Knowledge Representation for Science, Technology, Engineering, and Mathematics Summer Semester 2020

– Lecture Notes –

Prof. Dr. Michael Kohlhase & PD. Dr. Florian Rabe
Professur für Wissensrepräsentation und -verarbeitung
Informatik, FAU Erlangen-Nürnberg
Michael.Kohlhase,Florian.Rabe@FAU.de

July 8, 2020

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.

Organization: Theory and Practice: The KRMT course intended to give a small cohort of students (≤ 15) the opportunity to understand theoretical and practical aspects of knowledge representation for technical documents. The first aspect will be taught as a conventional lecture on computational logic (focusing on the expressive formalisms needed account for the complexity of mathematical objects) and the second will be served by the "KRMT Lab", where we will jointly (instructors and students) develop representations for technical documents and knowledge. Both parts will roughly have equal weight and will alternate weekly.

Prerequisites: The course builds on the logic courses in the FAU Bachelor's program, in particular the course "Grundlagen der Logik in der Informatik" (GLOIN). While prior exposure to logic and inference systems e.g. in GLOIN or the AI-1 course is certainly advantageous to keep up, it is not strictly necessary, as the course introduces all necessary prerequisites as we go along. So a strong motivation or exposure to strong abstraction and mathematical rigour in other areas should be sufficient.

Similarly, we do not presuppose any concrete mathematical knowledge – we mostly use (very) elementary algebra as example domain – but again, exposure to proof-based mathematical practice – whatever it may be – helps a lot.

Course Contents and Organization

The course concentrates on the theory and practice of representing mathematical knowledge in a wide array of mathematical software systems.

In the theoretical part we concentrate on computational logic and mathematical foundations; the course notes are in this document. In the practical part we develop representations of concrete mathematical knowledge in the MMT system, unveiling the functionality of the system step by step. This process is tracked in a tutorial separate document [OMT].

Excursions: As this course is predominantly about modeling natural language and not about the theoretical aspects of the logics themselves, we give the discussion about these as a "suggested readings" **?sec?**. This material can safely be skipped (thus it is in the appendix), but contains the missing parts of the "bridge" from logical forms to truth conditions and textual entailment.

This Document

This document contains the course notes for the course "Knowledge Representation for Mathematical/Technical Knowledge" ("Logik-Basierte Wissensrepräsentation für Mathematisch/Technisches Wissen") in the Summer Semesters 17 ff.

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; Koh20], a TeX/LaTeX extension for semantic markup, which allows to export the contents into active documents that adapt to the reader and can be instrumented with services based on the explicitly represented meaning of the documents.

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

Other Resources: The course notes are complemented by a tutorial on formalization mathematical Knowledge in the MMT system [OMT] and the formalizations at `https://gl.mathhub.info/Tutorials/Mathematicians`.

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 6 for details).

CompLog Students: The course is based on a series of courses "Computational Logic" held at Jacobs University Bremen and shares a lot of material with these. The following students have submitted corrections and suggestions to this and earlier versions of the notes: Rares Ambrus, Florian Rabe, Deyan Ginev, Fulya Horozal, Xu He, Enxhell Luzhnica, and Mihnea Iancu.

KRMT Students: The following students have submitted corrections and suggestions to this and earlier versions of the notes: Michael Banken, Nico Wittstock.

Recorded Syllabus for SS 2020

In this document, we record the progress of the course in the summer semester 2020 in the form of a "recorded syllabus", i.e. a syllabus that is created after the fact rather than before.

Recorded Syllabus Summer Semester 2020:

#	date	what	until	slide	page
1.	April 22.	Lecture	admin, some overview, OBB	11	7
2.	April 23.	Lecture	Theory Graphs Intro	33	18
3.	April 29.	Lab	Formalizing elementary algebra		
4.	April 30.	Lab	Formalizing more algebra (Structures)		
5.	May 6.	Lab	Views		
6.	May 7.	Lab	Formalizing Arithmetics		
7.	May 13.	Lecture	Applications of Framing	35	19
8.	May14.	Lab	More Arithmetics		
9.	May 20.	Lecture	Logic Ideas	43	24
	May 21.		Ascension		
10.	May 27.	Lecture	FOL, subsitutibility	76	47
11.	May 28	Lecture	Higher-Order Logic and λ-calculus	109	69
12.	June 3.	Lecture	λ-calculus via Judgments/Inference	119	74
13.	June 4.	Lab	propositional logic in MMT		
14.	June 10.	Lab	Implementing Propositional Logic		
15.	June 12.	Lab	Implementing FOL		
16.	June 17.	Lab	HW discussion, SFOL, and HOL		
17.	June 18.	Lab	product and function types		
18.	June 24.	Lab	HW Discussion, more λ-calculus rules		
19.	June 25.	Lab	Implementing HOL, Andrews/Pravitz		
20.	July 1.	Lecture	Henkin Semantics and Leibniz Equality	123	76
21.	July 2.	Lab	HOL & Computation/Description		
22.	July 8.	Lecture/Lab	Set Theory, ZFC	**??**	**??**

Here the syllabus of the last academic year for reference, the current year should be similar; see the course notes of last year available for reference at `http://kwarc.info/teaching/KRMT/notes-SS19.pdf`.

Recorded Syllabus Summer Semester 2018:

#	date	what	until	slide	page
1.	April 24.	Lecture	admin, some overview		
2.	April 25.	Lab	MMT Installation, Formalizing elementary algebra		
	May 1.		Tag der Arbeit		
3.	May 2.	Lecture	Theory Graphs Intro, FrameIT		
4.	May 8.	Lecture	Theory Graphs and Applications		
5.	May 9.	Lab	Elementary Algebra upto monoids		
6.	May 15	Lecture	Logics generally, and example logics		
7.	May 16.	Lab	propositional logic in MMT		
8.	May 22.	Lecture	First-Order Logic		
9.	May 23.	Lab	Implementing FOL		
10.	May 29.	Lab	FOL+Equality, untyped $\lambda - calculus$		
	May 30.		Ascension		
11.	June 5.	Lecture	typed λ-calculus		
12.	June 6.	Lab	typed λ-calculus in LF		
13.	June 12.	Lecture	HOL and description		
14.	June 13.	Lab	Implementing HOL		
15.	June 19.	Lecture	Set Theory, ZFC		
	June 20.		Public Holiday: Corpus Christi		
16.	June 26.	Lecture/Lab	ZFC/Implementation		
⋮	⋮	⋮			

Contents

Chapter 1

Administrativa

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.

- 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: there is none!
  - Course notes will be posted at `http://kwarc.info/teaching/KRMT`
  - KRMT Lab follows the tutorial at `https://gl.mathhub.info/Tutorials/Mathematicians/blob/master/tutorial/mmt-math-tutorial.pdf`
  - I mostly prepare/update them as we go along (semantically preloaded ⇝ 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!

 5

Next we come to a special project that is going on in parallel to teaching the course. I am using the course materials as a research object as well. This gives you an additional resource, but may affect the shape of the coures materials (which now serve 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)
  - ▹ (Eventually) feed it into the MathHub system (`http://mathhub.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)

 6

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 (information visualization)

▷ 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 [LI10]
- ▷ 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

 10

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)

 11

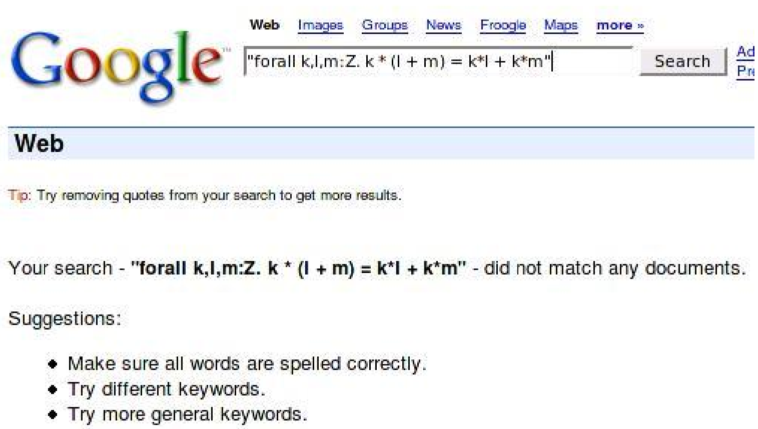
All of that is very abstract, high-level and idealistic, ... Let us look at an example, where we can see computer support for one of the postulated horizontal/MKM tasks in action.

2.2 Mathematical Formula Search

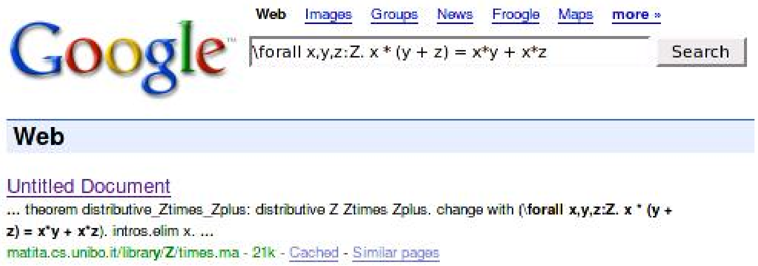
More Mathematics on the Web

- ▷ 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 $\mathbb{Z}$ ($\forall k, l, m \in \mathbb{Z}. k \cdot (l + m) = (k \cdot l) + (k \cdot m)$)

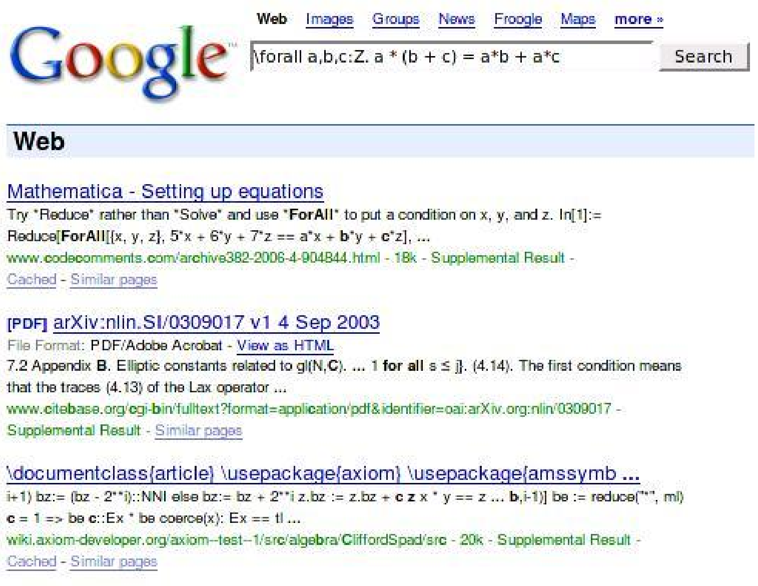
Searching for Distributivity



Searching for Distributivity

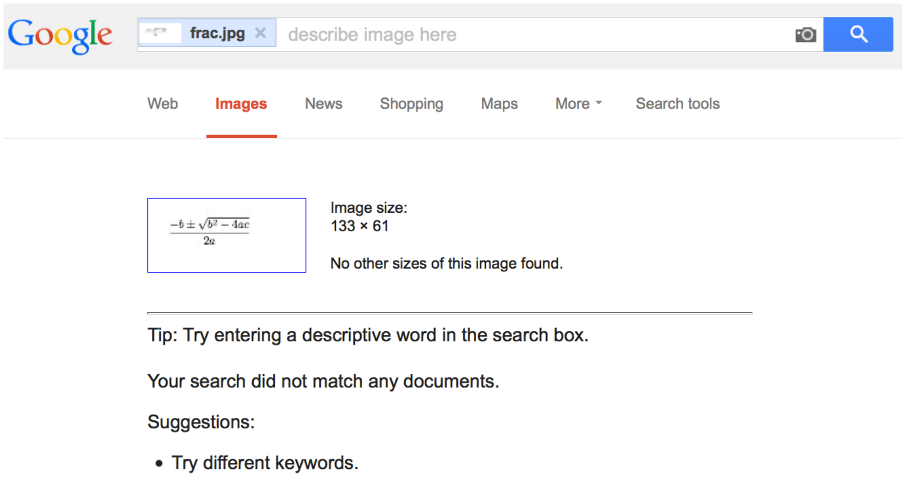


Searching for Distributivity



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 ([Asp+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, C_n^k, and ${}_k\mathcal{J}^n$ 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)

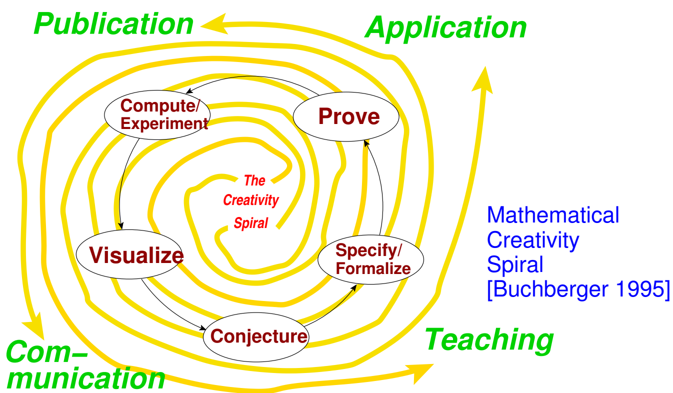
▷ Problem: Find a query language that is intuitive to learn

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

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.

 23

2.4 Modular Representation of mathematical Knowledge

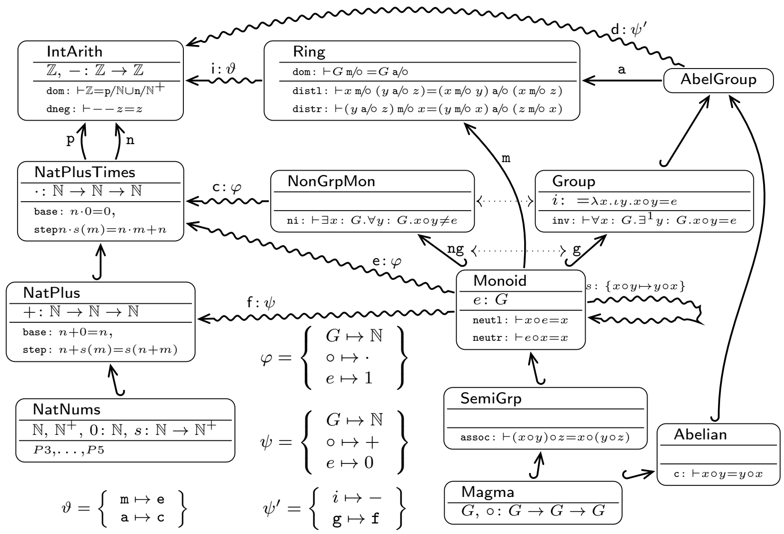
Modular Representation of Math (Theory Graph)

- 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-indepedent 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)

▷ **Example 2.4.2 (Elementary Algebra and Arithmetics)**

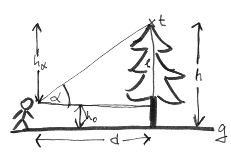


2.5 Application: Serious Games

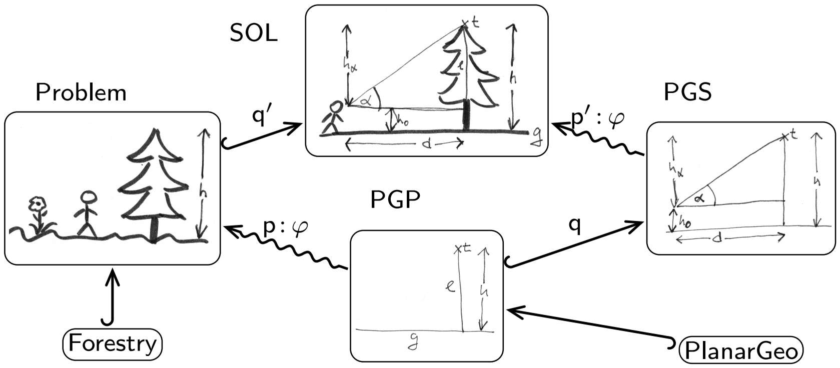
Framing for Problem Solving (The FrameIT Method)

▷ **Example 2.5.1 (Problem 0.8.15)**

How can you measure the height of a tree you cannot climb, when you only have a protractor and a tape measure at hand.

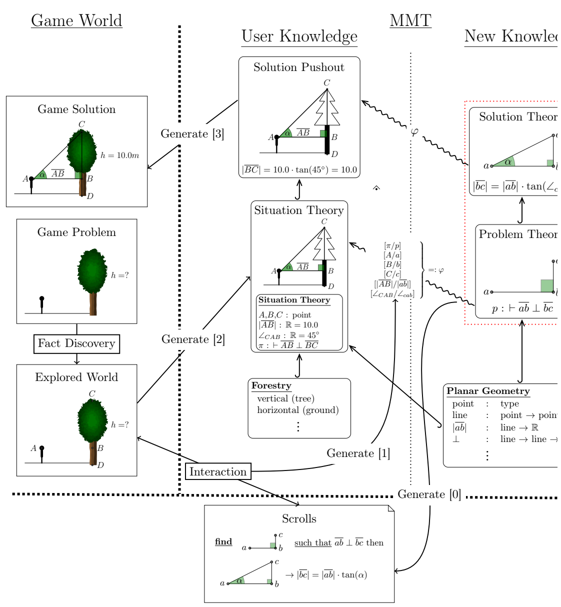


▷ Framing: view the problem as one that is already understood (using theory morphisms)



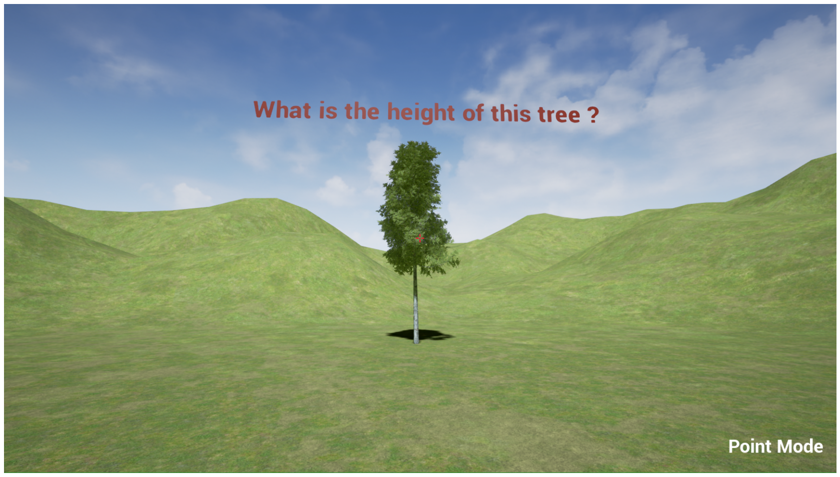
▷ squiggly (framing) morphisms guaranteed by metatheory of theories!

Example Learning Object Graph



FrameIT Method: Problem

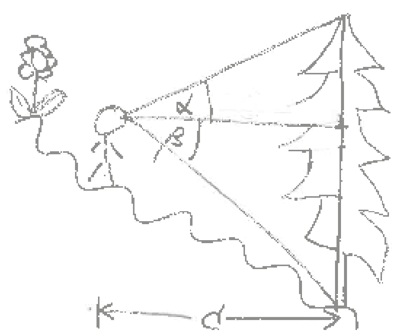
▷ Problem Representation in the game world (what the student should see)



▷ Student can interact with the environment via gadgets so solve problems

▷ "Scrolls" of mathematical knowledge give hints.

Combining Problem/Solution Pairs



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

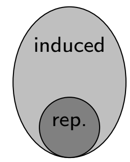
▷ create more complex P/S pairs (e.g. for trees on slopes)

Another whole set of applications and game behaviours 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

♭search: Indexing flattened Theory Graphs

- Simple Idea: We have all the necessary components: MMT and `MathWebSearch`
- **Definition 2.6.3** 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)

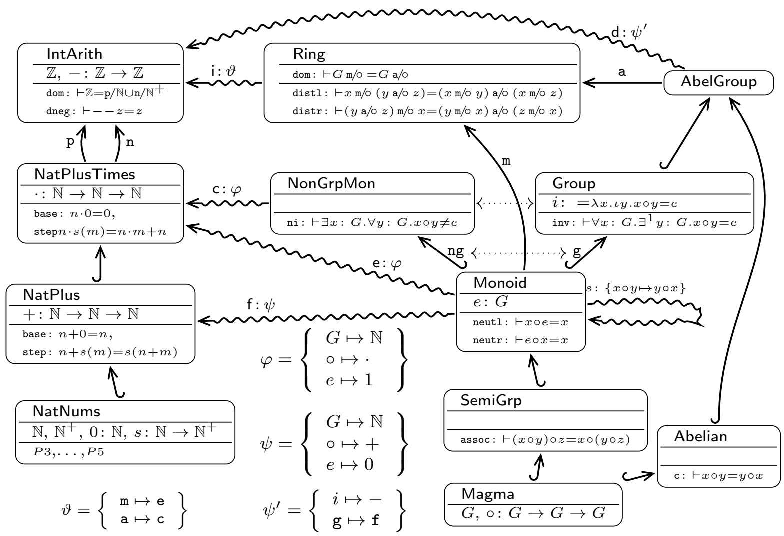
▷ Problem: Hit statement may look considerably different from the induced statement

▷ Solution: Template-based generation of NL explanations from MMT URIs.
MMT knows the necessary information from the components of the MMT URI.

 32

Modular Representation of Math (MMT Example)

▷ **Example 2.6.4 (Elementary Algebra and Arithmetics)**



 33

Example: Explaining a MMT URI

▷ **Example 2.6.5** ♭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

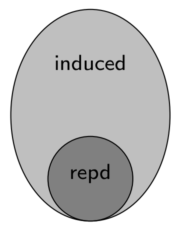
▷ ground the explanation by

In SemiGrps we have the axiom assoc : $\forall x, y, z : G.(x \circ y) \circ z = x \circ (y \circ z)$

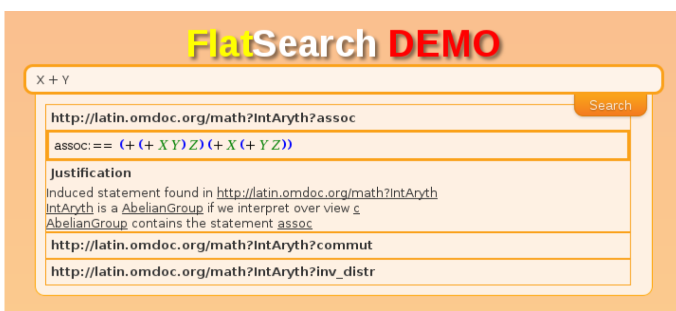
♭search on the LATIN Logic Atlas

▷ 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

<table>
<tr><td colspan="3">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, ...</td></tr>
<tr><td>Foundations of Math:
▷ MathML, OpenMath
▷ advanced Type Theories
▷ MMT: Meta Meta Theory
▷ Logic Morphisms/Atlas
▷ Theorem Prover/CAS Interoperability
▷ Mathematical Models/Simulation</td><td>KM & Interaction:
▷ Semantic Interpretation (aka. Framing)
▷ math-literate interaction
▷ MathHub: math archives & active docs
▷ Semantic Alliance: embedded semantic services</td><td>Semantization:
▷ LaTeXML: LaTeX → XML
▷ sTeX: Semantic LaTeX
▷ invasive editors
▷ Context-Aware IDEs
▷ Mathematical Corpora
▷ Linguistics of Math
▷ ML for Math Semantics Extraction</td></tr>
<tr><td colspan="3">Foundations: Computational Logic, Web Technologies, OMDoc/MMT</td></tr>
</table>

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. (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 ($x, y, f, g, p, 1, \pi, \in, \neg, \wedge \forall, \exists$)
- Model: 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 [Hei67].

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 (⇝ 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 ∼ 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 ∼ 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 ($\langle\mathbb{N},+,*\rangle$),
  - 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

Foundations of Mathematics

Chapter 4

Propositional Logic and Inference

4.1 Propositional Logic (Syntax/Semantics)

Propositional Logic (Syntax)

▷ **Definition 4.1.1 (Syntax)** The formulae of propositional logic (write PL^0) are made up from

- ▷ propositional variables: $\mathcal{V}_o := \{P, Q, R, P^1, P^2, \ldots\}$ (countably infinite)
- ▷ construtors/constants called connectives: $\Sigma_o := \{T, F, \neg, \vee, \wedge, \Rightarrow, \Leftrightarrow, \ldots\}$

We define the set $\mathit{wff}_o(\mathcal{V}_o)$ of well-formed propositional formulas as

- ▷ propositional variables T and F,
- ▷ the logical constants,
- ▷ negations $\neg\mathbf{A}$,
- ▷ conjunctions $\mathbf{A} \wedge \mathbf{B}$,
- ▷ disjunctions $\mathbf{A} \vee \mathbf{B}$,
- ▷ implications $\mathbf{A} \Rightarrow \mathbf{B}$,
- ▷ equivalences (or biimplications) $\mathbf{A} \Leftrightarrow \mathbf{B}$,

where $\mathbf{A}, \mathbf{B} \in \mathit{wff}_o(\mathcal{V}_o)$ themselves.

▷ **Example 4.1.2** $P \wedge Q, P \vee Q, (\neg P \vee Q) \Leftrightarrow (P \Rightarrow Q) \in \mathit{wff}_o(\mathcal{V}_o)$

▷ **Definition 4.1.3** propositional formulae without connectives are called atomic (or atoms) and complex otherwise.

 45 FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Alternative Notations for Connectives

Here	Elsewhere		
$\neg\mathbf{A}$	$\sim\mathbf{A}$	$\overline{\mathbf{A}}$	
$\mathbf{A}\wedge\mathbf{B}$	$\mathbf{A}\&\mathbf{B}$	$\mathbf{A}\bullet\mathbf{B}$	$\mathbf{A},\mathbf{B}$
$\mathbf{A}\vee\mathbf{B}$	$\mathbf{A}+\mathbf{B}$	$\mathbf{A}\mid\mathbf{B}$	$\mathbf{A};\mathbf{B}$
$\mathbf{A}\Rightarrow\mathbf{B}$	$\mathbf{A}\rightarrow\mathbf{B}$	$\mathbf{A}\supset\mathbf{B}$	
$\mathbf{A}\Leftrightarrow\mathbf{B}$	$\mathbf{A}\leftrightarrow\mathbf{B}$	$\mathbf{A}\equiv\mathbf{B}$	
F	$\bot$	0	
T	$\top$	1	

 46

Semantics of PL^0 (Models)

▷ **Definition 4.1.4** A model $\mathcal{M} := \langle\mathcal{D}_o, \mathcal{I}\rangle$ for propositional logic consists of

- ▹ the universe $\mathcal{D}_o = \{\mathsf{T}, \mathsf{F}\}$
- ▹ the interpretation $\mathcal{I}$ that assigns values to essential connectives
- ▹ $\mathcal{I}(\neg)\colon \mathcal{D}_o \to \mathcal{D}_o; \mathsf{T} \mapsto \mathsf{F}, \mathsf{F} \mapsto \mathsf{T}$
- ▹ $\mathcal{I}(\wedge)\colon \mathcal{D}_o \times \mathcal{D}_o \to \mathcal{D}_o; \langle\alpha, \beta\rangle \mapsto \mathsf{T}$, iff $\alpha = \beta = \mathsf{T}$

We call a constructor a logical constant, iff its value is fixed by the interpretation

▷ Treat the other connectives as abbreviations, e.g. $\mathbf{A}\vee\mathbf{B}\widehat{=}\neg(\neg\mathbf{A}\wedge\neg\mathbf{B})$ and $\mathbf{A}\Rightarrow\mathbf{B}\widehat{=}\neg\mathbf{A}\vee\mathbf{B}$, and $T\widehat{=}=P\vee\neg P$ (only need to treat $\neg, \wedge$ directly)

 47

Semantics of PL^0 (Evaluation)

▷ Problem: The interpretation function only assigns meaning to connectives.

▷ **Definition 4.1.5** A variable assignment $\varphi\colon \mathcal{V}_o \to \mathcal{D}_o$ assigns values to propositional variables.

▷ **Definition 4.1.6** The value function $\mathcal{I}_\varphi\colon \mathit{wff}_o(\mathcal{V}_o) \to \mathcal{D}_o$ assigns values to PL^0 formulae. It is recursively defined,

- ▹ $\mathcal{I}_\varphi(P) = \varphi(P)$ (base case)
- ▹ $\mathcal{I}_\varphi(\neg\mathbf{A}) = \mathcal{I}(\neg)(\mathcal{I}_\varphi(\mathbf{A}))$.
- ▹ $\mathcal{I}_\varphi(\mathbf{A}\wedge\mathbf{B}) = \mathcal{I}(\wedge)(\mathcal{I}_\varphi(\mathbf{A}), \mathcal{I}_\varphi(\mathbf{B}))$.

▷ Note that $\mathcal{I}_\varphi(\mathbf{A}\vee\mathbf{B}) = \mathcal{I}_\varphi(\neg(\neg\mathbf{A}\wedge\neg\mathbf{B}))$ is only determined by $\mathcal{I}_\varphi(\mathbf{A})$ and $\mathcal{I}_\varphi(\mathbf{B})$, so we think of the defined connectives as logical constants as well.

 48

We will now use the distribution of values of a Boolean expression under all (variable) assignments to characterize them semantically. The intuition here is that we want to understand theorems, examples, counterexamples, and inconsistencies in mathematics and everyday reasoning[1].

The idea is to use the formal language of Boolean expressions as a model for mathematical language. Of course, we cannot express all of mathematics as Boolean expressions, but we can at least study the interplay of mathematical statements (which can be true or false) with the copula "and", "or" and "not".

Semantic Properties of Propositional Formulae

- **Definition 4.1.7** Let $\mathcal{M} := \langle \mathcal{U}, \mathcal{I} \rangle$ be our model, then we call $\mathbf{A}$
  - true under φ (φ satisfies $\mathbf{A}$) in $\mathcal{M}$, iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$ (write $\mathcal{M} \models^\varphi \mathbf{A}$)
  - false under φ (φ falsifies $\mathbf{A}$) in $\mathcal{M}$, iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{F}$ (write $\mathcal{M} \not\models^\varphi \mathbf{A}$)
  - satisfiable in $\mathcal{M}$, iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$ for some assignment φ
  - valid in $\mathcal{M}$, iff $\mathcal{M} \models^\varphi \mathbf{A}$ for all assignments φ (write $\mathcal{M} \models \mathbf{A}$)
  - falsifiable in $\mathcal{M}$, iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{F}$ for some assignments φ
  - unsatisfiable in $\mathcal{M}$, iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{F}$ for all assignments φ
- **Example 4.1.8** $x \vee x$ is satisfiable and falsifiable.
- **Example 4.1.9** $x \vee \neg x$ is valid and $x \wedge \neg x$ is unsatisfiable.
- **Notation 4.1.10** (alternative) Write $[\![\mathbf{A}]\!]_\varphi^\mathcal{M}$ for $\mathcal{I}_\varphi(\mathbf{A})$, if $\mathcal{M} = \langle \mathcal{U}, \mathcal{I} \rangle$. (and $[\![\mathbf{A}]\!]^\mathcal{M}$, if $\mathbf{A}$ is ground, and $[\![\mathbf{A}]\!]$, if $\mathcal{M}$ is clear)
- **Definition 4.1.11 (Entailment)** (aka. logical consequence)

  We say that $\mathbf{A}$ entails $\mathbf{B}$ ($\mathbf{A} \models \mathbf{B}$), iff $\mathcal{I}_\varphi(\mathbf{B}) = \mathsf{T}$ for all φ with $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$ (i.e. all assignments that make $\mathbf{A}$ true also make $\mathbf{B}$ true)

Let us now see how these semantic properties model mathematical practice.

In mathematics we are interested in assertions that are true in all circumstances. In our model of mathematics, we use variable assignments to stand for circumstances. So we are interested in Boolean expressions which are true under all variable assignments; we call them valid. We often give examples (or show situations) which make a conjectured assertion false; we call such examples counterexamples, and such assertions "falsifiable". We also often give examples for certain assertions to show that they can indeed be made true (which is not the same as being valid yet); such assertions we call "satisfiable". Finally, if an assertion cannot be made true in any circumstances we call it "unsatisfiable"; such assertions naturally arise in mathematical practice in the form of refutation proofs, where we show that an assertion (usually the negation of the theorem we want to prove) leads to an obviously unsatisfiable conclusion, showing that the negation of the theorem is unsatisfiable, and thus the theorem valid.

4.2 Calculi for Propositional Logic

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.

[1]Here (and elsewhere) we will use mathematics (and the language of mathematics) as a test tube for understanding reasoning, since mathematics has a long history of studying its own reasoning processes and assumptions.

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 4.2.1** 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 4.2.2** 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 4.2.3** 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 4.2.4** An inference rule without assumptions is called an axiom (schema).

▷ **Definition 4.2.5** 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 $\mathcal{S}$.

 50 FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

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 4.2.6** 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 $\frac{\mathbf{A}_{l_1} \quad \cdots \quad \mathbf{A}_{l_k}}{\mathbf{A}_i}$ 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 4.2.7**

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.

$$\dfrac{\dfrac{}{P \Rightarrow Q \Rightarrow P}K \quad P}{Q \Rightarrow P}MP$$

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 4.2.7 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 4.2.7.

Formal Systems

- ⊳ **Observation 4.2.8** *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 4.2.6 is a derivation relation in the sense of Definition 4.2.1.*[1]

- ⊳ **Definition 4.2.9** 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 4.2.10** 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 4.2.11** 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.

- ⊳ **Definition 4.2.12** An inference rule $\dfrac{\mathbf{A}_1 \quad \cdots \quad \mathbf{A}_n}{\mathbf{C}}$ is called derivable in $\mathcal{C}$, if there is a $\mathcal{C}$-derivation $\{\mathbf{A}_1, \ldots, \mathbf{A}_n\} \vdash_{\mathcal{C}} \mathbf{C}$.

- ⊳ **Observation 4.2.13** *Derivable inference rules are admissible, but not the other way around.*

[a] EdNote: MK: this should become a view!

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?

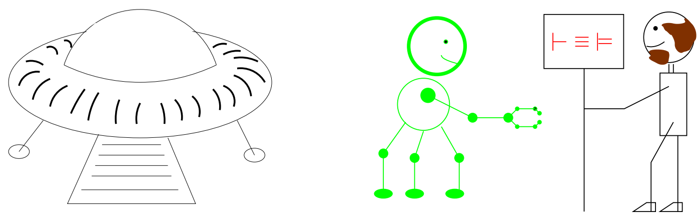
Soundness and Completeness

- ⊳ **Definition 4.2.14** 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.

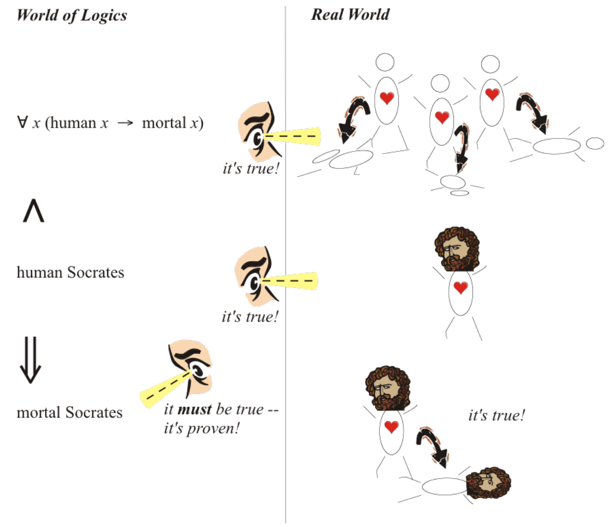
Of course, the logics we have studied so far are very simple, and not able to express interesting facts about the world, but we will study them as a simple example of the fundamental problem of Computer Science: How do the formal representations correlate with the real world.

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!)

4.3 Propositional Natural Deduction Calculus

We will now introduce the "natural deduction" calculus for propositional logic. The 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.

We will introduce natural deduction in two styles/notation, both were invented by Gerhard Gentzen in the 1930's and are very much related. The Natural Deduction style (ND) uses "local hypotheses" in proofs for hypothetical reasoning, while the "sequent style" is a rationalized version and extension of the ND calculus that makes certain meta-proofs simpler to push through by making the context of local hypotheses explicit in the notation. The sequent notation also constitutes a more adequate data struture for implementations, and user interfaces.

Rather than using a minimal set of inference rules, the natural deduction calculus provides two/three 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).

Calculi: Natural Deduction ($\mathcal{ND}^0$; Gentzen [Gen34])

- ▷ Idea: $\mathcal{ND}^0$ tries to mimic human theorem proving behavior (non-minimal)
- ▷ **Definition 4.3.1** The propositional natural deduction calculus $\mathcal{ND}^0$ has rules for the introduction and elimination of connectives

Introduction	Elimination	Axiom
$\dfrac{\mathbf{A}\quad\mathbf{B}}{\mathbf{A}\wedge\mathbf{B}}\wedge I$	$\dfrac{\mathbf{A}\wedge\mathbf{B}}{\mathbf{A}}\wedge E_l \quad \dfrac{\mathbf{A}\wedge\mathbf{B}}{\mathbf{B}}\wedge E_r$	$\dfrac{}{\mathbf{A}\vee\neg\mathbf{A}}\text{TND}$
$\dfrac{\begin{array}{c}[\mathbf{A}]^1\\ \hline\hline \mathbf{B}\end{array}}{\mathbf{A}\Rightarrow\mathbf{B}}\Rightarrow I^1$	$\dfrac{\mathbf{A}\Rightarrow\mathbf{B}\quad\mathbf{A}}{\mathbf{B}}\Rightarrow E$	

▷ TND is used only in classical logic (otherwise constructive/intuitionistic)

The most characteristic rule in the natural deduction calculus is the $\Rightarrow I$ rule. It corresponds to the mathematical way of proving an implication $\mathbf{A}\Rightarrow\mathbf{B}$: We assume that $\mathbf{A}$ is true and show $\mathbf{B}$ from this assumption. When we can do this we discharge (get rid of) the assumption and conclude $\mathbf{A}\Rightarrow\mathbf{B}$. This mode of reasoning is called hypothetical reasoning. Note that the local hypothesis is discharged by the rule $\Rightarrow I$, i.e. it cannot be used in any other part of the proof. As the $\Rightarrow I$ rules may be nested, we decorate both the rule and the corresponding assumption with a marker (here the number 1).

Let us now consider an example of hypothetical reasoning in action.

Natural Deduction: Examples

▷ **Example 4.3.2 (Inference with Local Hypotheses)**

$$\dfrac{\dfrac{\dfrac{[\mathbf{A}\wedge\mathbf{B}]^1}{\mathbf{B}}\wedge E_r \quad \dfrac{[\mathbf{A}\wedge\mathbf{B}]^1}{\mathbf{A}}\wedge E_l}{\mathbf{B}\wedge\mathbf{A}}\wedge I}{\mathbf{A}\wedge\mathbf{B}\Rightarrow\mathbf{B}\wedge\mathbf{A}}\Rightarrow I^1$$

$$\begin{array}{c}[A]^1\\ [B]^2\\ \dfrac{\dfrac{A}{B\Rightarrow A}\Rightarrow I^2}{A\Rightarrow B\Rightarrow A}\Rightarrow I^1\end{array}$$

Here we see reasoning with local hypotheses at work. In the left example, we assume the formula $\mathbf{A}\wedge\mathbf{B}$ and can use it in the proof until it is discharged by the rule $\wedge E_l$ on the bottom – therefore we decorate the hypothesis and the rule by corresponding numbers (here the label "1"). Note the assumption $\mathbf{A}\wedge\mathbf{B}$ is *local to the proof fragment* delineated by the corresponding hypothesis and the discharging rule, i.e. even if this proof is only a fragment of a larger proof, then we cannot use its hypothesis anywhere else. Note also that we can use as many copies of the local hypothesis as we need; they are all discharged at the same time.

In the right example we see that local hypotheses can be nested as long as hypotheses are kept local. In particular, we may not use the hypothesis $\mathbf{B}$ after the $\Rightarrow I^2$, e.g. to continue with a $\Rightarrow E$.

One of the nice things about the natural deduction calculus is that the deduction theorem is almost trivial to prove. In a sense, the triviality of the deduction theorem is the central idea of the calculus and the feature that makes it so natural.

A Deduction Theorem for $\mathcal{ND}^0$

▷ **Theorem 4.3.3** $\mathcal{H}, \mathbf{A} \vdash_{\mathcal{ND}^0} \mathbf{B}$, *iff* $\mathcal{H} \vdash_{\mathcal{ND}^0} \mathbf{A} \Rightarrow \mathbf{B}$.

▷ Proof: We show the two directions separately

P.1 If $\mathcal{H}, \mathbf{A} \vdash_{\mathcal{ND}^0} \mathbf{B}$, then $\mathcal{H} \vdash_{\mathcal{ND}^0} \mathbf{A} \Rightarrow \mathbf{B}$ by $\Rightarrow I$, and

P.2 If $\mathcal{H} \vdash_{\mathcal{ND}^0} \mathbf{A} \Rightarrow \mathbf{B}$, then $\mathcal{H}, \mathcal{A} \vdash_{\mathcal{ND}^0} \mathbf{A} \Rightarrow \mathbf{B}$ by weakening and $\mathcal{H}, \mathcal{A} \vdash_{\mathcal{ND}^0} \mathbf{B}$ by $\Rightarrow E$. □

Another characteristic of the natural deduction calculus is that it has inference rules (introduction and elimination rules) for all connectives. So we extend the set of rules from Definition 5.2.1 for disjunction, negation and falsity.

More Rules for Natural Deduction

▷ **Definition 4.3.4** $\mathcal{ND}^0$ has the following additional rules for the remaining connectives.

$$\frac{\mathbf{A}}{\mathbf{A} \vee \mathbf{B}} \vee I_l \qquad \frac{\mathbf{B}}{\mathbf{A} \vee \mathbf{B}} \vee I_r \qquad \frac{\mathbf{A} \vee \mathbf{B} \quad \begin{matrix}[\mathbf{A}]^1 \\ \vdots \\ \mathbf{C}\end{matrix} \quad \begin{matrix}[\mathbf{B}]^1 \\ \vdots \\ \mathbf{C}\end{matrix}}{\mathbf{C}} \vee E^1$$

$$\frac{\begin{matrix}[\mathbf{A}]^1 \\ \vdots \\ F\end{matrix}}{\neg \mathbf{A}} \neg I^1 \qquad \frac{\neg\neg \mathbf{A}}{\mathbf{A}} \neg E$$

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

Natural Deduction in Sequent Calculus Formulation

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

▷ **Definition 4.3.5** A judgment is a meta-statement about the provability of propositions

▷ **Definition 4.3.6** 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.

Write $\vdash \mathbf{A}$ for $\emptyset \vdash \mathbf{A}$.

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

▷ **Example 4.3.7** We give the sequent-style version of Example 5.2.2

$$\dfrac{\dfrac{\dfrac{}{\mathbf{A}\wedge\mathbf{B}\vdash\mathbf{A}\wedge\mathbf{B}}\,\mathrm{Ax}}{\mathbf{A}\wedge\mathbf{B}\vdash\mathbf{B}}\,\wedge E_r \quad \dfrac{\dfrac{}{\mathbf{A}\wedge\mathbf{B}\vdash\mathbf{A}\wedge\mathbf{B}}\,\mathrm{Ax}}{\mathbf{A}\wedge\mathbf{B}\vdash\mathbf{A}}\,\wedge E_l}{\dfrac{\mathbf{A}\wedge\mathbf{B}\vdash\mathbf{B}\wedge\mathbf{A}}{\vdash\mathbf{A}\wedge\mathbf{B}\Rightarrow\mathbf{B}\wedge\mathbf{A}}\,\Rightarrow I}\,\wedge I \qquad \dfrac{\dfrac{\dfrac{}{\mathbf{A},\mathbf{B}\vdash\mathbf{A}}\,\mathrm{Ax}}{\mathbf{A}\vdash\mathbf{B}\Rightarrow\mathbf{A}}\,\Rightarrow I}{\vdash\mathbf{A}\Rightarrow\mathbf{B}\Rightarrow\mathbf{A}}\,\Rightarrow I$$

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 4.3.8** The following inference rules make up the propositional sequent-style natural deduction calculus $\mathcal{ND}^0_\vdash$:

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

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

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

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

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

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

Linearized Notation for (Sequent-Style) ND Proofs

▷ Linearized notation for sequent-style ND proofs

1.	$\mathcal{H}_1$	$\vdash$	$\mathbf{A}_1$	$(\mathcal{J}_1)$
2.	$\mathcal{H}_2$	$\vdash$	$\mathbf{A}_2$	$(\mathcal{J}_2)$
3.	$\mathcal{H}_3$	$\vdash$	$\mathbf{A}_3$	$(\mathcal{R}1,2)$

corresponds to

$$\dfrac{\mathcal{H}_1\vdash\mathbf{A}_1\quad\mathcal{H}_2\vdash\mathbf{A}_2}{\mathcal{H}_3\vdash\mathbf{A}_3}\,\mathcal{R}$$

▷ **Example 4.3.9** We show a linearized version of Example 5.2.7

#	hyp	⊢	formula	NDjust
1.	1	⊢	$\mathbf{A} \wedge \mathbf{B}$	Ax
2.	1	⊢	$\mathbf{B}$	$\wedge E_r 1$
3.	1	⊢	$\mathbf{A}$	$\wedge E_l 1$
4.	1	⊢	$\mathbf{B} \wedge \mathbf{A}$	$\wedge I 2, 1$
5.		⊢	$\mathbf{A} \wedge \mathbf{B} \Rightarrow \mathbf{B} \wedge \mathbf{A}$	$\Rightarrow I 4$

#	hyp	⊢	formula	NDjust
1.	1	⊢	$\mathbf{A}$	Ax
2.	2	⊢	$\mathbf{B}$	Ax
3.	1, 2	⊢	$\mathbf{A}$	weaken 1, 2
4.	1	⊢	$\mathbf{B} \Rightarrow \mathbf{A}$	$\Rightarrow I 3$
5.		⊢	$\mathbf{A} \Rightarrow \mathbf{B} \Rightarrow \mathbf{A}$	$\Rightarrow I 4$

 61

Each row in the table represents one inference step in the proof. It consists of line number (for referencing), a formula for the asserted property, a justification via a ND rules (and the rows this one is derived from), and finally a list of row numbers of proof stepsx0 that are local hypotheses in effect for the current row.

Chapter 5

First Order Predicate Logic

5.1 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.

First-Order Predicate Logic (PL^1)

- Coverage: We can talk about (*All humans are mortal*)
  - individual things and denote them by variables or constants
  - properties of individuals, (e.g. being human or mortal)
  - relations of individuals, (e.g. $sibling\_of$ relationship)
  - functions on individuals, (e.g. the $father\_of$ function)

  We can also state the existence of an individual with a certain property, or the universality of a property.
- But we cannot state assertions like
  - *There is a surjective function from the natural numbers into the reals*.
- First-Order Predicate Logic has many good properties (complete calculi, compactness, unitary, linear unification,...)
- But too weak for formalizing: (at least directly)
  - natural numbers, torsion groups, calculus, ...
  - generalized quantifiers (*most, at least three, some,...*)

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 Section 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.

5.1.1 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 5.1.1** 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 5.1.2** 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: $\Sigma_k^f = \{f, g, h, \ldots\}$ (functions on individuals)
  - predicate constants: $\Sigma_k^p = \{p, q, r, \ldots\}$ (relations among inds.)
  - (Skolem constants: $\Sigma_k^{sk} = \{f_1^k, f_2^k, \ldots\}$) (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: $\mathcal{V}_\iota = \{X_\iota, Y_\iota, Z_\iota, X_\iota^1, X_\iota^2, \ldots\}$ (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 5.1.3** Terms: $\mathbf{A} \in wff_\iota(\Sigma_\iota)$ (denote individuals: type ι)

- ▷ $\mathcal{V}_\iota \subseteq wff_\iota(\Sigma_\iota)$,
- ▷ if $f \in \Sigma_k^f$ 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 5.1.4** Propositions: $\mathbf{A} \in wff_o(\Sigma)$ (denote truth values: type o)

- ▷ if $p \in \Sigma_k^p$ 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 $\mathbf{A}, \mathbf{B} \in wff_o(\Sigma)$ and $X \in \mathcal{V}_\iota$, then $T, \mathbf{A} \wedge \mathbf{B}, \neg \mathbf{A}, \forall X.\mathbf{A} \in wff_o(\Sigma)$.

▷ **Definition 5.1.5** 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 5.1.6** We use $\exists X.\mathbf{A}$ as an abbreviation for $\neg(\forall X.\neg \mathbf{A})$. (existential quantifier)

▷ **Definition 5.1.7** 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 5.1.8** 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 5.1.9** We define the set free(**A**) of free variables of a formula **A** inductively:

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

▷ **Definition 5.1.10** We call a formula **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 5.1.11** 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)
- ▹ $\mathcal{I}\colon \Sigma_k^f \to {\mathcal{D}_\iota}^k \to \mathcal{D}_\iota$ (interpret function symbols as arbitrary functions)
- ▹ $\mathcal{I}\colon \Sigma_k^p \to \mathcal{P}({\mathcal{D}_\iota}^k)$ (interpret predicates as arbitrary relations)

▷ **Definition 5.1.12** 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)

- ▷ $\mathcal{I}_\varphi\colon wff_\iota(\Sigma_\iota) \to \mathcal{D}_\iota$ assigns values to terms.
  - ▷ $\mathcal{I}_\varphi(X) := \varphi(X)$ and
  - ▷ $\mathcal{I}_\varphi(f(\mathbf{A}_1,\ldots,\mathbf{A}_k)) := \mathcal{I}(f)(\mathcal{I}_\varphi(\mathbf{A}_1),\ldots,\mathcal{I}_\varphi(\mathbf{A}_k))$
- ▷ $\mathcal{I}_\varphi\colon wff_o(\Sigma) \to \mathcal{D}_o$ 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 $\langle\mathcal{I}_\varphi(\mathbf{A}^1),\ldots,\mathcal{I}_\varphi(\mathbf{A}^k)\rangle \in \mathcal{I}(p)$
  - ▷ $\mathcal{I}_\varphi(\forall X.\mathbf{A}) := \mathsf{T}$, iff $\mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{A}) = \mathsf{T}$ for all $\mathsf{a}\in\mathcal{D}_\iota$.

▷ **Definition 5.1.13 (Assignment Extension)** Let φ be a variable assignment and $\mathsf{a}\in\mathcal{D}_\iota$, then we denote with $\varphi,[\mathsf{a}/X]$ the extended assignment $\{(Y,\mathsf{b})\in\varphi \mid Y\neq X\}\cup\{(X,\mathsf{a})\}$. ($\varphi,[\mathsf{a}/X]$ coincides with φ off X, and gives the result a there)

 68

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 $\forall x.\mathbf{A}$, the occurrences of the variable x in $\mathbf{A}$ that were bound in $\forall x.\mathbf{A}$ become free and are amenable to evaluation by the variable assignment $\psi := \varphi,[\mathsf{a}/X]$. 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 $\mathcal{I}_\varphi(\exists x.\mathbf{A})$ of $\exists x.\mathbf{A}$, which we have defined to be $\neg(\forall x.\neg\mathbf{A})$ is true, iff it is not the case that $\mathcal{I}_\varphi(\forall x.\neg\mathbf{A}) = \mathcal{I}_\psi(\neg\mathbf{A}) = \mathsf{F}$ for all $\mathsf{a}\in\mathcal{D}_\iota$ and $\psi := \varphi,[\mathsf{a}/X]$. 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.

Semantics Computation: Example

▷ **Example 5.1.14** We define an instance of first-order logic:

- ▷ Signature: Let $\Sigma_0^f := \{j,m\}$, $\Sigma_1^f := \{f\}$, and $\Sigma_2^p := \{o\}$
- ▷ Universe: $\mathcal{D}_\iota := \{J,M\}$
- ▷ Interpretation: $\mathcal{I}(j) := J$, $\mathcal{I}(m) := M$, $\mathcal{I}(f)(J) := M$, $\mathcal{I}(f)(M) := M$, and $\mathcal{I}(o) := \{(M,J)\}$.

Then $\forall X.o(f(X),X)$ is a sentence and with $\psi := \varphi,[\mathsf{a}/X]$ for $\mathsf{a}\in\mathcal{D}_\iota$ we have

$$\begin{aligned}\mathcal{I}_\varphi(\forall X. o(f(X),X)) = \mathsf{T} \quad & \text{iff} \quad \mathcal{I}_\psi(o(f(X),X)) = \mathsf{T} \text{ for all } \mathsf{a} \in \mathcal{D}_\iota \\ & \text{iff} \quad (\mathcal{I}_\psi(f(X)), \mathcal{I}_\psi(X)) \in \mathcal{I}(o) \text{ for all } \mathsf{a} \in \{J,M\} \\ & \text{iff} \quad (\mathcal{I}(f)(\mathcal{I}_\psi(X)), \psi(X)) \in \{(M,J)\} \text{ for all } \mathsf{a} \in \{J,M\} \\ & \text{iff} \quad (\mathcal{I}(f)(\psi(X)), a) = (M,J) \text{ for all } \mathsf{a} \in \{J,M\} \\ & \text{iff} \quad \mathcal{I}(f)(a) = M \text{ and } a = J \text{ for all } \mathsf{a} \in \{J,M\}\end{aligned}$$

But $\mathsf{a} \neq J$ for $\mathsf{a} = M$, so $\mathcal{I}_\varphi(\forall X. o(f(X),X)) = \mathsf{F}$ in the model $\langle \mathcal{D}_\iota, \mathcal{I} \rangle$.

5.1.2 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 5.1.15** We call $\sigma\colon wff_\iota(\Sigma_\iota) \to wff_\iota(\Sigma_\iota)$ 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 5.1.16** *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 5.1.17** 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 5.1.18** $[a/x],[f(b)/y],[a/z]$ instantiates $g(x,y,h(z))$ to $g(a,f(b),h(a))$.
- ▷ **Definition 5.1.19** 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

- **Definition 5.1.20 (Substitution Extension)** Let σ be a substitution, then we denote with $\sigma, [\mathbf{A}/X]$ the function $\{(Y, \mathbf{B}) \in \sigma \,|\, Y \neq X\} \cup \{(X, \mathbf{A})\}$. ($\sigma, [\mathbf{A}/X]$ coincides with σ off X, and gives the result $\mathbf{A}$ there.)
- Note: If σ is a substitution, then $\sigma, [\mathbf{A}/X]$ is also a substitution.
- **Definition 5.1.21** 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 5.1.22** 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 5.1.17 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 5.1.23** $\sigma(\forall X.\mathbf{A}) := (\forall X.\sigma_{-X}(\mathbf{A}))$.
- 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 5.1.24** 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 $Y \in \mathrm{free}(\mathbf{B})$.
- 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 5.1.25** *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 5.1.26** *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.

5.1.3 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 5.1.27** *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 5.1.28** 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, 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 5.1.29 (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 5.1.30** 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.
- ▷ Alternative: Replace variables with numbers in formulae (de Bruijn indices).

5.2 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.

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.

5.2.1 Propositional Natural Deduction Calculus

We will now introduce the "natural deduction" calculus for propositional logic. The 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.

We will introduce natural deduction in two styles/notation, both were invented by Gerhard Gentzen in the 1930's and are very much related. The Natural Deduction style (ND) uses "local hypotheses" in proofs for hypothetical reasoning, while the "sequent style" is a rationalized version and extension of the ND calculus that makes certain meta-proofs simpler to push through by making the context of local hypotheses explicit in the notation. The sequent notation also constitutes a more adequate data struture for implementations, and user interfaces.

Rather than using a minimal set of inference rules, the natural deduction calculus provides two/three 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).

Calculi: Natural Deduction ($\mathcal{ND}^0$; Gentzen [Gen34])

- ▷ Idea: $\mathcal{ND}^0$ tries to mimic human theorem proving behavior (non-minimal)
- ▷ **Definition 5.2.1** The propositional natural deduction calculus $\mathcal{ND}^0$ has rules for the introduction and elimination of connectives

Introduction

$$\frac{\mathbf{A}\quad\mathbf{B}}{\mathbf{A}\wedge\mathbf{B}}\wedge I \qquad \frac{\begin{array}{c}[\mathbf{A}]^1\\ \vdots \\ \mathbf{B}\end{array}}{\mathbf{A}\Rightarrow\mathbf{B}}\Rightarrow I^1$$

Elimination

$$\frac{\mathbf{A}\wedge\mathbf{B}}{\mathbf{A}}\wedge E_l \qquad \frac{\mathbf{A}\wedge\mathbf{B}}{\mathbf{B}}\wedge E_r \qquad \frac{\mathbf{A}\Rightarrow\mathbf{B}\quad\mathbf{A}}{\mathbf{B}}\Rightarrow E$$

Axiom

$$\frac{}{\mathbf{A}\vee\neg\mathbf{A}}\text{TND}$$

- ▷ TND is used only in classical logic (otherwise constructive/intuitionistic)

The most characteristic rule in the natural deduction calculus is the $\Rightarrow I$ rule. It corresponds to the mathematical way of proving an implication $\mathbf{A}\Rightarrow\mathbf{B}$: We assume that $\mathbf{A}$ is true and show $\mathbf{B}$ from this assumption. When we can do this we discharge (get rid of) the assumption and conclude $\mathbf{A}\Rightarrow\mathbf{B}$. This mode of reasoning is called hypothetical reasoning. Note that the local hypothesis is discharged by the rule $\Rightarrow I$, i.e. it cannot be used in any other part of the proof. As the $\Rightarrow I$ rules may be nested, we decorate both the rule and the corresponding assumption with a marker (here the number 1).

Let us now consider an example of hypothetical reasoning in action.

Natural Deduction: Examples

- ▷ **Example 5.2.2 (Inference with Local Hypotheses)**

$$\frac{\dfrac{\dfrac{[\mathbf{A}\wedge\mathbf{B}]^1}{\mathbf{B}}\wedge E_r \quad \dfrac{[\mathbf{A}\wedge\mathbf{B}]^1}{\mathbf{A}}\wedge E_l}{\mathbf{B}\wedge\mathbf{A}}\wedge I}{\mathbf{A}\wedge\mathbf{B}\Rightarrow\mathbf{B}\wedge\mathbf{A}}\Rightarrow I^1$$

$$\begin{array}{c}[A]^1\\ [B]^2\end{array}\qquad \frac{\dfrac{A}{B\Rightarrow A}\Rightarrow I^2}{A\Rightarrow B\Rightarrow A}\Rightarrow I^1$$

Here we see reasoning with local hypotheses at work. In the left example, we assume the formula $\mathbf{A} \wedge \mathbf{B}$ and can use it in the proof until it is discharged by the rule $\wedge E_l$ on the bottom – therefore we decorate the hypothesis and the rule by corresponding numbers (here the label "1"). Note the assumption $\mathbf{A} \wedge \mathbf{B}$ is *local to the proof fragment* delineated by the corresponding hypothesis and the discharging rule, i.e. even if this proof is only a fragment of a larger proof, then we cannot use its hypothesis anywhere else. Note also that we can use as many copies of the local hypothesis as we need; they are all discharged at the same time.

In the right example we see that local hypotheses can be nested as long as hypotheses are kept local. In particular, we may not use the hypothesis $\mathbf{B}$ after the $\Rightarrow I^2$, e.g. to continue with a $\Rightarrow E$.

One of the nice things about the natural deduction calculus is that the deduction theorem is almost trivial to prove. In a sense, the triviality of the deduction theorem is the central idea of the calculus and the feature that makes it so natural.

A Deduction Theorem for $\mathcal{ND}^0$

▷ **Theorem 5.2.3** $\mathcal{H}, \mathbf{A} \vdash_{\mathcal{ND}^0} \mathbf{B}$, *iff* $\mathcal{H} \vdash_{\mathcal{ND}^0} \mathbf{A} \Rightarrow \mathbf{B}$.

▷ Proof: We show the two directions separately

P.1 If $\mathcal{H}, \mathbf{A} \vdash_{\mathcal{ND}^0} \mathbf{B}$, then $\mathcal{H} \vdash_{\mathcal{ND}^0} \mathbf{A} \Rightarrow \mathbf{B}$ by $\Rightarrow I$, and

P.2 If $\mathcal{H} \vdash_{\mathcal{ND}^0} \mathbf{A} \Rightarrow \mathbf{B}$, then $\mathcal{H}, \mathcal{A} \vdash_{\mathcal{ND}^0} \mathbf{A} \Rightarrow \mathbf{B}$ by weakening and $\mathcal{H}, \mathcal{A} \vdash_{\mathcal{ND}^0} \mathbf{B}$ by $\Rightarrow E$. □

Another characteristic of the natural deduction calculus is that it has inference rules (introduction and elimination rules) for all connectives. So we extend the set of rules from Definition 5.2.1 for disjunction, negation and falsity.

More Rules for Natural Deduction

▷ **Definition 5.2.4** $\mathcal{ND}^0$ has the following additional rules for the remaining connectives.

$$\frac{\mathbf{A}}{\mathbf{A} \vee \mathbf{B}} \vee I_l \qquad \frac{\mathbf{B}}{\mathbf{A} \vee \mathbf{B}} \vee I_r \qquad \frac{\mathbf{A} \vee \mathbf{B} \quad \begin{array}{c}[\mathbf{A}]^1 \\ \vdots \\ \mathbf{C}\end{array} \quad \begin{array}{c}[\mathbf{B}]^1 \\ \vdots \\ \mathbf{C}\end{array}}{\mathbf{C}} \vee E^1$$

$$\frac{\begin{array}{c}[\mathbf{A}]^1 \\ \vdots \\ F\end{array}}{\neg \mathbf{A}} \neg I^1 \qquad \frac{\neg \neg \mathbf{A}}{\mathbf{A}} \neg E$$

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

Natural Deduction in Sequent Calculus Formulation

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

▷ **Definition 5.2.5** A judgment is a meta-statement about the provability of propositions

▷ **Definition 5.2.6** 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.

Write $\vdash \mathbf{A}$ for $\emptyset \vdash \mathbf{A}$.

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

▷ **Example 5.2.7** We give the sequent-style version of Example 5.2.2

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{A}\wedge\mathbf{B}\vdash\mathbf{A}\wedge\mathbf{B}}\,\mathrm{Ax}}{\mathbf{A}\wedge\mathbf{B}\vdash\mathbf{B}}\,\wedge E_r \quad \dfrac{\dfrac{}{\mathbf{A}\wedge\mathbf{B}\vdash\mathbf{A}\wedge\mathbf{B}}\,\mathrm{Ax}}{\mathbf{A}\wedge\mathbf{B}\vdash\mathbf{A}}\,\wedge E_l}{\mathbf{A}\wedge\mathbf{B}\vdash\mathbf{B}\wedge\mathbf{A}}\,\wedge I}{\vdash\mathbf{A}\wedge\mathbf{B}\Rightarrow\mathbf{B}\wedge\mathbf{A}}\,\Rightarrow I
\qquad
\dfrac{\dfrac{\dfrac{}{\mathbf{A},\mathbf{B}\vdash\mathbf{A}}\,\mathrm{Ax}}{\mathbf{A}\vdash\mathbf{B}\Rightarrow\mathbf{A}}\,\Rightarrow I}{\vdash\mathbf{A}\Rightarrow\mathbf{B}\Rightarrow\mathbf{A}}\,\Rightarrow I$$

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 5.2.8** The following inference rules make up the propositional sequent-style natural deduction calculus $\mathcal{ND}^0_\vdash$:

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

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

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

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

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

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

Linearized Notation for (Sequent-Style) ND Proofs

▷ Linearized notation for sequent-style ND proofs

1. $\mathcal{H}_1 \vdash \mathbf{A}_1 \quad (\mathcal{J}_1)$
2. $\mathcal{H}_2 \vdash \mathbf{A}_2 \quad (\mathcal{J}_2)$
3. $\mathcal{H}_3 \vdash \mathbf{A}_3 \quad (\mathcal{R}1, 2)$

corresponds to $$\frac{\mathcal{H}_1 \vdash \mathbf{A}_1 \quad \mathcal{H}_2 \vdash \mathbf{A}_2}{\mathcal{H}_3 \vdash \mathbf{A}_3}\mathcal{R}$$

▷ **Example 5.2.9** We show a linearized version of Example 5.2.7

#	*hyp*	$\vdash$	*formula*	*NDjust*
1.	1	$\vdash$	$\mathbf{A} \wedge \mathbf{B}$	Ax
2.	1	$\vdash$	$\mathbf{B}$	$\wedge E_r 1$
3.	1	$\vdash$	$\mathbf{A}$	$\wedge E_l 1$
4.	1	$\vdash$	$\mathbf{B} \wedge \mathbf{A}$	$\wedge I 2, 1$
5.		$\vdash$	$\mathbf{A} \wedge \mathbf{B} \Rightarrow \mathbf{B} \wedge \mathbf{A}$	$\Rightarrow I 4$

#	*hyp*	$\vdash$	*formula*	*NDjust*
1.	1	$\vdash$	$\mathbf{A}$	Ax
2.	2	$\vdash$	$\mathbf{B}$	Ax
3.	1, 2	$\vdash$	$\mathbf{A}$	weaken 1, 2
4.	1	$\vdash$	$\mathbf{B} \Rightarrow \mathbf{A}$	$\Rightarrow I 3$
5.		$\vdash$	$\mathbf{A} \Rightarrow \mathbf{B} \Rightarrow \mathbf{A}$	$\Rightarrow I 4$

Each row in the table represents one inference step in the proof. It consists of line number (for referencing), a formula for the asserted property, a justification via a ND rules (and the rows this one is derived from), and finally a list of row numbers of proof stepsx0 that are local hypotheses in effect for the current row.

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

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

$$\frac{[\mathbf{B}/X](\mathbf{A})}{\exists X.\mathbf{A}}\exists I \qquad \frac{\exists X.\mathbf{A} \quad \begin{array}{c}[[c/X](\mathbf{A})]^1\\ \vdots\\ \mathbf{C}\end{array} \quad c \in \Sigma_0^{sk} \text{ new}}{\mathbf{C}}\exists E^1$$

* 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.

A Complex $\mathcal{ND}^1$ Example

▷ **Example 5.2.11** We prove $\neg(\forall X.P(X)) \vdash_{\mathcal{ND}^1} \exists X.\neg P(X)$.

$$\dfrac{\dfrac{\dfrac{\neg(\forall X.P(X)) \qquad \dfrac{\dfrac{\dfrac{\dfrac{[\neg(\exists X.\neg P(X))]^1 \qquad \dfrac{[\neg P(X)]^2}{\exists X.\neg P(X)}\exists I}{F}FI}{\neg\neg P(X)}\neg I^2}{P(X)}\neg E}{\forall X.P(X)}\forall I}{F}FI}{\neg\neg(\exists X.\neg P(X))}\neg I^1}{\exists X.\neg P(X)}\neg E$$

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.

First-Order Natural Deduction in Sequent Formulation

▷ Rules for propositional connectives just as always

▷ **Definition 5.2.12 (New Quantifier Rules)**

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

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

 86

Natural Deduction with Equality

▷ **Definition 5.2.13 (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 5.2.14** 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, we will use the following rules in $\mathcal{ND}^1$

▷ **Definition 5.2.15** $\Leftrightarrow I$ is derivable and $\Leftrightarrow E$ is admissible in $\mathcal{ND}^1$:

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

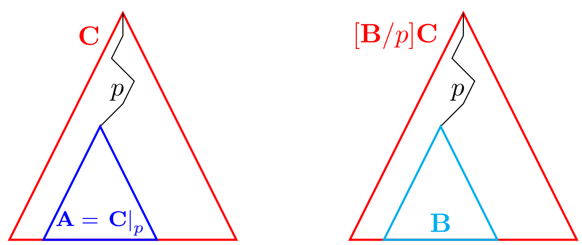
 87

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

To make sure that we understand the constructions here, let us get back to the "replacement at position" operation used in the equality rules.

Positions in Formulae

- ▷ Idea: Formulae are (naturally) trees, so we can use tree positions to talk about subformulae
- ▷ **Definition 5.2.16** A formula position p is a list of natural number that in each node of a formula (tree) specifies into which child to descend. For a formula $\mathbf{A}$ we denote the subformula at p with $\mathbf{A}|_p$.
- ▷ We will sometimes write a formula $\mathbf{C}$ as $\mathbf{C}[\mathbf{A}]_p$ to indicate that $\mathbf{C}$ the subformula $\mathbf{A}$ at position p.
- ▷ **Definition 5.2.17** Let p be a position, then $[\mathbf{A}/p]\mathbf{C}$ is the formula obtained from $\mathbf{C}$ by replacing the subformula at position p by $\mathbf{A}$.
- ▷ **Example 5.2.18 (Schematically)**



The operation of replacing a subformula at position p is quite different from e.g. (first-order) substitutions:

- We are replacing subformulae with subformulae instead of instantiating variables with terms.
- substitutions replace all occurrences of a variable in a formula, whereas formula replacement only affects the (one) subformula at position p.

We conclude this Subsection with an extended example: the proof of a classical mathematical result in the natural deduction calculus with equality. This shows us that we can derive strong properties about complex situations (here the real numbers; an uncountably infinite set of numbers).

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

- ▷ We can do real Maths with $\mathcal{ND}^1_=$:
- ▷ **Theorem 5.2.19** $\sqrt{2}$ *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 $2\ q^2 = p^2$.

  P.4 But $2\ q^2$ has an odd number of prime factors while p^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 PL^1 syntax and come up with justifications for them in terms of $\mathcal{ND}^1$ inference rules. The next two slides show such a proof, where we write $/n$ to denote that n is prime, use $\#(n)$ for the number of prime factors of a number n, and write $\mathrm{irr}(r)$ if r is irrational.

$\mathcal{ND}^1_=$ 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		$\forall n,p.\mathrm{prime}(p)\Rightarrow\#(p\,n)=\#(n)+1$	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	$\neg\,\mathrm{irr}(\sqrt{2})$	Ax
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	$\sqrt{2}=p/q$	Ax
10	6,9	$2\,q^2=p^2$	arith(9)
11	6,9	$\#(p^2)=2\,\#(p)$	$\forall E^2(2)$
12	6,9	$\mathrm{prime}(2)\Rightarrow\#(2\,q^2)=\#(q^2)+1$	$\forall E^2(1)$

Lines 6 and 9 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.

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

13		$\mathrm{prime}(2)$	lemma
14	6,9	$\#(2\,q^2)=\#(q^2)+1$	$\Rightarrow E(13,12)$
15	6,9	$\#(q^2)=2\,\#(q)$	$\forall E^2(2)$
16	6,9	$\#(2\,q^2)=2\,\#(q)+1$	$=E(14,15)$
17		$\#(p^2)=\#(p^2)$	$=I$
18	6,9	$\#(2\,q^2)=\#(q^2)$	$=E(17,10)$
19	6.9	$2\,\#(q)+1=\#(p^2)$	$=E(18,16)$
20	6.9	$2\,\#(q)+1=2\,\#(p)$	$=E(19,11)$
21	6.9	$\neg(2\,\#(q)+1)=(2\,\#(p))$	$\forall E^2(1)$
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 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 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.

Chapter 6

Higher-Order Logic and λ-Calculus

In this Chapter 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 Chapter. Once we have that we can reconstruct a clean version of higher-order logic by adding special provisions for propositions.

6.1 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 ($\mathrm{PL}\Omega$)

- ▷ 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 6.1.1** Given a set $\mathcal{BT}$ 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 $\mathcal{BT}$ under function types the set of types over $\mathcal{BT}$.
- **Definition 6.1.2** 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 $(\mathbf{C}\mathbf{A}) \in wff_\beta(\Sigma, \mathcal{V}_\mathcal{T})$
  - 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 6.1.3** 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 $\mathcal{D}_{\alpha\to\beta} = \mathcal{D}_\alpha \to \mathcal{D}_\beta$

  interpretation of constants: typed mapping $\mathcal{I}\colon \Sigma \to \mathcal{D}$ (i.e. $\mathcal{I}(\Sigma_\alpha) \subseteq \mathcal{D}_\alpha$)
- **Definition 6.1.4** 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 6.1.5** value function: typed mapping $\mathcal{I}_\varphi\colon wff_\mathcal{T}(\Sigma, \mathcal{V}_\mathcal{T}) \to \mathcal{D}$

- ▷ $\mathcal{I}_\varphi|_{\mathcal{V}_\mathcal{T}} = \varphi$ $\quad\mathcal{I}_\varphi|_{\Sigma_\mathcal{T}} = \mathcal{I}$
- ▷ $\mathcal{I}_\varphi(\mathbf{A}\mathbf{B}) = \mathcal{I}_\varphi(\mathbf{A})(\mathcal{I}_\varphi(\mathbf{B}))$
- ▷ $\mathcal{I}_\varphi(\forall X_\alpha.\mathbf{A}) = \mathsf{T}$, 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

▷ **Definition 6.1.6 (Leibniz equality)** $\mathbf{Q}^\alpha \mathbf{A}_\alpha \mathbf{B}_\alpha = \forall P_{\alpha\to o}.P\mathbf{A} \Leftrightarrow P\mathbf{B}$ (indiscernability)

▷ Note: $\forall P_{\alpha\to o}.P\mathbf{A} \Rightarrow P\mathbf{B}$ (get the other direction by instantiating P with Q, where $QX \Leftrightarrow (\neg PX)$)

▷ **Theorem 6.1.7** *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 6.1.8** We write $\mathbf{A} = \mathbf{B}$ for $\mathbf{QAB}$($\mathbf{A}$ 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 $r = \{\mathcal{I}_\varphi(\mathbf{A})\} \in \mathcal{D}_{\alpha\to o}$ (exists in a standard model)

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 6.1.9 (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 6.1.10 (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 6.1.10 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 6.1.11 ($\mathcal{H}_\Omega$ 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 6.1.12 ($\mathcal{H}_\Omega$ 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 6.1.13** *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)

▷ **Example 6.1.14** Valid sentences that are not $\mathcal{H}_\Omega$-theorems:

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

$\mathbf{Ext}^{\alpha\,\beta}$ $\quad\forall F_{\alpha\to\beta}.\forall G_{\alpha\to\beta}.(\forall X_\alpha.FX = GX)\Rightarrow F = G$

$\mathbf{Ext}^{o}$ $\quad\forall F_o.\forall G_o.(F\Leftrightarrow G)\Leftrightarrow F = G$

▷ 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 calculuIi:

▷ Idea: generalize so that the carrier only contains those functions that are requested by the comprehension axioms.

▷ **Theorem 6.1.15 (Henkin 1950)** $\mathcal{H}_\Omega$ *is complete wrt. this semantics.*

▷ Proof Sketch: more models ⇝ less valid sentences (these are $\mathcal{H}_\Omega$-theorems) □

▷ Henkin-models induce sensible measure of completeness for higher-order logic.

6.2 A better Form of Comprehension and Extensionality

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 6.2.1** 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 6.2.2** 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 6.2.3** We say that $\mathbf{A}$ and $\lambda X_\alpha.\mathbf{A}X$ are η-equal, (write $\mathbf{A}_{\alpha\to\beta} =_\eta (\lambda X_\alpha.\mathbf{A}X)$), iff $X \notin \mathrm{free}(\mathbf{A})$.
- **Theorem 6.2.4** *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.

6.3 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_α^{Sk} of Skolem contants).

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

▷ **Definition 6.3.1** 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 6.3.2** 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 6.3.3 (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: $(\lambda X.\mathbf{A}) =_\alpha (\lambda Y.[Y/X](\mathbf{A}))$.
- ▷ β conversion: $(\lambda X.\mathbf{A})\mathbf{B} =_\beta [\mathbf{B}/X](\mathbf{A})$.
- ▷ η conversion: $(\lambda X.\mathbf{A}X) =_\eta \mathbf{A}$ if $X \notin \mathrm{free}(\mathbf{A})$.

The intuitions about functional structure of λ-terms and about free and bound variables are encoded into three transformation rules $\Lambda^{\rightarrow}$: 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 $\Lambda^{\rightarrow}$ more palatable. This makes $\Lambda^{\rightarrow}$ 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 6.3.4 (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. $\mathbf{A.BC}$ abbreviates $\mathbf{A}(\mathbf{BC})$.
- ▷ **Notation 6.3.5 (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 6.3.6 (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 $\mathbf{B}$.[2] EdN:2

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 Section 5.1 by introducing substitutions with a substitutability condition like the one in Definition 5.1.24, then establishing the soundness of α conversion, and only then postulating defining capture-avoiding substitution application as in Definition 5.1.29. The development for $\Lambda^{\rightarrow}$ 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 $\Lambda^{\rightarrow}$ as we will see below.

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

$\alpha\beta\eta$-Equality (Overview)

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

▷ **Theorem 6.3.7** *$\beta\eta$-reduction is well-typed, terminating and confluent in the presence of $=_\alpha$-conversion.*

▷ **Definition 6.3.8 (Normal Form)** We call a λ-term $\mathbf{A}$ a normal form (in a reduction system $\mathcal{E}$), iff no rule (from $\mathcal{E}$) can be applied to $\mathbf{A}$.

▷ **Corollary 6.3.9** *$\beta\eta$-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 6.3.10 (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 6.3.11** Head Reduction always has a unique β redex

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

▷ **Theorem 6.3.12** *The syntactic heads of β-normal forms are constant or variables.*

▷ **Definition 6.3.13** 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 6.3.14** We call a λ-term a η-long form, iff its matrix has base type.

▷ **Definition 6.3.15** η-Expansion makes η-long forms

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

▷ **Definition 6.3.16** 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 $\Lambda^{\rightarrow}$ separately and thus reduce argumentational complexity.

A Test Generator for Higher-Order Unification

▷ **Definition 6.3.17 (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 6.3.18** *Subtraction and (integer) division on Church numberals can be automted via higher-order unification.*

▷ **Example 6.3.19** $5 - 2$ 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.

Excursion: We will discuss the computational properties in the appendix and the semantics in the appendix as well. Together they show that the simply typed λ calculus is an adequate logic for modeling (the equality) of functions and their applications.

6.4 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 6.4.1** A grammar for the raw terms of the simply typed λ-

calculus:

$$\begin{array}{lcl} \alpha & :== & c \mid \alpha \to \alpha \\ \Sigma & :== & \cdot \mid \Sigma, [c : \mathrm{type}] \mid \Sigma, [c : \alpha] \\ \Gamma & :== & \cdot \mid \Gamma, [x : \alpha] \\ \mathbf{A} & :== & c \mid X \mid \mathbf{A}^1\mathbf{A}^2 \mid \lambda X_\alpha.\mathbf{A} \end{array}$$

▷ 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 6.4.2** Judgments make statements about complex properties of the syntactic entities defined by the grammar.

▷ **Definition 6.4.3** Judgments for the simply typed λ-calculus

$\vdash \Sigma : \mathrm{sig}$	Σ is a well-formed signature
$\Sigma \vdash \alpha : \mathrm{type}$	α is a well-formed type given the type assumptions in Σ
$\Sigma \vdash \Gamma : \mathrm{ctx}$	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

▷ $\mathbf{A} \in \mathit{wff}_\alpha(\Sigma, \mathcal{V}_\mathcal{T})$, iff $\Gamma \vdash_\Sigma \mathbf{A} : \alpha$ derivable in

$$\frac{\Sigma \vdash \Gamma : \mathrm{ctx} \quad \Gamma(X) = \alpha}{\Gamma \vdash_\Sigma X : \alpha}\mathsf{wff{:}var} \qquad \frac{\Sigma \vdash \Gamma : \mathrm{ctx} \quad \Sigma(c) = \alpha}{\Gamma \vdash_\Sigma c : \alpha}\mathsf{wff{:}const}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} : \beta \to \alpha \quad \Gamma \vdash_\Sigma \mathbf{B} : \beta}{\Gamma \vdash_\Sigma \mathbf{AB} : \alpha}\mathsf{wff{:}app} \qquad \frac{\Gamma, [X : \beta] \vdash_\Sigma \mathbf{A} : \alpha}{\Gamma \vdash_\Sigma \lambda X_\beta.\mathbf{A} : \beta \to \alpha}\mathsf{wff{:}abs}$$

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

$$\frac{}{\vdash \cdot : \mathrm{sig}}\mathsf{sig{:}empty} \qquad \frac{\vdash \Sigma : \mathrm{sig}}{\vdash \Sigma, [\alpha : \mathrm{type}] : \mathrm{sig}}\mathsf{sig{:}type}$$

$$\frac{\vdash \Sigma : \mathrm{sig} \quad \Sigma \vdash \alpha : \mathrm{type}}{\vdash \Sigma, [c : \alpha] : \mathrm{sig}}\mathsf{sig{:}const}$$

$$\frac{\Sigma \vdash \alpha : \mathrm{type} \quad \Sigma \vdash \beta : \mathrm{type}}{\Sigma \vdash \alpha \to \beta : \mathrm{type}}\mathsf{typ{:}fn} \qquad \frac{\vdash \Sigma : \mathrm{sig} \quad \Sigma(\alpha) = \mathrm{type}}{\Sigma \vdash \alpha : \mathrm{type}}\mathsf{typ{:}start}$$

$$\frac{\vdash \Sigma : \mathrm{sig}}{\Sigma \vdash \cdot : \mathrm{ctx}}\mathsf{ctx{:}empty} \qquad \frac{\Sigma \vdash \Gamma : \mathrm{ctx} \quad \Sigma \vdash \alpha : \mathrm{type}}{\Sigma \vdash \Gamma, [X : \alpha] : \mathrm{ctx}}\mathsf{ctx{:}var}$$

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 $\mathcal{B}$

$$\frac{\vdash \cdot : \mathrm{sig}}{\vdash [\alpha : \mathrm{type}] : \mathrm{sig}}\,\mathsf{sig{:}type} \qquad \frac{\mathcal{A} \quad [\alpha : \mathrm{type}](\alpha) = \mathrm{type}}{[\alpha : \mathrm{type}] \vdash \alpha : \mathrm{type}}\,\mathsf{typ{:}start}$$

and with these we can construct the derivation $\mathcal{C}$

$$\frac{\mathcal{A} \quad \dfrac{\mathcal{B} \quad \dfrac{\mathcal{B} \quad \mathcal{B}}{[\alpha : \mathrm{type}] \vdash \alpha \to \alpha : \mathrm{type}}\,\mathsf{typ{:}fn}}{[\alpha : \mathrm{type}] \vdash \alpha \to \alpha \to \alpha : \mathrm{type}}\,\mathsf{typ{:}fn}}{\vdash \Sigma : \mathrm{sig}}\,\mathsf{sig{:}const}$$

Example: A Well-Formed λ-Term

▷ using Σ from above, we can show that $\Gamma := [X : \alpha]$ is a well-formed context:

$$\frac{\dfrac{\mathcal{C}}{\Sigma \vdash \cdot : \mathrm{ctx}}\,\mathsf{ctx{:}empty} \quad \dfrac{\mathcal{C} \quad \Sigma(\alpha) = \mathrm{type}}{\Sigma \vdash \alpha : \mathrm{type}}\,\mathsf{typ{:}start}}{\Sigma \vdash \Gamma : \mathrm{ctx}}\,\mathsf{ctx{:}var}$$

We call this derivation $\mathcal{G}$ and use it to show that

▷ $\lambda X_\alpha . f X X$ is well-typed and has type $\alpha \to \alpha$ in Σ. This is witnessed by the type derivation

$$\frac{\dfrac{\dfrac{\dfrac{\mathcal{C} \quad \Sigma(f) = \alpha \to \alpha \to \alpha}{\Gamma \vdash_\Sigma f : \alpha \to \alpha \to \alpha}\,\mathsf{wff{:}const} \quad \dfrac{\mathcal{G}}{\Gamma \vdash_\Sigma X : \alpha}\,\mathsf{wff{:}var}}{\Gamma \vdash_\Sigma f X : \alpha \to \alpha}\,\mathsf{wff{:}app} \quad \dfrac{\mathcal{G}}{\Gamma \vdash_\Sigma X : \alpha}\,\mathsf{wff{:}var}}{\Gamma \vdash_\Sigma f X X : \alpha}\,\mathsf{wff{:}app}}{\cdot \vdash_\Sigma \lambda X_\alpha . f X X : \alpha \to \alpha}\,\mathsf{wff{:}abs}$$

$\beta\eta$-Equality by Inference Rules: One-Step Reduction

- One-step Reduction ($+ \in \{\alpha, \beta, \eta\}$)

$$\frac{\Gamma, [X:\alpha] \vdash_\Sigma \mathbf{A} \colon \alpha \quad \Gamma \vdash_\Sigma \mathbf{B} \colon \beta}{\Gamma \vdash_\Sigma (\lambda X.\mathbf{A})\mathbf{B} \to^1_\beta [\mathbf{B}/X](\mathbf{A})}\text{wff}\beta\text{:top}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \colon \beta \to \alpha \quad X \notin \mathbf{dom}(\Gamma)}{\Gamma \vdash_\Sigma \lambda X.\mathbf{A}X \to^1_\eta \mathbf{A}}\text{wff}\eta\text{:top}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \to^1_+ \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{AC} \colon \alpha}{\Gamma \vdash_\Sigma \mathbf{AC} \to^1_+ \mathbf{BC}}\text{tr:app}fn$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \to^1_+ \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{CA} \colon \alpha}{\Gamma \vdash_\Sigma \mathbf{CA} \to^1_+ \mathbf{CB}}\text{tr:app}arg$$

$$\frac{\Gamma, [X:\alpha] \vdash_\Sigma \mathbf{A} \to^1_+ \mathbf{B}}{\Gamma \vdash_\Sigma \lambda X.\mathbf{A} \to^1_+ \lambda X.\mathbf{B}}\text{tr:abs}$$

 115

$\beta\eta$-Equality by Inference Rules: Multi-Step Reduction

- Multi-Step-Reduction ($+ \in \{\alpha, \beta, \eta\}$)

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \to^1_+ \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{A} \to^*_+ \mathbf{B}}\text{ms:start} \qquad \frac{\Gamma \vdash_\Sigma \mathbf{A} \colon \alpha}{\Gamma \vdash_\Sigma \mathbf{A} \to^*_+ \mathbf{A}}\text{ms:ref}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \to^*_+ \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{B} \to^*_+ \mathbf{C}}{\Gamma \vdash_\Sigma \mathbf{A} \to^*_+ \mathbf{C}}\text{ms:trans}$$

- Congruence Relation

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} \to^*_+ \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{B}}\text{eq:start}$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{B} =_+ \mathbf{A}}\text{eq:sym} \qquad \frac{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{B} =_+ \mathbf{C}}{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{C}}\text{eq:trans}$$

 116

6.5 De Bruijn Indices

We now come to a very neat – and by now classical – trick that allows us to solve the problem that we often want to consider alphabetical variants of formulae as "identical". Using the de Bruijn indices we introduce in this Section we can actually do that, as a consequence this technique is often used for implementing formal languages with binding operators.

The λ calculus is where the technique originates and the most natural setting in which to explain the idea.

De Bruijn Indices: Nameless Dummies for Bound Variables

- Problem: We consider alphabetically equal λ terms as "syntactically equal".
- Idea: Get rid of variables by replacing them with nameless dummies (numbers).
- **Definition 6.5.1 (Formally)** Raw λ-terms with de Bruijn indices are expressions given by the following production: in Definition 6.4.1.

$$\mathbf{A} \quad ::== \quad c \mid n \mid \mathbf{A}^1\mathbf{A}^2 \mid \lambda\mathbf{A}$$

  A variable n is bound if it is in the scope of at least n binders (λ); otherwise it is free. The binding site for a variable n is the nth binder it is in the scope of, starting from the innermost binder.
- **Example 6.5.2** $(\lambda x.\lambda y.zx(\lambda u.ux))(\lambda w.wx)$, becomes $(\lambda\lambda 42(\lambda 13))(\lambda 51)$,
- Problem: De Bruijn indices are less readable than standard λ terms.
- Solution: Maintain a UI with names even when using de Bruijn indices internally.
- Problem: Substitution and β-reduction become complicated. (see below)

De Bruijn Indices: β-Reduction

- **Definition 6.5.3** For β-reducing $(\lambda\mathbf{M})\mathbf{N}$ we must:
  1. find variable occurrences $n_1, n_2, \ldots, n_k$ in $\mathbf{M}$ bound by outer λ in $\lambda\mathbf{M}$
  2. decrement the free variables of $\mathbf{M}$ to match the removal of the outer λ,
  3. replace n_i with $\mathbf{N}$, suitably incrementing the free variables in $\mathbf{N}$ each time, to match the number of λ-binders, under which n_i occurs.
- **Example 6.5.4** We perform the steps outlined above on $(\lambda\lambda 42(\lambda 13))(\lambda 51)$:
  1. we obtain $\lambda 4n_1(\lambda 1n_2)$
  2. we obtain $\lambda 3n_1(\lambda 1n_2)$ decrementing free variables.
  3. we replace X with the argument $\lambda 51$.
     - n_1 is under one $\lambda \leadsto$ replace it with $\lambda 61$
     - n_2 is under two λs $\leadsto$ replace it with $\lambda 71$.

  The final result is $\lambda 3(\lambda 61)(\lambda 1(\lambda 71))$

6.6 Simple Type Theory

In this Section we will revisit the higher-order predicate logic introduced in Section 6.1 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 6.6.1** 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 $\mathrm{PL}\Omega$ (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 6.6.2** We assume logical constants Π^α and Σ^α of type $(\alpha \to o) \to o$.

  Regain quantifiers as abbreviations:

  $$(\forall X_\alpha.\mathbf{A}) := \Pi^\alpha(\lambda X_\alpha.\mathbf{A}) \qquad (\exists X_\alpha.\mathbf{A}) := \Sigma^\alpha(\lambda X_\alpha.\mathbf{A})$$

- **Definition 6.6.3** We must fix the semantics of logical constants:
  1. $\mathcal{I}(\Pi^\alpha)(p) = \mathsf{T}$, iff $p(a) = \mathsf{T}$ for all $a \in \mathcal{D}_\alpha$ (i.e. if p is the universal set)
  2. $\mathcal{I}(\Sigma^\alpha)(p) = \mathsf{T}$, iff $p(a) = \mathsf{T}$ for some $a \in \mathcal{D}_\alpha$ (i.e. iff p is non-empty)
- With this, we re-obtain the semantics we have given for quantifiers above:

  $$\mathcal{I}_\varphi(\forall X_\iota.\mathbf{A}) = \mathcal{I}_\varphi(\Pi^\iota(\lambda X_\iota.\mathbf{A})) = \mathcal{I}(\Pi^\iota)(\mathcal{I}_\varphi(\lambda X_\iota.\mathbf{A})) = \mathsf{T}$$

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: $\mathrm{HOL}^=$

- 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	$\lambda X_o.\lambda Y_o.(\lambda G_{o\to o\to o}.G\,T\,T = \lambda G_{o\to o\to o}.GXY)$
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	$q^o F$
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 6.1.7 does not hold (see [And72] for details). Indeed the proof of Theorem 6.1.7 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 $\mathrm{HOL}^\to$

- **Definition 6.6.4** 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 6.6.5** 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 6.6.6** 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 6.6.7** 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 6.6.8 (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.

 122 FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

▷ 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 $\iota^o := =(=T)$
- ▷ $\iota^{\alpha\to\beta} := \lambda H_{(\alpha\to\beta)\to o}X_\alpha.\iota^\beta(\lambda Z_\beta.(\exists F_{\alpha\to\beta}.(HF) \wedge (FX) = Z))$

 123 FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Chapter 7

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 Chapter 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 Chapter is delightfully covered in [Dox+09].

7.1 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 7.1.1** A set is "everything that can form a unity in the face of God". (Georg Cantor (∗1845, †1918))

▷ **Example 7.1.2** (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

- 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.	$\vdots$	$\vdots$

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 7.1.3 (The set of all set and friends)**
  $\{M \mid M \text{ set}\}$, $\{M \mid M \text{ set}, M \in M\}$, . . .
- **Definition 7.1.4 (Problem)** Russell's Antinomy:

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

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

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 $\mathcal{M} \not\in \mathcal{M}$.

▷ 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.

7.2 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	$\forall N.\exists M.\forall Z.(Z \in M) \Leftrightarrow (Z \in N \land \mathbf{E})$ From a given set N we can separate all members described by expression $\mathbf{E}$.	

▷ **Theorem 7.2.1** $\forall M,N.(M \subseteq N) \land (N \subseteq M) \Rightarrow M = N$

▷ **Theorem 7.2.2** M *is uniquely determined in* **Sep**

▷ Proof Sketch: With **Ext** □

▷ **Notation 7.2.3** 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 7.2.4** *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 7.2.5** 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 – $\cup$**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, $\bigcup$**Ax** makes $\cup$**Ax** redundant, but minimality of the axiom system is not a concern for us currently.

The Axioms for Set Union

- **Axiom 7.2.6 (Small Union Axiom ($\cup$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 7.2.7** $M \cup N := \{X \in W \mid X \in M \vee X \in N\}$ (exists by **Sep.**)
- **Axiom 7.2.8 (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 7.2.9** $\bigcup(M) := \{X \mid \exists Y.Y \in M \wedge X \in Y\}$ (exists by **Sep.**)
- This also gives us intersections over families (without another axiom):
- **Definition 7.2.10**

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

In Definition 7.2.10 we note that $\bigcup$**Ax** also guarantees us intersection over families. Note that we could not have defined that in analogy to Definition 7.2.5 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 7.2.11 (Power Set Axiom)** For each set M there is a set W that contains all subsets of M: $\wp\mathrm{Ax} := (\forall M.\exists W.\forall X.(X \subseteq M) \Rightarrow X \in W)$
- **Definition 7.2.12** Power Set: $\mathcal{P}(M) := \{X \mid X \subseteq M\}$ (Exists by **Sep.**)
- **Definition 7.2.13** singleton set: $\{X\} := \{Y \in \mathcal{P}(X) \mid X = Y\}$
- **Axiom 7.2.14 (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 $\wp\mathrm{Ax}$: $\{M, N\} := \{M\} \cup \{N\}$.
- **Definition 7.2.15 (Finite Sets)** $\{X, Y, Z\} := \{X, Y\} \cup \{Z\} \ldots$
- **Theorem 7.2.16** $\forall Z, X_1, \ldots, X_n.(Z \in \{X_1, \ldots, X_n\}) \Leftrightarrow (Z = X_1 \vee \ldots \vee Z = X_n)$

The Foundation Axiom

- **Axiom 7.2.17 (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 7.2.18** *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 7.2.19 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 7.2.20 (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 7.2.21** M is inductive: $\mathbf{Ind}(M) := \emptyset \in M \wedge (\forall Z.Z \in M \Rightarrow (Z \cup \{Z\}) \in M)$.
- **Definition 7.2.22** Set of the Inductive Set: $\omega := \{Z \mid \forall W.\mathbf{Ind}(W) \Rightarrow Z \in W\}$
- **Theorem 7.2.23** ω *is inductive.*

The Replacement Axiom

- We have ω, $\wp(M)$, but not $\{\omega, \wp(\omega), \wp(\wp(\omega)), \ldots\}$.
- **Axiom 7.2.24 (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 $\forall U.\exists V.V = \{F(X) \mid X \in U\}$.

▷ **Example 7.2.25** 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. $V = \{\{X\} \mid X \in U = \{\{1\}, \{\{2, 3\}\}\}\}$.

Zermelo Fraenkel Set Theory

▷ **Definition 7.2.26 (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 7.2.27 (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. $\leadsto$ we can "pick out" an arbitrary element.

▷ Reasons for **AC**:

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

▷ **Definition 7.2.28 (Zermelo Fraenkel Set Theory with Choice)** The theory **ZF** together with **AC** is called ZFC with choice and denoted as **ZFC**.

7.3 ZFC Applications

Limits of ZFC

▷ **Conjecture 7.3.1 (Cantor's Continuum Hypothesis (CH))** *There is no set whose cardinality is strictly between that of integers and real numbers.*

▷ **Theorem 7.3.2** *If* **ZFC** *is consistent, then neither* **CH** *nor* ¬**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 7.3.3** Define $\langle X,Y\rangle := \{\{X\},\{X,Y\}\}$

▷ **Lemma 7.3.4** $\langle X,Y\rangle = \langle U,V\rangle \Rightarrow X = U \wedge Y = V$

▷ **Lemma 7.3.5** $U \in X \wedge V \in Y \Rightarrow \langle U,V\rangle \in \mathcal{P}(\mathcal{P}(X \cup Y))$

▷ **Definition 7.3.6** 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 7.3.7** right projection π_r analogous.

Relations

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

▷ **Definition 7.3.8** 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 7.3.9** The domain and codomain of a function are defined as usual

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

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

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

Functions

- **Definition 7.3.10** A function f from X to Y is a right-unique relation with $\mathrm{Dom}(f) = X$ and $\mathrm{coDom}(f) = Y$; write $f\colon X \to Y$.
- **Definition 7.3.11** 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:
  - $\forall A.\exists R.R = \{\langle U, V\rangle \,|\, U \in A \wedge V \in A \wedge p(U \wedge V)\}$ for all predicates p.
  - $\forall A.\exists F.F = \{\langle X, f(X)\rangle \,|\, X \in A\}$ for all functions f.
- As the natural numbers can be epxressed in set theory, the logical calculus can be expressed by Gödelization.

Bibliography

[And02] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. second. Kluwer Academic Publishers, 2002.

[And72] Peter B. Andrews. "General Models and Extensionality". In: *Journal of Symbolic Logic* 37.2 (1972), pp. 395–397.

[Asp+06] Andrea Asperti et al. "A Content Based Mathematical Search Engine: Whelp". In: *Types for Proofs and Programs, International Workshop, TYPES 2004, revised selected papers*. Ed. by Jean-Christophe Filliâtre, Christine Paulin-Mohring, and Benjamin Werner. LNCS 3839. Springer Verlag, 2006, pp. 17–32.

[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)". In: *Mathematische Annalen* 46 (1895), pp. 481–512. DOI: `10.1007/bf02124929`.

[Can97] Georg Cantor. "Beiträge zur Begründung der transfiniten Mengenlehre (2)". In: *Mathematische Annalen* 49 (1897), pp. 207–246. DOI: `doi:10.1007/bf01444205`.

[Chu40] Alonzo Church. "A Formulation of the Simple Theory of Types". In: *Journal of Symbolic Logic* 5 (1940), pp. 56–68.

[Dox+09] A.K. Doxiadēs et al. *Logicomix: An Epic Search for Truth*. Bloomsbury, 2009. ISBN: 9780747597209.

[Fre79] Gottlob Frege. *Begriffsschrift: eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. 1879.

[Gen34] Gerhard Gentzen. "Untersuchungen über das logische Schließen I". In: *Mathematische Zeitschrift* 39.2 (1934), pp. 176–210.

[Göd31] Kurt Gödel. "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I". In: *Monatshefte der Mathematischen Physik* 38 (1931). English Version in [Hei67], pp. 173–198.

[Hei67] Jean van Heijenoort. *From Frege to Gödel: a source book in mathematical logic 1879-1931*. 3^{rd} printing, 1997. Source books in the history of the sciences series. Cambridge, MA: Harvard Univ. Press, 1967. ISBN: 0-674-32450-1.

[Hil26] David Hilbert. "Über das Unendliche". In: *Mathematische Annalen* 95 (1926), pp. 161–190. DOI: `10.1007/BF01206605`.

[Jin10] Arif Jinha. "Article 50 million: an estimate of the number of scholarly articles in existence". In: *Learned Publishing* 23.3 (2010), pp. 258–263. DOI: `10.1087/20100308`.

[KK06] Andrea Kohlhase and Michael Kohlhase. "Communities of Practice in MKM: An Extensional Model". In: *Mathematical Knowledge Management (MKM)*. Ed. by Jon Borwein and William M. Farmer. LNAI 4108. Springer Verlag, 2006, pp. 179–193. URL: `https://kwarc.info/kohlhase/papers/mkm06cp.pdf`.

[Koh08] Michael Kohlhase. "Using LaTeX as a Semantic Markup Format". In: *Mathematics in Computer Science* 2.2 (2008), pp. 279–304. URL: https://kwarc.info/kohlhase/papers/mcs08-stex.pdf.

[Koh20] Michael Kohlhase. *sTeX: Semantic Markup in TEX/LaTeX*. Tech. rep. Comprehensive TEX Archive Network (CTAN), 2020. URL: http://www.ctan.org/get/macros/latex/contrib/stex/sty/stex.pdf.

[LI10] Peder Olesen Larsen and Markus von Ins. "The rate of growth in scientific publication and the decline in coverage provided by Science Citation Index". In: *Scientometrics* 84.3 (2010), pp. 575–603. DOI: 10.1007/s11192-010-0202-z.

[LM06] Paul Libbrecht and Erica Melis. "Methods for Access and Retrieval of Mathematical Content in ActiveMath". In: *Proceedings of ICMS-2006*. Ed. by N. Takayama and A. Iglesias. LNAI 4151. http://www.activemath.org/publications/Libbrecht-Melis-Access-and-Retrieval-ActiveMath-ICMS-2006.pdf. Springer Verlag, 2006, pp. 331–342. URL: http://www.activemath.org/publications/Libbrecht-Melis-Access-and-Retrieval-ActiveMath-ICMS-2006.pdf.

[MG11] Jozef Misutka and Leo Galambos. "System Description: EgoMath2 As a Tool for Mathematical Searching on Wikipedia.org". In: *Intelligent Computer Mathematics*. Ed. by James Davenport et al. LNAI 6824. Springer Verlag, 2011, pp. 307–309. ISBN: 978-3-642-22672-4.

[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*. Seattle, Washington, USA: ACM Press, 2006, pp. 735–735. ISBN: 1-59593-369-7. DOI: http://doi.acm.org/10.1145/1148170.1148348.

[MY03] Bruce R. Miller and Abdou Youssef. "Technical Aspects of the Digital Library of Mathematical Functions". In: *Annals of Mathematics and Artificial Intelligence* 38.1-3 (2003), pp. 121–136. URL: citeseer.ist.psu.edu/599441.html.

[OMT] Michael Kohlhase and Dennis Müller. *OMDoc/MMT Tutorial for Mathematicians*. URL: https://gl.mathhub.info/Tutorials/Mathematicians/blob/master/tutorial/mmt-math-tutorial.pdf (visited on 10/07/2017).

[WR10] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica*. 2nd ed. Vol. I. Cambridge, UK: Cambridge University Press, 1910.

[Zer08] Ernst Zermelo. "Untersuchungen über die Grundlagen der Mengenlehre. I." In: *Mathematische Annalen* 65 (1908), pp. 261–281.

Index