Logic of Proof Assistants

Prof. Floris van Doorn *

University of Bonn

15th May, 2025

Contents

*LaTeX-realization by Hannah Scholz

1 Introduction

The topics of this class are:

(i) First-order Logic/Set Theory

(ii) Lambda Calculus

(iii) Simple Type Theory (Higher-Order Logic)

(iv) Dependent Type Theory/Homotopy Type Theory

Example 1.1. Here are examples of proof assistants for these different types of logics:

(i) First-order Logic/Set Theory: Mizar, Metamath

(ii) Simple Type Theory: Isabelle/HoL, HoL Light

(iii) Dependent Type Theory: Lean, Rocq (formerly Coq), Agda

(iv) Homotopy Type Theory: cubicaltt, rzk

Remark 1.2. You might want to have the following criteria for a logic:

(i) Appropriate (You can encode mathematical arguments.)

(ii) Simple (It is relatively easy to understand.)

(iii) Expressive (Mathematical arguments are convenient to express.)

Theorem 1.3. *Let π be the prime counting function, i.e. $\pi\colon \mathbb{R} \to \mathbb{N}$, $x \mapsto |\{p \leq x \mid p \text{ prime}\}|$. Then $\lim_{x\to\infty} \frac{\pi(x)}{x/\log(x)} = 1$.*

Remark 1.4. When formalizing/stating this theorem in a formal logic there are a few things that you need to think about:

(i) What do you do about division by zero?

(ii) What does division even mean? (Do you define division for $\mathbb{R}$ explicitly? Do you define it generally for a field? Or even for a group? How do you ensure that the "correct" field structure on $\mathbb{R}$ gets used?)

(iii) How do you define a limit? (Do you define a limit for $\mathbb{R}$ explicitly? Or for every topological space? How do you ensure the "correct" topology on $\mathbb{R}$ gets used? How do you deal with potentially non-unique limits (for example in non-Hausdorff spaces)?)

Remark 1.5. You can make the following design choices for "a logic":

(i) Is the logic typed or untyped?

(ii) Is the logic constructive or classical?

(iii) Does the logic support computation?

Remark 1.6. In logic there is the **object language** and we reason about it in a **meta-language** ("ordinary mathematical reasoning").

1.1 Inductive Definitions

Example 1.7. The natural numbers are inductively defined by $0 \in \mathbb{N}$ and $S\colon \mathbb{N} \to \mathbb{N}$, $n \mapsto n+1$.

Definition 1.8. Let U be a set and $\mathcal{C} \subseteq \bigcup_{n\in\mathbb{N}}(U^n \to U)$ a set of **constructors**. Where $c\colon U^n \to U$ is called an **n-ary function** and $(U^n \to U)$ is the collection on n-ary functions.

(i) $A \subseteq U$ is **closed under $\mathcal{C}$** if for any n-ary $c \in \mathcal{C}$ and for all $x_1, \dots, x_n \in A$ we have that $c(x_1, \dots, x_n) \in A$.

(ii) $A \subseteq U$ is **generated by $\mathcal{C}$** or **inductively defined by $\mathcal{C}$** if A is the smallest set that is closed under $\mathcal{C}$, i.e. $A = \bigcap\{B \subseteq U \mid B \text{ is closed under } \mathcal{C}\}$.

(iii) $A \subset U$ is **freely generated by $\mathcal{C}$** if

  (a) each constructor is injective on A and

  (b) the images of different constructors are disjoint.

Remark 1.9. $\varnothing$ is closed under $\mathcal{C}$ iff $\mathcal{C}$ has no nullary constructors.

Exercise 1.10. $\bigcap\{B \subseteq U \mid B \text{ is closed under } \mathcal{C}\}$ *is closed under* $\mathcal{C}$.

Example 1.11.

(i) The free group.

(ii) The σ-algebra generated by a collection of subsets. (This is not freely generated.)

(iii) The topology generated by a collection of subsets. (This is not freely generated.)

Theorem 1.12 (Structural Induction)**.** *If $A \subset U$ is generated by $\mathcal{C}$ and $P\colon A \to \{\top, \bot\}$ is a predicate on A, to prove $\forall a \in A, P(a)$ it suffices to show: for any n-ary $c \in \mathcal{C}$ and any $x_1, \ldots, x_n \in A$ if $P(x_1), \ldots, P(x_n)$ then $P(c(x_1, \ldots, x_n))$.*

Proof. Exercise. □

Remark 1.13. The base case of the induction is given by nullary constructors.

Theorem 1.14 (Structural Recursion)**.** *If $A \subset U$ is freely generated by $\mathcal{C}$, B is a set and for any n-ary $c \in \mathcal{C}$ we have a $g_c\colon B^n \to B$ then there is a unique function $f\colon A \to B$ such that $f(c(a_1, \ldots, a_n)) = g_c(f(a_1), \ldots, f(a_n))$ for every $c \in \mathcal{C}$ and $a_1, \ldots a_n \in A$.*

Proof. Exercise. □

Example 1.15. For $A = \mathbb{N}$ this reduces to $f(0) \coloneqq g_0$ and $f(S(n)) \coloneqq g_s(f(n))$.

2 First-Order Logic

Definition 2.1. A (first-order) **language** $\mathcal{L}$ is a triple $(\mathcal{F}, \mathcal{R}, a)$ where $\mathcal{F}$ is a set of function symbols, $\mathcal{R}$ is a set of relation symbols, $\mathcal{F}$ and $\mathcal{R}$ are disjoint and $a\colon \mathcal{F} \cup \mathcal{R} \to \mathbb{N}$ is the arity function.

Example 2.2. A language for groups $\mathcal{L}_{\text{Group}}$ has $\mathcal{F} \coloneqq \{\cdot, ^{-1}, 1\}$, $\mathcal{R} \coloneqq \varnothing$, $a(\cdot) = 2$, $a(^{-1}) = 1$ and $a(1) = 0$.

Definition 2.3. We fix an infinite set of **variables** $\mathcal{V} \coloneqq \{x_0, x_1, \dots\}$.

Remark 2.4. We use x for variables, f and g for functions and R and S for relations.

Definition 2.5. We can define the **terms** $T_{\mathcal{L}}$ in the language $\mathcal{L}$ using the **Backus–Naur form (BNF)** :

$$s, t ::= x \mid f(t_1, \dots, t_n)$$

where f is an n-ary function symbol.

Definition 2.6. Formally, we define the **terms** $T_{\mathcal{L}}$ in the language $\mathcal{L}$ in the following way. We define the set of **symbols** $S \coloneqq \mathcal{F} \dot\cup \mathcal{V} \dot\cup \{\text{“(”}, \text{“)”}, \text{“,”}\}$ and the set of finite sequences of symbols S^*. Let $\mathcal{C}$ be defined as:

(i) for each variable $x \in \mathcal{V}$ there is a nullary constructor $c_x \coloneqq x$

(ii) for each n-ary function symbol f there is an n-ary constructor $c_f\colon (S^*)^n \to S^*$, $c_f(t_1, \dots, t_n) \coloneqq f\text{“(”}t_1\text{“,”}\dots\text{“,”}t_n\text{“)”}$

Then $T_{\mathcal{L}} \subseteq S^*$ is the set generated by $\mathcal{C}$.

Example 2.7.

(i) “(”“)”“,”f is in S^* but not in $T_{\mathcal{L}}$.

(ii) If f is binary then $f\text{“(”}x_0\text{“,”}x_1\text{“)”}$ is in $T_{\mathcal{L}}$.

Remark 2.8. Technically, the brackets and commas are not necessary. They are however necessary when you use infix notation. (For example the meaning of $a \cdot b + c$ is unclear.)

Definition 2.9. First-order **formulas** $\Phi_{\mathcal{L}}$ are specified by

$$\varphi, \psi ::= \bot \mid s = t \mid R(t_1, \dots t_n) \mid (\varphi \wedge \psi) \mid (\varphi \vee \psi) \mid (\varphi \to \psi) \mid (\forall x.\varphi) \mid (\exists x.\varphi)$$

where $\mathcal{R}$ is an n-ary relation symbol and $t_1, \dots, t_n \in T_{\mathcal{L}}$.

Remark 2.10. In classical logic one could omit the rules $(\varphi \wedge \psi)$ and $(\varphi \vee \psi)$ (as they can be defined using the other rules). They are however necessary for constructive logic.

Remark 2.11. We can define other connectives:

(i) $\neg\varphi \coloneqq (\varphi \to \bot)$

(ii) $\varphi \leftrightarrow \psi \coloneqq ((\varphi \to \psi) \wedge (\psi \to \varphi))$

Remark 2.12. When writing formulas we omit some parentheses:

(i) $\varphi \to \psi \to \theta$ means $\varphi \to (\psi \to \theta)$

(ii) $\forall x.\varphi \to \psi$ means $\forall x.(\varphi \to \psi)$

Remark 2.13. We want $\forall x.x = x$ and $\forall y.y = y$ to mean the same thing. Options to achieve this are:

(i) Define $(\forall x.x = x) \equiv_\alpha (\forall y.y = y)$ to be **α-equivalent**. And then define the set of formulas to be $\Phi_{\mathcal{L}} / \equiv_\alpha$.

(ii) We could not use variable names for bound variables and use **de Bruijn indices** instead.

Remark 2.14. $\forall x.x = y$ has **bound variables** $\{x\}$ and **free variables** $\{y\}$. For a formula φ or a term t we also write **fv**(φ) and **fv**(t) for the set of free variables in φ and t.

Definition 2.15. A **sentence** is a formula without free variables.

Definition 2.16. **Substitution** $s[t/x]$ of x by t in a term s is defined recursively by

(i) $y[t/x] \coloneqq \begin{cases} t & \text{if } y = x \\ y & \text{otherwise} \end{cases}$

(ii) $f(s_1, \dots, s_n)[t/x] \coloneqq f(s_1[t/x], \dots, s_n[t/x])$

Example 2.17. Defining substitution in formulas is a little bit harder as we need to avoid **variable capture**: $(\exists x.x \leq z)[(x+1)/z]$ should not be $\exists x.x \leq x+1$ but $\exists y.y \leq x+1$.

Definition 2.18. For a formula φ **substitution** $\varphi[t/x]$ is defined as:

(i) $(s = s')[t/x] \coloneqq (s[t/x] = s'[t/x])$

(ii) $R(t_1, \ldots, t_n)[t/x] \coloneqq R(t_1[t/x], \ldots, t_n[t(x)])$

(iii) $(\varphi \vee \psi)[t/x] \coloneqq (\varphi[t/x] \vee \psi[t/x])$

(iv) $(\varphi \wedge \psi)[t/x] \coloneqq (\varphi[t/x] \wedge \psi[t/x])$

(v) $(\varphi \to \psi)[t/x] \coloneqq (\varphi[t/x] \to \psi[t/x])$

(vi) $(\forall y.\varphi)[t/x] \coloneqq \begin{cases} \forall y.\varphi & \text{if } y = x \\ \forall z.\varphi[z/y][t/x] & \text{otherwise} \end{cases}$ where z does not occur in t.

(vii) $(\exists y.\varphi)[t/x] \coloneqq \begin{cases} \exists y.\varphi & \text{if } y = x \\ \exists z.\varphi[z/y][t/x] & \text{otherwise} \end{cases}$ where z does not occur in t.

Definition 2.19. α**-equivalence** is the **congruence closure** of

(i) $(\forall x.\varphi) \equiv_\alpha (\forall y.\varphi[y/x])$

(ii) $(\exists x.\varphi) \equiv_\alpha (\exists y.\varphi[y/x])$

i.e. it is the smallest equivalence relation containing these two rules and respecting the connectives:

(i) $(\varphi_1 \wedge \varphi_2) \equiv_\alpha (\psi_1 \wedge \psi_2)$ for $\varphi_1 \equiv_\alpha \psi_1$ and $\varphi_2 \equiv_\alpha \psi_2$

(ii) $(\varphi_1 \vee \varphi_2) \equiv_\alpha (\psi_1 \vee \psi_2)$ for $\varphi_1 \equiv_\alpha \psi_1$ and $\varphi_2 \equiv_\alpha \psi_2$

(iii) $(\varphi_1 \to \varphi_2) \equiv_\alpha (\psi_1 \to \psi_2)$ for $\varphi_1 \equiv_\alpha \psi_1$ and $\varphi_2 \equiv_\alpha \psi_2$

(iv) $(\forall x.\varphi) \equiv_\alpha (\forall x.\psi)$ if $\varphi \equiv_\alpha \psi$

(v) $(\exists x.\varphi) \equiv_\alpha (\exists x.\psi)$ if $\varphi \equiv_\alpha \psi$

Remark 2.20. We will treat α-equivalence as an equivalence relation. You could also define the formulas as $\Phi_{\mathcal{L}} / \equiv_\alpha$ and thus treat α-equivalence as equality.

2.1 Provability

Definition 2.21. Let Γ be a set of formulas and φ a formula. Then $\boldsymbol{\Gamma \vdash \varphi}$ (read : "Γ proves φ") is defined inductively by

(i) $\dfrac{}{\Gamma, \varphi \vdash \varphi}$ (assumption rule)

(ii) $\dfrac{\Gamma \vdash \varphi \qquad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi}$ ($\wedge$-introduction)

(iii) $\dfrac{\Gamma \vdash \varphi_1 \wedge \varphi_2}{\Gamma \vdash \varphi_i}$ for $i = 1, 2$ ($\wedge$-elimination)

(iv) $\dfrac{\Gamma \vdash \varphi_i}{\Gamma \vdash \varphi_1 \vee \varphi_2}$ for $i = 1, 2$ ($\vee$-introduction)

(v) $\dfrac{\Gamma \vdash \varphi \vee \psi \qquad \Gamma, \varphi \vdash \theta \qquad \Gamma, \psi \vdash \theta}{\Gamma \vdash \theta}$ ($\vee$-elimination)

(vi) $\dfrac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi}$ ($\to$-introduction)

(vii) $\dfrac{\Gamma \vdash \varphi \to \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$ ($\to$-elimination)

(viii) $\dfrac{\Gamma, \neg\varphi \vdash \bot}{\Gamma \vdash \varphi}$ (proof by contradiction)

(ix) $\dfrac{\Gamma \vdash \varphi}{\Gamma \vdash \forall x.\varphi}$ for $x \notin \mathrm{fv}(\Gamma)$ ($\forall$-introduction)

(x) $\dfrac{\Gamma \vdash \forall x.\varphi}{\Gamma \vdash \varphi[t/x]}$ ($\forall$-elimination)

(xi) $\dfrac{\Gamma \vdash \varphi[t/x]}{\Gamma \vdash \exists x.\varphi}$ ($\exists$-introduction)

(xii) $\dfrac{\Gamma \vdash \exists x.\varphi \qquad \Gamma, \varphi \vdash \psi}{\Gamma \vdash \psi}$ for $x \notin \mathrm{fv}(\Gamma, \psi)$ ($\exists$-elimination)

(xiii) $\dfrac{}{\Gamma \vdash t = t}$ (=-introduction)

(xiv) $\dfrac{\Gamma \vdash s = t \qquad \Gamma \vdash \varphi[t/x]}{\Gamma \vdash \varphi[s/x]}$ (=-elimination)

(xv) $\dfrac{\Gamma \vdash \varphi}{\Gamma \vdash \psi}$ for $\varphi \equiv_\alpha \psi$ (α-equivalence)

Remark 2.22. Read "$\dfrac{\text{A}}{\text{B}}$" as: "Under the assumptions A we can prove B". With "Γ, φ" we really mean $\Gamma \cup \{\varphi\}$.

Example 2.23. If φ and ψ are formulas then we can show $\vdash (\varphi \wedge \psi) \to (\psi \wedge \varphi)$ using the following **proof tree** :

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\varphi \wedge \psi \vdash \varphi \wedge \psi}\text{Assump.}}{\varphi \wedge \psi \vdash \psi}\wedge\text{-elim.} \quad \dfrac{\dfrac{}{\varphi \wedge \psi \vdash \varphi \wedge \psi}\text{Assump.}}{\varphi \wedge \psi \vdash \varphi}\wedge\text{-elim.}}{\varphi \wedge \psi \vdash \psi \wedge \varphi}\wedge\text{-intro.}}{\vdash (\varphi \wedge \psi) \to (\psi \wedge \varphi)}\to\text{-intro.}$$

2.2 Semantics

Definition 2.24. An **$\mathcal{L}$-structure** $\mathcal{M}$ consists of

(i) a non-empty set $|\mathcal{M}|$

(ii) for any n-ary function symbol f a function $f_{\mathcal{M}} \colon |\mathcal{M}|^n \to |\mathcal{M}|$

(iii) for any n-ary relation symbol R a set $R_{\mathcal{M}} \subseteq |\mathcal{M}|^n$

Definition 2.25. If t is an $\mathcal{L}$-term and $\sigma \colon \mathcal{V} \to |\mathcal{M}|$ we define $[\![t]\!]_{\mathcal{M},\sigma}$ as:

(i) $[\![x]\!]_{\mathcal{M},\sigma} \coloneqq \sigma(x)$

(ii) $[\![f(t_1, \ldots, t_n)]\!]_{\mathcal{M},\sigma} \coloneqq f_{\mathcal{M}}([\![t_1]\!]_{\mathcal{M},\sigma}, \ldots, [\![t_n]\!]_{\mathcal{M},\sigma})$

For formulas we define $\mathcal{M} \models_\sigma \varphi$ holds as

(i) $\mathcal{M} \models_\sigma R(t_1, \ldots, t_n)$ iff $R_{\mathcal{M}}([\![t_1]\!]_{\mathcal{M},\sigma}, \ldots, [\![t_n]\!]_{\mathcal{M},\sigma})$

(ii) $\mathcal{M} \models_\sigma \bot$ never holds

(iii) $\mathcal{M} \models_\sigma s = t$ iff $[\![s]\!]_{\mathcal{M},\sigma} = [\![t]\!]_{\mathcal{M},\sigma}$

(iv) $\mathcal{M} \models_\sigma \varphi \wedge \psi$ iff $\mathcal{M} \models_\sigma \varphi$ and $\mathcal{M} \models_\sigma \psi$

(v) $\mathcal{M} \models_\sigma \varphi \vee \psi$ iff $\mathcal{M} \models_\sigma \varphi$ or $\mathcal{M} \models_\sigma \psi$

(vi) $\mathcal{M} \models_\sigma \varphi \to \psi$ iff $\mathcal{M} \models_\sigma \varphi$ implies $\mathcal{M} \models_\sigma \psi$

(vii) $\mathcal{M} \models_\sigma \forall x.\varphi$ iff for all $a \in |\mathcal{M}|$ we know that $\mathcal{M} \models_{\sigma, \mathrm{x} \mapsto \mathrm{a}} \varphi$ where

$$(\sigma, x \mapsto a)(y) \coloneqq \begin{cases} a & y = x \\ \sigma(y) & \text{otherwise} \end{cases}$$

(viii) $\mathcal{M} \models_\sigma \exists x.\varphi$ iff there is $a \in |\mathcal{M}|$ such that $\mathcal{M} \models_{\sigma, \mathrm{x} \mapsto \mathrm{a}} \varphi$

Remark 2.26. We write $\varphi(\bar{x})$ to mean that $\mathrm{fv}(\varphi) \subseteq \bar{x}$ and $\varphi(\bar{t})$ for $\varphi[\bar{t}/\bar{x}]$.

Remark 2.27. $[\![t]\!]_{\mathcal{M},\sigma}$ and $\mathcal{M} \models_\sigma \varphi$ only depend on the values $\sigma(x)$ where $x \in \mathrm{fv}(t)$ and $x \in \mathrm{fv}(\varphi)$ respectively. If φ is a sentence then $\mathcal{M} \models_\sigma \varphi$ does not depend on σ and is denoted $\mathcal{M} \models \varphi$ (read: "$\mathcal{M}$ realizes φ").

Definition 2.28. If Γ is a set of formulas and φ is a formula then $\Gamma \models \varphi$ means that for any $\mathcal{L}$-structure $\mathcal{M}$ and assignment $\sigma \colon \mathcal{V} \to |\mathcal{M}|$ such that $\mathcal{M} \models_\sigma \psi$ for all $\psi \in \Gamma$ we have $\mathcal{M} \models_\sigma \varphi$.

Theorem 2.29 (Soundness theorem)**.** *If $\Gamma \vdash \varphi$ then $\Gamma \models \varphi$.*

Theorem 2.30 (Completeness theorem)**.** *If $\Gamma \models \varphi$ then $\Gamma \vdash \varphi$.*

Theorem 2.31 (Compactness theorem)**.** *If $\Gamma \models \varphi$ then for some finite $\Gamma' \subseteq \Gamma$ we have $\Gamma' \models \varphi$.*

2.3 Definite descriptions

Definition 2.32. $\exists! x.\varphi(x,\bar{z}) \coloneqq \exists x.(\varphi(x,\bar{z}) \wedge \forall y.\varphi(y,\bar{z}) \to y = x)$.

Definition 2.33. Suppose Γ is a set of $\mathcal{L}$-sentences, Γ' a set of $\mathcal{L}'$-sentences and $\mathcal{L} \subseteq \mathcal{L}'$. Then Γ' **is conservative over** Γ if $\Gamma \subseteq \Gamma'$ and for all $\mathcal{L}$-formulas ψ such that $\Gamma' \vdash \psi$ we have $\Gamma \vdash \psi$.

Theorem 2.34. *Suppose that $\Gamma \vdash \forall \bar{x}.\exists! \varphi(\bar{x}, y)$ and that f is a fresh function symbol (i.e. not among the function symbols of $\mathcal{L}$) then $\Gamma \cup \{\forall \bar{x}.\varphi(\bar{x}, f(\bar{x}))\}$ is conservative over Γ.*

Definition 2.35 (Axioms of ZFC)**.** $\mathcal{L}_{ZFC}$ has no function symbol and one binary relation "$\in$". The axioms of ZFC are

(i) Extensionality : $\forall x \forall y.(\forall z.z \in x \leftrightarrow z \in y) \to x = y$

(ii) Pairing: $\forall x \forall y \exists z \forall w.w \in z \leftrightarrow w = x \vee w = y$ ("$z = \{x, y\}$")

This allows us to define $\{x\} \coloneqq \{x, x\}$.

(iii) Union: $\forall x \exists y \forall z. z \in y \leftrightarrow \exists w.(w \in x \wedge z \in w)$ ("$y = \bigcup x$")

This allows us to define $x \cup y \coloneqq \bigcup \{x, y\}$.

(iv) Power set: $\forall x \exists y \forall z. z \in y \leftrightarrow w \in x$ ("$y = \mathcal{P}(x)$")

(v) Separation (axiom schema): for any formula $\varphi(\bar{x}, y)$ we have $\forall \bar{x} \forall y \exists z \forall w. w \in z \leftrightarrow (w \in y \wedge \varphi(\bar{x}, w))$ ("$z = \{w \in y \mid \varphi(\bar{x}, w)\}$")

(vi) Infinity: $\exists x. \varnothing \in x \wedge \forall y. y \in x \to y \cup \{y\} \in x$ where $\varnothing \coloneqq \{w \in y \mid \bot\}$

(vii) Foundation: $\forall x.(\exists y. y \in x) \to \exists y. y \in x \wedge \forall z. z \in x \to z \notin y$ ("Every set x contains an element y disjoint from x")

(viii) Replacement (axiom schema): For every formula $\varphi(z, w, \bar{y})$ we have $\forall x \forall \bar{y}(\forall z. z \in x \to \exists! w \varphi(z, w, \bar{y})) \to \exists u \forall w. w \in u \leftrightarrow \exists z. z \in x \wedge \varphi(z, w, \bar{y})$ ("If φ is a function with domain x then the image of φ is a set.")

(ix) Choice: $\forall x. \varnothing \notin x \to \exists f. f \in (x \to \bigcup x) \wedge \forall y. y \in x \to f(y) \in y$

where we define $(x, y) \coloneqq \{\{x\}, \{x, y\}\}$,

$A \times B \coloneqq \{z \in \mathcal{P}(\mathcal{P}(A \cup B)) \mid \exists x \in A \exists y \in B. z = (x, y)\}$,

$(A \to B) \coloneqq \{f \in \mathcal{P}(A \times B) \mid \forall x \in A \exists! y.(x, y) \in f\}$ and

$$f(x) \coloneqq \begin{cases} y & \text{if } (x, y) \in f \\ \varnothing & \text{if no such } y \text{ exists} \end{cases}$$

Remark 2.36. The existence of at least one set is provable and therefore the empty set also exists. Nonetheless, the existence of the empty set is often added as an axiom.

Remark 2.37. In principle, you can do almost all math in the combination of FOL and ZFC. In practice, you want to do meta-logical operations (e.g. quantifying over formulas). For example:

(i) You want to be able to define new definitions with definite descriptions.

(ii) You want to be able to talk about theorem schemes.

Mizar and Metamath are two proof systems implementing FOL + ZFC in a metalogic.

3 λ-calculus

Definition 3.1. Let $\mathcal{V} = \{x_0, x_1, \dots\}$ a countably infinite set of variables and $\mathbf{C} = \{c_0, c_1, \dots\}$ be any set of constants. The terms of the **λ-calculus** are given by the following BNF:

$$s, t ::= x \mid c \mid (st) \mid (\lambda x.t)$$

where x is a variable and c a constant.

Remark 3.2.

(i) Think of $(\lambda x.t)$ as the function $x \mapsto t(x)$ and (st) as function application.

(ii) Anything can apply to any term, e.g. (xx) is a term.

(iii) Abbreviate $((rs)t)u$ to $rstu$ and $\lambda x.\lambda y.\lambda z.t$ to $\lambda xyz.t$. For example, $\lambda xy.xy$ means $(\lambda x.(\lambda y.(xy)))$.

(iv) $\lambda x.t$ binds the variable x. Like in first-order logic we can define α-equivalence ($\equiv_\alpha$) and substitution ($t[s/x]$). Here we identify α-equivalent terms, e.g. $\lambda x.x = \lambda y.y$.

Example 3.3. $(x(\lambda x.x))[s/x] = s(\lambda x.x)$

Remark 3.4. Consider $(\lambda x.t)s$. We want this to correspond to $t[s/x]$.

Definition 3.5.

(i) **β-contraction ($\triangleright_\beta$)** is defined as $(\lambda x.t)s \triangleright_\beta t[s/x]$.

(ii) **One-step-β-reduction ($\to_{\beta,1}$)** is defined as the compatible closure if $\triangleright_\beta$, i.e.

  (a) If $s \triangleright_\beta t$ then $s \to_{\beta,1} t$.

  (b) If $s \to_{\beta,1} t$ then $su \to_{\beta,1} tu$, $us \to_{\beta,1} ut$ and $\lambda x.s \to_{\beta,1} \lambda x.t$.

(iii) **β-reduction ($\to_\beta$)** is the reflexive transitive closure of $\to_{\beta,1}$, i.e. it is the smallest relation that is reflexive, transitive and contains $\to_{\beta,1}$.

(iv) **β-equivalence ($\equiv_\beta$)** is the smallest equivalence relation containing $\to_\beta$.

Example 3.6.

(i) $(\lambda x.xxy)(yz) \triangleright_\beta yz(yz)y$

(ii)
$$\begin{array}{ccc} (\lambda x.xx)y((\lambda z.yz)(ww)) & \xrightarrow{\beta,1} & (\lambda x.xx)y(y(ww)) \\ \downarrow{\scriptstyle \beta,1} & & \downarrow{\scriptstyle \beta,1} \\ yy((\lambda z.yz)(ww)) & \xrightarrow{\beta,1} & yy(y(ww)) \end{array}$$

(iii) $(\lambda x.xx)(\lambda x.xx) \to_{\beta,1} (\lambda x.xx)(\lambda x.xx)$

Definition 3.7. We define the following combinators:

(i) $\mathbf{I} = \lambda x.x$

(ii) $\mathbf{K} = \lambda xy.x$

(iii) $\mathbf{K}_* = \lambda xy.y$

(iv) $\mathbf{S} = \lambda xyz.xz(yz)$

Remark 3.8. Every term can be defined using K and S up to β-equivalence.

Example 3.9. $SKK \to_\beta \lambda z.Kz(Kz) \to_\beta \lambda z.z = I$.

Proposition 3.10. *There exists a* ***fixed-point combinator*** Y *such that* $Yt \to_\beta t(Yt)$.

Proof. Let $A \coloneqq \lambda fx.x(ffx)$ and let $Y \coloneqq AA$ be the **Turing operator**. Then $Yt = AAt \to_\beta t(AAt) = t(Yt)$. □

Definition 3.11. We can define the **pairing** $\mathbf{P} \coloneqq \lambda stx.xst$ and denote $(\mathbf{s}, \mathbf{t}) \coloneqq Pst \to_\beta \lambda x.xst$.

Remark 3.12. Naming this a pairing makes sense because we have $(s,t)K \to_\beta Kst \to_\beta s$ and $(s,t)K_* \to_\beta K * (s,t) \to_\beta t$.

Definition 3.13. We can define **Church numerals**. If n is a natural number we encode it as $[\mathbf{n}] \coloneqq \lambda fx.f^n x$ where $f^0 x \coloneqq x$ and $f^{n+1}x = f(f^n x) = f^n(fx)$.

Definition 3.14. Let A be a set and $\to_1$ a binary relation on A with reflective transitive closure $\to$.

(i) $t \in A$ is **in normal form** if there is no s such that $t \to_1 s$.

(ii) $t \in A$ **has normal form** s if s is in normal form and $t \to s$.

(iii) $t \in A$ is **strongly normalizing** if there exists no infinite sequence $t \to_1 t_1 \to_1 t_2 \to_1 t_3 \to_1 \cdots$.

(iv) $\to_1$ is **(weakly) normalizing** if every $t \in A$ has a normal form.

(v) $\to_1$ is **strongly normalizing** if every $t \in A$ is strongly normalizing.

(vi) $\to_1$ is **confluent** (has the **Church-Rosser property**) if whenever $u \leftarrow t \to v$ there is an $s \in A$ with $u \to s \leftarrow v$.

$$\begin{array}{ccc} t & \longrightarrow & v \\ \downarrow & & \vdots \\ u & \dashrightarrow & s \end{array}$$

Remark 3.15. β-reduction is neither weakly nor strongly normalizing.

Proof. See the counterexamples presented in Example 3.6 (iii) and Proposition 3.10. □

Theorem 3.16 (Church-Rosser)**.** $\to_{\beta,1}$ *is confluent.*

Remark 3.17. It does not suffice to prove

$$\begin{array}{ccc} t & \xrightarrow{\beta,1} & v \\ \downarrow{\scriptstyle \beta,1} & & \vdots{\scriptstyle \beta} \\ u & \overset{\beta}{\dashrightarrow} & s \end{array}$$

i.e. for a general binary relation Theorem 3.16 does not follow from this.

Definition 3.18. **Parallel reduction ($\Rightarrow$)** is defined inductively as

(i) $x \Rightarrow x$ and $c \Rightarrow c$ where x is a variable and c is a constant.

(ii) If $t \Rightarrow t'$ then $\lambda x.t \Rightarrow \lambda x.t'$.

If $t \Rightarrow t'$ and $u \Rightarrow u'$ then

(iii) $tu \Rightarrow t'u'$.

(iv) $(\lambda x.t)u \Rightarrow t'[u'/x]$.

Lemma 3.19. *Parallel induction is reflexive, i.e.* $t \Rightarrow t$.

Proof. We show this by induction on t. Consider $t = \lambda x.s$. Then by induction hypothesis $s \Rightarrow s$, so by Definition 3.18 (ii) $\lambda x.s \Rightarrow \lambda x.s$. The rest of the cases are similar. □

Lemma 3.20. *If $t \to_{\beta,1} s$ then $t \Rightarrow s$.*

Proof. We show this by induction on $t \to_{\beta,1} s$. If $t \rhd_\beta s$, say $(\lambda x.u)v \rhd_\beta u[v/x]$. Then $(\lambda x.u)v \Rightarrow u[v/x]$ by Definition 3.18 (iv). The other cases are also easy to show. □

Lemma 3.21. *If $t \Rightarrow t'$ then $t \to_\beta t'$.*

Proof. We show this by induction on $t \Rightarrow t'$. Suppose the last rule was Definition 3.18 (iv), i.e. concluding $(\lambda x.t)u \Rightarrow t'[u'/x]$ from $t \Rightarrow t'$ and $u \Rightarrow u'$. By induction hypothesis we have $t \to_\beta t'$ and $u \to_\beta u'$. Then $(\lambda x.t)u \to_\beta (\lambda x.t')u \to_\beta (\lambda x.t')u' \to_{\beta,1} t'[u'/x]$. The other cases are similar. □

Lemma 3.22. *If $t \Rightarrow t'$ and $w \Rightarrow w'$ then $t[w/y] \Rightarrow t'[w'/y]$.*

Proof. We show this by induction on $t \Rightarrow t'$. Suppose the last step was Definition 3.18 (iv), i.e. concluding $(\lambda x.t)u \Rightarrow t'[u'/x]$ from $t \Rightarrow t'$ and $u \Rightarrow u'$. By induction hypothesis we know that $t[w/y] \Rightarrow t'[w/y]$ and $u[w/y] \Rightarrow u'[w/y]$. We have to show $(\lambda x.t[w/y])u[w/y] \Rightarrow t'[u'/x][w'/y]$. x is bound so we may assume that x is not free in w'. Then one can prove that $t'[u'/x][w'/y] = t'[w'/y][(u'[w'/y])/x]$. Now the claim follows from Definition 3.18 (iv). The rest of the cases are similar. □

Definition 3.23. If t is a term then t^* is recursively defined as

(i) $x^* = x$ and $c^* = c$ for variables x and constants c.

(ii) $(\lambda x.t)^* = \lambda x.t^*$.

(iii) $(ts)^* = t^*s^*$ if t is not a λ.

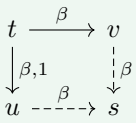
(iv) $((\lambda x.t)s)^* = t^*[s^*/x]$.

Lemma 3.24. *If $s \Rightarrow t$ then $t \Rightarrow s^*$.*

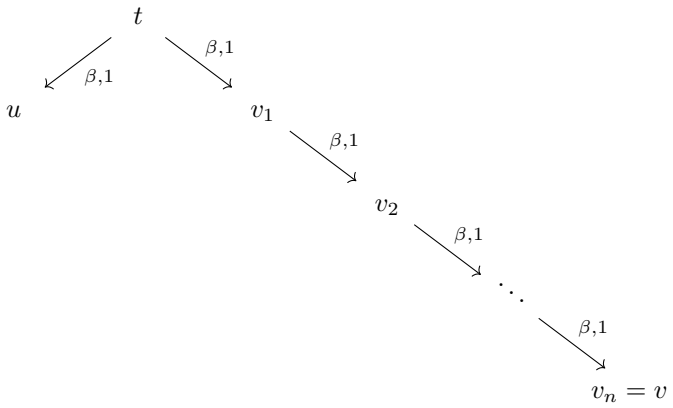
Proof. We show this by induction of the length of the derivation for $s \Rightarrow t$.
If the last step was Definition 3.18 (iii), i.e. concluding $tu \Rightarrow t'u'$ from $t \Rightarrow t'$ and $u \Rightarrow u'$, then by induction hypothesis $t' \Rightarrow t^*$ and $u' \Rightarrow u^*$. We need to show that $t'u' \Rightarrow (tu)^*$.

If t is not a λ then $(tu)^* = t^*u^*$, so we are done by Definition 3.18 (iii). If $t = \lambda x.s$ then $(tu)^* = s^*[u*/x]$. We know $\lambda x.s \Rightarrow t'$ which can only be derived using Definition 3.18 (ii). So $t' = \lambda x.s'$ with $s \Rightarrow s'$. By induction hypothesis $s' \Rightarrow s^*$. Then $(\lambda x.s')u' \Rightarrow s^*[u*/x]$ follows from Definition 3.23 (iv).
If the last step was Definition 3.18 (iv), i.e. concluding $(\lambda x.t)u \Rightarrow t'[u'/x]$ from $t \Rightarrow t'$ and $u \Rightarrow u'$, then by induction hypothesis we know that $t' \Rightarrow t^*$ and $u' \Rightarrow u^*$. Then $t'[u'/x] \Rightarrow ((\lambda x.t)u)^* = t^*[u^*/x]$ by Lemma 3.22.
The rest of the cases are easy. □

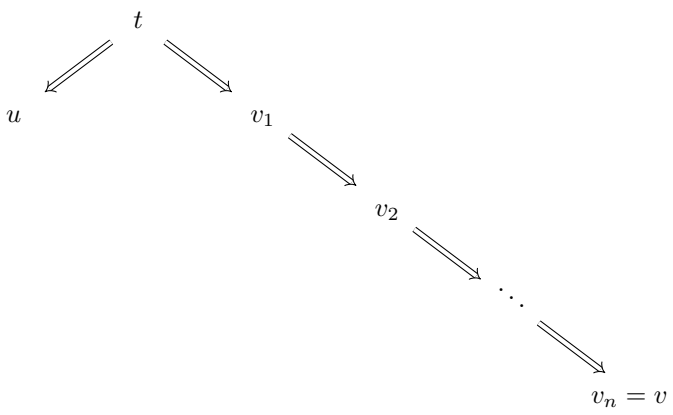
Lemma 3.25. *If $t \to_{\beta,1} u$ and $t \to_\beta v$ then there is an s with $u \to_\beta s \,{}_\beta\!\leftarrow v$.*



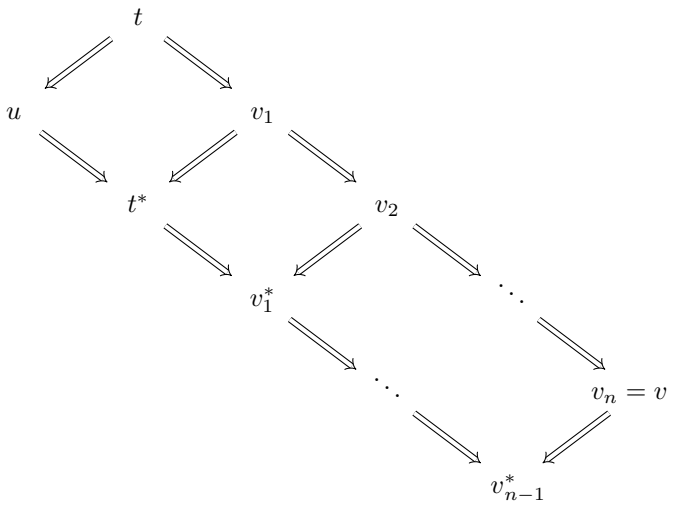
Proof. Decomposing $t \to_\beta v$ into individual steps gives us



which with Lemma 3.20 becomes



which with Lemma 3.24 yields



which implies our desired statement since by Lemma 3.21 parallel reduction implies β-reduction and β-reduction is transitive. □

Exercise 3.26. *Derive Theorem 3.16 (Church-Rosser) from Lemma 3.25.*

Corollary 3.27. *Every term t has at most one β-normal form.*

Corollary 3.28. *If $s \equiv_\beta t$, then there is u such that $s \to_\beta u \,{}_\beta\!\leftarrow t$.*

3.1 Partial recursive functions

Definition 3.29. Let $\mathbf{A \rightharpoonup B}$ be the set of **partial functions** from A to B, i.e. a partial function from A to B is a pair (X, f) such that $X \subseteq A$ and $f\colon X \to B$. We set $\mathrm{dom}(\mathbf{f}) \coloneqq X$ and call it the **domain** of f.

Notation 3.30. We set $\mathbf{f(\bar{x})\downarrow} \coloneqq \bar{x} \in \mathrm{dom}(f)$ and $\mathbf{f(\bar{x})\uparrow} \coloneqq \bar{x} \notin \mathrm{dom}(f)$.

Definition 3.31. The set of **partial recursive functions (P.R. functions)** is the subset of $\bigcup_{k\geq 0}(\mathbb{N}^k \rightharpoonup \mathbb{N})$ generated by

(i) $0\colon \mathbb{N}^0 \to \mathbb{N}$ is P.R.

(ii) $\mathrm{Succ}\colon \mathbb{N} \to \mathbb{N}, x \mapsto x+1$ is P.R.

(iii) The projection $\mathrm{P}_i^k\colon \mathbb{N}^k \to \mathbb{N}, \mathrm{P}_i^k(x_0, \ldots, x_{k-1}) = x_i$ is P.R. for $0 \leq i < k$.

(iv) Composition: if $f\colon \mathbb{N}^k \to \mathbb{N}$ is P.R. and $g_0, \ldots, g_{k-1}\colon \mathbb{N}^l \to \mathbb{N}$ are P.R. then $f \circ \bar{g}\colon \mathbb{N}^l \to \mathbb{N}, x \mapsto f(g_0(\bar{x}), \ldots, g_{k-1}(\bar{x}))$ is P.R. and $\bar{x} \in \mathrm{dom}(f \circ \bar{g})$ iff $\bar{x} \in \mathrm{dom}(g_i)$ for all i and $(g_0(\bar{x}), \ldots, g_{k-1}(\bar{x})) \in \mathrm{dom}(f)$.

(v) Primitive recursion: if $f_0\colon \mathbb{N}^k \to \mathbb{N}$ and $f_s\colon \mathbb{N}^{k+2} \to \mathbb{N}$ are P.R. then the function g defined by $g(0, \bar{x}) \coloneqq f_0(\bar{x})$ and $g(n+1, \bar{x}) \coloneqq f_s(n, g(n, \bar{x}), \bar{x})$ is P.R., $g(0, \bar{x})\downarrow$ iff $f_0(\bar{x})\downarrow$, and $g(n+1, \bar{x})\downarrow$ iff $g(n, \bar{x})\downarrow$ and $f_s(n, g(n, \bar{x}), \bar{x})\downarrow$.

(vi) Minimization (unbounded search): if $f\colon \mathbb{N}^{k+1} \to \mathbb{N}$ is P.R. then $\mu_f\colon \mathbb{N}^k \to \mathbb{N}$ is P.R. where we set $\mu_f(\bar{x}) = n$ iff $f(i, \bar{x})\downarrow$ for $i \leq n$, $f(i, \bar{x}) > 0$ for $i < n$ and $f(n, \bar{x}) = 0$, and we set $\mu_f(\bar{x})\uparrow$ if no such n exists.

Remark 3.32.

(i) The class of functions closed under the rules (i) to (iv) of Definition 3.31 are called the **primitive recursive functions**. All of them are total.

(ii) The partially recursive functions that are total are called **(totally) recursive**. Not all of them are primitive recursive functions.

(iii) The Church-Turing thesis is the claim that the partially recursive functions are precisely the functions that capture the intuitive notion of "computability".

Example 3.33.

(i) $(x, y) \mapsto x + y$, $(x, y) \mapsto x \cdot y$ and $(x, y) \mapsto x^y$ are all primitive recursive.

(ii) The **Ackermann function** is a total partial recursive function that is not primitive recursive.

Definition 3.34. A λ-term t **represents** $f\colon \mathbb{N}^k \to \mathbb{N}$ if for all $\bar{v} \in \mathbb{N}^k$ we have $t[n_0]\dots[n_{k-1}] \to_\beta [f(\bar{n})]$ if $f(\bar{n})\downarrow$ and that $t[n_0]\dots[n_{k-1}]$ has no normal form if $f(\bar{n})\uparrow$.

Proposition 3.35. *If $f\colon \mathbb{N}^k \to \mathbb{N}$ is P.R. then there are primitive recursive functions $u\colon \mathbb{N} \to \mathbb{N}$ and $t\colon \mathbb{N}^{k+1} \to \mathbb{N}$ such that $f = u \circ \mu_t$.*

Theorem 3.36. *$f\colon \mathbb{N}^k \to \mathbb{N}$ is P.R. iff it is represented by some λ-term.*

Proof sketch. For the backward direction of the proof first notice that primitive recursive functions are expressive enough to encode: pairs of natural numbers, finite sequences of natural numbers, λ-terms, substitution in λ-terms and β-reduction. Then we can show that there is a partial recursive function that searches for a reduction sequence $t \to_{\beta,1} t_1 \to_{\beta,1} t_2 \to_{\beta,1} \dots \to_{\beta,1} [m]$.

For the forward direction we first show that the claim is true for a primitive recursive function f. We now proceed inductively over the first five constructors of Definition 3.31. For (i) we see that $[0]$ represents $0\colon \mathbb{N}^0 \to \mathbb{N}$. Constructor (ii) is an exercise. For (iii) we can show that the projection P_i^k is represented by $\lambda x_0 x_1 \dots x_{k-1}.x_i$. For (iv) assume that F represents f and G_i represents g_i for all i. Then $f \circ \bar{g}$ is represented by $\lambda \bar{x}.F(G_0\bar{x})(G_1\bar{x})\dots(G_{k-1}\bar{x})$. Lastly, for (v) suppose that F_0 represents f_0 and F_s represents f_s. We want to find G such that

$$G[0]\bar{x} \equiv_\beta F_0\bar{x} \quad \text{and} \quad G[n+1]\bar{x} \equiv_\beta F_s[n](G[n]\bar{x})\bar{x}$$

This is sufficient because $G[0][\bar{n}] \equiv_\beta F_0[\bar{n}] \to_\beta [f_0(\bar{n})]$ so $G[0][\bar{n}] \to_\beta [f_0(\bar{n})]$ since $[f_0(\bar{n})]$ is in normal form. It is even enough to show that

$$G[0] \equiv_\beta F_0 \quad \text{and} \quad G[n+1] \equiv_\beta F_s[n](G[n])$$

by λ-abstraction. We want to try to encode the sequence $([0], G[0]), ([1], G[1]), \dots$ We set

$$T \coloneqq \lambda u.(\mathrm{Succ}(uK), F_s(uK)(uK_*))$$

Note that $T([n],t) \to_\beta (\mathrm{Succ}([n]), F_s[n]t) \to_\beta ([n+1], F_s[n]t)$. We will show later that means that T produces the next pair in the sequence. So we want to iterate T. Let

$$G \coloneqq \lambda v.vT([0],F_0)K_*$$

Then

$$G[0] \to_\beta [0]T([0],F_0)K_* \to_\beta ([0],F_0)K_* \to_\beta F_0$$

We want to show by induction that $[n]T([0],F_0) \equiv_\beta ([n], G[n])$. For the case $n = 0$ we get $[0]T([0],F_0) \to_\beta ([0],F_0) \;{}_\beta\!\leftarrow ([0], G[0])$. Thus $[0]T([0],F_0) \equiv_\beta ([0], G[0])$. For $n+1$ we get

$$[n+1]T([0],F_0) \to_\beta T([n]T([0],F_0)) \overset{IH}{\equiv_\beta} T([n],G[n]) \to_\beta ([n+1], F_s[n](G[n]))$$

and

$$G[n+1] \to_\beta [n+1]T([0],F_0)K_* \equiv_\beta ([n+1], F_s[n](G[n]))K* \to_\beta F_s[n](G[n])$$

Thus $[n+1]T([0],F_0) \equiv_\beta ([n+1], G[n+1])$ and consequently $G[n+1] \equiv_\beta F_s[n](G[n])$.

Now onto the general case. For minimization we use the Turing operator Y from the proof of Proposition 3.10 and the term D with $Dst[0] \to_\beta s$ and $Dst[n+1] \to_\beta t$. (The existence of D is an exercise.) Let T be a term. We define

$$W \coloneqq Y(\lambda vy.Dy(v(\mathrm{Succ}(y)))(Ty))$$

Then we get

$$W[n] \to_\beta D[n](W[n+1])(T[n]) \to_\beta \begin{cases} [n] & \text{if } T[n] \to_\beta [0] \\ W[n+1] & \text{if } T[n] \to_\beta [m+1] \text{ for some } m \in \mathbb{N} \end{cases}$$

By the previous proposition we can take primitive recursive functions u and t such that $f = u \circ \mu_t$. Suppose that U represents u and T represents t. Now we set

$$W' \coloneqq \lambda \bar{x}.Y(\lambda vy.Dy(v(\mathrm{Succ}(y)))(Ty\bar{x}))$$

and

$$F \coloneqq \lambda \bar{x}.U(W\bar{x}[0])$$

If $f(\bar{x})\downarrow$ then

$$F[\bar{n}] \to_\beta U(W'[n][0]) \to_\beta U([\mu_t(\bar{n})]) \to_\beta [f(\bar{n})]$$

and if $f(\bar{x})\uparrow$ then

$$F[\bar{x}] \to_\beta U(W'[0]) \to_\beta U(W'[1]) \to_\beta \dots$$

which is an infinite reduction sequence. We would have to show that no way of reduction leads to a normal form. This is true but we omit the proof. □

Theorem 3.37 (Normalization theorem)**.** *If t has a β-normal form, then iterated contraction of the leftmost β-redex leads to its normal form.*