivalent*

▷ Proof: We show that η-equality is special case of extensionality; the converse entailment is trivial

P.1 Let $\forall X_\alpha.\mathbf{A}X = \mathbf{B}X$, thus $\mathbf{A}X = \mathbf{B}X$ with $\forall E$

P.2 $\lambda X_\alpha.\mathbf{A}X = \lambda X_\alpha.\mathbf{B}X$, therefore $\mathbf{A} = \mathbf{B}$ with η

P.3 Hence $\forall F_{\alpha\to\beta}.\forall G_{\alpha\to\beta}.(\forall X_\alpha.FX = GX) \Rightarrow F = G$ by twice $\forall I$. □

▷ Axiom of truth values: $\forall F_o.\forall G_o.(F \Leftrightarrow G) \Leftrightarrow F = G$ unsolved.

The price to pay is that we need to pay for getting rid of the comprehension and extensionality axioms is that we need a logic that systematically includes the λ-generated names we used in the transformation as (generic) witnesses for the existential quantifier. Alonzo Church did just that with his "simply typed λ-calculus" which we will introduce next.

Chapter 13

Simply Typed λ-Calculus

In this section we will present a logic that can deal with functions – the simply typed λ-calculus. It is a typed logic, so everything we write down is typed (even if we do not always write the types down).

Simply typed λ-Calculus (Syntax)

▷ Signature $\Sigma = \bigcup_{\alpha\in\mathcal{T}} \Sigma_\alpha$ (includes countably infinite Signatures Σ^{Sk}_α of Skolem contants).

▷ $\mathcal{V}_\mathcal{T} = \bigcup_{\alpha\in\mathcal{T}} \mathcal{V}_\alpha$, such that $\mathcal{V}_\alpha$ are countably infinite

▷ **Definition 13.0.1** We call the set $wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ defined by the rules

- ▹ $\mathcal{V}_\alpha \cup \Sigma_\alpha \subseteq wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$
- ▹ If $\mathbf{C} \in wff_{\alpha\to\beta}(\Sigma, \mathcal{V}_\mathcal{T})$ and $\mathbf{A} \in wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, then $(\mathbf{CA}) \in wff_\beta(\Sigma, \mathcal{V}_\mathcal{T})$
- ▹ If $\mathbf{A} \in wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, then $(\lambda X_\beta\mathbf{.A}) \in wff_{\beta\to\alpha}(\Sigma, \mathcal{V}_\mathcal{T})$

the set of well-typed formula e of type α over the signature Σ and use $wff_\mathcal{T}(\Sigma, \mathcal{V}_\mathcal{T}) := \bigcup_{\alpha\in\mathcal{T}} wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ for the set of all well-typed formulae.

▷ **Definition 13.0.2** We will call all occurrences of the variable X in $\mathbf{A}$ bound in $\lambda X\mathbf{.A}$. Variables that are not bound in $\mathbf{B}$ are called free in $\mathbf{B}$.

▷ Substitutions are well-typed, i.e. $\sigma(X_\alpha) \in wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ and capture-avoiding.

▷ **Definition 13.0.3 (Simply Typed λ-Calculus)** The simply typed λ-calculus $\Lambda^\to$ over a signature Σ has the formulae $wff_\mathcal{T}(\Sigma, \mathcal{V}_\mathcal{T})$ (they are called λ-terms) and the following equalities:

- ▹ α conversion: $(\lambda X\mathbf{.A}) =_\alpha (\lambda Y\mathbf{.}[Y/X](\mathbf{A}))$
- ▹ β conversion: $(\lambda X\mathbf{.A})\mathbf{B} =_\beta [\mathbf{B}/X](\mathbf{A})$
- ▹ η conversion: $(\lambda X\mathbf{.A}X) =_\eta \mathbf{A}$

The intuitions about functional structure of λ-terms and about free and bound variables are encoded into three transformation rules $\Lambda^\to$: The first rule (α-conversion) just says that we can rename bound variables as we like. β-conversion codifies the intuition behind function application by replacing bound variables with argument. The equality relation induced by the η-reduction is

a special case of the extensionality principle for functions ($f = g$ iff $f(a) = g(a)$ for all possible arguments a): If we apply both sides of the transformation to the same argument – say $\mathbf{B}$ and then we arrive at the right hand side, since $(\lambda X_\alpha.\mathbf{A}X)\mathbf{B} =_\beta \mathbf{AB}$.

We will use a set of bracket elision rules that make the syntax of $\Lambda^\to$ more palatable. This makes $\Lambda^\to$ expressions look much more like regular mathematical notation, but hides the internal structure. Readers should make sure that they can always reconstruct the brackets to make sense of the syntactic notions below.

Simply typed λ-Calculus (Notations)

▷ **Notation 13.0.4 (Application is left-associative)** We abbreviate $(((\mathbf{F}\mathbf{A}^1)\mathbf{A}^2)\ldots)\mathbf{A}^n$ with $\mathbf{F}\mathbf{A}^1\ldots\mathbf{A}^n$ eliding the brackets and further with $\mathbf{F}\overline{\mathbf{A}^n}$ in a kind of vector notation.

▷ A . stands for a left bracket whose partner is as far right as is consistent with existing brackets; i.e. $\mathbf{A}.\mathbf{BC}$ abbreviates $\mathbf{A}(\mathbf{BC})$.

▷ **Notation 13.0.5 (Abstraction is right-associative)** We abbreviate $\lambda X^1.\lambda X^2.\cdots\lambda X^n.\mathbf{A}\cdots$ with $\lambda X^1\ldots X^n.\mathbf{A}$ eliding brackets, and further to $\lambda\overline{X^n}.\mathbf{A}$ in a kind of vector notation.

▷ **Notation 13.0.6 (Outer brackets)** Finally, we allow ourselves to elide outer brackets where they can be inferred.

Intuitively, $\lambda X.\mathbf{A}$ is the function f, such that $f(\mathbf{B})$ will yield $\mathbf{A}$, where all occurrences of the formal parameter X are replaced by $\mathbf{B}$.[7]

EdN:7

In this presentation of the simply typed λ-calculus we build-in α-equality and use capture-avoiding substitutions directly. A clean introduction would followed the steps in Chapter 7 by introducing substitutions with a substitutability condition like the one in Definition 7.2.10, then establishing the soundness of α conversion, and only then postulating defining capture-avoiding substitution application as in Definition 7.3.3. The development for $\Lambda^\to$ is directly parallel to the one for PL^1, so we leave it as an exercise to the reader and turn to the computational properties of the λ-calculus.

Computationally, the λ-calculus obtains much of its power from the fact that two of its three equalities can be oriented into a reduction system. Intuitively, we only use the equalities in one direction, i.e. in one that makes the terms "simpler". If this terminates (and is confluent), then we can establish equality of two λ-terms by reducing them to normal forms and comparing them structurally. This gives us a decision procedure for equality. Indeed, we have these properties in $\Lambda^\to$ as we will see below.

αβη-Equality (Overview)

▷ reduction with $\begin{cases} \beta: (\lambda X.\mathbf{A})\mathbf{B} \to_\beta [\mathbf{B}/X](\mathbf{A}) \\ \eta: (\lambda X.\mathbf{A}X) \to_\eta \mathbf{A} \end{cases}$ under $=_\alpha$: $\begin{array}{c} \lambda X.\mathbf{A} \\ =_\alpha \\ \lambda Y.[Y/X](\mathbf{A}) \end{array}$

▷ **Theorem 13.0.7** *$\beta\eta$-reduction is well-typed, terminating and confluent in the presence of $=_\alpha$-conversion.*

▷ **Definition 13.0.8 (Normal Form)** We call a λ-term $\mathbf{A}$ a normal form

[7] EdNote: rationalize the semantic macros for syntax!

(in a reduction system $\mathcal{E}$), iff no rule (from $\mathcal{E}$) can be applied to $\mathbf{A}$.

▷ **Corollary 13.0.9** *$\beta\eta$-reduction yields unique normal forms (up to α-equivalence).*

We will now introduce some terminology to be able to talk about λ-terms and their parts.

Syntactic Parts of λ-Terms

▷ **Definition 13.0.10 (Parts of λ-Terms)** We can always write a λ-term in the form $\mathbf{T} = \lambda X^1 \ldots X^k . \mathbf{H}\mathbf{A}^1 \ldots \mathbf{A}^n$, where $\mathbf{H}$ is not an application. We call

- ▷ $\mathbf{H}$ the syntactic head of $\mathbf{T}$
- ▷ $\mathbf{H}\mathbf{A}^1 \ldots \mathbf{A}^n$ the matrix of $\mathbf{T}$, and
- ▷ $\lambda X^1 \ldots X^k .$ (or the sequence $X_1, \ldots, X_k$) the binder of $\mathbf{T}$

▷ **Definition 13.0.11** Head Reduction always has a unique β redex

$$(\lambda \overline{X^n} . (\lambda Y . \mathbf{A})\mathbf{B}^1 \ldots \mathbf{B}^n) \to^h_\beta (\lambda \overline{X^n} . [\mathbf{B}^1/Y](\mathbf{A})\mathbf{B}^2 \ldots \mathbf{B}^n)$$

▷ **Theorem 13.0.12** *The syntactic heads of β-normal forms are constant or variables.*

▷ **Definition 13.0.13** Let $\mathbf{A}$ be a λ-term, then the syntactic head of the β-normal form of $\mathbf{A}$ is called the head symbol of $\mathbf{A}$ and written as head($\mathbf{A}$). We call a λ-term a j-projection, iff its head is the j^{th} bound variable.

▷ **Definition 13.0.14** We call a λ-term a η-long form, iff its matrix has base type.

▷ **Definition 13.0.15** η-Expansion makes η-long forms

$$\eta\big[(\lambda X^1 \ldots X^n . \mathbf{A})\big] := (\lambda X^1 \ldots X^n . \lambda Y^1 \ldots Y^m . \mathbf{A} Y^1 \ldots Y^m)$$

▷ **Definition 13.0.16** Long $\beta\eta$-normal form, iff it is β-normal and η-long.

η long forms are structurally convenient since for them, the structure of the term is isomorphic to the structure of its type (argument types correspond to binders): if we have a term $\mathbf{A}$ of type $\overline{\alpha_n} \to \beta$ in η-long form, where $\beta \in \mathcal{B}\,\mathcal{T}$, then $\mathbf{A}$ must be of the form $\lambda \overline{X_\alpha{}^n} . \mathbf{B}$, where $\mathbf{B}$ has type β. Furthermore, the set of η-long forms is closed under β-equality, which allows us to treat the two equality theories of $\Lambda^\to$ separately and thus reduce argumentational complexity.

Chapter 14

Computational Properties of λ-Calculus

As we have seen above, the main contribution of the λ-calculus is that it casts the comprehension and (functional) extensionality axioms in a way that is more amenable to automation in reasoning systems, since they can be oriented into a confluent and terminating reduction system. In this Chapter we prove the respective properties. We start out with termination, since we will need it later in the proof of confluence.

14.1 Termination of β-reduction

We will use the termination of β reduction to present a very powerful proof method, called the "logical relations method", which is one of the basic proof methods in the repertoire of a proof theorist, since it can be extended to many situations, where other proof methods have no chance of succeeding.

Before we start into the termination proof, we convince ourselves that a straightforward induction over the structure of expressions will not work, and we need something more powerful.

Termination of β-Reduction

▷ only holds for the typed case
$(\lambda X.XX)(\lambda X.XX) \to_\beta (\lambda X.XX)(\lambda X.XX)$

▷ **Theorem 14.1.1 (Typed β-Reduction terminates)** *For all* $\mathbf{A} \in wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, *the chain of reductions from* $\mathbf{A}$ *is finite.*

▷ proof attempts:

- ▷ Induction on the structure $\mathbf{A}$ must fail, since this would also work for the untyped case.
- ▷ Induction on the type of $\mathbf{A}$ must fail, since β-reduction conserves types.

▷ combined induction on both: Logical Relations [Tait 1967]

The overall shape of the proof is that we reason about two relations: $\mathcal{SR}$ and $\mathcal{LR}$ between λ-terms and their types. The first is the one that we are interested in, $\mathcal{LR}(\mathbf{A}, \alpha)$ essentially states the

property that $\beta\eta$ reduction terminates at $\mathbf{A}$. Whenever the proof needs to argue by induction on types it uses the "logical relation" $\mathcal{LR}$, which is more "semantic" in flavor. It coincides with $\mathcal{SR}$ on base types, but is defined via a functionality property.

Relations $\mathcal{SR}$ and $\mathcal{LR}$

▷ **Definition 14.1.2** $\mathbf{A}$ is called strongly reducing at type α (write $\mathcal{SR}(\mathbf{A},\alpha)$), iff each chain β-reductions from $\mathbf{A}$ terminates.

▷ We define a logical relation $\mathcal{LR}$ inductively on the structure of the type

- ▹ α base type: $\mathcal{LR}(\mathbf{A},\alpha)$, iff $\mathcal{SR}(\mathbf{A},\alpha)$
- ▹ $\mathcal{LR}(\mathbf{C},\alpha\to\beta)$, iff $\mathcal{LR}(\mathbf{CA},\beta)$ for all $\mathbf{A}\in \mathit{wff}_\alpha(\Sigma,\mathcal{V}_\mathcal{T})$ with $\mathcal{LR}(\mathbf{A},\alpha)$.

Proof: Termination Proof

▷ **P.1** $\mathcal{LR}\subseteq\mathcal{SR}$ (Lemma 14.1.4 b))

$\mathbf{A}\in \mathit{wff}_\alpha(\Sigma,\mathcal{V}_\mathcal{T})$ implies $\mathcal{LR}(\mathbf{A},\alpha)$ (Theorem 14.1.8 with $\sigma=\emptyset$)

thus $\mathcal{SR}(\mathbf{A},\alpha)$. □

P.2 **P.3** ▷ **Lemma 14.1.3 ($\mathcal{SR}$ is closed under subterms)** *If $\mathcal{SR}(\mathbf{A},\alpha)$ and $\mathbf{B}_\beta$ is a subterm of $\mathbf{A}$, then $\mathcal{SR}(\mathbf{B},\beta)$.*

▷ Proof Idea: Every infinite β-reduction from $\mathbf{B}$ would be one from $\mathbf{A}$. □

The termination proof proceeds in two steps, the first one shows that $\mathcal{LR}$ is a sub-relation of $\mathcal{SR}$, and the second that $\mathcal{LR}$ is total on λ-terms. Togther they give the termination result.

The next result proves two important technical side results for the termination proofs in a joint induction over the structure of the types involved. The name "rollercoaster lemma" alludes to the fact that the argument starts with base type, where things are simple, and iterates through the two parts each leveraging the proof of the other to higher and higher types.

$\mathcal{LR}\subseteq\mathcal{SR}$ (Rollercoaster Lemma)

▷ **Lemma 14.1.4 (Rollercoaster Lemma)**

a) If h is a constant or variable of type $\overline{\alpha_n}\to\alpha$ and $\mathcal{SR}(\mathbf{A}^i,\alpha^i)$, then $\mathcal{LR}(h\overline{\mathbf{A}^n},\alpha)$.

b) $\mathcal{LR}(\mathbf{A},\alpha)$ implies $\mathcal{SR}(\mathbf{A},\alpha)$.

Proof: we prove both assertions by simultaneous induction on α

▷ **P.1.1** α base type:

P.1.1.1.1 a): $h\overline{\mathbf{A}^n}$ is strongly reducing, since the $\mathbf{A}^i$ are (brackets!)

P.1.1.1.2 so $\mathcal{LR}(h\overline{\mathbf{A}^n},\alpha)$ as α is a base type ($\mathcal{SR}=\mathcal{LR}$) □

P.1.1.1.2 b): by definition □

$\alpha=\beta\to\gamma$:

P.1.2.1.1 a): Let $\mathcal{LR}(\mathbf{B},\beta)$.

P.1.2.1.1.2 by IH b) we have $\mathcal{SR}(\mathbf{B},\beta)$, and $\mathcal{LR}((h\overline{\mathbf{A}^n})\mathbf{B},\gamma)$ by IH a)

P.1.2.1.1.3 so $\mathcal{LR}(h\overline{\mathbf{A}^n},\alpha)$ by definition. □

P.1.2.1.2 b): Let $\mathcal{LR}(\mathbf{A},\alpha)$ and $X_\beta \notin \mathrm{free}(\mathbf{A})$.

P.1.2.1.2.2 $\mathcal{LR}(X,\beta)$ by IH a) with $n=0$, thus $\mathcal{LR}(\mathbf{A}X,\gamma)$ by definition.

P.1.2.1.2.3 By IH b) we have $\mathcal{SR}(\mathbf{A}X,\gamma)$ and by Lemma 14.1.3 $\mathcal{SR}(\mathbf{A},\alpha)$. □

□

□

The part of the rollercoaster lemma we are really interested in is part b). But part a) will become very important for the case where $n=0$; here it states that constants and variables are $\mathcal{LR}$.

The next step in the proof is to show that all well-formed formulae are $\mathcal{LR}$. For that we need to prove closure of $\mathcal{LR}$ under $=_\beta$ expansion

β-Expansion Lemma

▷ **Lemma 14.1.5** *If* $\mathcal{LR}([\mathbf{B}/X](\mathbf{A}),\alpha)$ *and* $\mathcal{LR}(\mathbf{B},\beta)$ *for* $X_\beta \notin \mathrm{free}(\mathbf{B})$*, then* $\mathcal{LR}((\lambda X_\alpha.\mathbf{A})\mathbf{B},\alpha)$.

▷ Proof:

P.1 Let $\alpha = \overline{\gamma_i} \to \delta$ where δ base type and $\mathcal{LR}(\mathbf{C}^i,\gamma^i)$

P.2 It is sufficient to show that $\mathcal{SR}((\lambda X.\mathbf{A})\mathbf{B})\overline{\mathbf{C}},\delta)$, as δ base type

P.3 We have $\mathcal{LR}([\mathbf{B}/X](\mathbf{A})\overline{\mathbf{C}},\delta)$ by hypothesis and definition of $\mathcal{LR}$.

P.4 thus $\mathcal{SR}([\mathbf{B}/X](\mathbf{A})\overline{\mathbf{C}},\delta)$, as δ base type.

P.5 in particular $\mathcal{SR}([\mathbf{B}/X](\mathbf{A}),\alpha)$ and $\mathcal{SR}(\mathbf{C}^i,\gamma^i)$ (subterms)

P.6 $\mathcal{SR}(\mathbf{B},\beta)$ by hypothesis and Lemma 14.1.4

P.7 So an infinite reduction from $((\lambda X.\mathbf{A})\mathbf{B})\overline{\mathbf{C}}$ cannot solely consist of redexes from $[\mathbf{B}/X](\mathbf{A})$ and the $\mathbf{C}^i$.

P.8 so an infinite reduction from $((\lambda X.\mathbf{A})\mathbf{B})\overline{\mathbf{C}}$ must have the form

$$\begin{array}{lll}((\lambda X.\mathbf{A})\mathbf{B})\overline{\mathbf{C}} & \to^*_\beta & ((\lambda X.\mathbf{A}')\mathbf{B}')\overline{\mathbf{C}'} \\ & \to^1_\beta & [\mathbf{B}'/X](\mathbf{A}')\overline{\mathbf{C}'} \\ & \to^*_\beta & \ldots\end{array}$$

where $\mathbf{A} \to^*_\beta \mathbf{A}'$, $\mathbf{B} \to^*_\beta \mathbf{B}'$ and $\mathbf{C}^i \to^*_\beta \mathbf{C}^{i'}$

P.9 so we have $[\mathbf{B}/X](\mathbf{A}) \to^*_\beta [\mathbf{B}'/X](\mathbf{A}')$

P.10 so we have the infinite reduction

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

which contradicts our assumption □

▷ **Lemma 14.1.6 ($\mathcal{LR}$ is closed under β-expansion)** *If $\mathbf{C} \to_\beta \mathbf{D}$ and $\mathcal{LR}(\mathbf{D}, \alpha)$, so is $\mathcal{LR}(\mathbf{C}, \alpha)$.*

Note that this Lemma is one of the few places in the termination proof, where we actually look at the properties of $=_\beta$ reduction.

We now prove that every well-formed formula is related to its type by $\mathcal{LR}$. But we cannot prove this by a direct induction. In this case we have to strengthen the statement of the theorem – and thus the inductive hypothesis, so that we can make the step cases go through. This is common for non-trivial induction proofs. Here we show instead that *every instance* of a well-formed formula is related to its type by $\mathcal{LR}$; we will later only use this result for the cases of the empty substitution, but the stronger assertion allows a direct induction proof.

$\mathbf{A} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ implies $\mathcal{LR}(\mathbf{A}, \alpha)$

▷ **Definition 14.1.7** We write $\mathcal{LR}(\sigma)$ if $\mathcal{LR}(\sigma(X_\alpha), \alpha)$ for all $X \in \mathbf{supp}(\sigma)$.

▷ **Theorem 14.1.8** *If $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, then $\mathcal{LR}(\sigma(\mathbf{A}), \alpha)$ for any substitution σ with $\mathcal{LR}(\sigma)$.*

▷ Proof: by induction on the structure of $\mathbf{A}$

P.1.1 $\mathbf{A} = X_\alpha \in \mathbf{supp}(\sigma)$: then $\mathcal{LR}(\sigma(\mathbf{A}), \alpha)$ by assumption

P.1.2 $\mathbf{A} = X \notin \mathbf{supp}(\sigma)$: then $\sigma(\mathbf{A}) = \mathbf{A}$ and $\mathcal{LR}(\mathbf{A}, \alpha)$ by Lemma 14.1.4 with $n = 0$.

P.1.3 $\mathbf{A} \in \Sigma$: then $\sigma(\mathbf{A}) = \mathbf{A}$ as above

P.1.4 $\mathbf{A} = \mathbf{BC}$: by IH $\mathcal{LR}(\sigma(\mathbf{B}), \gamma \to \alpha)$ and $\mathcal{LR}(\sigma(\mathbf{C}), \gamma)$

P.1.4.2 so $\mathcal{LR}(\sigma(\mathbf{B})\sigma(\mathbf{C}), \alpha)$ by definition of $\mathcal{LR}$. □

P.1.5 $\mathbf{A} = \lambda X_\beta.\mathbf{C}_\gamma$: Let $\mathcal{LR}(\mathbf{B}, \beta)$ and $\theta := \sigma, [\mathbf{B}/X]$, then θ meets the conditions of the IH.

P.1.5.2 Moreover $\sigma(\lambda X_\beta.\mathbf{C}_\gamma)\mathbf{B} \to_\beta \sigma, [\mathbf{B}/X](\mathbf{C}) = \theta(\mathbf{C})$.

P.1.5.3 Now, $\mathcal{LR}(\theta(\mathbf{C}), \gamma)$ by IH and thus $\mathcal{LR}(\sigma(\mathbf{A})\mathbf{B}, \gamma)$ by Lemma 14.1.6.

P.1.5.4 So $\mathcal{LR}(\sigma(\mathbf{A}), \alpha)$ by definition of $\mathcal{LR}$. □

□

In contrast to the proof of the roller coaster Lemma above, we prove the assertion here by an induction on the structure of the λ-terms involved. For the base cases, we can directly argue with the first assertion from Lemma 14.1.4, and the application case is immediate from the definition of $\mathcal{LR}$. Indeed, we defined the auxiliary relation $\mathcal{LR}$ exclusively that the application case – which cannot be proven by a direct structural induction; remember that we needed induction on types in Lemma 14.1.4– becomes easy.

The last case on λ-abstraction reveals why we had to strengthen the inductive hypothesis: $=_\beta$ reduction introduces a substitution which may increase the size of the subterm, which in turn keeps us from applying the inductive hypothesis. Formulating the assertion directly under all possible $\mathcal{LR}$ substitutions unblocks us here.

This was the last result we needed to complete the proof of termiation of β-reduction.

Remark: If we are only interested in the termination of head reductions, we can get by with a much simpler version of this lemma, that basically relies on the uniqueness of head β reduction.

Closure under Head β-Expansion (weakly reducing)

▷ **Lemma 14.1.9 ($\mathcal{LR}$ is closed under head β-expansion)** *If* $\mathbf{C} \to_\beta^h \mathbf{D}$ *and* $\mathcal{LR}(\mathbf{D},\alpha)$*, so is* $\mathcal{LR}(\mathbf{C},\alpha)$.

▷ Proof: by induction over the structure of α

P.1.1 α base type:

P.1.1.1 we have $\mathcal{SR}(\mathbf{D},\alpha)$ by definition

P.1.1.2 so $\mathcal{SR}(\mathbf{C},\alpha)$, since head reduction is unique

P.1.1.3 and thus $\mathcal{LR}(\mathbf{C},\alpha)$. □

P.1.2 $\alpha = \beta \to \gamma$:

P.1.2.1 Let $\mathcal{LR}(\mathbf{B},\beta)$, by definition we have $\mathcal{LR}(\mathbf{DB},\gamma)$.

P.1.2.2 but $\mathbf{CB} \to_\beta^h \mathbf{DB}$, so $\mathcal{LR}(\mathbf{CB},\gamma)$ by IH

P.1.2.3 and $\mathcal{LR}(\mathbf{C},\alpha)$ by definition. □

□

Note: This result only holds for weak reduction (any chain of β head reductions terminates) for strong reduction we need a stronger Lemma.

For the termination proof of head β-reduction we would just use the same proof as above, just for a variant of $\mathcal{SR}$, where $\mathcal{SR}\mathbf{A}\,\alpha$ that only requires that the head reduction sequence out of $\mathbf{A}$ terminates. Note that almost all of the proof except Lemma 14.1.3 (which holds by the same argument) is invariant under this change. Indeed Rick Statman uses this observation in [Sta85] to give a set of conditions when logical relations proofs work.

14.2 Confluence of $\beta\eta$ Conversion

We now turn to the confluence for $\beta\eta$, i.e. that the order of reductions is irrelevant. This entails the uniqueness of $\beta\eta$ normal forms, which is very useful.

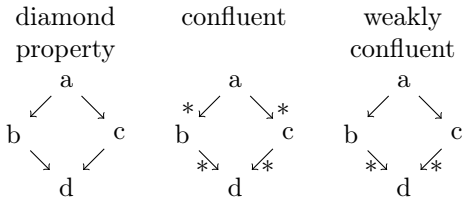
Intuitively confluence of a relation R means that "anything that flows apart will come together again." – and as a consequence normal forms are unique if they exist. But there is more than one way of formalizing that intuition.

▷ Confluence

▷ **Definition 14.2.1 (Confluence)** Let $R \subseteq A^2$ be a relation on a set A, then we say that

- ▷ has a diamond property, iff for every $a,b,c \in A$ with $a \to_R^1 b$ $a \to_R^1 c$ there is a $d \in A$ with $b \to_R^1 d$ and $c \to_R^1 d$.
- ▷ is confluent, iff for every $a,b,c \in A$ with $a \to_R^* b$ $a \to_R^* c$ there is a $d \in A$ with $b \to_R^* d$ and $c \to_R^* d$.

▷ weakly confluent iff for every $a, b, c \in A$ with $a \to_R^1 b$ $a \to_R^1 c$ there is a $d \in A$ with $b \to_R^* d$ and $c \to_R^* d$.



The diamond property is very simple, but not many reduction relations enjoy it. Confluence is the notion that that directly gives us unique normal forms, but is difficult to prove via a digram chase, while weak confluence is amenable to this, does not directly give us confluence.

We will now relate the three notions of confluence with each other: the diamond property (sometimes also called strong confluence) is stronger than confluence, which is stronger than weak confluence

Relating the notions of confluence

▷ **Observation 14.2.2** *If a rewrite relation has a diamond property, then it is weakly confluent.*

▷ **Theorem 14.2.3** *If a rewrite relation has a diamond property, then it is confluent.*

▷ Proof Idea: by a tiling argument, composing 1×1 diamonds to an $n \times m$ diamond. □

▷ **Theorem 14.2.4 (Newman's Lemma)** *If a rewrite relation is terminating and weakly confluent, then it is also confluent.*

Note that Newman's Lemma cannot be proven by a tiling argument since we cannot control the growth of the tiles. There is a nifty proof by Gérard Huet [Hue80] that is worth looking at.

After this excursion into the general theory of reduction relations, we come back to the case at hand: showing the confluence of $\beta\eta$-reduction.

η is very well-behaved – i.e. confluent and terminating

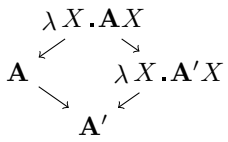
η-Reduction ist terminating and confluent

▷ **Lemma 14.2.5** *η-Reduction ist terminating*

▷ Proof: by a simple counting argument □

▷ **Lemma 14.2.6** *η-reduction is confluent.*

▷ Proof Idea: We show that η-reduction has the diamond property by diagram chase over



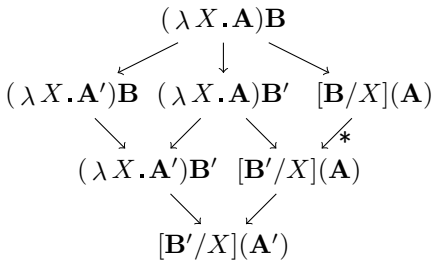
where $\mathbf{A} \to_\eta \mathbf{A}'$. Then the assertion follows by Theorem 14.2.3. □

For β-reduction the situation is a bit more involved, but a simple diagram chase is still sufficient to prove weak confluence, which gives us confluence via Newman's Lemma

β is confluent

▷ **Lemma 14.2.7** *β-Reduction is weakly confluent.*

▷ Proof Idea: by diagram chase over



□

▷ **Corollary 14.2.8** *β-Reduction is confluent.*

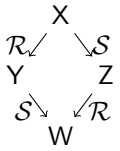
▷ Proof Idea: by Newman's Lemma. □

There is one reduction in the diagram in the proof of Lemma 14.2.7 which (note that $\mathbf{B}$ can occur multiple times in $[\mathbf{B}/X](\mathbf{A})$) is not necessary single-step. The diamond property is broken by the outer two reductions in the diagram as well.

We have shown that the β and η reduction relations are terminating and confluent and terminating individually, now, we have to show that $\beta\eta$ is a well. For that we introduce a new concept.

Commuting Relations

▷ **Definition 14.2.9** Let A be a set, then we say that relations $\mathcal{R} \in A^2$ and $\mathcal{S} \in A^2$ commute, if $X \to_\mathcal{R} Y$ and $X \to_\mathcal{S} Z$ entail the existence of a $W \in A$ with $Y \to_\mathcal{S} W$ and $Z \to_\mathcal{R} W$.



▷ **Observation 14.2.10** *If $\mathcal{R}$ and $\mathcal{S}$ commute, then $\to_\mathcal{R}$ and $\to_\mathcal{S}$ do as well.*

▷ **Observation 14.2.11** *$\mathcal{R}$ is confluent, if $\mathcal{R}$ commutes with itself.*

▷ **Lemma 14.2.12** *If $\mathcal{R}$ and $\mathcal{S}$ are terminating and confluent relations such that $\to^*_{\mathcal{R}}$ and $\to^*_{\mathcal{S}}$ commute, then $\to^*_{\mathcal{R}\cup\mathcal{S}}$ is confluent.*

▷ Proof Sketch: As $\mathcal{R}$ and $\mathcal{S}$ commute, we can reorder any reduction sequence so that all $\mathcal{R}$-reductions precede all $\mathcal{S}$-reductions. As $\mathcal{R}$ is terminating and confluent, the $\mathcal{R}$-part ends in a unique normal form, and as $\mathcal{S}$ is normalizing it must lead to a unique normal form as well. □

This directly gives us our goal.

$\beta\,\eta$ is confluent

▷ **Lemma 14.2.13** *$\to^*_\beta$ and $\to^*_\eta$ commute.*

▷ Proof Sketch: diagram chase □

Chapter 15

The Semantics of the Simply Typed λ-Calculus

The semantics of $\Lambda^{\to}$ is structured around the types. Like the models we discussed before, a model (we call them "algebras", since we do not have truth values in $\Lambda^{\to}$) is a pair $\langle \mathcal{D}, \mathcal{I} \rangle$, where $\mathcal{D}$ is the universe of discourse and $\mathcal{I}$ is the interpretation of constants.

Semantics of $\Lambda^{\to}$

▷ **Definition 15.0.1** We call a collection $\mathcal{D}_{\mathcal{T}} := \{\mathcal{D}_\alpha \,|\, \alpha \in \mathcal{T}\}$ a typed collection (of sets) and a collection $f_{\mathcal{T}} \colon \mathcal{D}_{\mathcal{T}} \to \mathcal{E}_{\mathcal{T}}$, a typed function, iff $f_\alpha \colon \mathcal{D}_\alpha \to \mathcal{E}_\alpha$.

▷ **Definition 15.0.2** A typed collection $\mathcal{D}_{\mathcal{T}}$ is called a frame, iff $\mathcal{D}_{\alpha\to\beta} \subseteq \mathcal{D}_\alpha \to \mathcal{D}_\beta$

▷ **Definition 15.0.3** Given a frame $\mathcal{D}_{\mathcal{T}}$, and a typed function $\mathcal{I} \colon \Sigma \to \mathcal{D}$, then we call $\mathcal{I}_\varphi \colon \mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}}) \to \mathcal{D}$ the value function induced by $\mathcal{I}$, iff

- ▷ $\mathcal{I}_\varphi|_{\mathcal{V}_{\mathcal{T}}} = \varphi$, $\quad \mathcal{I}_\varphi|_\Sigma = \mathcal{I}$
- ▷ $\mathcal{I}_\varphi(\mathbf{AB}) = \mathcal{I}_\varphi(\mathbf{A})(\mathcal{I}_\varphi(\mathbf{B}))$
- ▷ $\mathcal{I}_\varphi(\lambda X_\alpha \mathbf{.A})$ is that function $f \in \mathcal{D}_{\alpha\to\beta}$, such that $f(a) = \mathcal{I}_{\varphi,[a/X]}(\mathbf{A})$ for all $a \in \mathcal{D}_\alpha$

▷ **Definition 15.0.4** We call a frame $\langle \mathcal{D}, \mathcal{I} \rangle$ comprehension-closed or a Σ-algebra, iff $\mathcal{I}_\varphi \colon \mathit{wff}_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}}) \to \mathcal{D}$ is total. (every λ-term has a value)

15.1 Soundness of the Simply Typed λ-Calculus

We will now show is that $\alpha\beta\eta$-reduction does not change the value of formulae, i.e. if $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$, then $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$, for all $\mathcal{D}$ and φ. We say that the reductions are sound. As always, the main tool for proving soundess is a substitution value lemma. It works just as always and verifies that we the definitions are in our semantics plausible.

Substitution Value Lemma for λ-Terms

▷ **Lemma 15.1.1 (Substitution Value Lemma)** *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be terms, then* $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\psi(\mathbf{A})$, *where* $\psi = \varphi, [\mathcal{I}_\varphi(\mathbf{B})/X]$

▷ Proof: by induction on the depth of $\mathbf{A}$

P.1 we have five cases

P.1.1 $\mathbf{A} = X$: Then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](X)) = \mathcal{I}_\varphi(\mathbf{B}) = \psi(X) = \mathcal{I}_\psi(X) = \mathcal{I}_\psi(\mathbf{A})$.

P.1.2 $\mathbf{A} = Y \neq X$ and $Y \in \mathcal{V}_\tau$: then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](Y)) = \mathcal{I}_\varphi(Y) = \varphi(Y) = \psi(Y) = \mathcal{I}_\psi(Y) = \mathcal{I}_\psi(\mathbf{A})$.

P.1.3 $\mathbf{A} \in \Sigma$: This is analogous to the last case.

P.1.4 $\mathbf{A} = \mathbf{CD}$: then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{CD})) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{C})[\mathbf{B}/X](\mathbf{D})) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{C}))(\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{D}))) = \mathcal{I}_\psi(\mathbf{C})(\mathcal{I}_\psi(\mathbf{D})) = \mathcal{I}_\psi(\mathbf{CD}) = \mathcal{I}_\psi(\mathbf{A})$

P.1.5 $\mathbf{A} = \lambda Y_\alpha.\mathbf{C}$:

P.1.5.1 We can assume that $X \neq Y$ and $Y \notin \mathrm{free}(\mathbf{B})$

P.1.5.2 Thus for all $a \in \mathcal{D}_\alpha$ we have $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}))(a) = \mathcal{I}_\varphi([\mathbf{B}/X](\lambda Y.\mathbf{C}))(a) = \mathcal{I}_\varphi(\lambda Y.[\mathbf{B}/X](\mathbf{C}))(a) = \mathcal{I}_{\varphi,[a/Y]}([\mathbf{B}/X](\mathbf{C})) = \mathcal{I}_{\psi,[a/Y]}(\mathbf{C}) = \mathcal{I}_\psi(\lambda Y.\mathbf{C})(a) = \mathcal{I}_\psi(\mathbf{A})(a)$ □

□

Soundness of $\alpha\beta\eta$-Equality

▷ **Theorem 15.1.2** *Let* $\mathcal{A} := \langle \mathcal{D}, \mathcal{I} \rangle$ *be a* Σ*-algebra and* $Y \notin \mathrm{free}(\mathbf{A})$, *then* $\mathcal{I}_\varphi(\lambda X.\mathbf{A}) = \mathcal{I}_\varphi(\lambda Y.[Y/X]\mathbf{A})$ *for all assignments* φ.

▷ Proof: by substitution value lemma

$$\begin{aligned}\mathcal{I}_\varphi(\lambda Y.[Y/X]\mathbf{A})\,@\,\mathsf{a} &= \mathcal{I}_{\varphi,[a/Y]}([Y/X](\mathbf{A})) \\ &= \mathcal{I}_{\varphi,[a/X]}(\mathbf{A}) \\ &= \mathcal{I}_\varphi(\lambda X.\mathbf{A})\,@\,\mathsf{a}\end{aligned}$$

▷ **Theorem 15.1.3** *If* $\mathcal{A} := \langle \mathcal{D}, \mathcal{I} \rangle$ *is a* Σ*-algebra and* X *not bound in* $\mathbf{A}$, *then* $\mathcal{I}_\varphi((\lambda X.\mathbf{A})\mathbf{B}) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}))$.

▷ Proof: by substitution value lemma again

$$\begin{aligned}\mathcal{I}_\varphi((\lambda X.\mathbf{A})\mathbf{B}) &= \mathcal{I}_\varphi(\lambda X.\mathbf{A})\,@\,\mathcal{I}_\varphi(\mathbf{B}) \\ &= \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{A}) \\ &= \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}))\end{aligned}$$

Soundness of $\alpha\beta\eta$ (continued)

▷ **Theorem 15.1.4** *If* $X \notin \mathrm{free}(\mathbf{A})$, *then* $\mathcal{I}_\varphi(\lambda X.\mathbf{A}X) = \mathcal{I}_\varphi(\mathbf{A})$ *for all* φ.

▷ Proof: by calculation

$$\begin{array}{rcll}\mathcal{I}_\varphi(\lambda X.\mathbf{A}X)\,@\,\mathsf{a} &=& \mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{A}X) & \\ &=& \mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{A})\,@\,\mathcal{I}_{\varphi,[\mathsf{a}/X]}(X) & \\ &=& \mathcal{I}_\varphi(\mathbf{A})\,@\,\mathcal{I}_{\varphi,[\mathsf{a}/X]}(X) & \text{as } X \notin \text{free}(\mathbf{A}).\\ &=& \mathcal{I}_\varphi(\mathbf{A})\,@\,\mathsf{a} & \end{array}$$

▷ **Theorem 15.1.5** *$\alpha\beta\eta$-equality is sound wrt. Σ-algebras. (if $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$, then $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$ for all assignments φ)*

15.2 Completeness of $\alpha\beta\eta$-Equality

We will now show is that $\alpha\beta\eta$-equality is complete for the semantics we defined, i.e. that whenever $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$ for all variable assignments φ, then $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$. We will prove this by a model existence argument: we will construct a model $\mathcal{M} := \langle \mathcal{D}, \mathcal{I}\rangle$ such that if $\mathbf{A} \neq_{\alpha\beta\eta} \mathbf{B}$ then $\mathcal{I}_\varphi(\mathbf{A}) \neq \mathcal{I}_\varphi(\mathbf{B})$ for some φ.

As in other completeness proofs, the model we will construct is a "ground term model", i.e. a model where the carrier (the frame in our case) consists of ground terms. But in the λ-calculus, we have to do more work, as we have a non-trivial built-in equality theory; we will construct the "ground term model" from sets of normal forms. So we first fix some notations for them.

Normal Forms in the simply typed λ-calculus

▷ **Definition 15.2.1** We call a term $\mathbf{A} \in \mathit{wff}_\mathcal{T}(\Sigma, \mathcal{V}_\mathcal{T})$ a β normal form iff there is no $\mathbf{B} \in \mathit{wff}_\mathcal{T}(\Sigma, \mathcal{V}_\mathcal{T})$ with $\mathbf{A} \to_\beta \mathbf{B}$.

We call $\mathbf{N}$ a β normal form of $\mathbf{A}$, iff $\mathbf{N}$ is a β-normal form and $\mathbf{A} \to_\beta \mathbf{N}$.

We denote the set of β-normal forms with $\mathit{wff}_\mathcal{T}(\Sigma, \mathcal{V}_\mathcal{T})\downarrow_{\beta\,\eta}$.

▷ We have just proved that $\beta\eta$-reduction is terminating and confluent, so we have

▷ **Corollary 15.2.2 (Normal Forms)** *Every $\mathbf{A} \in \mathit{wff}_\mathcal{T}(\Sigma, \mathcal{V}_\mathcal{T})$ has a unique β normal form ($\beta\eta$, long $\beta\eta$ normal form), which we denote by $\mathbf{A}\downarrow_\beta$ ($\mathbf{A}\downarrow_{\beta\,\eta}$ $\mathbf{A}\downarrow^l_{\beta\,\eta}$)*

The term frames will be a quotient spaces over the equality relations of the λ-calculus, so we introduce this construction generally.

Frames and Quotients

▷ **Definition 15.2.3** Let $\mathcal{D}$ be a frame and $\sim$ a typed equivalence relation on $\mathcal{D}$, then we call $\sim$ a congruence on $\mathcal{D}$, iff $f \sim f'$ and $g \sim g'$ imply $f(g) \sim f'(g')$.

▷ **Definition 15.2.4** We call a congruence $\sim$ functional, iff for all $f, g \in \mathcal{D}_{\alpha\to\beta}$ the fact that $f(a) \sim g(a)$ holds for all $a \in \mathcal{D}_\alpha$ implies that $f \sim g$.

▷ **Example 15.2.5** $=_\beta$ ($=_{\beta\eta}$) is a (functional) congruence on $c\,wff_{\mathcal{T}}(\Sigma)$ by definition.

▷ **Theorem 15.2.6** *Let $\mathcal{D}$ be a Σ-frame and $\sim$ a functional congruence on $\mathcal{D}$, then the quotient space $\mathcal{D}/\sim$ is a Σ-frame.*

▷ Proof:

P.1 $\mathcal{D}/\sim = \{[f]_\sim \mid f \in \mathcal{D}\}$, define $[f]_\sim([a]_\sim) := [f(a)]_\sim$.

P.2 This only depends on equivalence classes: Let $f' \in [f]_\sim$ and $a' \in [a]_\sim$.

P.3 Then $[f(a)]_\sim = [f'(a)]_\sim = [f'(a')]_\sim = [f(a')]_\sim$

P.4 To see that we have $[f]_\sim = [g]_\sim$, iff $f \sim g$, iff $f(a) = g(a)$ since $\sim$ is functional.

P.5 This is the case iff $[f(a)]_\sim = [g(a)]_\sim$, iff $[f]_\sim([a]_\sim) = [g]_\sim([a]_\sim)$ for all $a \in \mathcal{D}_\alpha$ and thus for all $[a]_\sim \in \mathcal{D}/\sim$. □

To apply this result, we have to establish that $\beta\eta$-equality is a functional congruence.
We first establish $\beta\eta$ as a functional congruence on $wff_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})$ and then specialize this result to show that is is also functional on $c\,wff_{\mathcal{T}}(\Sigma)$ by a grounding argument.

$\beta\eta$-Equivalence as a Functional Congruence

▷ **Lemma 15.2.7** *$\beta\eta$-equality is a functional congruence on $wff_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})$.*

▷ Proof: Let $\mathbf{AC} =_{\beta\eta} \mathbf{BC}$ for all $\mathbf{C}$ and $X \in (\mathcal{V}_\gamma \setminus (\mathrm{free}(\mathbf{A}) \cup \mathrm{free}(\mathbf{B})))$.

P.1 then (in particular) $\mathbf{A}X =_{\beta\eta} \mathbf{B}X$, and

P.2 $(\lambda X.\mathbf{A}X) =_{\beta\eta} (\lambda X.\mathbf{B}X)$, since $\beta\eta$-equality acts on subterms.

P.3 By definition we have $\mathbf{A} =_\eta (\lambda X_\alpha.\mathbf{A}X) =_{\beta\eta} (\lambda X_\alpha.\mathbf{B}X) =_\eta \mathbf{B}$. □

▷ **Definition 15.2.8** We call an injective substitution $\sigma\colon \mathrm{free}(\mathbf{C}) \to \Sigma$ a grounding substitution for $\mathbf{C} \in wff_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})$, iff no $\sigma(X)$ occurs in $\mathbf{C}$.

Observation: They always exist, since all Σ_α are infinite and $\mathrm{free}(\mathbf{C})$ is finite.

⋙ **Theorem 15.2.9** *$\beta\eta$-equality is a functional congruence on $c\,wff_{\mathcal{T}}(\Sigma)$.*

▷ Proof: We use Lemma 15.2.7

P.1 Let $\mathbf{A}, \mathbf{B} \in c\,wff_{(\alpha\to\beta)}(\Sigma)$, such that $\mathbf{A} \neq_{\beta\eta} \mathbf{B}$.

P.2 As $\beta\eta$ is functional on $wff_{\mathcal{T}}(\Sigma, \mathcal{V}_{\mathcal{T}})$, there must be a $\mathbf{C}$ with $\mathbf{AC} \neq_{\beta\eta} \mathbf{BC}$.

P.3 Now let $\mathbf{C}' := \sigma(\mathbf{C})$, for a grounding substitution σ.

P.4 Any $\beta\eta$ conversion sequence for $\mathbf{AC}' \neq_{\beta\eta} \mathbf{BC}'$ induces one for $\mathbf{AC} \neq_{\beta\eta} \mathbf{BC}$.

P.5 Thus we have shown that $\mathbf{A} \neq_{\beta\eta} \mathbf{B}$ entails $\mathbf{AC}' \neq_{\beta\eta} \mathbf{BC}'$. □

Note that: the result for $c\,wff_{\mathcal{T}}(\Sigma)$ 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 $\{c\} = c\,wff_\iota(\Sigma)$ (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 $wff_{\mathcal{T}}(\Sigma,\mathcal{V}_{\mathcal{T}})$ we could always have used a variable.

This completes the preparation and we can define the notion of a term algebra, i.e. a Σ-algebra whose frame is made of $\beta\eta$-normal λ-terms.

A Herbrand Model for $\Lambda^{\to}$

▷ **Definition 15.2.10** We call $\mathcal{T}_{\beta\eta} := \langle c\,wff_{\mathcal{T}}(\Sigma)\downarrow_{\beta\,\eta}, \mathcal{I}^{\beta\,\eta}\rangle$ the Σ term algebra, if $\mathcal{I}^{\beta\,\eta} = \mathrm{Id}_\Sigma$.

▷ The name "term algebra" in the previous definition is justified by the following

▷ **Theorem 15.2.11** *$\mathcal{T}_{\beta\eta}$ is a Σ-algebra*

▷ Proof: We use the work we did above

P.1 Note that $c\,wff_{\mathcal{T}}(\Sigma)\downarrow_{\beta\,\eta} = c\,wff_{\mathcal{T}}(\Sigma)/=_{\beta\eta}$ and thus a Σ-frame by Theorem 15.2.6 and Lemma 15.2.7.

P.2 So we only have to show that the value function $\mathcal{I}^{\beta\,\eta} = \mathrm{Id}_\Sigma$ is total.

P.3 Let φ be an assignment into $c\,wff_{\mathcal{T}}(\Sigma)\downarrow_{\beta\,\eta}$.

P.4 Note that $\sigma := \varphi|_{\mathrm{free}(\mathbf{A})}$ is a substitution, since $\mathrm{free}(\mathbf{A})$ is finite.

P.5 A simple induction on the structure of $\mathbf{A}$ shows that $\mathcal{I}^{\beta\,\eta}_{\varphi}(\mathbf{A}) = \sigma(\mathbf{A})\downarrow_{\beta\,\eta}$.

P.6 So the value function is total since substitution application is. □

And as always, once we have a term model, showing completeness is a rather simple exercise.

We can see that $\alpha\beta\eta$-equality is complete for the class of Σ-algebras, i.e. if the equation $\mathbf{A} = \mathbf{B}$ is valid, then $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$. Thus $\alpha\beta\eta$ equivalence fully characterizes equality in the class of all Σ-algebras.

Completetness of $\alpha\beta\eta$-Equality

▷ **Theorem 15.2.12** $\mathbf{A} = \mathbf{B}$ *is valid in the class of Σ-algebras, iff* $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$.

▷ Proof: For $\mathbf{A}$, $\mathbf{B}$ closed this is a simple consequence of the fact that $\mathcal{T}_{\beta\eta}$ is a Σ-algebra.

P.1 If $\mathbf{A} = \mathbf{B}$ is valid in all Σ-algebras, it must be in $\mathcal{T}_{\beta\eta}$ and in particular $\mathbf{A}\downarrow_{\beta\,\eta} = \mathcal{I}^{\beta\,\eta}(\mathbf{A}) = \mathcal{I}^{\beta\,\eta}(\mathbf{B}) = \mathbf{B}\downarrow_{\beta\,\eta}$ and therefore $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$.

P.2 If the equation has free variables, then the argument is more subtle.

P.3 Let σ be a grounding substitution for $\mathbf{A}$ and $\mathbf{B}$ and φ the induced variable assignment.

P.4 Thus $\mathcal{I}^{\beta\,\eta}{}_{\varphi}(\mathbf{A}) = \mathcal{I}^{\beta\,\eta}{}_{\varphi}(\mathbf{B})$ is the $\beta\eta$-normal form of $\sigma(\mathbf{A})$ and $\sigma(\mathbf{B})$.

P.5 Since φ is a structure preserving homomorphism on well-formed formulae, $\varphi^{-1}(\mathcal{I}^{\beta\,\eta}{}_{\varphi}(\mathbf{A}))$ is the is the $\beta\eta$-normal form of both $\mathbf{A}$ and $\mathbf{B}$ and thus $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$. □

Theorem 15.2.12 and Theorem 15.1.5 complete our study of the sematnics of the simply-typed λ-calculus by showing that it is an adequate logic for modeling (the equality) of functions and their applications.

Chapter 16

Simply Typed λ-Calculus via Inference Systems

Now, we will look at the simply typed λ-calculus again, but this time, we will present it as an inference system for well-typedness jugdments. This more modern way of developing type theories is known to scale better to new concepts.

Simply Typed λ-Calculus as an Inference System: Terms

▷ Idea: Develop the λ-calculus in two steps

- ▷ A context-free grammar for "raw λ-terms" (for the structure)
- ▷ Identify the well-typed λ-terms in that (cook them until well-typed)

▷ **Definition 16.0.1** A grammar for the raw terms of the simply typed λ-calculus:

$$\begin{array}{lll} \alpha & :== & c \mid \alpha \to \alpha \\ \Sigma & :== & \cdot \mid \Sigma, [c : \text{type}] \mid \Sigma, [c : \alpha] \\ \Gamma & :== & \cdot \mid \Gamma, [x : \alpha] \\ \mathbf{A} & :== & c \mid X \mid \mathbf{A}^1 \mathbf{A}^2 \mid \lambda X_\alpha . \mathbf{A} \end{array}$$

▷ Then: Define all the operations that are possible at the "raw terms level", e.g. realize that signatures and contexts are partial functions to types.

Simply Typed λ-Calculus as an Inference System: Judgments

▷ **Definition 16.0.2** Judgments make statements about complex properties of the syntactic entities defined by the grammar.

▷ **Definition 16.0.3** Judgments for the simply typed λ-calculus

$\vdash \Sigma : \text{sig}$	Σ is a well-formed signature
$\Sigma \vdash \alpha : \text{type}$	α is a well-formed type given the type assumptions in Σ
$\Sigma \vdash \Gamma : \text{ctx}$	Γ is a well-formed context given the type assumptions in Σ
$\Gamma \vdash_\Sigma \mathbf{A} : \alpha$	$\mathbf{A}$ has type α given the type assumptions in Σ and Γ

Simply Typed λ-Calculus as an Inference System: Rules

▷ $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\tau)$, iff $\Gamma \vdash_\Sigma \mathbf{A}: \alpha$ derivable in

$$\frac{\Sigma \vdash \Gamma : \mathrm{ctx} \quad \Gamma(X) = \alpha}{\Gamma \vdash_\Sigma X: \alpha}\text{wff:var} \qquad \frac{\Sigma \vdash \Gamma : \mathrm{ctx} \quad \Sigma(c) = \alpha}{\Gamma \vdash_\Sigma c: \alpha}\text{wff:const}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A}: \beta \to \alpha \quad \Gamma \vdash_\Sigma \mathbf{B}: \beta}{\Gamma \vdash_\Sigma \mathbf{AB}: \alpha}\text{wff:app} \qquad \frac{\Gamma, [X : \beta] \vdash_\Sigma \mathbf{A}: \alpha}{\Gamma \vdash_\Sigma \lambda X_\beta.\mathbf{A}: \beta \to \alpha}\text{wff:abs}$$

Oops: this looks surprisingly like a natural deduction calculus. (⇝ Curry Howard Isomorphism)

⋙ To be complete, we need rules for well-formed signatures, types and contexts

$$\frac{}{\vdash \cdot : \mathrm{sig}}\text{sig:empty} \qquad \frac{\vdash \Sigma : \mathrm{sig}}{\vdash \Sigma, [\alpha : \mathrm{type}] : \mathrm{sig}}\text{sig:type}$$

$$\frac{\vdash \Sigma : \mathrm{sig} \quad \Sigma \vdash \alpha : \mathrm{type}}{\vdash \Sigma, [c : \alpha] : \mathrm{sig}}\text{sig:const}$$

$$\frac{\Sigma \vdash \alpha : \mathrm{type} \quad \Sigma \vdash \beta : \mathrm{type}}{\Sigma \vdash \alpha \to \beta : \mathrm{type}}\text{typ:fn} \qquad \frac{\vdash \Sigma : \mathrm{sig} \quad \Sigma(\alpha) = \mathrm{type}}{\Sigma \vdash \alpha : \mathrm{type}}\text{typ:start}$$

$$\frac{\vdash \Sigma : \mathrm{sig}}{\Sigma \vdash \cdot : \mathrm{ctx}}\text{ctx:empty} \qquad \frac{\Sigma \vdash \Gamma : \mathrm{ctx} \quad \Sigma \vdash \alpha : \mathrm{type}}{\Sigma \vdash \Gamma, [X : \alpha] : \mathrm{ctx}}\text{ctx:var}$$

Example: A Well-Formed Signature

▷ Let $\Sigma := [\alpha : \mathrm{type}], [f : \alpha \to \alpha \to \alpha]$, then Σ is a well-formed signature, since we have derivations $\mathcal{A}$ and $\mathcal{B}$

$$\frac{\vdash \cdot : \mathrm{sig}}{\vdash [\alpha : \mathrm{type}] : \mathrm{sig}}\text{sig:type} \qquad \frac{\mathcal{A} \quad [\alpha : \mathrm{type}](\alpha) = \mathrm{type}}{[\alpha : \mathrm{type}] \vdash \alpha : \mathrm{type}}\text{typ:start}$$

and with these we can construct the derivation $\mathcal{C}$

$$\dfrac{\mathcal{A} \quad \dfrac{\mathcal{B} \quad \mathcal{B}}{[\alpha : \mathrm{type}] \vdash \alpha \to \alpha : \mathrm{type}}\text{typ:fn}}{\vdash \Sigma : \mathrm{sig}}\text{sig:const}$$

Example: A Well-Formed λ-Term

▷ using Σ from above, we can show that $\Gamma := [X : \alpha]$ is a well-formed context:

$$\dfrac{\dfrac{\mathcal{C}}{\Sigma \vdash \cdot : \mathrm{ctx}}\,\text{ctx:empty} \quad \dfrac{\mathcal{C} \quad \Sigma(\alpha) = \mathrm{type}}{\Sigma \vdash \alpha : \mathrm{type}}\,\text{typ:start}}{\Sigma \vdash \Gamma : \mathrm{ctx}}\,\text{ctx:var}$$

We call this derivation $\mathcal{G}$ and use it to show that

▷ $\lambda X_\alpha . fXX$ is well-typed and has type $\alpha \to \alpha$ in Σ. This is witnessed by the type derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathcal{C} \quad \Sigma(f) = \alpha \to \alpha \to \alpha}{\Gamma \vdash_\Sigma f : \alpha \to \alpha \to \alpha}\,\text{wff:const} \quad \dfrac{\mathcal{G}}{\Gamma \vdash_\Sigma X : \alpha}\,\text{wff:var}}{\Gamma \vdash_\Sigma fX : \alpha \to \alpha}\,\text{wff:app} \quad \dfrac{\mathcal{G}}{\Gamma \vdash_\Sigma X : \alpha}\,\text{wff:var}}{\Gamma \vdash_\Sigma fXX : \alpha}\,\text{wff:app}}{\cdot \vdash_\Sigma \lambda X_\alpha . fXX : \alpha \to \alpha}\,\text{wff:abs}$$

$\beta\,\eta$-Equality by Inference Rules: One-Step Reduction

▷ One-step Reduction ($+ \in \{\alpha, \beta, \eta\}$)

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} : \alpha \quad \Gamma \vdash_\Sigma \mathbf{B} : \beta}{\Gamma \vdash_\Sigma (\lambda X . \mathbf{A})\mathbf{B} \to^1_\beta [\mathbf{B}/X](\mathbf{A})}\,\text{wff}\beta\text{:top}$$

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} : \beta \to \alpha \quad X \notin \mathbf{dom}(\Gamma)}{\Gamma \vdash_\Sigma \lambda X . \mathbf{A}X \to^1_\eta \mathbf{A}}\,\text{wff}\eta\text{:top}$$

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} \to^1_+ \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{AC} : \alpha}{\Gamma \vdash_\Sigma \mathbf{AC} \to^1_+ \mathbf{BC}}\,\text{tr:app}fn$$

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} \to^1_+ \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{CA} : \alpha}{\Gamma \vdash_\Sigma \mathbf{CA} \to^1_+ \mathbf{CB}}\,\text{tr:app}arg$$

$$\dfrac{\Gamma, [X : \alpha] \vdash_\Sigma \mathbf{A} \to^1_+ \mathbf{B}}{\Gamma \vdash_\Sigma \lambda X . \mathbf{A} \to^1_+ \lambda X . \mathbf{B}}\,\text{tr:abs}$$

$\beta\,\eta$-Equality by Inference Rules: Multi-Step Reduction

▷ Multi-Step-Reduction ($+ \in \{\alpha, \beta, \eta\}$)

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \to^1_+ \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{A} \to^*_+ \mathbf{B}}\text{ms:start} \qquad \frac{\Gamma \vdash_\Sigma \mathbf{A} : \alpha}{\Gamma \vdash_\Sigma \mathbf{A} \to^*_+ \mathbf{A}}\text{ms:ref}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \to^*_+ \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{B} \to^*_+ \mathbf{C}}{\Gamma \vdash_\Sigma \mathbf{A} \to^*_+ \mathbf{C}}\text{ms:trans}$$

▷ Congruence Relation

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \to^*_+ \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{B}}\text{eq:start}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{B} =_+ \mathbf{A}}\text{eq:sym} \qquad \frac{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{B} =_+ \mathbf{C}}{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{C}}\text{eq:trans}$$

Chapter 17

Higher-Order Unification

We now come to a very important (if somewhat non-trivial and under-appreciated) algorithm: higher-order unification, i.e. unification in the simply typed λ-calculus, i.e. unification modulo $\alpha\beta\eta$ equality.

17.1 Higher-Order Unifiers

Before we can start solving the problem of higher-order unification, we have to become clear about the terms we want to use. It turns out that "most general $\alpha\beta\eta$ unifiers may not exist – as Theorem 17.1.5 shows, there may be infinitely descending chains of unifiers that become more an more general. Thus we will have to generalize our concepts a bit here.

HOU: Complete Sets of Unifiers

▷ Question: Are there most general higher-order Unifiers?

▷ Answer: What does that mean anyway?

▷ **Definition 17.1.1** $\sigma =_{\beta\eta} \rho[W]$, iff $\sigma(X) =_{\alpha\beta\eta} \rho(X)$ for all $X \in W$. $\sigma =_{\beta\eta} \rho[\mathcal{E}]$ iff $\sigma =_{\beta\eta} \rho[\text{free}(\mathcal{E})]$

▷ **Definition 17.1.2** σ is more general than θ on W ($\sigma \leq_{\beta\eta} \theta[W]$), iff there is a substitution ρ with $\theta =_{\beta\eta} \rho \circ \sigma[W]$.

▷ **Definition 17.1.3** $\Psi \subseteq \mathbf{U}(\mathcal{E})$ is a complete set of unifiers, iff for all unifiers $\theta \in \mathbf{U}(\mathcal{E})$ there is a $\sigma \in \Psi$, such that $\sigma \leq_{\beta\eta} \theta[\mathcal{E}]$.

▷ **Definition 17.1.4** If $\Psi \subseteq \mathbf{U}(\mathcal{E})$ is complete, then $\leq_{\beta\eta}$-minimal elements $\sigma \in \Psi$ are most general unifiers of $\mathcal{E}$.

▷ **Theorem 17.1.5** *The set* $\{[\lambda uv.hu/F]\} \cup \{\sigma_i \mid i \in \mathbb{N}\}$ *where*

$$\sigma_i := [\lambda uv.g_n u(u(h_1^n uv))\ldots(u(h_n^n uv))/F], [\lambda v.z/X]$$

is a complete set of unifiers for the equation $FXa_\iota =^? FXb_\iota$, *where* F *and* X *are variables of types* $(\iota \to \iota) \to \iota \to \iota$ *and* $\iota \to \iota$

Furthermore, σ_{i+1} *is more general than* σ_i.

▷ Proof Sketch: [Hue76, Theorem 5] □

The definition of a solved form in $\Lambda^{\to}$ is just as always; even the argument that solved forms are most general unifiers works as always, we only need to take $\alpha\beta\eta$ equality into account at every level.

Unification

▷ **Definition 17.1.6** $X^1 =^? \mathbf{B}^1 \wedge \ldots \wedge X^n =^? \mathbf{B}^n$ is in solved form, if the X^i are distinct free variables $X^i \notin \text{free}(\mathbf{B}^j)$ and $\mathbf{B}^j$ does not contain Skolem constants for all j.

▷ **Lemma 17.1.7** *If $\mathcal{E} = X^1 =^? \mathbf{B}^1 \wedge \ldots \wedge X^n =^? \mathbf{B}^n$ is in solved form, then $\sigma_{\mathcal{E}} := [\mathbf{B}^1/X^1], \ldots, [\mathbf{B}^n/X^n]$ is the unique most general unifier of $\mathcal{E}$*

▷ Proof:

P.1 $\sigma(X^i) =_{\alpha\beta\eta} \sigma(\mathbf{B}^i)$, so $\sigma \in \mathbf{U}(\mathcal{E})$

P.2 Let $\theta \in \mathbf{U}(\mathcal{E})$, then $\theta(X^i) =_{\alpha\beta\eta} \theta(\mathbf{B}^i) = \theta \circ \sigma(X^i)$

P.3 so $\theta \leq_{\beta\eta} \theta \circ \sigma[\mathcal{E}]$. □

17.2 Higher-Order Unification Transformations

We are now in a position to introduce the higher-order unifiation transformations. We proceed just like we did for first-order unification by casting the unification algorithm as a set of unification inference rules, leaving the control to a second layer of development.

We first look at a group of transformations that are (relatively) well-behaved and group them under the concept of "simplification", since (like the first-order transformation rules they resemble) have good properties. These are usually implemented in a group and applied eagerly.

Simplification $\mathcal{SIM}$

▷ **Definition 17.2.1** The higher-order simplification transformations $\mathcal{SIM}$

consist of the rules below.

$$\frac{(\lambda X_\alpha.\mathbf{A}) =^? (\lambda Y_\alpha.\mathbf{B}) \wedge \mathcal{E} \quad s \in \Sigma^{Sk}_\alpha \text{new}}{[s/X](\mathbf{A}) =^? [s/Y](\mathbf{B}) \wedge \mathcal{E}}\,\mathcal{SIM}:\alpha$$

$$\frac{(\lambda X_\alpha.\mathbf{A}) =^? \mathbf{B} \wedge \mathcal{E} \quad s \in \Sigma^{Sk}_\alpha \text{new}}{[s/X](\mathbf{A}) =^? \mathbf{B}s \wedge \mathcal{E}}\,\mathcal{SIM}:\eta$$

$$\frac{h\overline{\mathbf{U}^n} =^? h\overline{\mathbf{V}^n} \wedge \mathcal{E} \quad h \in (\Sigma \cup \Sigma^{Sk})}{\mathbf{U}^1 =^? \mathbf{V}^1 \wedge \ldots \wedge \mathbf{U}^n =^? \mathbf{V}^n \wedge \mathcal{E}}\,\mathcal{SIM}:\text{dec}$$

$$\frac{\mathcal{E} \wedge X =^? \mathbf{A} \quad X \notin \text{free}(\mathbf{A}) \quad \mathbf{A} \cap \Sigma^{Sk} = \emptyset \quad X \in \text{free}(\mathcal{E})}{[\mathbf{A}/X](\mathcal{E}) \wedge X =^? \mathbf{A}}\,\mathcal{SIM}:\text{elim}$$

After rule applications all λ-terms are reduced to head normal form.

The main new feature of these rules (with respect to their first-order counterparts) is the handling of λ-binders. We eliminate them by replacing the bound variables by Skolem constants in the bodies: The $\mathcal{SIM}:\alpha$ standardizes them to a single one using α-equality, and $\mathcal{SIM}:\eta$ first η-expands the right-hand side (which must be of functional type) so that $\mathcal{SIM}:\alpha$ applies. Given that we are setting bound variables free in this process, we need to be careful that we do not use them in the $\mathcal{SIM}$:elim rule, as these would be variable-capturing.

Consider for instance the higher-order unification problem $(\lambda X.X) =^? (\lambda Y.W)$, which is unsolvable (the left hand side is the identity function and the right hand side some constant function – whose value is given by W). So after an application of $\mathcal{SIM}:\alpha$, we have $c =^? W$, which looks like it could be a solved pair, but the elimination rule prevents that by insisting that instances may not contain Skolem Variables.

Conceptually, $\mathcal{SIM}$ is a direct generalization of first-order unification transformations, and shares it properties; even the proofs go correspondingly.

Properties of Simplification

▷ **Lemma 17.2.2 (Properties of $\mathcal{SIM}$)** *$\mathcal{SIM}$ generalizes first-order unification.*

- ▷ *$\mathcal{SIM}$ is terminating and confluent up to α-conversion*
- ▷ *Unique $\mathcal{SIM}$ normal forms exist (all pairs have the form $h\overline{\mathbf{U}^n} =^? k\overline{\mathbf{V}^m}$)*

▷ **Lemma 17.2.3** $\mathbf{U}(\mathcal{E} \wedge \mathcal{E}_\sigma) = \mathbf{U}(\sigma(\mathcal{E}) \wedge \mathcal{E}_\sigma)$.

Proof: by the definitions

▷ **P.1** If $\theta \in \mathbf{U}(\mathcal{E} \wedge \mathcal{E}_\sigma)$, then $\theta \in (\mathbf{U}(\mathcal{E}) \cap \mathbf{U}(\mathcal{E}_\sigma))$.

So $\theta =_{\beta\eta} \theta \circ \sigma[\mathbf{supp}(\sigma)]$,

and thus $(\theta \circ \sigma) \in \mathbf{U}(\mathcal{E})$, iff $\theta \in \mathbf{U}(\sigma(\mathcal{E}))$. □

P.2 **P.3** **Theorem 17.2.4** *If $\mathcal{E} \vdash_{\mathcal{SIM}} \mathcal{F}$, then $\mathbf{U}(\mathcal{E}) \leq_{\beta\eta} \mathbf{U}(\mathcal{F})[\mathcal{E}]$.* *(correct, complete)*

Proof: By an induction over the length of the derivation

P.1 We the $\mathcal{SIM}$ rules individually for the base case

P.1.1 $\mathcal{SIM}:\alpha$: by α-conversion □

P.1.2 $\mathcal{SIM}:\eta$: By η-conversion in the presence of $\mathcal{SIM}:\alpha$ □

P.1.3 $\mathcal{SIM}:\mathrm{dec}$: The head $h \in (\Sigma \cup \Sigma^{Sk})$ cannot be instantiated. □

P.1.4 $\mathcal{SIM}:\mathrm{elim}$: By Lemma 17.2.3. □

P.2 The step case goes directly by inductive hypothesis and transitivity of derivation. □

Now that we have simplifiation out of the way, we have to deal with unification pairs of the form $h\overline{\mathbf{U}^n} =^? k\overline{\mathbf{V}^m}$. Note that the case where both h and k are contstants is unsolvable, so we can assume that one of them is a variable. The unification problem $F_{\alpha\to\alpha}a =^? a$ is a particularly simple example; it has solutions $[\lambda X_\alpha.a/F]$ and $[\lambda X_\alpha.X/F]$. In the first, the solution comes by instantiating F with a λ-term of type $\alpha \to \alpha$ with head a, and in the second with a 1-projection term of type $\alpha \to \alpha$, which projects the head of the argument into the right position. In both cases, the solution came from a term with a given type and an appropriate head. We will look at the problem of finding such terms in more detail now.

General Bindings

▷ Problem: Find all formulae of given type α and head h.

▷ sufficient: long $\beta\eta$ head normal form, most general

▷ General Bindings: $\mathbf{G}^h_\alpha(\Sigma) := (\lambda\overline{X^k_\alpha}.h(H^1\overline{X})\ldots(H^n\overline{X}))$

- ▷ where $\alpha = \overline{\alpha_k} \to \beta$, $h : \overline{\gamma_n} \to \beta$ and $\beta \in \mathcal{B}\,\mathcal{T}$
- ▷ and $H^i : \overline{\alpha_k} \to \gamma_i$ new variables.

▷ **Observation 17.2.5** *General bindings are unique up to choice of names for H^i.*

▷ **Definition 17.2.6** If the head h is j^{th} bound variable in $\mathbf{G}^h_\alpha(\Sigma)$, call $\mathbf{G}^h_\alpha(\Sigma)$ j-projection binding (and write $\mathbf{G}^j_\alpha(\Sigma)$) else imitation binding

▷ clearly $\mathbf{G}^h_\alpha(\Sigma) \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ and $\mathsf{head}(\mathbf{G}^h_\alpha(\Sigma)) = h$

For the construction of general bindings, note that their construction is completely driven by the intended type α and the (type of) the head h. Let us consider some examples.

Example 17.2.7 The following general bindings may be helpful: $\mathbf{G}^{a_\iota}_{\iota\to\iota}(\Sigma) = \lambda X_\iota.a$, $\mathbf{G}^{a_\iota}_{\iota\to\iota\to\iota}(\Sigma) = \lambda X_\iota Y_\iota.a$, and $\mathbf{G}^{a_{\iota\to\iota}}_{\iota\to\iota\to\iota}(\Sigma) = \lambda X_\iota Y_\iota.a(HXY)$, where H is of type $\iota \to \iota \to \iota$

We will now show that the general bindings defined in Definition 17.2.6 are indeed the most general λ-terms given their type and head.

Approximation Theorem

▷ **Theorem 17.2.8** *If* $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\tau)$ *with* $\mathrm{head}(\mathbf{A}) = h$, *then there is a general binding* $\mathbf{G} = \mathbf{G}^h_\alpha(\Sigma)$ *and a substitution* ρ *with* $\rho(\mathbf{G}) =_{\alpha\beta\eta} \mathbf{A}$ *and* $\mathrm{dp}(\rho) < \mathrm{dp}(\mathbf{A})$.

▷ Proof: We analyze the term structure of $\mathbf{A}$

P.1 If $\alpha = \overline{\alpha_k} \to \beta$ and $h : \overline{\gamma_n} \to \beta$ where $\beta \in \mathcal{B}\,\mathcal{T}$, then the long head normal form of $\mathbf{A}$ must be $\lambda\overline{X^k_\alpha}.h\overline{\mathbf{U}^n}$.

P.2 $\mathbf{G} = \mathbf{G}^h_\alpha(\Sigma) = \lambda\overline{X^k_\alpha}.h(H^1\overline{X})\ldots(H^n\overline{X})$ for some variables $H^i : \overline{\alpha_k} \to \gamma_i$.

P.3 Choose $\rho := [\lambda\overline{X^k_\alpha}.\mathbf{U}^1/H^1], \ldots, [\lambda\overline{X^k_\alpha}.\mathbf{U}^n/H^n]$.

P.4 Then we have
$$\begin{aligned}\rho(\mathbf{G}) &= \lambda\overline{X^k_\alpha}.h(\lambda\overline{X^k_\alpha}.\mathbf{U}^1\overline{X})\ldots(\lambda\overline{X^k_\alpha}.\mathbf{U}^n\overline{X})\\ &=_{\beta\eta} \lambda\overline{X^k_\alpha}.h\overline{\mathbf{U}^n}\\ &=_{\beta\eta} \mathbf{A}\end{aligned}$$

P.5 The depth condition can be read off as $\mathrm{dp}(\lambda\overline{X^k_\alpha}.\mathbf{U}^1) \leq \mathrm{dp}(\mathbf{A}) - 1$. □

With this result we can state the higher-order unification transformations.

Higher-Order Unification ($\mathcal{HOU}$)

▷ Recap: After simplification, we have to deal with pairs where one (flex/rigid) or both heads (flex/flex) are variables

▷ **Definition 17.2.9** Let $\mathbf{G} = \mathbf{G}^h_\alpha(\Sigma)$ (imitation) or $\mathbf{G} \in \{\mathbf{G}^j_\alpha(\Sigma) \mid 1 \leq j \leq n\}$, then $\mathcal{HOU}$ consists of the transformations (always reduce to $\mathcal{SIM}$ normal form)

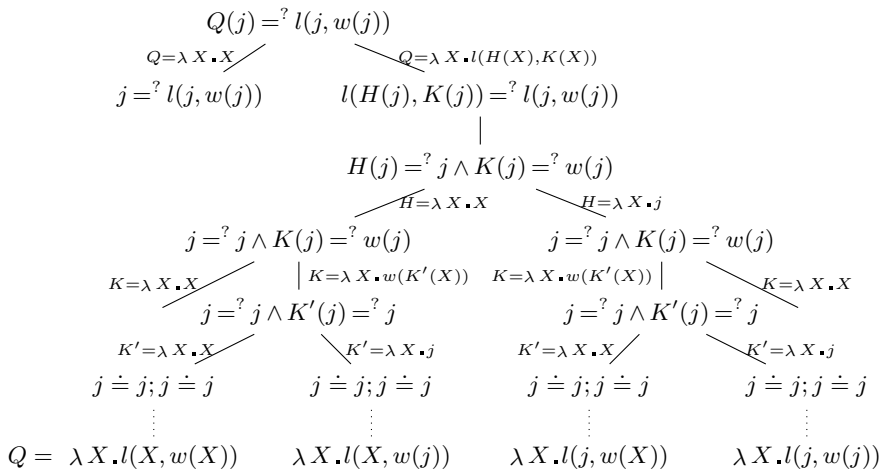
▷ Rule for flex/rigid pairs:
$$\frac{F_\alpha\overline{\mathbf{U}} =^? h\overline{\mathbf{V}} \wedge \mathcal{E}}{F =^? \mathbf{G} \wedge F\overline{\mathbf{U}} =^? h\overline{\mathbf{V}} \wedge \mathcal{E}}\,\mathcal{HOU}\text{:fr}$$

▷ Rules for flex/flex pairs:
$$\frac{F_\alpha\overline{\mathbf{U}} =^? H\overline{\mathbf{V}} \wedge \mathcal{E}}{F =^? \mathbf{G} \wedge F\overline{\mathbf{U}} =^? H\overline{\mathbf{V}} \wedge \mathcal{E}}\,\mathcal{HOU}\text{:ff}$$

Let us now fortify our intuition with a simple example.

$\mathcal{HOU}$ Example

Example 17.2.10 Let $Q, w : \iota \to \iota$, $l : \iota \to \iota \to \iota$, and $j : \iota$, then we have the following derivation tree in $\mathcal{HOU}$.



The first thing that meets the eye is that higher-order unification is branching. Indeed, for flex/-rigid pairs, we have to systematically explore the possibilities of binding the head variable the imitation binding and all projection bindings. On the initial node, we have two bindings, the projection binding leads to an unsolvable unification problem, whereas the imitation binding leads to a unification problem that can be decomposed into two flex/rigid pairs. For the first one of them, we have a projection and an imitation binding, which we systematically explore recursively. Eventually, we arrive at four solutions of the initial problem.

The following encoding of natural number arithmetics into $\Lambda^{\to}$ is useful for testing our unification algorithm

A Test Generator for Higher-Order Unification

▷ **Definition 17.2.11 (Church Numerals)** We define closed λ-terms of type $\nu := (\alpha \to \alpha) \to \alpha \to \alpha$

▷ Numbers: Church numerals: (n-fold iteration of arg1 starting from arg2)

$$n := (\lambda S_{\alpha\to\alpha}.\lambda O_\alpha.\underbrace{S(S\ldots S}_{n}(O)\ldots))$$

▷ Addition (N-fold iteration of S from N)

$$+ := (\lambda N_\nu M_\nu.\lambda S_{\alpha\to\alpha}.\lambda O_\alpha.NS(MSO))$$

▷ Multiplication: (N-fold iteration of MS ($=+m$) from O)

$$\cdot := (\lambda N_\nu M_\nu.\lambda S_{\alpha\to\alpha}.\lambda O_\alpha.N(MS)O)$$

▷ **Observation 17.2.12** *Subtraction and (integer) division on Church numerals can be automted via higher-order unification.*

▷ **Example 17.2.13** $5-2$ by solving the unification problem $2+x_\nu =^? 5$

Equation solving for Church numerals yields a very nice generator for test cases for higher-order unification, as we know which solutions to expect.

17.3 Properties of Higher-Order Unification

We will now establish the properties of the higher-order unification problem and the algorithms we have introduced above. We first establish the unidecidability, since it will influence how we go about the rest of the properties.

We establish that higher-order unification is undecidable. The proof idea is a typical for undecidable proofs: we reduce the higher-order unification problem to one that is known to be undecidable: here, the solution of Diophantine equations $\mathbb{N}$.

▷ Undecidability of Higher-Order Unification

▷ **Theorem 17.3.1** *Second-order unification is undecidable* *(Goldfarb '82 [Gol81])*

▷ Proof Sketch: Reduction to Hilbert's tenth problem (solving Diophantine equations) (known to be undecidable)

□

▷ **Definition 17.3.2** We call an equation a Diophantine equation, if it is of the form

- ▷ $x_i\ x_j = x_k$
- ▷ $x_i + x_j = x_k$
- ▷ $x_i = c_j$ where $c_j \in \mathbb{N}$

where the variables x_i range over $\mathbb{N}$.

▷ These can be solved by higher-order unification on Church numerals. (cf. Observation 17.2.12)

.

▷ **Theorem 17.3.3** *The general solution for sets of Diophantine equations is undecidable.* *(Matijasevič 1970 [Mat70])*

The argument undecidability proofs is always the same: If higher-order unification were decidable, then via the encoding we could use it to solve Diophantine equations, which we know we cannot by Matijasevič's Theorem.

The next step will be to analyze our transformations for higher-order unification for correctness and completeness, just like we did for first-order unification.

$\mathcal{HOU}$ is Correct

▷ **Lemma 17.3.4** *If* $\mathcal{E} \vdash_{\mathcal{HOU}_{\mathrm{fr}}} \mathcal{E}'$ *or* $\mathcal{E} \vdash_{\mathcal{HOU}_{\mathrm{ff}}} \mathcal{E}'$, *then* $\mathbf{U}(\mathcal{E}') \subseteq \mathbf{U}(\mathcal{E})$.

▷ Proof Sketch: $\mathcal{HOU}$:fr and $\mathcal{HOU}$:ff only add new pair. □

▷ **Corollary 17.3.5** *$\mathcal{HOU}$ is correct: If $\mathcal{E} \vdash_{\mathcal{HOU}} \mathcal{E}'$, then $\mathbf{U}(\mathcal{E}') \subseteq \mathbf{U}(\mathcal{E})$.*

Given that higher-order unification is not unitary and undecidable, we cannot just employ the notion of completeness that helped us in the analysis of first-order unification. So the first thing is to establish the condition we want to establish to see that $\mathcal{HOU}$ gives a higher-order unification algorithm.

Completeness of $\mathcal{HOU}$

▷ We cannot expect completeness in the same sense as for first-order unification: "If $\mathcal{E} \vdash_{\mathcal{U}} \mathcal{F}$, then $\mathbf{U}(\mathcal{E}) \subseteq \mathbf{U}(\mathcal{F})$" (see ?founifcalc-complete?) as the rules fix a binding and thus partially commit to a unifier (which excludes others).

▷ We cannot expect termination either, since HOU is undecidable.

▷ For a semi-decision procedure we only need termination on unifiable problems.

▷ **Theorem 17.3.6 ($\mathcal{HOU}$ derives Complete Set of Unifiers)** *If $\theta \in \mathbf{U}(\mathcal{E})$, then there is a $\mathcal{HOU}$-derivation $\mathcal{E} \vdash_{\mathcal{HOU}} \mathcal{F}$, such that $\mathcal{F}$ is in solved form, $\sigma_{\mathcal{F}} \in \mathbf{U}(\mathcal{E})$, and $\sigma_{\mathcal{F}}$ is more general than θ.*

▷ Proof Sketch: Given a unifier θ of $\mathcal{E}$, we guide the derivation with a measure μ_θ towards $\mathcal{F}$. □

So we will embark on the details of the completeness proof. The first step is to define a measure that will guide the $\mathcal{HOU}$ transformation out of a unification problem $\mathcal{E}$ given a unifier θ of cE.

Completeness of $\mathcal{HOU}$ (Measure)

▷ **Definition 17.3.7** We call $\mu(\mathcal{E}, \theta) := \langle \mu_1(\mathcal{E}, \theta), \mu_2(\theta) \rangle$ the unification measure for $\mathcal{E}$ and θ, if

- ▷ $\mu_1(\mathcal{E}, \theta)$ is the multiset of term depths of $\theta(X)$ for the unsolved $X \in \mathbf{supp}(\theta)$.
- ▷ $\mu_2(\mathcal{E})$ the multiset of term depths in $\mathcal{E}$.
- ▷ $\prec$ is the strict lexicographic order on pairs: ($\langle a, b \rangle \prec \langle c, d \rangle$, if $a < c$ or $a = c$ and $b < d$)
- ▷ Component orderings are multiset orderings: ($M \cup \{m\} < M \cup N$ iff $n < m$ for all $n \in N$)

▷ **Lemma 17.3.8** *$\prec$ is well-founded.* *(by construction)*

This measure will now guide the $\mathcal{HOU}$ transformation in the sense that in any step it chooses whether to use $\mathcal{HOU}$:fr or $\mathcal{HOU}$:ff, and which general binding (by looking at what θ would do). We formulate the details in Theorem 17.3.9 and look at their consequences before we proove it.

Completeness of $\mathcal{HOU}$ (μ-Prescription)

▷ **Theorem 17.3.9** *If $\mathcal{E}$ is unsolved and $\theta \in \mathbf{U}(\mathcal{E})$, then there is a unification problem $\mathcal{E}'$ with $\mathcal{E} \vdash_{\mathcal{HOU}} \mathcal{E}'$ and a substitution $\theta' \in \mathbf{U}(\mathcal{E}')$, such that*

- $\theta =_{\beta\eta} \theta'[\mathcal{E}]$
- $\mu(\mathcal{E}', \theta') \prec \mu(\mathcal{E}, \theta)$.

we call such a $\mathcal{HOU}$-step a μ-prescribed

▷ **Corollary 17.3.10** *If $\mathcal{E}$ is unifiable without μ-prescribed $\mathcal{HOU}$-steps, then $\mathcal{E}$ is solved.*

▷ In other words: μ guides the $\mathcal{HOU}$-transformations to a solved form

We now come to the proof of Theorem 17.3.9, which is a relatively simple consequence of Theorem 17.2.8.

Proof of Theorem 17.3.9

▷ Proof:

P.1 Let $\mathbf{A} =^? \mathbf{B}$ be an unsolved pair of the form $\mathbf{F}\overline{\mathbf{U}} =^? \mathbf{G}\overline{\mathbf{V}}$ in $\mathcal{F}$.

P.2 $\mathcal{E}$ is a $\mathcal{SIM}$ normal form, so $\mathbf{F}$ and $\mathbf{G}$ must be constants or variables,

P.3 but not the same constant, since otherwise $\mathcal{SIM}$:dec would be applicable.

P.4 We can also exclude $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$, as $\mathcal{SIM}$:triv would be be appliccable.

P.5 If $\mathbf{F} = \mathbf{G}$ is a variable not in $\mathbf{supp}(\theta)$, then $\mathcal{SIM}$:dec appliccable. By correctness we have $\theta \in \mathbf{U}(\mathcal{E}')$ and $\mu(\mathcal{E}', \theta) \prec \mu(\mathcal{E}, \theta)$, as $\mu_1(\mathcal{E}', \theta) \preceq \mu_1(\mathcal{E}, \theta)$ and $\mu_2(\mathcal{E}') \prec \mu_2(\mathcal{E})$.

P.6 Otherwise we either have $\mathbf{F} \neq \mathbf{G}$ or $\mathbf{F} = \mathbf{G} \in \mathbf{supp}(\theta)$.

P.7 In both cases $\mathbf{F}$ or $\mathbf{G}$ is an unsolved variable $F \in \mathbf{supp}(\theta)$ of type α, since $\mathcal{E}$ is unsolved.

P.8 Without loss of generality we choose $F = \mathbf{F}$.

P.9 By Theorem 17.2.8 there is a general binding $\mathbf{G} = \mathbf{G}^f_\alpha(\Sigma)$ and a substitution ρ with $\rho(\mathbf{G}) =_{\alpha\beta\eta} \theta(F)$. So,

- if $\mathrm{head}(\mathbf{G}) \notin \mathbf{supp}(\theta)$, then $\mathcal{HOU}$:fr is appliccable,
- if $\mathrm{head}(\mathbf{G}) \in \mathbf{supp}(\theta)$, then $\mathcal{HOU}$:ff is appliccable.

P.10 Choose $\theta' := \theta \cup \rho$. Then $\theta =_{\beta\eta} \theta'[\mathcal{E}]$ and $\theta' \in \mathbf{U}(\mathcal{E}')$ by correctness.

P.11 $\mathcal{HOU}$:ff and $\mathcal{HOU}$:fr solve $F \in \mathbf{supp}(\theta)$ and replace F by $\mathbf{supp}(\rho)$ in the set of unsolved variable of $\mathcal{E}$.

P.12 so $\mu_1(\mathcal{E}', \theta') \prec \mu_1(\mathcal{E}, \theta)$ and thus $\mu(\mathcal{E}', \theta') \prec \mu(\mathcal{E}, \theta)$. □

We now convince ourselves that if $\mathcal{HOU}$ terminates with a unification problem, then it is either solved – in which case we can read off the solution – or unsolvable.

Terminal $\mathcal{HOU}$-problems are Solved or Unsolvable

▷ **Theorem 17.3.11** *If $\mathcal{E}$ is a unsolved UP and $\theta \in \mathbf{U}(\mathcal{E})$, then there is a $\mathcal{HOU}$-derivation $\mathcal{E} \vdash_{\mathcal{HOU}} \sigma_\sigma$, with $\sigma \leq_{\beta\eta} \theta[\mathcal{E}]$.*

▷ Proof: Let $\mathcal{D}\colon \mathcal{E} \vdash_{\mathcal{HOU}} \mathcal{F}$ a maximal μ-prescribed $\mathcal{HOU}$-derivation from $\mathcal{E}$.

P.1 This must be finite, since $\prec$ is well-founded (ind. over length n of $\mathcal{D}$)

P.2 If $n = 0$, then $\mathcal{E}$ is solved and $\sigma_{\mathcal{E}}$ most general unifier

P.3 thus $\sigma_{\mathcal{E}} \leq_{\beta\eta} \theta[\mathcal{E}]$

P.4 If $n > 0$, then there is a μ-prescribed step $\mathcal{E} \vdash_{\mathcal{HOU}} \mathcal{E}'$ and a substitution θ' as in Theorem 17.3.9.

P.5 by IH there is a $\mathcal{HOU}$-derivation $\mathcal{E}' \vdash_{\mathcal{HOU}} \mathcal{F}$ with $\sigma_{\mathcal{F}} \leq_{\beta\eta} \theta'[\mathcal{E}']$.

P.6 by correctness $\sigma_{\mathcal{F}} \in \mathbf{U}(\mathcal{E}') \subseteq \mathbf{U}(\mathcal{E})$.

P.7 rules of $\mathcal{HOU}$ only expand free variables, so $\sigma_{\mathcal{F}} \leq_{\beta\eta} \theta'[\mathcal{E}']$

P.8 Thus $\sigma_{\mathcal{F}} \leq_{\beta\eta} \theta'[\mathcal{E}]$,

P.9 This completes the proof, since $\theta' =_{\beta\eta} \theta[\mathcal{E}]$ by Theorem 17.3.9. □

We now recap the properties of higher-order unification (HOU) to gain an overview.

Properties of HO-Unification

▷ HOU is undecidable, HOU need not have most general unifiers

▷ The $\mathcal{HOU}$ transformation induce an algorithm that enumerates a complete set of higher-order unifiers.

▷ $\mathcal{HOU}$:ff gives enormous degree of indeterminism

▷ HOU is intractable in practice consider restricted fragments where it is!

▷ HO Matching (decidable up to order four), HO Patterns (unitary, linear), ...

17.4 Pre-Unification

We will now come to a variant of higher-order unification that is used in higher-order theorem proving, where we are only interested in the exgistence of a unifier – e.g. in mating-style tableaux. In these cases, we can do better than full higher-order unification.

Pre-Unification

▷ $\mathcal{HOU}$:ff has a giant branching factor in the search space for unifiers. (makes HOU impracticable)

▷ In most situations, we are more interested in solvability of unification problems

than in the unifiers themselves.

▷ More liberal treatment of flex/flex pairs.

▷ **Observation 17.4.1** *flex/flex-pairs $F\overline{\mathbf{U}^n} =^? G\overline{\mathbf{V}^m}$ are always (trivially) solvable by $[\lambda\overline{X^n}.H/F], [\lambda\overline{Y^m}.H/G]$, where H is a new variable*

▷ Idea: consider flex/flex-pairs as pre-solved.

▷ **Definition 17.4.2 (Pre-Unification)** For given terms $\mathbf{A}, \mathbf{B} \in wff_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$ find a substitution σ, such that $\sigma(\mathbf{A}) =^p_{\beta\,\eta} \sigma(\mathbf{B})$, where $=^p_{\beta\,\eta}$ is the equality theory that is induced by $\beta\eta$ and $F\overline{\mathbf{U}} = G\overline{\mathbf{V}}$.

▷ **Lemma 17.4.3** *A higher-order unification problem is unifiable, iff it is pre-unifiable.*

The higher-order pre-unification algorithm can be obtained from $\mathcal{HOU}$ by simply omitting the offending $\mathcal{HOU}$:ff rule.

Pre-Unification Algorithm $\mathcal{HOPU}$

▷ **Definition 17.4.4** A unification problem is a pre-solved form, iff all of its pairs are solved or flex/flex

▷ **Lemma 17.4.5** *If $\mathcal{E}$ is solved and $\mathcal{P}$ flex/flex, then σ_σ is a most general unifier of a pre-solved form $\mathcal{E} \wedge \mathcal{P}$.*

▷ Restrict all $\mathcal{HOU}$ rule so that they cannot be applied to pre-solved pairs.

▷ In particular, remove $\mathcal{HOU}$:ff!

▷ $\mathcal{HOPU}$ only consists of $\mathcal{SIM}$ and $\mathcal{HOU}$:fr.

▷ **Theorem 17.4.6** *$\mathcal{HOPU}$ is a correct and complete pre-unification algorithm*

▷ Proof Sketch: with exactly the same methods as higher-order unification □

▷ **Theorem 17.4.7** *Higher-order pre-unification is infinitary, i.e. a unification problem can have infinitely many unifiers.* (Huet 76' [Hue76])

▷ **Example 17.4.8** $Y(\lambda X_\iota.X)a =^? a$, where a is a constant of type ι and Y a variable of type $(\iota \to \iota) \to \iota \to \iota$ has the most general unifiers $\lambda sz.s^n z$ and $\lambda sz.s^n a$, which are mutually incomparable and thus most general.

17.5 Applications of Higher-Order Unification

Application of HOL in NL Semantics: Ellipsis

▷ **Example 17.5.1** *John loves his wife. George does too*

- ▹ $\text{love}(\text{john}, \text{wife\_of}(\text{john})) \wedge Q(\text{george})$
- ▹ "*George* has property some Q, which we still have to determine"

Idea: If *John* has property Q, then it is that he *loves his wife*.

⋙ Equation: $Q(\text{john}) =_{\alpha\beta\eta} \text{love}(\text{john}, \text{wife\_of}(\text{john}))$

▷ Solutions (computed by HOU):

- ▹ $Q = \lambda z.\text{love}(z, \text{wife\_of}(z))$ and $Q = \lambda z.\text{love}(z, \text{wife\_of}(\text{john}))$
- * $Q = \lambda z.\text{love}(\text{john}, \text{wife\_of}(z))$ and $Q = \lambda z.\text{love}(\text{john}, \text{wife\_of}(\text{john}))$

▷ Readings: *George loves his own wife.* and *George loves Johns wife.*

Chapter 18

Simple Type Theory

In this Chapter we will revisit the higher-order predicate logic introduced in Chapter 12 with the base given by the simply typed λ-calculus. It turns out that we can define a higher-order logic by just introducing a type of propositions in the λ-calculus and extending the signatures by logical constants (connectives and quantifiers).

Higher-Order Logic Revisited

- ▷ Idea: introduce special base type o for truth values
- ▷ **Definition 18.0.1** We call a Σ-algebra $\langle \mathcal{D}, \mathcal{I} \rangle$ a Henkin model, iff $\mathcal{D}_o = \{\mathsf{T}, \mathsf{F}\}$.
- ▷ $\mathbf{A}_o$ valid under φ, iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$
- ▷ connectives in Σ: $\neg \in \Sigma_{o\to o}$ and $\{\vee, \wedge, \Rightarrow, \Leftrightarrow, \ldots\} \subseteq \Sigma_{o\to o\to o}$ (with the intuitive $\mathcal{I}$-values)
- ▷ quantifiers: $\Pi^\alpha \in \Sigma_{(\alpha\to o)\to o}$ with $\mathcal{I}(\Pi^\alpha)(p) = \mathsf{T}$, iff $p(a) = \mathsf{T}$ for all $a \in \mathcal{D}_\alpha$.
- ▷ quantified formula e: $\forall X_\alpha \,.\, \mathbf{A}$ stands for $\Pi^\alpha(\lambda X_\alpha \,.\, \mathbf{A})$
- ▷ $\mathcal{I}_\varphi(\forall X_\alpha \,.\, \mathbf{A}) = \mathcal{I}(\Pi^\alpha)(\mathcal{I}_\varphi(\lambda X_\alpha \,.\, \mathbf{A})) = \mathsf{T}$, iff $\mathcal{I}_{\varphi,[a/X]}(\mathbf{A}) = \mathsf{T}$ for all $a \in \mathcal{D}_\alpha$
- ▷ looks like PLΩ (Call any such system $\mathrm{HOL}^{\to}$)

There is a more elegant way to treat quantifiers in $\mathrm{HOL}^{\to}$. It builds on the realization that the λ-abstraction is the only variable binding operator we need, quantifiers are then modeled as second-order logical constants. Note that we do not have to change the syntax of $\mathrm{HOL}^{\to}$ to introduce quantifiers; only the "lexicon", i.e. the set of logical constants. Since Π^α and Σ^α are logical constants, we need to fix their semantics.

Higher-Order Abstract Syntax

- ▷ Idea: In $\mathrm{HOL}^{\to}$, we already have variable binder: λ, use that to treat quantification.
- ▷ **Definition 18.0.2** We assume logical constants Π^α and Σ^α of type $(\alpha \to o) \to$

o.

Regain quantifiers as abbreviations:

$$(\forall X_\alpha.\mathbf{A}) := \overset{\alpha}{\Pi}(\lambda X_\alpha.\mathbf{A}) \qquad (\exists X_\alpha.\mathbf{A}) := \overset{\alpha}{\Sigma}(\lambda X_\alpha.\mathbf{A})$$

▷ **Definition 18.0.3** We must fix the semantics of logical constants:

1. $\mathcal{I}(\Pi^\alpha)(p) = \mathsf{T}$, iff $p(a) = \mathsf{T}$ for all $\mathsf{a} \in \mathcal{D}_\alpha$ (i.e. if p is the universal set)
2. $\mathcal{I}(\Sigma^\alpha)(p) = \mathsf{T}$, iff $p(a) = \mathsf{T}$ for some $\mathsf{a} \in \mathcal{D}_\alpha$ (i.e. iff p is non-empty)

▷ With this, we re-obtain the semantics we have given for quantifiers above:

$$\mathcal{I}_\varphi(\forall X_\iota.\mathbf{A}) = \mathcal{I}_\varphi(\overset{\iota}{\Pi}(\lambda X_\iota.\mathbf{A})) = \mathcal{I}(\overset{\iota}{\Pi})(\mathcal{I}_\varphi(\lambda X_\iota.\mathbf{A})) = \mathsf{T}$$

iff $\mathcal{I}_\varphi(\lambda X_\iota.\mathbf{A})(a) = \mathcal{I}_{[a/X],\varphi}(\mathbf{A}) = \mathsf{T}$ for all $a \in \mathcal{D}_\alpha$

But there is another alternative of introducing higher-order logic due to Peter Andrews. Instead of using connectives and quantifiers as primitives and defining equality from them via the Leibniz indiscernability principle, we use equality as a primitive logical constant and define everything else from it.

Alternative: HOL$^=$

▷ only one logical constant $q^\alpha \in \Sigma_{\alpha\to\alpha\to o}$ with $\mathcal{I}(q^\alpha)(a,b) = \mathsf{T}$, iff $a = b$.

▷ Definitions (D) and Notations (N)

N	$\mathbf{A}_\alpha = \mathbf{B}_\alpha$	for	$q^\alpha\mathbf{A}_\alpha\mathbf{B}_\alpha$
D	T	for	$q^o = q^o$
D	F	for	$(\lambda X_o.T) = (\lambda X_o.X_o)$
D	Π^α	for	$q^{(\alpha\to o)}(\lambda X_\alpha.T)$
N	$\forall X_\alpha.\mathbf{A}$	for	$\Pi^\alpha(\lambda X_\alpha.\mathbf{A})$
D	$\wedge$	for	$\lambda X_o.\lambda Y_o.(\lambda G_{o\to o\to o}.GTT) = (\lambda G_{o\to o\to o}.GXY)$
N	$\mathbf{A}\wedge\mathbf{B}$	for	$\wedge\mathbf{A}_o\mathbf{B}_o$
D	$\Rightarrow$	for	$\lambda X_o.\lambda Y_o.X = X\wedge Y$
N	$\mathbf{A}\Rightarrow\mathbf{B}$	for	$\Rightarrow\mathbf{A}_o\mathbf{B}_o$
D	$\neg$	for	q^oF
D	$\vee$	for	$\lambda X_o.\lambda Y_o.\neg(\neg X\wedge\neg Y)$
N	$\mathbf{A}\vee\mathbf{B}$	for	$\vee\mathbf{A}_o\mathbf{B}_o$
D	$\exists X_\alpha.\mathbf{A}_o$	for	$\neg(\forall X_\alpha.\neg\mathbf{A})$
N	$\mathbf{A}_\alpha \neq \mathbf{B}_\alpha$	for	$\neg(q^\alpha\mathbf{A}_\alpha\mathbf{B}_\alpha)$

▷ yield the intuitive meanings for connectives and quantifiers.

In a way, this development of higher-order logic is more foundational, especially in the context of Henkin semantics. There, Theorem 12.0.6 does not hold (see [And72] for details). Indeed the proof of Theorem 12.0.6 needs the existence of "singleton sets", which can be shown to be equivalent to the existence of the identity relation. In other words, Leibniz equality only denotes the equality

relation, if we have an equality relation in the models. However, the only way of enforcing this (remember that Henkin models only guarantee functions that can be explicitly written down as λ-terms) is to add a logical constant for equality to the signature.

We will conclude this section with a discussion on two additional "logical constants" (constants with a fixed meaning) that are needed to make any progress in mathematics. Just like above, adding them to the logic guarantees the existence of certain functions in Henkin models. The most important one is the description operator that allows us to make definite descriptions like "the largest prime number" or "the solution to the differential equation $f' = f$.

More Axioms for $HOL^{\to}$

▷ **Definition 18.0.4** unary conditional $\mathbf{w} \in \Sigma_{o\to\alpha\to\alpha}$
$\mathbf{w}\mathbf{A}_o\mathbf{B}_\alpha$ means: "If $\mathbf{A}$, then $\mathbf{B}$"

▷ **Definition 18.0.5** binary conditional $\mathbf{if} \in \Sigma_{o\to\alpha\to\alpha\to\alpha}$
$\mathbf{if}\mathbf{A}_o\mathbf{B}_\alpha\mathbf{C}_\alpha$ means: "if $\mathbf{A}$, then $\mathbf{B}$ else $\mathbf{C}$".

▷ **Definition 18.0.6** description operator $\iota \in \Sigma_{(\alpha\to o)\to\alpha}$
if $\mathbf{P}$ is a singleton set, then $\iota\mathbf{P}_{\alpha\to o}$ is the element in $\mathbf{P}$,

▷ **Definition 18.0.7** choice operator $\gamma \in \Sigma_{(\alpha\to o)\to\alpha}$
if $\mathbf{P}$ is non-empty, then $\gamma\mathbf{P}_{\alpha\to o}$ is an arbitrary element from $\mathbf{P}$

▷ **Definition 18.0.8 (Axioms for these Operators)**

- ▷ unary conditional: $\forall\varphi_o.\forall X_\alpha.\varphi \Rightarrow \mathbf{w}\varphi X = X$
- ▷ conditional: $\forall\varphi_o.\forall X_\alpha, Y_\alpha, Z_\alpha.(\varphi \Rightarrow \mathbf{if}\varphi XY = X) \wedge (\neg\varphi \Rightarrow \mathbf{if}\varphi ZX = X)$
- ▷ description $\forall P_{\alpha\to o}.(\exists^1 X_\alpha.PX) \Rightarrow (\forall Y_\alpha.PY \Rightarrow \iota P = Y)$
- ▷ choice $\forall P_{\alpha\to o}.(\exists X_\alpha.PX) \Rightarrow (\forall Y_\alpha.PY \Rightarrow \gamma P = Y)$

Idea: These operators ensure a much larger supply of functions in Henkin models.

▷ More on the Description Operator

▷ ι is a weak form of the choice operator (only works on singleton sets)

▷ Alternative Axiom of Descriptions: $\forall X_\alpha.\iota^\alpha(=X) = X$.

- ▷ use that $\mathcal{I}_{[\mathsf{a}/X]}(=X) = \{\mathsf{a}\}$
- ▷ we only need this for base types $\neq o$
- ▷ Define $\iota^o := =(\lambda X_o.X)$ or $\iota^o := (\lambda G_{o\to o}.GT)$ or $\iota^o := =(=T)$
- ▷ $\iota^{\alpha\to\beta} := (\lambda H_{(\alpha\to\beta)\to o}X_\alpha.\iota^\beta(\lambda Z_\beta.(\exists F_{\alpha\to\beta}.(HF) \wedge (FX) = Z)))$

Chapter 19

Higher-Order Tableaux

In this Chapter we will extend the ideas from first-order tableaux to higher-order logic.

The rules fo standard tableaux are just like the ones for first-order logic, only that we can take advantage oft higher-order abstract syntax for the quantifiers

Tableau-Rules ($\mathcal{T}_\omega^s$)

▷ **Definition 19.0.1** The rules of $\mathcal{T}_\omega^s$ consist of the propositional tableau rules of $\mathcal{T}_0$ together with

$$\frac{\overset{\alpha}{\Pi}\mathbf{A}^{\mathsf{T}}}{\mathbf{A}\mathbf{C}^{\mathsf{T}}}\,\mathcal{T}_\omega^s{:}\forall \qquad \frac{\overset{\alpha}{\Pi}\mathbf{A}^{\mathsf{F}} \quad c\in(\Sigma_0^{sk}\backslash\mathcal{H})}{\mathbf{A}c^{\mathsf{F}}}\,\mathcal{T}_\omega^s{:}\exists$$

Higher-order, free-variable tableaux work exactly like first-order tableaux, except that the cut rule uses higher-order unification.

Higher-Order Free-Variable Tableaus ($\mathcal{T}_\omega$ first try)

▷ **Definition 19.0.2** The $\mathcal{T}_\omega$ calculus consists of the propositional tableau rules plus

$$\frac{\overset{\alpha}{\Pi}\mathbf{A}^{\mathsf{T}}}{\mathbf{A}X_\alpha{}^{\mathsf{T}}}\,\mathcal{T}_\omega{:}\forall \qquad \frac{\overset{\alpha}{\Pi}\mathbf{A}^{\mathsf{F}} \quad \mathrm{free}(\mathbf{A})=\{Y^1_{\alpha_1},\ldots,Y^n_{\alpha_n}\} \quad f\in\Sigma^{Sk}_{\overline{\alpha_n}\to\alpha}\ \text{new}}{\mathbf{A}(f\overline{Y^n})^{\mathsf{F}}}\,\mathcal{T}_\omega{:}\exists$$

▷ Problem: Unification in $\Lambda^{\to}$ is undecidable, so we need more

▷ Idea: explicit rule that residuates the unification problem

$$\frac{\mathbf{A}^{\alpha} \quad \mathbf{B}^{\beta}}{\mathbf{A} \neq^{?} \mathbf{B}}\,\mathcal{T}_{\omega}\text{:cut}$$

and adapt the $\mathcal{HOU}$ rules to tableaux (DNF instead of CNF)

Note that we cannot directly use the higher-order unification algorithm, since that is undecidable – this would not result in a fair proof search procedure. Therefore we reinterpret $\mathcal{HOPU}$ rules as tableau rules and mix them into the proof search procedure.

For the reinterpretation of $\mathcal{HOU}$ rules into tableau rules we change notation of the unification pairs, using $\mathbf{A} \neq^{?} \mathbf{B}$ instead of $\mathbf{A} =^{?} \mathbf{B}$, since in the tableau setting we want to refute that $\mathbf{A}$ and $\mathbf{B}$ cannot be made equal instead of finding conditions that make them equal (as we did for unification). Correspondingly, we we do not use a "conjunction" of equations, but a disjunction (using tableau branches) of "disequations". But up to this "double negation" the unification algorithm stays the same.

$\mathcal{T}_{\omega}$ (Pre-Unification)

▷ we can use $\mathcal{SIM}:\alpha$, $\mathcal{SIM}:\eta$, and $\mathcal{SIM}:\mathrm{triv}$ directly, for $\mathcal{SIM}:\mathrm{dec}$ and $\mathcal{SIM}:\mathrm{elim}$ we integrate into tableau setting more closely, obtaining

$$\frac{h\overline{\mathbf{U}^n} \neq^{?} h\overline{\mathbf{V}^n} \quad h \in (\Sigma \cup \Sigma^{Sk} \cup \mathcal{V}_{\mathcal{T}})}{\mathbf{U}^1 \neq^{?} \mathbf{V}^1 \;\big|\; \ldots \;\big|\; \mathbf{U}^n \neq^{?} \mathbf{V}^n}\,\mathcal{T}_{\omega}\text{:dec}$$

$$\frac{F_{\alpha}\overline{\mathbf{U}} \neq^{?} h\overline{\mathbf{V}}}{F \neq^{?} \mathbf{G} \;\big|\; F\overline{\mathbf{U}} \neq^{?} h\overline{\mathbf{V}}}\,\mathcal{T}_{\omega}\text{:fr}$$

$$\frac{X \neq^{?} \mathbf{A} \quad X \notin \mathrm{free}(\mathbf{A}) \quad \mathbf{A} \cap \Sigma^{Sk} = \emptyset}{\bot}\,\mathcal{T}_{\omega}\text{:elim}$$

where $\mathbf{G} = \mathbf{G}^{h}_{\alpha}(\Sigma)$ (imitation) or $\mathbf{G} \in \{\mathbf{G}^{j}_{\alpha}(\Sigma) \,|\, 1 \leq j \leq n\}$

▷ **Definition 19.0.3** We call a $\mathcal{T}_{\omega}$ tableau closed, if all branches end in a $\bot$ or a flex/flex pair.

Note that the elimination rule is particularly elegant in the tableau setting – it comes in the form of a closure rule: If we have a solved pair, then we can just make the branch unsatisfiable by applying its most general unifier to the whole tableau.

Note furthermore, that with the mixed propositional and pre-unification calculus in $\mathcal{T}_{\omega}$, the decision whether to do regular or matings-style tableaux boils down to a a decision of the strategy used to expand the $\mathcal{T}_{\omega}$ tableaux.

We will now fortify our intuition with an extended example: a $\mathcal{T}_\omega$ proof of (a version of) Cantor's theorem. The particular formulation we use below uses the whole universe of type ι for the set S and universe of type $\iota \to \iota$ for the power set.

$\mathcal{T}_\omega$ Example: Cantor's Theorem

▷ **Theorem 19.0.4** *There is no surjective function from the natural numbers into the sequences of natural numbers.*

▷ Formally: $\neg(\exists F_{\iota\to\iota\to\iota}.\forall G_{\iota\to\iota}.\exists J_\iota.FJ = G)$

▷ For the proof we use

▷ $\forall X_\iota.\neg X = sX$ (the successor function has no fixed points)

▷ an extensionality axiom

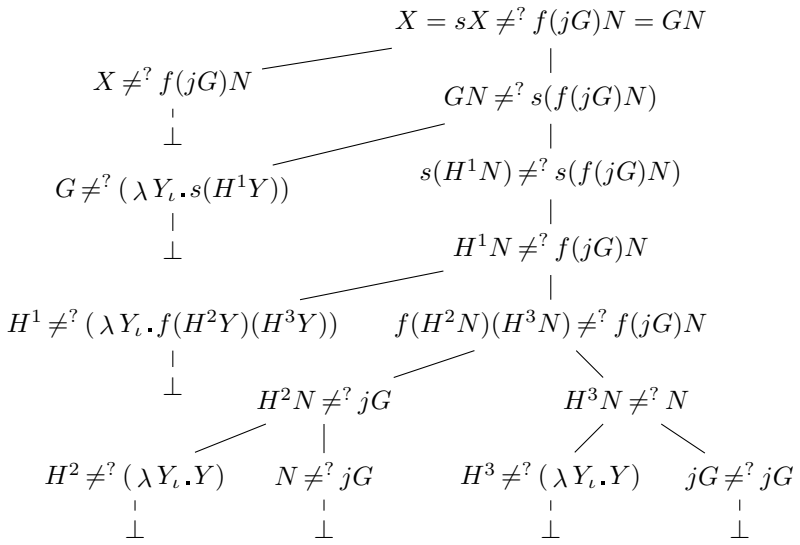
We initialize the tableau with the three formulae discussed above, and then employ the $\mathcal{T}_\omega$ rules.

$\mathcal{T}_\omega$-Proof (Cantor's Theorem)

▷ First the propositional part (analyzing formula structure)

$$\begin{array}{c}
\neg(\exists F_{\iota\to\iota\to\iota}.\forall G_{\iota\to\iota}.\exists J_\iota.FJ = G)^{\mathsf{F}} \\
\exists F_{\iota\to\iota\to\iota}.\forall G_{\iota\to\iota}.\exists J_\iota.FJ = G^{\mathsf{T}} \\
\forall G_{\iota\to\iota}.\exists J_\iota.f_{\iota\to\iota\to\iota}J = G^{\mathsf{T}} \\
\exists J_\iota.fJ = G^{\mathsf{T}} \\
f(jG) = G^{\mathsf{T}} \\
H = K \Rightarrow (\forall N_\iota.HN = KN)^{\mathsf{T}} \\
\begin{array}{c|c|c}
H = K^{\mathsf{F}} & & HN = KN^{\mathsf{T}} \\
H = K \neq^? f(jG) = G & & f(jG)N = GN^{\mathsf{T}} \\
H \neq^? f(jG) \mid K \neq^? G & & X = sX^{\mathsf{F}} \\
\bot \quad\quad \bot & & X = sX \neq^? f(jG)N = GN
\end{array}
\end{array}$$

▷ then we continue with unification tableau



▷ We found a closed tableau and completed the $\mathcal{T}_\omega$ proof.

In the higher-order unification tableau above we face the same problem we always face when we try to display the dynamics of free-variable tableaux: in the closure rules we have to instantiate the whole tableau. But this turns the tableau into a standard tableau. So we close the leftmost branch and apply the substitution to the branches to the right of the current branch only.

Note that at first sight $N \neq^? jG$ is not solved (and indeed unsolvable), since j is a Skolem constant. But we only need to forbid the Skolem constants that were introduced by the $\mathcal{SIM}{:}\alpha$ and $\mathcal{SIM}{:}\eta$ rules. So there is no problem here; since they were introduced by $\mathcal{T}_\omega{:}\exists$.

Even though we were successful in proving Cantor's theorem, $\mathcal{T}_\omega$ is not complete as we will see.

Problem for $\mathcal{T}_\omega$

▷ **Theorem 19.0.5** *There is a valid formula* $(\exists X_o . X)$

▷ This is clearly valid, (eg. $\mathbf{A} \vee \neg \mathbf{A}$)

▷ $\mathcal{T}_\omega$ attempt

$$\neg (\forall X_o . \neg X)^{\mathsf{F}}$$
$$\forall X_o . \neg X^{\mathsf{T}}$$
$$\neg X^{\mathsf{T}}$$
$$X^{\mathsf{F}}$$

▷ we are stuck!

▷ Observation: We have to instantiate X further, e.g. by $[\neg Q_o/X]$.

▷ then we can continue

$$X^{\mathsf{F}}$$
$$\neg Q^{\mathsf{F}}$$
$$Q^{\mathsf{T}}$$
$$X \neq^? Q$$

close with $[Q/X]$.

We see that unlike in first-order unification we cannot obtain all necessary instantiations by unification. Indeed in the presence of predicate variables – in our example above we can view X_o as a nullary predicate – we have to allow instantiations with (all) logical connectives and quantifiers. Fortunately, we can do this in a minimally committing fashion via general bindings, unfortunately, we have to systematically try out all possible ones – which is costly, since there are infinitely many quantifiers.

Primitive Substitutions

▷ Unification is not sufficient for $\mathcal{T}_\omega$

▷ We need a rule that instantiates head variables with terms that introduce logical constants.

▷ **Definition 19.0.6** We extend $\mathcal{T}_\omega$ with the rule $\mathcal{T}_\omega$:prim.

$$\frac{X_\alpha\overline{\mathbf{U}^n}^\alpha \quad \mathbf{G}\in\mathbf{G}^k_\alpha(\Sigma) \quad k\in(\{\wedge,\neg\}\cup\{\overset{\beta}{\Pi}\,|\,\beta\in\mathcal{T}\})}{X\neq^?\mathbf{G}\;\Big|\;X_\alpha\overline{\mathbf{U}^n}^\alpha}\,\mathcal{T}_\omega\text{:prim}$$

We call $[\mathbf{G}/X]$ a primitive substitution.

▷ In our example $\neg Q = \mathbf{G}^\neg_o(\Sigma)$.

There is another source of incompleteness as another example shows: we can have propositions embedded in formulae. Note that this is different from the situation in first-order logic, but quite natural in mathematics, e.g. for conditional statements of the form "if φ_o then $\mathbf{A}_\alpha$ else $\mathbf{B}_\alpha$.", where φ is a proposition embedded in a term of type α.

Another Example

▷ $\mathbf{A} = \neg(c_{o\to o}b_o)\vee(c^\neg\neg b)$ is valid

▷ $\mathcal{T}_\omega$ proof attempt

$$\begin{array}{c}\neg(cb)\vee(c^\neg\neg b)^{\mathsf{F}}\\ \neg(cb)^{\mathsf{F}}\\ c^\neg\neg b^{\mathsf{F}}\\ cb^{\mathsf{T}}\\ cb\neq^? c^\neg\neg b\\ b\neq^?\neg\neg b\end{array}$$

and we are stuck (again)

▷ Idea: theory unification with $X_o = \neg\neg X_o$

▷ But the problem is more general: If $\mathbf{A}\Leftrightarrow\mathbf{B}$ valid, then $\neg(c\mathbf{A})\wedge(c\mathbf{B})$ must be $\mathcal{T}_\omega$-refutable.

▷ Solution: call to the theorem prover recursively.

▷ **Definition 19.0.7** We extend $\mathcal{T}_\omega$ with the rule $\mathcal{T}_\omega$:rec,

$$\frac{\mathbf{A}_o \neq^? \mathbf{B}_o}{\begin{array}{c|c}\mathbf{A}^\mathsf{T} & \mathbf{A}^\mathsf{F}\\ \mathbf{B}^\mathsf{F} & \mathbf{B}^\mathsf{T}\end{array}}\mathcal{T}_\omega\text{:rec}$$

▷ **Observation 19.0.8** *We can prove $\mathbf{A}_o$ by unifying it with T_o.*

The $\mathcal{T}_\omega$:rec rule puts the propositional and unification rules of $\mathcal{T}_\omega$ at an equal footing. $\mathcal{T}_\omega$ can be seen as a calculus for theorem proving or as an unification algorithm that takes the theory of equivalence into account.

Each aspect of $\mathcal{T}_\omega$ can recurse into the the other; this is necessary, since in HOL$^\to$ the propositional level – which has a fixed interpretation and therefore special $\mathcal{T}_\omega$ rules – and the term level – which is freely interpreted and must thus be handled by unification – can recurse arbitrarily.

To make matters worse, we also have a soundness problem that comes from Skolemization: we can prove a version of the Axiom of Choice that is known to be independent of HOL$^\to$, and thus should not be provable.

Skolemization is not sound

▷ Axiom of Choice:
$\exists\gamma_{\alpha\to o\to\alpha}.\forall P_{\alpha\to o}.(\exists X_\alpha.PX)\Rightarrow(\forall Y_\alpha.(PY)\Rightarrow\gamma P = Y)$

▷ Weaker Version: (call it $\mathbf{C}$)

$$\forall R_{\alpha\to\alpha\to o}.(\forall X_\alpha.\exists Y_\alpha.RXY)\Rightarrow(\exists F_{(\alpha\to o)\to\alpha}.\forall Z_\alpha.RZ(FZ))$$

▷ Neither $\mathbf{C}$ nor $\neg\mathbf{C}$ are valid in HOL$^\to$ (independent)

▷ but $\mathbf{C}$ is provable $\mathcal{T}_\omega$. (see next slide)

In this proof, the Skolem constant f introduced for the assumption $\forall X_\alpha.\exists Y_\alpha.RXY$ becomes available as an instance for the variable F in (used to require the existence of a choice operator).

Skolemization is not sound (Choice Proof)

$$\forall R_{\alpha\to\alpha\to o}.(\forall X_\alpha.\exists Y_\alpha.RXY)\Rightarrow(\exists F_{(\alpha\to o)\to\alpha}.\forall Z_\alpha.RZ(FZ))^{\mathsf{F}}$$
$$\forall X_\alpha.\exists Y_\alpha.rXY^{\mathsf{T}}$$
$$\exists F_{(\alpha\to o)\to\alpha}.\forall Z_\alpha.rZ(FZ)^{\mathsf{F}}$$
$$\exists Y_\alpha.rXY^{\mathsf{T}}$$
$$rX(fX)^{\mathsf{T}}$$
$$\forall F_{(\alpha\to o)\to\alpha}.\neg(\forall Z_\alpha.rZ(fZ))^{\mathsf{T}}$$
$$\forall Z_\alpha.rZ(FZ)^{\mathsf{F}}$$
$$r(gF)(F(gF))^{\mathsf{F}}$$
$$rX(fX)\neq^?r(gF)(F(gF))$$

$$X\neq^?gF \quad \bot \qquad \Big| \qquad F\neq^?(\lambda Z_\alpha.f(H^1Z)) \quad \bot \qquad \Big| \qquad f(gF)\neq^?F(gF)$$
$$f(g(\lambda Z_\alpha.f(H^1Z)))\neq^?f(H^1(g(\lambda Z_\alpha.f(H^1Z))))$$
$$g(\lambda Z_\alpha.f(H^1Z))\neq^?H^1(g(\lambda Z_\alpha.f(H^1Z)))$$
$$H^1\neq^?(\lambda W_\alpha.W) \quad \bot \qquad \Big| \qquad g(\lambda Z_\alpha.fZ)\neq^?g(\lambda Z_\alpha.fZ) \quad \bot$$

In first-order logic, Skolemization is sound, since Skolem constants do not "lose their arguments", so they cannot be used to prove the axiom of choice.

The following part is still experimentals; not required for the course

Variable Conditions

▷ **Definition 19.0.9** Let Γ be an annotated variable context, Then a variable condition $\mathcal{R}$ is a relation on $\mathcal{R}\subseteq\mathbf{dom}(\Gamma)\times\mathbf{dom}(\Gamma^-)$.

▷ **Definition 19.0.10** We call a substitution σ with $\mathbf{supp}(\sigma)\subseteq\mathbf{dom}(\Gamma)\cup\mathbf{dom}(\Delta)$ a $\mathcal{R}$-substitution, iff $Y\notin\mathrm{free}(\sigma(X))$ for all $(x,y)\in\mathcal{R}$.

▷ Intuition: If $(X,Y^-)\in\mathcal{R}$, then no formula that contains Y^- freely may be substituted for X.

▷ We define a judgment $\Delta\vdash\overline{\mathcal{R}}(X,\mathbf{A})$ by

- ▷ $\Delta,\Gamma\vdash_\sigma\mathbf{A}:\Gamma(X)$ und $X\notin\mathrm{free}(\mathbf{A})$,
- ▷ $\{X\}\times\mathrm{free}(\mathbf{A})\cap\mathcal{R}=\emptyset$ (no variable $Y\in\mathrm{free}(\mathbf{A})$ is an $\mathcal{R}$-image of X)

▷ So σ is a $\mathcal{R}$-substitution, iff $\Delta\vdash_\Sigma\overline{\mathcal{R}}(X,\sigma(X))$ for all $X\in\mathbf{supp}(\sigma)$.

▷ Extension of variable conditions for instantiaton with $[\mathbf{A}/X]$:

$$\mathcal{R}(\mathbf{A}/X):=\{(Z,W)\in\mathcal{R}\,|\,Z\neq X\}\cup\{(Z,W)\,|\,Z\in\mathrm{free}(\mathbf{A}),\mathcal{R}(X,W)\}$$

Higher-Order Tableaux (final)

▷ Higher-order tableaux are triples $\langle\Gamma:\mathcal{R}\rangle.\mathcal{T}$

▷ propositional tableaux as always (do not change Γ or $\mathcal{R}$)

▷ New quantifier rules

$$\frac{\overset{\alpha}{\Pi}\mathbf{A}^{\mathsf{T}}}{\mathbf{A}X_\alpha{}^{\mathsf{T}}} \qquad \frac{\overset{\alpha}{\Pi}\mathbf{A}^{\mathsf{F}}}{\mathbf{A}Y^{-\mathsf{F}}}$$

where

- ▷ $X, Y^- \notin \mathbf{dom}(\Gamma)$
- ▷ $\Gamma' := \Gamma, [X : \alpha]$ and $\Gamma' := \Gamma, [Y^- : \alpha]$
- ▷ $\mathcal{R}' := \mathcal{R}$ and $\mathcal{R}' := \mathcal{R} \cup \mathrm{free}(\mathbf{A}) \times \{Y^-\}$.

▷ substitution rule: If a path in $\langle \Gamma, [X : \alpha] \colon \mathcal{R}\rangle.\mathcal{T}$ ends in an equation $X =^? \mathbf{A}$ with $\Gamma \vdash_\Sigma \overline{\mathcal{R}}(X, \mathbf{A})$, then generate $\langle \Gamma \colon \mathcal{R}(\mathbf{A}/X)\rangle.[\mathbf{A}/X]\mathcal{T}$.

▷ Primitive Substitution: If $\langle \Gamma, [X : \alpha] \colon \mathcal{R}\rangle.\mathcal{T}$ contains a formula $(X\overline{\mathbf{U}^n})^\alpha$, and $\mathbf{A} \in \mathbf{G}^k_\alpha(\Sigma, \Gamma, \mathcal{C})$ with $k \in (\{\wedge, \neg\} \cup \{\Pi^\beta \mid \beta \in \mathcal{T}\})$, then generate $\langle \Gamma \cup \mathcal{C} \colon \mathcal{R}(\mathbf{A}/X)\rangle.[\mathbf{A}/X]\mathcal{T}$

▷ Closed Tableau: every branch ends in a trivial equation $\mathbf{A} =^? \mathbf{A}$ or a pre-solved equation $F\overline{\mathbf{U}} \neq^? G\overline{\mathbf{V}}$.

▷ tableau-substitution closes the respective branch

▷ Side conditions

$$\frac{(\lambda X_\alpha.\mathbf{A}) \neq^? (\lambda Y_\alpha.\mathbf{B}) \quad Z \notin \mathbf{dom}(\Gamma)}{[Z/X](\mathbf{A}) \neq^? Z(Y)(\mathbf{B})}$$

$$\frac{(\lambda X_\alpha.\mathbf{A}) \neq^? \mathbf{B} \quad Z \notin \mathbf{dom}(\Gamma)}{[Z/X](\mathbf{A}) \neq^? \mathbf{B}Z}$$

where $\Gamma' = \Gamma, [Z^0 : \alpha]$ and $\mathcal{R}' := \mathcal{R}(Z/X)$

$$\frac{h\overline{\mathbf{U}^n} \neq^? h\overline{\mathbf{V}^n} \quad h \in (\Sigma \cup \mathbf{dom}(\Gamma^0) \cup \mathbf{dom}(\Gamma^-))}{\mathbf{U}^1 \neq^? \mathbf{V}^1 \;\Big|\; \ldots \;\Big|\; \mathbf{U}^n \neq^? \mathbf{V}^n}$$

$$\frac{F\overline{\mathbf{U}} =^? h\overline{\mathbf{V}} \quad \Gamma(F) = \alpha \quad \Gamma \vdash_\Sigma \overline{\mathcal{R}}(F, \mathbf{G})}{F \neq^? \mathbf{G} \;\Big|\; F\overline{\mathbf{U}} \neq^? h\overline{\mathbf{V}}}$$

Here we have

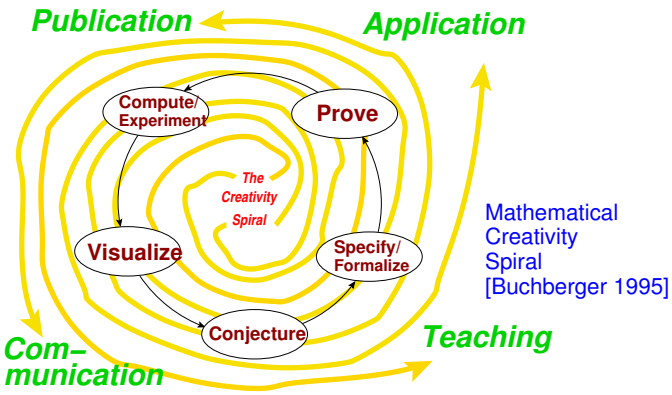
- ▷ $\mathbf{G} \in \mathbf{G}^h(\Sigma, \Delta, \mathcal{C})$
- ▷ $\Gamma' = \Gamma \cup \mathcal{C}$ and $\mathcal{R}' := \mathcal{R}$

Part V

Project Tetrapod

The way we do math will change dramatically

▷ **Definition 19.0.11 (Doing Math)** Buchberger's Math creativity spiral



▷ Every step will be supported by mathematical software systems

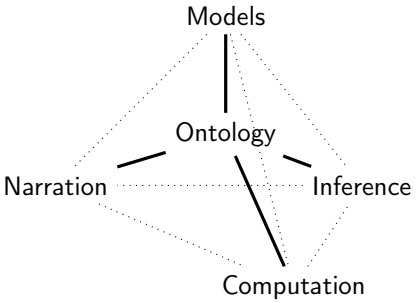
▷ Towards an infrastructure for web-based mathematics!

Knowledge Representation is only Part of "Doing Math"

▷ One of the key insights is that the mathematics ecosystem involves a body of knowledge described as an ontology and four aspects of it:

- ▷ inference: exploring theories, formulating conjectures, and constructing proofs
- ▷ computation: simplifying mathematical objects, re-contextualizing conjectures. . .
- ▷ models: collecting examples, applying mathematical knowledge to real-world problems and situations.
- ▷ narration: devising both informal and formal languages for expressing mathematical ideas, visualizing mathematical data, presenting mathematical developments, organizing and interconnecting mathematical knowledge

▷ We call the endeavour of creating a computer-supported mathematical ecosystem "Project tetrapod" as it needs to stand on four legs.



Collaborators: KWARC@FAU, McMaster University

Part VI

Summary and Review

Chapter 20

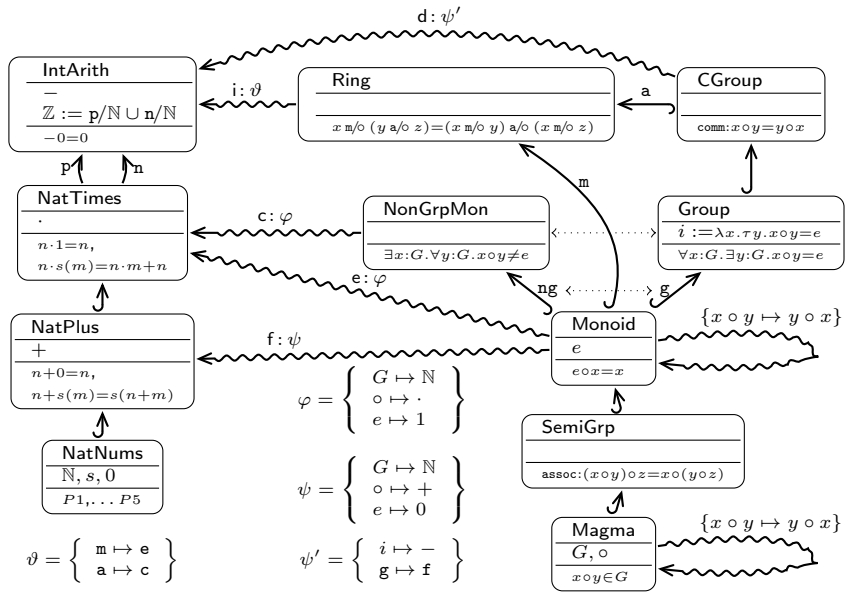
Modulare Repr"asentation mathematischen Wissens

▷ Modular Representation of Math (Theory Graph)

- ▷ Idea: Follow mathematical practice of generalizing and framing
  - ▷ framing: If we can view an object a as an instance of concept B, we can inherit all of B properties (almost for free.)
  - ▷ state all assertions about properties as general as possible (to maximize inheritance)
  - ▷ examples and applications are just special framings.
- ▷ Modern expositions of Mathematics follow this rule (radically e.g. in Bourbaki)
- ▷ formalized in the theory graph paradigm (little/tiny theory doctrine)
  - ▷ theories as collections of symbol declarations and axioms (model assumptions)
  - ▷ theory morphisms as mappings that translate axioms into theorems
- ▷ **Example 20.0.1 (MMT: Modular Mathematical Theories)** MMT is a foundation-indepent theory graph formalism with advanced theory morphisms.

  Problem: With a proliferation of abstract (tiny) theories readability and accessibility suffers (one reason why the Bourbaki books fell out of favor)

▷ Modular Representation of Math (MMT Example)



The MMT Module System

▷ Central notion: theory graph with theory nodes and theory morphisms as edges

▷ **Definition 20.0.2** In MMT, a theory is a sequence of constant declarations – optionally with type declarations and definitions

▷ MMT employs the Curry/Howard isomorphism and treats

- ▹ axioms/conjectures as typed symbol declarations (propositions-as-types)
- ▹ inference rules as function types (proof transformers)
- ▹ theorems as definitions (proof terms for conjectures)

▷ **Definition 20.0.3** MMT had two kinds of theory morphisms

- ▹ structures instantiate theories in a new context (also called: definitional link, import)
  they import of theory S into theory T induces theory morphism $S \to T$
- ▹ views translate between existing theories (also called: postulated link, theorem link)
  views transport theorems from source to target (framing)

▷ together, structures and views allow a very high degree of re-use

▷ **Definition 20.0.4** We call a statement t induced in a theory T, iff there is

- ▹ a path of theory morphisms from a theory S to T with (joint) assignment σ,
- ▹ such that $t = \sigma(s)$ for some statement s in S.

▷ In MMT, all induced statements have a canonical name, the MMT URI.

Applications for Theories in Physics

▷ Theory Morphisms allow to "view" source theory in terms of target theory.

▷ Theory Morphisms occur in Physics all the time.

Theory	Temp. in Kelvin	Temp. in Celsius	Temp. in Fahrenheit
Signature	°K	°C	°F
Axiom:	absolute zero at 0°K	Water freezes at 0°C	cold winter night: 0°F
Axiom:	$\delta(°\mathrm{K}1) = \delta(°\mathrm{C}1)$	Water boils at 100°C	domestic pig: 100°F
Theorem:	Water freezes at 271.3°K	domestic pig: 38°C	Water boils at 170°F
Theorem:	cold winter night: 240°K	absolute zero at -271.3°C	absolute zero at -460°F

Views: °C $\overset{+271.3°}{\longrightarrow}$ K, °C $\overset{-32/2°}{\longrightarrow}$ F, and °F $\overset{+240/2°}{\longrightarrow}$ K, inverses.

▷ Other Examples: Coordinate Transformations,

▷ Application: Unit Conversion: apply view morphism (flatten) and simplify with UOM. (For new units, just add theories and views.)

▷ Application: MathWebSearch on flattened theory (Explain view path)

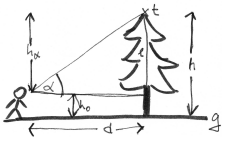
Chapter 21

Application: Serious Games

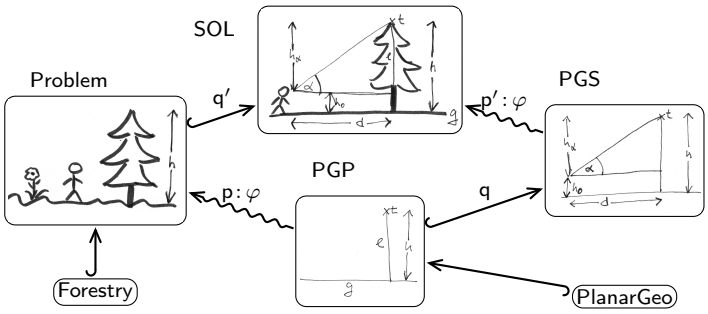
Framing for Problem Solving (The FrameIT Method)

▷ **Example 21.0.1 (Problem 0.8.15)**

How can you measure the height of a tree you cannot climb, when you only have a protactor and a tape measure at hand.

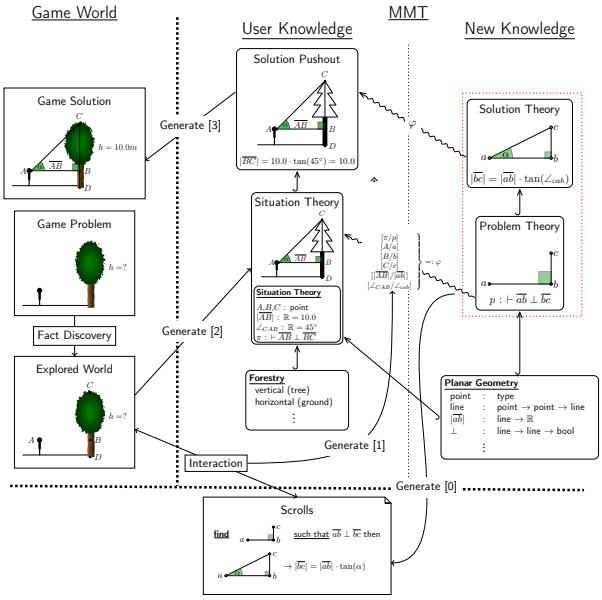


▷ Framing: view the problem as one that is already understood (using theory morphisms)



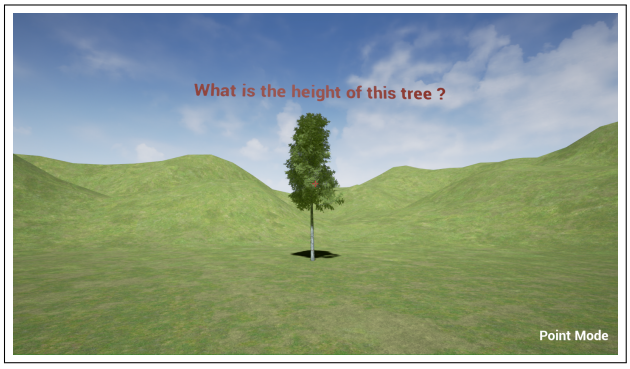
▷ squiggly (framing) morphisms guaranteed by metatheory of theories!

Example Learning Object Graph



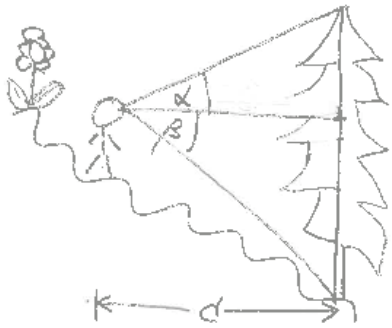
FrameIT: Problem Solving by Framing ⇝ Serious Games

▷ Implementation in custom web interface (Bachelor's thesis)



Currently: Minecraft mod (problem/solution pairs as objects that can be applied for automation)

▷ Combining Problem/Solution Pairs



▷ We can use the same mechanism for combining P/S pairs

▷ create more complex P/S pairs (e.g. for trees on slopes)

Another whole set of applications and game behaviors can come from the fact that LOGraphs give ways to combine problem/solution pairs to novel ones. Consider for instance the diagram on the right, where we can measure the height of a tree of a slope. It can be constructed by combining the theory SOL with a copy of SOL along a second morphism the inverts h to $-h$ (for the lower triangle with angle β) and identifies the base lines (the two occurrences of h_0 cancel out). Mastering the combination of problem/solution pairs further enhances the problem solving repertoire of the player.

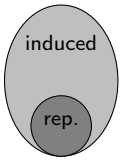
Chapter 22

Search in the Mathematical Knowledge Space

The Mathematical Knowledge Space

▷ **Observation 22.0.1** *The value of framing is that it induces new knowledge*

▷ **Definition 22.0.2** The mathematical knowledge space MKS is the structured space of represented and induced knowledge, mathematically literate have access to.



▷ Idea: make math systems mathematically literate by supporting the MKS

▷ In this talk: I will cover three aspects

- ▷ an approach for representing framing and the MKS (OMDoc/MMT)
- ▷ search modulo framing (MKS-literate search)
- ▷ a system for archiving the MKS (MathHub.info)

▷ Told from the Perspective of: searching the MKS

♭search: Indexing flattened Theory Graphs

▷ Simple Idea: We have all the necessary components: MMT and `MathWebSearch`

▷ **Definition 22.0.3** The ♭search systen is an integration of `MathWebSearch` and MMT that

- ▷ computes the induced formulae of a modular mathematical library via MMT (aka. flattening)
- ▷ indexes induced formulae by their MMT URIs in `MathWebSearch`
- ▷ uses `MathWebSearch` for unification-based querying (hits are MMT URIs)

▷ uses the MMT to present MMT URI (compute the actual formula)

▷ generates explanations from the MMT URI of hits.

▷ Implemented by Mihnea Iancu in ca. 10 days (MMT harvester pre-existed)

▷ almost all work was spent on improvements of MMT flattening

▷ `MathWebSearch` just worked (web service helpful)

♭search User Interface: Explaining MMT URIs

▷ Recall: ♭search (`MathWebSearch` really) returns a MMT URI as a hit.

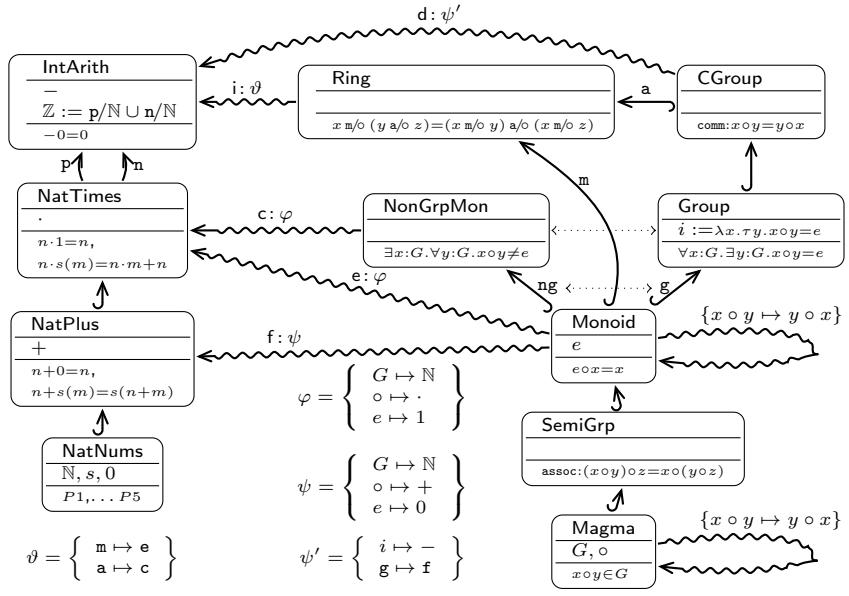
▷ Question: How to present that to the user? (for his/her greatest benefit)

▷ Fortunately: MMT system can compute induced statements (the hits)

▷ Problem: Hit statement may look considerably different from the induced statement

▷ Solution: Template-based generation of NL explanations from MMT URIs.
MMT knows the necessary information from the components of the MMT URI.

Modular Representation of Math (MMT Example)



Example: Explaining a MMT URI

▷ **Example 22.0.4** ♭ search search result u?IntArith?c/g/assoc for query $(\boxed{x}+\boxed{y})+\boxed{z}=\boxed{R}$.

▹ localize the result in the theory u?IntArithf with

Induced statement $\forall x,y,z:\mathbb{Z}.(x+y)+z=x+(y+z)$ found in `http://cds.omdoc.org/cds/elal?IntArith` (subst, justification).

▹ Justification: from MMT info about morphism c (source, target, assignment)

IntArith is a CGroup if we interpret $\circ$ as $+$ and G as $\mathbb{Z}$.

▹ skip over g, since its assignment is trivial and generate

CGroups are SemiGrps by construction

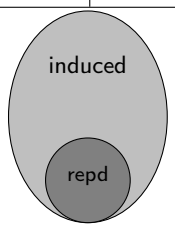
▹ ground the explanation by

In SemiGrps we have the axiom assoc : $\forall x,y,z:G.(x\circ y)\circ z=x\circ(y\circ z)$

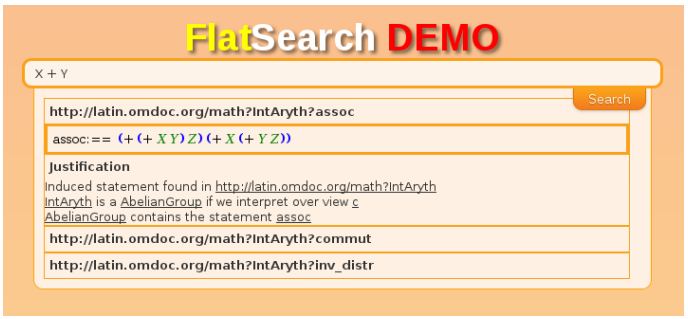
♭ search on the LATIN Logic Atlas

▷ Flattening the LATIN Atlas (once):

type	modular	flat	factor
declarations	2310	58847	25.4
library size	23.9 MB	1.8 GB	14.8
math sub-library	2.3 MB	79 MB	34.3
`MathWebSearch` harvests	25.2 MB	539.0 MB	21.3



▷ simple ♭ search frontend at `http://cds.omdoc.org:8181/search.html`



Overview: KWARC Research and Projects

Applications: eMath 3.0, Active Documents, Semantic Spreadsheets, Semantic CAD/CAM, Change Mangagement, Global Digital Math Library, Math Search Systems, SMGloM: Semantic Multilingual Math Glossary, Serious Games, ...		
Foundations of Math:	**KM & Interaction**:	**Semantization**:
▷ MathML, $OpenMath$	▷ Semantic Interpretation (aka. Framing)	▷ LaTeXML: LaTeX → XML
▷ advanced Type Theories	▷ math-literate interaction	▷ sTeX: Semantic LaTeX
▷ MMT: Meta Meta Theory	▷ MathHub: math archives & active docs	▷ invasive editors
▷ Logic Morphisms/Atlas	▷ Semantic Alliance: embedded semantic services	▷ Context-Aware IDEs
▷ Theorem Prover/CAS Interoperability		▷ Mathematical Corpora
		▷ Linguistics of Math
Foundations: Computational Logic, Web Technologies, $OMDoc$/MMT		

Take-Home Message

▷ Overall Goal: *Overcoming the "One-Brain-Barrier" in Mathematics* (by knowledge-based systems)

▷ Means: Mathematical Literacy by Knowledge Representation and Processing in theory graphs in Theoriegraphen. (Framing as mathematical practice)

Bibliography

[AGC+06] Andrea Asperti, Ferruccio Guidi, Claudio Sacerdoti Coen, Enrico Tassi, and Stefano Zacchiroli. A content based mathematical search engine: Whelp. In Jean-Christophe Filliâtre, Christine Paulin-Mohring, and Benjamin Werner, editors, *Types for Proofs and Programs, International Workshop, TYPES 2004, revised selected papers*, number 3839 in LNCS, pages 17–32. Springer Verlag, 2006.

[And72] Peter B. Andrews. General models and extensionality. *Journal of Symbolic Logic*, 37(2):395–397, 1972.

[And02] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Kluwer Academic Publishers, second edition, 2002.

[Bou68] Nicolas Bourbaki. *Theory of Sets*. Elements of Mathematics. Springer Verlag, 1968.

[Bou74] Nicolas Bourbaki. *Algebra I*. Elements of Mathematics. Springer Verlag, 1974.

[Bou89] N. Bourbaki. *General Topology 1-4*. Elements of Mathematics. Springer Verlag, 1989.

[Can95] Georg Cantor. Beiträge zur begründung der transfiniten mengenlehre (1). *Mathematische Annalen*, 46:481–512, 1895.

[Can97] Georg Cantor. Beiträge zur begründung der transfiniten mengenlehre (2). *Mathematische Annalen*, 49:207–246, 1897.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[DPPDD09] A.K. Doxiadēs, C.H. Papadimitriou, A. Papadatos, and A. Di Donna. *Logicomix: An Epic Search for Truth*. Bloomsbury, 2009.

[Fre79] Gottlob Frege. Begriffsschrift: eine der arithmetischen nachgebildete formelsprache des reinen denkens, 1879.

[Gen34] Gerhard Gentzen. Untersuchungen über das logische Schließen I. *Mathematische Zeitschrift*, 39(2):176–210, 1934.

[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].

[Gol81] Warren D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.

[Hil26] David Hilbert. Über das unendliche. *Mathematische Annalen*, 95:161–190, 1926.

[Hue76] Gérard P. Huet. *Résolution d'Équations dans des Langages d'ordre 1,2,...,w*. Thèse d'état, Université de Paris VII, 1976.

[Hue80] Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM (JACM)*, 27(4):797–821, 1980.

[Jin10] Arif Jinha. Article 50 million: an estimate of the number of scholarly articles in existence. *Learned Publishing*, 23(3):258–263, 2010.

[KK06] Andrea Kohlhase and Michael Kohlhase. Communities of Practice in MKM: An Extensional Model. In Jon Borwein and William M. Farmer, editors, *Mathematical Knowledge Management (MKM)*, number 4108 in LNAI, pages 179–193. Springer Verlag, 2006.

[Koh08] Michael Kohlhase. Using LaTeX as a semantic markup format. *Mathematics in Computer Science*, 2(2):279–304, 2008.

[Koh16] Michael Kohlhase. sTeX: Semantic markup in TeX/LaTeX. Technical report, Comprehensive TeX Archive Network (CTAN), 2016.

[LM06] Paul Libbrecht and Erica Melis. Methods for Access and Retrieval of Mathematical Content in ActiveMath. In N. Takayama and A. Iglesias, editors, *Proceedings of ICMS-2006*, number 4151 in LNAI, pages 331–342. Springer Verlag, 2006. http://www.activemath.org/publications/Libbrecht-Melis-Access-and-Retrieval-ActiveMath-ICMS-2006.pdf.

[LvI10] Peder Olesen Larsen and Markus von Ins. The rate of growth in scientific publication and the decline in coverage provided by science citation index. *Scientometrics*, 84(3):575–603, 2010.

[Mat70] Ju. V. Matijasevič. Enumerable sets are diophantine. *Soviet Math. Doklady*, 11:354–358, 1970.

[MG11] Jozef Misutka and Leo Galambos. System description: Egomath2 as a tool for mathematical searching on wikipedia.org. In James Davenport, William Farmer, Florian Rabe, and Josef Urban, editors, *Calculemus/MKM*, number 6824 in LNAI, pages 307–309. Springer Verlag, 2011.

[MM06] Rajesh Munavalli and Robert Miner. Mathfind: a math-aware search engine. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 735–735, New York, NY, USA, 2006. ACM Press.

[MY03] Bruce R. Miller and Abdou Youssef. Technical aspects of the digital library of mathematical functions. *Annals of Mathematics and Artificial Intelligence*, 38(1-3):121–136, 2003.

[Smu63] Raymond M. Smullyan. A unifying principle for quantification theory. *Proc. Nat. Acad Sciences*, 49:828–832, 1963.

[Sta85] Rick Statman. Logical relations and the typed lambda calculus. *Information and Computation*, 65, 1985.

[vH67] Jean van Heijenoort. *From Frege to Gödel: a source book in mathematical logic 1879-1931*. Source books in the history of the sciences series. Harvard Univ. Press, Cambridge, MA, 3rd printing, 1997 edition, 1967.

[WR10] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica*, volume I. Cambridge University Press, Cambridge, UK, 2 edition, 1910.

[Zer08] Ernst Zermelo. Untersuchungen über die Grundlagen der Mengenlehre. I. *Mathematische Annalen*, 65:261–281, 1908.

Index