Knowledge Representation for Science, Technology, Engineering, and Mathematics Summer Semester 2022

– 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

2023-04-25

0.0.1 Preface

0.0.1.1 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.

0.0.1.2 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 mathematical knowledge and not about the theoretical aspects of the logics themselves, we give the discussion about these as a "suggested readings" section. This material can safely be skipped (thus it is in the appendix).

0.0.1.3 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.

Licensing:

Knowledge Representation Experiment:

This document is also an experiment in knowledge representation. Under the hood, it uses the sTeX package [Koh08; sTeX], 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`.

0.0.1.4 Acknowledgments

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

0.0.2 Recorded Syllabus for SS 2022

In this subsection, we record the progress of the course in the summer semester 2022 in the form of a "recorded syllabus", i.e. a syllabus that is created after the fact rather than before. **Recorded Syllabus Summer Semester 2022:**

#	date	what	until	slide	page
1.	April 18.	Lecture	admin, ALeA, some overview	11	6
2	April 19.	Lab	Setup		

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-SS21.pdf`. **Recorded Syllabus Summer Semester 2020:**

#	date	what	until	slide	page
1.	April 22.	Lecture	admin, some overview, OBB		
2.	April 23.	Lecture	Theory Graphs Intro		
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		
8.	May14.	Lab	More Arithmetics		
9.	May 20.	Lecture	Logic Ideas		
	May 21.		Ascension		
10.	May 27.	Lecture	FOL, subsitutibility		
11.	May 28	Lecture	Higher-Order Logic and λ-calculus		
12.	June 3.	Lecture	λ-calculus via Judgments/Inference		
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		
21.	July 2.	Lab	HOL & Computation/Description		
22.	July 8.	Lecture/Lab	Set Theory, ZFC		

0.1 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 for KRMT

- **Content Prerequisites:** the mandatory courses in CS@FAU; Sem 1-4, in particular:
  - course "Grundlagen der Logik in der Informatik" (GLOIN)
  - CS Math courses "Mathematik C1-4" (IngMath1-4) (our "domain")
  - algorithms and data structures
  - AI-1 ("Artificial Intelligence I") (nice-to-have only)
- **Intuition:** (take them with a kilo of salt)
  - This is what I assume you know! (I have to assume something)
  - In many cases, the dependency of KRMT on these is partial and "in spirit".
  - If you have not taken these (or do not remember),
    - read up on them as needed! (preferred, do it in a group)
    - We can cover them in class (if there are more of you)
- **The real Prerequisite:** Motivation, Interest, Curiosity, hard work. (KRMT is non-trivial)
- You can do this course if you want! (We will help you)

KRMT Lab (Dogfooding our own Techniques)

- **Underlying Problem:** There are about 20 deep results/insights/tricks necessary to understand KRMT.
- **The Good News:** These are sufficient too, if you can apply them (non-trivial)
- **Consequence:** KRMT may be the course with the highest "pain-per-letter ratio" (but it is wonderful when the pain goes away)
- **General Plan:** We use the Wednesday slot to get our hands dirty with actual MMT formalizations.
- **Goal:** Reinforce what was taught on Tuesdays and have some fun.
- **How this works:** we jointly develop key formalizations in class
  - we discuss the pertinent issues, you dictate, we test in the system.
  - what is left over becomes homework (the routine parts)
  - we discuss problems, ... on the KRMT chat (details later)

▷ **Caveat:** Only by practical involvement will you be able to understand the difficult theoretical issues/ideas! (so come and participate)

Homeworks

▷ **Goal:** Homework assignments/problems reinforce what was taught in Lectures/Labs

▷ **Homeworks** will be small individual formalization tasks (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.
- ▷ We can be flexible about deadlines (but deadlines help you)

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

Grades (Academic Assessment)

▷ **What we used so far:** two parts (Portfolio Assessment)

- ▷ 20-30 min oral exam at the end of the semester (50%)
- ▷ results of the KRMT lab (50%)

This will not work with 50+ students, need to see how the course develops!

▷ **How about this:** three parts (Portfolio Assessment)

- ▷ 60 min written exam early October? (70%)
- ▷ results of the KRMT lab (30%)
- ▷ bonus project after the semester (10% bonus)

▷ If you have suggestions, I will probably be happy with that as well.

▷ Let's finalize this next week.

Actually, I do not really care what the grading scheme is, and so it is open to discussion. For all I care we would not have grades at all; but students need them to graduate. Generally, I would like to spend as little time as possible on the grades admin, to the extent that I can give grades without going to jail or blushing too much.

Textbook, Handouts and Information, Forums, Chat

- **(No) Textbook:** there is none!
  - Course notes will be posted at http://kwarc.info/teaching/KRMT
  - We mostly prepare/update them as we go along (semantically preloaded ⇝ research resource)
  - Please e-mail us any errors/shortcomings you notice. (improve for the group)
- The KRMT lab generally follows the MMT tutorial at https://gl.mathhub.info/Tutorials/Mathematicians/blob/master/tutorial/mmt-math-tutorial.pdf
- Announcements will be posted on the course forum
  - https://www.studon.fau.de/frm5126852.html
- Check the forum frequently for (adopt/use it, this is for you!)
  - announcements, homeworks, questions
  - discussion among your fellow students
- We have to choose a chat venue (Matrix or StudOn)

Do I need to attend the lectures

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

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

Next we come to a special project that is going on in parallel to teaching the course. I am using the 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: Learning Support with KWARC Technologies

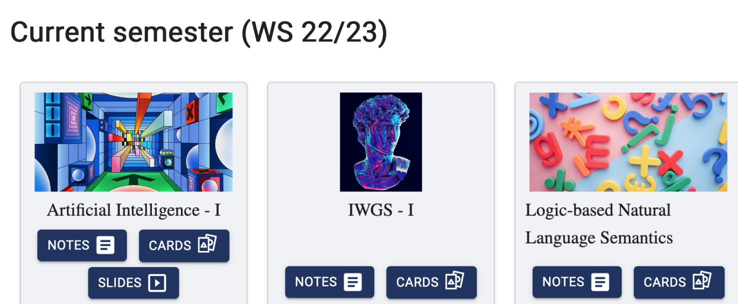
- **My research area:** Deep representation formats for (mathematical) knowledge
- **One Application:** Learning support systems(represent knowledge to transport it)
- **Experiment:** Start with this course (Drink my own medicine)

1. Re-Represent the slide materials in OMDoc (Open Mathematical Documents)
2. Feed it into the ALeA system (`http://courses.voll-ki.fau.de`)
3. Try it on you all (to get feedback from you)

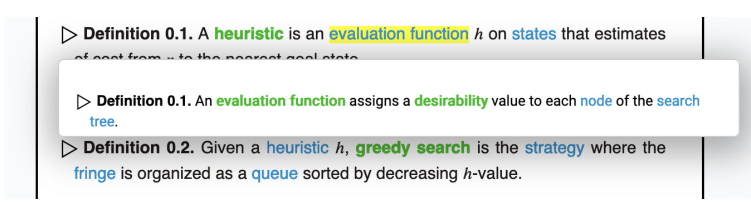
- Tasks (I cannot pay you for this)
  - help me complete the material on the slides (what is missing/would help?)
  - I need to remember "what I say", examples on the board. (take notes)
- Benefits for you (so why should you help?)
  - you will be mentioned in the acknowledgements (for all that is worth)
  - you will help build better course materials (think of next-year's students)

VoLL-KI Portal at `https://courses.voll-ki.fau.de`

- **Idea:** Provide HTML versions of the slides/notes and embed learning support services into them. (for pre/postparation of lectures)

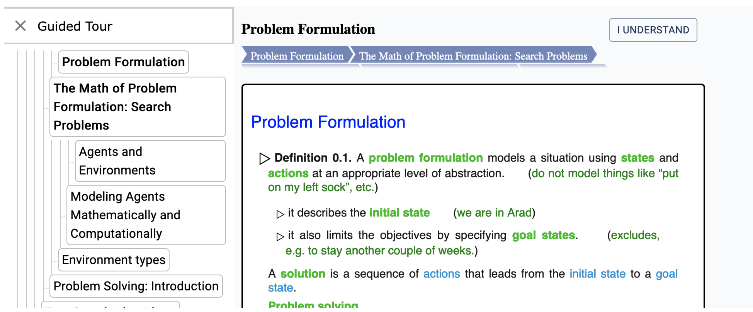


- **Definition 0.1.1.** Call a document active, iff it is interactive and adapts to specific information needs of the readers. (course notes on steroids)
- **Example 0.1.2 (Definition on Hover).** When we hover on a (cyan) term reference, hovering shows us the definition. (even works recursively)



When we click on the hover popup, we get even more information!

▷ **Example 0.1.3 (Guided Tour).** A guided tour for a concept c assembles definitions/etc. into a self-contained mini-course culminating at c.



▷ **Status:** The ALeA system is deployed at FAU for over 1000 students taking six courses

- ▷ (some) students use the system actively (our logs tell us)
- ▷ reviews are mostly positive/enthusiastic (error reports pour in)

Let us briefly look into how the learning support services introduced above might work, focusing on where the necessar information might come from.

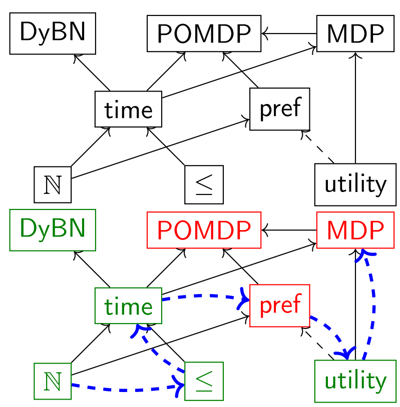
ALeA≙ Data-Driven & AI-enabled Learning Assistance

▷ **Ingredient 1:** Domain model ≙ knowledge/theory graph

▷ **Ingredient 2:** Learner model ≙ adding competency estimations

▷ **Ingredient 3:** A collection of ready-formulated learning objects

▷ **Ingredient 4:** Educational dialogue planner ⇝ guided tours



A theory graph provides (modular representation of the domain)

- ▷ symbols with URIs for all concepts, objects, and relations
- ▷ definitions, notations, and verbalizations for all symbols
- ▷ "object-oriented inheritance" and views between theories.

The learner model is a function from learner IDs × symbol URIs to competency values

- ▷ competency comes in six cognitive dimensions: `remember`, `understand`, `analyze`, `evaluate`, `apply`, and `create`.

- ▷ ALeA logs all learner interactions (keeps data learner-private)
- ▷ each interaction updates the learner model function.

Learning objects are the text fragments learners see and interact with; they are structured by

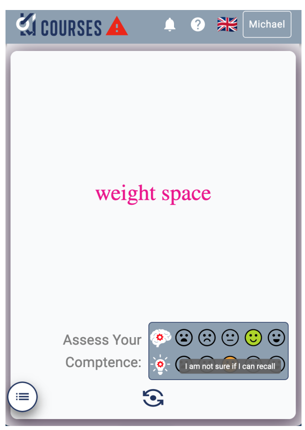
- ▷ didactic relations, e.g. tasks have prerequisites and learning objectives
- ▷ rhetoric relations, e.g. introduction, elaboration, and transition

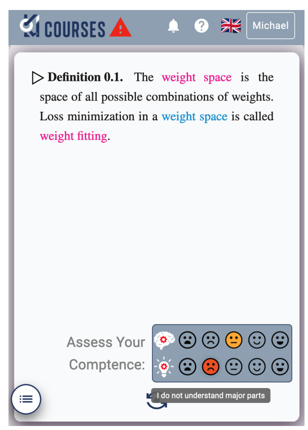
The dialogue planner assembles learning objects into active course materials using

- ▷ the domain model and didactic relations to determine the order of LOs
- ▷ the learner model to determine what to show
- ▷ the rhetoric relations to make the dialogue coherent

New Feature: Drilling with Flashcards

- ▷ Flashcards challenge you with a task (term/problem) on the front...





...and the definition/answer is on the back.

- ▷ Self-assessment updates the learner model (before/after)
- ▷ **Bonus:** Flashcards can be generated from existing semantic markup (educational equivalent to free beer)

0.2 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

0.2.1 Introduction & Motivation

To get a feeling for the issues discussed in the KRMT course, let us try to understand the words in the title of the course. We start with the concept of knowledge representation.

Knowledge-Representation and -Processing

▷ **Definition 0.2.1 (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/is assumed.

▷ **Definition 0.2.2.** Knowledge representation formulates knowledge in a formal language so that new knowledge can be induced by inferred via rule systems (inference).

▷ **Definition 0.2.3.** We call an information system knowledge based, if a large part of its behaviour is based on inference on represented knowledge.

▷ **Definition 0.2.4.** 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.

When one does research on such a lofty thing as knowledge representation and processing, then is is good to have a firm grounding in a domain in which we can study the phenomena "in the wild". The KWARC research group at FAU has chosen Mathematics.

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, use their intuitions to guide them through only partially understood situations, and can do meta-judgments (moral, practical, . . .)
- ▷ **In mathematics:** we exploit this, we
  - ▷ let humans explore mathematical theories and come up with novel insights/proofs,
  - ▷ delegate symbolic/numeric computation and typesetting of documents to computers.
  - ▷ (sometimes) delegate proof checking and search for trivial proofs to computers
- ▷ **Overlooked Opportunity:** management of existing mathematical knowledge
  - ▷ cataloguing, retrieval, refactoring, plausibilization, change propagation and in some cases even application do not require (human) insights and intuition
  - ▷ can even be automated in the near future given suitable representation formats and algorithms.
- ▷ **Math. Knowledge Management (MKM):** is the discipline that studies this.
- ▷ **Application:** Scaling Math beyond the One-Brain-Barrier

The One-Brain-Barrier

- ▷ **Observation 0.2.5.** *More than 10^5 math articles published annually in Math.*

- **Observation 0.2.6.** *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 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)

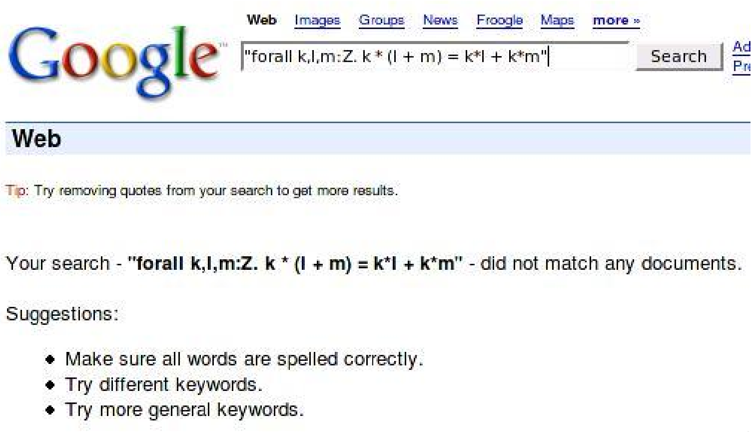
0.2.2 Mathematical Formula Search

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.

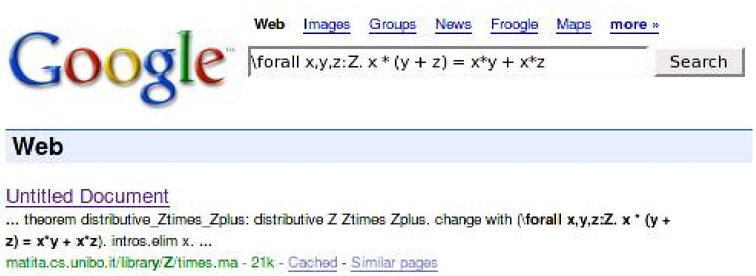
More Mathematics on the Web

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

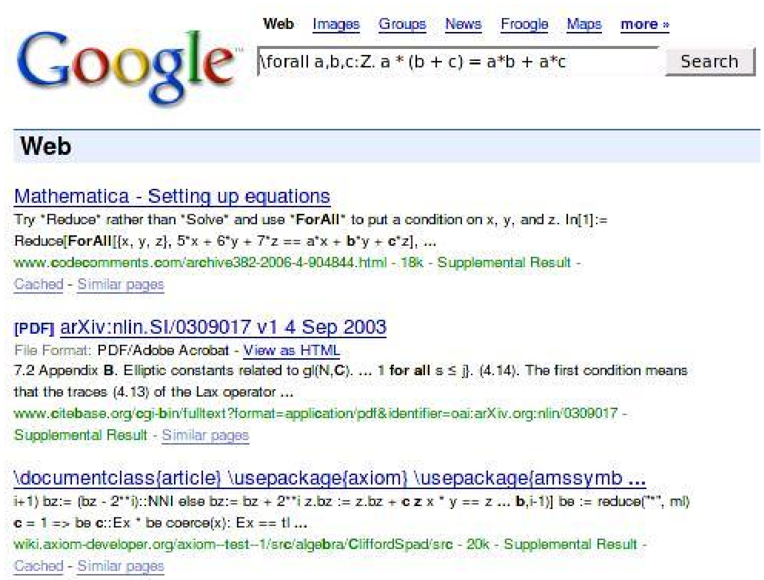
Searching for Distributivity



Searching for Distributivity

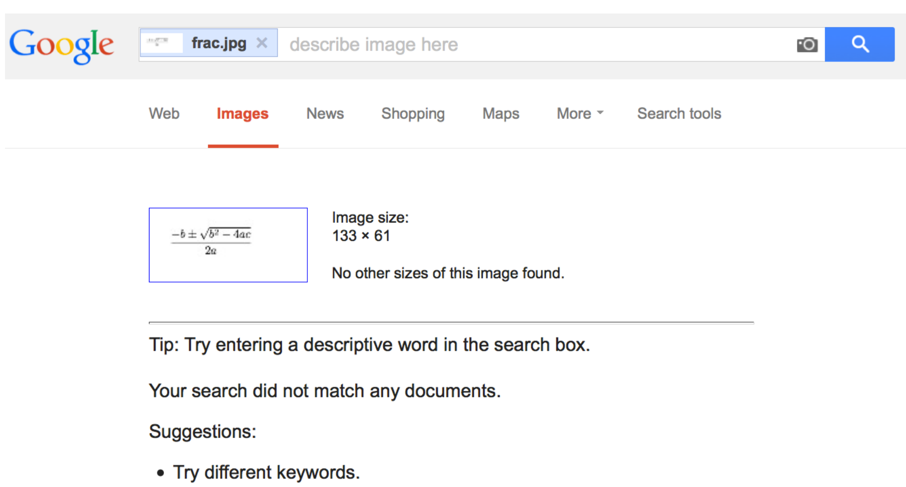


Searching for Distributivity



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

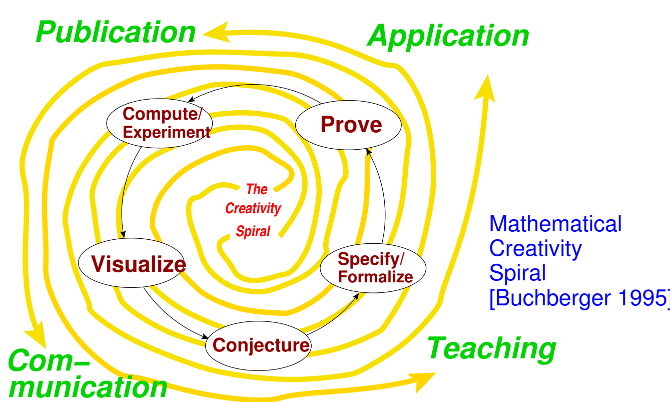
MathWebSearch: Search Math. Formulae on the Web

- ▷ **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

0.2.3 The Mathematical Knowledge Space

The way we do math will change dramatically

- ▷ **Definition 0.2.7 (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 0.2.8 (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 0.2.9.** We call the skillset that identifies mathematical training mathematical literacy (cf. ??)

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

▷ **Definition 0.2.11.** We are framing the point sets as algebraic varieties (sets of zeroes of polynomials).

▷ **Example 0.2.12 (Lie group).** Equipping a differentiable manifold with a (differentiable) group operation

▷ **Example 0.2.13 (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.

0.2.4 MMT: A Modular Framework for Representing Logics and Domains

We will use the OMDoc/MMT to represent both logical systems and the semantic domains (universes of discourse) of the various fragments. The MMT implements the OMDoc/MMT language, it can be used as

- a Java library that provides data structures and an API of logic oriented algorithms, and as

- a standalone knowledge-management service provider via web interfaces.

We will make use of both in the KRMT course and give a brief overview in this subsection. For a (math-themed) tutorial that introduces format and system in more detail see [OMT].

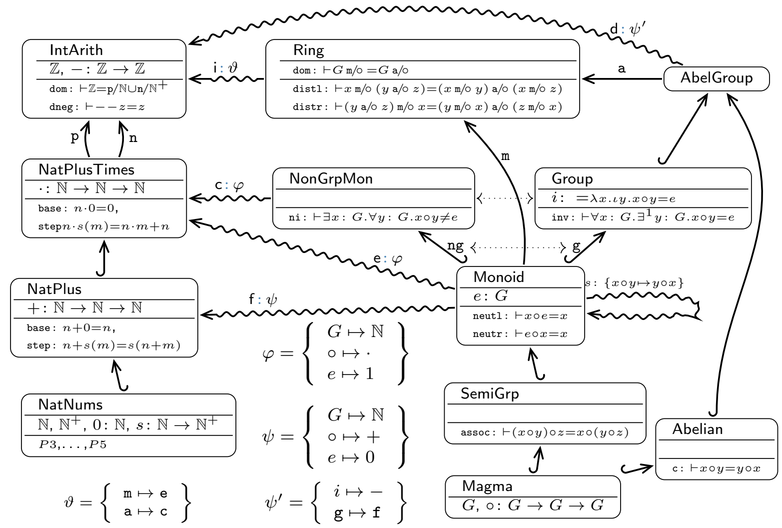
Representation language (MMT)

- **Definition 0.2.14.** MMT $\widehat{=}$ module system for mathematical theories
- Formal syntax and semantics
  - needed for mathematical interface language
  - but how to avoid foundational commitment?
- Foundation-independence
  - identify aspects of underlying language that are necessary for large scale processing
  - formalize exactly those, be parametric in the rest
  - observation: most large scale operations need the same aspects
- Module system
  - preserve mathematical structure wherever possible
  - formal semantics for modularity
- Web-scalable
  - build on XML, OpenMath, OMDoc
  - URI based logical identifiers for all declarations
- Implemented in the MMT API system.

The basic idea of the OMDoc/MMT format is that knowledge (originally mathematical knowledge for which the format is designed, but also world knowledge of the semantic domains in the fragments) can be represented modularly, using strong forms of inheritance to avoid duplicate formalization. This leads to the notion of a theory graph, where the nodes are theories that declare language fragments and axiomatize knowledge about the objects in the domain of discourse. The following theory graph is taken from [OMT].

Modular Representation of Math (MMT Example)

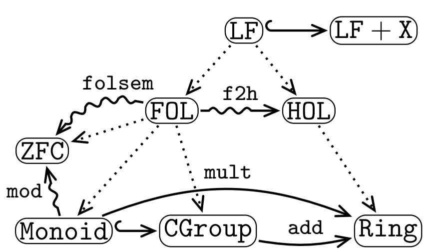
- **Example 0.2.15 (Elementary Algebra and Arithmetics).**



We will use the foundation-independence (bring-your-own logic) in this course, since the models for the different fragments come with differing logics and foundational theories (together referred to as "foundations"). Logics can be represented as theories in OMDoc/MMT – after all they just introduce language fragments and specify their behavior – and are subject to the same modularity and inheritance regime as domain theories. The only difference is that logics form the meta-language of the domain theories – they provide the language used to talk about the domain – and are thus connected to the domain theories by the meta relation. The next slide gives some details on the construction.

Representing Logics and Foundations as Theories

- ▷ **Example 0.2.16.** Logics and foundations represented as MMT theories



- ▷ **Definition 0.2.17.** Meta relation between theories special case of inclusion
- ▷ **Uniform Meaning Space:** morphisms between formalizations in different logics become possible via meta-morphisms.
- ▷ *Remark 0.2.18.* Semantics of logics as views into foundations, e.g., `folsem`.
- ▷ *Remark 0.2.19.* Models represented as views into foundations (e.g. ZFC)
- ▷ **Example 0.2.20.** `mod` $:= \{G \mapsto \mathbb{Z}, \circ \mapsto +, e \mapsto 0\}$ interprets Monoid in ZFC.

In the next slide we show the MMT surface language which gives a human-oriented syntax to the OMDoc/MMT format.

A MitM Theory in MMT Surface Language

- ▷ **Example 0.2.21.** A theory of Groups
  - ▷ Declaration $\widehat{=}$ name : type [= Def] [# notation]
  - ▷ Axioms $\widehat{=}$ Declaration with type $\vdash F$
  - ▷ ModelsOf makes a record type from a theory.

```
theory group : base:?Logic =
  theory group_theory : base:?Logic =
    include ?monoid/monoid_theory ❙

    inverse : U → U ❘ # 1 ⁻¹  prec 24 ❙
    inverseproperty : ⊦ ∀ [x] x ∘ x ⁻¹ ≐ e ❙
  ❚
  group = ModelsOf group_theory ❙
```

- ▷ **MitM Foundation:** optimized for natural math formulation
  - ▷ higher-order logic based on polymorphic λ-calculus
  - ▷ judgements-as-types paradigm: $\vdash F \widehat{=}$ type of proofs of F
  - ▷ dependent types with predicate subtyping, e.g. $\{\mathsf{n}\}\{'\mathsf{a} \in \mathsf{mat}(\mathsf{n},\mathsf{n})|\mathsf{symm}(\mathsf{a})'\}$
  - ▷ (dependent) record types for reflecting theories

Finally, we summarize the concepts and features of the OMDoc/MMT.

The MMT Module System

- ▷ **Central notion:** theory graph with theory nodes and theory morphisms as edges
- ▷ **Definition 0.2.22.** In MMT, a theory is a sequence of constant declarations optionally with type declarations and definitions
- ▷ MMT employs the Curry/Howard isomorphism and treats
  - ▷ axioms/conjectures as typed symbol declarations (propositions-as-types)
  - ▷ inference rules as function types (proof transformers)
  - ▷ theorems as definitions (proof terms for conjectures)
- ▷ **Definition 0.2.23.** MMT had two kinds of theory morphisms
  - ▷ structures instantiate theories in a new context (also called: definitional link, import)
    they import of theory S into theory T induces theory morphism $S \to T$
  - ▷ views translate between existing theories (also called: postulated link, theorem link)
    views transport theorems from source to target (framing).
- ▷ Together, structures and views allow a very high degree of re-use
- ▷ **Definition 0.2.24.** We call a statement t induced in a theory T, iff there is
  - ▷ a path of theory morphisms from a theory S to T with (joint) assignment σ,

▷ such that $t = \sigma(s)$ for some statement s in S.

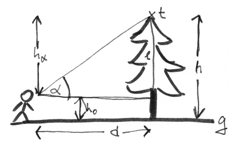
▷ **Definition 0.2.25.** In MMT, all induced statements have a canonical name, the MMT URI.

0.2.5 Application: Serious Games

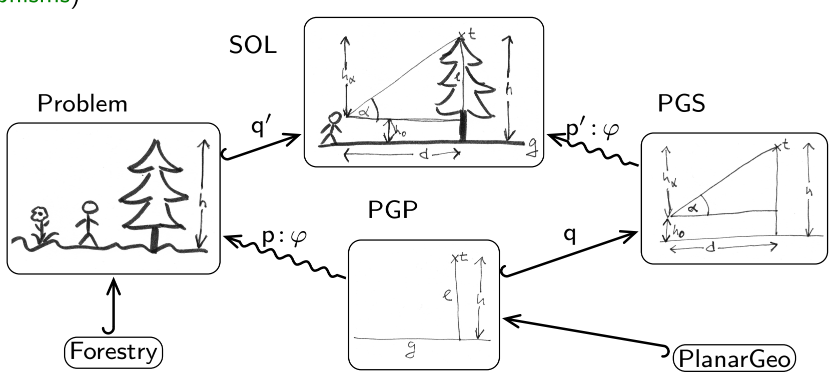
Framing for Problem Solving (The FrameIT Method)

▷ **Example 0.2.26 (Problem 0.8.15).**

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

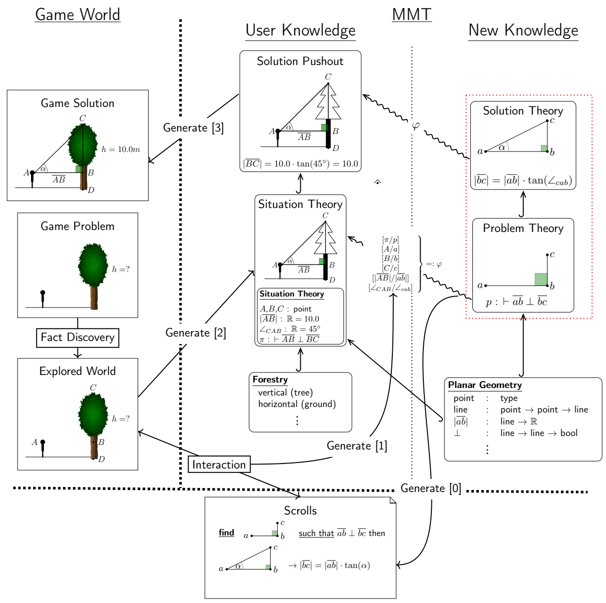


▷ Framing: view the problem as one that is already understood (using theory morphisms)



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

Example Learning Object Graph



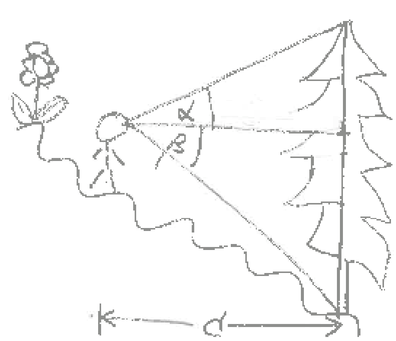
FrameIT Method: Problem

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

▷ 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

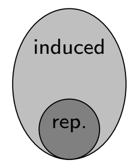
▷ create more complex P/S pairs (e.g. for trees on slopes)

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.

0.2.6 Search in the Mathematical Knowledge Space

The Mathematical Knowledge Space

- ▷ **Observation 0.2.27.** *The value of framing is that it induces new knowledge*
- ▷ **Definition 0.2.28.** 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

bsearch: Indexing flattened Theory Graphs

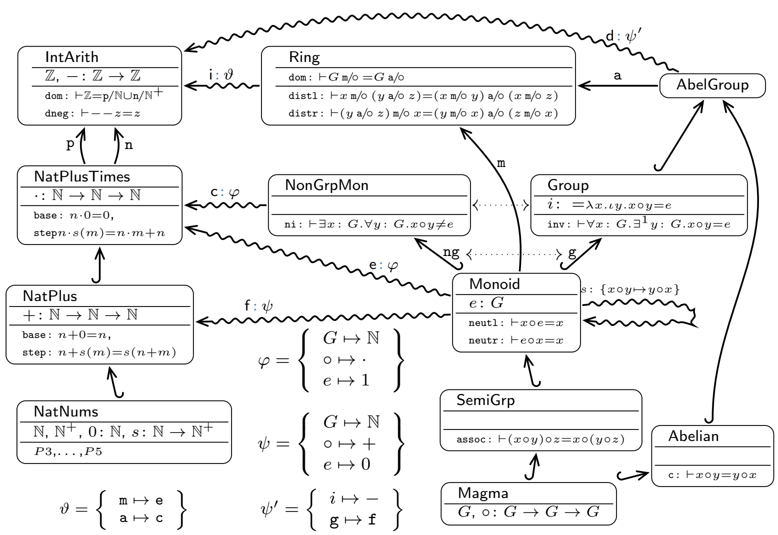
- ▷ **Simple Idea:** We have all the necessary components: MMT and `MathWebSearch`
- ▷ **Definition 0.2.29.** The bsearch systen is an integration of `MathWebSearch` and MMT that
  - ▷ computes the induced formulae of a modular mathematical library via MMT (aka. flattening)
  - ▷ indexes induced formulae 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)

bsearch User Interface: Explaining MMT URIs

- ▷ **Recall:** bsearch (MathWebSearch really) returns a MMT URI as a hit.
- ▷ **Question:** How to present that to the user? (for his/her greatest benefit)
- ▷ **Fortunately:** MMT system can compute induced statements (the hits)
- ▷ **Problem:** Hit statement may look considerably different from the induced statement
- ▷ **Solution:** Template-based generation of NL explanations from MMT URIs. MMT knows the necessary information from the components of the MMT URI.

Modular Representation of Math (MMT Example)

- ▷ **Example 0.2.30 (Elementary Algebra and Arithmetics).**



Example: Explaining a MMT URI

- ▷ **Example 0.2.31.** bsearch 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?IntArit` (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

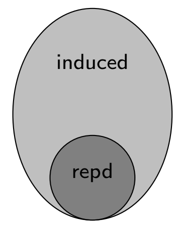
▷ ground the explanation by

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

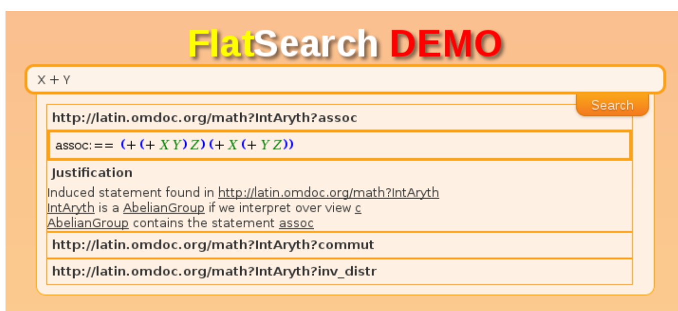
bsearch on the LATIN Logic Atlas

▷ 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 bsearch frontend at `http://cds.omdoc.org:8181/search.html`



Overview: KWARC Research and Projects

Applications: eMath 3.0, Active Documents, Active Learning, Semantic Spreadsheets/CAD/CAM, Change Mangagement, Global Digital Math Library, Math Search Systems, SMGloM: Semantic Multilingual Math Glossary, Serious Games, ...		
Foundations of Math: ▷ MathML, OpenMath ▷ advanced Type Theories ▷ MMT: Meta Meta Theory ▷ Logic Morphisms/Atlas ▷ Theorem Prover/CAS Interoperability ▷ Mathematical Models/Simulation	**KM & Interaction**: ▷ Semantic Interpretation (aka. Framing) ▷ math-literate interaction ▷ MathHub: math archives & active docs ▷ Active documents: embedded semantic services ▷ Model-based Education	**Semantization**: ▷ LaTeXML: LaTeX → XML ▷ sTeX: Semantic LaTeX ▷ invasive editors ▷ Context-Aware IDEs ▷ Mathematical Corpora ▷ Linguistics of Math ▷ ML for Math Semantics Extraction
Foundations: Computational Logic, Web Technologies, OMDoc/MMT		

Take-Home Message

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

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

0.3 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

▷ **Definition 0.3.1.** Computational Logic is the study of logic from a computational, proof-theoretic perspective. (model theory is mostly comprised under "mathematical logic".)

What is Logic?

▷ **Definition 0.3.2.** 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 ($x, y, f, g, p, 1, \pi, \in, \neg, \forall, \exists$)
- ▷ 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 $[\![\mathbf{A}]\!] = \mathsf{T}$ (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 (1623), (1642), and (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 connective) and a small calculus that gives us a set of rules how to manipulate formulae.

0.3.1 A History of Ideas in Logic

Before starting with the discussion on particular logical systemlogics and calculusinference 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 subsection 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 $\sim$ 1850]
  - \+ functional structure of formal language (propositions + connectives)
  - \+ mathematical semantics ($\rightsquigarrow$ Boolean Algebra)
  - – abstraction from internal structure of propositions

History of Ideas (continued): Predicate Logic

- ▷ Frege's "Begriffsschrift" [Fre79]
  - \+ functional structure of formal language (terms, atomic formulae, connectives, quantifiers)
  - – weird graphical syntax, no mathematical semantics
  - – paradoxes e.g. Russell's Paradox [R. 1901] (the set of sets that do not contain themselves)
- ▷ modern form of predicate logic [Peano $\sim$ 1889]
  - \+ modern notation for predicate logic ($\vee, \wedge, \Rightarrow, \forall, \exists$)

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

- ▷ Types ([Russell 1908])
  - – restriction to well-typed 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, semidecidable
  - + 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, and
  - ▷ whose consistency is provable in the system itself.
- ▷ Hilbert's Program is impossible! ([Gödel 1931]) Let $\mathcal{L}$ be a logical system that formalizes arithmetic ($\langle\mathbb{N}, +, *\rangle$),
  - ▷ 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 $\rightsquigarrow$ 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

Chapter 1

Foundations of Mathematics

1.1 Propositional Logic and Inference

1.1.1 Propositional Logic (Syntax/Semantics)

Propositional Logic (Syntax)

- **Definition 1.1.1 (Syntax).** The formulae of propositional logic (write PL^0) are made up from
  - propositional variables: $\mathcal{V}_0 := \{P, Q, R, P^1, P^2, \ldots\}$ (countably infinite)
  - constants/constructors called connectives: $\Sigma_0 := \{T, F, \neg, \vee, \wedge, \Rightarrow, \Leftrightarrow, \ldots\}$

  We define the set $\mathit{wff}_0(\mathcal{V}_0)$ of well-formed propositional formula (wffs) as
  - propositional variables,
  - the logical constants T and F,
  - negations $\neg\mathbf{A}$,
  - conjunctions $\mathbf{A} \wedge \mathbf{B}$($\mathbf{A}$ and $\mathbf{B}$ are called conjuncts),
  - disjunctions $\mathbf{A} \vee \mathbf{B}$ ($\mathbf{A}$ and $\mathbf{B}$ are called disjuncts),
  - implications $\mathbf{A} \Rightarrow \mathbf{B}$, and
  - equivalences (or biimplication). $\mathbf{A} \Leftrightarrow \mathbf{B}$,

  where $\mathbf{A}, \mathbf{B} \in \mathit{wff}_0(\mathcal{V}_0)$ themselves.
- **Example 1.1.2.** $P \wedge Q, P \vee Q, (\neg P \vee Q) \Leftrightarrow (P \Rightarrow Q) \in \mathit{wff}_0(\mathcal{V}_0)$
- **Definition 1.1.3.** Propositional formulae without connectives are called atomic (or an atom) and complex otherwise.

Alternative Notations for Connectives

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

Semantics of PL^0 (Models)

▷ **Definition 1.1.4.** 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.
- ▷ $\mathcal{I}(\neg)\colon\mathcal{D}_o\rightarrow\mathcal{D}_o; \mathsf{T}\mapsto\mathsf{F}, \mathsf{F}\mapsto\mathsf{T}$
- ▷ $\mathcal{I}(\wedge)\colon\mathcal{D}_o\times\mathcal{D}_o\rightarrow\mathcal{D}_o; \langle\alpha,\beta\rangle\mapsto\mathsf{T}$, 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)

Semantics of PL^0 (Evaluation)

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

▷ **Definition 1.1.5.** A variable assignment $\varphi\colon\mathcal{V}_0\rightarrow\mathcal{D}_o$ assigns values to propositional variables.

▷ **Definition 1.1.6.** The value function $\mathcal{I}_\varphi\colon wf\!f_0(\mathcal{V}_0)\rightarrow\mathcal{D}_o$ 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.

▷ **Definition 1.1.7.** Two formulae $\mathbf{A}$ and $\mathbf{B}$ are called equivalent, iff $\mathcal{I}_\varphi(\mathbf{A})=\mathcal{I}_\varphi(\mathbf{B})$ for all variable assignments φ.

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 1.1.8.** 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
  - 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 1.1.9.** $x \vee x$ is satisfiable and falsifiable.
- **Example 1.1.10.** $x \vee \neg x$ is valid and $x \wedge \neg x$ is unsatisfiable.
- **Alternative Notation:** Write $[\![\mathbf{A}]\!]_\varphi$ for $\mathcal{I}_\varphi(\mathbf{A})$, if $\mathcal{M} = \langle\mathcal{U},\mathcal{I}\rangle$. (and $[\![\mathbf{A}]\!]$, if $\mathbf{A}$ is ground, and $[\![\mathbf{A}]\!]$, if $\mathcal{M}$ is clear)
- **Definition 1.1.11 (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.

1.1.2 Calculi for Propositional Logic

[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.

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.
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 Relations and Inference Rules

- **Definition 1.1.12.** Let $\mathcal{L}:=\langle\mathcal{L},\mathcal{K},\models\rangle$ be a logical system, then we call a relation $\vdash\subseteq\mathcal{P}(\mathcal{L})\times\mathcal{L}$ a derivation relation for $\mathcal{L}$, if
  - $\mathcal{H}\vdash\mathbf{A}$, if $\mathbf{A}\in\mathcal{H}$ ($\vdash$ is proof reflexive),
  - $\mathcal{H}\vdash\mathbf{A}$ and $\mathcal{H}'\cup\{\mathbf{A}\}\vdash\mathbf{B}$ imply $\mathcal{H}\cup\mathcal{H}'\vdash\mathbf{B}$ ($\vdash$ is proof transitive),
  - $\mathcal{H}\vdash\mathbf{A}$ and $\mathcal{H}\subseteq\mathcal{H}'$ imply $\mathcal{H}'\vdash\mathbf{A}$ ($\vdash$ is monotonic or admits weakening).
- **Definition 1.1.13.** We call $\langle\mathcal{L},\mathcal{K},\models,\mathcal{C}\rangle$ a formal system, iff $\mathcal{L}:=\langle\mathcal{L},\mathcal{K},\models\rangle$ is a logical system, and $\mathcal{C}$ a calculus for $\mathcal{L}$.
- **Definition 1.1.14.**

  Let $\mathcal{L}$ be the formal language of a logical system, then an inference rule over $\mathcal{L}$ is a decidable $n+1$ ary relation on $\mathcal{L}$. Inference rules are traditionally written as

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

  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 of $\mathcal{N}$, and $\mathbf{C}$ is called its conclusion.
- **Definition 1.1.15.** An inference rule without assumptions is called an axiom.
- **Definition 1.1.16.** Let $\mathcal{L}:=\langle\mathcal{L},\mathcal{K},\models\rangle$ be a logical system, then we call a set $\mathcal{C}$ of inference rules over $\mathcal{L}$ a calculus (or inference system) for $\mathcal{L}$.

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

- **Definition 1.1.17.**Let $\mathcal{L}:=\langle\mathcal{L},\mathcal{K},\models\rangle$ be a logical system and $\mathcal{C}$ a calculus for $\mathcal{L}$, 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 $\frac{\mathbf{A}_{l_1} \quad \ldots \quad \mathbf{A}_{l_k}}{\mathbf{A}_i}$ in $\mathcal{C}$ with $l_j<i$ for all $j\leq k$. (rule

application)

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

▷ **Example 1.1.18.**

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 1.1.18 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 1.1.18.

Formal Systems

▷ Let $\langle\mathcal{L},\mathcal{K},\models\rangle$ be a logical system and $\mathcal{C}$ a calculus, then $\vdash_{\mathcal{C}}$ is a derivation relation and thus $\langle\mathcal{L},\mathcal{K},\models,\vdash_{\mathcal{C}}\rangle$ a derivation system.

▷ Therefore we will sometimes also call $\langle\mathcal{L},\mathcal{K},\models,\mathcal{C}\rangle$ a formal system, iff $\mathcal{L}{:=}\langle\mathcal{L},\mathcal{K},\models\rangle$ is a logical system, and $\mathcal{C}$ a calculus for $\mathcal{L}$.

▷ **Definition 1.1.19.** Let $\mathcal{C}$ be a calculus, then a $\mathcal{C}$-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 1.1.20. The act of finding a proof for a formula $\mathbf{A}$ is called proving $\mathbf{A}$.

▷ **Definition 1.1.21.**

An inference rule $\mathcal{I}$ is called admissible in a calculus $\mathcal{C}$, if the extension of $\mathcal{C}$ by $\mathcal{I}$ does not yield new theorems.

▷ **Definition 1.1.22.** An inference rule $\dfrac{\mathbf{A}_1\quad\ldots\quad\mathbf{A}_n}{\mathbf{C}}$ is called derivable (or a derived rule) in a calculus $\mathcal{C}$, if there is a $\mathcal{C}$ derivation $\mathbf{A}_1,\ldots,\mathbf{A}_n\vdash_{\mathcal{C}}\mathbf{C}$.

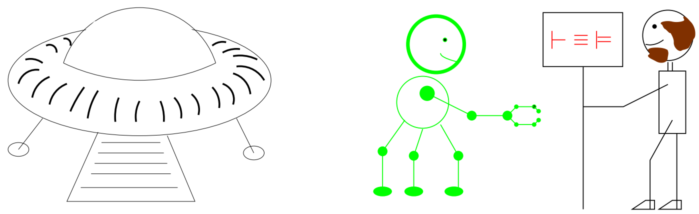
▷ **Observation 1.1.23.** *Derivable inference rules are admissible, but not the other way around.*

The notion of a formal system encapsulates the most general way we can conceptualize a system with a calculus, i.e. a system in which we can do "formal reasoning".

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 1.1.24.** Let $\mathcal{L}:=\langle\mathcal{L},\mathcal{K},\models\rangle$ be a logical system, then we call a calculus $\mathcal{C}$ for $\mathcal{L}$, iff
  - ▷ 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:** Find calculi C, such that $\vdash_{C}\mathbf{A}$ 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.

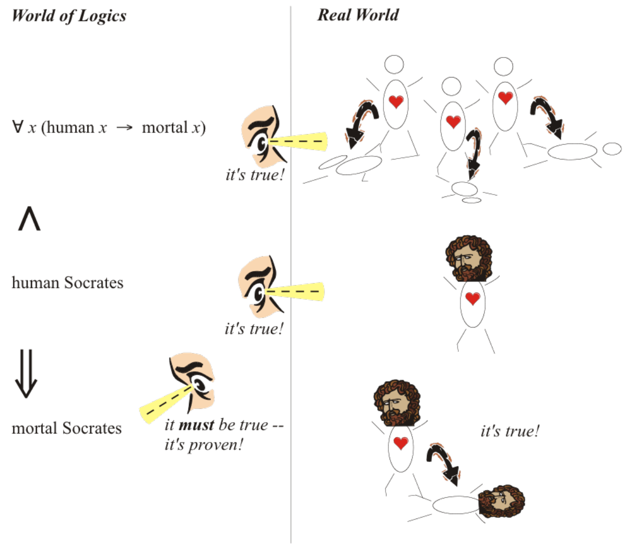
A consequence of this is that we can rely on purely syntactical means to make predictions about the world. Computers rely on formal representations of the world; if we want to solve a problem on our computer, we first represent it in the computer (as data structures, which can be seen as a formal language) 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!



Kohlhase & Rabe: KRMT 61 2023-04-25

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!)

1.1.3 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. In particular, it 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, we introduce a natural deduction calculus that provides two/three inference rules for every logical constant, 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 argumentation for theorem proving.

▷ **Definition 1.1.25.** The propositional natural deduction calculus $\mathcal{ND}_0$ has inference rules for the introduction and elimination of connectives:

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

$\Rightarrow I$ proves $\mathbf{A}\Rightarrow\mathbf{B}$ by exhibiting a $\mathcal{ND}_0$ derivation $\mathcal{D}$ (depicted by the double horizontal lines) of $\mathbf{B}$ from the local hypothesis $\mathbf{A}$; $\Rightarrow I$ then discharges (get rid of $\mathbf{A}$, which can only be used in $\mathcal{D}$) the hypothesis and concludes $\mathbf{A}\Rightarrow\mathbf{B}$. This mode of reasoning is called hypothetical reasoning.

▷ **Definition 1.1.26.**

Given a set $\mathcal{H}\subseteq \mathit{wff}_0(\mathcal{V}_0)$ of assumptions and a conclusion $\mathbf{C}$, we write $\mathcal{H}\vdash_{\mathcal{ND}_0}\mathbf{C}$, iff there is a $\mathcal{ND}_0$ derivation tree whose leaves are in $\mathcal{H}$.

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

The most characteristic rule in the natural deduction calculus is the $\Rightarrow I$ rule and the hypothetical reasoning it introduce. $\Rightarrow I$ 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 local hypothesis. When we can do this we discharge the assumption and conclude $\mathbf{A}\Rightarrow\mathbf{B}$.

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 1.1.27 (Inference with Local Hypotheses).**

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

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

Here we see hypothetical 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 local hypothesishypothesis and the discharging rule, i.e. even if this proof is only a fragment of a larger proof, then we cannot use its local hypothesishypothesis 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 they 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 1.1.28.** $\mathcal{H}, \mathbf{A} \vdash_{\mathcal{ND}_0} \mathbf{B}$, *iff* $\mathcal{H} \vdash_{\mathcal{ND}_0} \mathbf{A} \Rightarrow \mathbf{B}$.
- *Proof:* We show the two directions separately
  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
  2. If $\mathcal{H} \vdash_{\mathcal{ND}_0} \mathbf{A} \Rightarrow \mathbf{B}$, then $\mathcal{H}, \mathbf{A} \vdash_{\mathcal{ND}_0} \mathbf{A} \Rightarrow \mathbf{B}$ by weakening and $\mathcal{H}, \mathbf{A} \vdash_{\mathcal{ND}_0} \mathbf{B}$ 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 1.2.33 for disjunction, negation and falsity.

More Rules for Natural Deduction

- **Note:** $\mathcal{ND}_0$ does not try to be minimal, but comfortable to work in!x
- **Definition 1.1.29.** $\mathcal{ND}_0$ has the following additional inference rules for the remaining connectives.

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

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

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

- **Again:** $\neg E$ is used only in classical logic (otherwise constructive/intuitionistic)

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

Natural Deduction in Sequent Calculus Formulation

- **Idea:** Represent hypotheses explicitly. (lift calculus to judgments)
- **Definition 1.1.30.** A judgment is a meta statement about the provability of propositions.
- **Definition 1.1.31.** 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. We write $\vdash\mathbf{A}$ for $\emptyset\vdash\mathbf{A}$.
- **Idea:** Reformulate $\mathcal{ND}_0$ inference rules so that they act on sequents.
- **Example 1.1.32.** We give the sequent style version of Example 1.2.35:

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

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

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

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

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

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

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

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

Linearized Notation for (Sequent-Style) ND Proofs

▷ Linearized notation for sequent-style ND proofs

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

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 1.1.34.** We show a linearized version of the $\mathcal{ND}_0$ examples Example 1.2.40

#	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, 3
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

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 steps that are local hypotheses in effect for the current row.

1.2 First-Order Predicate Logic

1.2.1 First-Order Logic

First-order logic is the most widely used formal systems 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, few,...*)

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 subsection 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.

1.2.1.1 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.

PL¹ Syntax (Signature and Variables)

▷ **Definition 1.2.1.**

First-order logic (PL¹), is a formal system extensively used in mathematics, philosophy, linguistics, and computer science. It combines propositional logic with the ability to quantify over individuals.

▷ PL¹ talks about two kinds of objects: (so we have two kinds of symbols)

▷ truth values by reusing PL⁰

▷ individuals, e.g. numbers, foxes, Pokémon,...

▷ **Definition 1.2.2.** 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: $\Sigma_k^p = \{p, q, r, \ldots\}$ (relationships among individuals.)

▷ (Skolem constants: $\Sigma_k^{sk} = \{f_k^1, f_k^2, \ldots\}$) (witness constructors; countably ∞)

▷ We take Σ_ι to be all of these together: $\Sigma_\iota{:=}\Sigma^f \cup \Sigma^p \cup \Sigma^{sk}$, where $\Sigma^*{:=}\bigcup_{k\in\mathbb{N}}\Sigma_k^*$ and define $\Sigma{:=}\Sigma_\iota \cup \Sigma^o$.

▷ **Definition 1.2.3.** We assume a set of individual variables: $\mathcal{V}_\iota{:=}\{X, Y, Z, \ldots\}$. (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 1.2.4.** Terms: $\mathbf{A} \in wf\!f_\iota(\Sigma_\iota, \mathcal{V}_\iota)$ (denote individuals)
  - $\mathcal{V}_\iota \subseteq wf\!f_\iota(\Sigma_\iota, \mathcal{V}_\iota)$,
  - if $f \in \Sigma_k^f$ and $\mathbf{A}^i \in wf\!f_\iota(\Sigma_\iota, \mathcal{V}_\iota)$ for $i \leq k$, then $f(\mathbf{A}^1, \ldots, \mathbf{A}^k) \in wf\!f_\iota(\Sigma_\iota, \mathcal{V}_\iota)$.
- **Definition 1.2.5.** if Propositions: $\mathbf{A} \in wf\!f_o(\Sigma_\iota, \mathcal{V}_\iota)$: (denote truth values)
  - if $p \in \Sigma_k^p$ and $\mathbf{A}^i \in wf\!f_\iota(\Sigma_\iota, \mathcal{V}_\iota)$ for $i \leq k$, then $p(\mathbf{A}^1, \ldots, \mathbf{A}^k) \in wf\!f_o(\Sigma_\iota, \mathcal{V}_\iota)$,
  - if $\mathbf{A}, \mathbf{B} \in wf\!f_o(\Sigma_\iota, \mathcal{V}_\iota)$ and $X \in \mathcal{V}_\iota$, then $T, \mathbf{A} \wedge \mathbf{B}, \neg \mathbf{A}, \forall X.\mathbf{A} \in wf\!f_o(\Sigma_\iota, \mathcal{V}_\iota)$. $\forall$ is a binding operator called the universal quantifier.
- **Definition 1.2.6.** 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 1.2.7.** We use $\exists X.\mathbf{A}$ as an abbreviation for $\neg(\forall X.\neg \mathbf{A})$. $\exists$ is a binding operator called the existential quantifier.
- **Definition 1.2.8.** 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
$\forall x.\mathbf{A}$	$\bigwedge x.\mathbf{A}$ $\quad (x)\mathbf{A}$
$\exists x.\mathbf{A}$	$\bigvee x.\mathbf{A}$

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 1.2.9.** 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 1.2.10.** We define the set $\mathrm{free}(\mathbf{A})$ of frees variable of a formula $\mathbf{A}$:

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

▷ **Definition 1.2.11.** We call a formula $\mathbf{A}$ closed or ground, iff $\mathrm{free}(\mathbf{A})=\emptyset$. We call a closed proposition a sentence, and denote the set of all ground terms with $\mathit{cwff}_\iota(\Sigma_\iota)$ and the set of sentences with $\mathit{cwff}_o(\Sigma_\iota)$.

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)

▷ **Definition 1.2.12.** We inherit the universe $\mathcal{D}_o=\{\mathsf{T},\mathsf{F}\}$ of truth values from PL^0 and assume an arbitrary universe $\mathcal{D}_\iota\neq\emptyset$ of individuals (this choice is a parameter to the semantics)

▷ **Definition 1.2.13.** 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 1.2.14.** A variable assignment $\varphi\colon\mathcal{V}_\iota\to\mathcal{D}_\iota$ maps variables into the universe.

▷ **Definition 1.2.15.** A model $\mathcal{M}=\langle\mathcal{D}_\iota,\mathcal{I}\rangle$ of PL^1 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)

▷ **Definition 1.2.16.**

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

- ▷ $\mathcal{I}_\varphi \colon \mathit{wff}_\iota(\Sigma_\iota, \mathcal{V}_\iota) \to \mathcal{D}_\iota$ assigns values to terms.
  - ▷ $\mathcal{I}_\varphi(X) := \varphi(X)$ and
  - ▷ $\mathcal{I}_\varphi(f(\mathbf{A}_1, \ldots, \mathbf{A}_k)) := \mathcal{I}(f)(\mathcal{I}_\varphi(\mathbf{A}_1), \ldots, \mathcal{I}_\varphi(\mathbf{A}_k))$
- ▷ $\mathcal{I}_\varphi \colon \mathit{wff}_o(\Sigma_\iota, \mathcal{V}_\iota) \to \mathcal{D}_o$ 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 1.2.17 (Assignment Extension).** Let φ be a variable assignment into D and $a \in D$, then $\varphi,[a/X]$ is called the extension of φ with $[a/X]$ and is defined as $\{(Y,a) \in \varphi | Y \neq X\} \cup \{(X,a)\}$: $\varphi,[a/X]$ coincides with φ off X, and gives the result a there.

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 extension of the incoming 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 of the 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 1.2.18.** 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 $\psi{:=}\varphi,[a/X]$ for $a{\in}\mathcal{D}_\iota$ we have

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

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

1.2.1.2 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 1.2.19.**[for=sbstListfromto,sbstListdots,sbst]

  Let $wfe(\Sigma,\mathcal{V})$ be an expression language, then we call $\sigma\colon \mathcal{V}{\to}wfe(\Sigma,\mathcal{V})$ a substitution, iff the support $\mathrm{supp}(\sigma){:=}\{X|(X,\mathbf{A}){\in}\sigma, X \neq \mathbf{A}\}$ of σ is finite. We denote the empty substitution with ϵ.
- **Definition 1.2.20 (Substitution Application).**

  We define substitution application by
  - $\sigma(c) = c$ for $c{\in}\Sigma$
  - $\sigma(X) = \mathbf{A}$, iff $\mathbf{A}{\in}\mathcal{V}$ and $(X,\mathbf{A}){\in}\sigma$.
  - $\sigma(f(\mathbf{A}_1,\ldots,\mathbf{A}_n)) = f(\sigma(\mathbf{A}_1),\ldots,\sigma(\mathbf{A}_n))$,
  - $\sigma(\beta X.\mathbf{A}) = \beta X.\sigma_{-X}(\mathbf{A})$.
- **Example 1.2.21.** $[a/x],[f(b)/y],[a/z]$ instantiates $g(x,y,h(z))$ to $g(a,f(b),h(a))$.
- **Definition 1.2.22.** Let σ be a substitution then we call $\mathrm{intro}(\sigma){:=}\bigcup_{X\in\mathrm{supp}(\sigma)}\mathrm{free}(\sigma(X))$ 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 1.2.23 (Substitution Extension).**

  Let σ be a substitution, then we denote the extension of σ with $[\mathbf{A}/X]$ by $\sigma,[\mathbf{A}/X]$ and define it as $\{(Y,\mathbf{B})\in\sigma | Y \neq X\} \cup \{(X,\mathbf{A})\}$: $\sigma,[\mathbf{A}/X]$ 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.
- We also need the dual operation: removing a variable from the support:
- **Definition 1.2.24.** We can discharge a variable X from a substitution σ by setting $\sigma_{-X}{:=}\sigma,[X/X]$.

Note that the use of the comma notation for substitutions defined in **??** 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 1.2.25.** $\sigma(\forall X.\mathbf{A}){:=}(\forall X.\sigma_{-X}(\mathbf{A}))$.
- **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 1.2.26.** Let $\mathbf{B}\in wff_\iota(\Sigma_\iota,\mathcal{V}_\iota)$ and $\mathbf{A}\in wff_o(\Sigma_\iota,\mathcal{V}_\iota)$, 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 1.2.27.** *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 $\mathbf{A}$:

1. depth=0 *Then* $\mathbf{A}$ *is a variable (say* Y*), or constant, so we have three cases*
   1.1. $\mathbf{A} = Y = X$
      1.1.1. then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](X)) = \mathcal{I}_\varphi(\mathbf{B}) = \psi(X) = \mathcal{I}_\psi(X) = \mathcal{I}_\psi(\mathbf{A})$.
   1.2. $\mathbf{A} = Y \neq X$
      1.2.1. then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](Y)) = \mathcal{I}_\varphi(Y) = \varphi(Y) = \psi(Y) = \mathcal{I}_\psi(Y) = \mathcal{I}_\psi(\mathbf{A})$.
   1.3. $\mathbf{A}$ is a constant
      1.3.1. Analogous to the preceding case ($Y \neq X$).
   1.4. This completes the base case (depth = 0).
2. depth> 0
   2.1. 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
   2.2. This completes the inductive case, and we have proven the assertion.

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG Kohlhase & Rabe: KRMT 80 2023-04-25

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

Substitution Value Lemma for Propositions

▷ **Lemma 1.2.28.** *Let* $\mathbf{B}{\in}wff_\iota(\Sigma_\iota, \mathcal{V}_\iota)$ *be substitutable for* X *in* $\mathbf{A}{\in}wff_o(\Sigma_\iota, \mathcal{V}_\iota)$*, 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 $\mathbf{A}$

1. $n = 0$
   1.1. then $\mathbf{A}$ is an atomic proposition, and we can argue like in the inductive case of the substitution value lemma for terms.
2. $n{>}0$ and $\mathbf{A} = \neg\mathbf{B}$ or $\mathbf{A} = \mathbf{C} \circ \mathbf{D}$
   2.1. Here we argue like in the inductive case of the term lemma as well.
3. $n{>}0$ and $\mathbf{A} = \forall X.\mathbf{C}$
   3.1. 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 $\mathcal{I}_\varphi(\forall X.\mathbf{C}) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathsf{T}$.

4. $n>0$ and $\mathbf{A} = \forall Y.\mathbf{C}$ where $X \neq Y$
   4.1. 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.
   4.2. 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.

1.2.1.3 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 1.2.29.** *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:

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

▷ **Definition 1.2.30.** 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 1.2.31 (Capture-Avoiding Substitution Application).** Let σ be a substitution, $\mathbf{A}$ a formula, and $\mathbf{A}'$ an alphabetical variant of $\mathbf{A}$, such that intro$(\sigma)\cap$ BVar$(\mathbf{A})=\emptyset$. Then $\mathbf{A}_{=_\alpha}=\mathbf{A}'_{=_\alpha}$ and we can define $\sigma(\mathbf{A}_{=_\alpha}):=(\sigma(\mathbf{A}'))_{=_\alpha}$.

- **Notation:** 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).

Undecidability of First-Order Logic

- **Theorem 1.2.32.** *Validity in first-order logic is undecidable.*
- *Proof:* We prove this by contradiction
  1. Let us assume that there is a

1.2.2 First-Order Calculi

In this subsection 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 giving inference rules for the connectives and then extend this to a calculus for first-order logic by adding rules for the quantifiers.

1.2.2.1 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. In particular, it 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, we introduce a natural deduction calculus that provides two/three inference rules for every logical constant, 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 argumentation for theorem proving.
- **Definition 1.2.33.** The propositional natural deduction calculus $\mathcal{ND}_0$ has inference rules for the introduction and elimination of connectives:

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

  $\Rightarrow I$ proves $\mathbf{A}\Rightarrow\mathbf{B}$ by exhibiting a $\mathcal{ND}_0$ derivation $\mathcal{D}$ (depicted by the double horizontal lines) of $\mathbf{B}$ from the local hypothesis $\mathbf{A}$; $\Rightarrow I$ then discharges (get rid of $\mathbf{A}$, which can only be used in $\mathcal{D}$) the hypothesis and concludes $\mathbf{A}\Rightarrow\mathbf{B}$. This mode of reasoning is called hypothetical reasoning.

- **Definition 1.2.34.**

  Given a set $\mathcal{H}\subseteq wff_0(\mathcal{V}_0)$ of assumptions and a conclusion $\mathbf{C}$, we write $\mathcal{H}\vdash_{\mathcal{ND}_0}\mathbf{C}$, iff there is a $\mathcal{ND}_0$ derivation tree whose leaves are in $\mathcal{H}$.

- **Note:** TND is used only in classical logic (otherwise constructive/intuitionistic)

The most characteristic rule in the natural deduction calculus is the $\Rightarrow I$ rule and the hypothetical reasoning it introduce. $\Rightarrow I$ 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 local hypothesis. When we can do this we discharge the assumption and conclude $\mathbf{A}\Rightarrow\mathbf{B}$.

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 1.2.35 (Inference with Local Hypotheses).**

$$\cfrac{\cfrac{\cfrac{[\mathbf{A}\wedge\mathbf{B}]^1}{\mathbf{B}}\wedge E_r \quad \cfrac{[\mathbf{A}\wedge\mathbf{B}]^1}{\mathbf{A}}\wedge E_l}{\mathbf{B}\wedge\mathbf{A}}\wedge I}{\mathbf{A}\wedge\mathbf{B}\Rightarrow\mathbf{B}\wedge\mathbf{A}}\Rightarrow I^1 \qquad \begin{array}{c}[A]^1\\ [B]^2\\ \cfrac{\cfrac{A}{B\Rightarrow A}\Rightarrow I^2}{A\Rightarrow B\Rightarrow A}\Rightarrow I^1\end{array}$$

Here we see hypothetical 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 local hypothesishypothesis and the discharging rule, i.e. even if this proof is only a fragment of a larger proof, then we cannot use its local hypothesishypothesis 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 they 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 1.2.36.** $\mathcal{H},\mathbf{A}\vdash_{\mathcal{ND}_0}\mathbf{B}$, *iff* $\mathcal{H}\vdash_{\mathcal{ND}_0}\mathbf{A}\Rightarrow\mathbf{B}$.

▷ *Proof:* We show the two directions separately

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
2. If $\mathcal{H}\vdash_{\mathcal{ND}_0}\mathbf{A}\Rightarrow\mathbf{B}$, then $\mathcal{H},\mathbf{A}\vdash_{\mathcal{ND}_0}\mathbf{A}\Rightarrow\mathbf{B}$ by weakening and $\mathcal{H},\mathbf{A}\vdash_{\mathcal{ND}_0}\mathbf{B}$ 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 1.2.33 for disjunction, negation and falsity.

More Rules for Natural Deduction

▷ **Note:** $\mathcal{ND}_0$ does not try to be minimal, but comfortable to work in!x

▷ **Definition 1.2.37.** $\mathcal{ND}_0$ has the following additional inference rules for the remaining connectives.

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

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

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

▷ **Again:** $\neg E$ is used only in classical logic (otherwise constructive/intuitionistic)

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

Natural Deduction in Sequent Calculus Formulation

▷ **Idea:** Represent hypotheses explicitly. (lift calculus to judgments)

▷ **Definition 1.2.38.** A judgment is a meta statement about the provability of propositions.

▷ **Definition 1.2.39.** 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. We write $\vdash\mathbf{A}$ for $\emptyset\vdash\mathbf{A}$.

▷ **Idea:** Reformulate $\mathcal{ND}_0$ inference rules so that they act on sequents.

▷ **Example 1.2.40.** We give the sequent style version of Example 1.2.35:

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

▷ **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 1.2.41.** The following inference rules make up the propositional sequent style natural deduction calculus $\mathcal{ND}_\vdash^0$:

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

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

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

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

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

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

Linearized Notation for (Sequent-Style) ND Proofs

▷ Linearized notation for sequent-style ND proofs

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

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 1.2.42.** We show a linearized version of the $\mathcal{ND}_0$ examples Example 1.2.40

#	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, 3
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

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 steps 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 connectives just as always
- ▷ **Definition 1.2.43 (New Quantifier Rules).** The first-order natural deduction calculus $\mathcal{ND}^1$ extends $\mathcal{ND}_0$ by the following four rules:

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

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

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

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

Now we reformulate the classical formulation of the calculus of natural deduction as a sequent calculus by lifting it to the "judgements level" as we die for propositional logic. We only need provide new quantifier rules.

First-Order Natural Deduction in Sequent Formulation

- ▷ Rules for connectives from $\mathcal{ND}^0_\vdash$
- ▷ **Definition 1.2.45 (New Quantifier Rules).** The inference rules of the first-order sequent calculus $\mathcal{ND}^1_\vdash$ consist of those from $\mathcal{ND}^0_\vdash$ plus the following quantifier rules:

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

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

Natural Deduction with Equality

- ▷ **Definition 1.2.46 (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 $\mathcal{I}(=):=\{(x,x)|x\in\mathcal{D}_\iota\}$. We call the extended logic first-order logic with equality ($\mathrm{PL}^1_=$)
- ▷ We now extend natural deduction as well.
- ▷ **Definition 1.2.47.** For the calculus of natural deduction with equality ($\mathcal{ND}^1_=$) we add the following two 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 1.2.48.** $\Leftrightarrow I$ is derivable and $\Leftrightarrow E$ is admissible in $\mathcal{ND}^1$:

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

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

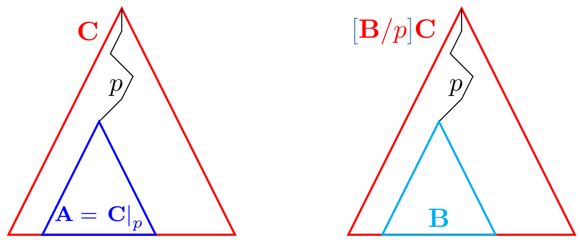
- ▷ **Idea:** Formulae are (naturally) trees, so we can use tree positions to talk about subformulae

▷ **Definition 1.2.49.** A position p is a tuple of natural numbers that in each node of a expression (tree) specifies into which child to descend. For a expression $\mathbf{A}$ we denote the subexpression at p with $\mathbf{A}|_p$.

We will sometimes write a expression $\mathbf{C}$ as $\mathbf{C}[\mathbf{A}]_p$ to indicate that $\mathbf{C}$ the subexpression $\mathbf{A}$ at position p.

▷ **Definition 1.2.50.** Let p be a position, then $[\mathbf{A}/p]\mathbf{C}$ is the expression obtained from $\mathbf{C}$ by replacing the subexpression at p by $\mathbf{A}$.

▷ **Example 1.2.51 (Schematically).**



Kohlhase & Rabe: KRMT 96 2023-04-25

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 1.2.52.** *$\sqrt{2}$ is irrational*

Proof: We prove the assertion by contradiction

1. Assume that $\sqrt{2}$ is rational.
2. Then there are numbers p and q such that $\sqrt{2} = p/q$.
3. So we know $2q^2 = p^2$.
4. But $2q^2$ has an odd number of prime factors while p^2 an even number.
5. This is a contradiction (since they are equal), so we have proven the assertion

Kohlhase & Rabe: KRMT 97 2023-04-25

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 $\prime n$ 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		$\forall n, m.\neg(2n+1) = (2m)$	lemma
2		$\forall n, m.\#(n^m) = m\#(n)$	lemma
3		$\forall n, p.\mathsf{prime}(p) \Rightarrow \#(pn) = (\#(n)+1)$	lemma
4		$\forall x.\mathsf{irr}(x) \Leftrightarrow (\neg(\exists p, q.x = p/q))$	definition
5		$\mathsf{irr}(\sqrt{2}) \Leftrightarrow (\neg(\exists p, q.\sqrt{2} = p/q))$	$\forall E(4)$
6	6	$\neg\mathsf{irr}(\sqrt{2})$	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	$2q^2 = p^2$	arith(9)
11	6,9	$\#(p^2) = 2\#(p)$	$\forall E^2(2)$
12	6,9	$\mathsf{prime}(2) \Rightarrow \#(2q^2) = (\#(q^2)+1)$	$\forall E^2(1)$

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG Kohlhase & Rabe: KRMT 98 2023-04-25

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		$\mathsf{prime}(2)$	lemma
14	6,9	$\#(2q^2) = \#(q^2)+1$	$\Rightarrow E(13,12)$
15	6,9	$\#(q^2) = 2\#(q)$	$\forall E^2(2)$
16	6,9	$\#(2q^2) = 2\#(q)+1$	$=E(14,15)$
17		$\#(p^2) = \#(p^2)$	$=I$
18	6,9	$\#(2q^2) = \#(q^2)$	$=E(17,10)$
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		$\neg\neg\mathsf{irr}(\sqrt{2})$	$\neg I^6(23)$
25		$\mathsf{irr}(\sqrt{2})$	$\neg E^2(23)$

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG Kohlhase & Rabe: KRMT 99 2023-04-25

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.

1.3 Higher-Order Logic and λ-Calculus

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

1.3.1 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 (PLΩ)

- ▷ Quantification over functions and Predicates: $\forall P.\exists F.P(a) \vee \neg P(F(a))$
- ▷ **Definition 1.3.1.** Comprehension: (Existence of Functions)
  $\exists F.\forall X.FX = \mathbf{A}$ e.g. $f(x) = 3x^2 + 5x + 7$
- ▷ **Definition 1.3.2.** Extensionality: (Equality of functions and truth values)
  $\forall F.\forall G.(\forall X.FX = GX) \Rightarrow F = G$
  $\forall P.\forall Q.PQ \Leftrightarrow P = Q$
- ▷ **Definition 1.3.3.** Leibniz Equality: (Indiscernability)
  $\mathbf{A} = \mathbf{B}$ for $\forall P.P\mathbf{A} \Rightarrow P\mathbf{B}$

Kohlhase & Rabe: KRMT 100 2023-04-25

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, prop (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 1.3.4.** 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 1.3.5.**

We will use ι for the type ofindividuals and prop for the type of truth values.

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

▷ **Vector Notation:**

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)

▷ **Definition 1.3.6.** Signature $\Sigma_{\mathcal{T}} = \bigcup_{\alpha \in \mathcal{T}} \Sigma_\alpha$ with

▷ **Definition 1.3.7.** Connectives: $\neg \in \Sigma_{\mathrm{prop} \to \mathrm{prop}}$ $\{\vee, \wedge, \Rightarrow, \Leftrightarrow, \ldots\} \subseteq \Sigma_{\mathrm{prop} \to \mathrm{prop} \to \mathrm{prop}}$

▷ **Definition 1.3.8.** Variables $\mathcal{V}_{\mathcal{T}} = \bigcup_{\alpha \in \mathcal{T}} \mathcal{V}_\alpha$, such that every $\mathcal{V}_\alpha$ countably infinite.

▷ **Definition 1.3.9.** Well typed formulae $wff_\alpha(\Sigma_\mathcal{T},\mathcal{V}_\mathcal{T})$ of type α

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

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

▷ 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 $\alpha \to \mathsf{prop}$.

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 1.3.10.** The universe of discourse (also carrier) consists of:

- ▹ an arbitrary, non-empty set of individuals $\mathcal{D}_\iota$,
- ▹ a fixed set of truth values $\mathcal{D}_{\mathsf{prop}} = \{\mathsf{T},\mathsf{F}\}$, and
- ▹ function universes $\mathcal{D}_{(\alpha\to\beta)} = \mathcal{D}_\alpha \to \mathcal{D}_\beta$.

▷ **Definition 1.3.11.** Interpretation of constants: typed mapping $\mathcal{I}\colon \Sigma_\mathcal{T}\to D$ (i.e. $\mathcal{I}(\Sigma_\alpha) \subseteq \mathcal{D}_\alpha$)

▷ **Definition 1.3.12.** We call a structure $\langle\mathcal{D},\mathcal{I}\rangle$, where $\mathcal{D}$ is a universe and $\mathcal{I}$ an interpretation a standard model of PLΩ.

▷ **Definition 1.3.13.** A variable assignment is a typed mapping $\varphi\colon \mathcal{V}_\mathcal{T}\to D$.

▷ **Definition 1.3.14.** A value function is a typed mapping $\mathcal{I}_\varphi\colon wff_\mathcal{T}(\Sigma_\mathcal{T},\mathcal{V}_\mathcal{T})\to D$ with

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

▷ **Definition 1.3.15.** $\mathbf{A}_{\mathsf{prop}}$ 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 1.3.16 (Leibniz equality).** $\mathbf{Q}^\alpha\mathbf{A}_\alpha\mathbf{B}_\alpha = \forall P_{\alpha\to\mathsf{prop}}.P\mathbf{A} \Leftrightarrow P\mathbf{B}$ (indiscernability)

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

▷ **Theorem 1.3.17.** *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:** 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:*

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 $r \in \mathcal{D}_{(\alpha\to\mathsf{prop})}$.
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 $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{B}) = r(\mathcal{I}_\varphi(\mathbf{B})) = \mathsf{T}$.
3. Thus $\mathcal{I}_\varphi(\mathbf{QAB}) = \mathsf{T}$.
4. Let $\mathcal{I}_\varphi(\mathbf{A}) \neq \mathcal{I}_\varphi(\mathbf{B})$ and $r = \{\mathcal{I}_\varphi(\mathbf{A})\} \in \mathcal{D}_{(\alpha\to\mathsf{prop})}$ (exists in a standard model)
5. so $r(\mathcal{I}_\varphi(\mathbf{A})) = \mathsf{T}$ and $r(\mathcal{I}_\varphi(\mathbf{B})) = \mathsf{F}$
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 $\mathcal{I}_{\varphi,[r/P]}(P\mathbf{B}) = r(\mathcal{I}_\varphi(\mathbf{B})) = \mathsf{F}$.

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG Kohlhase & Rabe: KRMT 106 2023-04-25

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

▷ $\Sigma_\mathcal{T} = \{[\mathbb{N}{:}\iota \to \mathsf{prop}], [0{:}\iota], [s{:}\iota \to \iota]\}$

▷ $\mathbb{N}0$ (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)

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

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG Kohlhase & Rabe: KRMT 107 2023-04-25

Finally, we show the expressivity of $\mathrm{PL}\Omega$ by formalizing a version of Cantor's theorem.

Expressive Formalism for Mathematics

▷ **Example 1.3.18 (Cantor's Theorem).** The cardinality of a set is smaller than that of its power set.

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

▷ **Example 1.3.19 (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 1.3.19 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 1.3.20 ($\mathcal{H}_\Omega$ Axioms).**
  - ▷ $\forall P_{\mathrm{prop}}, Q_{\mathrm{prop}}. P \Rightarrow Q \Rightarrow P$
  - ▷ $\forall P_{\mathrm{prop}}, Q_{\mathrm{prop}}, R_{\mathrm{prop}}.(P \Rightarrow Q \Rightarrow R) \Rightarrow (P \Rightarrow Q) \Rightarrow P \Rightarrow R$
  - ▷ $\forall P_{\mathrm{prop}}, Q_{\mathrm{prop}}.(\neg P \Rightarrow \neg Q) \Rightarrow P \Rightarrow Q$
- ▷ **Definition 1.3.21 ($\mathcal{H}_\Omega$ inference rules).**

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

- ▷ **Theorem 1.3.22.** *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 1.3.23.** Valid sentences that are not $\mathcal{H}_\Omega$-theorems:
  - ▷ Cantor's Theorem:
    $\neg(\exists F_{\iota \to \iota \to \iota}.\forall G_{\iota \to \iota}.(\forall K_\iota.\mathbb{N}\ K \Rightarrow \mathbb{N}\ G\ K) \Rightarrow (\exists.\mathbb{N}\ J \wedge FJ = G))$
    (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 $\exists G_{\iota \to \iota}.\forall X_\iota.GX = s(fXX)$
- ▷ **Definition 1.3.24 (New Axiom Schema).** Comprehension axiom $\exists F_{\alpha \to \beta}.\forall X_\alpha.F\ X = \mathbf{A}_\beta$ (for every variable X_α and every term $\mathbf{A} \in wff_\beta(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$)
- ▷ **Definition 1.3.25 (new axiom schemata).** Extensionality axiom

$\mathbf{Ext}^{\alpha\beta}$ $\forall F_{\alpha\to\beta}.\forall G_{\alpha\to\beta}.(\forall X_{\alpha}.FX = GX) \Rightarrow F = G$
$\mathbf{Ext}^{o}$ $\forall F_{\mathsf{prop}}.\forall G_{\mathsf{prop}}.FG \Leftrightarrow F = G$

▷ 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

▷ **Observation:** Gödel's incompleteness theorem only holds for standard semantics.

▷ **Idea:** Find generalization that admits complete calculi

▷ **Concretely:** Generalize so that the carrier only contains those functions that are requested by the comprehension axioms.

▷ **Theorem 1.3.26 (Henkin's theorem).** *$\mathcal{H}_{\Omega}$ is complete wrt. this semantics.*

▷ *Proof sketch:* more models $\leadsto$ less valid sentences (these are $\mathcal{H}_{\Omega}$-theorems)

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

1.3.2 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 difficult 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 $\mathbf{A}\in wf\!f_{\beta}(\Sigma_{\mathcal{T}},\mathcal{V}_{\mathcal{T}})$ (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 $\mathbf{B}\in wf\!f_{\alpha}(\Sigma_{\mathcal{T}},\mathcal{V}_{\mathcal{T}})$.

▷ **Definition 1.3.27.** Axiom of β equality: $(\lambda X_\alpha.\mathbf{A})\ \mathbf{B} = [\mathbf{B}/X](\mathbf{A}_\beta)$

▷ **Idea:** Introduce a new class of formulae (λ-calculus [Chu40])

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

From Extensionality to η-Conversion

▷ **Definition 1.3.28.** 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 1.3.29.** 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 1.3.30.** *η-equality and Extensionality are equivalent*

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

1. Let $\forall X_\alpha.\mathbf{A}X = \mathbf{B}X$, thus $\mathbf{A}X = \mathbf{B}X$ with $\forall E$
2. $\lambda X_\alpha.\mathbf{A}X = \lambda X_\alpha.\mathbf{B}X$, therefore $\mathbf{A} = \mathbf{B}$ with η
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: $\forall F_{\mathsf{prop}}.\forall G_{\mathsf{prop}}.FG \Leftrightarrow F = G$ 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.

1.3.3 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)

▷ **Definition 1.3.31.** Signature $\Sigma_\mathcal{T} = \bigcup_{\alpha\in\mathcal{T}}\Sigma_\alpha$ (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 1.3.32.** We call the set $\mathit{wff}_\alpha(\Sigma_\mathcal{T}, \mathcal{V}_\mathcal{T})$ defined by the rules

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

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

- ▷ **Definition 1.3.33.** 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. $\sigma(X_\alpha)\in\mathit{wff}_{\alpha}(\Sigma_{\mathcal{T}},\mathcal{V}_{\mathcal{T}})$ and capture-avoiding.
- ▷ **Definition 1.3.34 (Simply Typed λ-Calculus).** The simply typed λ calculus $\Lambda^{\to}$ over a signature $\Sigma_{\mathcal{T}}$ has the formulae $\mathit{wff}_{\mathcal{T}}(\Sigma_{\mathcal{T}},\mathcal{V}_{\mathcal{T}})$ (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 $\Lambda^{\to}$: The first rule (α-conversion) just says that we can rename bound variables as we like. β conversion codifies the intuition behind function application by replacing bound variables with argument. The equality relation induced by the η-reduction is a special case of the extensionality principle for functions ($f = g$ iff $f(a) = g(a)$ for all possible arguments a): If we apply both sides of the transformation to the same argument – say $\mathbf{B}$ and then we arrive at the right hand side, since $(\lambda X_\alpha.\mathbf{A}\ X)\mathbf{B}=_\beta\mathbf{A}\ \mathbf{B}$.
We will use a set of bracket elision rules that make the syntax of $\Lambda^{\to}$ more palatable. This makes $\Lambda^{\to}$ expressions look much more like regular mathematical notation, but hides the internal structure. Readers should make sure that they can always reconstruct the brackets to make sense of the syntactic notions below.

Simply typed λ-Calculus (Notations)

- ▷ **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.
- ▷ **Andrews' dot Notation:** A **.** stands for a left bracket whose partner is as far right as is consistent with existing brackets; i.e. $\mathbf{A}\ .\mathbf{B}\ \mathbf{C}$ abbreviates $\mathbf{A}\ (\mathbf{B}\ \mathbf{C})$.
- ▷ **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.
- ▷ **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 $\mathbf{B}$.[1] In this presentation of the simply typed λ-calculus we build-in $=_\alpha$-equality and use capture-avoiding substitutions directly. A clean introduction would followed the steps in subsection 1.2.1 by introducing substitutions with a substitutability condition

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

like the one in Definition 1.2.26, then establishing the soundness of $=_\alpha$ conversion, and only then postulating defining capture-avoiding substitution application as in ??. 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.

$=_{\alpha\beta\eta}$-Equality (Overview)

- reduction with $\begin{cases} =_\beta : (\lambda X.\mathbf{A})\,\mathbf{B} \rightarrow_\beta [\mathbf{B}/X](\mathbf{A}) \\ =_\eta : (\lambda X.\mathbf{A}\;X) \rightarrow_\eta \mathbf{A} \end{cases}$ under $=_\alpha$: $\begin{matrix}\lambda X.\mathbf{A} \\ =_\alpha \\ \lambda Y.[Y/X](\mathbf{A})\end{matrix}$
- **Theorem 1.3.35.** *β-reduction is well-typed, terminating and confluent in the presence of α-conversion.*
- **Definition 1.3.36 (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 1.3.37.** *$=_{\beta\eta}$-reduction yields unique normal forms (up to $=_\alpha$-equivalence).*

Kohlhase & Rabe: KRMT 116 2023-04-25

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

Syntactic Parts of λ-Terms

- **Definition 1.3.38 (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 1.3.39.**

  Head reduction always has a unique β redex

  $$(\lambda \overline{X^n}.\lambda Y.\mathbf{A}(\mathbf{B}^2, \ldots, \mathbf{B}^n)) \rightarrow^h_\beta (\lambda \overline{X^n}.[\mathbf{B}^1/Y](\mathbf{A})(\mathbf{B}^2, \ldots, \mathbf{B}^n))$$
- **Theorem 1.3.40.** *The syntactic heads of β-normal forms are constant or variables.*
- **Definition 1.3.41.** 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 $\mathrm{head}(\mathbf{A})$. We call a λ-term a j-projection, iff its head is the j^{th} bound variable.
- **Definition 1.3.42.** We call a λ-term a η long form, iff its matrix has base type.

▷ **Definition 1.3.43.** 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 1.3.44.** 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 $\beta \in \mathcal{BT}$, then $\mathbf{A}$ must be of the form $\lambda\overline{X^n_\alpha}.\mathbf{B}$, where $\mathbf{B}$ has type β. Furthermore, the set of η-long forms is closed under β-equality, which allows us to treat the two equality theories of $\Lambda^{\to}$ separately and thus reduce argumentational complexity.

A Test Generator for Higher-Order Unification

▷ **Definition 1.3.45 (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 1.3.46.** *Subtraction and (integer) division on Church numberals can be automted via higher-order unification.*

▷ **Example 1.3.47.**

$5-2$ 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.

1.3.4 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 1.3.48.**

A grammar for the raw terms of the simply typed λ-calculus:

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

▷ **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 1.3.49.** Judgments make statements about complex properties of the syntactic entities defined by the grammar.

▷ **Definition 1.3.50.** Judgments for the simply typed λ-calculus

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

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

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

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

▷ **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

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

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

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

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

Example: A Well-Formed Signature

- Let $\Sigma{:=}[\alpha : \mathsf{type}], [f{:}\alpha \to \alpha \to \alpha]$, then Σ is a well-formed signature, since we have derivations $\mathcal{A}$ and $\mathcal{B}$

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

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

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

Example: A Well-Formed λ-Term

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

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

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

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

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

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

$$\dfrac{\Gamma, [X{:}\beta] \vdash_\Sigma \mathbf{A} : \alpha \quad \Gamma \vdash_\Sigma \mathbf{B} : \beta}{\Gamma \vdash_\Sigma (\lambda X.\mathbf{A})\ \mathbf{B} \to_\beta^1 [\mathbf{B}/X](\mathbf{A})}\,wff\beta top$$

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} : \beta \to \alpha \quad X \notin \mathbf{dom}(\Gamma)}{\Gamma \vdash_\Sigma \lambda X.\mathbf{A}\ X \to_\eta^1 \mathbf{A}}\,wff\eta top$$

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} \to_+^1 \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{A}\ \mathbf{C} : \alpha}{\Gamma \vdash_\Sigma \mathbf{A}\ \mathbf{C} \to_+^1 \mathbf{B}\ \mathbf{C}}\,tr\,appfn$$

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} \to_+^1 \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{C}\ \mathbf{A} : \alpha}{\Gamma \vdash_\Sigma \mathbf{C}\ \mathbf{A} \to_+^1 \mathbf{C}\ \mathbf{B}}\,tr\,apparg$$

$$\dfrac{\Gamma, [X{:}\alpha] \vdash_\Sigma \mathbf{A} \to_+^1 \mathbf{B}}{\Gamma \vdash_\Sigma \lambda X.\mathbf{A} \to_+^1 \lambda X.\mathbf{B}}\,tr\,abs$$

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

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

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} \to_+^1 \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{A} \to_+^* \mathbf{B}}\,ms\,start \qquad \dfrac{\Gamma \vdash_\Sigma \mathbf{A} : \alpha}{\Gamma \vdash_\Sigma \mathbf{A} \to_+^* \mathbf{A}}\,ms\,ref$$

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} \to_+^* \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{B} \to_+^* \mathbf{C}}{\Gamma \vdash_\Sigma \mathbf{A} \to_+^* \mathbf{C}}\,ms\,trans$$

▷ Congruence Relation

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} \to_+^* \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{B}}\,eq\,start$$

$$\dfrac{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{B}}{\Gamma \vdash_\Sigma \mathbf{B} =_+ \mathbf{A}}\,eq\,sym \qquad \dfrac{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{B} \quad \Gamma \vdash_\Sigma \mathbf{B} =_+ \mathbf{C}}{\Gamma \vdash_\Sigma \mathbf{A} =_+ \mathbf{C}}\,eq\,trans$$

Type Computation: Manage Types Algorithmically

- **Questions:**
  - type check: Is $\Gamma \vdash_\Sigma \mathbf{A} : \alpha$?
  - type inference: are there Γ, α, such that $\Gamma \vdash_\Sigma \mathbf{A} : \alpha$?
  - type reconstruction the above without type annotations at bound variables?
- prenex fragment makes problems decidable (see Curry Howard)
- **Algorithm (Hindley & Milner):**
  - invert inference rules
  - first-order unification,
  - universal generalization, minimization

Example Computation

rule tree	constraint
$$\cfrac{\cfrac{\cfrac{\cfrac{[X{:}\alpha]}{\Gamma, [X{:}\beta]}}{\Gamma, [X{:}\beta] \vdash_\Sigma X : \alpha} \quad \Gamma \vdash_\Sigma \lambda X.X : \beta \to \alpha}{\cfrac{\cfrac{[W{:}\delta] \in \Gamma, [Z{:}\gamma]}{\Gamma, [Z{:}\gamma] \vdash_\Sigma W : \delta}}{\Gamma \vdash_\Sigma \lambda Z.W : \beta}} \;\Gamma \vdash_\Sigma \lambda X.X(\lambda Z.W) : \alpha$$	$\alpha = \beta$, $[W{:}\delta] \in \Gamma$, $\beta = \gamma \to \delta$

- unification: $\alpha = \beta = \gamma \to \delta$,
- minimization: $\Gamma = [W{:}\delta]$
- **Solution:** $[W{:}\delta]] \vdash_\Sigma \lambda X.X(\lambda Z.W) : \forall \gamma.\gamma \to \delta$

1.3.5 The Semantics of the Simply Typed λ-Calculus

The semantics of $\Lambda^{\to}$ is structured around the types. Like the models we discussed before, a model (we call them “algebras”, since we do not have truth values in $\Lambda^{\to}$) is a pair $\langle \mathcal{D}, \mathcal{I} \rangle$, where $\mathcal{D}$ is the universe of discourse and $\mathcal{I}$ is the interpretation of constants.

Semantics of $\Lambda^{\to}$

- **Definition 1.3.51.** We call a collection $\mathcal{D}_\mathcal{T} := \{\mathcal{D}_\alpha \mid \alpha \in \mathcal{T}\}$ a typed collection (of sets) and a collection $f_\mathcal{T} : \mathcal{D}_\mathcal{T} \to \mathcal{E}_\mathcal{T}$, a typed function, iff $f_\alpha : \mathcal{D}_\alpha \to \mathcal{E}_\alpha$.

▷ **Definition 1.3.52.** A typed collection $\mathcal{D}_{\mathcal{T}}$ is called a frame, iff $\mathcal{D}_{(\alpha\to\beta)} \subseteq \mathcal{D}_\alpha \to \mathcal{D}_\beta$

▷ **Definition 1.3.53.** Given a frame $\mathcal{D}_{\mathcal{T}}$, and a typed function $\mathcal{I}\colon \Sigma\to\mathcal{D}$, then we call $\mathcal{I}_\varphi\colon \mathit{wff}_{\mathcal{T}}(\Sigma_{\mathcal{T}},\mathcal{V}_{\mathcal{T}})\to\mathcal{D}$ the value function induced by $\mathcal{I}$, iff

- ▷ $\mathcal{I}_\varphi|_{\mathcal{V}_{\mathcal{T}}} = \varphi$, $\qquad \mathcal{I}_\varphi|_{\Sigma_{\mathcal{T}}} = \mathcal{I}$
- ▷ $\mathcal{I}_\varphi(\mathbf{A}\,\mathbf{B}) = \mathcal{I}_\varphi(\mathbf{A})(\mathcal{I}_\varphi(\mathbf{B}))$
- ▷ $\mathcal{I}_\varphi(\lambda X_\alpha.\mathbf{A})$ is that function $f\in\mathcal{D}_{(\alpha\to\beta)}$, such that $f(a) = \mathcal{I}_{\varphi,[a/X]}(\mathbf{A})$ for all $a\in\mathcal{D}_\alpha$

▷ **Definition 1.3.54.** We call a frame $\langle\mathcal{D},\mathcal{I}\rangle$ comprehension closed or a $\Sigma_{\mathcal{T}}$-algebra, iff $\mathcal{I}_\varphi\colon \mathit{wff}_{\mathcal{T}}(\Sigma_{\mathcal{T}},\mathcal{V}_{\mathcal{T}})\to\mathcal{D}$ is total. (every λ-term has a value)

1.3.5.1 Soundness of the Simply Typed λ-Calculus

We will now show is that $=_{\alpha\beta\eta}$-reduction does not change the value of formulae, i.e. if $\mathbf{A}=_{\alpha\beta\eta}\mathbf{B}$, then $\mathcal{I}_\varphi(\mathbf{A}) = \mathcal{I}_\varphi(\mathbf{B})$, for all $\mathcal{D}$ and φ. We say that the reductions are sound. As always, the main tool for proving soundess is a substitution value lemma. It works just as always and verifies that we the definitions are in our semantics plausible.

Substitution Value Lemma for λ-Terms

▷ **Lemma 1.3.55 (Substitution Value Lemma).** *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be terms, then* $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\psi(\mathbf{A})$, *where* $\psi = \varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]$

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

we have five cases

1. $\mathbf{A} = X$
   1.1. Then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](X)) = \mathcal{I}_\varphi(\mathbf{B}) = \psi(X) = \mathcal{I}_\psi(X) = \mathcal{I}_\psi(\mathbf{A})$.
2. $\mathbf{A} = Y \neq X$ and $Y\in\mathcal{V}_{\mathcal{T}}$
   2.1. then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](Y)) = \mathcal{I}_\varphi(Y) = \varphi(Y) = \psi(Y) = \mathcal{I}_\psi(Y) = \mathcal{I}_\psi(\mathbf{A})$.
3. $\mathbf{A}\in\Sigma_{\mathcal{T}}$
   3.1. This is analogous to the last case.
4. $\mathbf{A} = \mathbf{C}\,\mathbf{D}$
   4.1. then $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A})) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{C}\,\mathbf{D})) = \mathcal{I}_\varphi(([\mathbf{B}/X](\mathbf{C}))\ ([\mathbf{B}/X](\mathbf{D}))) = \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{C}))(\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{D}))) = \mathcal{I}_\psi(\mathbf{C})(\mathcal{I}_\psi(\mathbf{D})) = \mathcal{I}_\psi(\mathbf{C}\,\mathbf{D}) = \mathcal{I}_\psi(\mathbf{A})$
5. $\mathbf{A} = \lambda Y_\alpha.\mathbf{C}$
   5.1. We can assume that $X \neq Y$ and $Y \notin \mathrm{free}(\mathbf{B})$
   5.2. Thus for all $a\in\mathcal{D}_\alpha$ we have $\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}))(a) = \mathcal{I}_\varphi([\mathbf{B}/X](\lambda Y.\mathbf{C}))(a) = \mathcal{I}_\varphi(\lambda Y.[\mathbf{B}/X](\mathbf{C}))(a) = \mathcal{I}_{\varphi,[a/Y]}([\mathbf{B}/X](\mathbf{C})) = \mathcal{I}_{\psi,[a/Y]}(\mathbf{C}) = \mathcal{I}_\psi(\lambda Y.\mathbf{C})(a) = \mathcal{I}_\psi(\mathbf{A})(a)$

Soundness of αβη-Equality

▷ **Theorem 1.3.56.** *Let $\mathcal{A}:=\langle\mathcal{D},\mathcal{I}\rangle$ be a $\Sigma_{\mathcal{T}}$-algebra and $Y\notin\mathrm{free}(\mathbf{A})$, then $\mathcal{I}_\varphi(\lambda X.\mathbf{A})=\mathcal{I}_\varphi(\lambda Y.[Y/X]\mathbf{A})$ for all assignments φ.*

▷ *Proof:* by substitution value lemma

$$\begin{aligned}\mathcal{I}_\varphi(\lambda Y.[Y/X]\mathbf{A})@\mathsf{a} &= \mathcal{I}_{\varphi,[\mathsf{a}/Y]}([Y/X](\mathbf{A}))\\ &= \mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{A})\\ &= \mathcal{I}_\varphi(\lambda X.\mathbf{A})@\mathsf{a}\end{aligned}$$

▷ **Theorem 1.3.57.** *If $\mathcal{A}:=\langle\mathcal{D},\mathcal{I}\rangle$ is a $\Sigma_{\mathcal{T}}$-algebra and X not bound in $\mathbf{A}$, then $\mathcal{I}_\varphi((\lambda X.\mathbf{A})\ \mathbf{B})=\mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}))$.*

Proof: by substitution value lemma again

▷

$$\begin{aligned}\mathcal{I}_\varphi((\lambda X.\mathbf{A})\ \mathbf{B}) &= \mathcal{I}_\varphi(\lambda X.\mathbf{A})@\mathcal{I}_\varphi(\mathbf{B})\\ &= \mathcal{I}_{\varphi,[\mathcal{I}_\varphi(\mathbf{B})/X]}(\mathbf{A})\\ &= \mathcal{I}_\varphi([\mathbf{B}/X](\mathbf{A}))\end{aligned}$$

Soundness of $\alpha\beta\eta$ (continued)

▷ **Theorem 1.3.58.** *If $X\notin\mathrm{free}(\mathbf{A})$, then $\mathcal{I}_\varphi(\lambda X.\mathbf{A}\ X)=\mathcal{I}_\varphi(\mathbf{A})$ for all φ.*

▷ *Proof:* by calculation

$$\begin{aligned}\mathcal{I}_\varphi(\lambda X.\mathbf{A}\ X)@\mathsf{a} &= \mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{A}\ X)\\ &= \mathcal{I}_{\varphi,[\mathsf{a}/X]}(\mathbf{A})@\mathcal{I}_{\varphi,[\mathsf{a}/X]}(X)\\ &= \mathcal{I}_\varphi(\mathbf{A})@\mathcal{I}_{\varphi,[\mathsf{a}/X]}(X) \qquad \text{as } X\notin\mathrm{free}(\mathbf{A}).\\ &= \mathcal{I}_\varphi(\mathbf{A})@\mathsf{a}\end{aligned}$$

▷ **Theorem 1.3.59.** *$\alpha\beta\eta$-equality is sound wrt. $\Sigma_{\mathcal{T}}$-algebras. (if $\mathbf{A}=_{\alpha\beta\eta}\mathbf{B}$, then $\mathcal{I}_\varphi(\mathbf{A})=\mathcal{I}_\varphi(\mathbf{B})$ for all assignments φ)*

1.3.5.2 Completeness of $\alpha\beta\eta$-Equality

We will now show is that $=_{\alpha\beta\eta}$-equality is complete for the semantics we defined, i.e. that whenever $\mathcal{I}_\varphi(\mathbf{A})=\mathcal{I}_\varphi(\mathbf{B})$ for all variable assignments φ, then $\mathbf{A}=_{\alpha\beta\eta}\mathbf{B}$. We will prove this by a model existence argument: we will construct a model $\mathcal{M}:=\langle\mathcal{D},\mathcal{I}\rangle$ such that if $\mathbf{A}\neq_{\alpha\beta\eta}\mathbf{B}$ then $\mathcal{I}_\varphi(\mathbf{A})\neq\mathcal{I}_\varphi(\mathbf{B})$ for some φ.

As in other completeness proofs, the model we will construct is a "ground term model", i.e. a model where the carrier (the frame in our case) consists of ground terms. But in the λ-calculus, we have to do more work, as we have a non-trivial built-in equality theory; we will construct the "ground term model" from sets of normal forms. So we first fix some notations for them.

Normal Forms in the simply typed λ-calculus

▷ **Definition 1.3.60.** We call a term $\mathbf{A}\in wff_{\mathcal{T}}(\Sigma_{\mathcal{T}},\mathcal{V}_{\mathcal{T}})$ a β normal form iff there is no $\mathbf{B}\in wff_{\mathcal{T}}(\Sigma_{\mathcal{T}},\mathcal{V}_{\mathcal{T}})$ with $\mathbf{A}\rightarrow_{\beta}\mathbf{B}$.

We call $\mathbf{N}$ a β normal form of $\mathbf{A}$, iff $\mathbf{N}$ is a β-normal form and $\mathbf{A}\rightarrow_{\beta}\mathbf{N}$.

We denote the set of β-normal forms with $wff_{\mathcal{T}}(\Sigma_{\mathcal{T}},\mathcal{V}_{\mathcal{T}})\downarrow_{\beta\eta}$.

▷ We have just proved that $\beta\eta$-reduction is terminating and confluent, so we have

▷ **Corollary 1.3.61 (Normal Forms).** *Every $\mathbf{A}\in wff_{\mathcal{T}}(\Sigma_{\mathcal{T}},\mathcal{V}_{\mathcal{T}})$ has a unique β normal form ($\beta\eta$, long $\beta\eta$ normal form), which we denote by $\mathbf{A}\downarrow_{\beta}$ ($\mathbf{A}\downarrow_{\beta\eta}$ $\mathbf{A}\downarrow_{\beta\eta^l}$)*

The term frames will be a quotient spaces over the equality relations of the λ-calculus, so we introduce this construction generally.

Frames and Quotients

▷ **Definition 1.3.62.** Let $\mathcal{D}$ be a frame and $\sim$ a typed equivalence relation on $\mathcal{D}$, then we call $\sim$ a congruence on $\mathcal{D}$, iff $f\sim f'$ and $g\sim g'$ imply $f(g)\sim f'(g')$.

▷ **Definition 1.3.63.** We call a congruence $\sim$ functional, iff for all $f,g\in\mathcal{D}_{(\alpha\rightarrow\beta)}$ the fact that $f(a)\sim g(a)$ holds for all $a\in\mathcal{D}_{\alpha}$ implies that $f\sim g$.

▷ **Example 1.3.64.** $=_{\beta}$ ($=_{\beta\eta}$) is a (functional) congruence on $cwff_{\mathcal{T}}(\Sigma_{\mathcal{T}})$ by definition.

▷ **Theorem 1.3.65.** *Let $\mathcal{D}\mathcal{T}$ be a $\Sigma_{\mathcal{T}}$-frame and $\sim$ a functional congruence on $\mathcal{D}$, then the quotient space $\mathcal{D}/\sim$ is a $\Sigma_{\mathcal{T}}$-frame.*

▷ *Proof:*

1. $\mathcal{D}/\sim=\{f_{\sim}|f\in\mathcal{D}\}$, define $f_{\sim}(a_{\sim}):=f(a)_{\sim}$.
2. This only depends on equivalence classes: Let $f'\in f_{\sim}$ and $a'\in a_{\sim}$.
3. Then $f(a)_{\sim}=f'(a)_{\sim}=f'(a')_{\sim}=f(a')_{\sim}$
4. To see that we have $f_{\sim}=g_{\sim}$, iff $f\sim g$, iff $f(a)=g(a)$ since $\sim$ is functional.
5. This is the case iff $f(a)_{\sim}=g(a)_{\sim}$, iff $f_{\sim}(a_{\sim})=g_{\sim}(a_{\sim})$ for all $a\in\mathcal{D}_{\alpha}$ and thus for all $a_{\sim}\in\mathcal{D}/\sim$.

To apply this result, we have to establish that $=_{\beta\eta}$-equality is a functional congruence. We first establish $=_{\beta\eta}$ as a functional congruence on $wff_{\mathcal{T}}(\Sigma_{\mathcal{T}},\mathcal{V}_{\mathcal{T}})$ and then specialize this result to show that is is also functional on $cwff_{\mathcal{T}}(\Sigma_{\mathcal{T}})$ by a grounding argument.

$\beta\eta$-Equivalence as a Functional Congruence

▷ **Lemma 1.3.66.** *$\beta\eta$-equality is a functional congruence on $wff_{\mathcal{T}}(\Sigma_{\mathcal{T}},\mathcal{V}_{\mathcal{T}})$.*

▷ *Proof:* Let $\mathbf{A}\ \mathbf{C}=_{\beta\eta}\mathbf{B}\ \mathbf{C}$ for all $\mathbf{C}$ and $X\in(\mathcal{V}_{\gamma}\backslash\mathrm{free}(\mathbf{A})\cup\mathrm{free}(\mathbf{B}))$.

1. then (in particular) $\mathbf{A}\ X=_{\beta\eta}\mathbf{B}\ X$, and

2. $(\lambda X.\mathbf{A}\ X) =_{\beta\eta} (\lambda X.\mathbf{B}\ X)$, since $\beta\eta$-equality acts on subterms.
3. By definition we have $\mathbf{A} =_{\eta} (\lambda X_\alpha.\mathbf{A}\ X) =_{\beta\eta} (\lambda X_\alpha.\mathbf{B}\ X) =_{\eta} \mathbf{B}$.

▷ **Definition 1.3.67.** We call an injective substitution $\sigma\colon \mathrm{free}(\mathbf{C}) \to \Sigma_{\mathcal{T}}$ a grounding substitution for $\mathbf{C} \in \mathit{wff}_{\mathcal{T}}(\Sigma_{\mathcal{T}}, \mathcal{V}_{\mathcal{T}})$, iff no $\sigma(X)$ occurs in $\mathbf{C}$.

▷ **Observation:** They always exist, since all Σ_α are infinite and $\mathrm{free}(\mathbf{C})$ is finite.

▷ **Theorem 1.3.68.** *$\beta\eta$-equality is a functional congruence on $\mathit{cwff}_{\mathcal{T}}(\Sigma_{\mathcal{T}})$.*

▷ *Proof:* We use **??**

1. Let $\mathbf{A}, \mathbf{B} \in \mathit{cwff}_{(\alpha\to\beta)}(\Sigma_{\mathcal{T}})$, such that $\mathbf{A} \neq_{\beta\eta} \mathbf{B}$.
2. As $\beta\eta$ is functional on $\mathit{wff}_{\mathcal{T}}(\Sigma_{\mathcal{T}}, \mathcal{V}_{\mathcal{T}})$, there must be a $\mathbf{C}$ with $\mathbf{A}\ \mathbf{C} \neq_{\beta\eta} \mathbf{B}\ \mathbf{C}$.
3. Now let $\mathbf{C}' := \sigma(\mathbf{C})$, for a grounding substitution σ.
4. Any $\beta\eta$ conversion sequence for $\mathbf{A}\ \mathbf{C}' \neq_{\beta\eta} \mathbf{B}\ \mathbf{C}'$ induces one for $\mathbf{A}\ \mathbf{C} \neq_{\beta\eta} \mathbf{B}\ \mathbf{C}$.
5. Thus we have shown that $\mathbf{A} \neq_{\beta\eta} \mathbf{B}$ entails $\mathbf{A}\ \mathbf{C}' \neq_{\beta\eta} \mathbf{B}\ \mathbf{C}'$.

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG Kohlhase & Rabe: KRMT 135 2023-04-25

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

This completes the preparation and we can define the notion of a term algebra, i.e. a $\Sigma_{\mathcal{T}}$-algebra whose frame is made of $=_{\beta\eta}$-normal λ-terms.

A Herbrand Model for $\Lambda^{\to}$

▷ **Definition 1.3.69.** We call $\mathcal{T}_{\beta\eta} := \langle cwff_{\mathcal{T}}(\Sigma_{\mathcal{T}})\big\downarrow_{\beta\eta}, \mathcal{I}^{\beta\eta}\rangle$ the Σ term algebra, if $\mathcal{I}^{\beta\eta} = \mathrm{Id}_{\Sigma_{\mathcal{T}}}$.

▷ The name "term algebra" in the previous definition is justified by the following

▷ **Theorem 1.3.70.** *$\mathcal{T}_{\beta\eta}$ is a $\Sigma_{\mathcal{T}}$-algebra*

▷ *Proof:* We use the work we did above

1. Note that $cwff_{\mathcal{T}}(\Sigma_{\mathcal{T}})\big\downarrow_{\beta\eta} = \mathit{cwff}_{\mathcal{T}}(\Sigma_{\mathcal{T}})/{=_{\beta\eta}}$ and thus a $\Sigma_{\mathcal{T}}$-frame by **??** and **??**.
2. So we only have to show that the value function $\mathcal{I}^{\beta\eta} = \mathrm{Id}_{\Sigma_{\mathcal{T}}}$ is total.
3. Let φ be an assignment into $cwff_{\mathcal{T}}(\Sigma_{\mathcal{T}})\big\downarrow_{\beta\eta}$.
4. Note that $\sigma := \varphi|_{\mathrm{free}(\mathbf{A})}$ is a substitution, since $\mathrm{free}(\mathbf{A})$ is finite.
5. A simple induction on the structure of $\mathbf{A}$ shows that $\mathcal{I}^{\beta\eta}_\varphi(\mathbf{A}) = (\sigma(\mathbf{A}))\big\downarrow_{\beta\eta}$.
6. So the value function is total since substitution application is.

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG Kohlhase & Rabe: KRMT 136 2023-04-25

And as always, once we have a term model, showing completeness is a rather simple exercise. We can see that $\alpha\beta\eta$-equality is complete for the class of $\Sigma_{\mathcal{T}}$-algebras, i.e. if the equation $\mathbf{A} = \mathbf{B}$ is valid, then $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$. Thus $\alpha\beta\eta$ equivalence fully characterizes equality in the class of all $\Sigma_{\mathcal{T}}$-algebras.

Completetness of $\alpha\beta\eta$-Equality

▷ **Theorem 1.3.71.** $\mathbf{A} = \mathbf{B}$ *is valid in the class of* $\Sigma_{\mathcal{T}}$*-algebras, iff* $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$.

▷ *Proof:* For $\mathbf{A}$, $\mathbf{B}$ closed this is a simple consequence of the fact that $\mathcal{T}_{\beta\eta}$ is a $\Sigma_{\mathcal{T}}$-algebra.

1. If $\mathbf{A} = \mathbf{B}$ is valid in all $\Sigma_{\mathcal{T}}$-algebras, it must be in $\mathcal{T}_{\beta\eta}$ and in particular $\mathbf{A}{\downarrow_{\beta\eta}} = \mathcal{I}^{\beta\eta}(\mathbf{A}) = \mathcal{I}^{\beta\eta}(\mathbf{B}) = \mathbf{B}{\downarrow_{\beta\eta}}$ and therefore $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$.

If the equation has free variables, then the argument is more subtle.

2. Let σ be a grounding substitution for $\mathbf{A}$ and $\mathbf{B}$ and φ the induced variable assignment.
3. Thus ${\mathcal{I}^{\beta\eta}}_{\varphi}(\mathbf{A}) = {\mathcal{I}^{\beta\eta}}_{\varphi}(\mathbf{B})$ is the $\beta\eta$-normal form of $\sigma(\mathbf{A})$ and $\sigma(\mathbf{B})$.
4. Since φ is a structure preserving homomorphism on well-formed formulae, $\varphi^{-1}({\mathcal{I}^{\beta\eta}}_{\varphi}(\mathbf{A}))$ is the is the $\beta\eta$-normal form of both $\mathbf{A}$ and $\mathbf{B}$ and thus $\mathbf{A} =_{\alpha\beta\eta} \mathbf{B}$.

?? and **??** complete our study of the semantics of the simply-typed λ-calculus by showing that it is an adequate logic for modeling (the equality) of functions and their applications.

1.3.6 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 subsection 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 1.3.72 (Formally).**

Raw λ-terms with de Bruijn indices are expressions given by changing the last production in Definition 1.3.48 to

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

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 1.3.73.** $(\lambda x.\lambda y.z\ x\ (\lambda u.u\ x))\ (\lambda w.w\ x)$, becomes $(\lambda\lambda 4\ 2\ (\lambda 1\ 3))\ (\lambda 5\ 1)$,

▷ **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 1.3.74.** For β-reducing $(\lambda\mathbf{M})\ \mathbf{N}$ we must:

1. find variable occurrences n_1, n_2, ..., 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 1.3.75.** We perform the steps outlined above on $(\lambda\lambda 4\ 2\ (\lambda 1\ 3))\ (\lambda 5\ 1)$:

1. we obtain $\lambda 4\ n_1\ (\lambda 1\ n_2)$
2. we obtain $\lambda 3\ n_1\ (\lambda 1\ n_2)$ decrementing free variables.
3. we replace X with the argument $\lambda 5\ 1$.
   - ▷ n_1 is under one $\lambda \leadsto$ replace it with $\lambda 6\ 1$
   - ▷ n_2 is under two λs $\leadsto$ replace it with $\lambda 7\ 1$.

The final result is $\lambda 3\ (\lambda 6\ 1)\ (\lambda 1\ (\lambda 7\ 1))$

1.3.7 Simple Type Theory

In this subsection we will revisit the higher-order predicate logic introduced in subsection 1.3.1 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 prop for truth values

▷ **Definition 1.3.76.** We call a Σ-algebra $\langle\mathcal{D},\mathcal{I}\rangle$ a Henkin model, iff $\mathcal{D}_{\mathsf{prop}} = \{\mathsf{T},\mathsf{F}\}$.

▷ **Definition 1.3.77.** $\mathbf{A}_{\mathsf{prop}}$ valid under φ, iff $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$

▷ **Definition 1.3.78.** Connectives in Σ: $\neg\in\Sigma_{\mathsf{prop}\to\mathsf{prop}}$ and $\{\vee,\wedge,\Rightarrow,\Leftrightarrow,\ldots\}\subseteq\Sigma_{\mathsf{prop}\to\mathsf{prop}\to\mathsf{prop}}$ (with the intuitive $\mathcal{I}$-values)

▷ **Definition 1.3.79.** Quantifiers: $\Pi^\alpha\in\Sigma_{\alpha\to\mathsf{prop}\to\mathsf{prop}}$ with $\mathcal{I}(\Pi^\alpha)(p) = \mathsf{T}$, iff $p(a) = \mathsf{T}$ for all $a\in\mathcal{D}_\alpha$.

▷ **Definition 1.3.80.** [Quantified] formulae: $\forall X_\alpha.\mathbf{A}$ stands for $\Pi^\alpha\ (\lambda X_\alpha.\mathbf{A})$.

▷ $\mathcal{I}_\varphi(\forall X_\alpha.\mathbf{A}) = \mathcal{I}(\Pi^\alpha)(\mathcal{I}_\varphi(\lambda X_\alpha.\mathbf{A})) = \mathsf{T}$, iff $\mathcal{I}_{\varphi,[a/X]}(\mathbf{A}) = \mathsf{T}$ for all $a\in\mathcal{D}_\alpha$

▷ looks like PLΩ (Call any such system 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 1.3.81.** We assume logical constants Π^α and σ^α of type $\alpha \to \mathsf{prop} \to \mathsf{prop}$.

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 1.3.82.** 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: HOL^{∞}

▷ only one logical constant $q^\alpha\in\Sigma_{\alpha\to\alpha\to\mathsf{prop}}$ 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	$q^{\mathsf{prop}} = q^{\mathsf{prop}}$
D	F	for	$\lambda X_{\mathsf{prop}}.T = \lambda X_{\mathsf{prop}}.X_{\mathsf{prop}}$
D	Π^α	for	$q^{\alpha\to\mathsf{prop}}\ (\lambda X_\alpha.T)$
N	$\forall X_\alpha.\mathbf{A}$	for	$\Pi^\alpha\ (\lambda X_\alpha.\mathbf{A})$
D	$\wedge$	for	$\lambda X_{\mathsf{prop}}.\lambda Y_{\mathsf{prop}}.(\lambda G_{\mathsf{prop}\to\mathsf{prop}\to\mathsf{prop}}.GTT = \lambda G_{\mathsf{prop}\to\mathsf{prop}\to\mathsf{prop}}.GXY)$
N	$\mathbf{A} \wedge \mathbf{B}$	for	$\wedge\ (\mathbf{A}_{\mathsf{prop}})\ (\mathbf{B}_{\mathsf{prop}})$
D	$\Rightarrow$	for	$\lambda X_{\mathsf{prop}}.\lambda Y_{\mathsf{prop}}.(X = X \wedge Y)$
N	$\mathbf{A} \Rightarrow \mathbf{B}$	for	$\Rightarrow\ (\mathbf{A}_{\mathsf{prop}})\ (\mathbf{B}_{\mathsf{prop}})$
D	$\neg$	for	$q^{\mathsf{prop}}\ F$
D	$\vee$	for	$\lambda X_{\mathsf{prop}}.\lambda Y_{\mathsf{prop}}.\neg(\neg X \wedge \neg Y)$
N	$\mathbf{A} \vee \mathbf{B}$	for	$\vee\ (\mathbf{A}_{\mathsf{prop}})\ (\mathbf{B}_{\mathsf{prop}})$
D	$\exists X_\alpha.\mathbf{A}_{\mathsf{prop}}$	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, **??** does not hold (see [And72] for details). Indeed the proof of **??** 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.

Henkin's Theorem

▷ **Theorem 1.3.83 (Henkin's Theorem).** *Every $\mathcal{H}_\Omega$-consistent set of sentences has a model.*

▷ *Proof:*

1. Let Φ be a $\mathcal{H}_\Omega$-consistent set of sentences.
2. Extend Φ by adding sentences until Φ becoms a Hintikka set $\mathcal{H}$ with good closure properties.
3. Build a term Σ-algebra as a typed universe and interpret $TWFfcl\mathsf{prop}$ in $\mathcal{D}_{\mathsf{prop}}$ by setting $\mathcal{I}_\varphi(\mathbf{A}) = \mathsf{T}$, iff $\mathbf{A} \in \mathcal{H}$.

▷ **Theorem 1.3.84 (Completeness Theorem for $\mathcal{H}_\Omega$).** *If $\Phi \models \mathbf{A}$, then $\Phi \vdash_{\mathcal{H}_\Omega} \mathbf{A}$.*

Proof: We prove the result by playing with negations.

▷ 1. If $\mathbf{A}$ is valid in all models of Φ, then $\Phi \cup \{\neg \mathbf{A}\}$ has no model
2. Thus $\Phi \cup \{\neg \mathbf{A}\}$ is inconsistent by (the contrapositive of) Henkins Theorem.
3. So $\Phi \vdash_{\mathcal{H}_\Omega} \neg\neg \mathbf{A}$ by negation introduction and thus $\Phi \vdash_{\mathcal{H}_\Omega} \mathbf{A}$ by negation elimination.

Consequences of Henkin's Theorem

▷ **Theorem 1.3.85 (Compactness).** *If $\mathcal{H}\models\mathbf{A}$, then there is a finite $\mathcal{K}\subseteq\mathcal{H}$ with $\mathcal{K}\models\mathbf{A}$.*

▷ **Theorem 1.3.86 (Higher-Order Löwenheim/Skolem).** *If $\mathbf{A}$ is satisfiable, then there is a countable Henkin model $\mathcal{M}$ with $\mathcal{M}\models\mathbf{A}$.*

▷ **Corollary 1.3.87 (Skolem-Paradox).** *$\mathbb{R}$ is uncountable (by Cantor's theorem), but has a countable Henkin model.*

▷ **Problem:** Is there a contradiction?

▷ **Remark:** Look at the *exact* logical formulation of Cantor's theorem, what does that mean in terms of Henkin models!

▷ **Turns Out:** There is no contradiction in $\neg(\exists F:\mathbb{N}\to\mathbb{R}.F$ surjective$)$

- ▷ The non-existence of surjective functions only entails a cardinality difference for standard models.
- ▷ in Henkin models it only means that $\mathcal{D}_{(\alpha\to\beta)}$ contains no surjective functions.

▷ **Gödel Theorems:** There is no formal system that can distinguish between Henkin and standard models.

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 $f'=f$.

Are there Functions at all in Henkin Models?

▷ **In General:** All that can be written down! ($\Sigma_{\mathcal{T}}$-algebras are comprehension closed)

- ▷ Otherwise $\mathcal{D}_\alpha$ could be empty.
- ▷ $\mathcal{D}_{\mathsf{prop}}\neq\emptyset$, as $\mathcal{D}_{\mathsf{prop}}\supseteq\{\mathsf{T},\mathsf{F}\}$ as $\mathcal{I}_\varphi(\forall X_{\mathsf{prop}}.X\vee\neg X)=\mathsf{T}$ and $\mathcal{I}_\varphi(\forall X_{\mathsf{prop}}.X\wedge\neg X)=\mathsf{F}$.

▷ **What functions we write down?:**

- ▷ $\mathcal{D}_{(\alpha\to\alpha)}\neq\emptyset$, since $\mathcal{I}_\varphi(\lambda X_\alpha.X)\in\mathcal{D}_{(\alpha\to\alpha)}$.
- ▷ $\mathcal{D}_{(\mathsf{prop}\to\iota)}=\emptyset$, iff $\mathcal{D}_\iota=\emptyset$. ($\lambda X_{\mathsf{prop}}.Y_\iota$ does not help)

▷ **In General:** $\mathcal{D}_{(\alpha\to\beta)}=\emptyset$, sometimes! (Curry-Howard-Iso.)

▷ **Lambda-Definable Functions:**

- ▷ are always total (terminate on any input)
- ▷ e.g. on the natural numbers: $+,\cdot,\hat{}$ but not $-,/,\sqrt{}$

▷ **Idea:** Guarantee that $\mathcal{D}_\alpha\neq\emptyset$ by a constant $c\in\Sigma_\alpha$.

▷ **Problem:** But what are good constants that give us mathematically relevant function universes? (up next)

More Operators and Axioms for $\text{HOL}^{\rightarrow}$

- **Definition 1.3.88.** The unary conditional $\mathbf{w}^{\alpha} \in \Sigma_{\text{prop}\rightarrow\alpha\rightarrow\alpha}$
  $\mathbf{w}\ (\mathbf{A}_{\text{prop}})\mathbf{B}_{\alpha}$ means: "If $\mathbf{A}$, then $\mathbf{B}$".
- **Definition 1.3.89.** The binary conditional $\mathbf{if}^{\alpha} \in \Sigma_{\text{prop}\rightarrow\alpha\rightarrow\alpha\rightarrow\alpha}$
  $\mathbf{if}\ (\mathbf{A}_{\text{prop}})\ (\mathbf{B}_{\alpha})\ (\mathbf{C}_{\alpha})$ means: "if $\mathbf{A}$, then $\mathbf{B}$ else $\mathbf{C}$".
- **Definition 1.3.90.** The description operator $\iota^{\alpha} \in \Sigma_{\alpha\rightarrow\text{prop}\rightarrow\alpha}$
  if $\mathbf{P}$ is a singleton set, then $\iota\ (\mathbf{P}_{\alpha\rightarrow\text{prop}})$ is the (unique) element in $\mathbf{P}$.
- **Definition 1.3.91.** The choice operator $\gamma^{\alpha} \in \Sigma_{\alpha\rightarrow\text{prop}\rightarrow\alpha}$
  if $\mathbf{P}$ is non-empty, then $\gamma\ (\mathbf{P}_{\alpha\rightarrow\text{prop}})$ is an arbitrary element from $\mathbf{P}$.
- **Definition 1.3.92 (Axioms for these Operators).**
  - unary conditional: $\forall \varphi_{\text{prop}}. \forall X_{\alpha}. \varphi \Rightarrow \mathbf{w}\ \varphi X = X$
  - binary conditional: $\forall \varphi_{\text{prop}}. \forall X_{\alpha}, Y_{\alpha}, Z_{\alpha}. (\varphi \Rightarrow \mathbf{if}\ \varphi\ X\ Y = X) \wedge (\neg\varphi \Rightarrow \mathbf{if}\ \varphi\ Z\ X = X)$
  - description operator $\forall P_{\alpha\rightarrow\text{prop}}. (\exists^{1} X_{\alpha}. PX) \Rightarrow (\forall Y_{\alpha}. PY \Rightarrow \iota\ P = Y)$
  - choice operator $\forall P_{\alpha\rightarrow\text{prop}}. (\exists X_{\alpha}. PX) \Rightarrow (\forall Y_{\alpha}. PY \Rightarrow \gamma\ P = Y)$
- **Idea:** These operators ensure a much larger supply of functions in Henkin models.

More on the Description Operator

- ι ! is a weak form of the choice operator (only works on singleton sets)
- Alternative Axiom of Descriptions: $\forall X_{\alpha}. \iota^{\alpha} \;=\; X = X$.
  - use that $\mathcal{I}_{[\mathsf{a}/X]}(=\ X) = \{\mathsf{a}\}$
  - we only need this for base types $\neq$ prop
  - Define $\iota^{\text{prop}} := =\ (\lambda X_{\text{prop}}. X)$ or $\iota^{\text{prop}} := (\lambda G_{\text{prop}\rightarrow\text{prop}}. G\ T)$ or $\iota^{\text{prop}} := = =\ T$
  - $\iota^{(\alpha\rightarrow\beta)} := (\lambda H_{\alpha\rightarrow\beta\rightarrow\text{prop}} X_{\alpha}. \iota^{\beta}\ (\lambda Z_{\beta}. (\exists F_{\alpha\rightarrow\beta}. H\ F \wedge F\ X = Z)))$

1.4 Category Theory

Acknowledgement: The presentation of category theory below has been inspired by Daniele Turi's Category Lecture Notes [Tur01].

1.4.1 Introduction

The crucial observation for category theory is that we do very similar things when we define complex concepts, objects, or models. Here are some examples.

Common Structure to Mathematical Objects

▷ **Example 1.4.1.** Let A, B, and C be sets, and $f\colon A \to B$ and $g\colon B \to C$ be functions. Then $g \circ f$ is a function and we have functions Id_A and Id_B with $\mathsf{Id}_A \circ f = f = f \circ \mathsf{Id}_B$.

▷ **Example 1.4.2.** Let A, B, and C be topological spaces, and $f\colon A \to B$ and $g\colon B \to C$ be continuous functions. Then $g \circ f$, Id_A, and Id_B are continuous and $\mathsf{Id}_A \circ f = f = f \circ \mathsf{Id}_B$.

▷ **Example 1.4.3.** Let A, B, and C be posets, and $f\colon A \to B$ and $g\colon B \to C$ be monotone functions. Then $g \circ f$, Id_A, and Id_B are monotone and $\mathsf{Id}_A \circ f = f = f \circ \mathsf{Id}_B$.

▷ **Example 1.4.4.** Let A, B, and C be monoids, and $f\colon A \to B$ and $g\colon B \to C$ be monoid homomorphisms. Then $g \circ f$, Id_A, and Id_B are monoid homomorphisms and $\mathsf{Id}_A \circ f = f = f \circ \mathsf{Id}_B$.

Given the examples above – and there are hundreds more – it seem natural to try to find a common pattern, make that into a mathematical concept in its own right, and see what we can do in general with that.

Categories: The Definition

▷ **Definition 1.4.5.**

A category $\mathcal{C}$ consists of:

1. A class $\mathrm{ob}(\mathcal{C})$ of objects.
2. A class $\mathrm{Mor}_{\mathcal{C}}$ of arrows (also called morphism or map).
3. For each arrow f, two objects which are called domain $\mathrm{dom}(f)$ and codomain $\mathrm{cod}(f)$ of f. We write $f\colon \mathrm{dom}(f) \to \mathrm{cod}(f)$ and call two arrows f and g composable, iff $\mathrm{dom}(f) = \mathrm{cod}(g)$.
4. An associative operation $\circ$ called composition assigning to each pair (f,g) of composable arrows another arrow $g \circ f$ such that $\mathrm{dom}(g \circ f) = \mathrm{dom}(f)$ and $\mathrm{cod}(g \circ f) = \mathrm{cod}(g)$, i.e. $g \circ f\colon \mathrm{dom}(f) \to \mathrm{cod}(g)$.
5. For every object A an arrow $1_A\colon A \to A$ called the identity morphism, such that for any $f\colon A \to B$ we have $f \circ 1_A = f = 1_B \circ f$.

We write the class of arrows $f\colon A \to B$ as $\mathrm{Mor}_{\mathcal{C}}(A, B)$. The notations $\mathrm{Hom}_{\mathcal{C}}(A, B)$, $\mathcal{C}(A, B)$, $[A, B]_{\mathcal{C}}$, and $(A, B)_{\mathcal{C}}$ are also used.

▷ **Observation 1.4.6.** *Many classes of mathematical objects and their natural (structure-preserving) mappings form categories.*

▷ **Definition 1.4.7.** Category theory studies general properties of structures abstracting away from the concrete objects.

Actually we have already seen a few additional (and somewhat less classical) examples in the KRMT course itself.

Categories in KRMT

▷ **Remark:** We have already seen various examples of categories in KRMT

▷ **Example 1.4.8.** Types and functions in MMT/LF form a category.(abstract away from terms)

▷ **Example 1.4.9.** Contexts and substitutions in logics form a category:
A substitution σ induces a function from $\mathit{wff}(\Sigma, \Gamma \uplus \mathsf{supp}(\sigma))$ to $\mathit{wff}(\Sigma, \Gamma \uplus \mathsf{intro}(\sigma))$.

▷ **Example 1.4.10.** MMT theories and theory morphisms form a category:

A theory T defines a language (set of well typed terms) $\mathcal{L}_T$, and a theory morphism from S to T mapping between $\mathcal{L}_S$ and $\mathcal{L}_T$.

To get a feeling for the variety of categories, we will now discuss a couple of generic examples and a category constructor that will become useful later on.

Commonly used Categories

▷ **Definition 1.4.11.** The objects of the category of sets **Set** are sets and its arrows $f\colon A \to B$ are the functions.

▷ **Definition 1.4.12.** The objects of the category of topological spaces **Top** are topological spaces and its arrows are the continuous functions.

▷ **Definition 1.4.13.** A category $\mathcal{C}$ is called small (otherwise large), iff $\mathsf{ob}(\mathcal{C})$ and $\mathsf{Mor}_{\mathcal{C}}$ consist of sets (not classes).

▷ **Definition 1.4.14.** Let $\mathcal{C}$ be a category, then the opposite category (also called the dual category) $\mathcal{C}^{\mathsf{op}}$ is formed by reversing all the arrows of $\mathcal{C}$, i.e.

$$\mathsf{Mor}_{\mathcal{C}^{\mathsf{op}}} := \{f\colon B \to A \mid f\colon A \to B \in \mathsf{Mor}_{\mathcal{C}}\}$$

Just when we thought that we had reached the pinnacle of abstraction with categories, mathematics does it again, making categories themselves into objects and introducing a new concept for the corresponding arrows.

Functors

▷ **Definition 1.4.15.** Let $\mathcal{C}$ and $\mathcal{D}$ be categories, then a mapping F from $\mathcal{C}$ to $\mathcal{D}$ is called a (covariant) functor, iff F

- ▷ associates to each $X \in \mathrm{ob}(\mathcal{C})$ an object $F(X) \in \mathrm{ob}(\mathcal{D})$
- ▷ associates to each morphism $f\colon X \to Y \in \mathrm{Mor}_{\mathcal{C}}(X, Y)$ a morphism

$$F(f)\colon F(X) \to F(Y) \in \mathrm{Mor}_{\mathcal{D}}(F(X), F(Y))$$

such that the following two conditions hold:

- ▷ $F(1_X) = 1_{F(X)}$ for each $X \in \mathrm{ob}(\mathcal{C})$.
- ▷ $F(g \circ f) = F(g) \circ F(f)$ for all morphisms $f\colon X \to Y$ and $g\colon Y \to Z$ in $\mathcal{C}$.

That is, functors must preserve identity morphisms and morphism composition.

▷ **Definition 1.4.16.** The category of small categories (denoted as **Cat**) has all small categories as objects and functors as arrows.

▷ **Observation 1.4.17.** ***Cat*** *is itself a large category.*

1.4.2 Example/Motivation: Natural Numbers in Category Theory

We will now try to get an intution on how category theory "works", i.e. how we can work at the general level, i.e. the category theoretic level and apply the results down to all the concrete categories. This also serves as a motivation to the universal properties we will study in **??**.

For the construction of the natural number object, we will need a couple of category-theoretic concepts that we will only introduce in **??**; for now we will just (have to) take them on faith and come back to them later.

Lawvere's Natural Numbers Object

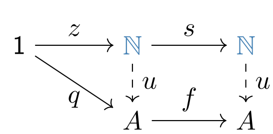
▷ **Recap:** In set theory, we define the natural numbers by the five Peano axioms about $\mathbb{N}$, $0 \in \mathbb{N}$, and $s\colon \mathbb{N} \to \mathbb{N}$.

▷ In category theory we can give a different answer! (need more terminology)

▷ **Definition 1.4.18.** A natural number object (NNO) in a (Cartesian closed) category E with terminal object 1 is an object $\mathbb{N}$ in E equipped with

- ▷ a morphism $z\colon 1 \to \mathbb{N}$ from the terminal object 1 (zero)
- ▷ a morphism $s\colon \mathbb{N} \to \mathbb{N}$ (successor)

such that for every other diagram $1 \xrightarrow{q} A \xrightarrow{f} A$ there is a unique morphism $u\colon \mathbb{N} \to A$ such that the following diagram commutes:



Natural Numbers $\widehat{=}$ natural number object in **Set**

▷ **Theorem 1.4.19.** *The natural number object in* ***Set*** *is isomorphic to Peano's* $\mathbb{N}$.

▷ Peano's $\mathbb{N}$ by the Recursion Theorem [ML86, §II.3].

▷ **Lemma 1.4.20.** *The natural number object* $\langle\mathbb{N}, z, s\rangle$ *in* ***Set*** *obeys Peano's axioms.*

▷ *Proof:*

1. For **P1** note that 1 in **Set** is a singleton set $\{a\}$, and any function $z\colon 1\to\mathbb{N}$ identifies an element $z(a)$ (let's call it z as well) in $\mathbb{N}$.
2. For **P2** note that s in **Set** is a function.
3. For **P3** assume $s(n) = z$ and consider a diagram $1 \xrightarrow{e} A \xrightarrow{f} A$ with $A = \{e, d\}$ and $u(e) = u(d) = d$. Then there is a function $f\colon \mathbb{N}\to A$ such that $f(z) = e$ and $f(s(n)) = u(f(n))$. But if $s(n) = z$ then $f(s(n)) = e \neq d = u(f(n))$.
4. Injectivity of s (**P4**) is left as an exercise.
5. **P5**, see ??

The Language of Diagrams

▷ **Definition 1.4.21.** A diagram in a category E is a directed graph, where the nodes are objects of E and the edges are arrows of E connecting the respective objects.

Diagrams often use dashed arrows to signify unique existence of arrows.

▷ **Definition 1.4.22.**

Let D be a diagram, then we say that D commutes (or is commuative), iff for any two paths $f_1, \ldots, f_n$ and $g_1, \ldots, g_m$ with the same start and end in D we have $f_n \circ \ldots \circ f_1 = g_m \circ \ldots \circ g_1$.

▷ **Example 1.4.23.**

Let $f\colon A\to B$, $g\colon A\to C$, $u\colon C\to D$, and $v\colon B\to D$ in a category $\mathcal{C}$, then we say that the diagram on the right commutes, iff $f \circ v = g \circ u$.

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ \downarrow g & & \downarrow v \\ C & \xrightarrow{u} & D \end{array}$$

▷ **Definition 1.4.24.**

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ & \searrow^{g} & \downarrow u \\ & & D \end{array}$$

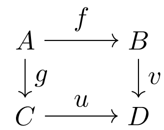
We treat the left diagram as an abbreviation of the right one.

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ \downarrow 1_A & & \downarrow u \\ A & \xrightarrow{g} & D \end{array}$$

Diagram Chase: the Proof Method in Category Theory

▷ **Definition 1.4.25 (Diagram Chase in Small Categories with Functions).**

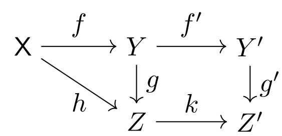
If $\mathcal{C}$ is small and f, g, u, and v are functions (e.g. in In **Set**), the diagram above commutes, iff the commutativity equation $v(f(a)) = u(g(a))$ holds for all $a \in A$.



We use the commutativity equation (and other properties of arrows) in the proof method of diagram chase (or diagrammatic search), which involves "chasing" elements around the diagram, until the desired element or result is constructed or verified.

▷ **Example 1.4.26.**

The diagram on the right commutes, iff $k(g(f(x))) = k(h(x)) = g'(f'(f(x)))$ for all $x \in X$.

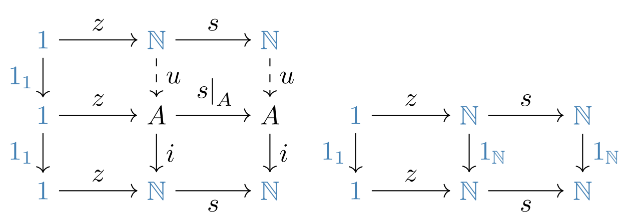


Natural Number Objects in **Set**: Induction

▷ **Lemma 1.4.27.** *The natural number object in **Set** is inductive: If $A \subseteq \mathbb{N}$ and from $z \in \mathbb{N}$ and $a \in A$ we obtain $s(a) \in A$ we obtain $A = \mathbb{N}$.*

▷ *Proof:* We translate the assumptions to diagrams and conduct a diagram chase.

1. We extend the NNO diagram with an inclusion function $i: A \to \mathbb{N}$ that corresponds to $A \subseteq \mathbb{N}$. Note that every cell commutes in the diagram on the left.



   Note that $s|_A : A \to A$ as $a \in A$ implies $s(a) \in A$. (induction step assumption)
2. Trivially, also the diagram on the right commutes, so by uniqueness in NNO, we have $i \circ u = 1_{\mathbb{N}}$.
3. Given two composable functions f and g, if $f \circ g$ is the identity, then f is injective.
4. So $U: \mathbb{N} \to A$ is injective, in other words: $\mathbb{N} \subseteq A$, and thus $A = \mathbb{N}$.

Uniqueness of Natural Numbers

▷ **Theorem 1.4.28.** *The natural number object is uniquely determined up to isomorphism in a category.*

▷ *Proof:* We prove that if there is another diagram $1 \xrightarrow{z'} \mathbb{N}' \xrightarrow{s'} \mathbb{N}'$, then $\mathbb{N}$ and $\mathbb{N}'$ are isomorphic.

1. We show that there are functions $f\colon \mathbb{N} \to \mathbb{N}'$ and $f'\colon \mathbb{N}' \to \mathbb{N}$, such that $f \circ f' = \mathsf{Id}_{\mathbb{N}'}$ and $f' \circ f = \mathsf{Id}_{\mathbb{N}}$.
2. We have the following two commuting diagrams

$$\begin{array}{ccccc} 1 & \xrightarrow{z} & \mathbb{N} & \xrightarrow{s} & \mathbb{N} \\ {\scriptstyle 1_1}\downarrow & & \downarrow{\scriptstyle f} & & \downarrow{\scriptstyle f} \\ 1 & \xrightarrow{z'} & \mathbb{N}' & \xrightarrow{s'} & \mathbb{N}' \\ {\scriptstyle 1_1}\downarrow & & \downarrow{\scriptstyle f'} & & \downarrow{\scriptstyle f'} \\ 1 & \xrightarrow{z} & \mathbb{N} & \xrightarrow[s]{} & \mathbb{N} \end{array} \qquad \begin{array}{ccccc} 1 & \xrightarrow{z} & \mathbb{N} & \xrightarrow{s} & \mathbb{N} \\ {\scriptstyle 1_1}\downarrow & & \downarrow{\scriptstyle 1_{\mathbb{N}}} & & \downarrow{\scriptstyle 1_{\mathbb{N}}} \\ 1 & \xrightarrow{z} & \mathbb{N} & \xrightarrow[s]{} & \mathbb{N} \end{array}$$

   The left one comes from the universal property of $1 \xrightarrow{z} \mathbb{N} \xrightarrow{s} \mathbb{N}$ and $1 \xrightarrow{z'} \mathbb{N}' \xrightarrow{s'} \mathbb{N}'$, the right one by construction. hence $f' \circ f = 1_{\mathbb{N}}$.
3. We obtain $f \circ f' = 1_{\mathbb{N}'}$ by a similar argument.

1.4.3 Universal Constructions in Category Theory

Now that we have seen how category theory "works" (i.e. how we can work with categories), we can now make good on the promise to introduce all the specific concepts we used in the construction of the natural numbers object. And while we are at it, we will also introduce other universal constructions that are often used in category theory and have inspired our developments in KRMT.

Initial and Terminal Objects

▷ **Definition 1.4.29.** Let $\mathcal{C}$ be a category, then we call an object $I \in \mathsf{ob}(\mathcal{C})$ initial (also cofinal or universal and written as 0), iff for every $X \in \mathsf{ob}(\mathcal{C})$ there is exactly one arrow $a\colon I \to X$. If every arrow into I is an isomorphism, then I is called strict initial object.

Definition 1.4.30. An object $T \in \mathsf{ob}(\mathcal{C})$ is called terminal or final, iff for every $X \in \mathsf{ob}(\mathcal{C})$ there is exactly one arrow $a\colon X \to T$. A terminal object is also called a terminator and write it as 1.

▷ **Observation 1.4.31.** *Initial and terminal objects are unique up to isomorphism, if they exist at all.* *(they need not exist in all categories)*

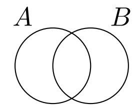
▷ **Example 1.4.32.** In **Set** the initial object is the empty set, while the terminal object is the (unique up to isomorphism) singleton set.

▷ **Remark:** We can think of the initial and terminal objects the category-theoretic generalizations ("universal characterizations") of the empty and singleton sets: they are characterized by objects and arrows only.

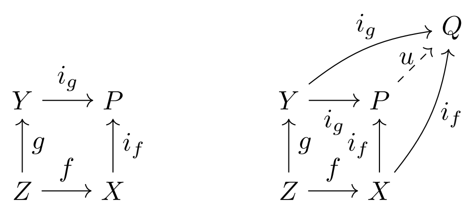
Pushouts: Unions on Steroids

▷ **Question:** Can we also characterize operations like union universally?

▷ **Idea:** In $A \cup B$, we use $A \cap B$ twice.
We have $A \cap B \subseteq A$ and $A \cap B \subseteq B$, which we can express with arrows (inclusions) $A \cap B \xhookrightarrow{\iota A} A$ and $A \cap B \xhookrightarrow{\iota B} B$. Similarly we have $A \subseteq A \cup B$ and $B \subseteq A \cup B$ which we express as $A \xhookrightarrow{\iota A} A \cup B$ and $B \xhookrightarrow{\iota B} A \cup B$.



▷ **Definition 1.4.33.** Let $\mathcal{C}$ be a category, then the pushout of morphisms $f\colon Z \to X$ and $g\colon Z \to Y$ consists of an object P together with two morphisms $i_f\colon X \to P$ and $i_g\colon Y \to P$, such that the left diagram below commutes and that $\langle P, i_f, i_g \rangle$ is universal with respect to this diagram – i.e., for any other such set $\langle Q, i_f, i_g \rangle$ for which the following diagram commutes, there must exist a unique $u\colon P \to Q$ also making the diagram commute, i.e.



Pushouts in Set

▷ As with all universal constructions, the pushout, if it exists, is unique up to a unique isomorphism.

▷ If X, Y, and Z are sets, and $f\colon Z \to X$ and $g\colon Z \to Y$ are function, then the pushout of f and g is the disjoint union $X \uplus Y$, where elements sharing a common preimage (in Z) are identified, i.e. $P = (X \uplus Y)/\sim$, where $\sim$ is the finest equivalence relation such that $\iota_1(f(z)) \sim \iota_2(g(z))$.

▷ **In particular:** if $X, Y \subseteq W$ for some larger set W, $Z = X \cap Y$, and f and g the inclusions of Z into X and Y, then the pushout can be canonically identified with $X \cup Y$.

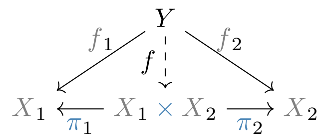
Product Objects and Exponentials in Categories

▷ **Question:** Can we also characterize functions (function spaces) in categories?

▷ **Idea:** Functions are sets of pairs with additional properties (left totality and right uniqueness)

▷ **Definition 1.4.34.** Let $\mathcal{C}$ be a category and $X_1, X_2 \in \mathrm{ob}(\mathcal{C})$. Then we call an object X together with two morphisms $\pi_1\colon X \to X_1$ and $\pi_2\colon X \to X_2$ the product of X_1 and X_2 and write it as $X_1 \times X_2$ if it satisfies the following universal property:

For every object Y and pair of morphisms $f_1 \colon Y \to X_1$ and $f_2 \colon Y \to X_2$ there exists a unique morphism $f \colon Y \to X_1 \times X_2$ such that the diagram on the right commutes:



The unique morphism f is called the product of morphisms f_1 and f_2 and is denoted $\langle f_1, f_2 \rangle$. The morphisms π_1 and π_2 are called the (canonical) projection or projection morphism.

Products in Set and Top

▷ **Example 1.4.35.** In **Set**, the product is the Cartesian product: Given sets X_1 and X_2, then we have the projections $\pi_i \colon X_1 \times X_2 \to X_i$. Given any set Y with functions $f_i \colon Z \to X_i$, the universal arrow f is defined as $f \colon Y \to X_1 \times X_2 ; y \mapsto \langle f_1(y), f_1(y) \rangle$.

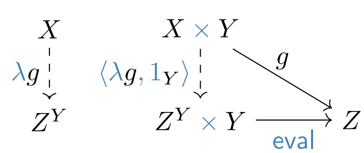
▷ **Example 1.4.36.**

In **Top**, the product of two objects is the product topology.

Exponentials in Categories

▷ **Definition 1.4.37.** If $A \times B$ exists for all objects A and B in a category $\mathcal{C}$, then we say that $\mathcal{C}$ has all binary products.

▷ **Definition 1.4.38.** Let $\mathcal{C}$ be a category that has all binary products and $Z, Y \in \mathrm{ob}(\mathcal{C})$, then we call an object Z^Y together with a morphism $\mathsf{eval} \colon Z^Y \times Y \to Z$ is called an exponential object, iff for any $X \in \mathrm{ob}(\mathcal{C})$ and $g \colon X \times Y \to Z \in \mathrm{Mor}_{\mathcal{C}}$ there is a unique morphism $\lambda g \colon X \to Z^Y$ (called the transpose of g) such that the following diagram commutes:



▷ **Lemma 1.4.39.** *In **Set**, $Z^Y = Y \to Z$ and $\mathsf{eval} \colon Z^Y \times Y \to Z ; (f, y) \mapsto f(y)$. For any map $g \colon X \times Y \to Z$ the map $\lambda g \colon X \to Z^Y$ is the Curried form of g: $\lambda g(x)(y) = g(x, y)$.*

Cartesian Closed Categories

▷ **Definition 1.4.40.** A category $\mathcal{C}$ is called Cartesian closed (a CCC), iff it satisfies the following three properties:

- ▷ $\mathcal{C}$ has a terminal object.
- ▷ Any two objects X and Y of $\mathcal{C}$ have a product $X \times Y$ in $\mathcal{C}$.
- ▷ Any two objects Y and Z of $\mathcal{C}$ have an exponential Z^Y in $\mathcal{C}$.

1.5 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 section 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 section is delightfully covered in [Dox+09].

1.5.1 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 1.5.1.** A set is "everything that can form a unity in the face of God". (Georg Cantor ($*1845$, $\dagger 1918$))
- **Example 1.5.2.** (determination by elementhood relation $\in$)
  - "the set that consists of the number 7 and the prime divisors of 510510"
  - $\{7, c\}$, $\{1, 2, 3, 4, 5n, \ldots\}$, $\{x | x \text{ is an integer}\}$, $\{X | \mathbf{P}(X)\}$
- **Questions (extensional/intensional):**
  - If $c = 7$, is $\{7, c\} = \{7\}$?
  - Is $\{X | X \in \mathbb{N}, X \neq X\} = \{X | X \in \mathbb{N}, X^2 < 0\}$?
  - 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: David Hilbert famously exclaimed "No one shall expel us from the Paradise that Cantor has created" in [Hil26, p. 170], 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: $\Sigma{:=}\{\in\ldots\}$ (sets are individuals)
- ▹ Extensionality: $\forall M,N.M=N\Leftrightarrow(\forall X.(X{\in}M)\Leftrightarrow(X{\in}N))$ (two sets are equal, iff they have the same elements)
- ▹ Comprehension: $\exists M.\forall X.(X{\in}M)\Leftrightarrow\mathbf{E}$ (all sets that we can write down exist)
- ▹ **Note**: The comprehension axiom is 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 1.5.3 (The set of all set and friends).**
$\{M|M \text{ set}\}$, $\{M|M \text{ set}, M{\in}M\}$, …

▷ **Definition 1.5.4 (Problem).** Russell's Antinomy:

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

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

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

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

▷ **Solutions: Define away the problems:**

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

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

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

▷ **Theorem 1.5.5.** $\forall M, N.(M \subseteq N) \wedge (N \subseteq M) \Rightarrow M = N$

▷ **Theorem 1.5.6.** *M is uniquely determined in* **Sep**

Proof sketch: With **Ext**

▷ **Notation:** 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 1.5.7.** *There is no universal set.*
- *Proof:*
  1. For each set M, there is a set $M_R{:=}\{X{\in}M\,|\,X{\notin}X\}$ by **Sep**.
  2. Show $\forall M.M_R{\notin}M$.
  3. If $M_R{\in}M$, then $M_R{\notin}M_R$, (also if $M_R{\notin}M$)
  4. Thus $M_R{\notin}M$ or $M_R{\in}M_R$.
- **Intuition:** 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?

- **Question:** Are there Interesting Sets at all?
- **Answer:** 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 $\emptyset{:=}\{X{\in}M\,|\,X\neq X\}$.
  - $\emptyset$ is empty and uniquely determined by **Ext**.
- **Even more:** Intersections: $M\cap N{:=}\{X{\in}M\,|\,X{\in}N\}$
- **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? $\leadsto$ Nicolas Bourbaki

- Is it possible to develop all of Mathematics from set theory?
  $\leadsto$ N. Bourbaki: Éléments de Mathématique~~s~~ (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 1.5.8 (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 1.5.9.** $M \cup N{:=}\{X{\in}W \mid X{\in}M \vee X{\in}N\}$ (exists by **Sep**.)
- **Axiom 1.5.10 (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 1.5.11.** $(\bigcup M){:=}\{X \mid \exists Y.Y{\in}M \wedge X{\in}Y\}$ (exists by **Sep**.)
- This also gives us intersections over families (without another axiom):
- **Definition 1.5.12.**

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

In Definition 1.5.12 we note that $\bigcup\mathbf{Ax}$ also guarantees us intersection over families. Note that we could not have defined that in analogy to **??** 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 1.5.13 (Power Set Axiom).** *For each set M there is a set W that contains all subsets of M: $\wp Ax := (\forall M. \exists W. \forall X. (X \subseteq M) \Rightarrow X \in W)$*
- **Definition 1.5.14.** Power Set: $\wp(M) := \{X \mid X \subseteq M\}$ (Exists by $\mathbf{Sep}$.)
- **Definition 1.5.15.** Singleton set: $\{X\} := \{Y \in \wp(X) \mid X = Y\}$
- **Axiom 1.5.16 (Pair Set (Axiom)).** *(is often assumed instead of $\cup\mathbf{Ax}$)*
  Given sets M and N there is a set W that contains exactly the elements M and N: $\forall M, N. \exists W. \forall X. (X \in W) \Leftrightarrow ((X = N) \vee (X = M))$
- Is derivable from $\wp$Ax: $\{M, N\} := \{M\} \cup \{N\}$.
- **Definition 1.5.17 (Finite Sets).** $\{X, Y, Z\} := \{X, Y\} \cup \{Z\} \ldots$
- **Theorem 1.5.18.** $\forall Z, X_1, \ldots, X_n. (Z \in \{X_1, \ldots, X_n\}) \Leftrightarrow (Z = X_1 \vee \ldots \vee Z = X_n)$

The Foundation Axiom

- **Axiom 1.5.19 (The Foundation Axiom $\mathbf{Fund}$).**
  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 1.5.20.** *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 1.5.21.** $\mathbf{Fund}$ guarantees a hierarchical structure (von Neumann Hierarchy) of the universe.
  1. 0. order: $\emptyset$,
  2. 1. order: $\{\emptyset\}$,
  3. 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
  - e.g. $\emptyset$, $\{\emptyset\}$, $\{\{\emptyset\}\}$, ... (iterated singleton set)
  - or $\emptyset$, $\{\emptyset\}$, $\{\emptyset, \{\emptyset\}\}$, ... (iterated pair set)

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

- **Axiom 1.5.22 (Infinity Axiom $\infty\mathbf{Ax}$).** *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 1.5.23.** M is inductive: $\mathbf{Ind}(M){:=}\emptyset{\in}M{\wedge}(\forall Z.Z{\in}M{\Rightarrow}(Z{\cup}\{Z\}){\in}M)$.
- **Definition 1.5.24.** Set of the Inductive Set: $\omega{:=}\{Z|\forall W.\mathbf{Ind}(W) \Rightarrow Z{\in}W\}$
- **Theorem 1.5.25.** *ω is inductive.*

The Replacement Axiom

- We have ω, $\wp(M)$, but not $\{\omega, \wp(\omega), \wp(\wp(\omega)), \ldots\}$.
- **Axiom 1.5.26 (The Replacement Axiom (Schema): $\mathbf{Rep}$).** *If for each X there is exactly one Y with property $\mathbf{P}(X,Y)$, then for each set U, that contains these X, there is a set V that contains the respective Y.* $(\forall X.\exists^1 Y.\mathbf{P}(X,Y)) \Rightarrow (\forall U.\exists V.\forall X,Y.X{\in}U \wedge \mathbf{P}(X,Y) \Rightarrow Y{\in}V)$
- **Intuitively:** A right-unique property $\mathbf{P}$ induces a replacement $\forall U.\exists V.V = \{F(X)|X{\in}U\}$.
- **Example 1.5.27.** Let $U = \{1, \{2,3\}\}$ and $\mathcal{P}(X \Leftrightarrow Y) \Leftrightarrow (\forall Z.Z{\in}Y \Rightarrow Z = X)$, then the induced function F maps each X to the set V that contains X, i.e. $V = \{\{X\}|X{\in}U = \{\{1\}, \{\{2,3\}\}\}\}$.

Zermelo Fraenkel Set Theory

- **Definition 1.5.28 (Zermelo Fraenkel Set Theory).** We call the first-order theory given by the axioms below Zermelo/Fraenkel set theory and denote it by $\mathbf{ZF}$.

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 1.5.29 (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.
$\forall X,Y,Z.Y{\in}X \wedge Z{\in}X \Rightarrow ((Y \neq \emptyset) \wedge (Y = Z \vee Y \cap Z = \emptyset) \Rightarrow (\exists.\forall.V{\in}X \Rightarrow (\exists.U \cap V = \{\})))$

▷ 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 1.5.30 (Zermelo Fraenkel Set Theory with Choice).**
The theory **ZF** together with **AC** is called **ZF** with choice and denoted as **ZFC**.

1.5.3 ZFC Applications

Limits of ZFC

▷ There is no set whose cardinality is strictly between that of integers and real numbers.

▷ **Theorem 1.5.31.**
*If **ZFC** is consistent, then neither **CH** nor $\neg$**CH** can be derived.* *(**CH** is independent of **ZFC**)*

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

▷ There are other examples like this.

Ordered Pairs

- **Empirically:** In **ZFC** we can define all mathematical concepts.
- **For Instance:** We would like a set that behaves like an odererd pair.
- **Definition 1.5.32.** Define $\langle X,Y\rangle := \{\{X\},\{X,Y\}\}$
- **Lemma 1.5.33.** $\langle X,Y\rangle = \langle U,V\rangle \Rightarrow X = U \wedge Y = V$
- **Lemma 1.5.34.** $U\in X \wedge V\in Y \Rightarrow \langle U,V\rangle \in \wp(\wp(X\cup Y))$
- **Definition 1.5.35.** 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 1.5.36.** Right projection π_r analogous.

Relations

- All mathematical objects are represented by sets in **ZFC**, in particular relations
- **Definition 1.5.37.** The Cartesian product of X and Y
  $X\times Y := \{Z\in\wp(\wp(X\cup Y)) \mid Z \text{ is ordered pair with } \pi_l(Z)\in X \wedge \pi_r(Z)\in Y\}$
  A relation is a subset of a Cartesian product.
- **Definition 1.5.38.** The domain and codomain of a function are defined as usual:

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

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

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

Functions

- **Definition 1.5.39.** 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 1.5.40.** 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.

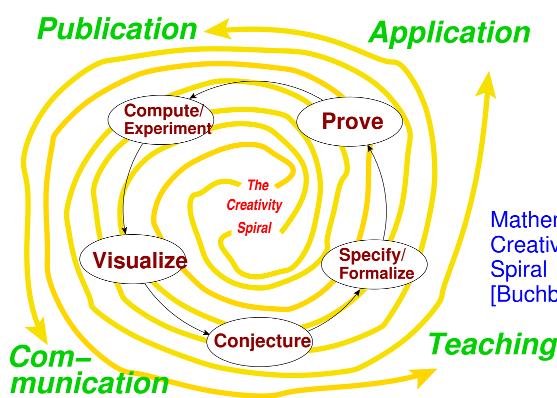
Chapter 2

Aspects of Knowledge Representation for Mathematics

2.1 Project Tetrapod

The way we do math will change dramatically

▷ **Definition 2.1.1 (Doing Math).** Buchberger's Math creativity spiral



Mathematical
Creativity
Spiral
[Buchberger 1995]

▷ Every step will be supported by mathematical software systems

▷ Towards an infrastructure for web-based mathematics!

Knowledge Representation is only Part of "Doing Math"

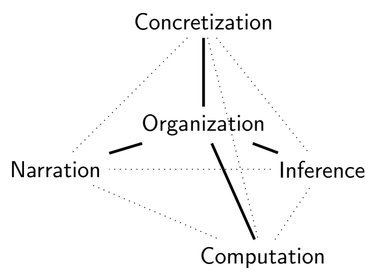
▷ **Definition 2.1.2.** One of the key insights is that the mathematics ecosystem involves a body of knowledge externalized in an ontology that provides organization and combines the following four aspects:

- ▷ Inference: exploring theories, formulating conjectures, and constructing proofs
- ▷ Computation: simplifying mathematical objects, re contextualizing conjectures...

- ▷ Concretization: collecting concrete examples/models, applying mathematical knowledge to real-world problems and situations.
- ▷ Narration: devising both informal and formal languages for expressing mathematical ideas, visualizing mathematical data, presenting mathematical developments, organizing and interconnecting mathematical knowledge

"Doing Math": as a Tetrapod

▷ We call the endeavour of creating a computer-supported mathematical ecosystem "Project tetrapod" as it needs to stand on four legs.



▷ **Collaborators:** KWARC@FAU, McMaster University

2.2 The Flexiformalist Program: Introduction

Background: Mathematical Documents

▷ Mathematics plays a fundamental role in Science, Technology, and Engineering (learn from Math, apply for STEM)

▷ Mathematical knowledge is rich in content, sophisticated in structure, and technical in presentation,

▷ its conservation, dissemination, and utilization constitutes a challenge for the community and an attractive line of inquiry.

▷ **Challenge:** How can/should we do mathematics in the 21^{st} century?

▷ Mathematical knowledge and objects are transported by documents

▷ **Three levels of electronic documents:**

0. printed (for archival purposes) ($\sim$90%)
1. digitized (usually from print) ($\sim$50%)

2. presentational: encoded text interspersed with presentation markup ($\sim 20\%$)
3. semantic: encoded text with functional markup for the meaning ($\leq 0.1\%$)

transforming down is simple, transforming up needs humans or AI.

▷ **Observation:** Computer support for access, aggregation, and application is (largely) restricted to the semantic level.

▷ **This talk:** How do we do maths and math documents at the semantic level?

Hilbert's (Formalist) Program

▷ **Definition 2.2.1.** Hilbert's Program called for a foundation of mathematics with

- ▷ A formal system that can express all of mathematics (language, models, calculus)
- ▷ Completeness: all valid mathematical statements can be proved in the formalism.
- ▷ Consistency: a proof that no contradiction can be obtained in the formalism of mathematics.
- ▷ Decidability: algorithm for deciding the truth or falsity of any mathematical statement.

▷ Originally proposed as "metamathematics" by David Hilbert in 1920.

▷ **Evaluation:**

The program was

- ▷ successful in that FOL+ZFC is a foundation [Göd30] (there are others)
- ▷ disappointing for completeness [Göd31], consistency [Göd31], decidability [Chu36; Tur36]
- ▷ inspiring for computer scientists building theorem provers
- ▷ largely irrelevant to current mathematicians (I want to address this!)

Formality in Logic and Artificial Intelligence

▷ AI, Philosophy, and Math identify formal representations with Logic

▷ **Definition 2.2.2.** A formal system $S{:=}\langle\mathcal{L},\mathcal{M},\mathcal{C}\rangle$ consists of

- ▷ a (computable) formal language $\mathcal{L}{:=}\mathcal{L}(S)$ (grammar for words/sentences)
- ▷ a model theory $\mathcal{M}$, (a mapping into (some) world)
- ▷ and a sound (complete?) proof calculus $\mathcal{C}$ (a syntactic method of establishing truth)

We use $\mathfrak{F}$ for the class of all formal systems.

▷ Reasoning in a formal system proceeds like a chess game: chaining "moves" allowed by the proof calculus via syntactic (depending only on the form) criteria.

▷ **Observation:** computers need $\mathcal{L}$ and $\mathcal{C}$ (adequacy hinges on relation to $\mathcal{M}$)

▷ Formality is a "all-or-nothing property".(a single "clearly" can ruin a formal proof)

▷ **Empirically:** formalization is not always achievable (too tedious for the gain!)

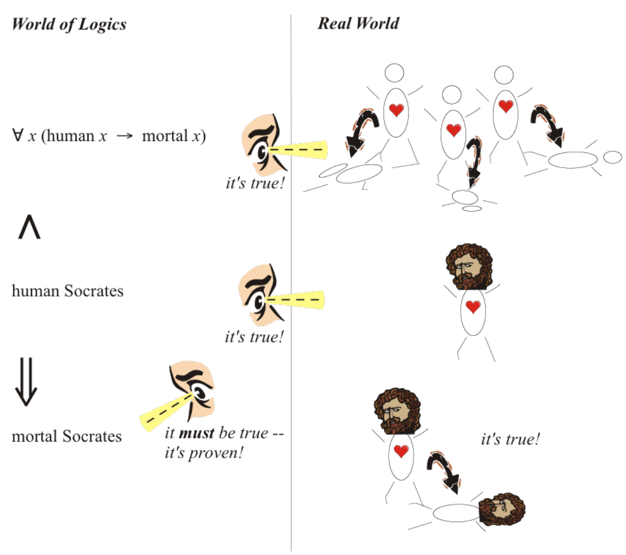
▷ Humans can draw conclusions from informal (not $\mathcal{L}$) representations by other means (not $\mathcal{C}$).

Kohlhase & Rabe: KRMT 191 2023-04-25

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!



Kohlhase & Rabe: KRMT 192 2023-04-25

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!)

Formalization in Mathematical Practice

▷ To formalize maths in a formal system $\mathcal{S}$, we need to choose a foundation, i.e. a foundational $\mathcal{S}$ theory, e.g. a set theory like ZFC.

▷ Formality is an all-or-nothing property (a single "obviously" can ruin it.)

▷ Almost all mathematical documents are informal in 4 ways:

- ▷ the foundation is unspecified (they are essentially equivalent)
- ▷ the language is informal (essentially opaque to MKM algos.)
- ▷ even formulae are informal (presentation markup)
- ▷ context references are underspecified
  - ▷ mathematical objects and concepts are often identified by name
  - ▷ statements (citations of definitions, theorems, and proofs) underspecified
  - ▷ theories and theory reuse not marked up at all

▷ The gold standard of mathematical communication is "rigor" (cf. [BC01])

- ▷ **Definition 2.2.3.** We call a mathematical document rigorous, if it could be formalized in a formal system given enough resources.
- ▷ This possibility is almost always unconsummated
- ▷ **Why?:** There are four factors that disincentivize formalization for Maths

  propaganda: *Maths is done with pen and paper*

  tedium: de Bruijn factors ~ 4 for current systems (details in [Wie12])

  inflexibility: formalization requires commitment to formal system and foundation

  proof verification useless: peer reviewing works just fine for Math
- ▷ **Definition 2.2.4.** The de Bruijn factor is the quotient of the lengths of the formalization and the original text.

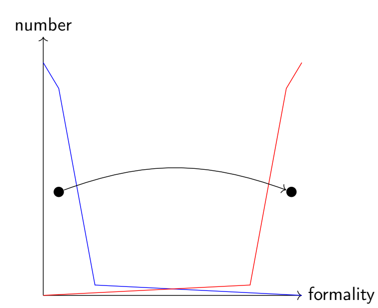
▷ **In Effect:** Hilbert's program has been comforting but useless

▷ **Question:** What can we do to change this?

Migration by Stepwise Formalization

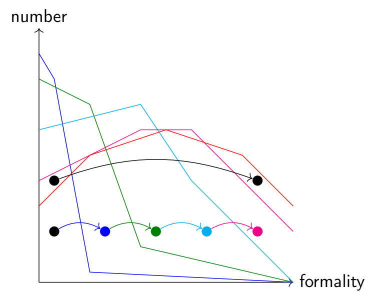
▷ Full Formalization is hard (we have to commit, make explicit)

▷ Let's look at documents and document collections.



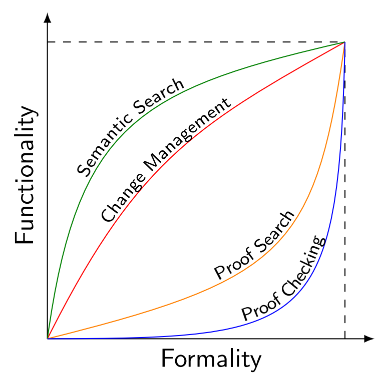
▷ Partial formalization allows us to

- ▷ formalize stepwise, and
- ▷ be flexible about the depth of formalization.



Functionality of Flexiformal Services

- ▷ **Generally:** Flexiformal services deliver according to formality level (GIGO: Garbage in ⇝ Garbage out!)
- ▷ **But:** Services have differing functionality profiles.
  - ▷ Math Search works well on informal documents
  - ▷ Change management only needs dependency information
  - ▷ Proof search needs theorem formalized in logic
  - ▷ Proof checking needs formal proof too



The Flexiformalist Program (Details in [Koh13])

- ▷ The development of a regime of partially formalizing
  - ▷ mathematical knowledge into a modular ontology of mathematical theories (content commons), and
  - ▷ mathematical documents by semantic annotations and links into the content commons (semantic documents),
- ▷ The establishment of a software infrastructure with
  - ▷ a distributed network of archives that manage the content commons and collections of semantic documents,

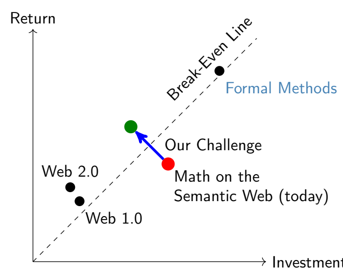
- ▷ semantic web services that perform tasks to support current and future mathematic practices
- ▷ active document players that present semantic documents to readers and give access to respective

▷ the re-development of comprehensive part of mathematical knowledge and the mathematical documents that carries it into a flexiformal digital library of mathematics.

Applications!

▷ A Business model for a Semantic Web for Math/Science?

▷ For uptake it is essential to match the return to the investment!



▷ Need to move the technology up (carrots) and left (easier)

2.3 What is formality?

The Process of Formalization

▷ Formalization in mathematics can be seen as a sequence of documents

1. an informal proof sketch on a blackboard, and
2. a high-level run-through of the essentials of a proof in a colloquium talk,
3. and the speaker's notes that contain all the *details* that are glossed over in
4. a *fully rigorous proof published in a journal*, which may lead to
5. a *mechanical verification* of the proof in a *proof checker*. (This is formal!)

▷ Intuitively, the steps get ever more formal, but our definition cannot predict this.

▷ **Example 2.3.1.** A recap of concepts from the intro of [CS09]

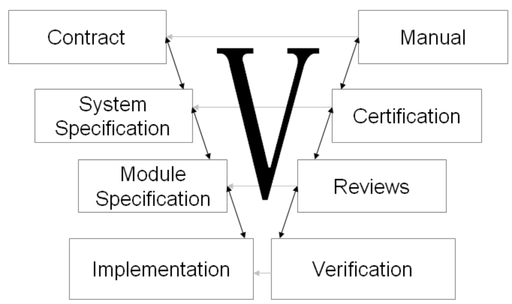
An accelerated Turing machine (sometimes called Zeno machine) is a Turing machine that takes 2^{-n} units of time (say seconds) to perform its n^{th} step.

▷ **Example 2.3.2.** A rigorous definition of the same concept.

Definition 1.3: An **accelerated Turing machine** is a Turing machine $M = \langle X, \Gamma, S, s_o, \square, \delta \rangle$ working with with a computational time structure $T = \langle \{t_i\}_i, <, + \rangle$ with $T \subseteq \mathbb{Q}_+$ ($\mathbb{Q}_+$ is the set of non-negative rationals) such that $\sum_{i \in \mathbb{N}} t_i < \infty$.

Multiple Dimensions in Formalization

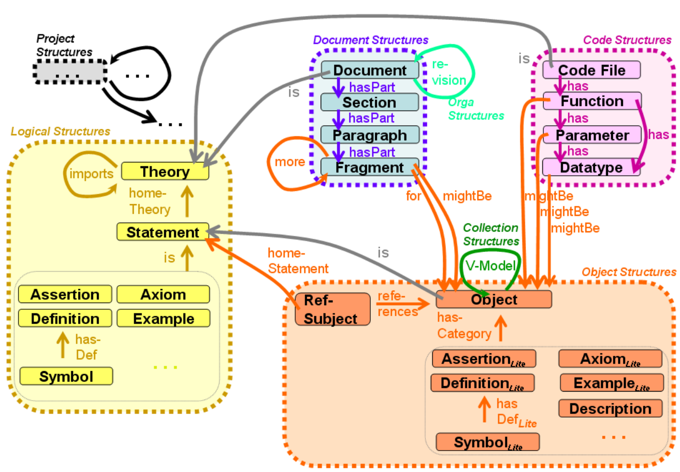
▷ **Example 2.3.3 (SAMS Case Study).** Formalize a set of robot design documents down to implementation and up again to documentation.



The V-Model requires explicit cross-references between the levels

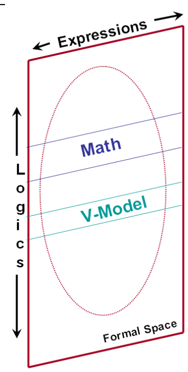
▷ **Observation:** The links between the document fragments are formalized by a graph structure for machine support. (e.g. requirements tracing)

▷ We ended with a complex, multi-dimensional collection domain model

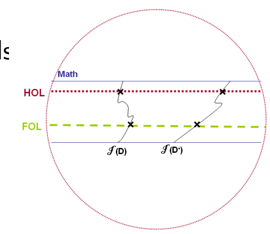


▷ In particular, the formalization process was linear in the dimensions at best.

What is Informal Mathematical Knowledge

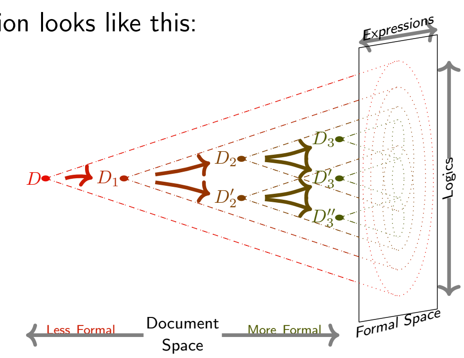


- ▷ **Idea:** Informal knowledge could be formalized (but isn't yet!)
- ▷ **Definition 2.3.4.** The meaning of a knowledge item is the set of all its formalizations.
- ▷ **Problem:** What is the space of formalizations?
- ▷ **Definition 2.3.5.** The formal space is the set $\mathcal{F}:=\{\langle S,e\rangle | S\in\mathfrak{F}, e\in\mathcal{L}(S)\}$, where $\mathfrak{F}$ is the class of formal systems and $\mathcal{L}(S)$ is the language of S. (i.e. every formal expression is a point in $\mathcal{F}$)
- ▷ Different Logics correspond to different bands
- ▷ The meaning of $\mathcal{D}$ is a set $\mathcal{I}(\mathcal{D}) \subseteq \mathcal{F}$.
- ▷ $\mathcal{D}$ can be formalized in multiple logics $\mathcal{I}(\mathcal{D})$ forms a cross-section of logic-bands.



A Formality Ordering on $\mathcal{F}$

- ▷ Stepwise formalization looks like this:



- ▷ **Definition 2.3.6.** $\mathcal{D}$ is more formal than $\mathcal{D}'$ (write $\mathcal{D}\lll\mathcal{D}'$), iff $\mathcal{I}(\mathcal{D})\subset\mathcal{I}(\mathcal{D}')$.
- ▷ This partial ordering relation answers the question of "graded formality" or the nature of "stepwise formalization" raised above.

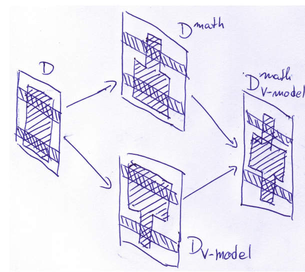
Stepwise Formalization in Multiple Dimensions

- ▷ **Empirically:** Formalization is a stepwise process of (order of steps may vary)
  - ▷ spotting semantic objects (from the surrounding text)

- ▹ chunking: grouping them for re use (e.g. assigning to home theories)
- ▹ relating: making their relationships explicit (this is used by semantic services)

▷ **In multi-dimensional situations:**

- ▹ any formalization step on $\mathcal{D}$ trims $\mathcal{I}(\mathcal{D})$.
- ▹ not all "steps" are comparable in $\lll$
- ▹ but per-dimension formalization is confluent



▷ **Observation:** This is the normal situation, we coin a new concept to describe it.

▷ **Definition 2.3.7.** We call a representation flexiform, iff it is of flexible formality in any of the adequate dimensions of formality.

Flexiforms and Flexiformalization

▷ **Definition 2.3.8.** "Flexiform" is an adjective, we are interested in

- ▹ flexiform fragments: e.g. definitions with formulae in MathML parallel markup (presentation/content).
- ▹ flexiform theories: formal theories with flexiform fragments.
- ▹ flexiform digital libraries: formality widely ranging, supports flexiformalization in collection.

Call all such representations flexiforms (noun)

▷ **Remark:** The set of flexiforms has very good closure properties.

- ▹ Flexiform fragments can be composed to flexiform documents,
- ▹ which can be collected to flexiform libraries,
- ▹ which in turn can be formalized to flexiform theory graphs
- ▹ or excerpted to flexiform documents.

All that without leaving the space of flexiforms!

2.4 A "formal" Theory of Flexiformality

How to model Flexiformal Mathematics

▷ **I hope to have convinced you:** that Math is informal:

- ▹ foundations unspecified (what a relief)

- ▷ natural language & presentation formulae (humans can disambiguate)
- ▷ context references (but math is better than the pack)

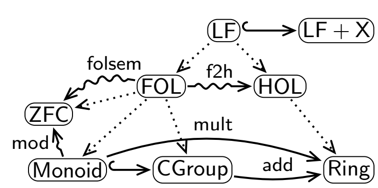
▷ **Problem:** How do we deal with that in our "formal" systems?

▷ **Proposed Answer:** learn from OpenMath/MathML

- ▷ referential theory of meaning (by pointing to symbol definitions)
- ▷ allow opaque content (presentation/natural language)
- ▷ parallel markup (mix formal/informal recursively at any level)
- ▷ pluralism at all levels (object/logic/foundation/metalogic)
- ▷ underspecification of symbol meaning

extend to statement/paragraph and theory/discourse levels (OMDoc)

OMDoc in a Nutshell (three levels of modeling) [Koh06]

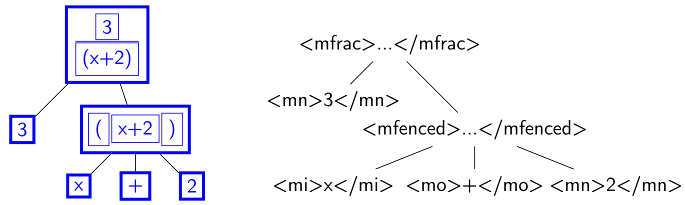
Formula level OpenMath/C-MathML ▷ Objects as logical formulae ▷ symbol meaning by reference to theory level	<apply> <csymbol cd="ring">plus</c.> <csymbol cd="ring">zero</c.> <ci>N</ci> </apply>
Statement level: ▷ Definition, Theorem, Proof, Example ▷ semantics via explicit forms and refs. ▷ parallel formal & natural language	<defn for="plus" **type**="rec"> <**CMP**>rec. eq. for plus</**CMP**> <**FMP**>$X + 0 = X$</**FMP**> <**FMP**>$X + s(Y) = s(X + Y)$</**FMP**> </defn>
Module level Theory Graph [RK13] ▷ inheritance via symbol-mapping ▷ views by proof-obligations ▷ logics as meta-theories (logic atlas) ▷ meta-logics as oracles for type/eq	

2.4.1 Parallel Markup in MathML

Layout Schemata and the MathML Box model

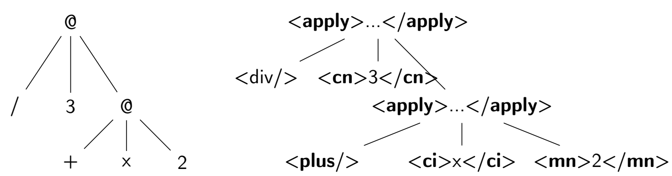
▷ Presentation MathML represents the visual appearance of a formula in a tree of layout primitives

▷ **Example 2.4.1 (Presentation MathML for $3/(x+2)$).**



Functional Markup in MathML: The "Operator Tree"

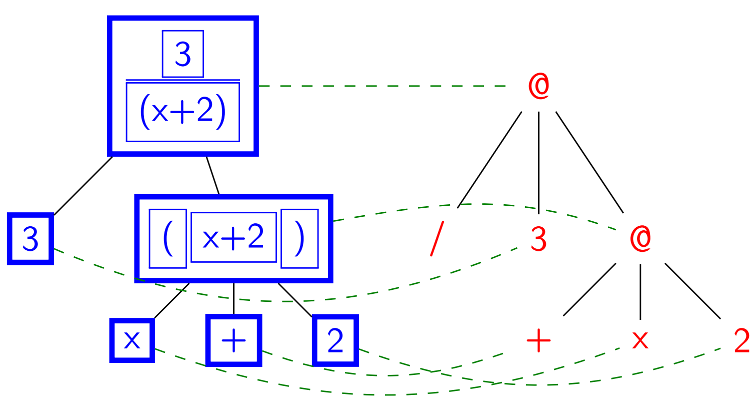
- Content MathML represents the functional structureof a formula in a tree of operators, via application and binding.
- **Example 2.4.2 (Content MathML for $3/(x+2)$).**



- **Extra Operators:** use <csymbol cd="⟪CD⟫">⟪Name⟫</csymbol>, where
  - CD is a content dictionary a document that defines Name
  - Name is the name of a symbol definition in CD.

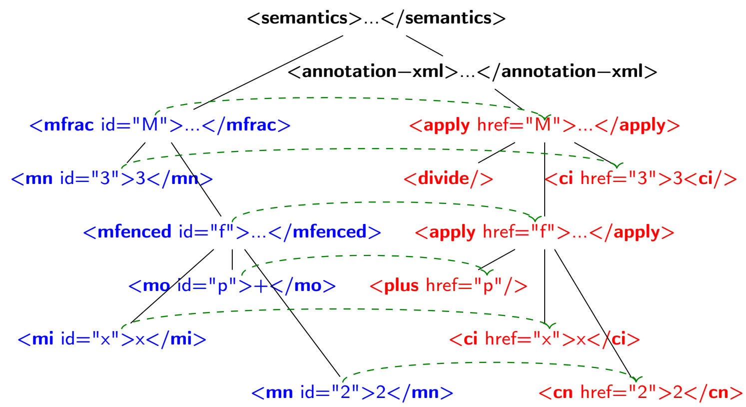
Parallel Markup e.g. in MathML

- **Idea:** Combine the presentation and content markup and cross-reference



- use e.g. for semantic copy and paste. (click o3n presentation, follow link and copy content)

▷ **Concrete Realization in MathML:** semantics element with presentation as first child and content in annotation−xml child



x0

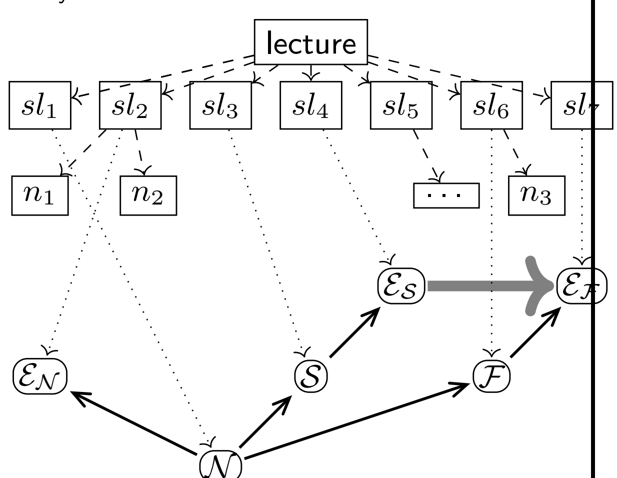
2.4.2 Parallel Markup in OMDoc

Separating Narrative– and Conceptual Structure

▷ Document structure is discourse-level presentation of content structure.

▷ **Example 2.4.3.** Introducing a theory via a straw man in a lecture

- ▷ sl_i are slides
- ▷ n_i is narrative text
- ▷ $\mathcal{E}_i$ are examples
- ▷ $\mathcal{N}$ is a naive theory
- ▷ $\mathcal{F}$ is the final theory
- ▷ $\mathcal{S}$ is the straw man



▷ **Idea:** have two documents content + narrative structure

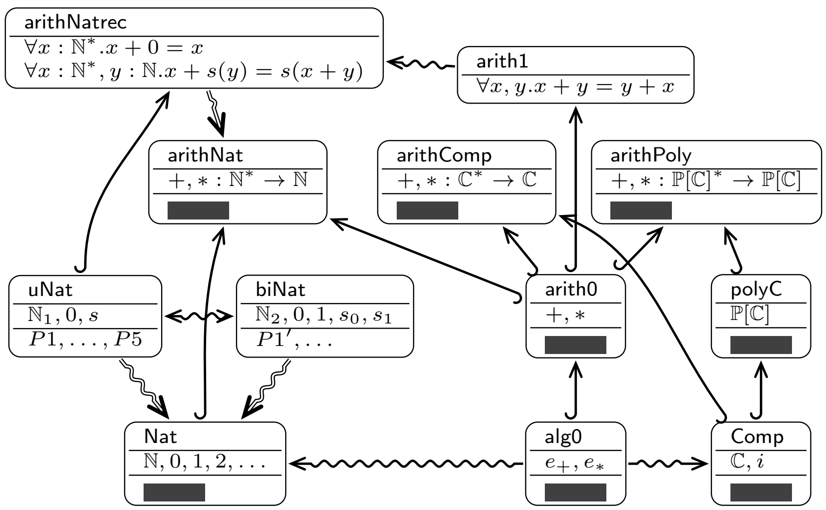
▷ **Narrative OMDoc:** only doc. structure + narr. elements + links into content.

▷ **Future:** Generate the narr. from content (need discourse-level content markup)

2.4.3 Flexible Symbol Grounding in OMDoc

A Formal Theory of Underspecification?

- ▷ Use theory graphs to specify "meaning" in stages e.g. arithmetics



- ▷ **Be non-committal:** In OpenMath, `arith1.ocd` only says that + is commutative
  this is a feature, not a bug (lets you remain uncommitted/underspecified)

2.5 Representing Mathematical Vernacular

Chapter 3

Summary and Review

3.1 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)
- ▷ **Definition 3.1.1.** In the theory graph paradigm, we have
  - ▷ theories as collections of symbol declarations and axioms (model assumptions)
  - ▷ theory morphisms as mappings that translate axioms into theorems

  The central object of knowledge curation is the theory graph which has theories as nodes and theory morphisms as edge.
- ▷ **Example 3.1.2 (MMT: Modular Mathematical Theories).** MMT is a foundation-independent theory graph formalism with advanced theory morphisms.

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG Kohlhase & Rabe: KRMT 213 2023-04-25

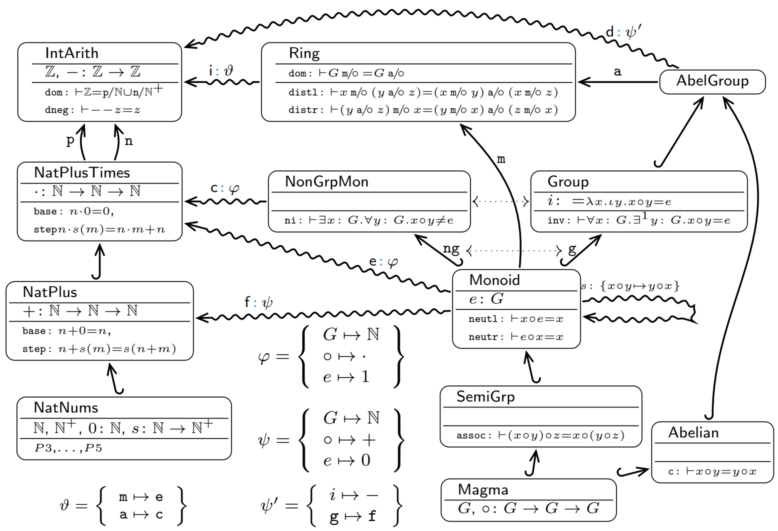
The Theory Graph Paradigm

- ▷ **Definition 3.1.3.** In the little theories doctrine, theories are made as small as reasonable to enhance modularity and re-use.
- ▷ **Definition 3.1.4.** In the tiny theories doctrine theories are minimal, i.e. have at most two declarations. (one inclusions and one payload)
- ▷ **Problem:** With a proliferation of abstract (tiny) theories readability and accessi-

bility suffers (one reason why the Bourbaki books fell out of favor)

Modular Representation of Math (MMT Example)

▷ **Example 3.1.5 (Elementary Algebra and Arithmetics).**



The MMT Module System

▷ **Central notion:** theory graph with theory nodes and theory morphisms as edges

▷ **Definition 3.1.6.** In MMT, a theory is a sequence of constant declarations optionally with type declarations and definitions

▷ MMT employs the Curry/Howard isomorphism and treats

- ▷ axioms/conjectures as typed symbol declarations (propositions-as-types)
- ▷ inference rules as function types (proof transformers)
- ▷ theorems as definitions (proof terms for conjectures)

▷ **Definition 3.1.7.** MMT had two kinds of theory morphisms

- ▷ structures instantiate theories in a new context (also called: definitional link, import)
  they import of theory S into theory T induces theory morphism $S \to T$
- ▷ views translate between existing theories (also called: postulated link, theorem link)
  views transport theorems from source to target (framing).

▷ Together, structures and views allow a very high degree of re-use

▷ **Definition 3.1.8.** We call a statement t induced in a theory T, iff there is

- ▹ a path of theory morphisms from a theory S to T with (joint) assignment σ,
- ▹ such that $t = \sigma(s)$ for some statement s in S.

▷ **Definition 3.1.9.** In MMT, all induced statements have a canonical name, the MMT URI.

Applications for Theories in Physics

▷ Theory Morphisms allow to "view" source theory in terms of target theory.

▷ Theory Morphisms occur in Physics all the time.

Theory	Temp. in Kelvin	Temp. in Celsius	Temp. in Fahrenheit
Signature	°K	°C	°F
Axiom:	absolute zero at 0°K	Water freezes at 0°C	cold winter night: 0°F
Axiom:	$\delta($°K$1) = \delta($°C$1)$	Water boils at 100°C	domestic pig: 100°F
Theorem:	Water freezes at 271.3°K	domestic pig: 38°C	Water boils at 170°F
Theorem:	cold winter night: 240°K	absolute zero at -271.3°C	absolute zero at -460°F

Views: °C $\stackrel{+271.3}{\longrightarrow}$ °K, °C $\stackrel{-32/2}{\longrightarrow}$ °F, and °F $\stackrel{+240/2}{\longrightarrow}$ °K, inverses.

▷ **Other Examples:** Coordinate Transformations,

▷ **Application:** Unit Conversion: apply view morphism (flatten) and simplify with UOM. (For new units, just add theories and views.)

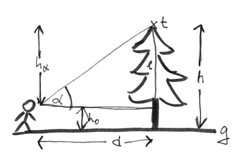
▷ **Application:** MathWebSearch on flattened theory (Explain view path)

3.2 Application: Serious Games

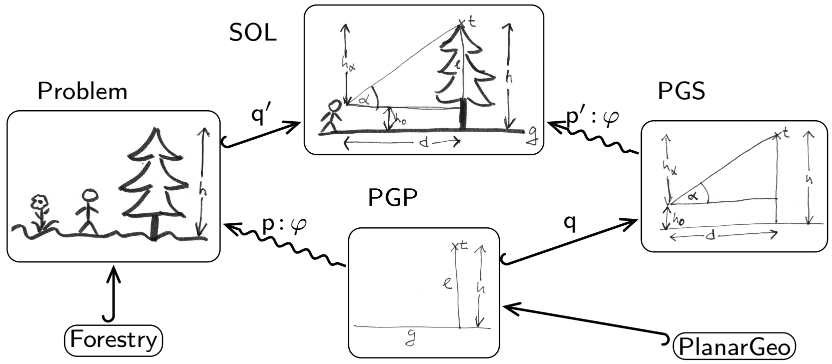
Framing for Problem Solving (The FrameIT Method)

▷ **Example 3.2.1 (Problem 0.8.15).**

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

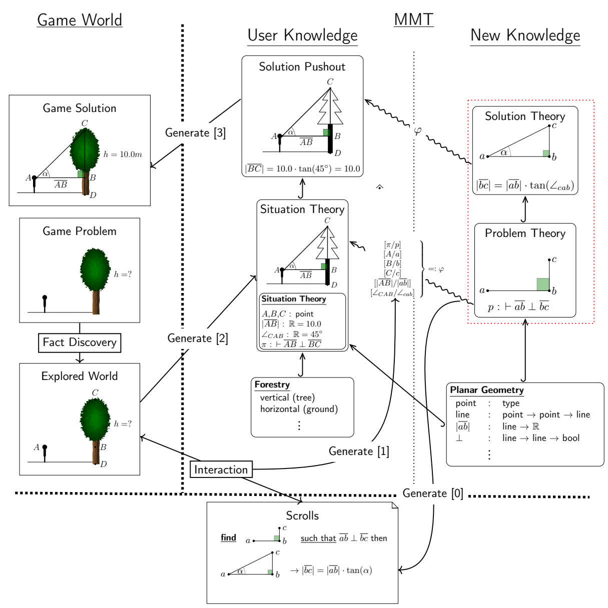


▷ Framing: view the problem as one that is already understood (using theory morphisms)



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

Example Learning Object Graph



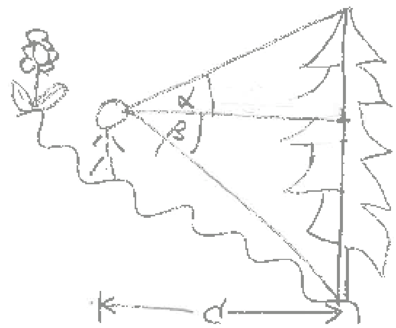
FrameIT Method: Problem

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

▷ 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)

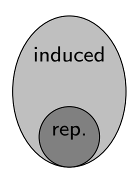
Kohlhase & Rabe: KRMT 221 2023-04-25

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.

3.3 Search in the Mathematical Knowledge Space

The Mathematical Knowledge Space

- ▷ **Observation 3.3.1.** *The value of framing is that it induces new knowledge*
- ▷ **Definition 3.3.2.** 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

Kohlhase & Rabe: KRMT 222 2023-04-25

♭search: Indexing flattened Theory Graphs

- **Simple Idea:** We have all the necessary components: MMT and MathWebSearch
- **Definition 3.3.3.** The ♭search systen is an integration of MathWebSearch and MMT that
  - computes the induced formulae of a modular mathematical library via MMT (aka. flattening)
  - indexes induced formulae by their MMT URIs in MathWebSearch
  - uses MathWebSearch for unification-based querying (hits are MMT URIs)
  - uses the MMT to present MMT URI (compute the actual formula)
  - generates explanations from the MMT URI of hits.
- Implemented by Mihnea Iancu in ca. 10 days (MMT harvester pre-existed)
  - almost all work was spent on improvements of MMT flattening
  - MathWebSearch just worked (web service helpful)

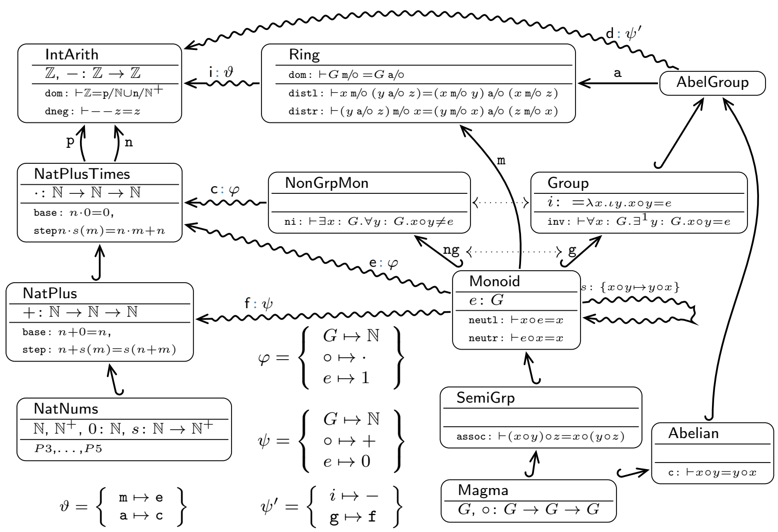
♭search User Interface: Explaining MMT URIs

- **Recall:** ♭search (MathWebSearch really) returns a MMT URI as a hit.
- **Question:** How to present that to the user? (for his/her greatest benefit)
- **Fortunately:** MMT system can compute induced statements (the hits)
- **Problem:** Hit statement may look considerably different from the induced statement
- **Solution:** Template-based generation of NL explanations from MMT URIs.

  MMT knows the necessary information from the components of the MMT URI.

Modular Representation of Math (MMT Example)

- **Example 3.3.4 (Elementary Algebra and Arithmetics).**



Example: Explaining a MMT URI

▷ **Example 3.3.5.** ♭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

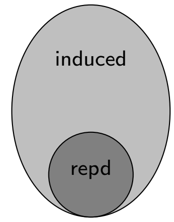
- ▷ ground the explanation by

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

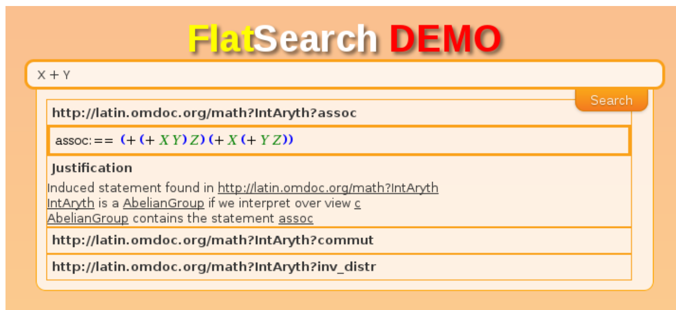
♭search on the LATIN Logic Atlas

▷ 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 bsearch frontend at `http://cds.omdoc.org:8181/search.html`



Overview: KWARC Research and Projects

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

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

Foundations: Computational Logic, Web Technologies, OMDoc/MMT

Take-Home Message

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

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

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.

[BC01] Henk Barendregt and Arjeh M. Cohen. "Electronic communication of mathematics and the interaction of computer algebra systems and proof assistants". In: *Journal of Symbolic Computation* 32 (2001), pp. 3–22.

[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`.

[Chu36] Alonzo Church. "A note on the Entscheidungsproblem". In: *Journal of Symbolic Logic* (May 1936), pp. 40–41.

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

[CS09] Cris Calude and Ludwig Staiger. *A Note on Accelerated Turing Machines*. CDMTCS Research Report 350. Centre for Discrete Mathematics and Theoretical Computer Science, Auckland University, 2009. URL: `http://www.cs.auckland.ac.nz/CDMTCS/researchreports/350cris.pdf`.

[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.

[Göd30] Kurt Gödel. "Die Vollständigkeit der Axiome des logischen Funktionenkalküls". In: *Monatshefte für Mathematik und Physik* 37 (1930). English Version in [Hei67], pp. 349–360.

[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*. 3rd 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.

[Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: http://omdoc.org/pubs/omdoc1.2.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.

[Koh13] Michael Kohlhase. "The Flexiformalist Manifesto". In: *14th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2012)*. Ed. by Andrei Voronkov et al. Timisoara, Romania: IEEE Press, 2013, pp. 30–36. ISBN: 978-1-4673-5026-6. URL: https://kwarc.info/kohlhase/papers/synasc13.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. Springer Verlag, 2006, pp. 331–342. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.453.3917&rep=rep1&type=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.

[ML86] Sounders Mac Lane. *Mathematics Form and Function*. Springer Verlag, 1986.

[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).

[RK13] Florian Rabe and Michael Kohlhase. "A Scalable Module System". In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: https://kwarc.info/frabe/Research/mmt.pdf.

[sTeX] *sTeX: A semantic Extension of TeX/LaTeX*. URL: https://github.com/sLaTeX/sTeX (visited on 05/11/2020).

[Tur01] Daniele Turi. *Category Theory Lecture Notes*. 2001. URL: http://www.dcs.ed.ac.uk/home/dt/CT/categories.pdf.

[Tur36] Alan Turing. "On computable numbers, with an application to the Entscheidungsproblem". In: *Proceedings of the London Mathematical Society, Series 2* 42 (June 1936), pp. 230–265.

[Wie12] Freek Wiedijk. *The "de Bruijn factor"*. web page at http://www.cs.ru.nl/~freek/factor/. Mar. 1, 2012. URL: http://www.cs.ru.nl/~freek/factor/.

[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