

# Logic of Proof Assistants

Prof. Floris van Doorn \*

University of Bonn

9<sup>th</sup> May, 2025

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Inductive Definitions . . . . .	3
<b>2</b>	<b>First-Order Logic</b>	<b>5</b>
2.1	Provability . . . . .	8
2.2	Semantics . . . . .	9
2.3	Definite descriptions . . . . .	10
<b>3</b>	<b><math>\lambda</math>-calculus</b>	<b>12</b>

---

\*L<sup>A</sup>T<sub>E</sub>X-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  $\pi$  be the prime counting function, i.e.  $\pi: \mathbb{R} \rightarrow \mathbb{N}$ ,  $x \mapsto |\{p \leq x \mid p \text{ prime}\}|$ . Then  $\lim_{x \rightarrow \infty} \frac{\pi(x)}{x/\log(x)} = 1$ .*

**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  $S: \mathbb{N} \rightarrow \mathbb{N}$ ,  $n \mapsto n + 1$ .

**Definition 1.8.** Let  $U$  be a set and  $\mathcal{C} \subseteq \bigcup_{n \in \mathbb{N}} (U^n \rightarrow U)$  a set of **constructors**. Where  $c: U^n \rightarrow U$  is called an  **$n$ -ary function** and  $(U^n \rightarrow U)$  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)  $A \subseteq U$  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.**  $\emptyset$  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  $\sigma$ -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  $P: A \rightarrow \{\top, \perp\}$  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  $x_1, \dots, x_n \in A$  if  $P(x_1), \dots, P(x_n)$  then  $P(c(x_1, \dots, x_n))$ .*

*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  $g_c: B^n \rightarrow B$  then there is a unique function  $f: A \rightarrow B$  such that  $f(c(a_1, \dots, a_n)) = g_c(f(a_1), \dots, f(a_n))$  for every  $c \in \mathcal{C}$  and  $a_1, \dots, a_n \in A$ .*

*Proof.* Exercise. □

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

## 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  $a: \mathcal{F} \cup \mathcal{R} \rightarrow \mathbb{N}$  is the arity function.

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

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

**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**  $S := \mathcal{F} \cup \mathcal{V} \cup \{“(”, “)”, “,”, “.”\}$  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  $c_x := x$
- (ii) for each  $n$ -ary function symbol  $f$  there is an  $n$ -ary constructor  $c_f: (S^*)^n \rightarrow S^*$ ,  $c_f(t_1, \dots, t_n) := f\text{“}(\text{“}t_1\text{“}, \dots, \text{“}t_n\text{“})\text{”}$

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  $f$ “(” $x_0$ “, ” $x_1$ “)” 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

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

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)  $\neg \varphi := (\varphi \rightarrow \perp)$
- (ii)  $\varphi \leftrightarrow \psi := ((\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi))$

**Remark 2.12.** When writing formulas we omit some parentheses:

- (i)  $\varphi \rightarrow \psi \rightarrow \theta$  means  $\varphi \rightarrow (\psi \rightarrow \theta)$
- (ii)  $\forall x. \varphi \rightarrow \psi$  means  $\forall x. (\varphi \rightarrow \psi)$

**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  **$\alpha$ -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  $\varphi$  or a term  $t$  we also write **fv**( $\varphi$ ) and **fv**( $t$ ) for the set of free variables in  $\varphi$  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)  $y[t/x] := \begin{cases} t & \text{if } y = x \\ y & \text{otherwise} \end{cases}$
- (ii)  $f(s_1, \dots, s_n)[t/x] := f(s_1[t/x], \dots, s_n[t/x])$

**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  $\varphi$  **substitution**  $\varphi[t/x]$  is defined as:

- (i)  $(s = s')[t/x] := (s[t/x] = s'[t/x])$
- (ii)  $R(t_1, \dots, t_n)[t/x] := R(t_1[t/x], \dots, t_n[t/x])$
- (iii)  $(\varphi \vee \psi)[t/x] := (\varphi[t/x] \vee \psi[t/x])$
- (iv)  $(\varphi \wedge \psi)[t/x] := (\varphi[t/x] \wedge \psi[t/x])$
- (v)  $(\varphi \rightarrow \psi)[t/x] := (\varphi[t/x] \rightarrow \psi[t/x])$
- (vi)  $(\forall y.\varphi)[t/x] := \begin{cases} \forall y.\varphi & \text{if } y = x \\ \forall z.\varphi[z/y][t/x] & \text{otherwise} \end{cases}$  where  $z$  does not occur in  $t$ .
- (vii)  $(\exists y.\varphi)[t/x] := \begin{cases} \exists y.\varphi & \text{if } y = x \\ \exists z.\varphi[z/y][t/x] & \text{otherwise} \end{cases}$  where  $z$  does not occur in  $t$ .

**Definition 2.19.**  $\alpha$ -**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)  $(\varphi_1 \rightarrow \varphi_2) \equiv_\alpha (\psi_1 \rightarrow \psi_2)$  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  $\alpha$ -equivalence as an equivalence relation. You could also define the formulas as  $\Phi_{\mathcal{L}}/\equiv_\alpha$  and thus treat  $\alpha$ -equivalence as equality.

## 2.1 Provability

**Definition 2.21.** Let  $\Gamma$  be a set of formulas and  $\varphi$  a formula. Then  $\Gamma \vdash \varphi$  (read : “ $\Gamma$  proves  $\varphi$ ”) is defined inductively by

- (i)  $\overline{\Gamma, \varphi \vdash \varphi}$  (assumption rule)
- (ii)  $\frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \wedge \psi}$  ( $\wedge$ -introduction)
- (iii)  $\frac{\Gamma \vdash \varphi_1 \wedge \varphi_2}{\Gamma \vdash \varphi_i}$  for  $i = 1, 2$  ( $\wedge$ -elimination)
- (iv)  $\frac{\Gamma \vdash \varphi_i}{\Gamma \vdash \varphi_1 \vee \varphi_2}$  for  $i = 1, 2$  ( $\vee$ -introduction)
- (v)  $\frac{\Gamma \vdash \varphi \vee \psi \quad \Gamma, \varphi \vdash \theta \quad \Gamma, \psi \vdash \theta}{\Gamma \vdash \theta}$  ( $\vee$ -elimination)
- (vi)  $\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \rightarrow \psi}$  ( $\rightarrow$ -introduction)
- (vii)  $\frac{\Gamma \vdash \varphi \rightarrow \psi \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$  ( $\rightarrow$ -elimination)
- (viii)  $\frac{\Gamma, \neg\varphi \vdash \perp}{\Gamma \vdash \varphi}$  (proof by contradiction)
- (ix)  $\frac{\Gamma \vdash \varphi}{\Gamma \vdash \forall x. \varphi}$  for  $x \notin \text{fv}(\Gamma)$  ( $\forall$ -introduction)
- (x)  $\frac{\Gamma \vdash \forall x. \varphi}{\Gamma \vdash \varphi[t/x]}$  ( $\forall$ -elimination)
- (xi)  $\frac{\Gamma \vdash \varphi[t/x]}{\Gamma \vdash \exists x. \varphi}$  ( $\exists$ -introduction)
- (xii)  $\frac{\Gamma \vdash \exists x. \varphi \quad \Gamma, \varphi \vdash \psi}{\Gamma \vdash \psi}$  for  $x \notin \text{fv}(\Gamma, \psi)$  ( $\exists$ -elimination)
- (xiii)  $\overline{\Gamma \vdash t = t}$  ( $=$ -introduction)
- (xiv)  $\frac{\Gamma \vdash s = t \quad \Gamma \vdash \varphi[t/x]}{\Gamma \vdash \varphi[s/x]}$  ( $=$ -elimination)
- (xv)  $\frac{\Gamma \vdash \varphi}{\Gamma \vdash \psi}$  for  $\varphi \equiv_\alpha \psi$  ( $\alpha$ -equivalence)

**Remark 2.22.** Read “ $\frac{A}{B}$ ” as: “Under the assumptions A we can prove B”. With “ $\Gamma, \varphi$ ” we really mean  $\Gamma \cup \{\varphi\}$ .



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

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

## 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  $f_{\mathcal{M}}: |\mathcal{M}|^n \rightarrow |\mathcal{M}|$
- (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  $\sigma: \mathcal{V} \rightarrow |\mathcal{M}|$  we define  $\llbracket t \rrbracket_{\mathcal{M}, \sigma}$  as:

- (i)  $\llbracket x \rrbracket_{\mathcal{M}, \sigma} := \sigma(x)$
- (ii)  $\llbracket f(t_1, \dots, t_n) \rrbracket_{\mathcal{M}, \sigma} := f_{\mathcal{M}}(\llbracket t_1 \rrbracket_{\mathcal{M}, \sigma}, \dots, \llbracket t_n \rrbracket_{\mathcal{M}, \sigma})$

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

- (i)  $\mathcal{M} \models_{\sigma} R(t_1, \dots, t_n)$  iff  $R_{\mathcal{M}}(\llbracket t_1 \rrbracket_{\mathcal{M}, \sigma}, \dots, \llbracket t_n \rrbracket_{\mathcal{M}, \sigma})$
- (ii)  $\mathcal{M} \models_{\sigma} \perp$  never holds
- (iii)  $\mathcal{M} \models_{\sigma} s = t$  iff  $\llbracket s \rrbracket_{\mathcal{M}, \sigma} = \llbracket t \rrbracket_{\mathcal{M}, \sigma}$
- (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)  $\mathcal{M} \models_{\sigma} \varphi \rightarrow \psi$  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  $\mathcal{M} \models_{\sigma, x \mapsto a} \varphi$  where

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

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

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

**Remark 2.27.**  $\llbracket t \rrbracket_{\mathcal{M}, \sigma}$  and  $\mathcal{M} \models_{\sigma} \varphi$  only depend on the values  $\sigma(x)$  where  $x \in \text{fv}(t)$  and  $x \in \text{fv}(\varphi)$  respectively. If  $\varphi$  is a sentence then  $\mathcal{M} \models_{\sigma} \varphi$  does not depend on  $\sigma$  and is denoted  $\mathcal{M} \models \varphi$  (read: “ $\mathcal{M}$  realizes  $\varphi$ ”).

**Definition 2.28.** If  $\Gamma$  is a set of formulas and  $\varphi$  is a formula then  $\Gamma \models \varphi$  means that for any  $\mathcal{L}$ -structure  $\mathcal{M}$  and assignment  $\sigma: \mathcal{V} \rightarrow |\mathcal{M}|$  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.**  $\exists! x. \varphi(x, \bar{z}) := \exists x. (\varphi(x, \bar{z}) \wedge \forall y. \varphi(y, \bar{z}) \rightarrow y = x)$ .

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

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

- (i) Extensionality :  $\forall x \forall y. (\forall z. z \in x \leftrightarrow z \in y) \rightarrow x = y$
- (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  $\{x\} := \{x, x\}$ .

- (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  $x \cup y := \bigcup \{x, y\}$ .

- (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:  $\exists x. \emptyset \in x \wedge \forall y. y \in x \rightarrow y \cup \{y\} \in x$  where  $\emptyset := \{w \in y \mid \perp\}$

- (vii) Foundation:  $\forall x. (\exists y. y \in x) \rightarrow \exists y. y \in x \wedge \forall z. z \in x \rightarrow z \not\in y$  (“Every set  $x$  contains an element  $y$  disjoint from  $x$ ”)

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

- (ix) Choice:  $\forall x. \emptyset \notin x \rightarrow \exists f. f \in (x \rightarrow \bigcup x) \wedge \forall y. y \in x \rightarrow f(y) \in y$

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

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

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

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

**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 $\lambda$ -calculus

**Definition 3.1.** Let  $\mathcal{V} = \{x_0, x_1, \dots\}$  a countably infinite set of variables and  $\mathcal{C} = \{c_0, c_1, \dots\}$  be any set of constants. The terms of the  $\lambda$ -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  $\alpha$ -equivalence ( $\equiv_\alpha$ ) and substitution  $t[s/x]$ . Here we identify  $\alpha$ -equivalent terms, i.e.  $\lambda x.x = \lambda y.y$ .

**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)  **$\beta$ -contraction** ( $\triangleright_\beta$ ) is defined as  $(\lambda x.t)s \triangleright_\beta t[s/x]$ .
- (ii) **One-step- $\beta$ -reduction** ( $\rightarrow_{\beta,1}$ ) is defined as the compatible closure of  $\triangleright_\beta$ , i.e.
  - (a) If  $s \triangleright_\beta t$  then  $s \rightarrow_{\beta,1} t$ .
  - (b) If  $s \rightarrow_{\beta,1} t$  then  $su \rightarrow_{\beta,1} tu$ ,  $us \rightarrow_{\beta,1} ut$  and  $\lambda x.s \rightarrow_{\beta,1} \lambda x.t$ .
- (iii)  **$\beta$ -reduction** ( $\rightarrow_\beta$ ) is the reflexive transitive closure of  $\rightarrow_{\beta,1}$ , i.e. it is the smallest relation that is reflexive, transitive and contains  $\rightarrow_{\beta,1}$ .
- (iv)  **$\beta$ -equivalence** ( $\equiv_\beta$ ) is the smallest equivalence relation containing  $\rightarrow_\beta$ .

**Example 3.6.**

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

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

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

**Definition 3.7.** We define the following combinators:

- (i) **I** =  $\lambda x.x$
- (ii) **K** =  $\lambda xy.x$
- (iii) **K**<sub>\*</sub> =  $\lambda xy.y$
- (iv) **S** =  $\lambda xyz.xz(yz)$

**Remark 3.8.** Every term can be defined using  $K$  and  $S$  up to  $\beta$ -equivalence.

**Example 3.9.**  $SKK \rightarrow_{\beta} \lambda z.Kz(Kz) \rightarrow_{\beta} \lambda z.z = I$ .

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

*Proof.* Let  $A := \lambda fx.x(ffx)$  and let  $Y := AA$  be the **Turing operator**. Then  $Yt = AAt \rightarrow_{\beta} t(AAt) = t(Yt)$ .  $\square$

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

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

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

**Definition 3.14.** Let  $A$  be a set and  $\rightarrow_1$  a binary relation on  $A$  with reflective transitive closure  $\rightarrow$ .

- (i)  $t \in A$  is **in normal form** if there is no  $s$  such that  $t \rightarrow_1 s$ .
- (ii)  $t \in A$  **has normal form**  $s$  if  $s$  is in normal form and  $t \rightarrow s$ .

- (iii)  $t \in A$  is **strongly normalizing** if there exists no infinite sequence  $t \rightarrow_1 t_1 \rightarrow_1 t_2 \rightarrow_1 t_3 \rightarrow_1 \dots$ .
- (iv)  $\rightarrow_1$  is **(weakly) normalizing** if every  $t \in A$  has a normal form.
- (v)  $\rightarrow_1$  is **strongly normalizing** if every  $t \in A$  is strongly normalizing.
- (vi)  $\rightarrow_1$  is **confluent** (has the **Church-Rosser property**) if whenever  $u \leftarrow t \rightarrow v$  there is an  $s \in A$  with  $u \rightarrow s \leftarrow v$ .

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

**Remark 3.15.**  $\beta$ -reduction is neither weakly nor strongly normalizing.

*Proof.* See the counterexamples presented in Example 3.6 (iii) and Proposition 3.10.  $\square$

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

**Remark 3.17.** It does not suffice to prove

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

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.  $\square$

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

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

**Lemma 3.21.** *If  $t \Rightarrow t'$  then  $t \rightarrow_{\beta} t'$ .*

*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  $t \rightarrow_{\beta} t'$  and  $u \rightarrow_{\beta} u'$ . Then  $(\lambda x.t)u \rightarrow_{\beta} (\lambda x.t')u \rightarrow_{\beta} (\lambda x.t')u' \rightarrow_{\beta,1} t'[u'/x]$ . The other cases are similar.  $\square$

**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.  $\square$

**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  $\lambda$ .
- (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  $\lambda$  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).

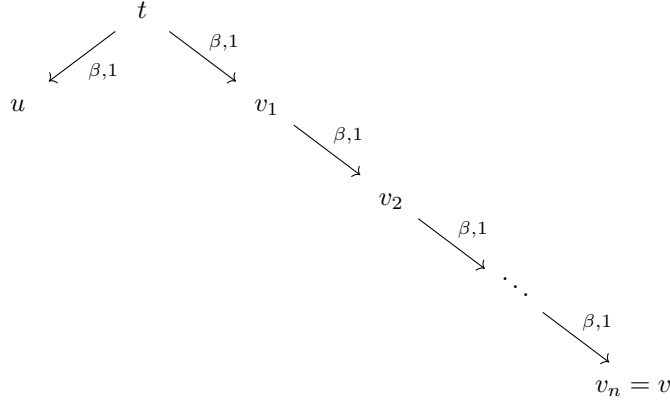
If the last step was Definition 3.18 (iv), i.e. concluding  $(\lambda x.t)u \Rightarrow t'[u'/x]$  from  $t \Rightarrow t'$  and  $u \Rightarrow u'$ , then by induction hypothesis we know that  $t' \Rightarrow t^*$  and  $u' \Rightarrow u^*$ . Then  $t'[u'/x] \Rightarrow ((\lambda x.t)u)^* = t^*[u^*/x]$  by Lemma 3.22.

The rest of the cases are easy. □

**Lemma 3.25.** *If  $t \rightarrow_{\beta,1} u$  and  $t \rightarrow_{\beta} v$  then there is an  $s$  with  $u \rightarrow_{\beta} s \leftarrow_{\beta} v$ .*

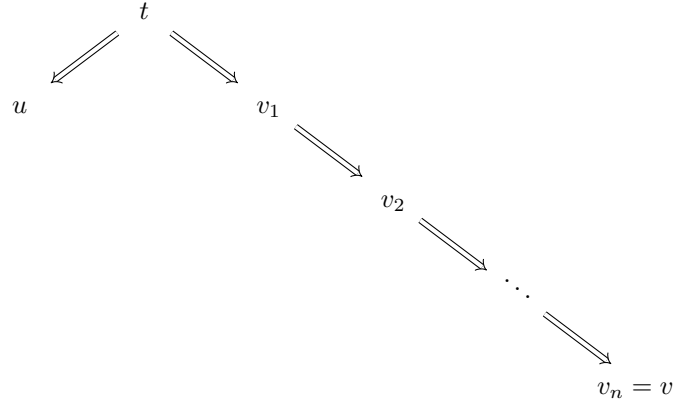
$$\begin{array}{ccc} t & \xrightarrow{\beta} & v \\ \downarrow \beta,1 & & \downarrow \beta \\ u & \dashrightarrow & s \end{array}$$

*Proof.* Decomposing  $t \rightarrow_{\beta} v$  into individual steps gives us

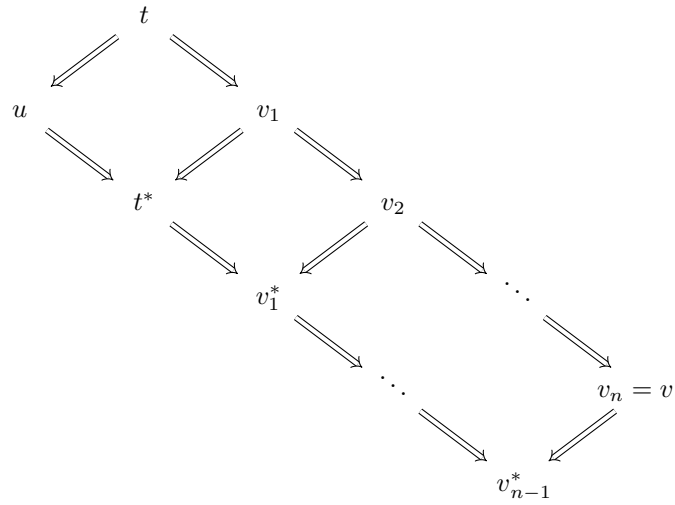




which with Lemma 3.20 becomes



which with Lemma 3.24 yields



which implies our desired statement since by Lemma 3.21 parallel reduction implies  $\beta$ -reduction and  $\beta$ -reduction is transitive.  $\square$

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