HA^ω, Constructive Reals and λ-Calculus.

Lecture notes of the second half of a course on constructive mathematics and λ-calculus, held at Uppsala University, autumn 1998.

Anton Setzer

January 18, 1999

Contents

Chapter 0

Introduction

These lecture notes contain the later parts of my lecture "Constructive Mathematics and λ-calculus", held in autumn 1998 at the Department of Mathematics, Uppsala University. The parts on HA_ω and real numbers in constructive mathematics are very much based on [TD88a] and [TD88b] and the parts about the λ-calculus are based on [HS86]. Numbers in brackets refer to the books, respectively.
In the first part the following parts of [TD88a] were treated (the numbers in brackets refer to that book):

- 1. Introduction (1)
  - 1.1. Examples of non-constructive proofs (1.2.)
  - 1.2. Directions in the foundations of mathematics (1.1, 1.4)
  - 1.3. The Brouwer-Heyting-Kolmogorov (BHK) interpretation of the logical connectives (1.3.1)
  - 1.4. Brouwerian counter examples (1.3.2 - 1.3.7)
- 2. Predicate logic and constructivism (2)
  - 2.1. Natural deduction and intuitionistic logic (2.1, 2.3.2)
  - 2.2. Logic with existence predicate (2.2.1. - 2.2.4)
  - 2.3. The double negation translation (2.3.1. - 2.3.8)
  - 2.4. Kripke Semantics (2.5.1 - 2.5.9, 2.5.11, 2.5.13)
  - 2.5. Soundness and completeness for Kripke Semantics (2.6.)
- 3. Arithmetic (3)
  - 3.1. Primitive recursive arithmetic (PRA) (3.2.)
  - 3.2. Heyting Arithmetic (HA) (3.3.)
  - 3.3. Friedman's A-translation (3.5.1 - 4)

- 3.4. Disjunction property and explicit definability for numbers in HA (3.5.6 - 3.5.12, 3.5.14 - 3.5.17)
- 3.5. Kleene-realizability (4.4)

The notes are currently not very well checked, they are still on the way to be written. Therefore I welcome comments very much.

Chapter 1

HA$^\omega$ and constructive reals ([TD88a, TD88b])

1.1 HA$^\omega$ (9.1.1 - 9.1.14)

Definition 1.1.1 (a) Let $\mathcal{G}$ be a non-empty set. The elements of $\mathcal{G}$ are called ground types.

The set of (finite) types $\mathcal{T}_\mathcal{G}$ w.r.t. $\mathcal{G}$ is inductively defined by

- $o \in \mathcal{G} \Rightarrow o \in \mathcal{T}_\mathcal{G}$.
- $\sigma, \tau \in \mathcal{T}_\mathcal{G} \Rightarrow (\sigma \times \tau), (\sigma \rightarrow \tau) \in \mathcal{T}_\mathcal{G}$.

The elements of $\mathcal{T}_\mathcal{G}$ are called types. Let in the following $\mathcal{G}$ be fixed and let σ, τ, ρ possibly with accents or subscripts denote elements of $\mathcal{T}_\mathcal{G}$, o be elements of $\mathcal{T}_\mathcal{G}$.

We will omit brackets:

- $\sigma \rightarrow \tau \rightarrow \rho := \sigma \rightarrow (\tau \rightarrow \rho)$.
- $\sigma \times \tau \times \rho := \sigma \times (\tau \times \rho)$.

(b) A language $\mathcal{L}$ w.r.t. $\mathcal{T}_\mathcal{G}$ consists of a set of function symbols f^σ for every type σ, and a set of relation symbols $R^{\sigma_1, \cdots, \sigma_n}$ for every finite sequence of types $\sigma_1, \ldots, \sigma_n$.

In the following f^σ, g^σ, h^σ, function symbols of type σ, $R^{\sigma_1, \cdots, \sigma_n}$, relation symbols of type σ, both possibly with accents or subscripts.

We will only mention the types of a (function-, relation-) symbol (or variable or later term) the first time it occurs, writing f instead of f^σ, x instead of x^σ etc.

(c) The set of terms of the language $\mathcal{L}$ is given by:

- x^σ is a term of type σ, where we have infinitely many variables, denoted by x^σ, y^σ, z^σ, possibly with superscripts for every type σ, and omit after the occurrence the superscript σ.
- f^σ is a term of type σ.
- If $s^{\sigma\to\tau}$, t^σ are terms of type $\sigma\to\tau$, σ, $(s\ t)$ is a term of type τ.
- If s^σ, t^τ are terms of types σ, τ, then $\langle s, t\rangle$ is a term of type $\sigma \times \tau$.
- If $s^{\sigma\times\tau}$ is a term of type $\sigma\times\tau$, then $s\ 0$ is a term of type σ and $s\ 1$ is a term of type τ.

In the following r^σ, s^σ, t^σ denote terms of type σ, (as before with subscripts, accents and we will omit the superscript σ after the first occurrence).

$s_1 \cdots s_n := (\cdots (s_1\ s_2) \cdots s_n)$.

(d) The set of prime formulas for a language $\mathcal{L}$ as before is given by:

- $\perp$ is a prime formula.
- $s^\sigma =_\sigma t^\sigma$ is a prime formula.
- $R^{\sigma_1,\dots,\sigma_n}(s^{\sigma_1},\dots,s^{\sigma_n})$ is a prime formula.

(e) The set of formulas $\mathcal{L}$ is given by:

- If A is a prime formula, then A is a formula.
- If A, B are formulas, x^σ is a variable, then $(A\wedge B)$, $(A\vee B)$, $(A\to B)$, $(\forall x^\sigma.A)$, $(\exists x^\sigma.A)$ are formulas.

(f) Substitution, substitutability, free variables etc. are defined as usual.

Definition 1.1.2 (a) A domain for the finite types $\mathcal{T}_\mathcal{G}$ is a tuple

$$\langle (M_\sigma)_{\sigma\in\mathcal{T}_\mathcal{G}}, (=_\sigma)_{\sigma\in\mathcal{T}_\mathcal{G}}, (\mathrm{Ap}_{\sigma,\rho})_{\sigma,\rho\in\mathcal{T}_\mathcal{G}}, (\pi_{\sigma,\rho})_{\sigma,\rho\in\mathcal{T}_\mathcal{G}}, (\mathrm{pr}_{0,\sigma,\rho})_{\sigma,\rho\in\mathcal{T}_\mathcal{G}}, (\mathrm{pr}_{1,\sigma,\rho})_{\sigma,\rho\in\mathcal{T}_\mathcal{G}}\rangle$$

such that

- M_σ is a set,
- $M_o \neq \emptyset$ for some $o \in \mathcal{G}$,
- $=_\sigma$ is an equivalence relation on M_σ written usually infix,
- $\mathrm{Ap}_{\sigma,\rho} \in (M_{\sigma\to\rho} \times M_\sigma)\to M_\rho$,
- $\pi_{\sigma,\rho} \in (M_\sigma \times M_\rho)\to M_{\sigma\times\rho}$,
- $\mathrm{pr}_{0,\sigma,\rho} \in M_{\sigma\times\rho}\to M_\sigma$,
- $\mathrm{pr}_{1,\sigma,\rho} \in M_{\sigma\times\rho}\to M_\rho$,
- $\forall a \in M_\sigma.\forall b \in M_\rho(\mathrm{pr}_{0,\sigma,\rho}(\pi_{\sigma,\rho}(a, b)) =_\sigma a \wedge \mathrm{pr}_{1,\sigma,\rho}(\pi_{\sigma,\rho}(a, b)) =_\rho b)$,
- $\forall a \in M_{\sigma\times\rho}.a =_{\sigma\times\rho} \pi_{\sigma,\rho}(\mathrm{pr}_{0,\sigma,\rho}(a), \mathrm{pr}_{1,\sigma,\rho}(a))$.

and $\mathrm{Ap}_{\sigma,\rho}$, $\pi_{\sigma,\rho}$, $\mathrm{pr}_{0,\sigma,\rho}$, $\mathrm{pr}_{1,\sigma,\rho}$ respect equality, i.e.

- $\forall a, b \in M_{\sigma,\rho}.\forall c, d \in M_\sigma (a =_{\sigma\to\rho} b \wedge c =_\sigma d) \Rightarrow \mathrm{Ap}_{\sigma,\rho}(a, c) =_\rho \mathrm{Ap}_{\sigma,\rho}(b, d)$,
- similarly for the $\pi_{\sigma,\rho}$, $\mathrm{pr}_{0,\sigma,\rho}$, $\mathrm{pr}_{1,\sigma,\rho}$.

(b) If not mentioned differently, a domain A for $\mathcal{T}_\mathcal{G}$ will be of the form

$$A = \langle (A_\sigma)_{\sigma\in\mathcal{T}_\mathcal{G}}, (=_\sigma)_{\sigma\in\mathcal{T}_\mathcal{G}}, (\mathrm{Ap}_{\sigma,\rho})_{\sigma,\rho\in\mathcal{T}_\mathcal{G}}, (\pi_{\sigma,\rho})_{\sigma,\rho\in\mathcal{T}_\mathcal{G}}, (\mathrm{pr}_{0,\sigma,\rho})_{\sigma,\rho\in\mathcal{T}_\mathcal{G}}, (\mathrm{pr}_{1,\sigma,\rho})_{\sigma,\rho\in\mathcal{T}_\mathcal{G}} \rangle$$

(c) A finite type structure $\mathcal{M}$ for the language $\mathcal{L}$ w.r.t. $\mathcal{T}_\mathcal{G}$ consists of

- a domain M for $\mathcal{T}_\mathcal{G}$,
- for each function symbol f^σ of $\mathcal{L}$ an element $f^\mathcal{M} \in M_\sigma$;
- for each relation symbol $R^{\sigma_1,\cdots,\sigma_n}$ a relation $R^\mathcal{M}$ on $M_{\sigma_1} \times \cdots \times M_{\sigma_1}$.

s. t.

$$\forall a_1^{\sigma_1}, \dots, a_n^{\sigma_n}, b_1^{\sigma_1}, \dots, b_n^{\sigma_n}. \qquad \begin{array}{l}(a_1 =_{\sigma_1} b_1 \wedge \cdots \wedge a_n =_{\sigma_n} b_n) \Rightarrow \\ (R^{\sigma_1,\cdots,\sigma_n,\mathcal{M}}(a_1, \dots, a_n) \Leftrightarrow R^{\sigma_1,\cdots,\sigma_n,\mathcal{M}}(b_1, \dots, b_n))\end{array} \ .$$

(d) An assignment w.r.t. a finite type structure $\mathcal{M}$ as above is a function ξ mapping variables x_σ to elements of M_σ.

(e) A model of finite type structure $\mathcal{L}_\mathcal{G}$ and language $\mathcal{L}$ is pair $\langle \mathcal{M}, \xi \rangle$ where $\mathcal{M}$ is a finite type structure for $\mathcal{L}$ and ξ is an assignment for it.

(f) If $\langle \mathcal{M}, \xi \rangle$ is a model for $\mathcal{T}_\mathcal{G}$, we define the interpretation $t^{\sigma\mathcal{M}}[\xi]$ of terms t^σ under it, which will be an element of M_σ:

- $x^\mathcal{M}[\xi] := \xi(x)$.
- $f^\mathcal{M}[\xi] := f^\mathcal{M}$.
- $(s^{\sigma\to\tau} t^\sigma)^\mathcal{M}[\xi] := \mathrm{Ap}_{\sigma,\tau}(s^\mathcal{M}[\xi], t^\mathcal{M}[\xi])$.
- $(\langle s^\sigma, t^\tau \rangle)^\mathcal{M}[\xi] := \pi_{\sigma,\tau}(s^\mathcal{M}[\xi], t^\mathcal{M}[\xi])$.
- $(s^{\sigma\times\tau}\ 0)^\mathcal{M}[\xi] := \mathrm{pr}_{0,\sigma,\tau}(s^\mathcal{M}[\xi])$.
- $(s^{\sigma\times\tau}\ 1)^\mathcal{M}[\xi] := \mathrm{pr}_{1,\sigma,\tau}(s^\mathcal{M}[\xi])$.

(g) For formulas A of $\mathcal{L}$ we define whether $\mathcal{M} \models A[\xi]$, where $\langle \mathcal{M}, \xi \rangle$ is a model of $\mathcal{L}$ as follows

- $\mathcal{M} \models \bot[\xi] :\Leftrightarrow \bot$.
- $\mathcal{M} \models s^\sigma =_\sigma t^\sigma[\xi] :\Leftrightarrow s^\mathcal{M}[\xi] =_\sigma t^\mathcal{M}[\xi]$.
- $\mathcal{M} \models R^{\sigma_1,\cdots,\sigma_n} s_1 \cdots s_n[\xi] :\Leftrightarrow R^\mathcal{M}(s_1^{\sigma_1}[\xi], \dots, s_n^{\sigma_n}[\xi])$.
- $\mathcal{M} \models A \wedge B[\xi] :\Leftrightarrow \mathcal{M} \models A[\xi] \wedge \mathcal{M} \models B[\xi]$,
  similar for $\vee, \to, \forall, \exists$.

(h) $\mathcal{M} \models A$, $\models A$ etc. is defined as usual. Note that validity will be preserved under logical and equality rules.

Definition 1.1.3 (a) The language of HA$^\omega$ is the language for the set of finite types $\mathcal{T}_{\text{nat}}$ with constants

- 0^{nat},
- $\mathrm{S}^{\text{nat}\to\text{nat}}$,
- $\mathbf{k}_{\sigma,\tau}^{\sigma\to(\tau\to\sigma)}$,
- $\mathbf{s}_{\sigma,\rho,\tau}^{(\sigma\to\tau\to\rho)\to(\sigma\to\tau)\to(\sigma\to\rho)}$,
- $\mathrm{p}_{\sigma,\rho}^{\sigma\to\rho\to(\sigma\times\rho)}$,
- $\mathbf{proj}_{0,\sigma,\rho}^{(\sigma\times\rho)\to\sigma}$,
- $\mathbf{proj}_{1,\sigma,\rho}^{(\sigma\times\rho)\to\rho}$,
- $\mathbf{R}_{\sigma}^{\sigma\to(\sigma\to\text{nat}\to\sigma)\to\text{nat}\to\sigma}$,

and no relations (apart from equality).

(b) The rules and axioms of HA$^\omega$ are

- intuitionistic propositional logic with equality based on many sorts $\mathcal{T}_{\{\text{nat}\}}$;
- equations for the type structures:
  - $\forall x^\sigma, y^\tau.(\langle x, y\rangle 0 = x \wedge \langle x, y\rangle 1 = y)$,
  - $\forall x^{\sigma\times\tau}.x = \langle x0, x1\rangle$;
- defining equations for the constants:
  - $\forall x^\sigma, y^\tau.\mathbf{k}_{\sigma,\tau}\ x\ y =_\sigma x^\sigma$.
  - $\forall x^{\sigma\to\tau\to\rho}, y^{\sigma\to\tau}, z^\sigma.\mathbf{s}_{\sigma,\rho,\tau}\ x\ y\ z =_\tau (x\ z)\ (y\ z)$.
  - $\forall x^\sigma.y^\rho \mathrm{p}_{\sigma,\rho}\ x\ y =_{\sigma\times\rho} \langle x, y\rangle$.
  - $\forall x^{\sigma\times\rho}.(\mathbf{proj}_{0,\sigma,\rho}\ x = x0 \wedge \mathbf{proj}_{1,\sigma,\rho}\ x = x1)$.
  - $\forall x^\sigma, y^{\sigma\to\text{nat}\to\sigma}, z^{\text{nat}}.(\mathbf{R}_\sigma\ x\ y\ 0 =_\sigma x \wedge \mathbf{R}_\sigma\ x\ y\ \mathrm{S}(z) =_\sigma y\ (\mathbf{R}_\sigma\ x\ y\ z)\ z)$.
- Arithmetical axioms:
  - $\forall x^{\text{nat}}, y^{\text{nat}}.(\mathrm{S}\ x^{\text{nat}} =_{\text{nat}} \mathrm{S}\ y^{\text{nat}} \to x^{\text{nat}} =_{\text{nat}} y^{\text{nat}})$.
  - $\forall x^{\text{nat}}.\neg(0 =_{\text{nat}} \mathrm{S}\ x^{\text{nat}})$.
  - If $\mathrm{FV}(A(x, \vec{y}) \subseteq \{x, \vec{y}\}$, then

$$\forall \vec{y}(A(0, \vec{y}) \to \forall x^{\text{nat}}(A(x, \vec{y}) \to A(\mathrm{S}\ x, \vec{y})) \to \forall x^{\text{nat}} A(x, \vec{y}) \ .$$

1.1.1 λ-terms in HA$^\omega$

Definition 1.1.4 ‘

(a) For terms t^ρ in $\mathcal{L}_{\mathrm{HA}}$ and variables x^σ we define $(\lambda x.t)^{\sigma\to\rho}$ as follows:

First replace in t occurrences of $\langle r, s\rangle$, $r0$, $r1$ by $\mathrm{p}_{\tau,\tau'}\ r\ s$, $\mathbf{proj}_{0,\tau,\tau'}\ r$, $\mathbf{proj}_{1,\tau,\tau'}$ for appropriate τ, τ'. Let r be the resulting term, which is an element of the set Term' of terms, which do not have $\langle s, t\rangle$, $s0$, $s1$ as subterms. Then $\lambda x.t := \lambda' x.r$, where for terms $r \in \mathrm{Term}'$ $\lambda' x.r$ is defined by induction on the length of r as follows:

- Case: $x \notin \mathrm{FV}(r)$. $\lambda' x.r := \mathbf{k}\ r$.
- Case $r \equiv s\ x$, $x \notin \mathrm{FV}(s)$. $\lambda' x.r := s$.
- Otherwise
  - Subcase $r = x^\sigma$, $\sigma = \rho$.

$$\lambda' x.r := \mathbf{s}^{(\sigma\to(\sigma\to\sigma)\to\sigma)\to(\sigma\to\sigma\to\sigma)\to\sigma\to\sigma}_{\sigma,\sigma\to\sigma,\sigma}\mathbf{k}^{\sigma\to(\sigma\to\sigma)\to\sigma}_{\sigma,\sigma\to\sigma}\mathbf{k}^{\sigma\to\sigma\to\sigma}_{\sigma,\sigma} \quad .$$

  - Subcase $r = s^{\tau\to\rho}\ t^\tau$.

$$\lambda' x.r := \mathbf{s}_{\sigma,\tau,\rho}(\lambda' x.s)^{\sigma\to\tau\to\rho}(\lambda' x.t)^{\sigma\to\tau} \quad .$$

Note that $\lambda' x.t = \lambda x.t$ for $t \in \mathrm{Term}'$, therefore we write in the following λ instead of λ'.

(b) $\lambda x_1, \ldots, x_n.t := \lambda x_1.\lambda x_2. \ldots \lambda x_n.t$.
$\lambda\vec{x}.t := \lambda x_1, \ldots, x_n.t$, if $\vec{x} = x_1, \ldots, x_n$.

Proposition 1.1.5 *(9.1.8) Assume t^ρ is a term, x^σ a variable. Then the following follows:*

(a) $\mathrm{HA}^\omega \vdash \forall y^\sigma.(\lambda x.t)\ y =_\rho t[x := y]$.

(b) $\mathrm{FV}(\lambda x.t) \subseteq \mathrm{FV}(t) \setminus \{x\}$.

(c) If $t = t'x$, $x \notin \mathrm{FV}(t')$, then $\lambda x.t = t'$.

Proof: Let s be defined for t as in Definition 1.1.4. Then, if the assertion holds for s, it holds for t as well, so prove the assertion for s by induction on s.
(c) follows immediately by definition and (b) follow easily using the IH.
Proof of (a):

- Case $x \notin \mathrm{FV}(s)$.

$$(\lambda x.s)\ y \equiv \mathbf{k}\ s\ y = s \equiv s[x := y] \quad .$$

- Case $s = s'\ x$, $x \notin \mathrm{FV}(s')$.

$$(\lambda x.s)\ y \equiv s'\ y \equiv s[x := y] \quad .$$

- Otherwise.
  - Subcase $s = x$.

$$(\lambda x.s)\ y \equiv \mathbf{s}\ \mathbf{k}\ \mathbf{k}\ y = \mathbf{k}\ y\ (\mathbf{k}\ y) = y \equiv s[x := y]\ \ .$$

  - Subcase $s = s_1\ s_2$.

$$\begin{aligned}(\lambda x.s)\ y &\equiv \mathbf{s}\ (\lambda x.s_1)\ (\lambda x.s_2)\ y\\ &= (\lambda x.s_1)\ y\ ((\lambda x.s_2)\ y)\\ &= s_1[x := y]\ s_2[x := y] \equiv s[x := y]\ \ .\end{aligned}$$

Exercise 1.1 *(a) For every types σ, ρ, τ, terms r^τ, t^ρ, variables x^σ, z^ρ, s.t.$z \not\equiv x$ and $z \notin \mathrm{FV}(t) \vee x \notin \mathrm{FV}(r)$,*

$$\mathrm{HA}^\omega \vdash (\lambda x.(t[z := r])) =_{\sigma\to\rho} (\lambda x.t)[z := r]\ \ .$$

(b) For every types σ, ρ, term t^ρ, variables x^σ, y^σ s.t.$y \notin \mathrm{FV}(t)$, it follows

$$\lambda x.t \equiv \lambda y.(t[x := y])\ \ .$$

1.1.2 The theories $\mathrm{E}-\mathrm{HA}^\omega$, $\mathrm{I}-\mathrm{HA}^\omega$.

Definition 1.1.6 *(9.1.11)*

(a) $\mathrm{E}-\mathrm{HA}^\omega$ is HA^ω with extensional equality:

$\mathrm{E}-\mathrm{HA}^\omega = \mathrm{HA}^\omega$ extended by the axioms (for every types σ, τ

$$(\mathrm{EXT})\quad \forall y^{\sigma\to\tau}, z^{\sigma\to\tau}(\forall x^\sigma(y\ x =_\tau z\ x) \to y =_{\sigma\to\tau} z)\ \ .$$

One easily verifies that the full type structure and HEO are models of $\mathrm{E}-\mathrm{HA}^\omega$.

(b) $\mathrm{I}-\mathrm{HA}^\omega$ is HA^ω with intensional equality:

$\mathrm{I}-\mathrm{HA}^\omega = \mathrm{HA}^\omega$ extended by additional function symbols $\mathrm{e}^{\sigma\to\sigma\to\mathrm{nat}}$ and the following axioms (for every type σ; let $1 := \mathrm{S}\ 0$)

$$(\mathrm{INT})\quad \forall x^\sigma, y^\sigma.((\mathrm{e}_\sigma\ x\ y =_{\mathrm{nat}} 0 \vee \mathrm{e}_\sigma\ x\ y =_{\mathrm{nat}} 1) \wedge (\mathrm{e}_\sigma\ x\ y =_{\mathrm{nat}} 0 \leftrightarrow x =_\sigma y))\ \ .$$

One easily verifies that the full type structure and HRO are models of $\mathrm{I}-\mathrm{HA}^\omega$.

Remark 1.1.7 *In both $\mathrm{E}-\mathrm{HA}^\omega$ and $\mathrm{I}-\mathrm{HA}^\omega$, the equality reduces to equality on nat as follows:*

(a) Define for r^σ, s^σ $r =_{\mathrm{e},\sigma} s$ by:

- $r =_{\mathrm{e},\mathrm{nat}} s := r =_{\mathrm{nat}} s$.

- $r =_{\mathrm{e},\sigma\to\tau} s := \forall x^\sigma . r\ x =_{\mathrm{e},\tau} s\ x.$
- $r =_{\mathrm{e},\sigma\times\tau} s := r0 =_{\mathrm{e},\sigma} s0 \wedge r1 =_{\mathrm{e},\tau} s1.$

Then it follows for every type σ

$$\mathrm{E}-\mathrm{HA}^\omega \vdash \forall x^\sigma, y^\sigma (x =_\sigma y \leftrightarrow x =_{\mathrm{e},\sigma} y) \ .$$

(b) $\mathrm{I}-\mathrm{HA}^\omega \vdash \forall x^\sigma, y^\sigma.(x =_\sigma y \leftrightarrow \mathrm{e}_\sigma\ x\ y =_{\mathrm{nat}} 0).$

1.1.3 Embedding of HA in HA^ω

Definition 1.1.8 *(9.1.10)*

(a) For every n-ary primitive recursive function we define a closed term t_f of type $\underbrace{\mathrm{nat}\to\cdots\to\mathrm{nat}}_{n \text{ times}}\to\mathrm{nat}$, s. t. if we replace in the defining axioms for prim. rec. functions in HA^ω $f(x_1,\ldots,x_n)$ by $t_f\ x_1\cdots x_n$, then the resulting formulas are provable in HA^ω. The definition is by recursion on the inductive definition of primitive recursive functions:

- $f(\vec{x}) = 0$: $t_f := \lambda\vec{x}.0$.
- $f(x) = \mathrm{S}(x)$: $t_f := \mathrm{S}$.
- $f(\vec{x}) = x_i$: $t_f := \lambda\vec{x}.x_i$.
- $f(\vec{x}) = g(h_1(\vec{x}),\ldots,h_n(\vec{x}))$: $t_f := \lambda\vec{x}.t_g\ (t_{h_1}\ \vec{x})\cdots(t_{h_m}\ \vec{x})$.
- $f(\vec{x},0) = g(\vec{x})$, $f(\vec{x},\mathrm{S}(y)) = h(\vec{x},y,f(\vec{x},y))$.

$$t_f := \lambda\vec{x}.\mathbf{R}(t_g\ \vec{x})(\lambda y, z.t_h\ \vec{x}\ z\ y) \ .$$

Verification of the axioms in the last case:

$$\begin{array}{rcl} t_f\ \vec{x}\ 0 & \equiv & \mathbf{R}(t_g\ \vec{x})(\lambda y, z.t_h\ \vec{x}\ z\ y)\ 0 = t_g\ \vec{x}. \\ t_f\ \vec{x}\ (\mathrm{S}\ y) & \equiv & \mathbf{R}(t_g\ \vec{x})(\lambda y, z.t_h\ \vec{x}\ z\ y)\ (\mathrm{S}\ y) \\ & = & (\lambda y, z.t_h\ \vec{x}\ z\ y)\ (\mathbf{R}(t_g\ \vec{x})(\lambda y, z.t_h\ \vec{x}\ z\ y)\ y)\ y \\ & = & t_h\ \vec{x}\ y\ (\mathbf{R}(t_g\ \vec{x})(\lambda y, z.t_h\ \vec{x}\ z\ y)\ y) \\ & \equiv & t_h\ \vec{x}\ y\ (t_f\ \vec{x}\ y). \end{array}$$

(b) If t is a term in $\mathcal{L}_{\mathrm{HA}}$, we define a term t^* of type nat in $\mathcal{L}_{\mathrm{HA}^\omega}$ by

- $x^* := x^{\mathrm{nat}}$.
- $0^* := 0$.
- $(\mathrm{S}\ t)^* := \mathrm{S}\ t^*$.
- $f(t_1,\ldots,t_n) := t_f\ (t_1^*)\cdots(t_n^*)$.

(c) If A is a formula in $\mathcal{L}_{\mathrm{HA}}$, we define its translation A^* in $\mathcal{L}_{\mathrm{HA}^\omega}$ by

- $(s = t)^* := s^* =_{\mathrm{nat}} t^*$.
- $\bot^* := \bot$.
- $(A \circ B)^* := A^* \circ B^*$ $(\circ \in \{\wedge, \vee, \to\})$.
- $(Sx.A)^* := Sx^{\mathrm{nat}}.A^*$, where $S \in \{\forall, \exists\}$.

Lemma 1.1.9 *If* $\mathrm{HA} \vdash A$ *then* $\mathrm{HA}^\omega \vdash A^*$.

Proof:
We show more generally: If $\mathrm{HA} \vdash B_1, \ldots, B_n \Rightarrow A$, then $\mathrm{HA}^\omega \vdash B_1^*, \ldots, B_n^* \Rightarrow A^*$ by induction on $\mathrm{HA} \vdash B_1, \ldots, B_n \Rightarrow A$.

- Defining equations of prim. rec. functions: see the Remark about these axioms in Definition 1.1.8 (a).
- Logical rules, arithmetical axioms, equality rules: they coincide in HA and HA^ω, $*$ commutes with the connectives.

Theorem 1.1.10 *(9.1.14)* $\mathrm{I}-\mathrm{HA}^\omega$ *and* $\mathrm{E}-\mathrm{HA}^\omega$ *are conservative extensions of* HA*, i.e. if* $A \in \mathcal{L}_{\mathrm{HA}}$*,* $\mathrm{I}-\mathrm{HA}^\omega \vdash A^*$ *or* $\mathrm{E}-\mathrm{HA}^\omega \vdash A^*$*, then* $\mathrm{HA} \vdash A$.

Proof:
Proof of the assertion for $\mathrm{E}-\mathrm{HA}^\omega$:
For every HA^ω formula A

$$\mathrm{HEO} \models A[x_1 := n_1, \ldots, x_m := n_m]$$

can be expressed as a formula in $\mathcal{L}_{\mathrm{HA}}$ (depending on free variables $n_1, \ldots, n_m$). (Note that this is not a formula depending on a Gödel-number for A).
1. Prove: If

$$\mathrm{E}-\mathrm{HA}^\omega \vdash B_1, \ldots, B_n \Rightarrow A \ ,$$

$\mathrm{FV}(B_1) \cup \cdots \cup \mathrm{FV}(B_n) \cup \mathrm{FV}(A) \subseteq \{x_1, \ldots, x_m\}$, then

$$\begin{aligned}\mathrm{HA} \quad \vdash \quad & \forall n_1, \ldots, n_m. \\ & (\mathrm{HEO} \models B_1[\vec{x} := \vec{n}] \wedge \cdots \wedge \mathrm{HEO} \models B_n[\vec{x} := \vec{n}]) \\ & \to \mathrm{HEO} \models A[\vec{x} := \vec{n}]\end{aligned}$$

This follows by an easy induction on the derivation. (One first observes, that HEO models $\mathrm{E}-\mathrm{HA}^\omega$, and then verifies, that this proof can be formalized in HA.
2. Prove: If t is a term of $\mathcal{L}_{\mathrm{HA}}$, $\mathrm{FV}(t) \subseteq \{x_1, \ldots, x_n\}$, then

$$\mathrm{HA} \vdash \forall n_1, \ldots, n_m.t[\vec{x} := \vec{n}] \simeq (t^*)^{\mathrm{HEO}}[\vec{x}^{\mathrm{nat}} := \vec{n}]$$

(where $[\vec{x}^{\mathrm{nat}} := \vec{n}] \equiv [x_1^{\mathrm{nat}} := n_1, \ldots, x_m^{\mathrm{nat}} := n_m]$.)

- $t = x_i$: $t[\vec{x} := \vec{n}] \equiv n_i$,

$$(t^*)^{\mathrm{HEO}}[\vec{x}^{\mathrm{nat}} := \vec{n}] \equiv (x_i^{\mathrm{nat}})^{\mathrm{HEO}}[\vec{x}^{\mathrm{nat}} := \vec{n}] \equiv n_i \ .$$

- $t = f(t_1, \ldots, t_n)$. $t^* \equiv f^* \, t_1^* \cdots t_n^*$. We show easily

  $$\mathrm{HA} \vdash \forall k_1, \ldots, k_n . f(k_1, \ldots, k_n) \simeq \{\cdots\{\{(f^*)^{\mathrm{HEO}}\}(k_1)\}\cdots\}(k_n) \ .$$

  Now the assertion follows using the IH.

3. Show If A is a formula of $\mathcal{L}_{\mathrm{HA}^\omega}$, $\mathrm{FV}(A) \subseteq \{x_1, \ldots, x_n\}$, then

$$\mathrm{HA} \vdash \forall n_1, \ldots, n_m (A[\vec{x} := \vec{n}] \leftrightarrow \mathrm{HEO} \models A^*[\vec{x}^{\mathrm{nat}} := \vec{n}]) \ .$$

For prime formulas it follows by 2. and for other formulas it follows since "$\models$ commutes with the logical connectives".
4. The assertion follows by 1. and 3.

Assertion for $\mathrm{I} - \mathrm{HA}^\omega$: Similar, using HRO instead of HEO.

1.2 Constructive real numbers (5.1 - 5.4, 6.1)

1.2.1 Introduction of $\mathbb{Z}$ in HA, HA^ω (5.1.1)

Let for $z \in \mathbb{Z}$,

$$z^* := \begin{cases} 0 & \text{if } z = 0 \\ 2z & \text{if } z > 0 \\ 2(-z) + 1 & \text{if } z < 0 \end{cases}$$

This yields a bijection $\lambda z.z^* : \mathbb{Z} \to \mathbb{N}$.
We can replace now formulas in which we have apart from nat a new ground type $\mathbb{Z}$ into formulas of HA (or HA^ω) as follows:

- interpret the ground type $\mathbb{Z}$ as nat.
- replace all functions and relations which originally referred to the ground type $\mathbb{Z}$, by operations, which simulate this operation on the codes. E.g. if the original expression was $z^{\mathbb{Z}} + z'^{\mathbb{Z}}$, replace now $+$ by a primitive recursive function $+'$ such that for all $z, z' \in \mathbb{Z}$, $z^* +' z'^* = (z + z')^*$.
- Verify, that the standard properties of the functions and relations on $\mathbb{Z}$ after the translation can be shown.

Let in the following z (possibly with subscripts, indices) range over $\mathbb{Z}$ (with the above interpretation), and i, j, k, l, n, m range over $\mathbb{N}$.

1.2.2 Introduction of $\mathbb{Q}$ in HA, HA^ω (5.1.1)

In a similar way we can define a bijection $\mathbb{Q} \to \mathbb{N}$.

Exercise 1.2 *Define such a bijection explicitly*

We can interpret $\mathbb{Q}$ in HA as before. Let in the following r, s, t, q range over elements of $\mathbb{Q}$ (possibly with subscripts, accents).

1.2.3 Principal ideas for embedding $\mathbb{R}$ into HA^ω (5.1.2)

There are two approaches:
1) Real numbers as equivalence classes of Cauchy sequences.
Idea: Real numbers are represented by sequences $(q_n)_{n\in\mathrm{nat}}$, s. t.

$$\forall m.\exists N.\forall k, l \geq N |q_k - q_l| < 2^{-m}$$

Explanations for those who do not know Cauchy sequences:
$\sqrt{2}$ should be represented by a sequence of rationals which approximates $\sqrt{2}$ better and better, i.e. we want

$$\forall m.\exists N.\forall k \geq N.|\sqrt{2} - q_k| < 2^{-m}$$

However, if we haven't introduced the reals yet, we don't know what $|\sqrt{2} - q_k| < 2^{-m}$ means. However, if we assume q_k approximates $\sqrt{2}$ "arbitrarily" well, and if $\forall k, l \geq N.|q_k - q_l| < 2^{-n}$, then "$|q_n - \sqrt{2}| \leq 2^{-n}$" holds.

2) Reals as Dedekind cuts. A Dedekind cut is a set $\emptyset \neq A \subseteq \mathbb{Q}$ s. t.

- A is bounded, i.e. $\exists q \in \mathbb{Q}.A < q$.
- A is open, i.e. $\forall q \in A \exists r \in A.q < r$.
- A is downward closed, i.e. $\forall q \in A.\forall q' < q.q' \in A$.

(The above is the classical definition, for the constructive definitions there are several variants, see 5.5.1) Now identify reals with Dedekind cuts.

- Advantage: Suitable for systems of 2^{nd} order logic.
- Disadvantages:
  - $(-x)$ cannot be defined so easily
  - not so concrete

In [TvD] both approaches are studied, we will only consider the first approach.

Not suitable approach Decimal representation is not a good representation. Problem: Not even multiplication by 3 can be computed. Consider the multiplication of 0.33333333333? by 3. If the next digit is 4, then we know the result starts with 1.00000000000. If it is 2 then we know the result starts with 0.99999999999. As long as we get only digits $0.333333\cdots$ we cannot determine therefore, what the first digit of the result is.

1.2.4 Theory in which the following can be formalized

We are going to work in HA^ω extended by the axiom of countable unique choice

$$(\mathrm{AC-NN!}) \quad (\forall n.\exists! m.A(n,m)) \rightarrow \exists \alpha^{\mathrm{nat}\rightarrow\mathrm{nat}}.\forall n.A(n,\alpha(n))$$

To work without it is almost impossible. If we used the axiom of countable choice

$$(\mathrm{AC-NN}) \quad (\forall n.\exists m.A(n,m)) \rightarrow \exists \alpha^{\mathrm{nat}\rightarrow\mathrm{nat}}.\forall n.A(n,\alpha(n))$$

the following would be easier.

Definition 1.2.1 (5.2.1)

(a) Let in the following

- α, β, γ range over elements of type nat$\rightarrow$nat,
- n, m, i, j, k range over nat.
- z range over $\mathbb{Z}$,
- q, r, s, t range over $\mathbb{Q}$
- e write $\alpha(n)$ instead of $\alpha\ n$, etc.

all with possible subscripts and accents.

(b) We denote functions from $\mathbb{N}$ to $\mathbb{Q}$ (as well to other sets) by sequences $(q_n)_{n\in\mathbb{N}}$ or $(q_n)_n$ or even (q_n). If introduced as a new sequence $(q_n)_{n\in\mathbb{N}}$ should be read as $\lambda n.q_n$.

1.2.5 Introduction of $\mathbb{R}$ in HA^ω (5.2.2)

Definition 1.2.2 (2.2)

(a) A fundamental sequence is a sequence $(q_n)_{n\in\mathbb{N}}$ of rationals together with some $\beta^{\mathrm{nat}\rightarrow\mathrm{nat}}$ (called Cauchy-modulus) s. t.

$$\forall k.\forall m, m' \geq \beta(k).(|q_m - q_{m'}| < 2^{-k})$$

(b)

$$(q_n) \approx (r_n) :\Leftrightarrow \forall k.\exists N.\forall m \geq N.|q_m - r_m| < 2^{-k}$$

(c) We will usually only indicate the sequence $(q_n)_{n\in\mathrm{nat}}$ of a fundamental sequence, and refer to the Cauchy-modulus as the "Cauchy-modulus of (q_n)".

(d) The set of Cauchy-reals $\mathbb{R}$ is the set of equivalence classes of fundamental sequences modulo $\approx$. We write $[(q_n)]$ for the equivalence class of (q_n) modulo $\approx$.

Remark 1.2.3 *(a) $\approx$ is an equivalence relation.*

(b) With $(\mathrm{AC}-\mathrm{NN})$ *we could define fundamental sequences by demanding*

$$\forall k.\exists N.\forall m, m' \geq N(|q_m - q_{m'}| < 2^{-k})$$

since $(\mathrm{AC}-\mathrm{NN})$ *provides then immediately from this property a Cauchy-modulus.*

Definition 1.2.4 The set of reals can now be formalized as follows:

- $\forall x^{\mathbb{R}}$ has to be replaced by

  $\forall q^{\mathrm{nat}\to\mathbb{Q}}.\forall \alpha^{\mathrm{nat}\to\mathrm{nat}}.(q_n)$ `is a fundamental sequence with Cauchy-modulus` $\alpha \to \cdots$

  where in the following x has to be replaced by (q_n).
- Equality between elements of $\mathbb{R}$ has to be replaced by $\approx$.
- Operations on reals have to be replaced by operations on the sequences, as will be introduced below.

Let in the following x, y, z range over $\mathbb{R}$ (with indices and accents).

1.2.6 The ordering of the reals (5.2.3 - 5.2.15)

Definition 1.2.5 (a) Let HA^+ be the extension of HA^ω by $(\mathrm{AC}-\mathrm{NN}!)$.

(b) In the following all statements (in so far they are statements of HA^+ can be proved in HA^+.

(c) We write $(q_n) \in x$ for "the real number x is given by the fundamental sequence (q_n)" (with some Cauchy modulus)

(d) We write $\langle (q_n), \alpha \rangle \in x$ for "the real number x is given by the fundamental sequence (q_n) with Cauchy modulus α.

Remark 1.2.6 *To define a function/relation on reals means in the following to define a function/relation on fundamental sequences such that* HA^+ *proves, that it respects the equality $\approx$ on reals.*
From this it follows that the equality axioms with respect to the extension by symbols for the new functions and relations are provable in HA^+ *(i.e. their translation into formulas of $\mathcal{L}_{\mathrm{HA}^\omega}$ are theorems of* HA^+*, where the new relations and functions are translated by their definition).*

Proposition 1.2.7 *(Prop. 5.2.3)*
Assume (q_n), (r_n) are fundamental sequences with Cauchy-modulus α, β, $(q_n) \approx (r_n)$ and define $\gamma(k) := \max\{\alpha(k+2), \beta(k+2)\}$. Then

$$\forall m, m' \geq \gamma(k).|q_m - r_{m'}| < 2^{-k} \quad .$$

Proof:
Assume k. Assume m_0 s. t.

$$\forall l \geq m_0 . |q_l - r_l| < 2^{-(k+2)} \quad .$$

Define $m_1 := \max\{\gamma(k), m_0\}$. Then for all $m, m' \geq \gamma(k)$ we have

$$\begin{array}{rcl} |q_m - r_{m'}| & \leq & |q_m - q_{m_1}| + |q_{m_1} - r_{m_1}| + |r_{m_1} - r_{m'}| \\ & < & 2^{-(k+2)} + 2^{-(k+2)} + 2^{-(k+2)} < 2^{-k} \end{array}$$

Definition 1.2.8 (Def. 5.2.4)
For fundamental sequences (q_n), (r_n) we define

$$(q_n) < (r_n) :\Leftrightarrow \exists k, N. \forall m \geq N. r_m > q_m + 2^{-k} \quad .$$

Proposition 1.2.9 *If* $(q_n) < (r_n)$, $(q_n) \approx (q'_n)$, $(r_n) \approx (r'_n)$, *then* $(q'_n) < (r'_n)$.

Proof:
(Not to be carried out in the lecture).
There exists k, N_1, N_2, N_3 s. t.

$$\begin{array}{lcl} \forall m \geq N_1 & . & r_m > q_m + 2^{-k} \\ \forall m \geq N_2 & . & |q_m - q'_m| < 2^{-(k+2)} \\ \forall m \geq N_3 & . & |r_m - r'_m| < 2^{-(k+2)} \end{array}$$

Then with $N := \max\{N_1, N_2, N_3\}$ it follows

$$\forall m \geq N. r'_m > q'_m + 2^{-(k+2)}$$

Definition 1.2.10 (Def. 5.2.7.; $<$, $\leq$, $\#$ on $\mathbb{R}$)
Let $x, y \in \mathbb{R}$ $(q_n) \in x$, $(r_n) \in y$.
We define

$$\begin{array}{rcl} x < y & :\Leftrightarrow & (q_n) < (r_n) \\ x \leq y & :\Leftrightarrow & \neg(y < x) \\ x \mathrel{\#} y & :\Leftrightarrow & x < y \vee y < x \end{array}$$

$\#$ is called apartness.
$x \mathrel{\#} y$ is pronounced as "x is apart from y".
OBS $x \leq y$ is **not** defined as $x \leq y \vee x = y$.

Definition 1.2.11 (Definition 5.2.6.)
Define

$$\begin{array}{c} * : \mathbb{Q} \rightarrow \mathbb{R} \\ q \mapsto q^* := (q_n) \text{ with Cauch modulus } \lambda n.0 \end{array}$$

We identify q *with* q^*.

Proposition 1.2.12 *(Proposition 5.2.8.)*
Let $x, y \in \mathbb{R}$, $(r_n) \in x$, $(s_n) \in y$.
Then

$$x \# y \leftrightarrow \exists k, N. \forall n \geq N. |q_n - r_n| > 2^{-k} \quad .$$

Proof: Exercise.

Proposition 1.2.13

$$\begin{array}{lrcl} \text{(AP1)} & \neg(x \# y) & \leftrightarrow & x = y \\ \text{(AP2)} & x \# y & \leftrightarrow & y \# x \\ \text{(AP2)} & x \# y & \rightarrow & \forall z (x \# z \vee z \# y) \end{array}$$

Proof:
(Not to be carried out in the lecture)
(AP1):
$x = y \rightarrow \neg(x \# y)$ is clear.
Assume $\neg(x \# y)$ and show $x = y$.
Let $\langle (q_n), \alpha \rangle \in x$, $\langle (r_n), \beta \rangle \in y$.
Assume k. By assumption

$$\begin{array}{c} \forall N. \neg \forall m \geq N |q_m - r_m| > 2^{-(k+2)} \\ \forall m, m' \geq \alpha(k+2). |q_m - q_{m'}| \leq 2^{-(k+2)} \\ \forall m, m' \geq \beta(k+2). |r_m - r_{m'}| \leq 2^{-(k+2)} \end{array} \qquad (*)$$

Let

$$m \geq N := \max\{\alpha(k+2), \beta(k+2)\} \quad .$$

Assume

$$|q_m - r_m| \geq 2^{-k} \quad .$$

Then, for all $m' \geq N$

$$\begin{array}{rcl} |q_{m'} - r_{m'}| & \geq & |q_m - r_m| - |q_m - q_{m'}| - |r_m - r_{m'}| \\ & \geq & 2^{-k} - 2^{-(k+2)} - 2^{-(k+2)} \\ & = & 2^{-(k+1)} \\ & > & 2^{-(k+2)} \end{array}$$

contradicting $(*)$.
Therefore with N as above it follows the assertion.
(AP 2): trivial.
(AP 3): Exercise.

Corollary 1.2.14 *(Corollary 5.2.10).*
Stability of $=$ on $\mathbb{R}$:

$$\forall x, y. (\neg\neg x = y \leftrightarrow x = y)$$

Proof:

$$\neg\neg(x = y) \leftrightarrow \neg\neg\neg x \# y \leftrightarrow \neg x \# y \leftrightarrow x = y \quad .$$

Proposition 1.2.15 *(Proposition 5.2.11.)*
Assume $x, y, z \in \mathbb{R}$.

(a) $(x \leq y \wedge y \leq x) \rightarrow x = y$.

(b) $(x < y \wedge y < z) \rightarrow x < z$.

(c) $x < y \rightarrow (x < z \vee z < y)$.

(d) $(x < y \wedge y \leq z) \rightarrow x < z$.

(e) $(x \leq y \wedge y < z) \rightarrow x < z$.

(f) $(x \leq y \wedge y \leq z) \rightarrow x \leq z$.

(g) $x \leq y \rightarrow \neg\neg(x < y \vee x = y)$.

(h) $\neg\neg(x \leq y \vee y \leq x)$.

(i) $\neg\neg(x < y \vee x = y \vee y < x)$.

Proof:
(Not to be carried out in the lecture).
(a):

$$\begin{array}{rcl} x \leq y \wedge y \leq x & \rightarrow & \neg(y < x) \wedge \neg(x < y) \\ & \rightarrow & \neg(y < x \vee x < y) \\ & \leftrightarrow & \neg x \mathrel{\#} y \\ & \leftrightarrow & x = y \end{array}$$

(b): Easy.
(c):
By $x < y$ it follows

$$x \mathrel{\#} y$$

by (AP 3)

$$x \mathrel{\#} z \vee z \mathrel{\#} y$$

therefore

$$(x < z \vee z < x) \vee (z < y \vee y < z)$$

By (b) and $x < y$ therefore

$$x < z \vee z < y \vee z < y \vee x < z \ ,$$

the assertion.
(d) Assume $(x < y) \wedge y \leq z$. By (c)

$$x < z \vee z < y \ .$$

By $y \leq z$ we have $\neg(z < y)$, therefore

$$x < z \ .$$

(e) Similarly.
(f) Assume $x \le y$, $y \le z$.
Assume

$$z < x \ .$$

Then

$$\begin{array}{ll} z < y & \text{by } x \le y, \text{(d)} \\ z < z & \text{by } y \le z, \text{(d)} \\ z \# z & \\ z \ne z & \text{contradiction.} \end{array}$$

(g) - (i): Exercise.

Weak counterexample for $x = 0 \lor x \ne 0$.
If $A(y)$ is a decidable statement s. t. whether $\forall y.A(y)$ holds is not known, let α be defined by

$$q_n := \begin{cases} 2^{-m} & \text{if } m \le n, \neg A(m), \forall k < m.A(k) \\ 2^{-n} & \text{if } \forall k \le n.A(k) \end{cases}$$

Then q_n is a fundamental sequence with Cauchy-modulus $\lambda n.n + 1$, and with x given by (q_n),

$$\begin{array}{rcl} \forall n.A(n) & \to & x = 0 \\ \exists n.\neg A(n) & \to & \exists n.x = 2^{-n} \text{ therefore } x \ne 0 \end{array}$$

therefore $x = 0 \lor x \ne 0$ is not constructively valid.
Taking $A(n) := \neg \mathrm{T}(m, m, n)$ we get $A(n) \leftrightarrow \{m\}(m) \uparrow$, therefore from

$$\forall x.(x = 0 \lor x \ne 0)$$

we could conclude

$$\forall m(\{m\}(m) \downarrow \lor \{m\}(m) \uparrow)$$

which is not Kleene-realizable, therefore (since all theorems of HA^+ are realizable) not provable in HA^+.

Definition 1.2.16 (Definition 5.2.13) For $x, y \in \mathbb{R}$ we define $x+y, x-y, |x| \in \mathbb{R}$ as follows:
Let $(q_n) \in x$, $(r_n) \in y$.
Then one verifies that for some appropriate Cauchy-moduli

$$\begin{array}{c} (q_n + r_n) \ , \\ (q_n - r_n) \ , \\ (|q_n|) \ , \end{array}$$

are fundamental sequences.
The definition respects $\approx$. Let now $x + y$, $x - y$, $|x|$ be given by $(q_n + r_n)$, $(q_n - r_n)$, $(|q_n|)$.

Remark 1.2.17 *For* $r \in \mathbb{Q}$

$$r > 0 \rightarrow x < x + r$$

Proof: Easy.

Proposition 1.2.18 *(Proposition 5.2.14)*

$$x \leq y \leftrightarrow \forall k. x < y + 2^{-k} \quad .$$

Proof:
(Not to be carried out in the lecture).
"$\rightarrow$": $x \leq y < y + 2^{-k}$.
"$\leftarrow$" Assume $\forall k. x < y + 2^{-k}$, $y < x$. From the definition of $<$ it follows immediately for some m

$$y + 2^m < x$$

Therefore

$$y + 2^m < x < y + 2^m$$

a contradiction. $x \leq y$.

Proposition 1.2.19 *Assume* $x, y, z \in \mathbb{R}$, $(q_n) \in x$.

(a) $\forall m \geq n(|q_n - q_m| \leq 2^{-k} \rightarrow |x - q_n| \leq 2^{-k}$

(b) $|x - y| \leq r \leftrightarrow x - r \leq y \leq x + r$

(c) $(|x - y| \leq r \wedge |y - z| \leq r') \rightarrow |x - z| \leq r + r'$

Proof: [TvD] or exercise.

1.2.7 Real valued functions (5.3.3)

Definition 1.2.20 (a) $\sigma^k := \underbrace{\sigma \times \cdots \times \sigma}_{k\text{times}}$.

(b) We write

- $(a_1, \ldots, a_n)$ for $\langle a_1, \langle a_2, \ldots \langle a_{n-1}, a_n \rangle \rangle \rangle$,
- $f(a_1, \ldots, a_n)$ for $f((a_1, \ldots, a_n))$,
- $f(a)$ for $f\ a$.

Lemma 1.2.21 *(a)* $\forall x((\forall k(|x| < 2^{-k}) \rightarrow x = 0)$.

(b) $\forall x. \forall k. \exists q. |x - q| < 2^{-k}$.

(c) $\forall x. \exists k. |x| < k$.

(d) $\forall x, y, z(|x - z| \leq |x - y| + |y - z|)$.

(e) $\forall x, x', y, y'((x < y \wedge x' < y') \rightarrow x + x' < y + y')$.

Proof: Easy exercise.

Definition 1.2.22 A function $f : \mathbb{R}^m \rightarrow \mathbb{R}$ is in the following given by functions

$$\begin{array}{rcl} f_0 & : & (\mathrm{nat} \rightarrow \mathbb{Q})^m \rightarrow (\mathrm{nat} \rightarrow \mathbb{Q}), \\ \alpha & : & (\mathrm{nat} \rightarrow \mathbb{Q})^m \rightarrow (\mathrm{nat} \rightarrow \mathrm{nat})^m \rightarrow (\mathrm{nat} \rightarrow \mathrm{nat}) \end{array}$$

where α is called the Cauchy-modulus-function, such that

- If (q_n^i) are fundamental sequences with Cauchy-modulus β^k, then $f_0((q_n^1), \ldots, (q_n^m))$ is a fundamental sequence with Cauchy-modulus
$$\alpha_0((q_n^1), \ldots, (q_n^m), \beta_0, \ldots, \beta_m) \ .$$
- If $(q_n^i) \approx (r_n^i)$, then
$$f_0((q_n^1), \ldots, (q_n^m)) \approx f_0((r_n^1), \ldots, (r_n^m)) \ .$$

Remark 1.2.23 *(a) What we really would like to have is that a function*

$$f : \mathbb{R}^m \rightarrow \mathbb{R}$$

is a function, which takes as arguments fundamental sequence (q_n^i), Cauchy-moduli α_i and proofs that q_n^i are fundamental sequences with moduli α^i and maps this to a fundamental sequence (q_n'), a Cauchy-modulus β and a proof that (q_n') is a fundamental sequence with modulus β together with a proof that this function respects $\approx$. However this cannot be expressed in HA^+ *(but for instance in dependent type theory), neither could we make use there of the proof that q_n^i are fundamental sequences except for the proof of q_n' being a fundamental sequence, and this dependency we have.*

(b) It might be that f_0 has to depend on the Cauchy-moduli α_i for (q_n^i) as well as well.

Definition 1.2.24 (Definition 5.3.1)
Assume $\vec{x}, \vec{y} \in \mathbb{R}^n$ or $\vec{x}, \vec{y} \in \mathbb{Q}^n$, $r \in \mathbb{Q}$. Let

$$\vec{0} := (0, \ldots, 0) \ .$$

(a)
$$|\vec{x} - \vec{y}| \leq r := |x_1 - y_1| \leq r \wedge \cdots \wedge |x_n - y_n| \leq r \ .$$

(b)
$$|\vec{x}| \leq r := |\vec{x} - \vec{0}| \leq r$$

Definition 1.2.25 (Definition 5.3.2)
Let $f : X^n \rightarrow Y$, where $X, Y \in \{\mathbb{Q}, \mathbb{R}\}$.

(a) f is uniform continuous with modulus α iff

$$\forall k.\forall \vec{x}, \vec{y} \in X^n(|\vec{x} - \vec{y}| < 2^{-\alpha(k)} \rightarrow |f(\vec{x}) - f(\vec{y})| < 2^{-k})$$

(b) f is locally continuous with modulus α iff

$$\begin{array}{ll}\forall k, m.\forall \vec{x}, \vec{y} \in X^n & ((|\vec{x}| < m \land |\vec{y}| < m) \\ & \rightarrow |\vec{x} - \vec{y}| < 2^{-\alpha(k,m)} \\ & \rightarrow |f(\vec{x}) - f(\vec{y})| < 2^{-k})\end{array}$$

Remark 1.2.26 *Under assumption of* (AC − NN) *we can omit the Cauchy-modulus and replace the conditions above by demanding for all k (for all k, m) the existence of an i such that for all $\vec{x}, \vec{y}$ the matrix of the above formulas with $\alpha(k)$ ($\alpha(k, m)$) replaced by i holds.*

Definition 1.2.27 (Definition 5.3.4)
Let $f : \mathbb{Q}^n \rightarrow \mathbb{Q}$ be locally continuous. Then the canonical extension $f^* : \mathbb{R}^n \rightarrow \mathbb{R}$ is given by:
If $(q^k_n)_i \in x_i$ then

$$f^*(\vec{x}) := (f(q^1_n, \dots, q^n_n))_{n\in\text{nat}} \ ,$$

where the corresponding Cauchy-modulus-function and the proof that f^* respects equality are left as an exercise.

Exercise 1.3 *(a) Show $\forall x \in \mathbb{R} \exists n.|x| < n$.*

(b) Determine the Cauchy-modulus-function and the proof that f^* respects equality in Definition 1.2.27.

Remark 1.2.28 *For an arbitrary (not in general locally continuous) function $f : \mathbb{Q}^n \rightarrow \mathbb{Q}$ we cannot in general determine an extension f^*. (Why? Do non-locally continuous functions exists?)*

Lemma 1.2.29 *(Theorem 5.3.5) For every locally-continuous function $f : \mathbb{Q}^n \rightarrow \mathbb{Q}$ then f^* is the unique locally continuous function $f^* : \mathbb{R}^n \rightarrow \mathbb{R}$ extending f (i.e. such that*

$$\forall \vec{q} \in \mathbb{Q}^n.f^*(\vec{q}) = f(\vec{q})\ ) \ .$$

Proof:
For notational simplicity assume f is uniformly continuous and $n = 1$.
Let f be uniformly continuous with modulus α. We show f^* is uniformly continuous with modulus $\lambda k.\alpha(k + 1)$.
Assume k, x, y, $|x - y| < 2^{-\alpha(k+1)}$, $(q_n) \in x$, $(r_n) \in y$. There exists N s. t.

$$\forall k \geq N.|q_k - r_k| < 2^{-\alpha(k+1)}$$

Then

$$\forall k \geq N.|f(q_k) - f(r_k)| < 2^{-(k+1)}$$

and therefore

$$\forall k \geq N. |f(q_k) - f(r_k)| + 2^{-(k+2)} < 2^{-k}$$
$$|f(x) - f(y)| < 2^{-k} \ .$$

Uniqueness of f. Let f^o be another locally continuous extension of f. We assume for simplicity f^o is uniformly continuous with modulus β. Let $x \in \mathbb{R}$, $k \in \mathbb{N}$.
By the continuity of f^* and f^o there exists l such that for y, $|y-x| < 2^{-l}$, we have

$$\begin{array}{rcl} |f^*(y) - f^*(x)| & < & 2^{-(k+1)} \ , \\ |f^o(y) - f^o(x)| & < & 2^{-(k+1)} \ . \end{array}$$

Let $q \in \mathbb{Q}$ such that $|x - q| < 2^{-l}$. Then $f^*(q) = f(q) = f^o(q)$ and therefore

$$\begin{array}{rcl} |f^*(x) - f^o(x)| & \leq & |f^*(x) - f^*(q)| + |f^o(q) - f^o(x)| \\ & < & 2^{-(k+1)} + 2^{-(k+1)} \\ & = & 2^{-k} \end{array}$$

Since k is arbitrary it follows $f^*(x) = f^o(x)$.

Lemma 1.2.30 *(Corollary 3.5, Prop 3.6, Prop 3.8)*

(a) $+, -, \cdot, |\cdot|$ *are the unique locally continuous extensions of the corresponding functions on* $\mathbb{Q}$ *to* $\mathbb{R}$.

(b) $\lambda x, y. \max\{x, y\}$, $\lambda x, y. \min\{x, y\}$ *can be extended from* $\mathbb{Q}$ *to* $\mathbb{R}$.

(c) Let $\varphi(x_1, \ldots, x_n), \psi(x_1, \ldots, x_n)$ *be terms build from Variables* x_i *using some* $q \in \mathbb{Q}$ *and* $+, -, \cdot, |\cdot|$, $\lambda x, y. \max\{x, y\}$, $\lambda x, y. \min\{x, y\}$. *Then from*

$$\forall q_1, \ldots, q_n \in \mathbb{Q}. \varphi(q_1, \ldots, q_n) = \psi(q_1, \ldots, q_n)$$

it follows

$$\forall x_1, \ldots, x_n \in \mathbb{Q}. \varphi(x_1, \ldots, x_n) = \psi(x_1, \ldots, x_n)$$

The same holds with $=$ *replaced by* $\leq$.

Proof:
(a), (b). Immediate.
(c) It follows easily that $\lambda \vec{x}. \varphi(\vec{x})$, $\lambda \vec{x}. \psi(\vec{x})$ are locally continuous and the unique extensions of the corresponding functions on $\mathbb{Q}$. Therefore the assertion for $=$ follows.
For $\leq$ we show below in Lemma 1.2.31 (b)

$$x \leq y \leftrightarrow \max\{x, y\} = y$$

and then the assertion follows by replacing the above equation by

$$\max\{\varphi(\vec{x}), \psi(\vec{x})\} = \psi(\vec{x}) \ .$$

Lemma 1.2.31 *(Proposition 5.3.7 and 5.3.9)*

(a) $(y < x \vee y = x) \rightarrow$
$(\max\{x, y\} = x \wedge \min\{x, y\} = x)$.

(b) $x \leq y \leftrightarrow \max\{x, y\} = y \leftrightarrow \min\{x, y\} = x$

(c) $|x - y| = \max\{x, y\} - \min\{x, y\}$.

(d) $x < y \leftrightarrow x + z < y + z$.

(e) $x \leq y \leftrightarrow x + z \leq y + z$.

(f) $|x - y| \leq z \leftrightarrow x - z \leq y \leq x + z$.

(g) $\min\{x, y\} \leq x \leq \max\{x, y\}$.

(h) $\min\{x, y\} \leq y \leq \max\{x, y\}$.

(i) $|x - y| \geq ||x| - |y||$.

(j) $x \# y \rightarrow (x + z) \# (y + z)$.

(k) $(x + y) \# 0 \rightarrow (x \# 0 \vee y \# 0)$.

(l) $xy \# 0 \rightarrow (x \# 0 \wedge y \# 0)$.

Proof: [TvD]. But everybody should be able to do this by hand.

1.2.8 Completeness of $\mathbb{R}$ (5.4)

Definition 1.2.32 (a) A sequence of reals is a function

$$q : \text{nat} \rightarrow \text{nat} \rightarrow \text{nat}$$

written as

$$(q_{n,m})_{n,m \in \text{nat}} \ ,$$

together with a modulus of it

$$\alpha : \text{nat} \rightarrow \text{nat} \rightarrow \text{nat}$$

such that for $n \in \text{nat}$

$$(q_{n,m})_{m \in \text{nat}} \text{ is a fundamental sequence with modulus } \alpha(n) \ .$$

If x_n is the real given by

$$(q_{n,m})_{n,m \in \text{nat}} \ ,$$

then we denote such a sequence by

$$(x_n)_{n \in \text{nat}} \ .$$

(b) A sequence of reals $(x_n)_{n\in\mathrm{nat}}$ converges against x iff there exists a modulus of convergence $\beta : \mathrm{nat}\to\mathrm{nat}$ such that

$$\forall n.\forall b \geq \beta(n)(|x_n - x| < 2^{-n}) \ .$$

(c) A sequence of reals $(x_n)_{n\in\mathrm{nat}}$ is a Cauchy sequence with Cauchy-modulus $\beta : \mathrm{nat}\to\mathrm{nat}$ iff

$$\forall k.\forall l, m \geq \beta(k)|x_l - x_m| < 2^{-k} \ .$$

Theorem 1.2.33 *(Theorem 5.4.2) $\mathbb{R}$ is Cauchy-complete, i.e. if (x_n) is a Cauchy sequence, then there exists an $x \in \mathbb{R}$ such that x_n converges against x.*

Proof:
Let α be a modulus, β be a Cauchy-modulus for (x_n), (x_n) given by $(q_{n,m})_{n,m}$.
Let γ defined by

$$\gamma(0) := \beta(0), \qquad \gamma(n+1) := \max\{\beta(n+1), \gamma(n)\} + 1$$

γ is as well a Cauchy-modulus for (x_n) such that $\forall n, m(n < m \to \gamma(n) < \gamma(m))$.
Let

$$r_n := q_{\alpha(\gamma(n),n),\gamma(n)} \ .$$

Then

$$|x_{\gamma(n)} - r_n| \leq 2^{-n}$$

and therefore for $m, m' \geq n$

$$\begin{array}{rcl} |r_m - r_{m'}| & \leq & |r_m - x_{\gamma(m)}| + |x_{\gamma(m)} - x_{\gamma(m')}| + |x_{\gamma(m')} - r_{m'}| \\ & \leq & 2^{-n} + 2^{-n} + 2^{-n} < 2^{-n+2} \end{array}$$

Therefore (r_n) is a fundamental sequence with modulus $\lambda n.n + 2$. Let x be given by (r_n). (x_n) converges against x with modulus $\lambda n.\gamma(n+3)$:
Let $m \geq \gamma(n+3)$. Then

$$\begin{array}{rcl} |x - x_m| & \leq & |x - r_{n+3}| + |r_{n+3} - x_{\gamma(n+3)}| + |x_{\gamma(n+3)} - x_m| \\ & \leq & 2^{-(n+1)} + 2^{-(n+3)} + 2^{-(n+3)} \\ & < & 2^{-n} \end{array}$$

1.2.9 Intermediate value and existence of minimum/maximum theorems (6.1)

6.1. in [TvD] shows which variant of the intermediate value theorem and the theorem of the existence of minimum/maximum is constructively valid and which not (e.g. the theorem of the existence of minimum/maximum isn't but the existence of a supremum/infimum for totally bounded functions is). This section is very easy to read and therefore not worked out here.

Chapter 2

λ-Calculus and Combinatory Logic ([HS86])

Numbers will refer from now on to the book of Hindley and Seldin.

2.1 λ-calculus (1)

2.1.1 Introduction

We have already dealt considered λ-terms already in our investigations of HA^ω. $\lambda x.t$ was there the function, which applied to a term s yields $t[x := s]$. In the case of HA^ω we had obtained typed λ-terms: $\lambda x.t$ was of type $\sigma \to \tau$, and could only be applied to a term of type σ. Forming terms like this yields the *typed λ-calculus*.
This excludes however the application of a function to itself. If we consider the identity function $\lambda x.x$, then this function could be applied to any object, as well the function $\lambda x.x$. So to form $(\lambda x.x)(\lambda x.x)$ makes sense (this can be typed, but both occurrences of $\lambda x.x$ get different types), therefore as well $(\lambda x.xx)(\lambda x.x)$ which should yield the above result. But the sub-term xx cannot be typed: x must have type $\sigma \to \rho$ and at the same time σ for some σ, ρ.
In programming it is interesting to apply a program to itself. However we cannot expect that a program behaves very well. We will see is that the typed λ-calculus is strongly normalizing, i.e. when we reduce a term in a way as above described (applying such reductions to sub-terms as well), we always obtain an irreducible term, independent of the choice of reductions. For the untyped λ-calculus this is not the case: Take the term $(\lambda x.xx)(\lambda x.xx)$. It reduces to itself, and has therefore an infinite reduction sequence.
The relationship between typed λ-calculus and untyped λ-calculus is somehow similar to that of primitive recursive functions and partial recursive

functions. Primitive recursive functions always terminate, but there are computable functions which are not primitive recursive. Partial recursive functions might terminate or might not, but they contain all recursive function (we will see that all partial recursive functions can be represented in the untyped λ-calculus; the class of definable functions in the typed λ-calculus is very small: it's the least class of numeric functions containing projections, the constant functions, the signumbar function $(0 \mapsto 1), (\mathrm{S}(n) \mapsto 0)$ (from which the signum function $0 \mapsto 0$, $\mathrm{S}(n) \mapsto 1$ can be defined), addition, multiplication and is closed under composition).
We can extend the class of primitive recursive functions, but as long as we do this in a recursively enumerable way (i.e. such that there is a recursively enumerable subset A of $\mathbb{N} \times \mathbb{N}$ such that the resulting class of functions is $\{\{e\}^n \mid \langle e, n\rangle \in A\}$) we will not exhaust the class of recursive functions.
Similarly, by extending a type system in such a way that, whether a term can by typed, is decidable and such that all terms which can be typed are normalizing, we will never obtain all normalizing terms. If we have a type system such that the set of terms which can be typed and map certain encodings of the natural numbers (so called Church-numerals) to such encodings is recursive enumerable, then there will always be a term mapping Church-numerals to Church-numerals which cannot be typed.
We will start with the untyped λ-calculus and will later look at the typed one.
Further, we will look as well at combinators, which allowed us in the case of HA^ω to define all λ-terms. Essentially the theory of combinators and the theory of λ-terms are equivalent, but there are fine distinctions to be made, which will be explored.

Definition 2.1.1 Some conventions

(a) Let $m_1, \dots, m_n \mapsto \langle m_1, \dots, m_n\rangle$ some primitive recursive coding of sequence such that standard properties hold and such that $0 = \langle\rangle$.

(b) If $m = \langle m_1, \dots, m_k\rangle$, $(m)_i := m_i$ and the *length of the sequence* m $(\mathrm{seqlength}(m))$ is defined as k.

(c) If $m = \langle m_1, \dots, m_k\rangle$, $n = \langle n_1, \dots, n_l\rangle$, $m * n := \langle m_1, \dots, m_k, n_1, \dots, n_l\rangle$.

2.1.2 Definition of λ-terms (1A, 1B)

Definition 2.1.2 (Definition 1.1).

(a) Assume some infinite sequence of distinct symbols, called *variables*, and a (finite, infinite or empty) sequence of distinct symbols called *constants*.

(If the sequence of constants is empty, the system is called *pure*, otherwise *applied*).

The set of expressions called *λ-terms* is inductively defined as follows:

- All variables and constants are λ-terms (called *atoms*).

- If M, N are λ-terms, so is (MN) (called an *application*).
- If M is a λ-term, x a variable, then $(\lambda x.M)$ is a λ-term (called an *abstraction*).

(b) In this chapter capital roman letters will denote λ-terms. In this part x, y, z, u, v, w will denote variables.

(c) Parenthesis will be omitted with the following conventions

- Application is associative to the left ($MNPQ$ denotes $(((MN)P)Q)$).
- The scope of $\lambda x.$ is maximal (i.e. $\lambda x.PQ$ denotes $\lambda x.(PQ)$,

(d) $\lambda x_1, \ldots, x_n.M := \lambda x_1.\lambda x_2.\ldots\lambda x_n.M.$

(e) We write $M \equiv N$ for syntactic identity of the terms M, N.

Definition 2.1.3 (Definition 1.5)

(a) The *length* of a term M (written as *lgh*(M)) is defined as:

- $\mathrm{lgh}(a) := 1$ if a is an atom.
- $\mathrm{lgh}(MN) := \mathrm{lgh}(M) + \mathrm{lgh}(N)$.
- $\mathrm{lgh}(\lambda x.M) := 1 + \mathrm{lgh}(M)$.

Induction on M means in the following (**OBS!**) induction on $\mathrm{lgh}(M)$.

Definition 2.1.4 We define the sets of free variables $\mathrm{FV}(N)$ and of bound variables $\mathrm{BV}(N)$ of a term N as follows:

- $\mathrm{FV}(a) := \mathrm{BV}(a) := \emptyset$ if a is a constant.
- $\mathrm{FV}(x) := \{x\}$, $\mathrm{BV}(x) := \emptyset$.
- $\mathrm{FV}(MN) := \mathrm{FV}(M) \cup \mathrm{FV}(N)$, $\mathrm{BV}(MN) := \mathrm{BV}(M) \cup \mathrm{BV}(N)$.
- $\mathrm{FV}(\lambda x.M) := \mathrm{FV}(M) \setminus \{x\}$, $\mathrm{BV}(\lambda x.M) := \mathrm{BV}(M) \cup \{x\}$.

$\mathrm{Var}(M) := \mathrm{FV}(M) \cup \mathrm{BV}(M)$.

2.1.3 Substitution (In 1B)

Definition 2.1.5 (1.11) For M, N, x we define $M[x := N]$, the *result of substituting N for x in M* by induction on the M in such a way that, if N is a variable, then $\mathrm{lgh}(M[x := N]) = \mathrm{lgh}[M]$, as follows:

- $x[x := N] := N$.
- $a[x := N] := a$ (a an atom, $a \not\equiv x$).
- $(PQ)[x := N] := (P[x := N])(Q[x := N])$.

- $(\lambda x.P)[x := N] := \lambda x.P$.
- $(\lambda y.P)[x := N] := \lambda y.(P[x := N])$, if $x \not\equiv y$, $y \notin \mathrm{FV}(N) \vee x \notin \mathrm{FV}(P)$.
- $(\lambda y.P)[x := N] := \lambda z.((P[y := z])[x := N])$, if $x \not\equiv y$, $y \in \mathrm{FV}(N) \wedge x \in \mathrm{FV}(P)$, z the first variable s. t. $z \notin \mathrm{FV}(N) \cup \mathrm{FV}(P)$.

Remark: The last clause in Definition 2.1.5 renames the bound variable y first, such that there is no variable clash and therefore N is now substitutable for x in the new term, and then carries out the substitution.

Lemma 2.1.6 *(1.14)*

(a) $M[x := x] \equiv M$.

(b) $x \notin \mathrm{FV}(M) \Rightarrow M[x := N] \equiv M$.

(c) $x \in \mathrm{FV}(M) \Rightarrow \mathrm{FV}(M[x := N]) = \mathrm{FV}(N) \cup (\mathrm{FV}(M) \setminus \{x\})$.

(d) $\mathrm{lgh}(M[x := y]) = \mathrm{lgh}(M)$.

Proof: Easy.

Lemma 2.1.7 *(1.15) Let x, y, v be distinct variables,* $\mathrm{BV}(M) \cap \mathrm{FV}(vPQ) = \emptyset$.

(a) $v \notin \mathrm{FV}(M) \Rightarrow M[x := v][v := P] \equiv M[x := P]$.

(b) $v \notin \mathrm{FV}(M) \Rightarrow M[x := v][v := x] \equiv M$.

(c) $y \notin \mathrm{FV}(P) \Rightarrow M[y := Q][x := P] \equiv M[x := P][y := Q[x := P]]$.

(d) $M[x := Q][x := P] \equiv M[x := (Q[x := P])]$.

Proof:
(a), (c): Induction on M. (b) follows from (a) and Lemma 2.1.6 (a), (d) from (c) and 2.1.6 (b).

Definition 2.1.8 New version of (1.16)
The relation $M \equiv_\alpha N$, M *is α-equivalent to* N is defined as follows:

- If a is an atom, $a \equiv_\alpha a$.
- If $M \equiv_\alpha M'$, $N \equiv_\alpha N'$, then $MN \equiv_\alpha M'N'$.
- If $v \notin \mathrm{Var}(M) \cup \mathrm{Var}(N)$, $M[x := v] \equiv_\alpha N[y := v]$, then $\lambda x.M \equiv_\alpha \lambda y.N$.

Lemma 2.1.9 *(a) $\equiv_\alpha$ is reflexive and symmetric.*

(b) If $M \equiv_\alpha N$, then $M[x := v] \equiv_\alpha N[x := v]$.

(c) $\equiv_\alpha$ is an equivalence relation.

(d) If $P \equiv_\alpha Q$ then $\mathrm{FV}(P) = \mathrm{FV}(Q)$.

(e) *For every term P and $x_1, \dots, x_n$ there exists $P' \equiv_\alpha P$ such that $x_1, \dots, x_n \notin$* $\mathrm{BV}(P')$.

Proof:
(a): trivial. (b): Easy induction on M, using Lemma 2.1.7 (a) and (c).
(c): Induction on M, using (b) and Lemma 2.1.7 (a) in the case $M \equiv \lambda x.P$.
(d) Trivial.
(e) Easy.

Lemma 2.1.10 *(1.18, 1.19) Let x, y, v be distinct variables.*

(a) $v \notin \mathrm{FV}(M) \Rightarrow M[x := v][v := P] \equiv_\alpha M[x := P]$.

(b) $v \notin \mathrm{FV}(M) \Rightarrow M[x := v][v := x] \equiv_\alpha M$.

(c) $y \notin \mathrm{FV}(P) \Rightarrow M[y := Q][x := P] \equiv_\alpha M[x := P][y := Q[x := P]]$.

(d) $M[x := Q][x := P] \equiv_\alpha M[x := (Q[x := P])]$.

(e) $M \equiv_\alpha M'$, $N \equiv_\alpha N' \Rightarrow M[x := N] \equiv_\alpha M'[x := N']$.

Proof: as before, in (e) Induction on M.

2.1.4 β-reduction (1C)

Definition 2.1.11 (a) Inductive definition of the relation $M \longrightarrow_\beta N$:

- $(\lambda x.M)N \longrightarrow_\beta M[x := N]$.
- If $M \longrightarrow_\beta M'$ then
  - $MN \longrightarrow_\beta M'N$,
  - $NM \longrightarrow_\beta NM'$,
  - $\lambda x.M \longrightarrow_\beta \lambda x.M'$.

(b) $\longrightarrow_\beta^*$ is the transitive reflexive closure of $\longrightarrow_\beta \cup \equiv_\alpha$.

(c) A term N is in β-normal form, if there is no term N' s. t. $N \longrightarrow_\beta N'$.

(d) If $P \longrightarrow_\beta^* Q$, Q is a term in β-normal form, then Q is called a *β-normal form of P*.
P normalizes, if it has a β-normal form.

(e) We omit the subscript β in the above definitions, if there is no confusion.

Example 2.1.12 (a) $(\lambda x.(\lambda y.yx)z)v$ has β-normal form zv.

(b) Let $L := (\lambda x.xxy)(\lambda x.xxy)$

$$L \longrightarrow_\beta Ly \longrightarrow_\beta Lyy \longrightarrow_\beta \cdots \quad .$$

L has no β-normal form.

(c) Let $P := (\lambda u.v)L$, L as in (b). There are (among others) two reduction sequences:

- $P \longrightarrow_\beta v$.
- $P \longrightarrow_\beta (\lambda u.v)(Ly) \longrightarrow_\beta (\lambda u.v)(Lyy) \longrightarrow_\beta \cdots$.

P has normal form v, but also an infinite reduction sequence.

Lemma 2.1.13 *(1.27) If $P \equiv_\alpha P'$, $Q \equiv_\alpha Q'$, $P \longrightarrow^*_\beta Q$, then $P' \longrightarrow^*_\beta Q'$.*

Proof:
Trivial: $P' \equiv_\alpha P \longrightarrow^*_\beta Q \equiv_\alpha Q'$.

Lemma 2.1.14 *(1.28; Substitution lemma for β-reduction) Assume $P \longrightarrow^*_\beta Q$.*

(a) $\mathrm{FV}(Q) \subseteq \mathrm{FV}(P)$.

(b) $M[x := P] \longrightarrow^*_\beta M[x := Q]$.

(c) $P[x := N] \longrightarrow^*_\beta Q[x := N]$.

Proof: It suffices to consider the case $P \longrightarrow_\beta Q$ (the case $P \equiv_\alpha Q$ follows by Lemma 2.1.10 (e), 2.1.9 (d)).
(a) By Lemma 2.1.6 (b), (c) $\mathrm{FV}(M[x := N]) \subseteq \mathrm{FV}((\lambda x.M)N)$.
(b) By Lemma 2.1.10 (e) we can assume $\mathrm{BV}(M) \cap (\mathrm{Var}(P) \cup \mathrm{Var}(Q)) = \emptyset$. Now induction on M.
(c) Again assume $\mathrm{BV}(P) \cap \mathrm{FV}(N) = \emptyset$. The only difficult case is where $P \equiv (\lambda y.H)J$ and $Q \equiv H[y := J]$.
Then

$$\begin{array}{rcll} P[x := N] & \equiv & ((\lambda y.H)J)[x := N] & \\ & \equiv & ((\lambda y.(H[x := N]))(J[x := N]) & \\ & \longrightarrow_\beta & H[x := N][y := J[x := N]] & \\ & \equiv & H[y := J][x := N] & \text{by Lemma 2.1.7 (c), 2.1.10 (c)} \\ & \equiv & Q[x := N] & \end{array}$$

Theorem 2.1.15 *(1.29, Church-Rosser theorem for β-reduction)*
If $P \longrightarrow_\beta M$, $P \longrightarrow_\beta N$, then there exists T such that $M \longrightarrow^*_\beta T$, $N \longrightarrow^*_\beta T$.

Proof:
We follow Takahashi, [Tak95]

Definition 2.1.16 (a) ([Tak95] 1.1)

The parallel β-reduction, denoted by $\Longrightarrow_\beta$ is defined inductively defined as

- $a \Longrightarrow_\beta a$, if a is an atom.
- If $v \notin \mathrm{Var}(M) \cup \mathrm{Var}(M')$, $M[x := v] \Longrightarrow_\beta M'[y := v]$, then $\lambda x.M \Longrightarrow_\beta \lambda y.M'$.

- If $M \Longrightarrow_\beta M'$, $N \Longrightarrow_\beta N'$, then $MN \Longrightarrow_\beta M'N'$.
- If $M[x := u] \Longrightarrow_\beta M'$, $N \Longrightarrow_\beta N'$, $u \notin \mathrm{Var}(M) \cup \mathrm{Var}(N')$, then $(\lambda x.M)N \Longrightarrow_\beta M'[u := N']$.

(b) For λ-terms M define M^* by induction on M as follows:

- $a^* := a$ (a an atom).
- $(\lambda x.M)^* := \lambda x.M^*$.
- $(M_1 M_2)^* := (M_1^*)(M_2^*)$, if M_1 not of the form $\lambda x.M'$.
- $((\lambda x.M_1)M_2)^* := M_1^*[x := M_2^*]$.

Lemma 2.1.17 *(a) $M \Longrightarrow_\beta M$.*

(b) $M \Longrightarrow_\beta M'$, $v \notin \mathrm{Var}(M) \cup \mathrm{Var}(M')$, then $M[x := v] \Longrightarrow_\beta M'[x := v]$.

(c) If $M \equiv_\alpha M' \Longrightarrow_\beta N' \equiv_\alpha N$, then $M \Longrightarrow_\beta N$.

(d) If $M \Longrightarrow_\beta M'$, then $M[x := N] \Longrightarrow_\beta M'[x := N]$.

(e) If $M \longrightarrow_\beta M'$, then $M \Longrightarrow_\beta M'$.

(f) If $M \Longrightarrow_\beta M'$, then $M \longrightarrow_\beta^* M'$.

(g) If $M \Longrightarrow_\beta M'$, $N \Longrightarrow_\beta N'$, then $M[y := N] \Longrightarrow_\beta M'[y := N']$.

(h) $\longrightarrow_\beta^*$ is the transitive closure of $\Longrightarrow_\beta$.

(i) If $M \Longrightarrow_\beta N$ then $N \Longrightarrow_\beta M^*$.

Proof: (a) - (d) are easy. (e) by Induction on $M \longrightarrow_\beta M'$. (f), (g) by induction on M. (h) by (e), (f).
(i): Induction on M:
Case $M \equiv a \Longrightarrow_\beta N$:

$$N \equiv a \Longrightarrow_\beta a \equiv M^* \ .$$

Case $M \equiv \lambda x.M_1 \Longrightarrow_\beta N$. Then $N \equiv \lambda y.N_1$ for some N_1,

$$M_1[x := v] \Longrightarrow_\beta N_1[y := v]$$

for some variable $v \notin \mathrm{Var}(M_1) \cup \mathrm{Var}(N_1)$. W.l.o.g. (by (b)) $v \notin \mathrm{Var}(M_1^*)$. $M_1[x := v]$ has smaller length than N, therefore by IH $N_1[y := v] \Longrightarrow_\beta M_1[x := v]^* \equiv M_1^*[x := v]$,

$$\lambda y.N_1 \Longrightarrow_\beta \lambda x.M_1^* \equiv M^* \ .$$

Case $M \equiv M_1 M_2 \Longrightarrow_\beta N$, M not a β-redex. Then $N \equiv N_1 N_2$ for some N_i s. t. $M_i \Longrightarrow_\beta N_i$ $(i = 1, 2)$,

$$N_1 N_2 \Longrightarrow_\beta M_1^* M_2^* \equiv M^* \ .$$

Case $M \equiv (\lambda x.M_1)M_2 \Longrightarrow_\beta N$. Then

$$N \equiv (\lambda x.N_1)N_2 \quad \text{or } N \equiv N_1[x := N_2]$$

for some N_i s. t.

$$M_i \Longrightarrow_\beta N_i$$

$(i = 1, 2)$. By IH

$$N_i \Longrightarrow_\beta M_i^*$$

$(i = 1, 2)$.
Subcase $N \equiv (\lambda x.N_1)N_2$.

$$N \Longrightarrow_\beta M_1^*[x := M_2^*] \equiv M^* \ .$$

Subcase $N \equiv N_1[x := N_2]$.

$$N \Longrightarrow_\beta M_1^*[x := M_2^*] \equiv M^*$$

by (g).

Lemma 2.1.18 *(Diamond property for $\Longrightarrow_\beta$) Assume*

- $M \Longrightarrow_\beta N_1$,
- $M \Longrightarrow_\beta N_2$,

Then there exists some M' such that

- $N_1 \Longrightarrow_\beta M'$,
- $N_2 \Longrightarrow_\beta M'$.

Proof: $M' := M^*$.

Definition 2.1.19 Define $P \Longrightarrow_\beta^k Q$ by:

- $P \Longrightarrow_\beta^0 P$.
- If $P \Longrightarrow_\beta^k Q \Longrightarrow_\beta R$, then $P \Longrightarrow_\beta^{k+1} R$.

Lemma 2.1.20 *If*

- $M \Longrightarrow_\beta^k N_1$,
- $M \Longrightarrow_\beta^l N_2$,

then there exists some M' s. t.

- $N_1 \Longrightarrow_\beta^l M'$,
- $N_2 \Longrightarrow_\beta^k M'$.

Proof: First proof for $l = 1$ by induction on k:
$k = 0$: $N_1 \equiv M$, $M' := N_2$.
$k \to k+1$: Let

$$M \Longrightarrow_\beta^k N_1' \Longrightarrow_\beta N_1 \ .$$

By IH there exists some M'' s. t.

$$\begin{array}{lll} N_1' & \Longrightarrow_\beta & M'', \\ N_2 & \Longrightarrow_\beta^k & M'' \ . \end{array}$$

By the Diamond property Lemma 2.1.18 there exists M' s. t.

$$\begin{array}{rll} N_1 & \Longrightarrow_\beta & M' \ , \\ M'' & \Longrightarrow_\beta & M' \ . \end{array}$$

Similarly follows now by induction on l the full assertion.

Proof of Theorem 2.1.15: Lemmata 2.1.20, 2.1.17 (h).

Corollary 2.1.21 *(Corollary 1.29.1) If P has β-normal forms M, N then $M \equiv_\alpha N$.*

Proof: There exists T, s. t.

$$\begin{array}{rll} M & \longrightarrow_\beta^* & T \ , \\ N & \longrightarrow_\beta^* & T \end{array}$$

By M, N in normal form follows

$$\begin{array}{rll} M & \equiv_\alpha & T \\ N & \equiv_\alpha & T \end{array}$$

Lemma 2.1.22 *(Lemma 1.30) The class of β-normal forms is the smallest class $\mathcal{A}$ s. t.*

- *All atoms are in $\mathcal{A}$.*
- *If $M_1, \ldots, M_N \in \mathcal{A}$, a is any atom, then*

$$aM_1, \ldots, M_n \in \mathcal{A} \ .$$

- *If $M \in \mathcal{A}$, then $\lambda x.M \in \mathcal{A}$.*

Proof: Immediate.

2.1.5 β-equality (1D)

Definition 2.1.23 (1.32) Let $=_\beta$ be the symmetric and transitive closure of $\longrightarrow^*_\beta$.
We say P is β-*equal* or β-*convertible* to Q if $P =_\beta Q$.
It follows immediately that $=_\beta$ is reflexive as well.

Lemma 2.1.24 *(1.33)*
If

$$P' \equiv_\alpha P =_\beta Q \equiv_\alpha Q'$$

then

$$P' =_\beta Q' \ .$$

Lemma 2.1.25 *(1.34, Substitution lemma for β-equality).*
Assume $P =_\beta Q$.

(a) $M[x := P] =_\beta M[x := Q]$.

(b) $P[x := N] =_\beta Q[x := N]$.

Theorem 2.1.26 *(1.35; Church-Rosser theorem for β-equality)*
If

$$P =_\beta Q$$

then there exists T s. t.

$$P \longrightarrow^*_\beta T \longleftarrow^*_\beta Q$$

Proof:
Define

$$P \approx_\beta Q :\Leftrightarrow \exists T. P \longrightarrow^*_\beta T \longleftarrow^*_\beta Q$$

Then we have

- $P \approx_\beta Q \Rightarrow P =_\beta Q$.
- $P \equiv_\alpha Q \Rightarrow P \approx_\beta Q$.
- $P \longrightarrow_\beta Q \Rightarrow P \approx_\beta Q$.
- $\approx_\beta$ is symmetric.
- $\approx_\beta$ is transitive. (By Church Rosser).
- $=_\beta \subseteq \approx_\beta$.
- $=_\beta$ and $\approx_\beta$ coincide.

Corollary 2.1.27 *(Corollary 1.35.1) If*

- $P =_\beta Q$
- *Q in β-normal form*

then

$$P \longrightarrow_\beta^* Q \ .$$

Proof:

$$P \longrightarrow_\beta^* T \longleftarrow_\beta^* Q$$

for some T.

$$T \equiv_\alpha Q \ .$$

$$P \longrightarrow_\beta^* T \longrightarrow_\beta^* Q \ .$$

Corollary 2.1.28 *(Corollary 1.35.2) If $P =_\beta Q$ then*

- *either P and Q do not have β-normal forms or*
- *P and Q have the same β-normal forms.*

Corollary 2.1.29 *(Corollary 1.35.3) If $P =_\beta Q$, P, Q are in normal form then $P \equiv_\alpha Q$.*

Corollary 2.1.30 *(Corollary 1.35.4) A term can be β-equal to at most one β-normal form (up to $\equiv_\alpha$).*

Corollary 2.1.31 *(Corollary 1.35.5) Assume*

$$xM_1 \cdots M_m =_\beta yN_1 \cdots N_n$$

Then $x \equiv y$, $m = n$, $M_i =_\beta N_i$ for $i = 1, \ldots, m$.

Proof: Let

$$xM_1 \cdots M_m \longrightarrow_\beta^* T \longleftarrow_\beta^* yN_1, \ldots, N_n$$

Then T must be of the form $xT_1, \ldots, T_m$ s. t. $M_i \longrightarrow_\beta^* T_i$. Similarly $N_j \longrightarrow_\beta^* T_i$, $y \equiv x$, $n = m$ and the assertion.

2.2 Combinatory logic (2)

2.2.1 Introduction (2A)

Definition 2.2.1 (2.1, 2.2, 2.3, 2.4., 2.5)

(a) The *set of terms in combinatory logic*, in short *set of* CL-*terms* is defined as the set of λ-terms, but with two additional constants **k** and **s** and without closure under λ-abstraction.

(b) If **k**, **s** are the only constants, the system is called *pure*, otherwise *applied*

(c) In this section capital roman letters denote CL-terms. The other conventions are as in the last section.

(d) lgh(A), substitution is defined as for λ-terms.

2.2.2 Weak reduction (2B)

Definition 2.2.2 (2.7, 2.8)

(a) Inductive definition of $U \longrightarrow_{\mathrm{w}} V$ (U weakly contracts to V) for CL-terms U, V:

- $\mathbf{k}xy \longrightarrow_{\mathrm{w}} x$.
- $\mathbf{s}xyz \longrightarrow_{\mathrm{w}} xz(yz)$.
- If $U \longrightarrow_{\mathrm{w}} V$ then $XU \longrightarrow_{\mathrm{w}} XV$, and $UX \longrightarrow_{\mathrm{w}} VX$.

(b) $\longrightarrow_{\mathrm{w}}^{*}$ is the reflexive and transitive closure of $\longrightarrow_{\mathrm{w}}$. We say U weakly reduces to V for $U \longrightarrow_{\mathrm{w}}^{*} V$.

(c) A weak normal form is a term U s. t. U contracts to no other term.

(d) If U weakly reduces to a weak normal form X, then X is called the weak normal form of U.

This is a direct continuation of Handout 6.

Lemma 2.2.3 *(2.12; Substitution lemma for weak reduction) Assume $P \longrightarrow_{\mathrm{w}}^{*} Q$.*

(a) $\mathrm{FV}(Q) \subseteq \mathrm{FV}(P)$.

(b) $M[x := P] \longrightarrow_{\mathrm{w}}^{*} M[x := Q]$.

(c) $P[x := N] \longrightarrow_{\mathrm{w}}^{*} Q[x := N]$.

Proof: As Lemma 2.1.14 (1.28).

Theorem 2.2.4 *(2.13, Church-Rosser theorem for weak reduction)*
If $P \longrightarrow_{\mathrm{w}} M$, $P \longrightarrow_{\mathrm{w}} N$, then there exists T such that $M \longrightarrow_{\mathrm{w}}^{*} T$, $N \longrightarrow_{\mathrm{w}}^{*} T$.

Proof: An easy adaption of Takahashi's proof. Left as an exercise.

2.2.3 Definition of λ-abstraction in combinatory logic (2C)

As in HA^{ω} we can now introduce λ-terms:

Definition 2.2.5 (a) For CL-terms M and variables x we define $\lambda^{*}x.M$ as follows:

- Case: $x \notin \mathrm{FV}(M)$. $\lambda^{*}x.M := \mathbf{k}\ M$.
- Case $M \equiv N\ x$, $x \notin \mathrm{FV}(N)$. $\lambda^{*}x.M := N$.
- Otherwise
  - Subcase $M = x$.

$$\lambda^{*}x.M := \mathbf{s}\ \mathbf{k}\ \mathbf{k}\ .$$

– Subcase $M = P\ Q$.

$$\lambda^* x.M := \mathbf{s}\ (\lambda^* x.P)\ (\lambda^* x.Q)\ \ .$$

(b) $\lambda^* x_1, \dots, x_n.t := \lambda^* x_1.\lambda^* x_2.\dots.\lambda^* x_n.t$.
$\lambda^* \vec{x}.t := \lambda^* x_1, \dots, x_n.t$, if $\vec{x} = x_1, \dots, x_n$.

Theorem 2.2.6 *(2.15)* $(\lambda^* x.M)N \longrightarrow^*_{\mathrm{w}} M[x := N]$.

Proof:
By Lemma 2.2.3 (c) it suffices to prove:

$$(\lambda^* x.M)\ x \longrightarrow^*_{\mathrm{w}} M\ \ .$$

Induction on lgh(M).
Case: $x \notin \mathrm{FV}(M)$.

$$(\lambda^* x.M)\ x \equiv \mathbf{k}\ M\ x \longrightarrow_{\mathrm{w}} M\ \ .$$

Case $M \equiv N\ x$, $x \notin \mathrm{FV}(N)$.

$$(\lambda^* x.M)\ x \equiv N\ x \equiv M\ \ .$$

Case Otherwise.
Subcase $M \equiv x$.

$$(\lambda^* x.x)\ x \equiv \mathbf{s}\ \mathbf{k}\ \mathbf{k}\ x \longrightarrow_{\mathrm{w}} (\mathbf{k}\ x)\ (\mathbf{k}\ x) \longrightarrow_{\mathrm{w}} x$$

Subcase $M \equiv P\ Q$.

$$\begin{array}{rcl}(\lambda^* x.M)\ x & \equiv & \mathbf{s}\ (\lambda^* x.P)\ (\lambda^* x.Q)\ x \\ & \longrightarrow_{\mathrm{w}} & ((\lambda^* x.P)\ x)\ ((\lambda^* x.Q)\ x) \\ & \longrightarrow^*_{\mathrm{w}} & P\ Q \equiv M\ \ .\end{array}$$

Lemma 2.2.7 *(2.21; Substitution and abstraction lemma)*

(a) $\mathrm{FV}(\lambda^* M) = \mathrm{FV}(M) \setminus \{x\}$.

(b) If $y \notin \mathrm{FV}(M)$, *then*

$$\lambda^* x.M \equiv \lambda^* y.(M[x := y])$$

(c) If $y \notin \mathrm{FV}(x\ N)$, *then*

$$(\lambda^* y.M)[x := N] \equiv \lambda^* y.(M[x := N])$$

Proof: Induction on M.

2.2.4 Weak equality (2D)

Definition 2.2.8 (2.22) Let $=_{\mathrm{w}}$ be the symmetric and transitive closure of $\longrightarrow^*_{\mathrm{w}}$.
We say P is *weakly equal* or *weakly convertible* to Q if $P =_{\mathrm{w}} Q$.
It follows immediately that $=_{\mathrm{w}}$ is reflexive as well.

Lemma 2.2.9 *(1.34, Substitution lemma for weak equality).*
Assume $P =_{\mathrm{w}} Q$.

(a) $M[x := P] =_{\mathrm{w}} M[x := Q]$.

(b) $P[x := N] =_{\mathrm{w}} Q[x := N]$.

Theorem 2.2.10 *(2.24; Church-Rosser theorem for weak equality)*
If

$$P =_{\mathrm{w}} Q$$

then there exists T s. t.

$$P \longrightarrow^*_{\mathrm{w}} T \longleftarrow^*_{\mathrm{w}} Q$$

Proof: As for Theorem 2.1.26(1.35).

Corollary 2.2.11 *(Corollary 2.24.1 - 5)*

(a) If $P =_{\mathrm{w}} Q$ and Q is in weak normal form then $P \longrightarrow^*_{\mathrm{w}} Q$.

(b) If $P =_{\mathrm{w}} Q$ then

- *either P and Q do not have weak normal forms or*
- *P and Q have the same weak normal forms.*

(c) If P, Q are distinct weak normal forms, then $P \neq_{\mathrm{w}} Q$. In particular $\mathbf{s} \neq_{\mathrm{w}} \mathbf{k}$, $=_{\mathrm{w}}$ is nontrivial.

(d) If $P =_{\mathrm{w}} Q$, P, Q are in normal form, then $P \equiv Q$.

(e) A term can be weakly equal to at most one weak normal form.

(f) Assume

$$xM_1 \cdots M_m =_{\mathrm{w}} yN_1 \cdots N_n$$

Then $x \equiv y$, $m = n$, $M_i =_{\mathrm{w}} N_i$ for
$i = 1, \ldots, m$.

Remark 2.2.12 *(**Warning**, 2.25) It seems that combinatory logic and λ-calculus are exactly the same. But there is one difference: In λ-calculus we have the so called ξ-rule:*

$$M \longrightarrow_\beta N \Rightarrow \lambda x.M \longrightarrow_\beta \lambda x.N \quad ,$$

which can be weakened to

$$M =_\beta N \Rightarrow \lambda x.M =_\beta \lambda x.N \quad .$$

However in general in combinatory logic we do **not** *have*

$$M =_{\mathrm{w}} N \Rightarrow \lambda^* x.M =_{\mathrm{w}} \lambda^* x.N \ .$$

Counterexample

$$M := \mathbf{s}\ x\ y\ z \quad N := (x\ z)\ (y\ z)$$

We have

$$\begin{array}{rcl} M & =_{\mathrm{w}} & N \\ \lambda^* x.M & \equiv & \lambda^* x.(\mathbf{s}\ x\ y\ z) \\ & \equiv & \mathbf{s}\ (\lambda^* x.\mathbf{s}\ x\ y)\ (\lambda^* x.z) \\ & \equiv & \mathbf{s}\ (\mathbf{s}\ (\lambda^* x.\mathbf{s}\ x)\ (\lambda^* x.y))\ (\mathbf{k}\ z) \\ & \equiv & \mathbf{s}\ (\mathbf{s}\ \mathbf{s}\ (\mathbf{k}\ y))\ (\mathbf{k}\ z)\ \ , \\ \lambda^* x.N & \equiv & \lambda^* x.(x\ z)\ (y\ z) \\ & \equiv & \mathbf{s}\ (\lambda^* x.x\ z)\ (\lambda^* x.y\ z) \\ & \equiv & \mathbf{s}\ (\mathbf{s}\ (\lambda^* x.x)\ (\lambda^* x.z))\ (\mathbf{k}\ (y\ z)) \\ & \equiv & \mathbf{s}\ (\mathbf{s}\ (\mathbf{s}\ \mathbf{k}\ \mathbf{k})\ (\mathbf{k}\ z))\ (\mathbf{k}\ (y\ z)) \end{array}$$

$\lambda^* x.M$ *and* $\lambda^* x.N$ *are in normal form and different, therefore*

$$\lambda^* x.M \neq_{\mathrm{w}} \lambda^* x.N$$

2.3 The fixed point and quasi-leftmost-reduction theorem (3B, 3D)

2.3.1 Introduction (3A)

‘

Notation 2.3.1 (3.1., 3.2.)

(a) This section works both for λ-calculus and CL. The notation used in this section should be read in λ-calculus and in CL as follows

Notation	Interpretation for λ	Interpretation for CL
Term	λ-term	CL-term
$X \equiv Y$	$X \equiv_\alpha Y$	X, Y are identical
$X \longrightarrow_{\beta,\mathrm{w}} Y$	$X \longrightarrow_\beta Y$	$X \longrightarrow_{\mathrm{w}} Y$
$X \longrightarrow^*_{\beta,\mathrm{w}} Y$	$X \longrightarrow^*_\beta Y$	$X \longrightarrow^*_{\mathrm{w}} Y$
$X =_{\beta,\mathrm{w}} Y$	$X =_\beta Y$	$X =_{\mathrm{w}} Y$
$\lambda x.Y$	$\lambda x.Y$	$\lambda^* x.Y$
Combinator	Closed Term not containing constants	Term that has as atoms only $\mathbf{k}$ and $\mathbf{s}$
$\mathbf{s}$	$\lambda x, y, z.x\ z\ (y\ z)$	$\mathbf{s}$
$\mathbf{k}$	$\lambda x, y.x$	$\mathbf{k}$

(b)

$$\begin{array}{rcll}
\mathbf{I} & := & \lambda x.x & (\text{in CL } \equiv \ \mathbf{s\,k\,k}) \\
\mathbf{B} & := & \lambda x,y,z.x\ (y\ z) & (\text{in CL } \equiv \ \mathbf{s}\ (\mathbf{k\,s})\ \mathbf{k}) \\
\mathbf{W} & := & \lambda x,y.(x\ y)\ y & (\text{in CL } \equiv \ \mathbf{s\,s}\ (\mathbf{k\,I}))
\end{array}$$

2.3.2 The fixed-point theorem (3B)

Definition 2.3.2 (3.4) We define two fixed point combinators (there are others):

(a) $\mathbf{Y}_{\text{Curry}} := \lambda x.VV$ with $V := \lambda y.x\ (y\ y)$

(b) $\mathbf{Y}_{\text{Turing}} := Z\ Z$ with $Z := \lambda z,x.x\ (z\ z\ x)$.

(c) If not denoted differently $\mathbf{Y}$ will in the following be $\mathbf{Y}_{\text{Turing}}$ (if (a) of the next theorem suffices, one can use $\mathbf{Y}_{\text{Curry}}$ as well).

Theorem 2.3.3 *(a)* $\mathbf{Y}_{\text{Curry}}\ x =_{\beta,\mathrm{w}} x\ (\mathbf{Y}_{\text{Curry}}\ x)$

(b) $\mathbf{Y}_{\text{Turing}}\ x \longrightarrow^*_{\beta,\mathrm{w}} x\ (\mathbf{Y}_{\text{Turing}}\ x)$

Proof:
Only (b): Let $\mathbf{Y} := \mathbf{Y}_{\text{Turing}}$.

$$\begin{array}{rcl}
\mathbf{Y}\ x \equiv Z\ Z\ x & \equiv & (\lambda z,x.x\ (z\ z\ x))\ Z\ x \\
& \longrightarrow^*_{\beta,\mathrm{w}} & x\ (Z\ Z\ x) \\
& \equiv & x\ (\mathbf{Y}\ x)
\end{array}$$

Theorem 2.3.4 *(3.3.1, 3.3.2)*

(a) In both λ *and* CL*, for any term* Z *(possibly containing the variables* $x, y_1, \ldots, y_n$ *free) and* $n \geq 0$ *there is a term* X *s. t.*

$$X y_1 \cdots y_n \longrightarrow^*_{\beta,\mathrm{w}} Z[x := X]\ .$$

(b) In both λ *and* CL*, for any* $k > 0$*,* $n > 0$*, terms* $Z_0, \ldots, Z_{k-1}$ *(possibly containing the variables* $x_0, \ldots, x_{k-1},, y_1, \ldots, y_n$ *free) there exists terms* X_i *s. t.*

$$\begin{array}{rcl}
X_i\ y_1 \cdots y_n & \longrightarrow^*_{\beta,\mathrm{w}} & Z_i[x_0 := X_0, \ldots, x_{k-1} := X_{k-1}] \\
& & (i = 0, \ldots, k-1)\ .
\end{array}$$

(c) In both λ *and* CL*, for any terms* X, Y *there exists* P*,* Q *s. t.*

$$P \longrightarrow_{\beta,\mathrm{w}} X\ P\ Q \quad Q =_{\beta,\mathrm{w}} Y\ P\ Q$$

Proof:
(a):

$$X := \mathbf{Y}\ (\lambda x, y_1, \ldots, y_n.Z)\ .$$

$$\begin{array}{rcl} X\ y_1 \cdots y_n & \equiv & \mathbf{Y}\ (\lambda x, y_1, \ldots, y_n.Z) y_1 \cdots y_n \\ & \longrightarrow^*_{\beta,\mathrm{w}} & (\lambda x, y_1, \ldots, y_n.Z)\ X\ y_1, \ldots, y_n \\ & \longrightarrow^*_{\beta,\mathrm{w}} & Z[x := X] \end{array}$$

(b) We prove first the following Lemma:

Lemma 2.3.5 *(2.26)*

(a) There exists a pairing combinator $\mathbf{p}$ *and corresponding projections* $\mathbf{p}_i$ *(*$i = 0, 1$*) s. t.*

$$\mathbf{p}_i\ (\mathbf{p}\ x_0\ x_1) \longrightarrow^*_{\beta,\mathrm{w}} x_i \quad (i = 0, 1)\ .$$

(b) For every $n \geq 1$ *there exists sequence combinators* $\mathbf{p}^n$ *and corresponding projections* $\mathbf{p}^n_i$ *(*$i = 0, \ldots, n-1$*) s. t.*

$$\mathbf{p}_i\ (\mathbf{p}^n_i\ x_0\ x_1 \cdots x_n) \longrightarrow^*_{\beta,\mathrm{w}} x_i \quad (i = 0, , \ldots, n-1)\ .$$

Proof:
(a):

$$\begin{array}{rcl} \mathbf{p} & := & \lambda x, y, z.z\ x\ y \\ \mathbf{p}_i & := & \lambda u.u(\lambda x_0, x_1.x_i) \end{array}$$

Then

$$\begin{array}{rcl} \mathbf{p}_i\ (\mathbf{p}\ x_0\ x_1) & \equiv & \mathbf{p}_i((\lambda x, y, z.z\ x\ y)\ x_0\ x_1) \\ & \longrightarrow^*_{\beta,\mathrm{w}} & \mathbf{p}_i(\lambda z.z\ x_0\ x_1) \\ & \equiv & (\lambda u.u\ (\lambda x_0, x_1.x_i))\ (\lambda z.z\ x_0\ x_1) \\ & \longrightarrow^*_{\beta,\mathrm{w}} & (\lambda z.z\ x_0\ x_1)\ (\lambda x_0, x_1.x_i) \\ & \longrightarrow^*_{\beta,\mathrm{w}} & (\lambda x_0, x_1.x_i)\ x_0\ x_1 \\ & \longrightarrow^*_{\beta,\mathrm{w}} & x_i \end{array}$$

(b): Induction on n:
$n = 1$: $\mathbf{p}^1 := \mathbf{p}^1_0 := \mathbf{I}$.
$n \longrightarrow n + 1$:

$$\begin{array}{rcl} \mathbf{p}^{n+1} & := & \lambda x_1, \ldots, x_{n+1}.\mathbf{p}\ (\mathbf{p}^n\ x_1, \ldots, x_n)\ x_{n+1} \\ \mathbf{p}^{n+1}_i & := & \lambda u.\mathbf{p}^n_i\ (\mathbf{p}_0\ u) \quad (i = 0, \ldots, n-1) \\ \mathbf{p}^{n+1}_n & := & \mathbf{p}_1 \end{array}$$

Proof of part (b) of Theorem 2.3.4:

Let

$$\begin{array}{rcl}
\vec{y} & := & y_1, \ldots, y_k \ , \\
Z'_i & := & Z_i[x_0 := \lambda\vec{y}.\mathbf{p}^k_0\ (x\ \vec{y}), \ldots, x_{k-1} := \lambda\vec{y}.\mathbf{p}^k_{k-1}\ (x\ \vec{y})] \\
Z & := & \mathbf{p}^k\ Z'_0 \cdots Z'_{k-1} \\
 & & X \text{ chosen according to 2.3.4 (a) s. t.} \\
X\ \vec{y} & \longrightarrow_{\beta,\mathrm{w}} & Z[x := X] \ , \\
X_i & := & \lambda\vec{y}.\mathbf{p}^k_i(X\ \vec{y}) \\
 & & \text{then} \\
X_i\ \vec{y} & \equiv & \mathrm{p}^k_i\ (X\ \vec{y}) \\
 & \longrightarrow^*_{\beta,\mathrm{w}} & \mathrm{p}^k_i\ (Z[x := X]) \\
 & \longrightarrow^*_{\beta,\mathrm{w}} & Z'_i[x := X] \\
 & \equiv & Z_i[x_0 := \lambda\vec{y}.\mathbf{p}^k_0\ (X\ \vec{y}), \ldots, x_{k-1} := \lambda\vec{y}.\mathbf{p}^k_{k-1}\ (X\ \vec{y})] \\
 & \equiv & Z_i[x_0 := X_0, \ldots, x_{k-1} := X_{k-1}]
\end{array}$$

Proof of part (c) of Theorem 2.3.4:
Let in 2.3.4 (b) $n := k := 2$, $Z_i := y_i\ (x_0\ y_1\ y_2)\ (x_1\ y_1\ y_2)$.
Let X_1, X_2 s. t.

$$X_i\ y_1\ y_2 \longrightarrow_{\beta,\mathrm{w}} Z_i[x_0 := X_0, x_1 := X_1]$$

Let $P := X_0\ X\ Y$, $Q := X_1\ X\ Y$.
Then

$$\begin{array}{rcl}
P & \longrightarrow_{\beta,\mathrm{w}} & Z_0[x_0 := X, x_1 := Y, y_1 := X, y_2 := Y] \\
 & \equiv & X\ (X_0\ X\ Y)\ (X_1\ X\ Y) \\
 & \equiv & X\ P\ Q
\end{array}$$

similarly

$$Q \longrightarrow_{\beta_{\mathrm{w}}} Y\ P\ Q$$

2.3.3 The quasi-leftmost-reduction theorem (3D)

The Quasi-leftmost reduction theorem will not be shown in this lecture. The only proof we could find is quite complicated, but we assume that using the new techniques available a simpler proof can be given.

Definition 2.3.6 (approx. 3.17, 3.18)

(a) We define $N \longrightarrow_{\mathrm{l},\beta} M$ and $N \longrightarrow_{\mathrm{l,w}} M$, where we write $N \longrightarrow_{\mathrm{l},\beta,\mathrm{w}} M$ for either $N \longrightarrow_{\mathrm{l},\beta} M$ (in the case of λ-calculus) or $N \longrightarrow_{\mathrm{l,w}} M$ (in the case of combinatory logic) and pronounce this expression as N *leftmost weakly reduces/β-reduces to* M, inductively by:

- $\mathbf{k}\ N\ M \longrightarrow_{\mathrm{l,w}} N$;
- $\mathbf{s}\ N\ M\ P \longrightarrow_{\mathrm{l,w}} (N\ P)\ (M\ P)$;
- $(\lambda x.M)\ N \longrightarrow_{\mathrm{l},\beta} M[x := N]$;

- If $M \longrightarrow_{1,\beta} N$, then $\lambda x.M \longrightarrow_{1,\beta} \lambda x.N$.
- If $M \longrightarrow_{1,\beta,\mathrm{w}} N$, M not of the form $\lambda x.P$, $\mathbf{k}\ P$, $\mathbf{s}\ P\ Q$, then $M\ R \longrightarrow_{1,\beta,\mathrm{w}} N\ R$.
- If $M \longrightarrow_{1,\beta,\mathrm{w}} N$, R in normal form and not of the form $\lambda x.P$, $\mathbf{k}\ P$, $\mathbf{s}\ P\ Q$, then $R\ M \longrightarrow_{1,\beta,\mathrm{w}} R\ N$.

So $N \longrightarrow_{1,\beta,\mathrm{w}} M$ means, that the leftmost maximal redex in N is reduced in the reduction of N to M.

(b) $N \longrightarrow^{+}_{1,\beta,\mathrm{w}} M :\Leftrightarrow \exists N' N \longrightarrow^{*}_{\beta,\mathrm{w}} N' \longrightarrow_{1,\beta,\mathrm{w}} M$.

(c) A *quasi-leftmost reduction* of a term X is

- either a finite sequence $(X_1, \ldots, X_n)$ s. t.

$$X \equiv X_1 \longrightarrow^{+}_{1,\beta,\mathrm{w}} X_2 \longrightarrow^{+}_{1,\beta,\mathrm{w}} \cdots \longrightarrow^{+}_{1,\beta,\mathrm{w}} X_n$$

  and X_n is normal or
- an infinite sequence $X_1, X_2 \cdots$ s. t.

$$X \equiv X_1 \longrightarrow^{+}_{1,\beta,\mathrm{w}} X_2 \longrightarrow^{+}_{1,\beta,\mathrm{w}} \cdots$$

Theorem 2.3.7 *(Quasi-leftmost-reduction theorem 3.19, 3.19.1)*

(a) In both λ-calculus and combinatory logic, if X has a normal form X^*, then every quasi-leftmost reduction of X is finite and terminates at X^*.

(b) In both λ-calculus and combinatory logic, X has no normal form iff some quasi-leftmost reduction of X is infinite.

Proof:
(b) follows directly by (a). (a) will not be proved.

2.4 Representing the recursive functions (4)

Notation 2.4.1 (a) The notation as in 2.3.1 apply to this section as well.

(b) i, j, k, m, n denote natural numbers.

Definition 2.4.2 (a) $X^0\ Y := Y$, $X^{n+1}\ Y := X(X^n\ Y)$
(or $X^n\ Y = \underbrace{X(X(\cdots(X}_{n \text{ times}}Y)\cdots)$)

(b) A partial function n-ary function on the natural numbers is a function $\mathbb{N}^n \longrightarrow \mathbb{N} \cup \{\bot\}$, where $\bot$ is a symbol for undefined. Abbreviations like $f(\vec{n}) \downarrow$, $f(\underline{n}) \uparrow$, $f(\underline{n}) \simeq m$ are as usual. For terms t we define, whether it is defined, and for two terms s, t, whether $s \simeq t$, as usual (where for being defined we require that all sub-terms are defined, even if they are not needed in the computation of the whole term).

(c) If not defined differently let in the following
$\langle X, Y\rangle := \mathbf{p}\ X\ Y,\ X0 := \mathbf{p}_0\ x,\ X1 := \mathbf{p}_1\ x$.

Definition 2.4.3 (Church numerals, 4.2)

$$\begin{array}{rcl} \mathbf{N}_n := \underline{n} & := & \lambda x, y.x^n\ y \\ & & \text{(in Combinatory logic } \equiv (\mathbf{s}\ \mathbf{B})^n\ (\mathbf{k}\ \mathbf{I}))\ . \end{array}$$

$\underline{n}$ is called the *nth Church numeral* or the *Church numeral representing* n

Therefore we have

$$\underline{n}\ F\ X \longrightarrow_{\beta,\mathrm{w}} F^n\ X\ .$$

There are other representations of the natural numbers.
Proof of the above equality by induction on n:

$$\begin{array}{rcl} \underline{0} & \equiv & \lambda x, y.y \\ & \equiv & \lambda x.\mathbf{I} \\ & \equiv & \mathbf{k}\ \mathbf{I} \\ \underline{n+1} & \equiv & \lambda x, y.x\ (x^n\ y) \\ & \equiv & \lambda x.\mathbf{s}\ (\mathbf{k}\ x)\ (\lambda y.x^n\ y) \\ & \equiv & \mathbf{s}\ (\lambda x.\mathbf{s}\ (\mathbf{k}\ x))\ (\lambda x, y.x^n\ y) \\ & \equiv & \mathbf{s}\ (\mathbf{s}\ (\mathbf{k}\ \mathbf{s})\ \mathbf{k})\ \underline{n} \\ & \equiv & (\mathbf{s}\ \mathbf{B})\ (\mathbf{s}\ \mathbf{B})^n\ (\mathbf{k}\ \mathbf{I}) \\ & \equiv & (\mathbf{s}\ \mathbf{B})^{n+1}\ (\mathbf{k}\ \mathbf{I}) \end{array}$$

Definition 2.4.4 (4.4)

(a) Let f be an n-ary partial function. We say a λ-term X *λ-defines* or a CL-term *combinatorially defines* (or if talking about λ- or CL-terms a term *defines*) f iff
for all $m_1, \ldots, m_n$ it holds

- $f(m_1, \ldots, m_n)\downarrow$ iff $X\ \underline{m_1} \cdots \underline{m_n}$ has a normal form;
- If $f(m_1, \ldots, m_n)\downarrow$ then

$$X\ \underline{m_1} \cdots \underline{m_n} =_{\beta,\mathrm{w}} \underline{f(m_1, \ldots, m_n)}\ .$$

Lemma 2.4.5 *(a) A term defines at most one function.*

(b) If a λ-term or CL*-term defines f, then f is partial recursive.*

Proof:
(a): The Church numerals are in normal form and pairwise not α-equivalent. Since all normal forms of a term are (α)-equivalent and every term in normal form that is w/β-equivalent to a term is a normal form of it, it follows that the function is uniquely defined.
(b) Let $X^{*,n}$ be defined by

$$X^{*,0} := X \quad X^{*,n+1} := (X^{*,n})^*$$

From Lemma 2.1.17 (h) and (i) it follows that if M is a normal form of a term N, then $N \equiv M^{*,k}$ for some k and for all $l > k$ $M^{*,l} = M^{*,k}$. $M^{*,k}$ is primitive recursive in (a code for the) term M and k. Further we can determine from a term, which is a α-equivalent to a Church numeral the number it represents in a primitive recursive way (for instance since $\mathrm{lgh}(\underline{n}) = n \cdot a + b$ for some global constant a, b) and of course a (code for a) $\underline{n}$ is primitive recursive in n. Therefore if N defines f we can compute f as follows: For input $m_1, \ldots, m_n$, let $U := N\underline{m_1} \cdots \underline{m_n}$. Evaluate $U^{*,k}$ until for some k $U^{*,k} \equiv U^{*,k+1}$ (which is primitive recursively decidable). If this doesn't happen, $f(m_1, \ldots, m_n)$ is undefined and the procedure doesn't terminate. If there is some k, then $U^{*,k}$ is α-equivalent to a Church numeral $\underline{\mathrm{n}}$. Determine n which is the result $f(n_1, \ldots, n_k)$.
The above procedure can now be written as a partial recursive function, which determines $f(n_1, \ldots, n_k)$ on input $n_1, \ldots, n_k$.

Next steps: We will show that every recursive function can be defined by a λ-term and a CL-term. We will first show that all primitive recursive functions can be defined in such a way.

Lemma 2.4.6 *(a) The successor function can be defined by* $\widehat{\mathrm{S}} := \lambda u, x, y.x\ (u\ x\ y)$ *(in* CL *this is* $\equiv \mathbf{s\ B}$*).*
$\widehat{\mathrm{S}}$ *is a combinator in normal form.*

(b) $\underline{0}$ *is defined by* $\underline{0}$ *(or* $\mathbf{k\ I}$*), which is a combinator in normal form.*

(c) The function $f(n_1, \ldots, n_k) = n_i$ *is defined by* $\lambda x_1, \ldots, x_k.x_i$*, which is a combinator in normal form.*

(d) If the n-ary functions g_i *are defined by terms* $\widehat{g_i}$*, the k-ary function* h *defined by* $\widehat{h}$ *and* $f(\vec{m}) = h(g_1(\vec{m}), \ldots, g_m(\vec{m}))$*, then* f *is defined by* $\widehat{f} := \lambda\vec{x}.\widehat{h}(\widehat{g_1}\ \vec{x}) \cdots (\widehat{g_m}\ \vec{x})$*.*
If $\widehat{h}$*,* $\widehat{g_i}$ *are combinators in normal form, so is* $\widehat{f}$*.*

(e) There is a combinator $\mathbf{R}$ *s. t.*

$$\begin{aligned} \mathbf{R}\ X\ Y\ \underline{0} &=_{\beta,\mathrm{w}} X\ , \\ \mathbf{R}\ X\ Y\ \underline{k+1} &=_{\beta,\mathrm{w}} Y\ \underline{k}\ (\mathbf{R}\ X\ Y\ \underline{k})\ . \end{aligned}$$

$\mathbf{R}$ *can be chosen as a combinator in normal form.*

(f) Every primitive recursive function can be defined by a combinator $\widehat{f}$ *in normal form.*

Proof:
(a) In λ-calculus

$$\widehat{\mathrm{S}}\ \underline{n} \longrightarrow^*_\beta \lambda x, y.x\ (\underline{n}\ x\ y) \longrightarrow^*_\beta \lambda x, y.x\ (x^n\ y) \equiv \underline{\mathrm{S}(n)}$$

In CL

$$\widehat{\mathrm{S}}\ \underline{n} \equiv (\mathbf{s\ B})\ (\mathbf{s\ B})^n\ (\mathbf{k\ I}) \equiv (\mathbf{s\ B})^{\mathrm{S}(n)}\ .$$

(b) - (d) Trivial.
(e) First step:
For every term Y there exists a term Z_Y s. t.

$$\forall n \in \mathbb{N}. Z_Y\ \langle \underline{n}, x\rangle =_{\beta,\mathrm{w}} \langle \underline{n+1}, (Y\ \underline{n}\ x)\rangle \quad :$$

Define

$$Z_Y := \lambda x.\langle \widehat{\mathrm{S}}\ (x0), Y\ (x0)\ (x1)\rangle \quad .$$

By $\langle x_0, x_1\rangle i =_{\beta,\mathrm{w}} x_i$ it follows the assertion.
Second Step:
Define

$$U_{X,Y} := \lambda x.x\ Z_Y\ \langle \underline{0}, X\rangle$$

Then

$$\begin{array}{rcl} U_{X,Y}\ \underline{0} & =_{\beta,\mathrm{w}} & \langle \underline{0}, X\rangle \\ U_{X,Y}\ \underline{n+1} & =_{\beta,\mathrm{w}} & \langle \underline{n}, Y\ \underline{n}\ ((U_{X,Y}\ \underline{n})1)\rangle \end{array}$$

The first assertion is clear, the second assertion follows by induction on n:

$$\begin{array}{rcl} U_{X,Y}\ \underline{1} & =_{\beta,\mathrm{w}} & Z_Y\ \langle \underline{0}, X\rangle \\ & =_{\beta,\mathrm{w}} & \langle \underline{1}, Y\ \underline{0}\ X\rangle \\ & =_{\beta,\mathrm{w}} & \langle \underline{1}, Y\ \underline{0}\ ((U_{X,Y}\ \underline{0})1)\rangle \\ U_{X,Y}\ \underline{n+2} & =_{\beta,\mathrm{w}} & Z_Y^{n+2}\ \langle \underline{0}, X\rangle \\ & =_{\beta,\mathrm{w}} & Z_Y\ (Z_Y^{n+1}\ \langle \underline{0}, X\rangle) \\ & =_{\beta,\mathrm{w}} & Z_Y\ (U_{X,Y}\ \underline{n+1}) \\ & =_{\beta,\mathrm{w}} & Z_Y\ \langle \underline{n+1}, Y\ \underline{n}\ X\rangle \\ & =_{\beta,\mathrm{w}} & Z_Y\ \langle \underline{n+1}, (U_{X,Y}\ (\underline{n+1})1)\rangle \\ & =_{\beta,\mathrm{w}} & \langle \underline{n+2}, Y\ \underline{n+1}\ ((U_{X,Y}\ \underline{n+1})1)\rangle \end{array}$$

Third Step:

$$\mathbf{R} := \lambda x, y, u.(U_{x,y}\ u)1$$

Then

$$\begin{array}{rcl} \mathbf{R}\ X\ Y\ \underline{0} & =_{\beta,\mathrm{w}} & X \\ \mathbf{R}\ X\ Y\ \underline{n+1} & =_{\beta,\mathrm{w}} & (U_{X,Y}\ \underline{n+1})1 \\ & = & \langle \underline{n+1}, Y\ \underline{n}\ ((U_{X,Y}\ \underline{n})1)\rangle 1 \\ & = & \langle \underline{n+1}, Y\ \underline{n}\ (\mathbf{R}\ X\ Y\ \underline{n})\rangle 1 \\ & = & Y\ \underline{n}\ (\mathbf{R}\ X\ Y\ \underline{n}) \end{array}$$

$\mathbf{R}$ is not in normal form, but one sees immediately that he reduces to a normal form.
(f): All cases except of primitive recursion follow directly by (a) - (e).
Case primitive recursion: Assume

$$f(\vec{n}, 0) = g(\vec{n}) \quad f(\vec{n}, m+1) = h(\vec{n}, m, f(\vec{n}, m))$$

Let $\widehat{g}$, $\widehat{h}$ terms (in normal form) defining g, h.

$$\widehat{f} := \lambda \vec{x}.\mathbf{R}\ (\widehat{g}\ \vec{x})\ (\lambda u, v.\widehat{h}\ \vec{x}\ u\ v)$$

By induction on m follows

$$\widehat{f}(\underline{n_1}, \ldots, \underline{n_m}, \underline{m}) =_{\beta,\mathrm{w}} \underline{f(n_1, \ldots, n_m, m)}\ .$$

Now we will show that all partial recursive functions can be defined by a normal combinator.

Lemma 2.4.7 *(a) There exists a combinator* $\mathbf{E}$ *in normal form s. t.*

$$\begin{array}{rcl} \mathbf{E}\ X\ Y\ \underline{0} & =_{\beta,\mathrm{w}} & X\ , \\ \mathbf{E}\ X\ Y\ \underline{n+1} & =_{\beta,\mathrm{w}} & Y\ . \end{array}$$

We write `if` x `then` y `else` z *for* $\mathbf{E}\ y\ z\ x$*.*

(b) There exists a combinator $\widehat{\mu}$ *in normal form s. t.*

$$\begin{array}{rcl} \widehat{\mu}\ X\ Y & =_{\beta,\mathrm{w}} & Y \\ & & \quad \textit{if } X\ Y =_{\beta,\mathrm{w}} \underline{0}\ , \\ \widehat{\mu}\ X\ Y & =_{\beta,\mathrm{w}} & P\ X\ (\widehat{\mathrm{S}}\ Y) \\ & & \quad \textit{if } X\ Y =_{\beta,\mathrm{w}} \underline{n+1}\ . \end{array}$$

Especially we have therefore, if $\widehat{f}$ *represents the unary function* f*,*

$$\mu y \geq n(f(y) = 0) \downarrow \Rightarrow \widehat{\mu}\ \widehat{f}\ \underline{n} =_{\beta,\mathrm{w}} \underline{\mu y \geq n(f(y) = 0)}\ .$$

Proof:
(a) $\mathbf{E} := \lambda x, y, z.z\ (\mathbf{k}\ y)\ x$.

$$\begin{array}{rcl} \mathbf{E}\ X\ Y\ \underline{0} & \equiv & \underline{0}\ (\mathbf{k}\ Y)\ X \\ & =_{\beta,\mathrm{w}} & (\lambda x, y.y)\ (\mathbf{k}\ Y)\ X\ , \\ & =_{\beta,\mathrm{w}} & X\ , \\ \mathbf{E}\ X\ Y\ \underline{n+1} & \equiv & \underline{n+1}\ (\mathbf{k}\ Y)\ X \\ & =_{\beta,\mathrm{w}} & (\mathbf{k}\ Y)^{n+1}\ X \\ & =_{\beta,\mathrm{w}} & Y\ . \end{array}$$

(b) Define

$$\mathbf{T}_X := \lambda y.\ \texttt{if}\ y\ \texttt{then}\ (\lambda u, v.v)\ \texttt{else}\ \lambda u, v.u\ (X\ (\widehat{\mathrm{S}}\ v))\ u\ (\widehat{\mathrm{S}}\ v)$$

Then we get

$$\begin{array}{rcl} \mathbf{T}_X\ (X\ Y)\ \mathbf{T}_X\ Y & =_{\beta,\mathrm{w}} & (\lambda u, v.v)\ \mathbf{T}_X\ Y \\ & =_{\beta,\mathrm{w}} & Y \\ & & \quad (\text{if } X\ Y =_{\beta,\mathrm{w}} \underline{0})\ , \\ \mathbf{T}_X\ (X\ Y)\ \mathbf{T}_X\ Y & =_{\beta,\mathrm{w}} & (\lambda u, v.u\ (X\ (\widehat{\mathrm{S}}\ v))\ u\ (\widehat{\mathrm{S}}\ v))\ \mathbf{T}_X\ Y \\ & =_{\beta,\mathrm{w}} & \mathbf{T}_X\ (X\ (\widehat{\mathrm{S}}\ Y))\ \mathbf{T}_X\ (\widehat{\mathrm{S}}\ Y) \\ & & \quad (\text{if } X\ Y =_{\beta,\mathrm{w}} \underline{n+1})\ . \end{array}$$

Let

$$\widehat{\mu} := \lambda x, y.\mathbf{T}_x\ (x\ y)\ \mathbf{T}_x\ y$$

Then $\widehat{\mu}$ fulfills the equations of the assertion.

Next step: We will show that every *total* recursive function can be defined by a term:

Lemma 2.4.8 *(Kleene, 4.15). Every recursive total function f can be defined by a combinator $\widehat{f}$.*

Note that $\widehat{f}$ is an overloaded notation, since we introduced it as a term representing a primitive recursive function and a term defining a recursive function (and will define it as a term defining a partial-recursive function) and these definitions do not coincide. However this will not cause problems since later we need just one term defining a function, independently of how it is exactly defined.

Proof:
By the Kleene normal form, f can be written as (with $\vec{m} := m_1, \ldots, m_n$)

$$f(\vec{m}) = h(\mu k.g(\vec{m}, k) = 0)\ ,$$

where h and g are primitive recursive.
Let in the following $\vec{x} = x_1, \ldots, x_n$, $\vec{X} := X_1, \ldots, X_n$.
Let

$$N :\equiv \lambda\vec{x}, y.\widehat{\mu}\ (\widehat{g}\ \vec{x})\ y$$

Then

$$\begin{array}{lll} N\ \vec{X}\ Y & =_{\beta,\mathrm{w}} & \widehat{\mu}\ (\widehat{g}\ \vec{X})\ Y \\ & =_{\beta,\mathrm{w}} & \begin{cases} Y & \text{if } \widehat{g}\ \vec{X}\ Y =_{\beta,\mathrm{w}} \underline{0}\ , \\ N\ \vec{X}\ (\widehat{\mathrm{S}}\ Y) & \text{if } \widehat{g}\ \vec{X}\ Y =_{\beta,\mathrm{w}} \underline{n+1}\ . \end{cases} \end{array}$$

Now

$$\widehat{f} := \lambda\vec{x}.\underline{h}(N\ \vec{x}\ \underline{0})\ .$$

Remark: The book claims that $\widehat{f}$ is in normal form, without giving a proof, but by saying that the proof is boring. The result is probably true here, but only because f is total and requires not (as said in the book) a boring but quite a sophisticated proof. Probably one needs to show that the terms defining the primitive recursive functions can be typed in the system F (this type system is not treated in this course), in the sense that if f is n-ary then $\widehat{f}$ is of type $\mathrm{nat}^n \longrightarrow \mathrm{nat}$, where $\mathrm{nat} = \forall\alpha.(\alpha \longrightarrow \alpha) \longrightarrow \alpha \longrightarrow \alpha$. Now one needs to show that $N\ \vec{x}\ \underline{0}$ can be assigned the type nat. Whether this is the case and exactly in which sense I don't know, and it can only work, if f is total, since otherwise for suitable $\underline{\vec{n}}$ N $\underline{\vec{n}}$ $\underline{0}$ does not normalize. If it can be done then $\widehat{f}$ can be typed as well and is therefore normalizing, so such an argument does not work for the term defined in the proof of the next theorem.

Theorem 2.4.9 *In both λ-calculus and combinatory logic every partial-recursive function f can be defined by a combinator $\underline{f}$ in normal form.*

From the proof of Lemma 2.4.8 we obtain as well for partial recursive functions f a term $\widehat{f}$ s. t. if $f(\vec{m})$ is defined, then

$$\widehat{f}\underline{\vec{m}} =_{\beta,\mathrm{w}} \underline{f(\vec{m})}$$

However, from this it does not follow that if $f(\vec{m})$ is undefined, $\widehat{f}\underline{\vec{m}}$ does not normalize. We will modify the definition given there in order to obtain a term which has the second property as well.

Proof:
As before write f as

$$f(\vec{m}) \simeq h(\mu k.g(\vec{m}, k) = 0) \ ,$$

where h and g are primitive recursive.
Let F be the term we obtained in 2.4.8 as $\widehat{f}$,

$$F :\equiv \lambda\vec{x}.\widehat{h}(N\ \vec{x}\ \underline{0}) \ .$$

Define

$$\widehat{f} := \lambda\vec{x}.((\widehat{\mu}\ (\widehat{g}\ \vec{x})\ \underline{0})\ \mathbf{I}\ (F\ \vec{x}) \ .$$

Assume $\vec{m}$ s. t. $f(\vec{m})$ is defined. Therefore there exists minimal j s. t. $g(\vec{m}, j) = 0$, and $f(\vec{m}) = h(j)$. Then

$$\begin{array}{rcl} \widehat{f}\underline{\vec{m}} & =_{\beta,\mathrm{w}} & \underline{j}\ \mathbf{I}\ (F\ \underline{\vec{m}}) \\ & =_{\beta,\mathrm{w}} & \mathbf{I}^j(F\ \underline{\vec{m}}) \\ & =_{\beta,\mathrm{w}} & F\ \underline{\vec{m}} \\ & =_{\beta,\mathrm{w}} & \underline{f(\vec{m})} \end{array}$$

Assume now $f(\vec{m}) \uparrow$. Let $p_k := g(\vec{m}, k) - 1$. Since $g(\vec{m}, k) > 0$, $g(\vec{m}, k) = p_k + 1$. Define

$$X := \widehat{g}\ \underline{\vec{m}}, \quad G := F\ \underline{\vec{m}}$$

$$\forall k.X\ \underline{k} =_{\beta,\mathrm{w}} \underline{p_k + 1}$$

by Church-Rosser and since the Church-numerals are in normal form it follows

$$\forall k.X\ \underline{k} \longrightarrow_{\beta,\mathrm{w}} \underline{p_k + 1}$$

We show $\widehat{f}\underline{\vec{m}}$ has no normal form by giving an infinite quasi-left-most reduction and using Theorem 2.3.7 (b).

$$\begin{array}{rcl} \widehat{f}\underline{\vec{m}} & \longrightarrow^*_{\beta,\mathrm{w}} & (\widehat{\mu}\ X\ \underline{0})\ \mathbf{I}\ G \\ & \longrightarrow^*_{\beta,\mathrm{w}} & (T_X\ (X\ \underline{0})\ T_X\ \underline{0})\ \mathbf{I}\ G \\ & \longrightarrow^*_{\beta,\mathrm{w}} & (T_X\ (\underline{p_0 + 1})\ T_X\ \underline{0})\ \mathbf{I}\ G \\ & \longrightarrow^*_{\beta,\mathrm{w}} & (\widehat{\mu}\ X\ \underline{1})\ \mathbf{I}\ G \\ & \longrightarrow^*_{\beta,\mathrm{w}} & (T_X\ (X\ \underline{1})\ T_X\ \underline{1})\ \mathbf{I}\ G \end{array}$$

$$\begin{array}{ll}\longrightarrow^*_{\beta,\mathrm{w}} & (T_X\ (\underline{p_1+1})\ T_X\ \underline{1})\ \mathbf{I}\ G \\ \longrightarrow^*_{\beta,\mathrm{w}} & \cdots\end{array}$$

$$X\underline{i} \longrightarrow \underline{p_i+1}$$

must contain one left-most-reduction (otherwise the left most redex in X remains unchanged contradicting that $\underline{p_i+1}$ is in normal form, and therefore

$$(T_X\ (X\ \underline{1})\ T_X\ \underline{1})\ \mathbf{I}\ G \longrightarrow^*_{\beta,\mathrm{w}} (T_X\ (\underline{p_1+1})\ T_X\ \underline{1})\ \mathbf{I}\ G$$

must include one left-most reduction. Therefore the above sequence is an infinite quasi-left-most reduction.

Remark $\underline{\mathrm{f}}$ is not normalizing in λ-calculus. Let $g(\vec{m}) = 1$. $\widehat{g}\vec{x} \longrightarrow_{\beta,\mathrm{w}} \underline{1}$ and with almost the same sequence as in the proof (replace $\widehat{f}\underline{\vec{m}}$ by $\widehat{f}$ and everywhere $\underline{\vec{m}}$ by $\vec{x}$) we get an infinite quasi-leftmost-reduction of $\widehat{f}$ (but not in combinatory logic, because the reduction will be "after the λ").

2.5 The undecidablity theorem (5)

Notation 2.5.1 (5.1)

(a) Relative to earlier versions of this scriptum we have made the following changes:

- A term representing a (primitive recursive, recursive or partial recursive) function f will now be denoted by $\widehat{f}$.
- Therefore the standard term representing the successor function is therefore denoted by $\widehat{\mathrm{S}}$.
- The function $\mathbb{N} \ni n \mapsto \underline{n}$ will now be denoted by $\mathbf{N}$ (and the term representing it therefore $\widehat{\mathbf{N}}$).
- We write $\widehat{\mu}$ instead of $\mathbf{P}$ since

$$\mu y \geq n(f(y) = 0) \downarrow \Rightarrow \widehat{\mu}\ \widehat{f}\ \underline{\mathrm{n}} =_{\beta,\mathrm{w}} \underline{\mu y \geq n(f(y) = 0)}\ .$$

(b) Again the notation as in 2.3.1 apply to this section as well.

(c) We assume some coding $\ulcorner M \urcorner$ of (λ- or CL-) terms s. t.

- there is a recursive (total) function $\circ$ s. t. $\circ(\ulcorner M \urcorner, \ulcorner N \urcorner) = \ulcorner M\ N \urcorner$ and
- the function $\mathbf{N} : \mathbb{N} \ni n \mapsto \ulcorner \underline{n} \urcorner$ is recursive.

(Of course both functions can be chosen primitive recursive).

(d) Let for $\mathcal{A}$ a set of terms

$$\ulcorner \mathcal{A} \urcorner := \{\ulcorner M \urcorner \mid M \in \mathcal{A}\}\ .$$

Note that this is the image of the function $\lambda x.\ulcorner x \urcorner$ to the set $\mathcal{A}$, therefore it is a set and not a natural number.

Remark 2.5.2 *In the book* $\ulcorner X \urcorner$ *is the Church numeral corresponding to the Gödelnumber of* X*. We write it as* $\underline{\ulcorner X \urcorner}$.

We can with almost no effort obtain the following undecidability theorem, which will then be generalized in Theorem 2.5.5.

Theorem 2.5.3 *(a) Both* $=_\beta$ *and* $=_\mathrm{w}$ *are undecidable.*

(b) It is undecidable whether a term has a normal form.

Proof:
(a) Let the unary partial recursive function f be defined by

$$f(e) :\simeq \begin{cases} 0 & \text{if } \{e\}(e)\downarrow, \\ \perp & \text{otherwise.} \end{cases}$$

Let $\widehat{f}$ be a term defining f. Then

$$\widehat{f}\ \underline{e} =_{\beta,\mathrm{w}} \underline{0} \Leftrightarrow \{e\}(e)\downarrow\ ,$$

so, if $=_{\beta,\mathrm{w}}$ were decidable, one could decide for $e \in \mathbb{N}$, whether $\{e\}(e)\downarrow$, which is undecidable.
(b): For the same $\widehat{f}$

$$\widehat{f}\ \underline{e} \text{ has a normal form} \Leftrightarrow \{e\}(e)\downarrow\ .$$

Definition 2.5.4 (5.4, 5.5)

(a) Assume $\mathcal{A}, \mathcal{B} \subseteq \mathbb{N}$. $\mathcal{A}$, $\mathcal{B}$ are called *recursively separable* iff there is a recursive set $\mathcal{C}$ s. t. $\mathcal{A} \subseteq \mathcal{C}$, $\mathcal{B} \cap \mathcal{C} = \emptyset$.

$\mathcal{A}$, $\mathcal{B}$ are *recursively inseparable*, iff they are not recursively separable.

(b) A set of $\mathcal{A}$ of terms is *closed under equality* iff for all terms X, Y

$$(X \in \mathcal{A} \wedge X =_{\beta,\mathrm{w}} Y) \Rightarrow Y \in \mathcal{A}\ .$$

The following theorem is a variant of the generalization of Rice's theorem[1], which states that if $\mathcal{A}$, $\mathcal{B}$ are disjoint nontrivial sets of recursive functions then the sets $\{e \mid \{e\} \in \mathcal{A}\}$ and $\{e \mid \{e\} \in \mathcal{B}\}$ are recursively inseparable. The proof is almost identical to the proof of that variant, which is more or less identical to the proof of the standard version of Rice's theorem.

Theorem 2.5.5 *(5.6, Scott-Curry undecidability theorem)*
For both λ*-terms and* β*-equality and* CL*-terms and weak equality it holds: Assume* $\mathcal{A}$, $\mathcal{B}$ *are sets of terms, s. t.*

[1] We think that this variant is called as well Rice's theorem, although we could not find this version in text books.

- *$\mathcal{A}$, $\mathcal{B}$ are closed under equality,*
- *$\mathcal{A} \cap \mathcal{B} = \emptyset$,*
- *$\mathcal{A} \neq \emptyset \neq \mathcal{B}$,*

then

$$\ulcorner \mathcal{A} \urcorner, \ulcorner \mathcal{B} \urcorner \textit{ are recursively inseparable } .$$

Proof:
Assume $\ulcorner \mathcal{A} \urcorner$, $\ulcorner \mathcal{B} \urcorner$ can be recursively separated. Let

- $\ulcorner \mathcal{A} \urcorner \subseteq \mathcal{C}$,
- $\mathcal{C} \cap \ulcorner \mathcal{B} \urcorner = \emptyset$,
- $\mathcal{C}$ a recursive subset of $\mathbb{N}$,
- $f(x) = \begin{cases} 1 & x \in \mathcal{C}, \\ 0 & x \notin \mathcal{C}, \end{cases}$
- $\widehat{f}$ define f.

Therefore we have

$$\begin{array}{rcl} X \in A & \Rightarrow & \widehat{f}\, \underline{\ulcorner X \urcorner} =_{\beta,\mathrm{w}} \underline{1} \\ X \in B & \Rightarrow & \widehat{f}\, \underline{\ulcorner X \urcorner} =_{\beta,\mathrm{w}} \underline{0} \end{array}$$

Let $\widehat{\circ}$, $\widehat{\mathbf{N}}$ define the functions $\circ$, $\mathbf{N}$, i.e.

$$\begin{array}{rcl} \widehat{\circ}\, \underline{\ulcorner M \urcorner}\, \underline{\ulcorner N \urcorner} & =_{\beta,\mathrm{w}} & \underline{\ulcorner M\ N \urcorner} \\ \widehat{\mathbf{N}}\, \underline{n} & =_{\beta,\mathrm{w}} & \underline{\ulcorner n \urcorner} \end{array}$$

Let $A \in \mathcal{A}$, $B \in \mathcal{B}$.
We will define a term J s. t.

$$J =_{\beta,\mathrm{w}} \mathtt{if}\ (\widehat{f}\, \underline{\ulcorner J \urcorner})\ \mathtt{then}\ A\ \mathtt{else}\ B \ .$$

Then we have

$$\begin{array}{rcccl} \widehat{f}\, \underline{\ulcorner J \urcorner} =_{\beta,\mathrm{w}} \underline{1} & \Rightarrow & J =_{\beta,\mathrm{w}} B & \Rightarrow & J \in \mathcal{B} \\ & \Rightarrow & \widehat{f}\, \underline{\ulcorner J \urcorner} =_{\beta,\mathrm{w}} \underline{0} & & \\ \widehat{f}\, \underline{\ulcorner J \urcorner} =_{\beta,\mathrm{w}} \underline{0} & \Rightarrow & J =_{\beta,\mathrm{w}} A & \Rightarrow & J \in \mathcal{A} \\ & \Rightarrow & \widehat{f}\, \underline{\ulcorner J \urcorner} =_{\beta,\mathrm{w}} \underline{1} & & \end{array}$$

Since $f(\underline{\ulcorner J \urcorner}) \in \{0, 1\}$,

$$\begin{array}{c} \widehat{f}\, \underline{\ulcorner J \urcorner} =_{\beta,\mathrm{w}} \underline{0} \vee \widehat{f}\, \underline{\ulcorner J \urcorner} =_{\beta,\mathrm{w}} \underline{1} \ , \\ \underline{0} =_{\beta,\mathrm{w}} \underline{1} \end{array}$$

contradicting that $\underline{0}$ and $\underline{1}$ are in normal form and not $\equiv$.
We have to define J. We would like to use the fixed point theorem 2.3.4 (a), but it cannot be applied, since the term on the right side depends on $\ulcorner J \urcorner$, not on J. Instead we prove the following fixed point lemma, from which with

$$X := \lambda y.\,\texttt{if}\ (\widehat{f}\ y)\ \texttt{then}\ A\ \texttt{else}\ B$$

the existence of J follows.

Lemma 2.5.6 *(Barendregt, 5.9.(c))*
For every (λ- or CL-)term X there exists a term J s. t.

$$J =_{\beta,\mathrm{w}} X\,\underline{\ulcorner J \urcorner}\ .$$

Proof:
Define

$$\begin{array}{rcl} M & := & \lambda y.X\,(\widehat{\circ}\ y\ (\widehat{\mathbf{N}}\ y)) \\ J & := & M\,\underline{\ulcorner M \urcorner} \end{array}$$

Then

$$\begin{array}{rcl} J & =_{\beta,\mathrm{w}} & X\,(\widehat{\circ}\,\underline{\ulcorner M \urcorner}\,(\widehat{\mathbf{N}}\,\underline{\ulcorner M \urcorner})) \\ & =_{\beta,\mathrm{w}} & X\,(\widehat{\circ}\,\underline{\ulcorner M \urcorner}\,\underline{\ulcorner \underline{\ulcorner M \urcorner} \urcorner}) \\ & =_{\beta,\mathrm{w}} & X\,\underline{\ulcorner M\,\underline{\ulcorner M \urcorner} \urcorner} \\ & \equiv & X\,\underline{\ulcorner J \urcorner}\ . \end{array}$$

Corollary 2.5.7 *(5.6.1)*
If $\mathcal{A}$ is a set of λ- or CL-terms closed under $=_{\beta,\mathrm{w}}$, and neither $\mathcal{A}$ nor its complement are nonempty, then $\mathcal{A}$ is non-recursive

Proof: Let $\mathcal{B}$ be the complement of $\mathcal{A}$.

Theorem 2.5.3 follows from Corollary 2.5.7 now as a corollary:

- If $=_{\beta,\mathrm{w}}$ were decidable, then for a term N the set $\mathcal{A} := \{M \mid M =_{\beta,\mathrm{w}} N\}$ would be a non-trivial recursive set closed under $=_{\beta,\mathrm{w}}$ contradicting Corollary 2.5.7.
- The set of terms with normal form is a non-trivial set closed under $=_{\beta,\mathrm{w}}$, therefore non-recursive.

2.6 The formal theories $\lambda\beta$ and $\mathrm{CL_w}$ (6A)

2.6.1 Definition of the theories (6A)

We will now introduce formal theories, which derive the relations $\longrightarrow_\beta$, $\longrightarrow_\mathrm{w}$, $=_\beta$, $=_\mathrm{w}$. These more abstract notions will allow us to define more easily extensions of these notions by e.g. extensional concepts.

Definition 2.6.1 (6.1)

(a) A *formal theory* $\mathcal{I}$ is a pair $(\mathcal{F}, \mathcal{R})$ s. t.

- $\mathcal{F}$ is a set, the elements of it are called *formulas*,
- $\mathcal{R}$ is a set of pairs $(\Gamma; A)$, where Γ is a set of formulas (i.e. $\Gamma \subseteq \mathcal{F}$) and A is a formula i.e. $A \in \mathcal{F}$)

As usual

- Γ, Δ denote in the following (possibly infinite) subsets of $\mathcal{F}$,
- $\Gamma, A := \Gamma \cup \{A\}$,
- $A_1, \ldots, A_n := \{A_1, \ldots, A_n\}$.

We will denote a rule $(A_1, , \ldots, A_n; A)$ by

$$\frac{A_1 \quad \cdots \quad A_n}{A}$$

(b) A rule $(\emptyset; F)$ of a formal theory is called an *axiom*, and will be denoted by F.
We usually define theories by defining the set of formulas, the set of axioms and then the set of rules which are not axioms.

(c) If $\mathcal{I} = (\mathcal{F}, \mathcal{R})$ we define for $F \in \mathcal{F}$, $\Delta \subseteq \mathcal{F}$ inductively

$$\mathcal{I}, \Delta \vdash F \text{ or shorter } \Delta \vdash_{\mathcal{I}} F \text{ or sometimes even shorter } \Delta \vdash F$$

by:

- $F \in \Delta \Rightarrow \Delta \vdash_{\mathcal{I}} F$.
- If
  - $(\Gamma; A) \in \mathcal{R}$,
  - for all $G \in \Gamma$ we have $\Delta \vdash_{\mathcal{I}} G$,

  then $\Delta \vdash_{\mathcal{I}} A$.

$$B \text{ if a theorem of } \mathcal{I} :\Leftrightarrow \mathcal{I} \vdash B :\Leftrightarrow \vdash_{\mathcal{I}} B :\Leftrightarrow \emptyset \vdash_{\mathcal{I}} B \ .$$

Definition 2.6.2 (6.2)
The *formal theory of β-equality* in short $\lambda\beta$ is defined as follows:

- Formulas:
  $$\text{Equations } M = N \text{ for } \lambda\text{-terms } M, N.$$
- Axioms: For all variables x, y and λ-terms M, N

$$\begin{array}{ll} (\alpha) & \lambda x.M = \lambda y.(M[x := y]) \quad \text{if } y \notin \mathrm{FV}(M) \\ (\beta) & (\lambda x.M)\ N = M[x := N] \\ (\rho) & M = M \end{array}$$

- Rules:

$$(\mu)\ \frac{M = M'}{N\ M = N\ M'} \qquad (\tau)\ \frac{M = N \quad N = P}{M = P}$$

$$(\nu)\ \frac{M = M'}{M\ N = M'\ N} \qquad (\sigma)\ \frac{M = N}{N = M}$$

$$(\xi)\ \frac{M = M'}{\lambda x.M = \lambda x.M'}$$

We write

$$\lambda\beta \vdash M = N$$

for $M = N$ provable in the above theory.

(The names (α), (β) , ... are from Curry and Feys, [CF58]).

Definition 2.6.3 (6.3)
The *formal theory of β-reduction* in short $\lambda\beta$ (from the context it will be clear whether Definition 2.6.2 or 2.6.3 is meant) is defined as before, but with $=$ replaced by $\longrightarrow$ and omitting the rule (σ). For convenience we spell it out:

- Formulas:

$$M \longrightarrow N, \text{ where } M,\ N \text{ are } \lambda\text{-terms .}$$

- Axioms: For all variables x, y and λ-terms M, N

$$\begin{array}{ll} (\alpha) & \lambda x.M \longrightarrow \lambda y.(M[x :\longrightarrow y]) \quad \text{if } y \notin \mathrm{FV}(M) \\ (\beta) & (\lambda x.M)\ N \longrightarrow M[x :\longrightarrow N] \\ (\rho) & M \longrightarrow M \end{array}$$

- Rules:

$$(\mu)\ \frac{M \longrightarrow M'}{N\ M \longrightarrow N\ M'} \qquad (\tau)\ \frac{M \longrightarrow N \quad N \longrightarrow P}{M \longrightarrow P}$$

$$(\nu)\ \frac{M \longrightarrow M'}{M\ N \longrightarrow M'\ N} \qquad (\xi)\ \frac{M \longrightarrow M'}{\lambda x.M \longrightarrow \lambda x.M'}$$

We write

$$\lambda\beta \vdash M \longrightarrow N$$

for $M \longrightarrow N$ provable in the above theory.

Lemma 2.6.4 *(6.4)*

(a) $M \longrightarrow_\beta^* N \Leftrightarrow \lambda\beta \vdash M \longrightarrow N$.

(b) $M =_\beta N \Leftrightarrow \lambda\beta \vdash M = N$.

Proof: Straightforward

Definition 2.6.5 (6.5)
The *formal theory of weak equality* in short $\mathrm{CL_w}$ is defined as follows:

- Formulas:

  Equations $M = N$ for CL-terms M, N.

- Axioms: For all CL-terms M, N

$$\begin{array}{ll} (\mathbf{k}) & \mathbf{k}\, M\, N = M. \\ (\mathbf{s}) & \mathbf{s}\, M\, N\, P = M\, P\, (N\, P). \\ (\rho) & M = M. \end{array}$$

- Rules:

$$\begin{array}{llll} (\mu) & \dfrac{M = M'}{N\, M = N\, M'} & (\tau) & \dfrac{M = N \quad N = P}{M = P} \\ (\nu) & \dfrac{M = M'}{M\, N = M'\, N} & (\sigma) & \dfrac{M = N}{N = M} \end{array}$$

  We write

$$\mathrm{CL_w}\vdash M = N$$

  for $M = N$ provable in the above theory.

Definition 2.6.6 (6.6)
The *formal theory of weak reduction* in short $\mathrm{CL_w}$ (again from the context it will be clear whether Definition 2.6.5 or 2.6.6 is meant) is defined as before, but with $=$ replaced by $\longrightarrow$ and omitting the rule (σ). For convenience we spell it out:

- Formulas:

  $M \longrightarrow N$, where M, N are λ-terms .

- Axioms: For all CL-terms M, N

$$\begin{array}{ll} (\mathbf{k}) & \mathbf{k}\, M\, N \longrightarrow M. \\ (\mathbf{s}) & \mathbf{s}\, M\, N\, P \longrightarrow M\, P\, (N\, P). \\ (\rho) & M \longrightarrow M. \end{array}$$

- Rules:

$$\begin{array}{llll} (\mu) & \dfrac{M \longrightarrow M'}{N\, M \longrightarrow N\, M'} & (\tau) & \dfrac{M \longrightarrow N \quad N \longrightarrow P}{M \longrightarrow P} \\ (\nu) & \dfrac{M \longrightarrow M'}{M\, N \longrightarrow M'\, N} & (\sigma) & \dfrac{M \longrightarrow N}{N \longrightarrow M} \end{array}$$

  We write

$$\mathrm{CL_w}\vdash M \longrightarrow N$$

  for $M \longrightarrow N$ provable in the above theory.

Lemma 2.6.7 *(6.7)*

(a) $M \longrightarrow^*_{\mathrm{w}} N \Leftrightarrow \mathrm{CL_w}\vdash M \longrightarrow N.$

(b) $M =_{\mathrm{w}} N \Leftrightarrow \mathrm{CL_w} \vdash M = N$.

Proof: Straightforward

Remark 2.6.8 *(6.8)*
By the Church-Rosser theorem and Lemmata 2.6.4 and 2.6.7 it follows that $\lambda\beta$ *and* $\mathrm{CL_w}$ *are consistent, i.e. not all formulas are provable.*

2.6.2 First order theories and derivable rules (6B)

Definition 2.6.9 (6.9)

(a) A first order theory is a pair $(\mathcal{L}, \mathcal{R})$, where

- $\mathcal{L}$ is a language, i.e. a collection of n-ary function and relation symbols;
  the set of terms and formulas is then defined as usual
  (w.r.t. the usual connectives of classical predicate calculus; $=$ is always included unless mentioned)
- $\mathcal{R}$ is a set of rules given as before w.r.t. the set of formulas just defined

Axioms are then defined as before, and derivability in this theory means derivability in the formal theory given by:

- Formulas are the formulas in the language $\mathcal{L}$.
- Rules are
  - The rules in $\mathcal{R}$.
  - The rules of classical predicate calculus with equality axioms are rules of the formal theory.

(b) We might change the underlying logic as well, where a logic consists of:

- A set of n-ary connectives
- A set of quantifiers.
- The set of formulas of a first order language $\mathcal{L}$ w.r.t the above connectives and quantifiers is then defined as:
  - Prime formulas of $\mathcal{L}$ in the usual sense are formulas.
  - If $A_1, \dots, A_n$ are formulas, $\circ$ an n-ary connective, then $\circ(A_1, \dots, A_n)$ is a formula
  - If A is a formula, Q a quantifier, x a variable, then $Qx.A$ is a formula
- A set of rules w.r.t. the above set of formulas.

In this case a first order theory consists of:

- A language $\mathcal{L}$ defined as before.
- A logic.
- Rules as before w.r.t. the set of formulas in $\mathcal{L}$ w.r.t. the language $\mathcal{L}$.

The *Formulas of a language $\mathcal{L}$ w.r.t. a logic* are the formulas of $\mathcal{L}$ w.r.t. the connectives and quantifiers of the logic.

Derivability etc. is now the straightforward generalization of part (a) of this definition.

There are lots of more generalizations possible (e.g. a sorted language, more generalized quantifiers).

Definition 2.6.10 (6.11)
The theory $\mathrm{CL}_{\mathrm{w}}^{+}$ is the theory in classical predicate calculus given by:

- Language: Two constants **k**, **s** and one binary function symbol Ap.
  We write $(M\ N)$ instead of $\mathrm{Ap}(M, N)$, and identify terms with their corresponding CL-terms.
  No relation symbols (except of equality).
- Logic: Classical logic with equality axioms.
- Axioms:
$$\begin{array}{l} \forall x.y.(\mathbf{k}\ x\ y = x) \\ \forall x.y, z.(\mathbf{s}\ x\ y\ z = x\ z\ (y\ z)) \\ \neg(\mathbf{s} = \mathbf{k}) \end{array}$$

Lemma 2.6.11 *(6.12)*
$\mathrm{CL}_{\mathrm{w}}^{+}$ *is a conservative extension of* CL_{w}*, i.e. provable equations of* $\mathrm{CL}_{\mathrm{w}}^{+}$ *and* CL_{w} *coincide.*

Proof:
One first verifies easily that every equation provable in CL_{w} is provable as well in $\mathrm{CL}_{\mathrm{w}}^{+}$.
On the other hand, we can see that the set of open CL-terms with

$$\mathrm{Ap}(N, M) := N\ M\ \ ,$$

k, s interpreted by themselves and equality defined as $=_{\mathrm{w}}$ is a model $\mathcal{M}$ of $\mathrm{CL}_{\mathrm{w}}^{+}$.
Further, we have that for every term N of the language of $\mathrm{CL}_{\mathrm{w}}^{+}$, $N^{*}[\xi]$ is the result of replacing x_i by $\xi(x_i)$ and of replacing $\mathrm{Ap}(N, M)$ by $N\ M$. Especially, if $\xi(x) = x$, $N^{*}[\xi]$ is the CL-term we identify N with. Therefore we get for CL-terms N, M, with N', M' being the corresponding terms in $\mathrm{CL}_{\mathrm{w}}^{+}$,

$$N'^{*}[\xi] \equiv N, \quad M'^{*}[\xi] \equiv M\ \ ,$$

and

$$\mathrm{CL}_{\mathrm{w}}^{+} \vdash N' = M'$$

$$\begin{array}{ll} \Rightarrow & \mathcal{M} \models (N' = M')[\xi] \\ \Leftrightarrow & {N'}^*[\xi] =_{\mathrm{w}} {M'}^*[\xi] \\ \Leftrightarrow & N =_{\mathrm{w}} M \\ \Leftrightarrow & \mathrm{CL_w} \vdash N = M \ . \end{array}$$

Definition 2.6.12 (6.13)

(a) A rule

$$\frac{A_1 \quad \cdots \quad A_n}{A}$$

of a formal theory is *derivable* in a theory $\mathcal{I}$, if

$$A_1, \ldots, A_n \vdash_{\mathcal{I}} A \ .$$

(b) A rule

$$\frac{A_1 \quad \cdots \quad A_n}{A}$$

of a formal theory is *admissible* in a theory $\mathcal{I}$, if

$$(\vdash_{\mathcal{I}} A_1 \wedge \cdots \vdash_{\mathcal{I}} A_n) \Rightarrow \vdash_{\mathcal{I}} A$$

Lemma 2.6.13 *(6.14)*

(a) A rule $\mathcal{R}$ of a formal theory is admissible iff adding of $\mathcal{R}$ to the theory does not change the set of theorems.

(b) Derivable rules of a formal theory are admissible, but not vice-verca.

(c) If $\mathcal{R}$ is a derivable rule in a formal theory $\mathcal{I}$, then $\mathcal{R}$ is derivable in any extension of $\mathcal{I}$ obtained by adding new rules.

Definition 2.6.14 (6.15)
Let $\mathcal{I}$, $\mathcal{I}'$ be formal theories with the same set of formulas.

(a) $\mathcal{I}$, $\mathcal{I}'$ are *theory-equivalent*, iff every rule of $\mathcal{I}$ is admissible in $\mathcal{I}'$ and vice verca.

(b) $\mathcal{I}$, $\mathcal{I}'$ are *rule-equivalent*, iff every rule of $\mathcal{I}$ is derivable in $\mathcal{I}'$ and vice verca.

Lemma 2.6.15 *(6.16)*
Two formal theories $\mathcal{I}$, $\mathcal{I}'$ with the same set of formulas are theorem equivalent iff they have the same set of theorems.

Proof: trivial

Definition 2.6.16 (6.17)
If $\mathcal{I}$ is a formal theory and let some formulas be equations $X = Y$, X, Y terms according to some definition (usually clear from the context). Then the *equality relation determined by* $\mathcal{I}$ is called $=_{\mathcal{I}}$ and defined by

$$X =_{\mathcal{I}} Y :\Leftrightarrow \vdash_{\mathcal{I}} X = Y \ .$$

Lemma 2.6.17 *(6.18)*
If $\mathcal{I}$, $\mathcal{I}'$ are formal theories with the same set of formulas and some equations be defined as in the previous definition. If $\mathcal{I}$, $\mathcal{I}'$ are theorem-equivalent, then $=_{\mathcal{I}}$ and $=_{\mathcal{I}'}$ coincide.

Proof: trivial.

2.7 Extensionality in λ-calculus (7)

2.7.1 Extensional equality

We usually treat equality as extensional: For two functions f, g

$$f = g \Leftrightarrow \forall x.(f(x) = g(x)) \ .$$

In $\lambda\beta$, equality is not extensional but intensional.

$$f = g \Leftrightarrow f, g \text{ reduce to the same normal form } \ .$$

For instance we have that y and $\lambda x.y\ x$ are extensional the same, but intensional different (they are both already in normal form).

Notation 2.7.1 (7.1)
In this section, term means λ-term.

Definition 2.7.2 (7.1, 7.2)

(a) The following are possible additional rules, which can be added to $\lambda\beta$

$$(\text{ext}) \quad \frac{M\ P = N\ P \quad (\text{for all terms } P)}{M = N}$$

$$(\omega) \quad \frac{M\ P = N\ P \quad (\text{for all closed terms } P)}{M = N}$$

$$(\zeta) \quad \frac{M\ x = N\ x}{M = N} \qquad \text{if } x \notin \text{FV}(M\ N)$$

$$(\eta) \quad \lambda x.M\ x = M \qquad \text{if } x \notin \text{FV}(M)$$

Note that the first two rules have infinitely many premises.

(b) The following 4 formal theories are extensions of $\lambda\beta$ by the following rules

$\lambda\beta + (\text{ext})$	add (ext);
$\lambda\beta\omega$	add (ω);
$\lambda\beta\zeta$	add (ζ);
$\lambda\beta\eta$	add (η);

We will focus on (ext), (ζ), (η). (ext) is admissible in (ω) but the converse does not hold See [HS86].

Theorem 2.7.3 *(7.4).*
$\lambda\beta+$(ext), $\lambda\beta\zeta$, $\lambda\beta\eta$ are rule-equivalent (therefore as well theorem-equivalent).

Proof:

- (ext) is trivially derivable in $\lambda\beta\zeta$.
- (ζ) is derivable in $\lambda\beta\eta$: If $M\ x = N\ x$, $x \notin \mathrm{FV}(M\ N)$, then

$$M = \lambda x.M\ x = \lambda x.N\ x = N\ \ .$$

- (η) is provable in $\lambda\beta$ + (ext): $(\lambda x.M\ \ x)P = M\ \ P$ for all P, therefore $\vdash_{\lambda\beta+(\mathrm{ext})}\lambda x.M\ x = M$.

Definition 2.7.4 (7.5)
$=_{\beta\eta}$ is the equality relation determined by $\lambda\beta\eta$:

$$M =_{\beta\eta} N \Leftrightarrow \vdash_{\lambda\beta\eta} M = N$$

2.7.2 $\lambda\beta\eta$-reduction (7B)

Definition 2.7.5 (7.6. - 7.9)

(a) $\longrightarrow_\eta$ is defined as $\longrightarrow_\beta$, but based on $\lambda x.M\ x \longrightarrow M$ instead of $(\lambda x.M)\ N \longrightarrow M[x := N]$. $\longrightarrow^*_\eta$, $\longrightarrow_{\eta,l}$ etc. are defined similarly

(b) $\longrightarrow^*_{\beta\eta}$ is the transitive closure of $\longrightarrow_\beta \cup \longrightarrow_\eta \cup \equiv_\alpha$. Similarly we define $\longrightarrow_{l\beta\eta}$.

(c) $\beta\eta$-normal forms are defined as β-normal forms, but w.r.t. $\longrightarrow^*_{\beta,\eta}$.

(d) The formal theory $\lambda\beta\eta$ of $\beta\eta$-reduction is the extension of the formal theory of β-reduction by

$$(\eta) \quad \lambda x.M\ x \longrightarrow M \quad \text{if } x \notin \mathrm{FV}(M\ N)$$

Lemma 2.7.6

$$P \longrightarrow^*_{\beta,\eta} Q \Leftrightarrow \lambda\beta\eta \vdash P \longrightarrow Q\ \ .$$

Lemma 2.7.7 *(7.11; Substitution lemma for $\beta\eta$-reduction) Assume $P \longrightarrow^*_{\beta\eta} Q$.*

(a) $\mathrm{FV}(Q) \subseteq \mathrm{FV}(P)$.

(b) $M[x := P] \longrightarrow^*_{\beta\eta} M[x := Q]$.

(c) $P[x := N] \longrightarrow^*_{\beta\eta} Q[x := N]$.

Proof: As usual, in (c) note, that If $P \equiv \lambda y.Q\ y$, $y \notin \mathrm{FV}(Q)$, then

$$(\lambda y.(Q\ y))[x := N] \longrightarrow_{\beta,\eta} Q[x := N]\ .$$

Theorem 2.7.8 *(7.12, Church-Rosser theorem for βη-reduction)*
If $P \longrightarrow_{\beta\eta} M$, $P \longrightarrow_{\beta\eta} N$, then there exists T such that $M \longrightarrow^*_{\beta\eta} T$, $N \longrightarrow^*_{\beta\eta} T$.

Proof: Extend Takahashis proof. Note that, whether we reduce $(\lambda x.N\ x)\ M$ by an η-contraction or a β-reduction we obtain the same result. This is reflected by the precise definition of the extension of the definition of M^*.

2.7.3 The postponent theorem (7.13 - 7.14)

We will prove the following two theorems.

Theorem 2.7.9 *(7.13) A λ-term has a βη-normal form iff it has a β-normal form.*

Theorem 2.7.10 *(Postponent-theorem, 7.14)*
If $M \longrightarrow^*_{\beta,\eta} N$ then there exists some P s. t. $M \longrightarrow^*_{\beta} P \longrightarrow^*_{\eta} N$.

We follow Takahashi [Tak95].

Definition 2.7.11 ([Tak95] 3.1)
The parallel η-reduction, denoted by $\Longrightarrow_\eta$ is defined inductively defined as

- $a \Longrightarrow_\eta a$, if a is an atom.
- If $v \notin \mathrm{Var}(M) \cup \mathrm{Var}(M')$, $M[x := v] \Longrightarrow_\eta M'[y := v]$, then $\lambda x.M \Longrightarrow_\eta \lambda y.M'$.
- If $M \Longrightarrow_\eta M'$, $N \Longrightarrow_\eta N'$, then $M\ N \Longrightarrow_\eta M'\ N'$.
- If $M \Longrightarrow_\eta M'$, $z \notin \mathrm{FV}(M)$, then $\lambda z.M\ z \Longrightarrow_\eta M'$.

Lemma 2.7.12 *(a) $M \Longrightarrow_\eta M$.*

(b) $M \Longrightarrow_\eta M'$, $v \notin \mathrm{Var}(M) \cup \mathrm{Var}(M')$, then $M[x := v] \Longrightarrow_\eta M'[x := v]$.

(c) If $M \equiv_\alpha M' \Longrightarrow_\eta N' \equiv_\alpha N$, then $M \Longrightarrow_\eta N$.

(d) If $M \Longrightarrow_\eta M'$, then $M[x := N] \Longrightarrow_\eta M'[x := N]$.

(e) If $M \longrightarrow_\eta M'$, then $M \Longrightarrow_\eta M'$.

(f) If $M \Longrightarrow_\eta M'$, then $M \longrightarrow^*_\eta M'$.

(g) If $M \Longrightarrow_\eta M'$, $N \Longrightarrow_\eta N'$, then $M[y := N] \Longrightarrow_\eta M'[y := N']$.

(h) $\longrightarrow^*_\eta$ *is the transitive closure of* $\Longrightarrow_\eta$.

Definition 2.7.13 (a) We define $M \rhd_k N$ (M is the k-fold η-expansion of N) by

- $M \rhd_0 M$.
- $M \rhd_{k+1} N$ iff $M \equiv \lambda z.M'\ z$ for some M' s. t. $z \notin \mathrm{FV}(M')$, $M' \rhd_k M$.

(b) $M \rhd N$ iff $M \rhd_k N$ for some k.

Lemma 2.7.14 *([Tak95] 3.2)*

(a) $M \Longrightarrow_\eta x$ *iff* $M \rhd x$.

(b) $M \Longrightarrow_\eta N_1\ N_2$ *iff* $M \rhd M_1\ M_2$ *for some* M_i *s. t.* $M_i \Longrightarrow_\eta N_i$.

(c) $M \Longrightarrow_\eta \lambda x.N$ *iff* $M \rhd (\lambda y.M')$ *for some* M' *s. t. for some* $z \notin \mathrm{Var}(M') \cup \mathrm{Var}(N)$ $M'[y := z] \Longrightarrow_\eta N[x := z]$.

Proof:
(a): "$\Rightarrow$" Induction on $M \Longrightarrow_\eta x$.
Case $M \equiv x$. Trivial.
Case $M \equiv \lambda z.M'\ z$, $z \notin \mathrm{FV}(M')$, $M' \Longrightarrow_\eta x$. By IH $M' \rhd_k x$ some k, $M \rhd_{k+1} x$.
"$\Leftarrow$" trivial.
(b), (c): similarly.

Lemma 2.7.15 *([Tak95] 3.3)*
Assume $M \Longrightarrow_\beta M'$, $N \Longrightarrow_\beta N'$, $k \geq 0$.

(a) If $M_k \rhd_k \lambda x.M$, *then* $M_k \Longrightarrow_\beta \lambda u.M'[x := u]$.

(b) If $M_k \rhd_k \lambda x.M$, *then* $M_k\ N \Longrightarrow_\beta M'[x := N']$.

(c) If $M_k \rhd_k M$, *then* $M_k\ N \Longrightarrow_\beta M'\ N'$.

Proof:
Proof by induction on k.
$k = 0$ is trivial.
$k \longrightarrow k + 1$:
(a), (b): Let

$$M_{k+1} \equiv \lambda z.M_k\ z\ ,$$

$$M_k \rhd_k \lambda x.M\ .$$

(a): By IH

$$M_k \Longrightarrow_\beta \lambda u.M'[x := u]\ ,$$

Therefore

$$M_k\ v \Rightarrow M'[x := v]\ ,$$

(v fresh), therefore

$$M_{k+1} = \lambda z.M_k\ z \Rightarrow \lambda u.M'[x := u]\ .$$

(b) By IH (c) $M_k\ u \Longrightarrow_\beta M'\ u$ for some fresh u,

$$N \Longrightarrow_\beta N'\ .$$

$$(M_k\ z)[z := u] \equiv M_k\ u \Longrightarrow_\beta M'\ u\ ,$$

therefore

$$M_{k+1}\ N \Longrightarrow_\beta (M'\ u)[u := N'] \equiv M'\ N'\ .$$

(c) Let

$$M_{k+1} = \lambda z.M_k\ z \quad M_k \rhd_k M\ , \quad z \notin \mathrm{FV}(M_k)\ .$$

By IH

$$M_k\ u \Longrightarrow_\beta M'\ u\ ,$$

therefore

$$M_{k+1}\ N \equiv (\lambda z.M_k\ z)\ N \Longrightarrow_\beta (M'\ u)[u := N'] \equiv M'\ N'\ .$$

Lemma 2.7.16 *([Tak95] 3.4)*

$$M \Longrightarrow_\eta P \Longrightarrow_\beta N$$

implies

$$M \Longrightarrow_\beta P' \Longrightarrow_\eta N$$

for some P'

Proof:
Induction on $P \Longrightarrow_\beta N$.
Case P an atom, $P \equiv N$. Trivial.
Case

$$P \equiv \lambda x.P_1, \quad N \equiv \lambda y.N_1, \quad P_1[x := u] \Longrightarrow_\beta N_1[y := u]\ .$$

Then $M \rhd \lambda z.M_1$ s. t.

$$M_1[z := v] \Longrightarrow_\eta P_1[x := v]\ .$$

W.l.o.g. $v \equiv u$. By IH

$$M_1[z := u] \Longrightarrow_\beta P_1' \Longrightarrow_\eta N_1[y := u]$$

for some P_1'.

$$M_1[z := u] \Longrightarrow_\beta P_1' \quad M \rhd \lambda z.M_1$$

$$M_1 \Longrightarrow_\beta P_1'[u := z]$$

therefore

$$M \Longrightarrow_\beta \lambda u.P_1' \Longrightarrow_\eta \lambda y.N_1 \equiv N$$

Case $P \equiv P_1\ P_2$, $N \equiv N_1\ N_2$, $P_i \Longrightarrow_\beta N_i$. Then

$$M \rhd M_1\ M_2$$

s. t. $M_i \Longrightarrow_\eta P_i$. By IH

$$M_i \Longrightarrow_\beta P_i' \Longrightarrow_\eta N_i\ \ .$$

therefore for some P_3 s. t. $P_3 \rhd P_1'\ P_2'$ we have

$$M \Longrightarrow_\beta P_3 \Longrightarrow_\eta P_1'\ P_2' \Longrightarrow_\eta N$$

Case $P \equiv (\lambda x.P_1)\ P_2$, $P_i \Longrightarrow_\beta N_i$, $N \equiv N_1[x := N_2]$.
Then

$$M \rhd_k M_1'\ M_2, \quad M_1' \rhd \lambda u.M_1$$

for some k, M_i s. t.

$$M_2 \Longrightarrow_\eta P_2 \quad M_1[u := v] \Longrightarrow_\eta P_1[x := v]\ \ .$$

By IH

$$M_2 \Longrightarrow_\beta P_2' \Longrightarrow_\eta N_2\ \ ,$$

$$M_1[u := v] \Longrightarrow_\beta P_1' \Longrightarrow_\eta N_1[x := v]\ \ .$$

Now

$$M_1'\ M_2 \Longrightarrow_\beta P_1'[v := P_2'] \Longrightarrow_\eta N_1[x := N_2]\ \ ,$$

and for some $P_3 \rhd_k P_1'[v := P_2']$ we get

$$M \Longrightarrow_\beta P_3 \Longrightarrow_\eta P_1'[v := P_2'] \Longrightarrow_\eta N_1[x := N_2]\ \ .$$

Proof of Theorem 2.7.10:
By Lemma 2.7.16, since $\Longrightarrow_\beta$, $\Longrightarrow_\eta$ are the transitive closures of $\longrightarrow_\beta$, $\longrightarrow_\eta$.

Lemma 2.7.17 *([Tak95] 3.6, 3.7)*
Assume $P \Longrightarrow_\eta Q$.

(a) If P is in β-normal form, so is Q.

(b) If Q has a β-normal form, so has P.

Proof:
In this proof normal form means β-normalform.
(a) Induction on P:
Case P is atom:
$$P \equiv Q \ .$$
Case $P \equiv \lambda x.P'$:
Subcase $Q \equiv \lambda x.Q'$, $P' \Longrightarrow_\eta Q'$. P' in normal form, by IH therefore Q', therefore Q as well.
Subcase
$$P' \equiv P''\ x, \quad x \notin \mathrm{FV}(P''), \quad P'' \Longrightarrow_\eta Q \ .$$
P'' is in normal form, by IH Q as well.
Case $P \equiv P_1\ P_2$. Then
$$Q \equiv Q_1\ Q_2, \quad P_i \Longrightarrow_\eta Q_i \ .$$
By IH Q_i in normal form, P_1 is no application therefore Q_1 neither, Q in normal form.
(b) By 2.7.16 it suffices to show the assertion for Q in normal form. We show:
$$\begin{array}{rcl} P \Longrightarrow_\eta Q, Q \text{ in normal form} & \Rightarrow & P \text{ has a normal form } P^* \text{ s. t.} \\ P \text{ not an application} & \Rightarrow & P^* \rhd N \text{ for some } N \text{ s. t.} \\ & & \quad N \text{ is not an application:} \end{array}$$
Induction on Q:
Case Q an atom: $P \equiv Q$.
Case $Q \equiv \lambda x.Q'$. Then
$$P \rhd \lambda x.P' \quad P' \Longrightarrow_\eta Q'$$
Q is in normalform, therefore as well Q', P' has normal form P'', $P' \Longrightarrow_\beta^* P''$, therefore
$$P \Longrightarrow_\beta^* \lambda x.P'' \ .$$
Case $Q \equiv Q_1\ Q_2$. Then Q_1 is not an application.
$$P \rhd P_1\ P_2, \quad P_i \Longrightarrow_\eta Q_i$$
By IH P_i have normal form P_i^*, $P_1^* \rhd N_1$ where N_1 not an application, and for some P', P'' s. t.
$$\begin{array}{rcl} P' & \rhd & P_1^*\ P_2^* \\ P'' & \rhd & N\ P_2 \end{array}$$
we get
$$P \Longrightarrow_\beta^* P' \Longrightarrow_\beta^* P'' \ ,$$
P'' in normal form.

Proof of Theorem 2.7.9.
Assume N has $\beta\eta$-normal form M. Then

$$N \Longrightarrow_{\beta}^{*} N' \Longrightarrow_{\eta}^{*} M \ .$$

M is in β-normal form, so N' has β-normal form.
Assume N has β-normal form N'. Since in every η-step, the length of a term is reduced, there is a N'' s. t. N'' has no η-redex and

$$N' \Longrightarrow_{\eta}^{*} N'' \ .$$

N' is in β-normal form, by Lemma 2.7.17 (a) as well N'', therefore N'' is in $\beta\eta$ normal form.

2.8 Extensionality in combinatory logic

2.8.1 Extensional equality

Notation 2.8.1 (8.1)
In this section, "term" means "CL-term".

Definition 2.8.2 (8.1, 8.2)

(a) The following are possible additional rules, which can be added to $\mathrm{CL_w}$

$$\text{(ext)}\quad \frac{M\ P = N\ P \quad \text{(for all terms } P\text{)}}{M = N}$$

$$(\zeta)\quad \frac{M\ x = N\ x}{M = N} \qquad \text{if } x \notin \mathrm{FV}(M\ N)$$

$$(\xi)\quad \frac{M = N}{\lambda^* x.M = \lambda^* x.N}$$

$$(\eta)\quad \lambda^* x.M\ x = M \qquad \text{if } x \notin \mathrm{FV}(M)$$

((ω) will not be treated here;
(η) allows only to derive new equations, if we replace the definition of λ^* by a different definition (see below) since with our definition for $x \notin \mathrm{FV}(M)$

$$\lambda^* x.M\ x \equiv M \ ,$$

Note that the first two rules have infinitely many premises.

(b) The following 4 formal theories are extensions of $\mathrm{CL_w}$ by the following rules

$\mathrm{CL} + \text{(ext)}$	add (ext);
CL_ζ	add (ζ);
CL_ξ	add (ξ);

$\mathrm{CL_w}$ extended by (η) alone will not be treated because that is identical with $\mathrm{CL_w}$.

Theorem 2.8.3 *(8.4, 8.7).*

(a) $\mathrm{CL} + (\mathrm{ext})$ *and* CL_ζ *are theorem-equivalent.*

(b) CL_ζ *and* CL_ξ *are rule-equivalent and therefore as well theorem-equivalent.*

Remark $\mathrm{CL} + (\mathrm{ext})$ and CL_ζ are not rule-equivalent. (Consider e.g. $\mathbf{k}\ x = \mathbf{s}\ x$. From it it is probably not possible to prove in $\mathrm{CL} + (\mathrm{ext})$ that $\mathbf{k} = \mathbf{s}$, but in CL_ζ this is an instance of a rule.
However to show

$$\neg(\mathbf{k}\ x = \mathbf{s}\ x \vdash_{\mathrm{CL}+(\mathrm{ext})} \mathbf{k} = \mathbf{s})$$

is probably not easy.
Proof of Theorem 2.8.3:

- (ext) is trivially derivable in CL_ζ.
- (ζ) is admissible in $\mathrm{CL} + (\mathrm{ext})$:
  Show that, if $\mathrm{CL} + (\mathrm{ext}) \vdash M = N$, then

$$\begin{aligned} \mathrm{CL} + (\mathrm{ext}) \quad \vdash \quad & M[x_1 := K_1, \ldots , x_n := K_n] = \\ & N[x_1 := K_1, \ldots , x_n := K_n] \end{aligned}$$

  by induction on the derivation.
  We write $[\vec{x} := \vec{K}]$ for

$$[x_1 := K_1, \ldots , x_n := K_n] \ .$$

  - If the last rule is a rule in CL this is clear.
  - Case last rule (ext): Assume

$$M\ P = N\ P \text{ for all terms } P$$

    If we naively apply the IH to the premise of that rule we get

$$M[\vec{x} := \vec{K}]\ P[\vec{x} := \vec{K}] = N[\vec{x} := \vec{K}]\ P[\vec{x} := \vec{K}]$$

    for all P which does not suffice to prove by using (ext).

$$M[\vec{x} := \vec{K}] = N[\vec{x} := \vec{K}] \ .$$

    So we have to rename the variables in P appropriately.
    Let P be a term. Let x_i^* be distinct variables s. t.

$$x_i^* \notin \mathrm{FV}(M\ N\ P\ K_1 \cdots K_n\ x_1 \cdots x_n)$$

$$P^* := P[x_1 := x_1^*, \ldots , x_n := x_n^*] \ .$$

    Write $[\vec{x} := \vec{x}^*]$ for $[x := x_1^*, \ldots , x_n := x_n^*]$. Then

$$P^*[\vec{x} := \vec{K}, \vec{x}^* := \vec{x}] \quad \equiv \quad P \ ,$$

$$\begin{array}{rcl} M[\vec{x} := \vec{K}, \vec{x}^* := \vec{x}] & \equiv & M[\vec{x} := \vec{K}] \ , \\ N[\vec{x} := \vec{K}, \vec{x}^* := \vec{x}] & \equiv & N[\vec{x} := \vec{K}] \ , \end{array}$$

We have

$$\mathrm{CL} + (\mathrm{ext}) \vdash M\ P^* = N\ P^*$$

therefore by IH

$$\begin{array}{rcl} \mathrm{CL} + (\mathrm{ext}) & \vdash & (M\ P^*)[\vec{x} := \vec{K}, \vec{x}^* := \vec{x}] \\ & & = (N\ P^*)[\vec{x} := \vec{K}, \vec{x}^* := \vec{x}] \ , \end{array}$$

therefore

$$\mathrm{CL} + (\mathrm{ext}) \vdash M[\vec{x} := \vec{K}]\ P = N[\vec{x} := \vec{K}]\ P$$

and, since P was arbitrary, therfore by (ext)

$$\mathrm{CL} + (\mathrm{ext}) \vdash M[\vec{x} := \vec{K}] = N[\vec{x} := \vec{K}]$$

Now if $\mathrm{CL} + (\mathrm{ext})$ proves $M\ x = N\ x$, $x \notin \mathrm{FV}(M\ N)$, then it proves (substitute for x P)

$$M\ P = N\ P$$

for all P and therefore $M = N$, (ζ) is admissible in $\mathrm{CL} + (\mathrm{ext})$.

– (ζ) is derivable in (ξ):
Assume $M\ x = N\ x$, $x \notin \mathrm{FV}(M\ N)$, and show in CL_ξ $M = N$:
By (ξ)

$$\lambda^* x.M\ x = \lambda^* x.N\ x \ ,$$

by definition

$$\begin{array}{rcl} M & \equiv & \lambda^* x.M\ x \\ N & \equiv & \lambda^* x.N\ x \end{array}$$

therefore $M = N$.

– (ξ) is derivable in (ζ):
Assume $M = N$, and show in CL_ζ $\lambda x^* M = \lambda x^*.N$:
Now

$$x \notin \mathrm{FV}(\lambda^* x.M)(\lambda^* x.N) \ ,$$

and

$$(\lambda^* x.M)\ x = M = N = (\lambda^* x.N)\ x$$

by ζ therefore

$$\lambda^* x.M = \lambda^* x.N \ .$$

Definition 2.8.4 (8.5)
$=_{\mathrm{c}\beta\eta}$ is the equality relation determined by $\mathrm{CL} + (\mathrm{ext})$:

$$M =_{\mathrm{c}\beta\eta} N \Leftrightarrow \vdash_{\mathrm{CL}+(\mathrm{ext})} M = N$$

Remark 2.8.5 *(8.6, 8.8)*

(a) *In combinatory logic we have trivially η and the rule which makes combinatory logic extensional is the ξ-rule.*

(b) *If we defined $\lambda^* x.M$ by not having the special case $\lambda^* x.M\ x := M$ if $x \notin \mathrm{FV}(M)$, then we would get Theorem 2.8.3 (a), but in 2.8.3 (b) only: $\mathrm{CL} + (\zeta)$ and $\mathrm{CL} + (\xi) + (\eta)$ are rule-equivalent.*

2.8.2 An axiomatisation of extensionality by finitely many equations

(ζ) and (ξ) are difficult to handle:

- For verifying extensional equality using the (ζ)-rule, we need to need to test whether some subterm P of a term M, applied to a fresh variable x, is equal to another term of the form $P'\ x$, in order to replace now using (ζ) P by P', where the equation $P\ x =_{\mathrm{c},\zeta} P'\ x$ might have a long derivation, so might not follow by only a one step reduction or expansion.
- For verifying extensional equality using the (ξ)-rule, we need to test, whether there is a subterm of a given term which is of the form $\lambda^* x.P$, in order to replace it by $\lambda^* x.P'$ for some other term P' which is equal to P. This is decidable, but still complicated to check.

We will in the following develop an axiomatization of extensional equality in the form of finitely many equations of the form $M = N$. Then, one needs to verify only, whether a subterm of a given term is of the form of the left or right side of one of these equations in order to replace it by the other side. However, unless we have for a corresponding reduction we have Church-Rosser this will not provide us with an easy procedure for determining whether two terms are equal – we might need to expand a term first to apply one of the equations. The book does not treat how to derive a reduction system from these axioms, which is Church Rosser. (However the extension of the $\longrightarrow_{\mathrm{w}}$ by the (ξ) is Church Rosser, see [HS86], 9.16). However, it seems that the best way of checking equality of combinator terms is by translating them into λ-terms and test whether they are $\beta\eta$-equal. The interest of the following axioms is more theoretical. We will derive this axiomatization in 3 steps. In each step we will consider extensions of CL by finitely many equations as axioms.

Definition 2.8.6 A finite extension $\mathrm{CL}_{\mathrm{ax}}$ of CL is the extension of CL by finitely many equations $M_i = N_i$ as axioms.

Step 1: We have

$$\lambda^* x.M\ N \equiv \mathbf{s}\ (\lambda^* x.M)\ (\lambda^* x.N)$$

in case $x \in \mathrm{FV}(M\ N)$, $N \not\equiv x \vee x \in \mathrm{FV}(N)$. If we replace λ^* by λ, the above equation holds for λ-terms with $\equiv$ replaced by $=$.

We will later show that extensional λ-calculus and combinatory logic prove the same equations w.r.t. this translation, so we want that in our proposed finite extension $\mathrm{CL_{ax}}$ of CL

$$\mathrm{CL_{ax}} \vdash \lambda^* x.M\ N = \mathbf{s}\ (\lambda^* x.M)\ (\lambda^* x.N)\ ,$$

which would then reduce proofs by induction over the form of $\lambda^* x.N$ only to cases $N \equiv P\ Q$ and N an atom only.

Lemma 2.8.7 *(8.10)*
Assume $\mathrm{CL_{ax}}$ *is a finite extension of* CL. *Assume* $\mathrm{CL_{ax}}$ *proves*

$$\begin{array}{lrcl} (\mathrm{Ax1}) & \lambda^* x, y.\mathbf{s}\ (\mathbf{k}\ x)\ (\mathbf{k}\ y) & = & \lambda^* x, y.\mathbf{k}\ (x\ y) \\ (\mathrm{Ax2}) & \lambda^* x.\mathbf{s}\ (\mathbf{k}\ x)\ \mathbf{I} & = & \lambda^* x.x \end{array}$$

Then for all M, N, x

$$\mathrm{CL_{ax}} \vdash \lambda^* x.M\ N = \mathbf{s}\ (\lambda^* x.M)\ \ (\lambda^* x.N)\ .$$

Proof:
If $x \in \mathrm{FV}(M\ N)$ and $x \in \mathrm{FV}(M) \vee N \not\equiv x$,

$$\lambda^* x.M\ N \equiv \mathbf{s}\ (\lambda^* x.M)\ \ (\lambda^* x.N)\ .$$

In all other cases we will look at the above equation from left to right.
Case $x \notin \mathrm{FV}(M\ N)$:

$$\begin{array}{rcl} \mathbf{s}\ (\lambda^* x.M)\ (\lambda^* x.N) & \equiv & \mathbf{s}\ (\mathbf{k}\ M)\ (\mathbf{k}\ N) \\ & =_{\mathrm{w}} & (\lambda^* x, y.\mathbf{s}\ (\mathbf{k}\ x)\ (\mathbf{k}\ y))\ M\ N \\ & \overset{\mathrm{Ax}\ 1}{\equiv} & (\lambda^* x, y.\mathbf{k}\ (x\ y))\ M\ N \\ & =_{\mathrm{w}} & \mathbf{k}\ (M\ N) \\ & \equiv & \lambda^* x.M\ N \end{array}$$

Case $x \notin \mathrm{FV}(M)$, $N \equiv x$.

$$\begin{array}{rcl} \mathbf{s}\ (\lambda^* x.M)\ (\lambda^* x.N) & \equiv & \mathbf{s}\ (\mathbf{k}\ M)\ \mathbf{I} \\ & =_{\mathrm{w}} & (\lambda^* x.\mathbf{s}\ (\mathbf{k}\ x)\ \mathbf{I})\ M \\ & \overset{\mathrm{Ax}\ 2}{\equiv} & (\lambda^* x.x)\ M \\ & =_{\mathrm{w}} & M \\ & \equiv & \lambda^* x.M\ N \end{array}$$

Step 2 We want a finite extension $\mathrm{CL_{ax}}$ of CL s. t.

$$\mathrm{CL_{ax}} \vdash M = N \Rightarrow \mathrm{CL_{ax}} \vdash \lambda^* x.M = \lambda^* x.N\ .$$

If we have (Ax1), (Ax2) above fulfilled, the case where M, N is not an outermost redex reduces to subterms of M, N. The difficult case is when M, is a redex, i.e. we need to find a finite extension $\mathrm{CL_{ax}}$ s. t.

$$\begin{array}{lll} (1) & \mathrm{CL_{ax}} & \vdash & \lambda^* x.\mathbf{k}\ M\ N = \lambda^* x.M \\ (2) & \mathrm{CL_{ax}} & \vdash & \lambda^* x.\mathbf{s}\ M\ N\ P = \lambda^* x.(M\ P)\ (N\ P) \end{array}$$

Now, if $\mathrm{CL}_{\mathrm{ax}}$ proves (Ax1) (Ax2), then the left and right side of (1), (2) are, with

$$\begin{array}{rcl} M^* & := & \lambda^* x.M \\ N^* & := & \lambda^* x.N \\ P^* & := & \lambda^* x.P \end{array}$$

provable in $\mathrm{CL}_{\mathrm{ax}}$ equal to

$$\begin{array}{lrcl}
(1a) & \lambda^* x.\mathbf{k}\ M\ N & \overset{\text{Lemma 2.8.7}}{\equiv} & \mathbf{s}\ (\lambda^* x.\mathbf{k}\ M)\ N^* \\
 & & \overset{\text{Lemma 2.8.7}}{\equiv} & \mathbf{s}\ (\mathbf{s}\ (\mathbf{k}\ \mathbf{k})\ M^*)\ N^* \\
(1b) & \lambda^* x.M & \equiv & M^* \\
(2a) & \lambda^* x.\mathbf{s}\ M\ N\ P & \overset{\text{Lemma 2.8.7}}{\equiv} & \mathbf{s}\ (\lambda^* x.\mathbf{s}\ M\ N)\ P^* \\
 & & \overset{\text{Lemma 2.8.7}}{\equiv} & \mathbf{s}\ (\mathbf{s}\ (\lambda^* x.\mathbf{s}\ M)\ N^*)\ P^* \\
 & & \overset{\text{Lemma 2.8.7}}{\equiv} & \mathbf{s}\ (\mathbf{s}\ (\mathbf{s}\ (\mathbf{k}\ \mathbf{s})\ M^*)\ N^*)\ P^* \\
(2b) & \lambda^* x.M\ P\ (N\ P) & \overset{\text{Lemma 2.8.7}}{\equiv} & \mathbf{s}\ (\lambda^* x.M\ P)\ (\lambda^* x.N\ P) \\
 & & \overset{\text{Lemma 2.8.7}}{\equiv} & \mathbf{s}\ (\mathbf{s}\ M^*\ P^*)\ (\mathbf{s}\ N^*\ P^*)
\end{array}$$

Therefore, if $\mathrm{CL}_{\mathrm{ax}}$ proves that the righthandsides of (1a), (1b) and of (2a), (2b) are equal, then (1), (2) follow. Now we just axiomatize this condition and get the following Lemma

Lemma 2.8.8 *(8.11)*
Assume $\mathrm{CL}_{\mathrm{ax}}$ *is a finite extension of* CL. *Assume* $\mathrm{CL}_{\mathrm{ax}}$ *proves* (Ax1), (Ax2) *from Lemma 2.8.8 and*

$$\begin{array}{ll} (\mathrm{Ax3}) & \lambda^* x, y.\mathbf{s}\ (\mathbf{s}\ (\mathbf{k}\ \mathbf{k})\ x)\ y = \lambda^* x, y.x \\ (\mathrm{Ax4}) & \lambda^* x, y, z.\mathbf{s}\ (\mathbf{s}\ (\mathbf{s}\ (\mathbf{k}\ \mathbf{s})\ x)\ y)\ z = \\ & \quad \lambda^* x, y, z.\mathbf{s}\ (\mathbf{s}\ x\ z)\ (\mathbf{s}\ y\ z) \end{array}$$

Then for all M, N, P, x

$$\begin{array}{llll} (1) & \mathrm{CL}_{\mathrm{ax}} & \vdash & \lambda^* x.\mathbf{k}\ M\ N = \lambda^* x.M \\ (2) & \mathrm{CL}_{\mathrm{ax}} & \vdash & \lambda^* x.\mathbf{s}\ M\ N\ P = \lambda^* x.(M\ P)\ (N\ P) \end{array}$$

Proof: By the above and

$$(\lambda^* x.Q)\ R = Q[x := R]\ .$$

Definition 2.8.9 (8.12)

(a) The *combinatory* $\beta\eta$*-axioms* are

$$\begin{array}{ll} (\beta - \mathrm{ax1}) & \lambda^* x, y.\mathbf{s}\ (\mathbf{k}\ x)\ (\mathbf{k}\ y) = \lambda^* x, y.\mathbf{k}\ (x\ y) \\ (\beta - \mathrm{ax2}) & \lambda^* x.\mathbf{s}\ (\mathbf{k}\ x)\ \mathbf{I} = \lambda^* x.x \\ (\beta - \mathrm{ax3}) & \lambda^* x, y.\mathbf{s}\ (\mathbf{s}\ (\mathbf{k}\ \mathbf{k})\ x)\ y = \lambda^* x, y.x \\ (\beta - \mathrm{ax4}) & \lambda^* x, y, z.\mathbf{s}\ (\mathbf{s}\ (\mathbf{s}\ (\mathbf{k}\ \mathbf{s})\ x)\ y)\ z = \\ & \quad \lambda^* x, y, z.\mathbf{s}\ (\mathbf{s}\ x\ z)\ (\mathbf{s}\ y\ z) \end{array}$$

Note that this axioms expand to purely combinatorial equations, which do not contain λ^* (especially they are closed).

(b) $\mathrm{CL}_{\beta\eta\mathrm{ax}}$ is the extension of CL by the above axioms.

Theorem 2.8.10 *(8.13)* $\mathrm{CL}_{\beta\eta\mathrm{ax}}$ *is theorem equivalent to* CL_ξ, CL_ζ *and* $\mathrm{CL} + \mathrm{ext}$.
Especially, the equality determined by $\mathrm{CL}_{\beta\eta\mathrm{ax}}$ *is* $=_{\mathrm{c}\beta\eta}$.

Proof:
We show $\mathrm{CL}_{\beta\eta\mathrm{ax}}$ is theorem equivalent to CL_ξ.
First all additional axioms of $\mathrm{CL}_{\beta\eta\mathrm{ax}}$ are provable in CL_ξ: Any of the additional axioms is of the form

$$\lambda^* x_1, \ldots, x_n.M = \lambda^* x_1, \ldots, x_n.N \ .$$

By ξ it can be derived in CL_ξ, if

$$M = N$$

can be derived.
Now M and N are in all cases w.r.t. $\longrightarrow_{\mathrm{w}}$ irreducible terms. However, we have

$$M \equiv \lambda^* u.M\ u \quad N \equiv \lambda^* u.N\ u$$

for some new variable u, and by ξ it suffices to show

$$M\ u = N\ u \ .$$

Now in all cases $M\ u$ and $N\ u$ reduce with $\longrightarrow_{\mathrm{w}}$ to the same normalform, therefore $M\ u = N\ u$ is provable in CL and the axiom is provable in CL_ξ.
Verification of this:

$$\begin{array}{llcl}
(\beta - \mathrm{ax1}) & \mathbf{s}\ (\mathbf{k}\ x)\ (\mathbf{k}\ y)\ u & = & \mathbf{k}\ x\ u\ (\mathbf{k}\ y\ u) \\
 & & = & x\ y \\
 & & = & \mathbf{k}\ (x\ y)\ u \\
 & & & \\
(\beta - \mathrm{ax2}) & \mathbf{s}\ (\mathbf{k}\ x)\ \mathbf{I}\ u & = & \mathbf{k}\ x\ u\ (\mathbf{I}\ u) \\
 & & = & x\ u \\
 & & & \\
(\beta - \mathrm{ax3}) & \mathbf{s}\ (\mathbf{s}\ (\mathbf{k}\ \mathbf{k})\ x)\ y\ u & = & \mathbf{s}\ (\mathbf{k}\ \mathbf{k})\ x\ u\ (y\ u) \\
 & & = & \mathbf{k}\ \mathbf{k}\ u\ (x\ u)\ (y\ u) \\
 & & = & \mathbf{k}\ (x\ u)\ (y\ u) \\
 & & = & x\ u \\
 & & & \\
(\beta - \mathrm{ax4}) & \mathbf{s}\ (\mathbf{s}\ (\mathbf{s}\ (\mathbf{k}\ \mathbf{s})\ x)\ y)\ z\ u & = & \mathbf{s}\ (\mathbf{s}\ (\mathbf{k}\ \mathbf{s})\ x)\ y\ u\ (z\ u) \\
 & & = & \mathbf{s}\ (\mathbf{k}\ \mathbf{s})\ x\ u\ (y\ u)\ (z\ u) \\
 & & = & \mathbf{k}\ \mathbf{s}\ u\ (x\ u)\ (y\ u)\ (z\ u) \\
 & & = & \mathbf{s}\ (x\ u)\ (y\ u)\ (z\ u) \\
 & & = & x\ u\ (z\ u)\ ((y\ u)\ (z\ u)) \\
 & & = & \mathbf{s}\ x\ z\ u\ (\mathbf{s}\ y\ z\ u) \\
 & & = & \mathbf{s}\ (\mathbf{s}\ x\ z)\ (\mathbf{s}\ y\ z)\ u \ .
\end{array}$$

For the other direction we show that (ξ) is admissible in $\mathrm{CL}_{\beta\eta\mathrm{ax}}$, i.e.

$$\mathrm{CL}_{\beta\eta\mathrm{ax}} \vdash M = M' \Rightarrow \mathrm{CL}_{\beta\eta\mathrm{ax}} \vdash \lambda^* x.M = \lambda^* x.M'$$

by induction on the derivation of

$$\mathrm{CL}_{\beta\eta\mathrm{ax}} \vdash M = M' \ .$$

Case: axiom ($\mathbf{k}$) or ($\mathbf{s}$): Lemma 2.8.8.
Case: (ρ), i.e. $M \equiv M'$: trivial.
Case: transitivity (τ) or symmetry (σ): by IH, (τ), (σ).
Case: (μ), (ν). So assume

$$\begin{array}{rcl} M & \equiv & N\,Q \\ M' & \equiv & N'\,Q' \\ \mathrm{CL}_{\beta\eta\mathrm{ax}} & \vdash & N = N' \\ \mathrm{CL}_{\beta\eta\mathrm{ax}} & \vdash & Q = Q' \end{array}$$

(where in fact either $N \equiv N'$ or $Q \equiv Q'$). Then by IH (or the rule (ρ))

$$\begin{array}{rcl} \mathrm{CL}_{\beta\eta\mathrm{ax}} & \vdash & \lambda^* x.N = \lambda^* x.N' \\ \mathrm{CL}_{\beta\eta\mathrm{ax}} & \vdash & \lambda^* x.Q = \lambda^* x.Q' \end{array}$$

By Lemma 2.8.7 $\mathrm{CL}_{\beta\eta\mathrm{ax}}$ proves

$$\begin{array}{rcl} \lambda^* x.N\,Q & = & \mathbf{s}\,(\lambda^* x.N)\,(\lambda^* x.Q) \\ & = & \mathbf{s}\,(\lambda^* x.N')\,(\lambda^* x.Q') \\ & = & \lambda^* x.N'\,Q' \ . \end{array}$$

2.8.3 $\beta\eta$-strong reduction

Definition 2.8.11 (8.16)

(a) The formal theory of *$\beta\eta$-strong reduction* is obtained from the theory CL_{w} of weak reduction by replacing $\longrightarrow_{\mathrm{w}}$ by $\longrightarrow_{\mathrm{s}}$ and adding the rule

$$(\xi) \quad \frac{M \longrightarrow_{\mathrm{s}} N}{\lambda^* x.M \longrightarrow_{\mathrm{s}} \lambda^* x.N} \ .$$

(b)

$$M \longrightarrow_{\mathrm{s}} N :\Leftrightarrow \mathrm{CL}_{\mathrm{w}} \vdash M \longrightarrow_{\mathrm{s}} N \ .$$

Theorem 2.8.12 *(8.17)*

(a) $\longrightarrow_{\mathrm{s}}$ *is transitive and reflexive.*

(b) $P \longrightarrow_{\mathrm{s}} Q \Rightarrow \mathrm{FV}(Q) \subseteq \mathrm{FV}(P)$.

(c) $P \longrightarrow_{\mathrm{s}} Q \Rightarrow M[x := P] \longrightarrow_{\mathrm{s}} M[x := Q]$.

(d) $P \longrightarrow_{\mathrm{s}} Q \Rightarrow P[x := N] \longrightarrow_{\mathrm{s}} Q[x := N]$.

(e) *(Church-Rosser)*
If $P \longrightarrow_{\mathrm{s}} M$, $P \longrightarrow_{\mathrm{s}} N$, then there exists T such that $M \longrightarrow_{\mathrm{s}}^{*} T$, $N \longrightarrow_{\mathrm{s}}^{*} T$.

Proof: (a) - (d) are easy. (e) can be proved in the next section, by translating combinatory logic into λ-calculus, using Church-Rosser there and then translating it back again.

Remark: $\longrightarrow_{\mathrm{s}}$ does not behave very nicely: $\mathbf{I}$ is not in normal form: We have

$$\begin{array}{rcl} \mathbf{s}\,\mathbf{k} & \equiv & \lambda^{*}x, y.\mathbf{s}\,\mathbf{k}\,x\,y \\ & \longrightarrow_{\mathrm{s}}^{*} & \lambda^{*}x, y.\mathbf{k}\,y(x\,y) \\ & \longrightarrow_{\mathrm{s}}^{*} & \lambda^{*}x, y.y \\ & \equiv & \mathbf{k}\,\mathbf{I} \end{array}$$

therefore

$$\mathbf{I} \equiv \mathbf{s}\,\mathbf{k}\,\mathbf{k} \longrightarrow_{\mathrm{s}}^{*} \mathbf{k}\,\mathbf{I}\,\mathbf{k} \ ,$$

$$\mathbf{I} \longrightarrow_{\mathrm{s}}^{*} \mathbf{k}\,\mathbf{I}\,\mathbf{k} \longrightarrow_{\mathrm{s}}^{*} \mathbf{k}\,(\mathbf{k}\,\mathbf{I}\,\mathbf{k})\,\mathbf{k} \longrightarrow_{\mathrm{s}}^{*} \mathbf{k}\,(\mathbf{k}\,(\mathbf{k}\,\mathbf{I}\,\mathbf{k})\,\mathbf{k})\,\mathbf{k} \longrightarrow_{\mathrm{s}}^{*} \cdots$$

In order to get reduction which behaves better, one can add $\mathbf{I}$ as a constant (not a defined combinator) together with the reduction rule

$$\mathbf{I}\,x \longrightarrow_{\mathrm{s}} x \ .$$

And when adding axioms $\mathbf{I}\,x = x$, $\mathbf{I}\,x \longrightarrow_{\mathrm{s}} x$, we get the same lemmata as before. But still, $\longrightarrow_{\mathrm{s}}$ does not behave well, especially, when a term is in normal form is relatively complicated.

Definition 2.8.13 (3.7) *Strong normal forms.* The class of *strong nomral forms* or *terms in strong normal form* is inductively defined by

- If $n \geq 0$, $M_1, \ldots, M_n$ are in strong normal form, a an atom which is not $\mathbf{s}$, $\mathbf{k}$, then
$$a\ M_1 \cdots M_n \text{ is in strong normal form.}$$
- If M is in strong normal form so is $\lambda^{*}x.M$.

Definition 2.8.14 (8.19) A CL-term M *has a strong normal form* M^{*} iff M^{*} is in strong normal form and
$$M \longrightarrow_{\mathrm{s}} M^{*}$$

Lemma 2.8.15 *(a) A term M has at most one strong normal form.*

(b) *If M^* is in strong normal form, $N =_{c\beta\eta} M^*$, then M^* is the strong normal form of N.*

(c) *M^* is the strong normal from of M iff M^* is in strong normal form and*

$$M =_{c\beta\eta} M^*$$

Proof: (c) follows from (b). If we had $\mathbf{I}$ is an atom as above, every strong normal form would be irreducible and the lemma follows. Without it, one needs to go via translation into λ-calculus.

2.9 The correspondence between λ and CL (9)

2.9.1 The extensional equalities (9A, 9B)

Definition 2.9.1 (9.2)
For CL-terms M, we define its λ-*transformation* M^λ by

$$\begin{array}{rcl} x^\lambda & := & x \ , \\ \mathbf{k}^\lambda & := & \lambda x, y.x \\ \mathbf{s}^\lambda & := & \lambda x, y, z.x\ z\ (y\ z) \ , \\ (MN)^\lambda & := & M^\lambda\ N^\lambda \end{array}$$

Remark 2.9.2 *(9.3, 9.13)*

(a) *$M \mapsto M^\lambda$ is injective (modulo $\equiv$), we even have*

$$M \not\equiv N \Rightarrow M^\lambda \not\equiv_\alpha N^\lambda \ .$$

(b) *$\mathrm{FV}(M^\lambda) = \mathrm{FV}(M)$.*

(c) *$(M[x := N])^\lambda \equiv_\alpha M^\lambda[x := N^\lambda]$. .*

(d) *$\mathbf{I}^\lambda =_\beta \lambda x.x$.*

(e) *$(\lambda^* x.M)^\lambda =_{\beta\eta} \lambda x.M^\lambda$.*

Proof: (a) - (c) are clear.
(d):

$$\begin{array}{rcl} \mathbf{I}^\lambda & \equiv & \mathbf{s}^\lambda\ \mathbf{k}^\lambda\ \mathbf{k}^\lambda \\ & =_\beta & \lambda z.\mathbf{k}^\lambda\ z\ (\mathbf{k}^\lambda\ z) \\ & =_\beta & \lambda z.z \ . \end{array}$$

(e): Induction on M:
Case $M \equiv x$:

$$(\lambda x.x)^\lambda \quad \equiv \quad \mathbf{I}^\lambda$$

$$\begin{array}{rl} =_\beta & \lambda x.x \\ \equiv & \lambda x.x^\lambda \ . \end{array}$$

Case $x \notin \mathrm{FV}(M)$. Then $x \notin \mathrm{FV}(M^\lambda)$.

$$\begin{array}{rcl} (\lambda^* x.M)^\lambda & \equiv & (\mathbf{k}\ M)^\lambda \\ & \equiv_\alpha & (\lambda z, x.z)\ \mathbf{k}^\lambda\ M^\lambda \\ & =_\beta & \lambda x.M^\lambda \end{array}$$

(Note that in the last line we used $x \notin \mathrm{FV}(M^\lambda)$ in order to guarantee that the bounded variable x is not replaced by a new one).
Case $M \equiv N\ x$, $x \notin \mathrm{FV}(N)$.

$$\begin{array}{rcl} (\lambda^* x.M)^\lambda & \equiv & N^\lambda \\ & =_\eta & \lambda x.N^\lambda\ x \\ & \equiv & \lambda x.M^\lambda \ . \end{array}$$

Case otherwise, $M \equiv P\ Q$.

$$\begin{array}{rcl} (\lambda^* x.M) & \equiv & \mathbf{s}^\lambda\ (\lambda^* x.P)^\lambda\ (\lambda^* x.Q)^\lambda \\ & =_\beta & \lambda x.(\lambda^* x.P)^\lambda\ x((\lambda^* x.Q)^\lambda\ x) \\ & \overset{\mathrm{IH}}{=}_\beta & \lambda x.(\lambda x.P^\lambda)\ x((\lambda x.Q^\lambda)\ x) \\ & =_\beta & \lambda x.P^\lambda\ Q^\lambda \\ & \equiv & \lambda x.M^\lambda \end{array}$$

Definition 2.9.3 (9.7)
For each λ-term we associate a CL-term $M^{c\eta}$ by

$$\begin{array}{rcl} x^{c\eta} & := & x \ . \\ (M\ N)^{c\eta} & := & M^{c\eta}\ N^{c\eta} \ . \\ (\lambda x.M)^{c\eta} & := & \lambda^* x.(M^{c\eta}) \ . \end{array}$$

Lemma 2.9.4 *(9.10)*

(a) $\mathrm{FV}(M^{c\eta}) = \mathrm{FV}(M)$.

(b) $M \equiv_\alpha N \Rightarrow M^{c\eta} \equiv N^{c\eta}$.

(c) $(M[x := N])^{c\eta} \equiv M^{c\eta}[x := N^{c\eta}]$.

Proof:
(a): easy.
(b): Prove first simultaneously (a) and (c) for N a variable by induction on M.
(c): easy, using (b).

Theorem 2.9.5 *(9.8, 9.14b)*

(a) $(M^{\lambda})^{c\eta} \equiv M$.

(b) $(M^{c\eta})^{\lambda} =_{\beta\eta} M$.

Proof:
(a) Induction on M:
$M \equiv x$: trivial.
$M \equiv N\ P$: by IH.
$M \equiv \mathbf{k}$:

$$\begin{array}{rcl} (\mathbf{k}^{\lambda})^{c\eta} & \equiv & (\lambda x, y.x)^{c\eta} \\ & \equiv & \lambda^* x.\lambda^* y.x \\ & \equiv & \lambda^* x.\mathbf{k}\ x \\ & \equiv & \mathbf{k}\ . \end{array}$$

$M \equiv \mathbf{s}$:

$$\begin{array}{rcl} (\mathbf{s}^{\lambda})^{c\eta} & \equiv & (\lambda x, y, z.x\ z\ (y\ z))^{c\eta} \\ & \equiv & \lambda^* x.\lambda^* y.\lambda^* z.x\ z\ (y\ z) \\ & \equiv & \lambda^* x.\lambda^* y.\mathbf{s}\ (\lambda^* z.x\ z)\ (\lambda^* z.y\ z) \\ & \equiv & \lambda^* x.\lambda^* y.\mathbf{s}\ x\ y \\ & \equiv & \lambda^* x.\mathbf{s}\ x \\ & \equiv & \mathbf{s}\ . \end{array}$$

(b) Induction on M:
M variable: trivial.
$M \equiv P\ Q$: IH.
$M \equiv \lambda x.N$:

$$\begin{array}{rcl} (M^{c\eta})^{\lambda} & \equiv & (\lambda^* x.N^{c\eta})^{\lambda} \\ & \overset{2.9.2\ (e)}{=_{\beta\eta}} & \lambda x.(N^{c\eta})^{\lambda} \\ & \overset{IH}{=_{\beta\eta}} & \lambda x.N \end{array}$$

Lemma 2.9.6 *(9.5)*

(a)
$$M \longrightarrow_{w}^{*} N \Rightarrow M^{\lambda} \longrightarrow_{\beta}^{*} N^{\lambda}\ .$$

(b)
$$M =_{w}^{*} N \Rightarrow M^{\lambda} =_{\beta} N^{\lambda}\ .$$

(c)
$$M =_{c\beta\eta} N \Rightarrow M^{\lambda} =_{\beta\eta} N^{\lambda}\ .$$

Proof: (a): It suffices to consider the case $M \longrightarrow_{w} N$. Induction on $M \longrightarrow_{w} N$.
(b): by (a).

(c): Assume $M =_{\mathrm{c}\beta\eta} N$. Then

$$\mathrm{CL}_{\xi} \vdash M^{\lambda} =_{\beta\eta} N^{\lambda} \ .$$

Induction on this derivation.
Only difficult case (ξ):
Assume $\lambda^* x.M = \lambda^* x.N$ is derived from $M = N$ By IH

$$M^{\lambda} =_{\beta\eta} N^{\lambda} \ ,$$

therefore by Remark 2.9.2 (e)

$$(\lambda^* x.M)^{\lambda} =_{\beta\eta} \lambda x.M^{\lambda} =_{\beta\eta} \lambda x.N^{\lambda} =_{\beta\lambda} (\lambda^* x.N)^{\lambda} \ .$$

Lemma 2.9.7 *(9.11)*

$$M =_{\beta\eta} N \Rightarrow M^{\mathrm{c}\eta} =_{\mathrm{c}\beta\eta} N^{\mathrm{c}\eta} \ .$$

Proof:
Suffices to consider $M \longrightarrow_{\beta} N$, $M \longrightarrow_{\eta} N$, $M \equiv_{\alpha} N$. Difficult cases:
Case $M \equiv_{\alpha} N$: by the above lemma.
Case $M \equiv (\lambda x.P)\, Q$, $N \equiv P[x := Q]$.

$$\begin{array}{rcl} M^{\mathrm{c}\eta} & \equiv & (\lambda^* x.P^{\mathrm{c}\eta})\, Q^{\mathrm{c}\eta} \\ & =_{\mathrm{w}} & P^{\mathrm{c}\eta}[x := Q^{\mathrm{c}\eta}] \\ & \equiv & (P[x := Q])^{\mathrm{c}\eta} \end{array}$$

Case $M \equiv \lambda x.P\, x$, $x \notin \mathrm{FV}(P)$, $N \equiv P$.

$$\begin{array}{rcl} M^{\mathrm{c}\eta} & \equiv & \lambda^* x.P^{\mathrm{c}\eta}\, x \\ & \equiv & P^{\mathrm{c}\eta} \\ & \equiv & N^{\mathrm{c}\eta} \end{array}$$

Case $M \equiv \lambda x.M'$, $N \equiv \lambda x.N'$, $N \longrightarrow_{\beta} N'$ or $N \longrightarrow_{\eta} N'$.
By IH $N^{\mathrm{c}\eta} =_{\mathrm{c}\eta} {N'}^{\mathrm{c}\eta}$, therefore

$$\begin{array}{rcl} M^{\mathrm{c}\eta} & \equiv & \lambda^* x.N^{\mathrm{c}\eta} \\ & =_{\mathrm{c}\eta} & \lambda^* x.{N'}^{\mathrm{c}\eta} \\ & \equiv & {M'}^{\mathrm{c}\eta} \end{array}$$

Theorem 2.9.8 *(9.12, 9.14)*

(a) $M =_{\mathrm{c}\beta\eta} N \Leftrightarrow M^{\lambda} =_{\beta\eta} N^{\lambda}$.

(b) $M =_{\beta\eta} N \Leftrightarrow M^{\mathrm{c}\eta} =_{\mathrm{c}\beta\eta} N^{\mathrm{c}\eta}$.

Proof:
(a) "⇒" Lemma 2.9.6 (c).
"⇐" If $M^\lambda =_{\beta\eta} N^\lambda$ then

$$M \overset{2.9.5\ (a)}{\equiv_\alpha} (M^\lambda)^{c\eta} \overset{2.9.7}{=_\beta} (N^\lambda)^{c\eta} \overset{2.9.5\ (a)}{\equiv_\alpha} N \ .$$

(b) "⇒" Lemma 2.9.7.
"⇐" If $M^{c\eta} =_{\beta\eta} N^{c\eta}$ then

$$M \overset{2.9.5\ (b)}{\equiv_\alpha} (M^{c\eta})^\lambda \overset{(a)}{=_\beta} (N^{c\eta})^\lambda \overset{2.9.5\ (b)}{\equiv_\alpha} N \ .$$

2.9.2 Combinatory β-equality (9C)

In this section in [HS86] the relationship between $=_\beta$ and a combinatory version of it is studied. The main tool is to redefine $\lambda^*x.N$ by omitting the special case $\lambda^*x.N\ x \equiv N$ if $x \notin \mathrm{FV}(N)$, which automatically makes $\lambda^*x.M$ preserve η-equality.
Unfortunately, the equality on combinators which corresponds to β-equality is of similar complexity as the c$\beta\eta$ equality. We omit the details. The interested reader might look at the book where most details (but not all) are carried out.

2.10 Models for $\mathrm{CL_w}$ (10)

2.10.1 Applicative structures (10A)

Notation 2.10.1 (10.1)

(a) In this section, term means CL-term (without constants apart from **k**, **s**).

(b) Var will be the set of variables used in the definition of CL-terms.

Definition 2.10.2 (10.2, 10.1)

(a) An *applicative structure* is a pair $\mathcal{D} = (D, \cdot)$, s.t.

- D is a set with at least two elements, called the *domain of* $\mathcal{D}$ and
- $\cdot : D \times D \to D$.

(b) Let in this subsection $(D, \cdot)$ be an applicative structure, and a, b, c, d, e denote (in this section) elements of D.

(c) We write $\cdot$ infix and omit parenthesis with the convention that $\cdot$ is associative to the left, i.e.

$$a \cdot b \cdot c \cdot d := (((a \cdot b) \cdot c) \cdot d) \ .$$

(d) An *assignment w.r.t.* D (in [HS86] it is called *valuation*) is a function

$$\rho : \mathrm{Var} \rightarrow D \ .$$

(e) If ρ is an assignment w.r.t. $(D, \cdot)$, $x \in \mathrm{Var}$, $a \in D$ then ρ_x^a is the assignment defined by

$$\rho_x^a(y) := \begin{cases} a & \text{if } x \equiv y, \\ \rho(y) & \text{if } x \not\equiv y. \end{cases}$$

(f) If t is any expression depending on $d_1, \ldots, d_n \in D$, let

$$\lambda\backslash\, d_1, \ldots, d_n \in D.t$$

or (if D is clear from the context)

$$\lambda\backslash\, d_1, \ldots, d_n.t$$

be the (set theoretic) function, which assigns to $d_1, \ldots, d_n \in D$ t.

Definition 2.10.3 (10.3, 10.4)

(a) A function $f : D^n \rightarrow D$ $(n \geq 1)$ is *representable in* $(D, \cdot)$ or, if $\cdot$ is clear from the context, short *representable in* D, iff there exists an $a \in D$ s. t.

$$\forall d_1, \ldots, d_n \in D.f(d_1, \ldots, d_n) = a \cdot d_1 \cdot d_2 \cdot \cdots \cdot d_n \ ,$$

in other words

$$f = \lambda\backslash\, d_1, \ldots, d_n \in D.a \cdot d_1 \cdot d_2 \cdot \cdots \cdot d_n \ .$$

In this case a is called *a representative of* f.

(b) $(D^n \rightarrow_{\mathrm{rep}} D)$ is the set of n-ary representable functions in D.

(c) For $a \in D$, $\mathrm{Fun}(a)$ is the unary function represented by a i.e.

$$\mathrm{Fun}(a) : D \rightarrow D, \quad \mathrm{Fun}(a)(d) := a \cdot d \ ,$$

in other words

$$\mathrm{Fun}(a) = \lambda\backslash\, d \in D.a \cdot d \ .$$

(d) For $f : D \rightarrow D$ let

$$\mathrm{Rep}(f) := \{a \mid \mathrm{Fun}(a) = f\} \ .$$

Definition 2.10.4 (10.5, 10.7)

(a) For $a, b \in D$, a *is extensionally equivalent to* b, $(a \sim b)$ iff $\mathrm{Fun}(a) = \mathrm{Fun}(b)$.

(b) For $a \in D$ the *extensional equvivalence class containing of* a, $\tilde{a}$ is defined by

$$\tilde{a} := \{b \in D \mid b \sim a\} \ .$$

Further

$$D/\sim := \{\tilde{a} \mid a \in D\} \ .$$

(c) $(D, \cdot)$ is *extensional* iff

$$\forall a, b \in D.(\mathrm{Fun}(a) = \mathrm{Fun}(b) \rightarrow a = b) \ .$$

Lemma 2.10.5 *(10.6 d, 10.8)*

(a) Rep *is a bijection from* $D \rightarrow_{\mathrm{rep}} D$ *to* $D/\sim$.

(b) The following are equivalent:

(a) $(D, \cdot)$ *is extensional.*

(b) $\forall a \in D.\tilde{a}$ *is a singleton.*

(c) $\forall f \in D \rightarrow_{\mathrm{rep}} D.\mathrm{Rep}(f)$ *is a singleton.*

(d) $Fun : D \rightarrow (D \rightarrow_{\mathrm{rep}} D)$ *is injective.*

(e) $Fun : D \rightarrow (D \rightarrow_{\mathrm{rep}} D)$ *is bijective.*

2.10.2 Combinatory algebras (10B)

Definition 2.10.6 (a) A *model of* $\mathrm{CL_w}$ is a quadrupel

$$(D, \cdot, \mathbf{k}^D, \mathbf{s}^D)$$

s. t.

- $(D, \cdot)$ is an applicative structure,
- $\mathbf{k}^D, \mathbf{s}^D \in D$,
- $\forall a, b \in D(\mathbf{k}^D \cdot a \cdot b = a)$.
- $\forall a, b, c \in D(\mathbf{s}^D \cdot a \cdot b \cdot c = a \cdot c \cdot (b \cdot c))$.

(b) A *combinatorial algebra* is an applicative structure $(D, \cdot)$ s. t. for some $\mathbf{k}^D, \mathbf{s}^D \in D$

$$(D, \cdot, \mathbf{k}^D, \mathbf{s}^D) \text{ is a model of } \mathrm{CL_w} \ .$$

(c) Let in this subsection $(D, \cdot, \mathbf{k}^D, \mathbf{s}^D)$ be a model of $\mathrm{CL_w}$.

Remark 2.10.7 *(a)* $\mathbf{k}^D \neq \mathbf{s}^D$.

(b) In an extensional combinatory algebra $\mathbf{k}^D$, $\mathbf{s}^D$ *are uniquely defined.*

Proof: (a) Otherwise

$$\begin{array}{rcl}\mathbf{k}^D\cdot\mathbf{k}^D & = & \mathbf{k}^D\cdot\mathbf{k}^D\cdot\mathbf{k}^D\cdot\mathbf{k}^D\\ & = & \mathbf{s}^D\cdot\mathbf{k}^D\cdot\mathbf{k}^D\cdot\mathbf{k}^D\\ & = & \mathbf{k}^D\cdot\mathbf{k}^D\cdot(\mathbf{k}^D\cdot\mathbf{k}^D)\\ & = & \mathbf{k}^D\end{array}$$

and for $a, b \in D$ s. t. $a \neq b$

$$\begin{array}{rcl}a & = & \mathbf{k}^D\cdot a\cdot b\\ & = & \mathbf{k}^D\cdot\mathbf{k}^D\cdot a\cdot b\\ & = & \mathbf{s}^D\cdot\mathbf{k}^D\cdot a\cdot b\\ & = & \mathbf{k}^D\cdot b\cdot(a\cdot b) = b\ .\end{array}$$

(b) The equations

$$\begin{array}{rcl}\forall a, b \in D & . & (\mathbf{k}^D\cdot a\cdot b = a)\ ,\\ \forall a, b, c \in D & . & (\mathbf{s}^D\cdot a\cdot b\cdot c = a\cdot c\cdot(b\cdot c))\ .\end{array}$$

determine $\mathbf{k}^D$, $\mathbf{s}^D$ uniquely by extensionality.

Definition 2.10.8 (10.10, 10.13)

(a) Let $\mathcal{D} = (D, \cdot, \mathbf{k}^D, \mathbf{s}^D)$ be a model of $\mathrm{CL_w}$. For assignments ρ and terms N we define $[\![\ N\ ]\!]^{\mathcal{D}}_{\rho} \in D$ by

- $[\![\ x\ ]\!]^{\mathcal{D}}_{\rho} := \rho(x)$.
- $[\![\ \mathbf{k}\ ]\!]^{\mathcal{D}}_{\rho} := \mathbf{k}^D$.
- $[\![\ \mathbf{s}\ ]\!]^{\mathcal{D}}_{\rho} := \mathbf{s}^D$.
- $[\![\ M\ N\ ]\!]^{\mathcal{D}}_{\rho} := [\![\ M\ ]\!]^{\mathcal{D}}_{\rho}\cdot[\![\ N\ ]\!]^{\mathcal{D}}_{\rho}$.

If there is no confusion we write $[\![\ M\ ]\!]$, $[\![\ M\ ]\!]_{\rho}$ or $[\![\ M\ ]\!]^{\mathcal{D}}$ instead of $[\![\ M\ ]\!]^{\mathcal{D}}_{\rho}$.

(b)

$$\begin{array}{rcl}\mathcal{D}\models M = N[\rho] & :\Leftrightarrow & [\![\,M\,]\!]^{\mathcal{D}}_{\rho} = [\![\,N\,]\!]^{\mathcal{D}}_{\rho}\ ,\\ \mathcal{D}\models M = N & :\Leftrightarrow & \forall\rho\ \text{assignment}.\mathcal{D}\models M = N[\rho]\ .\end{array}$$

(c) A *combinatory* $\beta\eta$*-model* or *model of* $\mathrm{CL}_{\beta\eta_{\mathrm{ax}}}$ or *Curry-algebra* is a model of $\mathrm{CL_w}$ which fulfills the $\beta\eta$ axioms (Definition 2.8.9).

Lemma 2.10.9 *(10.11, 10.11.1, 10.12)*

(a) If $\forall x \in \mathrm{FV}(M).\rho(x) = \sigma(x)$ *then* $[\![\ M\ ]\!]_{\rho} = [\![\ M\ ]\!]_{\sigma}$.

(b) If $\mathrm{FV}(M) = \emptyset$*, for all* ρ, σ $[\![\ M\ ]\!]_{\rho} = [\![\ M\ ]\!]_{\sigma}$.

(c) $[\![\, M[x := N] \,]\!]_\rho = [\![\, M \,]\!]_{\rho_x^{[\![\, N \,]\!]_\rho}}$

Lemma 2.10.10 *Each model of* $\mathrm{CL_w}$ *or* $\mathrm{CL}_{\beta\eta\mathrm{ax}}$ *fulfills all the provable equations of the corresponding theory.*

Proof: trivial.

Definition 2.10.11 (a) Let T be $\mathrm{CL_w}$ or $\mathrm{CL}_{\beta\eta\mathrm{ax}}$.

Define for each CL-term M $[M] := \{N \text{ CL-term} | \, T \vdash M = N\}$.

The *term model of* T, $\mathcal{M}(T)$, is

$$(D, \cdot, \mathbf{k}, \mathbf{s})$$

with

- $D = \{[M] \mid M \text{CL-term }\}$.
- $[M] \cdot [N] := [M\ N]$.
- $\mathbf{k}^D := [\mathbf{k}]$.
- $\mathbf{s}^D := [\mathbf{s}]$.

Note that the domain of the term model are equivalence classes of *open* terms.

Theorem 2.10.12 *(10.20)*
Let $\mathcal{D} := (D, \cdot, \mathbf{k}^D, \mathbf{s}^D)$ *be the term model of* T*,* $T \in \{\mathrm{CL_w}, \mathrm{CL}_{\beta\eta\mathrm{ax}}\}$.

(a) $\cdot$ *is well-defined.*

(b) $\mathcal{D}$ *is a model of* T*.*

(c) If

- $\mathrm{FV}(M) = \{x_1, \ldots, x_n\}$,
- $\rho(x_i) = [N_i]$ *(*$i = 1, \ldots, n$*)*

then

$$[\![\, M \,]\!]_\rho^{\mathcal{D}} = [M[x_1 := N_1, \ldots, x_n := N_n]] \ .$$

The following theorem will fail for the λ-calculus:

Theorem 2.10.13 *(10.22, the Submodel theorem).*
Assume

- $T \in \{\mathrm{CL_w}, \mathrm{CL}_{\beta\eta\mathrm{ax}}\}$,
- $\mathcal{D} := (D, \cdot, \mathbf{k}, \mathbf{s})$ *is a model of* T*.*
- $D' \subseteq D$,

- *D' closed under $\cdot$, $\mathbf{k}^D$, $\mathbf{s}^D$, i.e.*
  - $\cdot[D' \times D'] \subseteq D'$ *(i.e. $\forall d, d' \in D'.d \cdot d' \in D'$),*
  - $\mathbf{k}^D, \mathbf{s}^D \in D'$.

Then the submodel of $\mathcal{D}$ given by D', $\mathcal{D} \restriction D' := (D', \cdot \restriction (D' \times D'), \mathbf{k}^D, \mathbf{s}^D)$ is a model of T as well.

Proof: D' has two elements ($\mathbf{k}^D$, $\mathbf{s}^D$) and fulfills all the axioms and is closed under all the rules, since D is.

Definition 2.10.14 (10.23).
The *interior*, written as $\mathcal{D}^{\mathrm{o}}$, of a model of T $\mathcal{D} = (D, \cdot, \mathbf{k}^D, \mathbf{s}^D)$ ($T \in \{\mathrm{CL_w}, \mathrm{CL}_{\beta\eta\mathrm{ax}}\}$) is $\mathcal{D} \restriction D^{\mathrm{o}}$ with

$$D^o := \{[\![M]\!]^{\mathcal{D}} \mid M \text{ closed } \}.$$

Lemma 2.10.15 *$\mathcal{D}^{\mathrm{o}}$ is the least submodel of $\mathcal{D}$:*

- *$\mathcal{D}^{\mathrm{o}}$ is a submodel of $\mathcal{D}$ and*
- *for every other submodel $\mathcal{D}' = (D', \dots)$ we have $D^{\mathrm{o}} \subseteq D'$.*

Proof: D^0 is the least subset of D closed under $\mathbf{k}$, $\mathbf{s}$, $\cdot$.

Remark 2.10.16 *(10.24). One sees immediately, that every extensional model of $\mathrm{CL_w}$ is a model of $\mathrm{CL}_{\beta\eta\mathrm{ax}}$.*
However, there are non-extensional models of $\mathrm{CL}_{\beta\eta\mathrm{ax}}$:
Plotkin ([Plo74]) has constructed closed λ-terms M, N s. t.

- $\lambda\beta + (\mathrm{ext}) \vdash M\ Q = N\ Q$ *for all closed λ-terms Q, but*
- $\lambda\beta + (\mathrm{ext}) \not\vdash M = N$.

With $M' := M^{\mathrm{c}\eta}$, $N' := N^{\mathrm{c}\eta}$ it follows

- $\mathrm{CL}_{\beta\eta\mathrm{ax}} \vdash M'\ Q = N'\ Q$ *for all closed* CL*-terms Q, but*
- $\mathrm{CL}_{\beta\eta\mathrm{ax}} \not\vdash M' = N'$.

Now the interior $\mathcal{D}$ of the term model of $\mathrm{CL}_{\beta\eta\mathrm{ax}}$ has domain

$$D = \{[M] \mid M \ closed\ \mathrm{CL} - term\} \ .$$

Therefore for all $x \in D$, $x = [Q]$ for some closed term Q and therefore

$$\begin{array}{rcl} [M'] \cdot x & = & [M'] \cdot [Q] \\ & = & [M'\ Q] = [N'\ Q] \\ & = & [N'] \cdot [Q] \\ & = & [N'] \cdot x \ , \\ & & but \\ [M'] & \neq & [N'] \ . \end{array}$$

2.10.3 A more abstract definition of combinatory algebras (10.25 - 10.28)

Definition 2.10.17 (10.26, 10.27)

(a) A *combination* of variables $x_1, \dots, x_n$ is a CL-term built from atoms $x_1, \dots, x_n$ only (especially no atoms **k**, **s**!).

(b) For combinations M of $x_1, \dots, x_n$, assignments ρ and applicative structures $\mathcal{D} := (D, \cdot)$ we define $[\![M]\!]^{\mathcal{D}}_{\rho}$ by:

- $[\![x]\!]^{\mathcal{D}}_{\rho} := \rho(x)$.
- $[\![M\ N]\!]^{\mathcal{D}}_{\rho} := [\![M]\!]^{\mathcal{D}}_{\rho} \cdot [\![N]\!]^{\mathcal{D}}_{\rho}$.

(c) An applicative structure $\mathcal{D} := (D, \cdot)$ is *combinatorially complete* iff for any sequence $u, x_1, \dots, x_n$ of variables and every combination M of $x_1, \dots, x_n$ the formula

$$\exists u.\forall x_1, \dots, x_n.u\ x_1 \cdots x_n = M$$

is true in $\mathcal{D}$, which means

$$\exists a \in D.\forall d_1, \dots, d_n \in D.a \cdot d_1 \cdot \dots \cdot d_n = [\![M]\!]_{\rho^{d_1}_{x_1}{}^{d_2}_{x_2}\cdots{}^{d_n}_{x_n}} \quad ,$$

in other words there exists $a \in D$ which represents

$$\lambda\!\backslash\, d_1, \dots, d_n \in D.[\![M]\!]_{\rho^{d_1}_{x_1}{}^{d_2}_{x_2}\cdots{}^{d_n}_{x_n}} \quad .$$

Theorem 2.10.18 *(10.28, combinatory completeness theorem. An applicative structure $(D, \cdot)$ is combinatory complete iff it is a combinatory algebra.*

Proof:
If $(D, \cdot)$ is combinatorially complete, then we can choose

- $\mathbf{k}^D \in D$ such that

$$\lambda\!\backslash\, d, e \in D.\mathbf{k}^D \cdot d \cdot e = \lambda\!\backslash\, d, e \in D.[\![x]\!]_{\rho^{d}_{x}{}^{e}_{y}}$$

- and $\mathbf{s}^D \in D$ such that

$$\lambda\!\backslash\, d, e, f \in D.\mathbf{s}^D \cdot d \cdot e \cdot f = \lambda\!\backslash\, d, e, f \in D.[\![x\ z\ (y\ z)]\!]_{\rho^{d}_{x}{}^{e}_{y}{}^{f}_{z}}$$

On the other hand in a combinatory algebra $\mathcal{D}$ with correspdonding model of $\mathrm{CL_w}$ $\mathcal{D}'$,

$$[\![\lambda^* x_1, \dots, x_n.M]\!]^{\mathcal{D}'}$$

represents

$$\lambda\!\backslash\, d_1, \dots, d_n.[\![M]\!]^{\mathcal{D}}_{\rho^{d_1}_{x_1}{}^{d_2}_{x_2}\cdots{}^{d_n}_{x_n}} \quad :$$

For all $d_1, \ldots, d_n \in D$

$$
\begin{aligned}
& [\![\lambda^* x_1, \ldots, x_n.M]\!]^{\mathcal{D}'} \cdot d_1 \cdot \cdots \cdot d_n \\
= \quad & [\![\lambda^* x_1, \ldots, x_n.M]\!]^{\mathcal{D}}_{\rho^{d_1}_{x_1}{}^{d_2}_{x_2}\cdots{}^{d_n}_{x_n}} \cdot [\![x_1]\!]_{\rho^{d_1}_{x_1}{}^{d_2}_{x_2}\cdots{}^{d_n}_{x_n}} \cdot \cdots \cdot [\![x_n]\!]_{\rho^{d_1}_{x_1}{}^{d_2}_{x_2}\cdots{}^{d_n}_{x_n}} \\
= \quad & [\![(\lambda^* x_1, \ldots, x_n.M)\, x_1 \cdots x_n]\!]^{\mathcal{D}}_{\rho^{d_1}_{x_1}{}^{d_2}_{x_2}\cdots{}^{d_n}_{x_n}} \\
= \quad & [\![M]\!]^{\mathcal{D}}_{\rho^{d_1}_{x_1}{}^{d_2}_{x_2}\cdots{}^{d_n}_{x_n}} \quad .
\end{aligned}
$$

2.11 Models for $\lambda\beta$ (11)

2.11.1 The definition of a λ-model (11A)

Notation 2.11.1 (11.1) In this section, term means λ-term.

Why not take models of combinatory logic as models for the λ-calculus?
The definition of models of combinatory logic was easy this was just the straightforward of models for first order logic (in fact for essentially only the interpretation of terms was needed) to the combinatory logic. To define models for the λ-calculus however is quite complicated, since we have to interpret $\lambda x.t$. Especially closed terms are built from open terms. A tempting trivial solution would be to choose an extension of combinatory logic which is via some translation equivalent to $\lambda\beta$. (Such an axiomatization was treated in chapter 9C of the book and yields a theory $\mathrm{CL}_{\beta\mathrm{ax}}$ similar to $\mathrm{CL}_{\beta\eta\mathrm{ax}}$; the translation of λ-terms into CL just makes use of the variant $\lambda^{\mathrm{w}}x.M$ of $\lambda^* x.M$ which omits the case $\lambda^* x.M\ x = M$ for closed M).
However in $\mathrm{CL}_{\beta\mathrm{ax}}$ the translation of the ξ-rule is admissible, but not derivable, and therefore the models of $\mathrm{CL}_{\beta\mathrm{ax}}$ fulfill all equations derivable (without assumptions) using the translation of the ξ-rule, but are not closed under it. Therefore as well, if we considered models of $\mathrm{CL}_{\beta\mathrm{ax}}$ as models of the λ-calculus via the translation into combinators, we would not get a model closed under the ξ-rule. (That we actually get models which violate the ξ-rule is shown in [HS86]).

We will in the following define in the following first λ-models in a almost trivial way. It will be very difficult to construct with this definition λ-models, therefore we will look then at more abstract concepts.

Definition 2.11.2 (11.1)
If C, D are sets, $f : C \to D$, $g : D \to C$,

$$g \circ f = \mathrm{id}_C \quad ,$$

then

- g is called a *left inverse of* f,
- f is called a *right inverse of* g,
- C is called a *retract of* C,
- (f, g) is called a *retraction.*

Remark 2.11.3 *Assume* (f, g) *is a retraction. Then*

(a) $h := f \circ g$ *is idempotent, i.e.* $h \circ h = h$,

(b) g *is surjective,* f *is injective.*

Further we have that for every pair (f, g) *with* $f : C \to D$, $g : D \to C$ *which fulfill (a), (b), that* (f, g) *is a retraction, i.e.* $g \circ f = \mathrm{id}_D$.

Proof: That a retraction fulfills (a), (b) is easy.
On the other hand, if f, g fulfill (a), (b), $x \in C$, then $x = g(y)$ for some $y \in D$, and

$$f(x) = f(g(y)) = f(g(f(g(y)))) = f(g(f(x))) \ ,$$

therefore

$$g(f(x)) = x \ ,$$
$$g \circ f = \mathrm{id}_C \ .$$

Definition 2.11.4 (11.3)
A *λ-model* or *model of* $\lambda\beta$ is a tripel

$$\mathcal{D} = (D, \cdot, [\![\cdot]\!])$$

s. t.

- $(D, \cdot)$ is an applicative structure,
- $[\![\cdot]\!]$ is a mapping from λ-terms M and valuations σ to elements $[\![M]\!]_\sigma$ of D

s. t. for all variables x, y, terms P, Q, M, $d \in D$, valuations σ, σ the following holds:

(a) $[\![x]\!]_\sigma = \sigma(x)$;

(b) $[\![P\,Q]\!]_\sigma = [\![P]\!]_\sigma \cdot [\![Q]\!]_\sigma$.

(c) $[\![\lambda x.P]\!]_\sigma \cdot d = [\![P]\!]_{\sigma^d_x}$.

(d) $[\![M]\!]_\sigma = [\![M]\!]_\rho$ if $\sigma \restriction \mathrm{FV}(M) = \rho \restriction \mathrm{FV}(M)$.

(e) $[\![\lambda x.M]\!]_\sigma = [\![\lambda y.(M[x := y])]\!]_\sigma$, if $y \notin \mathrm{FV}(M)$,

(f) If

$$\forall d \in D.[\![M]\!]_{\sigma^d_x} = [\![N]\!]_{\sigma^d_x} \ ,$$

then

$$[\![\lambda x.M]\!]_\sigma = [\![\lambda x.N]\!]_\sigma \ .$$

We write sometimes $[\![M]\!]^{\mathcal{D}}_\sigma$, instead of $[\![M]\!]_\sigma$, (especially if there are several models involved) and omit σ, if $[\![M]\!]$ does not depend on σ.
Further, if $\mathrm{FV}(M) \subseteq \{x_1, \ldots, x_n\}$, we write

$$[\![M]\!]_{[x_1:=d_1,\ldots,x_n:=d_n]}$$

for $[\![M]\!]_\sigma$, where σ is any assignment s.t. $\sigma(x_i) = d_i$.

Definition 2.11.5 (11.11).
Let $\mathcal{D} = (D, \cdot, [\![\cdot]\!])$ be a λ-model. Then

$$\begin{array}{rcl} \mathcal{D} \models M = N[\sigma] & :\Leftrightarrow & [\![M]\!]_\sigma = [\![M]\!]_\sigma \ , \\ \mathcal{D} \models M = N & :\Leftrightarrow & \forall \sigma \text{ assignment } \mathcal{D} \models M = N[\sigma] \end{array}$$

We say $\mathcal{D}$ *models* or *satisfies* $M = N$ for $\mathcal{D} \models M = N$.

Remark 2.11.6 *(11.4).*

(a) Conditions (a), (b) express compositionality of the model.

(b) Condition (c) express that $[\![\lambda x.P]\!]_\sigma$ *is a representative of the function*

$$\lambda\!\!\backslash\ d \in D.[\![P]\!]_{\sigma^d_x} : D \to D \ .$$

(c) Condition (d) is needed, since only in extensional applicative structures $[\![\lambda x.M]\!]_\sigma$ *is by (c) completely defined which would guarantee (c). Otherwise* $[\![\lambda x.M]\!]_\sigma$ *might really depend on the choice of other variables.*

(d) Condition (e) corresponds to (α).

(e) Condition (f) corresponds to (ξ)*:*

$$\lambda\!\!\backslash\ d.[\![M]\!]_{\sigma^d_x} = \lambda\!\!\backslash\ d.[\![N]\!]_{\sigma^d_x} \Rightarrow [\![\lambda x.M]\!]_\sigma = [\![\lambda x.N]\!]_\sigma \ .$$

(f) From condition (c) and (f) follows weak extensionality:

$$\lambda\!\!\backslash\ d.[\![M]\!]_{\sigma^d_x} \sim \lambda d.[\![N]\!]_{\sigma^d_x} \Rightarrow [\![\lambda x.M]\!]_\sigma = [\![\lambda x.N]\!]_\sigma \ .$$

(g) One can replace the conditions in Definition 2.11.5 (a) - (f) by the following conditions:

(a) $[\![x]\!]_\sigma = \sigma(x)$;

(b) $[\![P\ Q]\!]_\sigma = [\![P]\!]_\sigma \cdot [\![Q]\!]_\sigma$.

(c) $[\![\lambda x.P]\!]_\sigma \cdot d = [\![P]\!]_{\sigma^d_x}$.

(d) $[\![\lambda x.P]\!]_\sigma \sim [\![\lambda y.Q]\!]_\rho \Rightarrow [\![\lambda x.P]\!]_\sigma = [\![\lambda y.Q]\!]_\rho$.

That the above four conditions hold will be proved below in Lemma 2.11.8 (b). Further from the new conditions follow of the original ones:

- *(a) - (c) are new conditions as well,.*
- *(d) follows by induction on M,*
- *(e) follows by proving Lemma 2.11.7 (a) below from the new contitions by induction on M and then using the new condition (d),*
- *(f) follows directly by the new conditions (c) and (d).*

Lemma 2.11.7 *(11.7)*
Let $(D, \cdot, [\![\cdot]\!])$ be a λ-model.

(a) If $y \notin \mathrm{FV}(M)$ then

$$[\![M]\!]_\sigma = [\![M[x := y]]\!]_{\sigma_y^{\sigma(x)}} .$$

(b) If

- $\mathrm{FV}(M) \subseteq \{x_1, \ldots, x_n\}$,
- $x_1, \ldots, x_n, y_1, \ldots, y_n$ *are distinct,*
- $\sigma(x_i) = \rho(y_i)$ *$(i = 1, \ldots, n)$*

then

$$[\![M[x_1 := y_1, \ldots, x_n := y_n]]\!]_\rho = [\![M]\!]_\sigma .$$

Proof:
(a):
Let $d := \sigma(x)$.

$$\begin{aligned} [\![M]\!]_\sigma &= [\![M]\!]_{\sigma^d_x} \\ &= [\![\lambda x.M]\!]_\sigma \cdot d \\ &= [\![\lambda y.M[x := y]]\!]_\sigma \cdot d \\ &= [\![M[x := y]]\!]_{\sigma^d_y} \\ &= [\![M[x := y]]\!]_{\sigma_y^{\sigma(x)}} . \end{aligned}$$

(b)

$$\begin{aligned} &\forall y \in \mathrm{FV}(M[x_1 := y_1, \ldots, x_n := y_n]).\rho(y) \\ = \ &\sigma_{y_1}^{\sigma(x_1)}{}_{y_2}^{\sigma(x_2)} \cdots {}_{y_n}^{\sigma(x_n)}(y) \end{aligned}$$

therefore

$$\begin{aligned} [\![M]\!]_\sigma &\overset{\text{(a)}}{=} [\![M[x_1 := y_1, \ldots, x_n := y_n]]\!]_{\sigma_{y_1}^{\sigma(x_1)}{}_{y_2}^{\sigma(x_2)} \cdots {}_{y_n}^{\sigma(x_n)} x} \\ &= [\![M[x_1 := y_1, \ldots, x_n := y_n]]\!]_\rho . \end{aligned}$$

Lemma 2.11.8 *(11.8, Berry's extensionality property)*
Let $(D, \cdot, [\![\,\cdot\,]\!])$ be a λ-model.

(a)
$$(\forall d \in D. [\![\, P \,]\!]_{\sigma_x^d} = [\![\, Q \,]\!]_{\rho_y^d}) \Rightarrow [\![\, \lambda x.P \,]\!]_{\sigma} = [\![\, \lambda y.Q \,]\!]_{\rho} \ .$$

(b)
$$[\![\, \lambda x.P \,]\!]_{\sigma} \sim [\![\, \lambda y.Q \,]\!]_{\rho} \Rightarrow [\![\, \lambda x.P \,]\!]_{\sigma} = [\![\, \lambda y.Q \,]\!]_{\rho} \ .$$

Proof:
(b) follows by (a).
(a):
Assume
$$\forall d \in D. [\![\, P \,]\!]_{\sigma_x^d} = [\![\, Q \,]\!]_{\rho_y^d} \ ,$$

and let

$$\begin{array}{rcl} \mathrm{FV}(P) \setminus \{x\} & = & \{x_1, \ldots, x_m\} \ , \\ \mathrm{FV}(Q) \setminus \{y\} & = & \{y_1, \ldots, y_n\} \ . \end{array}$$

Let $z, u_1, \ldots, u_m, v_1, \ldots, v_n$ be distinct fresh variables,

$$\begin{array}{rcl} P' & := & P[x := z, x_1 := u_1, \ldots, x_n := u_n] \ , \\ Q' & := & Q[y := z, y_1 := v_1, \ldots, y_m := v_m] \\ \tau & := & \sigma_{u_1}^{\sigma(x_1)}{}_{u_2}^{\sigma(x_2)} \cdots {}_{u_n}^{\sigma(x_n)}{}_{v_1}^{\rho(y_1)}{}_{v_2}^{\rho(y_2)} \cdots {}_{v_m}^{\rho(y_m)} \end{array}$$

Then

$$\begin{array}{rcl} [\![\, P' \,]\!]_{\tau_z^d} & \overset{2.11.7\ (b)}{=} & [\![\, P \,]\!]_{\sigma_x^d} \\ & = & [\![\, Q \,]\!]_{\rho_y^d} \\ & \overset{2.11.7\ (b)}{=} & [\![\, Q' \,]\!]_{\tau_z^d} \end{array}$$

and therefore

$$\begin{array}{rcl} [\![\, \lambda x.P \,]\!]_{\rho} & = & [\![\, \lambda z.P[x := z] \,]\!]_{\rho} \\ & \overset{2.11.7\ (b)}{=} & [\![\, \lambda z.P' \,]\!]_{\tau} \\ & = & [\![\, \lambda z.Q' \,]\!]_{\tau} \\ & \overset{2.11.7\ (b)}{=} & [\![\, \lambda z.Q[x := z] \,]\!]_{\rho} \\ & = & [\![\, \lambda y.Q \,]\!]_{\rho} \end{array}$$

Lemma 2.11.9 *(11.10).*
Let $(D, \cdot, [\![\,\cdot\,]\!])$ be a λ-model.

(a)
$$[\![\, M[x := N] \,]\!]_{\sigma} = [\![\, M \,]\!]_{\sigma_x^{[\![N]\!]_\sigma}} \ .$$

(b)
$$[\![(\lambda x.M)\ N]\!]_\sigma = [\![M[x := N]]\!]_\sigma \ .$$

Proof:
(a): Induction on M. Let $b := [\![N]\!]_\sigma$.
Case $M \equiv x$:
$$[\![M[x := N]]\!]_\sigma = [\![N]\!]_\sigma = b = [\![x]\!]_{\sigma^b_x} = [\![M]\!]_{\sigma^b_x} \ .$$
Case $M \equiv y \not\equiv x$:
$$[\![M[x := N]]\!]_\sigma = [\![y]\!]_\sigma = \sigma(y) = \sigma^b_x(y) = [\![M]\!]_{\sigma^b_x} \ .$$
Case $M \equiv P\ Q$: Immediate by IH.
Case $M \equiv \lambda x.P$: $x \notin \mathrm{FV}(M)$, therefore
$$[\![M[x := N]]\!]_\sigma = [\![M]\!]_\sigma = [\![M]\!]_{\sigma^b_x} \ .$$
Case $M \equiv \lambda y.P$, $y \notin \mathrm{FV}(M)$: Since $[\![\cdot]\!]$ respects α-equality, w.l.o.g. $y \notin \mathrm{FV}(N)$. By IH follows for all $d \in D$
$$[\![P[x := N]]\!]_{\sigma^d_y} = [\![P]\!]_{\sigma^{d\,b}_{y\,x}} = [\![P]\!]_{\sigma^{b\,d}_{x\,y}}$$
and by Lemma 2.11.8 (b) therefore
$$[\![\lambda y.P[x := N]]\!]_\sigma = [\![\lambda y.P]\!]_{\sigma^b_x} \ .$$
(b) Let $b := [\![N]\!]_\sigma$.
$$\begin{array}{rcl} [\![(\lambda x.M)\ N]\!]_\sigma & = & [\![\lambda x.M]\!]_\sigma \cdot b \\ & = & [\![M]\!]_{\sigma^b_x} \\ & \overset{(a)}{=} & [\![M[x := N]]\!]_\sigma \ . \end{array}$$

Theorem 2.11.10 *(11.12).*
Every λ-model models all provable equations of $\lambda\beta$.

Proof:
Induction on the derivation.
Closure under (ρ) (reflexivity), (σ) (symmetry) and (τ) (transitivity) are trivial.
Closure under (α), (μ), (ν), (ξ) follow by definition.
Closure under (β) follows by Lemma 2.11.9 (b).

Corollary 2.11.11 *(11.12.1) If $(D, \cdot, [\![\cdot]\!])$ is a λ-model, then $(D, \cdot)$ is a combinatory algebra, especially combinatorially complete.*

Proof:
$$\mathbf{k} := [\![\lambda x, y.x]\!] \quad \mathbf{s} := [\![\lambda x, y, z.x\ z\ (y\ z)]\!] \ .$$

Definition 2.11.12 (11.14)
A *model of* $\lambda\beta\eta$ is a λ-model which models $\lambda x.M\ x = M$ for all M and all $x \notin \mathrm{FV}(M)$.

Remark 2.11.13 *A model of $\lambda\beta\eta$ fulfills all provable equations of $\lambda\beta\eta$.*

Note that the following theorem does not hold for combinatory logic, see Remark 2.10.16.

Theorem 2.11.14 *(11.15) A λ-model $\mathcal{D}$ is extensional iff it is a model of $\lambda\beta\eta$.*

Proof:
Let $\mathcal{D} = (D, \cdot, [\![\cdot]\!])$.
Assume $\mathcal{D}$ is extensional, M a term. Then

$$\begin{array}{rcl} [\![\lambda x.M\ x]\!]_\sigma \cdot d & = & [\![M\ x]\!]_{\sigma_x^d} \\ & = & [\![M]\!]_{\sigma_x^d} \cdot [\![x]\!]_{\sigma_x^d} \\ & = & [\![M]\!]_\sigma \cdot d \ , \end{array}$$

therefore by extensionality

$$[\![\lambda x.M\ x]\!]_\sigma = [\![M]\!]_\sigma \ .$$

Assume $\mathcal{D}$ is a model of $\lambda\beta\eta$.
Assume

$$\forall d \in D.[\![M]\!]_\sigma \cdot d = [\![N]\!]_\sigma \cdot d \ ,$$

$x \notin \mathrm{FV}(M\ N)$. Then

$$\begin{array}{rcl} [\![M\ x]\!]_{\sigma_x^d} & = & [\![M]\!]_\sigma \cdot d \\ & = & [\![N]\!]_\sigma \cdot d \\ & = & [\![N\ x]\!]_{\sigma_x^d} \end{array}$$

Therefore

$$[\![\lambda x.M\ x]\!]_\sigma = [\![\lambda x.N\ x]\!]_\sigma$$

and

$$\begin{array}{rcl} [\![M]\!]_\sigma & = & [\![\lambda x.M\ x]\!]_\sigma \\ & = & [\![\lambda x.N\ x]\!]_\sigma \\ & = & [\![N]\!]_\sigma \ . \end{array}$$

Definition 2.11.15 (11.16).
Let $T \in \{\lambda\beta, \lambda\beta\eta\}$.
Define for λ-terms M

$$[M] := \{N \mid T \vdash M = N\} \ .$$

The *term model* of T, called $\mathcal{M}T$ is

$$(D, \cdot, [\![\cdot]\!])$$

with

$$D := \{[M] \mid M \lambda\text{-term} \ ,$$
$$[M] \cdot [N] := [M\ N] \ ,$$

and, if $\mathrm{FV}(M) = \{x_1, \ldots, x_n\}$

$$[\![M]\!]_\sigma := [M[x_1 := \sigma(x_1), \ldots, x_n := \sigma(x_n)] \ .$$

Remark 2.11.16 *In $\mathcal{M}(T)$ above $\cdot$ and $[\![\cdot]\!]$ are well-defined and $\mathcal{M}(T)$ is a model of T.*

2.11.2 A syntax free definition of λ-models (11B)

The above definition of a λ-model makes it difficult to define models, since we have to give a complete definition of $[\![\cdot]\!]$ and verify all its properties. Instead we are going now to give a more abstract and more algebraic definition, on which the model D^∞ we construct (and other models as well) will be based.
The idea is to start with a combinatory algebra $(D, \cdot)$ i.e.

$$(D, \cdot) \text{ is combinatorially complete} \tag{1}$$

Based on it we try to define $[\![M]\!]_\rho$ by induction on M. Our definition should be general, i.e. we want to obtain all possible choices of $[\![\cdot]\!]$ s.t.

$$(D, \cdot, [\![\cdot]\!]) \text{ is a } \lambda\text{-model} \ ,$$

which means that in the course of developping $[\![\cdot]\!]$ we will need to add additional parameters, which determine $[\![\cdot]\!]$ on $(D, \cdot)$. In fact, only one parameter, namely a function $\Lambda : D \to D$ will be needed. This Λ will define $[\![\cdot]\!]$ completely, i.e. we will get that for every choice of $[\![\cdot]\!]$ there exists a unique Λ corresponding to it, which fulfills certain equations and that for every Λ fulfilling these equations there will exists a unique $[\![\cdot]\!]$ based on it.
Case $M \equiv x$. By the conditions of λ-model there is no freedom of choice,

$$[\![M]\!]_\sigma = \sigma(x) \ .$$

Case $M \equiv P\ Q$. If $[\![P]\!]_\sigma$, $[\![Q]\!]_\sigma$ are chosen, there is only one choice possible for $[\![M]\!]_\sigma$, namely

$$[\![P\ Q]\!]_\sigma := [\![P]\!]_\sigma \cdot [\![Q]\!]_\sigma \ .$$

Case $M \equiv \lambda x.P$. If $[\![P]\!]_\rho$ is defined for all ρ we get the condition

$$[\![\lambda x.P]\!]_\sigma \cdot d = [\![P]\!]_{\sigma_x^d} \ ,$$

i.e. $[\![\lambda x.P]\!]_\sigma$ must be one representative of the function

$$\lambda\!\backslash\, d.[\![P]\!]_{\sigma_x^d} \ .$$

First we need to guarantee the existence of such a representative. Now we see immediately that we need to define something more generally, namely for every term M and variables $x_1, \ldots, x_n$ s.t. $\mathrm{FV}(M) \subseteq \{x_1, \ldots, x_n\}$ an element $\mathrm{a}_{M,x_1,\ldots,x_n} \in D$ s.t.

$$[\![M]\!]_\sigma = \mathrm{a}_{M,x_1,\ldots,x_n} \cdot \sigma(x_1) \cdot \cdots \cdot \sigma(x_n) \ .$$

Note that for every choice of $[\![\cdot]\!]$ there might be several choices of $\mathrm{a}_{M,\vec{x}}$. However it is not necessary to consider all such choices, we need only to guarantee that we finally obtain all choices of $[\![\cdot]\!]$.
It will be useful in the following to abbreviate for $\vec{x} = x_1, \ldots, x_n$, and $\vec{a} = a_1, \ldots, a_n$

$$\begin{array}{rcl} b \cdot \sigma(\vec{x}) & := & b \cdot \sigma(x_1) \cdot \cdots \cdot \sigma(x_n) \ , \\ b \cdot \vec{a} & := & b \cdot a_1 \cdot \cdots \cdot a_n \ . \end{array}$$

We will show that we can find $\mathrm{a}_{M,\vec{x}}$ in the cases treated before.
In case $M \equiv x_i$ we could define

$$\mathrm{a}_{M,x_1,\ldots,x_n} := (\lambda x_1, \ldots, x_n.x_i)^*$$

where $(\lambda x_1, \ldots, x_n.x_i)^*$ is a representative of the function

$$\lambda\backslash\, d_1, \ldots, d_n.d_i \ ,$$

which exists by combinatorially completeness of $(D, \cdot)$ (which one we choose does not matter). For convenience, in case the variables x_i occurs more than once in $x_1, \ldots, x_n$ we choose the last such occurrence.
In case $M \equiv P\ Q$ we can define

$$\mathrm{a}_{P\ Q,\vec{x}} := (\lambda u, v, \vec{x}.u\ \vec{x}\ (v\ \vec{x}))^* \cdot \mathrm{a}_{P,\vec{x}} \cdot \mathrm{a}_{Q,\vec{x}} \ ,$$

where $(\lambda u, v, \vec{x}.u\ \vec{x}\ (v\ \vec{x}))^*$ is a representative of the function

$$\lambda\backslash\, a, b, \vec{d}.a \cdot \vec{d} \cdot (b \cdot \vec{d})$$

which again exists by (1).
Now, back to the definition of $[\![\lambda x.P]\!]_\sigma$. If we have defined $\mathrm{a}_{P,\vec{x},x}$, then

$$\mathrm{a}_{P,\vec{x},x} \cdot \sigma(\vec{x})$$

is one representative of the function

$$\lambda d.[\![P]\!]_{\sigma_x^d} (= \lambda d.\mathrm{a}_{P,\vec{x},x} \cdot \sigma(\vec{x}) \cdot d) \ .$$

(Note that because of our choice of interpretation of variables, if $x \in \vec{x}$ the last occurrence of x overrides this occurrence of $x \in \vec{x}$).
However, there are several representatives possible. By Berry's extensionality property we know 2.11.8 that

$$\lambda d.[\![P]\!]_{\sigma_x^d} = \lambda d.[\![Q]\!]_{\rho_y^d} \Rightarrow [\![\lambda x.P]\!]_\sigma = [\![\lambda y.Q]\!]_\rho \ ,$$

which means that

$$\mathrm{a}_{P,\vec{x},x}\cdot\sigma(\vec{x})\sim\mathrm{a}_{Q,\vec{y},y}\cdot\rho(\vec{y})\Rightarrow[\![\,\lambda x.P\,]\!]_\sigma=[\![\,\lambda y.Q\,]\!]_\rho\ ,$$

i.e. in every model of the λ-calculus based on $(D,\cdot)$ there is for every $d\in D$ at most one $\Lambda(d)$ s.t.

$$[\![\,\lambda x.P\,]\!]_\sigma=\Lambda(\mathrm{a}_{P,\vec{x},x}\cdot\sigma(\vec{x}))\ ,$$

and we have

$$d\sim d'\Rightarrow\Lambda(d)=\Lambda(d')\ . \tag{2}$$

On the other hand we have that

$$\mathrm{a}_{x\ y,y,x}\cdot d\cdot a=d\cdot a$$

i.e.

$$\mathrm{a}_{x\ y,y,x}\cdot d\sim d\ ,$$

therefore in every equivalence class modulo $\sim$ of e there exists a d s.t. $\Lambda(d)$ is defined. Therefore Λ can be extended in a unique way to a function

$$\Lambda:D\to D$$

s.t. (2) holds.
Therefore we get that for every semantic $(D,\cdot,[\![\,\cdot\,]\!])$ there exists a unique

$$\Lambda:D\to D$$

fulfilling (2) and s.t.

$$[\![\,\lambda x.P\,]\!]_\sigma=\Lambda(\mathrm{a}_{P,\vec{x},x}\cdot\sigma(\vec{x}))\ , \tag{+}$$

and to every function

$$\Lambda:D\to D$$

fulfilling (2) there exists at most one semantics fulfilling (+). (Note that by (2) and (+) $[\![\,\lambda x.P\,]\!]_\sigma$ is uniquely determined already by $[\![\,P\,]\!]_{\sigma^d_x}$ $(d\in D)$, since for all choices of $\mathrm{a}_{P,\vec{x},x}$ s.t. $\mathrm{a}_{P,\vec{x},x}\cdot\sigma(\vec{x})$ represents $\lambda d.[\![\,M\,]\!]_{\sigma^d_x}$ $\Lambda(\mathrm{a}_{P,\vec{x},x}\cdot\sigma(\vec{x}))$ yields the same result.
What remains is to find out the remaining conditions on Λ needed s.t. it corresponds to a semantics and to determine $\mathrm{a}_{\lambda x.P,\vec{x}}$.
Since

$$[\![\,M\,]\!]_\sigma=\Lambda(\mathrm{a}_{M,\vec{x},x}\cdot\sigma(\vec{x}))\sim\mathrm{a}_{M,\vec{x},x}\cdot\sigma(\vec{x})$$

and for every d $d\sim\mathrm{a}_{x\ y,y,x}\cdot d$, it follows

$$\Lambda(d)\sim\Lambda(\mathrm{a}_{x\ y,y,x})\sim\mathrm{a}_{x\ y,y,x}\sim d$$

i.e.

$$\Lambda(d) \sim d \ . \tag{3}$$

Further we have

$$\begin{array}{rcl} [\![\lambda y.x\ y]\!]_{[x:=d]} \cdot a & = & [\![x\ y]\!]_{[x:=d,y:=a]} \\ & = & d \cdot a \ , \end{array}$$

therefore

$$\begin{array}{rcl} [\![\lambda y.x\ y]\!]_{[x:=d]} & \sim & d \ , \\ [\![\lambda y.x\ y]\!]_{[x:=d]} & = & \Lambda(\mathrm{a}_{x\ y,x,y} \cdot d) \\ & = & \Lambda(d) \\ \Lambda & = & \lambda\!\backslash\, d.[\![\lambda y.x\ y]\!]_{[x:=d]} \\ & = & [\![\lambda x,y.x\ y]\!] \cdot d \end{array}$$

therefore

$$\exists e \in D. \forall d \in D. e \cdot d = \Lambda(d) \ . \tag{4}$$

Once we have e according to (4) we can now define $\mathrm{a}_{\lambda x.P,\vec{x}}$:

$$\begin{array}{rcl} [\![\lambda x.P]\!]_{\sigma} & = & \Lambda(\mathrm{a}_{P,\vec{x},x} \cdot \sigma(\vec{x})) \\ & = & e \cdot (\mathrm{a}_{P,\vec{x},x} \cdot \sigma(\vec{x})) \\ & = & (\lambda u,v,\vec{x}.u\ (v\ \vec{x}))^{*} \cdot e \cdot \mathrm{a}_{P,\vec{x},x} \cdot \sigma(\vec{x}) \ , \end{array}$$

i.e. we can define

$$\mathrm{a}_{\lambda x.P,\vec{x}} := (\lambda u,v,\vec{x}.u\ (v\ \vec{x}))^{*} \cdot e \cdot \mathrm{a}_{P,\vec{x},x} \ ,$$

where $(\lambda u,v,\vec{x}.u\ (v\ \vec{x}))^{*}$ is a representative of the function

$$\lambda\!\backslash\, a,b,\vec{c}.a \cdot (b \cdot \vec{c}) \ .$$

We take now the conditions (1) - (4) as above as the definition of a syntax free λ-model and verify then, although this is already almost implicitly shown in the above discussion, that the new notion is equivalent to the notion of a λ-model, we had before.

Definition 2.11.17 (11.19).
A *syntax free λ-model* or (if this does not cause confusion with the definition above *λ-model*) is a tripel

$$(D,\cdot,\Lambda)$$

where

- $(D,\cdot)$ is an applicative structure,
- $\Lambda : D \to D$,

s.t.

(a) $(D, \cdot)$ is combinatorially complete,

(b) $\forall a \in D.\Lambda(a) \sim a$,

(c) $\forall a, b \in D.(a \sim b \Rightarrow \Lambda(a) = \Lambda(b))$,

(d) $\exists e \in D.\forall a \in D.e \cdot a = \Lambda(a)$.

Definition 2.11.18 (11.20, first part)
Let $(D, \cdot, \Lambda)$ be a syntax free λ-model. We define $[\![\cdot]\!]$ s.t. $(D, \cdot, [\![\cdot]\!])$ is a λ-model as follows:
Let for every combination M of distinct variables $x_1, \ldots, x_n$

$$(\lambda x_1, \ldots, x_n.M)^*$$

be an element in D representing this function.
Let e be s.t.

$$\forall a \in D.e \cdot a = \Lambda(a) \ .$$

First define for every term N, distinct variables $x_1, \ldots, x_n$ s.t.

$$\mathrm{FV}(N) \subseteq \{x_1, \ldots, x_n\}$$

an element

$$\mathrm{a}_{N,x_1,\ldots,x_n} \in D$$

by

(a)

$$\mathrm{a}_{x_i,x_1,\ldots,x_n} := (\lambda x_1, \ldots, x_n.x_i)^*$$

(b)

$$\begin{array}{l} \mathrm{a}_{N\ M,x_1,\ldots,x_n} := \\ (\lambda u, v, x_1, \ldots, x_n.u\ x_1 \cdots x_n\ (v\ x_1 \cdots x_n))^* \\ \quad \cdot \mathrm{a}_{N,x_1,\ldots,x_n} \cdot \mathrm{a}_{M,x_1,\ldots,x_n} \end{array}$$

i.e. $\mathrm{a}_{N\ M,x_1,\ldots,x_n}$ represents the function

$$\lambda\backslash\, d_1, \ldots, d_n.\mathrm{a}_{N,x_1,\ldots,x_n} \cdot d_1 \cdot \cdots \cdot d_n \cdot (\mathrm{a}_{M,x_1,\ldots,x_n} \cdot d_1 \cdot \cdots \cdot d_n) \ .$$

(c)

$$\begin{array}{l} \mathrm{a}_{\lambda x.N,x_1,\ldots,x_n} := \\ (\lambda u, v, x_1, \ldots, x_n.u\ (v\ x_1 \cdots x_n))^* \cdot e \cdot \mathrm{a}_{N,x_1,\ldots,x_n,x} \end{array}$$

i.e. $\mathrm{a}_{\lambda x.N,x_1,\ldots,x_n}$ represents

$$\lambda\backslash\, d_1, \ldots, d_n.e \cdot (\mathrm{a}_{N,x_1,\ldots,x_n,x} \cdot d_1 \cdot d_2 \cdot \cdots \cdot d_n) \ .$$

Then define, if $\mathrm{FV}(M) = \{x_1, \dots, x_n\}$.

$$[\![M]\!]_\sigma := \mathrm{a}_{M,x_1,\dots,x_n} \cdot \sigma(x_1) \cdot \dots \cdot \sigma(x_n) \ .$$

Theorem 2.11.19 *(11.20, second part).*
Definition 2.11.18 yields a λ-model.

Proof:
We first verify that the definition

$$[\![M]\!]_\sigma := \mathrm{a}_{M,x_1,\dots,x_n} \cdot \sigma(x_1) \cdot \dots \cdot \sigma(x_n) \ .$$

is independent of the choice of distinct variables $x_1, \dots, x_n$, i.e. if

$$\mathrm{FV}(M) \subseteq \{x_1, \dots, x_n\} \cap \{y_1, \dots, y_m\}$$

then for all assignments

$$\mathrm{a}_{M,\vec{x}} \cdot \sigma(\vec{x}) = \mathrm{a}_{M,\vec{y}} \cdot \sigma(\vec{y})$$

by induction on M.
Case $M \equiv x_i \equiv y_j$:

$$\mathrm{a}_{M,\vec{x}} \cdot \sigma(\vec{x}) = \sigma(x_i) = \sigma(y_j) = \mathrm{a}_{M,\vec{y}} \cdot \sigma(\vec{y}) \ .$$

Case $M \equiv P\ Q$:

$$\begin{array}{rcl} \mathrm{a}_{M,\vec{x}} \cdot \sigma(\vec{x}) & = & \mathrm{a}_{P,\vec{x}} \cdot \sigma(\vec{x}) \cdot (\mathrm{a}_{Q,\vec{x}} \cdot \sigma(\vec{x})) \\ & \overset{\mathrm{IH}}{=} & \mathrm{a}_{P,\vec{y}} \cdot \sigma(\vec{y}) \cdot (\mathrm{a}_{Q,\vec{y}} \cdot \sigma(\vec{y})) \\ & = & \mathrm{a}_{M,\vec{y}} \cdot \sigma(\vec{x}) \end{array}$$

Case $M \equiv \lambda x.P$:
By IH

$$\forall d \in D.\mathrm{a}_{P,\vec{x},x} \cdot \sigma(\vec{x}) \cdot d = \mathrm{a}_{P,\vec{y},x} \cdot \sigma(\vec{y}) \cdot d \ ,$$

therefore

$$\Lambda(\mathrm{a}_{P,\vec{x},x} \cdot \sigma(\vec{x})) = \Lambda(\mathrm{a}_{P,\vec{y},x} \cdot \sigma(\vec{y}))$$

and

$$\begin{array}{rcl} \mathrm{a}_{M,\vec{x},x} \cdot \sigma(\vec{x}) & = & e \cdot (\mathrm{a}_{P,\vec{x},x} \cdot \sigma(\vec{x})) \\ & = & \Lambda(\mathrm{a}_{P,\vec{x},x} \cdot \sigma(\vec{x})) \\ & = & \Lambda(\mathrm{a}_{P,\vec{y},x} \cdot \sigma(\vec{y})) \\ & = & e \cdot (\mathrm{a}_{P,\vec{y},x} \cdot \sigma(\vec{y})) \\ & = & \mathrm{a}_{M,\vec{y},x} \cdot \sigma(\vec{y}) \ . \end{array}$$

A similar proof (both assertions need to be shown simultaneously) shows

- $M \equiv_\alpha N \Rightarrow \mathrm{a}_{M,\vec{x}} = \mathrm{a}_{N,\vec{x}}$

- If $\mathrm{FV}(M) \subseteq \{x_1, \ldots, x_n\}$, y_i distinct, then

$$\mathrm{a}_{M,\vec{x}} = \mathrm{a}_{M[x_1:=y_1,\ldots,x_n:=y_n],\vec{y}} \ .$$

Now we verify that we obtain a λ-model:

$$[\![x]\!]_\sigma = \sigma(x)$$

is obvious.

$$[\![P\, Q]\!]_\sigma = [\![P]\!]_\sigma \cdot [\![Q]\!]_\sigma \ ,$$

follows by definition.

$$[\![\lambda x.P]\!]_\sigma \cdot d = [\![P]\!]_{\sigma^d_x}$$

$$\begin{array}{rcl} [\![\lambda x.P]\!]_\sigma \cdot d & = & \Lambda(\mathrm{a}_{P,\vec{x},x} \cdot \sigma(\vec{x})) \cdot d \\ & \overset{\Lambda(a)\sim a}{=} & \mathrm{a}_{P,\vec{x},x} \cdot \sigma(\vec{x}) \cdot d \\ & = & [\![P]\!]_{\sigma^d_x} \ . \end{array}$$

$$[\![M]\!]_\sigma = [\![N]\!]_\sigma \text{ if } \sigma \restriction \mathrm{FV}(M) = \sigma \restriction \mathrm{FV}(N)$$

is guaranteed by definition.

$$[\![\lambda x.M]\!]_\sigma = [\![\lambda y.(M[x := y])]\!]_\sigma, \text{ if } y \notin \mathrm{FV}(M)$$

has been verified above.

$$\lambda\backslash\, d.[\![M]\!]_{\sigma^d_x} \sim \lambda\backslash\, d.[\![N]\!]_{\sigma^d_x} \Rightarrow [\![\lambda x.M]\!]_\sigma = [\![\lambda x.N]\!]_\sigma \ .$$

By assumption

$$\mathrm{a}_{M,\vec{x},x} \cdot \sigma(\vec{x}) \sim \mathrm{a}_{N,\vec{x},x} \cdot \sigma(\vec{x})$$

therefore

$$[\![\lambda x.M]\!]_\sigma = \Lambda(\mathrm{a}_{M,\vec{x},x} \cdot \sigma(\vec{x})) = \Lambda(\mathrm{a}_{N,\vec{x},x} \cdot \sigma(\vec{x})) = [\![\lambda x.N]\!]_\sigma \ .$$

Theorem 2.11.20 *(11.20, third part)*
Let $(D, \cdot, [\![\cdot]\!])$ be a λ-model,

$$\Lambda := \lambda\backslash\, d.[\![\lambda x.y\ x]\!]_{[y:=d]} \ .$$

Then $(D, \cdot, \Lambda)$ is a syntax free λ-model.

Proof: We verify the condition in Definition 2.11.4 (a) - (d):
(a): $(D, \cdot)$ is combinatorially complete.
(b):

$$\begin{array}{rcl} \Lambda(a) \cdot d & = & [\![\lambda x.y\ x]\!]_{[y:=a]} \cdot d \\ & = & [\![y\ x]\!]_{[y:=a,x:=d]} \\ & = & a \cdot d \end{array}$$

therefore

$$\Lambda(a) \sim a \ .$$

(c):
If $a \sim b$, then

$$\begin{aligned} [\![\, \lambda x.y\ x \,]\!]_{[y:=a]} \cdot d &= [\![\, y\ x \,]\!]_{[y:=a,x:=d]} \\ &= a \cdot d \\ &= b \cdot d \\ &= [\![\, \lambda x.y\ x \,]\!]_{[y:=b]} \cdot d \end{aligned}$$

therefore

$$\Lambda(a) = [\![\, \lambda x.y\ x \,]\!]_{[y:=a]} = [\![\, \lambda x.y\ x \,]\!]_{[y:=b]} = \Lambda(b) \ .$$

(d):
Let $e := [\![\, \lambda x, y.y\ x \,]\!]$. Then for all $d \in D$

$$\begin{aligned} e \cdot d &= [\![\, \lambda y, x.y\ x \,]\!] \cdot d \\ &= [\![\, \lambda x.y\ x \,]\!]_{[y:=d]} \\ &= \Lambda(d) \ . \end{aligned}$$

Theorem 2.11.21 *The constructions in Definition 2.11.18 and Theorem 2.11.20 are inverse.*

Proof:
Assume first a syntax free λ-modul $(D, \cdot, \Lambda)$, let $[\![\cdot]\!]$ be defined as in Definition 2.11.18. Show, that the Λ' obtained in Theorem 2.11.20 from $(D, \cdot, [\![\cdot]\!])$ is Λ:

$$\Lambda'(d) = [\![\, \lambda x.y\ x \,]\!]_{[y:=d]} = \Lambda(\mathrm{a}_{y\ x,y,x} \cdot d) \ .$$

$$a_{y\ x,y,x} \cdot d \cdot e = d \cdot e \ ,$$

therefore

$$a_{y\ x,y,x} \cdot d \sim d \ ,$$

$$\Lambda'(d) = \Lambda(\mathrm{a}_{y\ x,y,x} \cdot d) = \Lambda(d) \ .$$

Assume now a free λ-modul $(D, \cdot, [\![\cdot]\!])$, let Λ be defined as in Theorem 2.11.20. Show, that the $[\![\cdot]\!]'$ obtained in Definition 2.11.18 from $(D, \cdot, \Lambda)$ is $[\![\cdot]\!]$. Show $[\![\, M \,]\!]_\sigma = [\![\, M \,]\!]'_\sigma$ by induction on M:

- $M \equiv x$: both sides yield $\sigma(x)$.
- $M \equiv P\ Q$: IH.
- $M \equiv \lambda x.N$:

$$\begin{aligned} [\![\, \lambda x.M \,]\!]'_\sigma &= \Lambda(\mathrm{a}_{M,\vec{x},x} \cdot \sigma(\vec{x})) \\ &= [\![\, \lambda y.x\ y \,]\!]_{[x:=\mathrm{a}_{M,\vec{x},x} \cdot \sigma(\vec{x})]} \ . \end{aligned}$$

Now for all $d \in D$

$$\begin{array}{rcl} [\![\, x\ y \,]\!]_{[x:=\mathrm{a}_{M,\vec{x},x}\cdot\sigma(\vec{x})]^d_y} & = & \mathrm{a}_{M,\vec{x},x}\cdot\sigma(\vec{x})\cdot d \\ & \overset{\mathrm{IH}}{=} & [\![\, M \,]\!]_{\sigma^d_x}\ , \end{array}$$

therefore

$$\begin{array}{rcl} [\![\, \lambda x.M \,]\!]'_\sigma & = & [\![\, \lambda y.x\ y \,]\!]_{[x:=\mathrm{a}_{M,\vec{x},x}\cdot\sigma(\vec{x})]} \\ & = & [\![\, \lambda x.M \,]\!]_\sigma\ . \end{array}$$

Theorem 2.11.22 *(11.30)*
If $(D,\cdot)$ is an extensional combinatory algebra, and let

$$\Lambda : D \rightarrow D, a \mapsto a\ .$$

Then $(D,\cdot,\Lambda)$ is a syntax free λ-model and the only one extending $(D,\cdot)$.

Proof: Uniqueness follows, since every Λ has to select out of every equivalence class modolu $\sim$ one element. The equivalence classes have one element only, therefore $\Lambda(a) = a$.
With Λ as in the theorem follows immediately $\Lambda(a) \sim a$, $a \sim b \Rightarrow \Lambda(a) \sim \Lambda(b)$ and with $\mathbf{e}$ representing the identity we get $\mathbf{e}$ represents Λ.

2.11.3 Scott-Meyer λ-models (11.21 - 11.27)

Definition 2.11.17 can now be axiomatized as follows:

Definition 2.11.23 (11.21)
Syntax free λ-models can be formalized by the following set of axioms in the language with a binary function symbol $\cdot$, written infix, and a unary function symbol Λ:

(a) $\exists k.\forall a,b.k\cdot a\cdot b = a$.

(b) $\exists s.\forall a,b,c.s\cdot a\cdot b\cdot c = (a\cdot c)\cdot(b\cdot c)$.

(c) $\forall a,b.\Lambda(a)\cdot b = a\cdot b$.

(d) $\forall a,b.((\forall d.a\cdot d = b\cdot d)\rightarrow\Lambda(a) = \Lambda(b))$.

(e) $\exists e.\forall a.e\cdot a = \Lambda(a)$.

From the last axiom one can see, that we can replace $\Lambda(a)$ by $\mathbf{e}\cdot a$ for some constant $\mathbf{e}$. However, whereas to every λ-model corresponds a unique Λ s.t. we get a syntax free λ-model, there might be several $\mathbf{e}$ which represent the same Λ. Only, if we add that $\mathbf{e}$ is strict, where strict is defined as $\mathbf{e}\cdot\mathbf{e} = \mathbf{e}$, we get uniqueness. (Note that $\mathbf{e}$ strict in terms of Λ means $\Lambda(\mathbf{e}) = \mathbf{e}$, i.e. $\mathbf{e}$ is the "canonical choice").

Definition 2.11.24 (11.22)

(a) Let $(D, \cdot)$ be an applicative structure, $\mathbf{e} \in D$. $(D, \cdot, \mathbf{e})$ is called a *loose Scott-Meyer λ-model*, iff

  (a) $(D, \cdot)$ is combinatorially complete.

  (b) $\forall a, b \in D.\mathbf{e} \cdot a \cdot b = a \cdot b$.

  (c) $\forall a, b \in D((\forall d.a \cdot d = b \cdot d) \rightarrow \mathbf{e} \cdot a = \mathbf{e} \cdot b)$.

(b) A loose Scott-Meyer model $(D, \cdot, \mathbf{e})$ is a *strict Scott-Meyer λ-model* iff

$$\mathbf{e} \cdot \mathbf{e} = \mathbf{e} \ .$$

Lemma 2.11.25 *(approx. 11.23)*

(a) If $(D, \cdot, \mathbf{e})$, $(D, \cdot, \mathbf{e}_0)$ are two strict Scott-Meyer model (with the same applicative structure $(D, \cdot)$) s.t. $\mathbf{e} \sim \mathbf{e}_0$, then $\mathbf{e} = \mathbf{e}_0$.

(b) If $(D, \cdot, \mathbf{e})$ is a Scott-Meyer model, $\mathbf{e}_0 := \mathbf{e} \cdot \mathbf{e}$, then $(D, \cdot, \mathbf{e}_0)$ is a strict Scott meyer model s.t. $\mathbf{e}_0 \sim \mathbf{e}$ (i.e. $\mathbf{e}_0$ corresponds to the same Λ as $\mathbf{e}$).

Proof: (a).

$$\begin{array}{rcl} \mathbf{e} & = & \mathbf{e} \cdot \mathbf{e} \\ & = & \Lambda(\mathbf{e}) \\ & \stackrel{\mathbf{e} \sim \mathbf{e}_0}{=} & \Lambda(\mathbf{e}_0) \\ & = & \mathbf{e}_0 \cdot \mathbf{e}_0 \\ & = & \mathbf{e}_0 \ . \end{array}$$

(b): $\Lambda(a) := \mathbf{e} \cdot a$, then $\mathbf{e}_0 = \Lambda(\mathbf{e}) \sim \mathbf{e}$,

$$\mathbf{e}_0 \cdot \mathbf{e}_0 = \Lambda(\Lambda(\mathbf{e})) = \Lambda(\mathbf{e}) = \mathbf{e}_0 \ .$$

Theorem 2.11.26 *(a) Let $(D, \cdot, \Lambda)$ be a syntax free λ-model, $(D, \cdot, [\![\cdot]\!])$ the corresponding λ-model. Let*

$$\mathbf{e} := [\![\lambda y, x.y \ x]\!] (= [\![\underline{1}]\!]) \ .$$

Then $(D, \cdot, \mathbf{e})$ is a strict Scott-Meyer λ-model.

(b) If $(D, \cdot, \mathbf{e})$ is a loose Scott-Meyer λ-model, $\Lambda : D \rightarrow D$ defined by

$$\Lambda(d) := \mathbf{e} \cdot d \ .$$

Then $(D, \cdot, \Lambda)$ is a syntax free λ-model.

(c) The constructions in (a) and (b) are inverse bijections, if (b) is restricted to strict Scott-Meyer models.

(d) If we apply the construction (b) and then (a) to a loose Scott-Meyer model $(D, \cdot, \mathbf{e})$, we obtain a strict Scott-Meyer model $(D, \cdot, \mathbf{e}_0)$ s.t. $\mathbf{e}_0 \sim \mathbf{e}$.

Proof:
(a) By Theorem 2.11.21 it follows that

$$\Lambda(a) = [\![\, \lambda x.y\ x \,]\!]_{[y:=d]} \ ,$$

therefore

$$\Lambda(a) = [\![\, \lambda x, y, x\ y \,]\!] \cdot d,$$

and $\mathbf{e} := [\![\, \underline{1} \,]\!]$ fulfills the conditions of a loose Scott-Meyer model. Further for this $\mathbf{e}$ it follows

$$\begin{aligned} \mathbf{e} \cdot \mathbf{e} &= [\![\, \lambda x, y.x\ y \,]\!] \cdot [\![\, \lambda x, y.x\ y \,]\!] \\ &= [\![\, (\lambda x, y.x\ y) \lambda x, y.x\ y \,]\!] \\ &= [\![\, \lambda x, y.x\ y \,]\!] \\ &= \mathbf{e} \ , \end{aligned}$$

therefore $(D, \cdot, \mathbf{e})$ is actually strict.
(b): trivial.
(c): If we apply (a) and then (b), we obtain the syntax free λ-model $(D, \cdot, \Lambda)$, with

$$\Lambda'(a) = e \cdot a = [\![\, \lambda x.y\ x \,]\!]_{[y:=d]} \ ,$$

where by Theorem 2.11.21 $\Lambda(a) = \Lambda'(a)$.
If we apply (b) and then (a), we obtain a strict Scott-Meyer model

$$(D, \cdot, \mathbf{e}_0)$$

s.t. $\mathbf{e}_0$ represents Λ which is

$$a \mapsto \mathbf{e} \cdot a$$

for the $\mathbf{e}$ of the original Scott-Meyer model. Therefore

$$\mathbf{e} \sim \mathbf{e}_0 \ .$$

From this it follows (d), and if we started with a strict Scott-Meyer model by Lemma 2.11.25 it follows

$$\mathbf{e} = \mathbf{e}_0 \ .$$

Remark 2.11.27 *(11.26) Let $(D, \cdot, \Lambda)$ be a syntax free λ-model, define*

$$\mathrm{Repu} : (D \rightarrow_{\mathrm{rep}} D) \rightarrow D, \quad \mathrm{Repu}(f) = \Lambda(a) \textit{ for } a \in \mathrm{Rep}(f) \ .$$

Since for $b, c \in \mathrm{Rep}(f)$, $b \sim c$ it follows that Repu *is well-defined and*

$$\mathrm{Repu} \circ \mathrm{Fun} \textit{ is the identity on } D \rightarrow_{\mathrm{rep}} D \ .$$

Therefore Repu *is a left inverse to* Fun *and $D \rightarrow_{\mathrm{rep}} D$ is a retract of D.*
The image of Repu *is a subset F of D which contains of every equivalenceclass of $\sim$ exactly one element.*

2.12 The λ-model D_∞ (12)

2.12.1 Solutions of cpo-equations

We will in the following assume familiarity with cpo's, as described in 12A and 12B of [HS86].
The model D_∞ we are going to construct will be a non-trivial cpo s.t.

$$\mathrm{D}_\infty \cong [\mathrm{D}_\infty \to \mathrm{D}_\infty] \ .$$

Let $\alpha : \mathrm{D}_\infty \to [\mathrm{D}_\infty \to \mathrm{D}_\infty]$ be the isomorphism. Then we can define for $a, b \in \mathrm{D}_\infty$

$$a \cdot b := \alpha(a)(b) \ .$$

We will then verify that

$$(\mathrm{D}_\infty, \cdot)$$

is an extensional combinatory algebra (the extensionality is trivial, because α is an isomorphism), which can be (by defining $\Lambda := \mathrm{id}$) extended by Theorem 2.11.22 to a λ-model.
The construction generalizes very easily to solutions of more general cpo-equations, and instead of restricting ourselves to the special case we treat arbitrary solutions of cpo-equations. There is even a more general version of this ([Set94, Str94]), which works for all cpo-enriched categories, where a category $\mathcal{C}$ is cpo-enriched iff $\mathcal{C}(A, B)$ have a cpo structure s.t.

$$\circ : \mathcal{C}(B, C) \times \mathcal{C}(A, B) \to \mathcal{C}(A, C) \text{ is continuous.}$$

We restrict ourselves to ordinary cpos. The following is based on [Set94], which is again based on [Str94].
First we need some extremely basic category theory:

Definition 2.12.1 (a) A *category* is a 6-tupel

$$\mathcal{C} := (\mathrm{Obj}, \mathrm{Mor}, \mathrm{dom}, \mathrm{cod}, \circ, \mathrm{id})$$

s.t.

- Obj, Mor are classes (the elements of Obj are called *objects* and the elements of Mor *morphisms of the category* $\mathcal{C}$),
- $\mathrm{dom}, \mathrm{cod} : \mathrm{Mor} \to \mathrm{Obj}$,
- $f \circ g \in \mathrm{Mor}$ is defined for all $f, g \in \mathrm{Mor}$ s.t. $\mathrm{cod}(g) = \mathrm{dom}(f)$,
- $\mathrm{id} : \mathrm{Obj} \to \mathrm{Mor}$,

where we write, if $f \in \mathrm{Mor}$, $A, B \in \mathrm{Obj}$,

$$f : A \to B :\Leftrightarrow \mathrm{dom}(f) = A \wedge \mathrm{cod}(f) = B \ ,$$

and id_A for $\mathrm{id}(A)$ and have the following laws:

- $\mathrm{dom}(f \circ g) = \mathrm{dom}(g)$,
- $\mathrm{cod}(f \circ g) = \mathrm{cod}(f)$,
- $\mathrm{dom}(\mathrm{id}_A) = \mathrm{cod}(\mathrm{id}_A) = A$,
- $h \circ (g \circ f) = (h \circ g) \circ f$ (if defined),
- if $h : A \rightarrow B$, then $\mathrm{id}_B \circ h = h$, $h \circ \mathrm{id}_A = h$,

Let $\mathcal{C}(A, B) := \{f \in \mathrm{Mor} \mid f : A \rightarrow B\}$.

(b) A bifunctor from a category

$$\mathcal{C} = (\mathrm{Obj}_{\mathcal{C}}, \mathrm{Mor}_{\mathcal{C}}, \mathrm{dom}_{\mathcal{C}}, \mathrm{cod}_{\mathcal{C}}, \circ_{\mathcal{C}}, \mathrm{id}^{\mathcal{C}})$$

into a category

$$\mathcal{D} = (\mathrm{Obj}_{\mathcal{D}}, \mathrm{Mor}_{\mathcal{D}}, \mathrm{dom}_{\mathcal{D}}, \mathrm{cod}_{\mathcal{D}}, \circ_{\mathcal{D}}, \mathrm{id}^{\mathcal{D}}) \ ,$$

abbreviated by

$$F : (\mathcal{C}^{\mathrm{op}} \times \mathcal{C}) \rightarrow \mathcal{D} \ ,$$

is a function

$$F : \mathrm{Obj}_{\mathcal{C}} \times \mathrm{Obj}_{\mathcal{C}} \rightarrow \mathrm{Obj}_{\mathcal{C}}$$

together with for every every $A, A', B, B' \in \mathrm{Obj}_{\mathcal{C}}$, and $f : A' \rightarrow A$, $g : B \rightarrow B'$ a morphism in $\mathcal{D}$

$$F(f, g) : F(A, B) \rightarrow F(A', B')$$

(where we use the same name for $F(A, B)$ and $F(f, g)$ s.t.

- $F(\mathrm{id}_A, \mathrm{id}_B) = \mathrm{id}_{F(A,B)}$,
- if $f : A' \rightarrow A$, $f' : A'' \rightarrow A'$, $g : B \rightarrow B'$, $g' : B' \rightarrow B''$ then

$$F(f', g') \circ F(f, g) = F(f \circ f', g' \circ g) \ .$$

Definition 2.12.2 (a) The *category of cpo's* cpo consists of

- Obj: The class of cpo's.
- Mor: The class of pairs (f, B), where $f \in [\mathrm{dom}(f) \rightarrow B]$. $\mathrm{dom}(f, B) = \mathrm{dom}(f)$, $\mathrm{cod}(f, B) = B$. We will usually write f instead of (f, B).
- $(g, C) \circ (f, B) := (g \circ f, C)$.
- $\mathrm{id}_A := (\lambda\backslash\, d \in A.A, A)$.

(b) A bifunctor $F : (\mathrm{cpo}^{\mathrm{op}} \times \mathrm{cpo}) \rightarrow \mathrm{cpo}$ is *locally continuous*, iff for all cpo's A, A', B, B'

$$F : [A' \rightarrow A] \times [B \rightarrow B'] \rightarrow [F(A, B) \rightarrow F(A', B')]$$

is continuous.

Lemma 2.12.3 *(a) The bifunctor*

$$F : (\mathrm{cpo}^{\mathrm{op}} \times \mathrm{cpo}) \rightarrow \mathrm{cpo}$$

defined by

$$F(A, B) = B, \quad F(f, g) = g$$

is a locally continuous bifunctor.

(b) If

$$F, G : (\mathrm{cpo}^{\mathrm{op}} \times \mathrm{cpo}) \rightarrow \mathrm{cpo}$$

are locally continuous bifunctors, then

$$(F \rightarrow G) : (\mathrm{cpo}^{\mathrm{op}} \times \mathrm{cpo}) \rightarrow \mathrm{cpo}$$

is as well a locally continous bifunctor, where $F \rightarrow G$ is defined by

$$(F \rightarrow G)(A, B) := [F(B, A) \rightarrow G(A, B)] \ ,$$

and if

$$f : A' \rightarrow A, \quad g : B \rightarrow B', \quad h : F(B, A) \rightarrow G(A, B)$$

then

$$(F \rightarrow G)(f, g)(h) := G(f, g) \circ h \circ F(g, f) : F(B', A') \rightarrow G(A', B') \ .$$

(c) If $(F_i) : (\mathrm{cpo}^{\mathrm{op}} \times \mathrm{cpo}) \rightarrow \mathrm{cpo}$ are locally continuous bifunctors ($i \in I$), so are

$$\prod_{i \in I} F_i : (\mathrm{cpo}^{\mathrm{op}} \times \mathrm{cpo}) \rightarrow \mathrm{cpo} \ ,$$

defined by

$$(\prod_{i \in I} F_i)(A, B) := \prod_{i \in I} (F_i(A, B)) \ ,$$

with componentwise ordering,

$$(\prod_{i \in I} F_i)(f, g)((a_i)_{i \in I}) = (F_i(f, g)(a_i))_{i \in I} \ .$$

and

$$\sum_{i \in I} F_i : (\mathrm{cpo}^{\mathrm{op}} \times \mathrm{cpo}) \rightarrow \mathrm{cpo} \ ,$$

defined by

$$(\sum_{i \in I} F_i)(A, B) := \sum_{i \in I} (F_i(A, B)) \ ,$$

(disjoint union, with an additional $\perp$ added)

$$(\sum_{i\in I} F_i)(f,g)(\iota_i(a)) = \iota_i(F_i(f,g)(a))$$

(where ι_i are the canonical injections

$$F_i(A,B)\to\sum_{i\in I} F_i(A,B)\ \ ),$$

$$(\sum_{i\in I} F_i)(f,g)(\bot) := \bot\ \ .$$

Proof: All easy.

Definition 2.12.4 Let D, D' be cpo's.
An *embedding/projection pair* is a pair

$$(e : D\to D', p : D'\to D)$$

where

- D, D' are cpo's,
- $e : D\to D'$, $p : D'\to D$ are continuous functions, s.t.
- $p \circ e = \mathrm{id}$,
- $e \circ p \sqsubseteq \mathrm{id}$.

Remark 2.12.5 *(a) If (e,p), (e,p') are embedding/projection pairs, then $p = p'$.*

(b) If (e,p), (e',p) are embedding/projection pairs, then $e = e'$.

(c) If (e,p) is an embedding/projection pair, then e, p are strict.

Bevis: (a):

$$p = \mathrm{id}\circ p = p'\circ e\circ p \sqsubseteq p'\circ \mathrm{id} = p'\ \ ,$$

similarly

$$p' \sqsubseteq p\ \ .$$

(b):

$$e = e\circ p\circ e' \sqsubseteq \mathrm{id}\circ e' = e'\ \ ,$$

similarly

$$e' \sqsubseteq e\ \ .$$

(c) $p(\bot) \sqsubseteq p(e(\bot)) = \bot$, therefore

$$p(\bot) = \bot\ \ .$$

$e(\bot) = e(p(\bot)) \sqsubseteq \mathrm{id}(\bot) = \bot$,

$$e(\bot) = \bot\ \ .$$

Lemma 2.12.6 *Let $(D_n)_{n\in\mathbb{N}}$ be a sequence of cpo's,*

$$(\mathrm{e}_n : D_n \to D_{n+1}, \mathrm{p}_n : D_{n+1} \to D_n)$$

a sequence of embedding/projection pairs. Then there exists a cpo D and a sequence of embedding/projection pairs

$$(\mathrm{i}_n : D_n \to D, \rho_n : D \to D_n) \ ,$$

s.t.

(a) $\mathrm{i}_{n+1} \circ \mathrm{e}_n = \mathrm{i}_n$.

(b) $\sqcup_{n\in\mathbb{N}} \mathrm{i}_n \circ \rho_n = \mathrm{id}_D$.

Proof: Let

$$D := \{u \in \prod_{n\in\mathbb{N}} D_n \mid \forall n \in \mathbb{N}.\mathrm{p}_n(u_{n+1}) = u_n\} \ .$$

D is a cpo with componentwise ordering:

$$\perp_D = ((\perp_{D_n})_{n\in\mathbb{N}}) \in D \ ,$$

since by strictness of p_n

$$\mathrm{p}_n(\perp_{D_{n+1}}) = \perp_{D_n} \ .$$

Assume $B \subseteq D$ directed,

$$u_n := \sqcup\{v_n | v \in B\} \ .$$

Then

$$\begin{array}{rcl} \mathrm{p}_n(u_{n+1}) & = & \mathrm{p}_n(\sqcup\{v_{n+1}|v \in B\}) \\ & = & \sqcup\{\mathrm{p}_n(v_{n+1})|v \in B\} \\ & = & \sqcup\{v_n|v \in B\} = u_n \ , \end{array}$$

therefore $u \in D$.
Further u ist the supremum of B in $\prod_{n\in\mathbb{N}} D_n$, therefore as well in D.
Let for $n < m$

$$\begin{array}{rcl} \mathrm{e}_{n,m} & := & \mathrm{e}_{m-1} \circ \cdots \circ \mathrm{e}_n : D_n \to D_m \\ \mathrm{p}_{m,n} & := & \mathrm{p}_n \circ \cdots \circ \mathrm{p}_{m-1} : D_m \to D_n \end{array}$$

and $\mathrm{e}_{n,n} := \mathrm{p}_{n,n} := \mathrm{Id}_{D_n}$.
Define $\rho_n : D \to D_n$, $\rho_n(u) := u_n$.
Define $\mathrm{i}_n : D_n \to D$,

$$(\mathrm{i}_n(a))_m := \begin{cases} \mathrm{e}_{n,m}(a) & \text{if } n \leq m \\ \mathrm{p}_{n,m}(a) & \text{if } m \leq n \end{cases} \ .$$

$\mathrm{i}_n(a) \in D$:

If $n < m$ then

$$\begin{aligned}
\mathrm{p}_m((\mathrm{i}_n(a))_{m+1}) &= \mathrm{p}_m(\mathrm{e}_{n,m+1}(a)) \\
&= \mathrm{p}_m \circ \mathrm{e}_m \circ \mathrm{e}_{m-1} \circ \cdots \circ \mathrm{e}_n(a) \\
&= \mathrm{e}_{m-1} \circ \cdots \circ \mathrm{e}_n(a) = \mathrm{e}_{n,m}(a) = (\mathrm{i}_n(a))_m \quad .
\end{aligned}$$

If $m \leq n$ then

$$\begin{aligned}
\mathrm{p}_m((\mathrm{i}_n(a))_{m+1}) &= \mathrm{p}_m(\mathrm{p}_{n,m+1}(a)) \\
&= \mathrm{p}_m \circ \mathrm{p}_{m+1} \circ \mathrm{p}_{m+2} \circ \cdots \circ \mathrm{p}_{n-1}(a) \\
&= \mathrm{p}_{n,m}(a) \\
&= (\mathrm{i}_n(a))_m \quad .
\end{aligned}$$

ρ_n is continuous, since it is monotone and for $A \subseteq D$ directed

$$\rho_n(\sqcup A) = u_n = (\sqcup A)_n$$

with $u := \sqcup A$.
i_n is continuous, since it is monotone and for $A \subseteq D$ directed

$$(\mathrm{i}_n(\sqcup A))_m = \mathrm{e}_{n,m}(\sqcup A) = \sqcup \mathrm{e}_{n,m}(A) = \sqcup(\mathrm{i}_n(A)_m) = (\sqcup \mathrm{i}_n(A))_m \quad ,$$

if $n \leq m$ and

$$(\mathrm{i}_n(\sqcup A))_m = \mathrm{p}_{n,m}(\sqcup A) = \sqcup \mathrm{p}_{n,m}(A) = \sqcup(\mathrm{i}_n(A)_m) = (\sqcup \mathrm{i}_n(A))_m \quad ,$$

if $m \leq n$.
Further

$$\begin{aligned}
(\mathrm{i}_{n+1} \circ \mathrm{e}_n(u))_m &= \begin{cases} \mathrm{e}_{n+1,m} \circ \mathrm{e}_n(u) & n+1 \leq m \\ \mathrm{p}_{n+1,m} \circ \mathrm{e}_n(u) & m \leq n \end{cases} \\
&= \begin{cases} \mathrm{e}_{n,m}(u) & n+1 \leq m \\ \mathrm{p}_{n,m}(u) & m \leq n \end{cases} \\
&= (\mathrm{i}_n(u))_m \quad , \\
\mathrm{i}_{n+1} \circ \mathrm{e}_n &= \mathrm{i}_n \quad .
\end{aligned}$$

For $n \geq m$ and $u \in D$ it follows

$$\begin{aligned}
((\mathrm{i}_n \circ \rho_n)(u))_m &= \mathrm{p}_{n,m}(u_n) \\
&= \mathrm{p}_m \circ \cdots \circ \mathrm{p}_{n-1}(u_n) \\
&= u_m \quad .
\end{aligned}$$

Therefore

$$\begin{aligned}
(\sqcup(\mathrm{i}_n \circ \rho_n)(u)))_m &= \sqcup_{n \geq m}(\mathrm{i}_n \circ \rho_n(u))_m \\
&= \sqcup_{n \geq m} u_m = u_m \quad , \\
\sqcup(\mathrm{i}_n \circ \rho_n) &= \mathrm{id} \quad .
\end{aligned}$$

Remark 2.12.7 *Let $D_n, \mathrm{e}_n, \mathrm{p}_n, d, \mathrm{i}_n, \rho_n$ as in Theorem 2.12.6. Then it follows*

(a) $\mathrm{e}_n = \rho_{n+1} \circ \mathrm{i}_n$.

(b) $\mathrm{p}_n = \rho_n \circ \mathrm{i}_{n+1}$.

(c) $\rho_n = \mathrm{p}_n \circ \rho_{n+1}$.

(d) $\mathrm{i}_{n+1} \circ \rho_{n+1} \sqsupseteq \mathrm{i}_n \circ \rho_n$.

(e) $\mathrm{e}_{n,m} = \rho_m \circ \mathrm{i}_n$ *$(n \leq m)$.*

(f) $\mathrm{p}_{n,m} = \rho_m \circ \mathrm{i}_n.$*$(m \leq n)$.*

Proof:
(a):

$$\begin{array}{rcl} \mathrm{e}_n & = & \rho_{n+1} \circ \mathrm{i}_{n+1} \circ \mathrm{e}_n \\ & = & \rho_{n+1} \circ \mathrm{i}_n \end{array}$$

(b):

$$\begin{array}{rcl} (\rho_n \circ \mathrm{i}_{n+1}) \circ \mathrm{e}_n & = & \rho_n \circ (\mathrm{i}_{n+1} \circ \mathrm{e}_n) \\ & = & \rho_n \circ \mathrm{i}_n = \mathrm{id}_{D_n} \ . \\ \mathrm{e}_n \circ (\rho_n \circ \mathrm{i}_{n+1}) & = & \rho_{n+1} \circ \mathrm{i}_n \circ \rho_n \circ \mathrm{i}_{n+1} \\ & \sqsubseteq & \rho_{n+1} \circ \mathrm{i}_{n+1} \\ & = & \mathrm{id} \ . \end{array}$$

Therefore $(\mathrm{e}_n, \rho_n \circ \mathrm{i}_{n+1})$ is an embedding/projection pair. By uniqueness it follows

$$\rho_n \circ \mathrm{i}_{n+1} = \mathrm{p}_n \ .$$

(c):

$$\begin{array}{rcl} (\mathrm{p}_n \circ \rho_{n+1}) \circ \mathrm{i}_n & = & \mathrm{p}_n \circ \rho_{n+1} \circ \mathrm{i}_{n+1} \circ \mathrm{e}_n \\ & = & \mathrm{p}_n \circ \mathrm{e}_n \\ & = & \mathrm{id} \ , \\ \mathrm{i}_n \circ (\mathrm{p}_n \circ \rho_{n+1}) & = & \mathrm{i}_n \circ \rho_n \circ \mathrm{i}_{n+1} \circ \rho_{n+1} \\ & \sqsubseteq & \mathrm{id} \circ \mathrm{id} \\ & = & \mathrm{id} \ . \end{array}$$

Therefore

$$\mathrm{p}_n \circ \rho_{n+1} = \rho_n \ .$$

(d):

$$\begin{array}{rcl} \mathrm{i}_{n+1} \circ \rho_{n+1} & = & \mathrm{i}_{n+1} \circ \mathrm{id} \circ \rho_{n+1} \\ & \sqsupseteq & \mathrm{i}_{n+1} \circ \mathrm{e}_n \circ \mathrm{p}_n \circ \rho_{n+1} \\ & = & \mathrm{i}_n \circ \rho_n \ . \end{array}$$

(e): We show immediately by induction for $n \leq m$

$$\mathrm{i}_m \circ \mathrm{e}_{n,m} = \mathrm{i}_n \ ,$$

therefore

$$\begin{array}{rcl} \mathrm{e}_{n,m} & = & \mathrm{e}_{m-1} \circ \mathrm{e}_{n,m-1} \\ & = & \rho_m \circ \mathrm{i}_{m-1} \circ \mathrm{e}_{n,m-1} \\ & = & \rho_m \circ \mathrm{i}_n \ , \text{ for } m > n \text{ und} \\ \mathrm{e}_{n,n} & = & \rho_n \circ \mathrm{i}_n \ . \end{array}$$

(f): For $n > m$ we have

$$\begin{array}{rcl} \mathrm{p}_{n,m} & = & \mathrm{p}_m \circ \cdots \circ \mathrm{p}_{n-1} \\ & = & \mathrm{p}_m \circ \cdots \circ \mathrm{p}_{n-3} \circ \mathrm{p}_{n-2} \circ \rho_{n-1} \circ \mathrm{i}_n \\ & = & \mathrm{p}_m \circ \cdots \circ \mathrm{p}_{n-3} \circ \rho_{n-2} \circ \mathrm{i}_n \\ & = & \mathrm{p}_m \circ \cdots \circ \rho_{n-3} \ \circ \mathrm{i}_n \\ & = & \cdots = \rho_m \circ \mathrm{i}_n \end{array}$$

and for $n = m$

$$\mathrm{p}_{n,m} = \mathrm{id} = \rho_m \circ \mathrm{i}_n \ .$$

Lemma 2.12.8 *Assume*

- $F : (\mathrm{cpo}^{\mathrm{op}} \times \mathrm{cpo}) \to \mathrm{cpo}$ *is a locally continuous bifunctor,*
- D_0 *is a cpo,*
- $(\mathrm{e}_0 : D_0 \to F(D_0, D_0), \mathrm{p}_0 : F(D_0, D_0) \to D_0)$ *is an embedding/projection pair.*

Then there exists

- *a cpo D,*
- *an embedding/projection pair*

$$(e : D_0 \to D, p : D \to D_0)$$

  together with
- *an isomorphism (i.e. continuous bijection)*

$$\beta : F(D, D) \xrightarrow{\cong} D \ .$$

Proof:
Define inductively cpo's D_n for $n \geq 1$ by

$$D_n := F(D_{n-1}, D_{n-1}) \ ,$$

and simultaneously for $n \geq 1$ inductively

- $\mathrm{e}_n := F(\mathrm{p}_{n-1}, \mathrm{e}_{n-1})$
- $\mathrm{p}_n := F(\mathrm{e}_{n-1}, \mathrm{p}_{n-1})$.

Then

$$\mathrm{p}_n \circ \mathrm{e}_n = \mathrm{id}_{D_n} \quad :$$

$n = 0$ is clear and

$$\begin{array}{rcl} \mathrm{p}_{n+1} \circ \mathrm{e}_{n+1} & = & F(\mathrm{e}_n, \mathrm{p}_n) \circ F(\mathrm{p}_n, \mathrm{e}_n) \\ & = & F(\mathrm{p}_n \circ \mathrm{e}_n, \mathrm{p}_n \circ \mathrm{e}_n) \\ & \stackrel{IV}{=} & F(\mathrm{id}, \mathrm{id}) = \mathrm{id} \ . \end{array}$$

Further

$$\mathrm{e}_n \circ \mathrm{p}_n \sqsubseteq \mathrm{id}_{D_{n+1}} \quad :$$

$$\begin{array}{rcl} \mathrm{e}_0 \circ \mathrm{p}_0 & \sqsubseteq & \mathrm{id}_{D_1} \ , \\ \mathrm{e}_{n+1} \circ \mathrm{p}_{n+1} & = & F(\mathrm{p}_n, \mathrm{e}_n) \circ F(\mathrm{e}_n, \mathrm{p}_n) \\ & = & F(\mathrm{e}_n \circ \mathrm{p}_n, \mathrm{e}_n \circ \mathrm{p}_n) \\ & \stackrel{IH}{\sqsubseteq} & F(\mathrm{id}, \mathrm{id}) = \mathrm{id} \ . \end{array}$$

Therefore $(\mathrm{e}_n, \mathrm{p}_n)$ is a sequence of embedding/projection pairs.
By Lemma 2.12.6 there exists a cone of embedding/projection pairs

$$\mathrm{i}_n : D_n \to D, \quad \rho_n : D \to D_n$$

s.t.

- $\mathrm{i}_{n+1} \circ \mathrm{e}_n = \mathrm{i}_n$,
- $\mathrm{p}_n \circ \rho_{n+1} = \rho_n$,
- $\sqcup(\mathrm{i}_n \circ \rho_n) = \mathrm{id}_D$.

Therefore

$$\begin{array}{rcl} \mathrm{i}_{n+1} \circ F(\mathrm{i}_n, \rho_n) & = & \mathrm{i}_{n+1} \circ F(\mathrm{i}_{n+1} \circ \mathrm{e}_n, \mathrm{p}_n \circ \rho_{n+1}) \\ & = & \mathrm{i}_{n+1} \circ F(\mathrm{e}_n, \mathrm{p}_n) \circ F(\mathrm{i}_{n+1}, \rho_{n+1}) \\ & = & \mathrm{i}_{n+1} \circ \mathrm{p}_{n+1} \circ F(\mathrm{i}_{n+1}, \rho_{n+1}) \\ & = & \mathrm{i}_{n+2} \circ \mathrm{e}_{n+1} \circ \mathrm{p}_{n+1} \circ F(\mathrm{i}_{n+1}, \rho_{n+1}) \\ & \sqsubseteq & \mathrm{i}_{n+2} \circ \mathrm{id} \circ F(\mathrm{i}_{n+1}, \rho_{n+1}) \\ & = & \mathrm{i}_{n+2} \circ F(\mathrm{i}_{n+1}, \rho_{n+1}) \end{array}$$

$$\begin{array}{rcl} F(\rho_n, \mathrm{i}_n) \circ \rho_{n+1} & = & F(\mathrm{p}_n \circ \rho_{n+1}, \mathrm{i}_{n+1} \circ \mathrm{e}_n) \circ \rho_{n+1} \\ & = & F(\rho_{n+1}, \mathrm{i}_{n+1}) \circ F(\mathrm{p}_n, \mathrm{e}_n) \circ \mathrm{p}_{n+1} \circ \rho_{n+2} \\ & = & F(\rho_{n+1}, \mathrm{i}_{n+1}) \circ \mathrm{e}_{n+1} \circ \mathrm{p}_{n+1} \circ \rho_{n+2} \\ & \sqsubseteq & F(\rho_{n+1}, \mathrm{i}_{n+1}) \circ \mathrm{id} \circ \rho_{n+2} \\ & = & F(\rho_{n+1}, \mathrm{i}_{n+1}) \circ \rho_{n+2} \end{array}$$

Therefore there exists

$$\begin{array}{rcl} \beta & := & \sqcup(\mathrm{i}_{n+1} \circ F(\mathrm{i}_n, \rho_n)) \ , \\ \alpha & := & \sqcup F(\rho_n, \mathrm{i}_n) \circ \rho_{n+1} \ . \end{array}$$

and we get

$$\begin{array}{rcl} \beta \circ \alpha & = & \sqcup_{n,m \in \mathbb{N}} \mathrm{i}_{n+1} \circ F(\mathrm{i}_n, \rho_n) \circ F(\rho_m, \mathrm{i}_m) \circ \rho_{m+1} \\ & = & \sqcup_{n,m \in \mathbb{N}} \mathrm{i}_{n+1} \circ F(\rho_m \circ \mathrm{i}_n, \rho_n \circ \mathrm{i}_m) \circ \rho_{m+1} \\ & = & \sqcup_{n,m \in \mathbb{N}} \mathrm{i}_{n+1} \circ F(\genfrac{}{}{0pt}{}{\mathrm{e}_{n,m}}{\mathrm{p}_{n,m}}, \genfrac{}{}{0pt}{}{\mathrm{p}_{m,n}}{\mathrm{e}_{m,n}}) \circ \rho_{m+1} \\ & = & \sqcup_{n,m \in \mathbb{N}} \mathrm{i}_{n+1} \circ \genfrac{}{}{0pt}{}{\mathrm{p}_{m+1,n+1}}{\mathrm{e}_{m+1,n+1}} \circ \rho_{m+1} \\ & = & \sqcup_{n \in \mathbb{N}} \mathrm{i}_{n+1} \circ \rho_{n+1} \\ & = & \mathrm{id} \end{array}$$

$$\begin{array}{rcl} \alpha \circ \beta & = & \sqcup_{n,m \in \mathbb{N}} F(\rho_n, \mathrm{i}_n) \circ \rho_{n+1} \circ \mathrm{i}_{m+1} \circ F(\mathrm{i}_m, \rho_m) \\ & = & \sqcup_{n,m \in \mathbb{N}} F(\rho_n, \mathrm{i}_n) \circ \genfrac{}{}{0pt}{}{\mathrm{p}_{m+1,n+1}}{\mathrm{e}_{m+1,n+1}} \circ F(\mathrm{i}_m, \rho_m) \\ & = & \sqcup_{n,m \in \mathbb{N}} F(\rho_n, \mathrm{i}_n) \circ F(\genfrac{}{}{0pt}{}{\mathrm{e}_{n,m}}{\mathrm{p}_{n,m}}, \genfrac{}{}{0pt}{}{\mathrm{p}_{m,n}}{\mathrm{e}_{m,n}}) \circ F(\mathrm{i}_m, \rho_m) \\ & = & \sqcup_{n,m \in \mathbb{N}} F(\mathrm{id}, \mathrm{id}) \\ & = & \mathrm{id} \end{array}$$

Therefore β is an isomorphism, $\alpha = \beta^{-1}$.

2.12.2 D_∞ and other λ-models

Definition 2.12.9 (a) D_∞ is defined as the in Lemma 2.12.8 constructed cpo where

- $F(A, B) := A \to B$, $F(g, f)(h) := f \circ h \circ g$.
- $D_0 := \mathbb{N}_\perp$.
- $\mathrm{e}_0(n) := \lambda\backslash\, m.n$, $\mathrm{p}_0(f) := f(\perp)$.

(b) Let $\alpha : \mathrm{D}_\infty \to [\mathrm{D}_\infty \to \mathrm{D}_\infty]$ be the isomorphism, β its inverse.

Define $\cdot : \mathrm{D}_\infty \times \mathrm{D}_\infty \to \mathrm{D}_\infty$, $a \cdot b := \alpha(a)(b)$.

Define $\Lambda := \mathrm{id}_{\mathrm{D}_\infty} : \mathrm{D}_\infty \to \mathrm{D}_\infty$.

Theorem 2.12.10 $(\mathrm{D}_\infty, \cdot, \Lambda)$ *is a λ-model.*

Proof:
Let

$$\begin{array}{rcl} \mathbf{k} & := & \beta(\lambda\backslash\, a.\beta(\lambda\backslash\, b.a)) \\ \mathbf{s} & := & \beta(\lambda\backslash\, a.\beta(\lambda\backslash\, b.\beta(\lambda\backslash\, c.\alpha(\alpha(a)(c))(\alpha(b)(c))))) \end{array}$$

$\mathbf{k}$, $\mathbf{s}$ are well-defined and

$$\begin{aligned}\mathbf{k}\cdot a\cdot b &= \alpha(\alpha(\beta(\lambda\backslash\, a.\beta(\lambda\backslash\, b.a)))(a))(b)\\ &= \alpha(\beta(\lambda\backslash\, b.a))(b)\\ &= a\ ,\end{aligned}$$

$$\begin{aligned}&\mathbf{s}\cdot a\cdot b\cdot c\\ &=\alpha(\alpha(\alpha(\beta(\lambda\backslash\, a.\beta(\lambda\backslash\, b.\beta(\lambda\backslash\, c.\alpha(\alpha(a)(c))(\alpha(b)(c))))))(a))(b))(c)\\ &=\alpha(\alpha(\beta(\lambda\backslash\, b.\beta(\lambda\backslash\, c.\alpha(\alpha(a)(c))(\alpha(b)(c)))))(b))(c)\\ &=\alpha(\alpha(a)(c))(\alpha(b)(c))\\ &=a\cdot c\cdot(b\cdot c)\ .\end{aligned}$$

$(\mathrm{D}_\infty,\cdot)$ is extensional, since if $a\sim b$, then

$$\forall c\in\mathrm{D}_\infty.\alpha(a)(c)=a\cdot c=b\cdot c=\alpha(b)(c)$$

$$\alpha(a)=\alpha(b)\ ,$$

$$a=b\ .$$

Therefore by Theorem 2.11.22

$$(\mathrm{D}_\infty,\cdot,\lambda\backslash\, d.d)$$

is a λ-model.

Remark 2.12.11 *In the above we could have replaced $\mathbb{N}_\perp$ by any other (non-trivial) cpo. All the proofs above remain as before.*

Definition 2.12.12 Let $D:=\mathrm{D}_\infty$, D_n as in the construction of D_∞, i.e.

- $D_0:=\mathbb{N}_\perp$,
- $D_{n+1}:=F(D_n,D_n)$, where
- $F(D,E)=[D\to E]$, $F(g,f)(h)=f\circ h\circ g$.

For σ being an assignment of variables in D, M a λ-terms define $[\![\ M\ ]\!]^n_\sigma\in D_n$ as follows:

- $[\![\ x\ ]\!]^n_\sigma:=\rho_n(\sigma(x))$.
- $[\![\ M\ N\ ]\!]^n_\sigma:=[\![\ M\ ]\!]^{n+1}_\sigma([\![\ N\ ]\!]^n_\sigma)$.
- $[\![\ \lambda x.M\ ]\!]^0_\sigma:=\mathrm{p}_0(\lambda\backslash\, d\in D_0.[\![\ M\ ]\!]^0_{\sigma^{\mathrm{i}_0(d)}_x})$,

  $[\![\ \lambda x.M\ ]\!]^{n+1}_\sigma:=\lambda\backslash\, d\in D_n.[\![\ M\ ]\!]^n_{\sigma^{\mathrm{i}_n(d)}_x}$.

(where one simultaneously shows immediately, that

$$\lambda\backslash\, d_1,\dots,d_m\in D.[\![\ M\ ]\!]^n_{\sigma^{d_1\ d_2}_{x_1\ x_2}\cdots^{d_m}_{x_m}}\in[D^m\to D_n]\ ).$$

Lemma 2.12.13 *Let M be a λ-term, σ be an assignment.*

(a) $\mathrm{i}_n([\![M]\!]^n_\sigma) \sqsubseteq \mathrm{i}_{n+1}([\![M]\!]^{n+1}_\sigma)$.

(b) $[\![M]\!]_\sigma = \sqcup_{n\in\mathbb{N}}\mathrm{i}_n([\![M]\!]^n_\sigma)$.

Proof:
Let

$$\begin{array}{rcl}\alpha & := & \sqcup(F(\rho_n,\mathrm{i}_n)\circ\rho_{n+1}) : D\to F(D,D)\\ \beta & := & \sqcup(\mathrm{i}_{n+1}\circ F(\mathrm{i}_n,\rho_n)) : F(D,D)\to D\end{array}$$

the inverse isomorphisms.
We prove both (a) and (b) by induction on M:
Case $M \equiv x$:

$$\begin{array}{rcl}\mathrm{i}_n([\![M]\!]^n_\sigma) & = & \mathrm{i}_n(\rho_n(\sigma(x)))\\ & = & (\mathrm{i}_n\circ\rho_n)(\sigma(x))\ .\end{array}$$

Therefore it follows (a) by

$$\mathrm{i}_n\circ\rho_n \sqsubseteq \mathrm{i}_{n+1}\circ\rho_{n+1}$$

and (b) by

$$\sqcup_{n\in\mathbb{N}}\mathrm{i}_n\circ\rho_n = \mathrm{id}\ .$$

Case $M \equiv P\,Q$:

$$\begin{array}{rl}& \mathrm{i}_n([\![P\,Q]\!]^n_\sigma)\\ = & \mathrm{i}_n([\![P]\!]^{n+1}_\sigma([\![Q]\!]^n_\sigma))\\ = & \mathrm{i}_n((\rho_{n+1}(\mathrm{i}_{n+1}([\![P]\!]^{n+1}_\sigma)))(\rho_n(\mathrm{i}_n([\![Q]\!]^n_\sigma))))\\ = & (\mathrm{i}_n\circ\rho_{n+1}(\mathrm{i}_{n+1}([\![P]\!]^{n+1}_\sigma))\circ\rho_n)(\mathrm{i}_n([\![Q]\!]^n_\sigma))\\ = & (F(\rho_n,\mathrm{i}_n)\circ\rho_{n+1})(\mathrm{i}_{n+1}([\![P]\!]^{n+1}_\sigma))(\mathrm{i}_n([\![Q]\!]^n_\sigma))\end{array}$$

Let

$$g(n,m,k) := (F(\rho_n,\mathrm{i}_n)\circ\rho_{n+1})(\mathrm{i}_{m+1}([\![P]\!]^{m+1}_\sigma))(\mathrm{i}_k([\![Q]\!]^k_\sigma))\ .$$

Then $g(n,m,k)$ is monotone in n, m, k. Therefore

$$\begin{array}{rcl}\mathrm{i}_n([\![P\,Q]\!]^n_\sigma) & = & g(n,n,n)\\ & \sqsubseteq & g(n+1,n+1,n+1)\\ & = & \mathrm{i}_{n+1}([\![P\,Q]\!]^{n+1}_\sigma)\ ,\end{array}$$

and

$$\begin{array}{rl}& \sqcup_n\mathrm{i}_n([\![P\,Q]\!]^n_\sigma)\\ = & \sqcup_n g(n,n,n)\\ = & \sqcup_{n,m,k}g(n,m,k)\\ = & (\sqcup_n(F(\rho_n,\mathrm{i}_n)\circ\rho_{n+1}))(\sqcup_m(\mathrm{i}_{m+1}([\![P]\!]^{m+1}_\sigma)))(\sqcup_k(\mathrm{i}_k([\![Q]\!]^k_\sigma)))\\ = & \alpha([\![P]\!]_\sigma)([\![Q]\!]_\sigma)\ .\end{array}$$

Case $M = \lambda x.P$.

$$\begin{array}{rcl}
\mathrm{i}_0([\![\ \lambda x.P\ ]\!]^0_\sigma) & = & \mathrm{i}_0(\mathrm{p}_0(\lambda\backslash\ d \in D_0.[\![\ P\ ]\!]^0_{\sigma_x^{\mathrm{i}_0(d)}})) \\
& = & \mathrm{i}_1(\lambda\backslash\ d \in D_0.[\![\ P\ ]\!]^0_{\sigma_x^{\mathrm{i}_0(d)}}) \\
& = & \mathrm{i}_1([\![\ \lambda x.P\ ]\!]^1_\sigma)\ .
\end{array}$$

Further

$$\begin{array}{ll}
& \mathrm{i}_{n+1}([\![\ \lambda x.P\ ]\!]^n_\sigma) \\
= & \mathrm{i}_{n+1}(\lambda\backslash\ d \in D_n.[\![\ P\ ]\!]^n_{\sigma_x^{\mathrm{i}_n(d)}}) \\
= & \mathrm{i}_{n+1}(\lambda\backslash\ d \in D_n.\rho_n(\mathrm{i}_n([\![\ P\ ]\!]^n_{\sigma_x^{\mathrm{i}_n(d)}}))) \\
= & \mathrm{i}_{n+1}(\rho_n \circ (\lambda\backslash\ d \in D.\mathrm{i}_n([\![\ P\ ]\!]^n_{\sigma_x^d})) \circ \mathrm{i}_n) \\
= & (\mathrm{i}_{n+1} \circ F(\mathrm{i}_n, \rho_n))(\lambda\backslash\ d \in D.\mathrm{i}_n([\![\ P\ ]\!]^n_{\sigma_x^d}))\ .
\end{array}$$

By

$$\begin{array}{rcl}
\mathrm{i}_n([\![\ P\ ]\!]^n_{\sigma_x^d}) & \sqsubseteq & \mathrm{i}_{n+1}([\![\ P\ ]\!]^{n+1}_{\sigma_x^d}) \\
& \text{therefore} & \\
\lambda\backslash\ d \in D.\mathrm{i}_n([\![\ P\ ]\!]^n_{\sigma_x^d}) & \sqsubseteq & \lambda\backslash\ d \in D.\mathrm{i}_{n+1}([\![\ P\ ]\!]^{n+1}_{\sigma_x^d}) \\
& \text{further} & \\
\mathrm{i}_{n+1} \circ F(\mathrm{i}_n, \rho_n) & \sqsubseteq & \mathrm{i}_{n+2} \circ F(\mathrm{i}_{n+1}, \rho_{n+1}) \\
& \text{therefore} & \\
\mathrm{i}_{n+1}([\![\ \lambda x.P\ ]\!]^n_\sigma) & \sqsubseteq & \mathrm{i}_{n+2}([\![\ \lambda x.P\ ]\!]^{n+1}_\sigma)
\end{array}$$

Further by the monotonicity just mentioned it follows

$$\begin{array}{ll}
& \sqcup_n \mathrm{i}_{n+1}([\![\ \lambda x.P\ ]\!]^n_\sigma) \\
= & (\sqcup_n(\mathrm{i}_{n+1} \circ F(\mathrm{i}_n, \rho_n)))(\sqcup_m(\lambda\backslash\ d \in D.\mathrm{i}_m([\![\ P\ ]\!]^m_{\sigma_x^d}))) \\
= & \beta(\lambda\backslash\ d \in D.[\![\ P\ ]\!]_{\sigma_x^d}) \\
= & \Lambda(\beta(\lambda\backslash\ d \in D.[\![\ P\ ]\!]_{\sigma_x^d})) \\
= & [\![\ \lambda x.P\ ]\!]_\sigma\ .
\end{array}$$

We can easily define a non-extensional λ-model as well:

Definition 2.12.14 Let

$$D + E := \Sigma_{i \in \{0,1\}} D_i$$

where

$$D_0 := D, \quad D_1 := E\ .$$

Let $\iota_i : D_i \to D_0 + D_1$, be the two canonical embeddings.
Let D be a non-trivial solution of the cpo-equation

$$D \cong [D \to D] + [D \to D]\ .$$

$$\alpha : D \to ([D \to D] + [D \to D])$$

be the isomorphism and β its inverse.
Define

$$a \cdot b := f(b), \text{ if } \alpha(a) = \iota_i(f) \ .$$

Let

$$\Lambda_i : D \rightarrow D, \quad \Lambda_i(e) = \beta(\iota_i(f)) \text{ if } \alpha(e) = \iota_k(f) \ .$$

Then $(D, \cdot, \Lambda_i)$ are two structures which fulfill conditions (a) - (c) of Definition 2.11.17. Further for $j = 0, 1$

$$e_{j,i} := \beta(\iota_j(\Lambda_i))$$

are two different choices of e s.t. condition (d) is fulfilled as well, i.e.

$$(D, \cdot, \Lambda_i)$$

are two λ-models based on the same combinatory algebra, which are both not extensional and have each at least two possible solutions for e.

2.13 The typed λ-Calculus

We will in the following introduce the typed λ-calculus and prove strong normalization of it. In the next section, we will introduce the Curry-Howard isomorphism and show, how to prove normalization of intuitionistic predicate logic from normalization of the typed λ-calculus.
It will turn out, that the type $\sigma \rightarrow \tau$ corresponds to $A \rightarrow B$ and $\forall x.A$, and prime formulas of minimal logic to the ground type o. Other formula constructions will correspond to other types:

- $A \vee B$ will correspond to a type $\sigma + \tau$ which corresponds to the disjoint union of the types σ and τ.
- $A \wedge B$ will correspond to a type $\sigma \times_0 \tau$, and $\exists x.A$ will correspond to a product type $\sigma \times_1 \tau$, where $\times_0$ and $\times_1$ will be two choices of product types which differ in the choice of the elimination rule.
- Further for the $\perp$ we need a new type $\emptyset$.

We will in the following follow partly [MJ98].
For the new types new λ-terms have to be introduced.

Definition 2.13.1 (a) The set of *types* Type is inductively defined by

- $\emptyset, \mathrm{o} \in$ Type.
- If $\sigma, \tau \in$ Type, then $\sigma \rightarrow \tau, \sigma \times_0 \tau, \sigma \times_1 \tau, \sigma + \tau \in$ Type.

In the following σ, ρ, τ, possibly with sub/superscripts and/or accents denote elements of Type.

(b) We define inductively the set of *terms* Term together with their types. Here s^σ or $s : \sigma$ means: s is a term of type σ. There are infinitely many variables x^σ for every σ given. In the following x, y, z, u, v denote variables possibly with subscripts and/or indides, and $x^\sigma, y^\sigma, z^\sigma, u^\sigma, v^\sigma$ (again with accents or subscrips) denote variables of type σ.

- x^σ is a term of type σ.
- If r, s are terms of types as given by their superscripts below, then
  - $(\lambda x^\sigma.s^\tau)^{\sigma\to\tau}$,
  - $\langle r^\sigma, s^\tau\rangle_i^{\sigma\times_i\tau}$, $(i = 0, 1)$
  - $(\iota_{0,\tau} r^\sigma)^{\sigma+\tau}$,
  - $(\iota_{1,\tau} r^\sigma)^{\tau+\sigma}$,
  - $(r^{\sigma\to\tau}\ s^\sigma)^\tau$,
  - $(r^{\sigma\times_0\tau}\ 0)^\sigma$,
  - $(r^{\sigma\times_0\tau}\ 1)^\tau$,
  - $(r^{\sigma\times_1\tau}\ [\lambda x^\sigma, y^\tau.s^\rho])^\rho$,
  - $(r^{\sigma+\tau}\ [\lambda x^\sigma.s^\rho, \lambda y^\tau.t^\rho])^\rho$
  - $(r^\emptyset\ \mathrm{efq}_\sigma)^\sigma$

are terms of the type indicated by the superscript.

The *length* ($\mathrm{lgh}(s)$) of a term s is the number of steps in the above definition needed in order to derive that s is a term.

Brackets are omitted as usual.

The type index in $\iota_{i,\tau}$, and efq_σ is only needed to make the typing of a term unique, and will be usually omitted.

In the following all λ-terms occurring are assumed to be elements of Term, if $[\lambda x, y.s]$ occurs, s is assumed to be a term of the corresponding type, and if $[\lambda x.s, \lambda y.t]$ occurs, s, t are typed terms of the corresponding type.

(c) *Good elimination terms* are terms, 0, 1. $\mathrm{Term}_{0,1}$ is the set of good elimination terms.

For $A \subseteq \mathrm{Term}$ let $A_{0,1} := A \cup \{0, 1\}$.

In the following r, s, t possibly with sub/superscripts, accents denote terms, $\vec{r}, \vec{s}, \vec{t}$ with the same extensions denote sequences of terms, unless they are stated as elements of a set of terms or sequences of terms extended by $0, 1$, in which case they are good elimination terms.

Elimination terms are good elimination terms, efq_σ, $[\lambda x.r, \lambda y.s]$ and $[\lambda x, y.r]$.

In the following R, S, T with the usual extensions denote elimination terms and $\vec{R}$, $\vec{S}$, $\vec{T}$ with the same extensions denote sequences of elimination terms.

(d) Free, bounded variables, substitution, α-conversion is defined as usual. We will in the following identify α-equivalent terms.

(e) A^* is the set of finite (possibly empty) sequences of elements of A.

(f) The reduction relation $\longrightarrow \subseteq \mathrm{Term} \times \mathrm{Term}$ is inductively defined by

- $(\lambda x.r)\ s \longrightarrow r[x := s]$,
- $\langle r_0, r_1\rangle_0\ i \longrightarrow r_i$,
- $\langle r_0, r_1\rangle_1\ [\lambda x_0, x_1.s] \longrightarrow s[x_0 := r_0, x_1 := r_1]$,
- $\iota_i(r)\ [\lambda x_0.s_0, \lambda x_1.s_1] \longrightarrow s_i[x_i := r]$.
- $(r\ \mathrm{efq}_\sigma\ S)^\rho \longrightarrow r\ \mathrm{efq}_\rho$.
- $r\ [\lambda x, y.s]\ S \longrightarrow r\ [\lambda x, y.s\ S]\ (x, y \notin \mathrm{FV}(S))$.
- $r\ [\lambda x.s, \lambda y.t]\ S \longrightarrow r\ [\lambda x.s\ S, \lambda y.t\ S]\ (x, y \notin \mathrm{FV}(S))$.
  (The last three reductions are called "permutative conversions").
- If $r \longrightarrow r'$ then
  - $\lambda x.r \longrightarrow \lambda x.r'$,
  - $\langle r, s\rangle_i \longrightarrow \langle r', s\rangle_i$,
  - $\langle s, r\rangle_i \longrightarrow \langle s, r'\rangle_i$,
  - $\iota_i(r) \longrightarrow \iota_i(r')$,
  - $r\ S \longrightarrow r'\ S$,
  - $s\ r \longrightarrow s\ r'$,
  - $s\ [\lambda x, y.r] \longrightarrow s\ [\lambda x, y.r']$,
  - $s\ [\lambda x.r, \lambda y.t] \longrightarrow s\ [\lambda x.r', \lambda y.t]$,
  - $s\ [\lambda x.t, \lambda y.r] \longrightarrow s\ [\lambda x.t, \lambda y.r']$.

(g) $\longrightarrow^*$ is the reflexive transitive closure of $\longrightarrow$.

Remark 2.13.2 *(a) $\lambda x.s$, $\langle r, s\rangle_i$, $\iota_i(r)$ introduce new elements of the corresponding type. Therefore these constructions are called introductions. $r\ s$, $r\ i$, $r\ \mathrm{efq}_\sigma$, $r\ [\lambda x, y.s]$, $r\ [\lambda x.s, \lambda y.t]$, correspond to the formation of a new element of a different type from r, and one says that one eliminates r. Therefore these constructions are called eliminations.*

(b) The notations for the elimination of $+$, $\times_1$ and $\emptyset$ are non-standard. Standard-notations are for (the choice of letters C, E, efq varies)

$$\begin{gathered} \mathrm{E}(r, (x, y)s) \textit{ for } r\ [\lambda x, y.s]\ , \\ \mathrm{C}(r, (x)s, (y)t) \textit{ for } r\ [\lambda x.s, \lambda y.t]\ , \\ \mathrm{efq}_\sigma(r) \textit{ for } r\ \mathrm{efq}_\sigma\ . \end{gathered}$$

We use our notations because it will simplify the following proofs quite a lot: successive eliminations applied to a term r can be written as $r\ \vec{S}$.

Theorem 2.13.3 $\longrightarrow$ *is Church-Rosser.*

Proof: As before.

Lemma 2.13.4 *(a) If $r \longrightarrow r'$ then*

$$r[x := s] \longrightarrow r'[x := s] \ .$$

(b) If $s \longrightarrow s'$, then

$$r[x := s] \longrightarrow^* r[x := s'] \ .$$

Proof: Easy.

A term is *strongly normalizing*, if every reduction sequence terminates. A better way of defining normalizing is the following.

Definition 2.13.5 (a) We define inductively the subset SN of *strongly normalizing terms* by:

$$\forall r'.(r{\rightarrow}r'{\Rightarrow}r' \in \mathrm{SN}){\Rightarrow}r \in \mathrm{SN} \ .$$

(b) For $s \in \mathrm{SN}_{0,1}$ define height(s) by:

$$\begin{array}{rcl} r \in \mathrm{SN} & \Rightarrow & \mathrm{height}(r) = \max(\{\mathrm{height}(r') + 1 \mid r \longrightarrow r'\} \\ & & \quad \cup \{0\}) \ , \\ & & \mathrm{height}(0) := \mathrm{height}(1) := 0 \ . \end{array}$$

We note that α-equivalent terms have the same height and that, since a term reduces only (up to α-equivalence) to finitely many terms, the height of elements of $\mathrm{SN}_{0,1}$ is finite.

Proof that the strongly normalizing terms are exactly those s.t. every reduction sequence terminates:
First it follows immediately by induction on s that if $s \in \mathrm{SN}$ then every reduction sequence terminates after at most height(s) reductions.
On the other hand, assume $s \notin \mathrm{SN}$. Define a sequence $s = s_0 \longrightarrow s_1 \longrightarrow s_1 \longrightarrow \cdots$ s.t. for all n $s_n \notin \mathrm{SN}$ as follows: $s_0 := s$. s_{n+1} is the (w.r.t the Gödelnumbering) least term s.t. $s_n \longrightarrow s_{n+1}$, $s_{n+1} \notin \mathrm{SN}$. s is not normalizing in the original sense.

Remark 2.13.6 *If $r \in \mathrm{SN}$, $r \longrightarrow r'$, then* $\mathrm{height}(r') < \mathrm{height}(r)$.

Definition 2.13.7 We define inductively the subset terms in $\widetilde{\mathrm{SN}}$, where $\vec{r} \in \widetilde{\mathrm{SN}}^*_{0,1}$. $\widetilde{\mathrm{SN}}$ is defined by the following rules (i.e. if the premisses are fulfilled the conclusion holds).

(Var^0) $\dfrac{\vec{r} \in \widetilde{\mathrm{SN}}_{0,1}^*}{x\ \vec{r} \in \widetilde{\mathrm{SN}}}$ (Var^1) $\dfrac{\vec{r} \in \widetilde{\mathrm{SN}}_{0,1}^* \quad y^\sigma\ \vec{S} \in \widetilde{\mathrm{SN}}}{x\ \vec{r}\ \mathrm{efq}_\sigma\ \vec{S} \in \widetilde{\mathrm{SN}}}$

(Var^2) $\dfrac{\vec{r} \in \widetilde{\mathrm{SN}}_{0,1}^* \quad s\ \vec{S} \in \widetilde{\mathrm{SN}}}{x\ \vec{r}\ [\lambda x, y.s]\ \vec{S} \in \widetilde{\mathrm{SN}}}$

(Var^3) $\dfrac{\vec{r} \in \widetilde{\mathrm{SN}}_{0,1}^* \quad s\ \vec{S} \in \widetilde{\mathrm{SN}} \quad t\ \vec{S} \in \widetilde{\mathrm{SN}}}{x\ \vec{r}\ [\lambda x.s, \lambda y.t]\ \vec{S} \in \widetilde{\mathrm{SN}}}$

(λ^0) $\dfrac{r \in \widetilde{\mathrm{SN}}}{\lambda x.r \in \widetilde{\mathrm{SN}}}$ ($\langle\rangle_i^0$) $\dfrac{r \in \widetilde{\mathrm{SN}} \quad s \in \widetilde{\mathrm{SN}}}{\langle r, s\rangle_i \in \widetilde{\mathrm{SN}}}$

(ι^0) $\dfrac{r \in \widetilde{\mathrm{SN}}}{\iota_i(r) \in \widetilde{\mathrm{SN}}}$

(λ^1) $\dfrac{r[x := s]\ \vec{S} \in \widetilde{\mathrm{SN}} \quad s \in \widetilde{\mathrm{SN}}}{(\lambda x.r)\ s\ \vec{S} \in \widetilde{\mathrm{SN}}}$ ($\langle\rangle_0^1$) $\dfrac{r_i\ \vec{S} \in \widetilde{\mathrm{SN}} \quad r_{1-i} \in \widetilde{\mathrm{SN}}}{\langle r_0, r_1\rangle_0\ i\ \vec{S} \in \widetilde{\mathrm{SN}}}$

($\langle\rangle_1^1$) $\dfrac{s[x_0 := r_0, x_1 := r_1]\ \vec{S} \in \widetilde{\mathrm{SN}} \quad r_0 \in \widetilde{\mathrm{SN}} \quad r_1 \in \widetilde{\mathrm{SN}}}{\langle r_0, r_1\rangle_1\ [\lambda x_0, x_1.s]\ \vec{S} \in \widetilde{\mathrm{SN}}}$

(ι^1) $\dfrac{s_i[x_i := r]\ \vec{S} \in \widetilde{\mathrm{SN}} \quad s_{1-i} \in \widetilde{\mathrm{SN}} \quad r \in \widetilde{\mathrm{SN}}}{\iota_i(r)\ [\lambda x_0.s_0, \lambda x_1.s_1]\ \vec{S} \in \widetilde{\mathrm{SN}}}$

The *length of a derivation of* $r \in \widetilde{\mathrm{SN}}$ ($\mathrm{lgh}_{\widetilde{\mathrm{SN}}}$) is the number of rules needed for deriving $r \in \widetilde{\mathrm{SN}}$.
This extends to $r \in \widetilde{\mathrm{SN}}_{0,1}$ by $\mathrm{lgh}_{\widetilde{\mathrm{SN}}}(0) := \mathrm{lgh}_{\widetilde{\mathrm{SN}}}(1) := 0$.

Lemma 2.13.8

$$r \in \mathrm{SN} \Leftrightarrow r \in \widetilde{\mathrm{SN}}$$

Remark: For the proof of the strong normalization theorem only $\Rightarrow$ is really needed (we will prove that all typed terms are in $\widetilde{\mathrm{SN}}$ and therefore in SN), although we will for convenience make use of that fact. But it is interesting to see that $\widetilde{\mathrm{SN}}$ is just another definition of SN.

Proof:
"$\Rightarrow$" (Soundness) We show for all rules of $\widetilde{\mathrm{SN}}$:

- If the premise holds with $\widetilde{\mathrm{SN}}$, $\widetilde{\mathrm{SN}}_{0,1}$ replaced by SN, $\mathrm{SN}_{0,1}$,
- then the conclusion holds as well with the same replacement.

This will be proved for all rules by induction on the sum of the heights of the terms and elimination terms in the premise, and in case of (Var^1), (Var^2), (Var^3), $(\langle\rangle_1^1)$, (ι^1) by side-induction on the number of elimination terms in $\vec{S}$.

Case (Var^0): If $x\ \vec{r} \longrightarrow t$ then $t \equiv x\ \vec{r}_0\ r_i\ \vec{r}_1$ with $\vec{r} \equiv \vec{r}_0\ r'_i\ \vec{r}_1$ and $r_i \longrightarrow r'_i$. By IH $t \in \mathrm{SN}$.

Cases (Var^1), (Var^2), (Var^3), (λ^0), $(\langle\rangle_i^0)$, (ι^0): Again if the concluding term reduces to t, t can be derived by the same rule with sum of the heights of the premises smaller, or in the main case of a permutative conversion in (Var^i) by the same rule with the same premises (or with height $\leq$ the height of a premise), but with shorter $\vec{S}$. By IH follows $t \in \mathrm{SN}$.

Cases (λ^1), $(\langle\rangle_i^1)$, (ι^1): If the concluding term reduces to t, t is either the first premise, or (using Lemma 2.13.4) it can be derived by the same rule with sum of the heights of the premises smaller, or it is the result of a permutative conversion with the elimination term explicitely written in case $(\langle\rangle_i^1)$, (ι^1), and can be derived by the same rule, but with reduced length of $\vec{S}$. Therefore by main or side-IH $t \in \mathrm{SN}$.

"$\Leftarrow$" (Completeness) We show for all rules by induction on the height of the concluding term:

- If the conclusion holds with $\widetilde{\mathrm{SN}}$ replaced by SN, then the premise holds with $\widetilde{\mathrm{SN}}$, $\widetilde{\mathrm{SN}}_{0,1}$ replaced by SN, $\mathrm{SN}_{0,1}$.
- If s is the term in the conclusion and r a term in the premise, then

$$\mathrm{height}(r) < \mathrm{height}(s) \vee (\mathrm{height}(r) = \mathrm{height}(s) \wedge \mathrm{lgh}(r) < \mathrm{lgh}(s)) \ .$$

Since every term is the conclusion of one rule follows than by induction on height(s), side-induction on lgh(s)

$$s \in \mathrm{SN} \Rightarrow s \in \widetilde{\mathrm{SN}} \ .$$

Case (Var^i), (λ^0), $(\langle\rangle_i^0)$, (ι^0): The height of the premises will be less than or equal the height of the conclusion and the length of them is less than the length of the concluding term:

For the length this is clear and if for a premise r we have $r \longrightarrow r'$, than for the conclusion s there is a term s' s.t. $s \longrightarrow s'$ and s' can be derived from the same premises or a reduct of them and from r'. By IH these new premises are in $\mathrm{SN}_{0,1}$, therefore $r' \in \mathrm{SN}$ and therefore r as well and the assertion concerning the height follows.

Case (λ^1), $(\langle\rangle_i^1)$, (ι^1): The first premise is a reduct of the conclusion therefore an element of SN with smaller height.

The other premises have height less than or equal the conclusion and smaller length, by the same proof as before, using Lemma 2.13.4.

Lemma 2.13.9 *(a) If* $t \in \{(\lambda x.r),\ \iota_i(r),\ \langle r,s\rangle_i,\ \langle s,r\rangle_i,\ r\ \vec{R}, s\ r,\ s\ [\lambda x,y.r],\ s\ [\lambda x.r, \lambda y.r'],\ s\ [\lambda y.r', \lambda x.r]\}$, $t \in \widetilde{\mathrm{SN}}$*, then* $r \in \widetilde{\mathrm{SN}}$.

(b) $r \in \widetilde{\mathrm{SN}} \Leftrightarrow r[x := y] \in \widetilde{\mathrm{SN}}$.

Proof: (a) $t \in \widetilde{\mathrm{SN}}$, then $t \in \mathrm{SN}$, and every reduction in r corresponds to a reduction in t, therefore by induction on height(t) it follows $r \in \mathrm{SN} = \widetilde{\mathrm{SN}}$. (Induction on $t \in \widetilde{\mathrm{SN}}$ is as well possible, but then one needs to prove as well: $r[x := s] \in \widetilde{\mathrm{SN}} \Rightarrow r \in \widetilde{\mathrm{SN}}$).
(b) Induction on the derivation of $r \in \widetilde{\mathrm{SN}}$ or $r[x := y] \in \widetilde{\mathrm{SN}}$.

Lemma 2.13.10 *For all types ρ and all $r \in \widetilde{\mathrm{SN}}$ it follows*

(a) *If $\rho \equiv \rho_0 \to \rho_1$, $r : \rho$, $s \in \widetilde{\mathrm{SN}}$, then $r\ s \in \widetilde{\mathrm{SN}}$.*

(b) *If $\rho \equiv \rho_0 \times_0 \rho_1$, $r : \rho$ then $r\ 0, r\ 1 \in \widetilde{\mathrm{SN}}$.*

(c) *If $\rho \equiv \emptyset$, $r : \rho$, $y\ \vec{T} \in \widetilde{\mathrm{SN}}$, then $r\ \mathrm{efq}_\sigma\ \vec{T} \in \widetilde{\mathrm{SN}}$.*

(d) *If $\rho \equiv \rho_0 \times_1 \rho_1$, $r : \rho$, $t\ \vec{T} \in \widetilde{\mathrm{SN}}$, then $r\ [\lambda x_0, x_1.t]\ \vec{T} \in \widetilde{\mathrm{SN}}$.*

(e) *If $\rho \equiv \rho_0 + \rho_1$, $r : \rho$, $t_0\ \vec{T}, t_1\ \vec{T} \in \widetilde{\mathrm{SN}}$, then $r\ [\lambda x_0.t_0, \lambda x_1.t_1]\ \vec{T} \in \widetilde{\mathrm{SN}}$.*

(f) *If $s^\rho \in \widetilde{\mathrm{SN}}$ then $r[x^\rho := s^\rho] \in \widetilde{\mathrm{SN}}$.*

Proof: We prove (a) - (f) simultaneously by induction on ρ:
Proof of (a) - (e) simultaneously by side-induction on r. Let the conclusion be $r\ \vec{R} \in \widetilde{\mathrm{SN}}$:
Case $r \equiv x\ \vec{r}$: $r \in \widetilde{\mathrm{SN}}$ is derived by (Var^0). (a), (b) follows by (Var^0), (c) by (Var^1), (d) by (Var^2) (e) by (Var^3).
Case $r \equiv x\ \vec{r}\ [\lambda x_0, x_1.s]\ \vec{S}$: then $r \in \widetilde{\mathrm{SN}}$ is derived by (Var^2) from $r_i \in \widetilde{\mathrm{SN}}_{0,1}$, $s\ \vec{S} \in \widetilde{\mathrm{SN}}$. The type of $s\ \vec{S}$ is the same as that of r and it is derived as an element of $\widetilde{\mathrm{SN}}$ before r, therefore by side-IH it follows $s\ \vec{S}\ \vec{R} \in \widetilde{\mathrm{SN}}$ and therefore $r\ \vec{R} \in \widetilde{\mathrm{SN}}$.
Case $r \equiv x\ \vec{r}\ \mathrm{efq}_\sigma\ \vec{T}$ or $r \equiv x\ \vec{r}\ [\lambda x_0.s_0, \lambda x_1.s_1]\ \vec{S}$: similarly.
Case $r \equiv \lambda x.r'$ derived by (λ^0). Only (a) is possible, by main IH for (f) it follows $r'[x := s] \in \widetilde{\mathrm{SN}}$ and therefore by (λ^1) $r\ s \in \widetilde{\mathrm{SN}}$.
Case $r \equiv \langle r_0, r_1 \rangle_0$ derived by ($\langle\rangle_0^0$). Only (c) is possible, and the conclusion follows by ($\langle\rangle_0^1$).
Case $r \equiv \langle r_0, r_1 \rangle_1$ derived by ($\langle\rangle_1^0$): Only (d) is possible. By Lemma 2.13.9 (b) w.l.o.g. $x_i \notin \mathrm{FV}(\vec{T})$. Now we have

- $r_i \in \widetilde{\mathrm{SN}}$.
- By $t\ \vec{T} \in \widetilde{\mathrm{SN}}$, r_i having type ρ_i smaller than ρ, main IH for (f) it follows $t[x_0 := r_0, x_1 := r_1]\ \vec{T} \equiv (t\ \vec{T})[x_0 := r_0, x_1 := r_1] \in \widetilde{\mathrm{SN}}$.

Therefore by ($\langle\rangle_1^1$) it follows $r\ \vec{R} \in \widetilde{\mathrm{SN}}$.
Case $r \equiv \iota_i(r')$ derived by (ι^0): Similarly, but we use additionally that from $t_{1-i}\ \vec{T} \in \widetilde{\mathrm{SN}}$ it follows $t_{1-i} \in \widetilde{\mathrm{SN}}$.

Case $r \in \widetilde{\mathrm{SN}}$ derived by (λ^1), $(\langle\rangle_i^1)$, (ι^1). By side-IH follows the first premise for deriving $r\ \vec{R} \in \widetilde{\mathrm{SN}}$, the other premises are the same as for $r \in \widetilde{\mathrm{SN}}$, therefore $r\ \vec{R} \in \widetilde{\mathrm{SN}}$.

Proof for (f) by side-induction on r: We will additionally use that (a) - (e) is proved for ρ by the proof before.
Case $r \equiv y\ r_1 \cdots r_n$, $r_i \in \widetilde{\mathrm{SN}}$. By side-IH

$$r_i[x := s] \in \widetilde{\mathrm{SN}} \ .$$

If $x \not\equiv y$ follows the assertion by (Var^0), and if $x \equiv y$ it follows by (a), (b), where we use the main IH or the just proved assertion for ρ.
Case $r \equiv y\ r_1 \cdots r_n\ [\lambda x_0, x_1.t]\ \vec{S}$ derived from $r_i \in \widetilde{\mathrm{SN}}_{0,1}$, $s\ \vec{S} \in \widetilde{\mathrm{SN}}$. W.l.o.g. $x_i \not\equiv x$.
By main IH

$$r_i[x := s] \in \widetilde{\mathrm{SN}}_{0,1} \ ,$$

and by side-IH

$$t\ \vec{S}[x := s] \in \widetilde{\mathrm{SN}} \ .$$

If $x \not\equiv y$ follows by (Var^2) the assertion and if $x \equiv y$ it follows by (a) - (e), where we use the main IH or the just proved assertion for ρ.
Case $r \equiv y\ r_1 \cdots r_n\ [\lambda x_0.s_0, \lambda x_1.s_1]\ \vec{S}$, $r \equiv y\ r_1 \cdots r_n\ \mathrm{efq}_\sigma\ \vec{S}$: Similarly.
All other cases follow immediately by side-IH for (f).

Theorem 2.13.11 $r \in \mathrm{SN}$.

Proof: By induction on the length of r using Lemma 2.13.10.

Definition 2.13.12 (a) A term r is in normal form iff $\neg\exists r'.r \longrightarrow r'$.

(b) The set of terms NF is inductively defined as:

- $x \in \mathrm{NF}$.
- If $r_i, s, t \in \mathrm{NF}_{0,1}$, so are
  - $x\ r_1 \cdots r_n$,
  - $x\ r_1, \ldots, r_n$ efq,
  - $x\ r_1 \cdots r_n\ [\lambda y, z.s]$ and
  - $x\ r_1 \cdots r_n\ [\lambda y.s, \lambda z.t]$.
- If $r, s \in \mathrm{NF}$, so are

$$\lambda x.s, \langle r, s\rangle_i, \iota_i(r) \ .$$

Lemma 2.13.13 $r \in \mathrm{NF}$ *iff* r *is in normal form.*

Remark: This means that terms in normal form are those which are the result of applying successively introduction steps to terms which are the result of applying successively eliminations to variables, where only the

last elimination is an $+$-elimination, and such that for all terms used in eliminations the same holds.
Proof of Lemma 2.13.13:
"$\Rightarrow$": immediate.
"$\Leftarrow$": Induction on $\mathrm{lgh}(r)$. If

$$r = \lambda x.r', \quad \iota_i(r') \ ,$$

by IH $r' \in \mathrm{NF}$, $r \in \mathrm{NF}$. If

$$r = \langle r', s' \rangle_i \ ,$$

by IH $r', s' \in \mathrm{NF}$, $r \in \mathrm{NF}$. Otherwise

$$r = r_0 \ R_1 \cdots R_n \ ,$$

s.t. r_0 is not of the form

$$s \ S \ .$$

Then r_0 must be a variable,

$$R_0, \ldots, R_{n-1} \in \mathrm{Term}_{0,1} \ ,$$

$$R_i \in \mathrm{NF}_{0,1}(\mathrm{i{=}0}\ , \ \ldots\ , \ \mathrm{n\text{-}1}) \ ,$$

$$\begin{array}{l} R_n \in \mathrm{NF}_{0,1} \text{ or } R_n \equiv \mathrm{efq} \text{ or} \\ R_n \equiv [\lambda x, y.s] \text{ or } R_n \equiv [\lambda x.s, \lambda y.t] \text{ with } s, t \in \mathrm{NF} \ , \end{array}$$

$$r \in \mathrm{NF} \ .$$

Definition 2.13.14 (a) A term in normal form is in long η-normal form, iff for all maximal subterms of the form

$$s = x \ r_1 \cdots r_n \ R$$

it holds that

$$s : \mathrm{o} \text{ or } s : \emptyset \text{ or } R \text{ is of the form efq}, [\lambda x, y.t] \text{ or } [\lambda x.t_0, \lambda y.t_1] \ .$$

(b) The set of terms $\eta - \mathrm{NF}$ is inductively defined as:

- If $r_i \in (\eta - \mathrm{NF})_{0,1}$ $(i = 1, \ldots, n)$, $x \ r_1 \cdots r_n : \sigma$ where $\sigma \in \{\mathrm{o}, \emptyset\}$, then
$$x \ r_1 \cdots r_n \in (\eta - \mathrm{NF}) \ .$$
- If $r_i, s, t \in (\eta - \mathrm{NF})_{0,1}$, then
$$x \ r_1 \cdots r_n \ \mathrm{efq}, \quad x \ r_1 \cdots r_n \ [\lambda x, y.s], \quad x \ r_1 \cdots r_n \ [\lambda y.s, \lambda z.t] \in \eta - \mathrm{NF} \ .$$

- If $r, s \in \eta - \mathrm{NF}$, so are

$$\lambda x.s, \langle r, s\rangle_i, \iota_i(r) \in \eta - \mathrm{NF} \ .$$

(c) For terms of the form $r \equiv x\ r_1 \cdots r_n$ with $r_i \in \mathrm{Term}_{0,1}$, $r : \rho$ we define $\exp(r)$ by induction on ρ by:

- If $\rho = 0$ or $\rho = \emptyset$, then $\exp(r) := r$.
- If $\rho = \sigma \to \tau$, then $\exp(r) := \lambda x.\exp(r\ \exp(x))$ for a new variable x.
- If $\rho = \sigma \times_0 \tau$, $\exp(r) := \langle \exp(r\ 0), \exp(r\ 1)\rangle_0$
- If $\rho = \sigma \times_1 \tau$, $\exp(r) := r\ [\lambda x, y.\langle \exp(x), \exp(y)\rangle_1]$.
- If $\rho = \sigma + \tau$, $\exp(r) := r\ [\lambda x.\iota_0(\exp(x)), \lambda x.\iota_1(\exp(x))]$.

(d) We define for terms r in normal form by induction on the length of r the η-expansion r^η:

- $(x\ r_0 \cdots r_n)^\eta := \exp(x\ r_0^\eta \cdots r_n^\eta)$.
- $(x\ r_0 \cdots r_n\ \mathrm{efq})^\eta := x\ r_0^\eta \cdots r_n^\eta\ \mathrm{efq}$.
- $(x\ r_0 \cdots r_n\ [\lambda x, y.s])^\eta := x\ r_0^\eta \cdots r_n^\eta\ [\lambda x, y.s^\eta]$.
- $(x\ r_0 \cdots r_n\ [\lambda x.s, \lambda y.t])^\eta := x\ r_0^\eta \cdots r_n^\eta\ [\lambda x.s^\eta, \lambda y.t^\eta]$.
- $(\lambda x.r)^\eta := \lambda x.r^\eta$.
- $\langle r, s\rangle_i^\eta := \langle r^\eta, s^\eta\rangle_i$.
- $\iota_i(r)^\eta := \iota_i(r^\eta)$.

Lemma 2.13.15 *(a) r is in long η-normal form if $r \in \eta - \mathrm{NF}$.*

(b) If $r \equiv x\ r_1 \cdots r_n$, $r_i \in (\eta - \mathrm{NF})_{0,1}$, then $\exp(r) \in \eta - \mathrm{NF}$.

(c) If r is in normal form, then $r^\eta \in \eta - \mathrm{NF}$.

Proof: Easy

Remark: η-reductions can now be defined for terms in normal form as follows: Take a maximal subterm of r of the form

$$s \equiv x\ r_0 \cdots r_n\ R : \rho \ .$$

Assume that then $R \in \mathrm{Term}_{0,1}$ and ρ is not a type o or $\emptyset$. Then replace s by

- $\lambda x.s\ x$ för x fresh,
- $\langle s\ 0, s\ 1\rangle_0$,
- $s\ [\lambda x, y.\langle x, y\rangle_1]$ or
- $s\ [\lambda x.\iota_0(x), \lambda x.\iota_1(x)]$

(depending on the type of s).
One can show now that η-reductions reduce a term t to its long η-normal form t^η.

2.14 The Curry-Howard Isomorphism

Assumption 2.14.1 *In the following let $\mathcal{L}$ be some language for intuitionistic predicate logic.*

Definition 2.14.2 (a) For every formula A in $\mathcal{L}$ we assign some type $\sigma(A)$ (where $\neg A$ is an abbreviation for $A \rightarrow \bot$):

- If A is prime, $A \not\equiv \bot$, then $\sigma(A) := \mathrm{o}$.
- $\sigma(\bot) := \emptyset$.
- $\sigma(A \rightarrow B) := \sigma(A) \rightarrow \sigma(B)$.
- $\sigma(A \vee B) := \sigma(A) + \sigma(B)$.
- $\sigma(A \wedge B) := \sigma(A) \times_0 \sigma(B)$.
- $\sigma(\forall x.A) := \mathrm{o} \rightarrow \sigma(A)$.
- $\sigma(\exists x.A) := \mathrm{o} \times_1 \sigma(A)$.

(b) We assume an assignment of either a formula A s.t. $\sigma(A) = \sigma$ or, if $\sigma = \mathrm{o}$, the symbol nat, or if

$$\sigma = \underbrace{\mathrm{o} \rightarrow \cdots \rightarrow \mathrm{o}}_{n \text{ times}} \rightarrow \mathrm{o}$$

an n-ary function symbol to every variable of type σ s.t. for every A and for nat there are infinitely many variables, to which this is assigned, and to every function symbol there is exactly one variable this is assigned.

We write

$$x^A, y^A, z^A$$

for variables to which A is assignment, sometimes as well

$$x : A, y : A, z : A$$

(as usual with indices and accents), and similarly with nat. We write f for the variable to which f is assigned.

Further we replace efq_σ, $\iota_{i,\sigma}$ by many copies efq_A, $\iota_{i,A}$ for every formula A s.t. $\sigma(A) = \sigma$, which are each treated as efq_σ, $\iota_{i,A}$ before.

(c) The set of ground terms r of type o with their assigned term t of $\mathcal{L}$, written as r^t or $r : t$ is inductively defined by:

- $$x^{\mathrm{nat}} : x \ .$$
- If f is a n-ary function symbol, $s_i : t_i$, then

$$f\ s_1 \cdots s_n : f(t_1, \ldots, t_n) \ .$$

(d) The set of proof terms r together with the formula A it proves is inductively defined by (we write $r : A$ or r^A for r proves A)

- If A is a formula, then
$$x^A : A \enspace .$$
- If $r : B$, then
$$(\lambda x^A.r) : A \to B \enspace .$$
- If $r : A$, $s : B$ then
$$\langle r, s\rangle_0 : A \wedge B \enspace .$$
- If $r : A_i$, then
$$\iota_{i,A_{1-i}}(r) : A_0 \vee A_1 \enspace .$$
- If $r : A$, x not free in B for $y^B \in \mathrm{FV}(r)$, then
$$(\lambda x^{\mathrm{nat}}.r) : \forall x.A \enspace .$$
- If $r : t$, $s : B[x := t]$, then
$$\langle r, s\rangle_1 : \exists x.B \enspace .$$
- If $r : A \to B$, $s : A$, then
$$(r\ s) : B \enspace .$$
- If $r : A_0 \wedge A_1$, then
$$(r\ i) : A_i \enspace .$$
- If $r : A \vee B$, $s : C$, $t : C$, then
$$(r\ [\lambda x^A.s, \lambda y^B.t]) : C \enspace .$$
- If $r : \forall x.A$, $s : t$, then
$$(r\ s) : A[x := t] \enspace .$$
- If $r : \exists x.A$, $s : B$, $x \notin \mathrm{FV}(C)$ for $y^C \in \mathrm{FV}(s)$ then
$$(r\ [\lambda x^{\mathrm{nat}}, y^A.s^B]) : B \enspace .$$
- If $r : \bot$, then
$$(r\ \mathrm{efq}_A) : A \enspace .$$

Brackets are omitted as usual.

(e) For each proof term $r : A$ with free variables

$$x_i^{\mathrm{nat}} \quad (i = 1, \ldots, n), \quad y_j^{B_j} \quad (j = 1, \ldots, m) \ ,$$

and Γ s.t.

$$B_i \in \Gamma$$

assign we a derivation of $\Gamma \Rightarrow A$ in natural deduction for intutionistic predicate calculus (without equality) (more precisely in the following the derivation depends on the choice of Γ, but this influences only the choice of weakenings):

- x^A corresponds

$$\text{(Weak)}\ \frac{A \Rightarrow A}{\Gamma \Rightarrow A}$$

- $\lambda x^A.r^B$ corresponds to

$$(\to\!-\mathrm{I})\ \frac{\overset{[r^B]}{\Gamma, A \Rightarrow B}}{\Gamma \Rightarrow A \to B}$$

- $\langle r^A, s^B\rangle_0$ corresponds to

$$(\wedge\!-\mathrm{I})\ \frac{\overset{r^A}{\Gamma \Rightarrow A} \qquad \overset{s^B}{\Gamma \Rightarrow B}}{\Gamma \Rightarrow A \wedge B}$$

- $\iota_{i,A_{1-i}}(r^{A_i})$ corresponds to

$$(\vee\!-\mathrm{I})_i\ \frac{\overset{r^{A_i}}{\Gamma \Rightarrow A_i}}{\Gamma \Rightarrow A_0 \vee A_1}$$

- $\lambda x^{\mathrm{nat}}.r^A$ corresponds to (where Δ are the formulas in Γ in which x does not occur free)

$$\begin{array}{r} (\forall\!-\mathrm{I}) \\ \text{(Weak)} \end{array} \ \frac{\dfrac{\overset{r^A}{\Delta \Rightarrow A}}{\Delta \Rightarrow \forall x.A}}{\Gamma \Rightarrow \forall x.A}$$

- $\langle r^t, s^B[x := t]\rangle_1$ corresponds to

$$(\exists\!-\mathrm{I})\ \frac{\overset{s^{B[x:=t]}}{\Gamma \Rightarrow B[x := t]}}{\Gamma \Rightarrow \exists x.B}$$

- $r^{A \to B}\ s^A$ corresponds to

$$(\to\!-\mathrm{E})\ \frac{\overset{r^{A\to B}}{\Gamma \Rightarrow A \to B} \qquad \overset{s^A}{\Gamma \Rightarrow A}}{\Gamma \Rightarrow B}$$

- $r^{A_0 \wedge A_1}\ i$ corresponds to

$$\text{(}\wedge\text{−E}_i\text{)}\ \frac{\overset{r^{A_0\wedge A_1}}{\Gamma\Rightarrow A_0 \wedge A_1}}{\Gamma\Rightarrow A_i}$$

- $r^{A\vee B}[\lambda x^A.s^C, \lambda x^B.t^C]$ corresponds to

$$\text{(}\vee\text{−E)}\ \frac{\overset{r^{A\vee B}}{\Gamma\Rightarrow A\vee B} \qquad \overset{s^C}{\Gamma, A\Rightarrow C} \qquad \overset{t^C}{\Gamma, B\Rightarrow C}}{\Gamma\Rightarrow C}$$

- $r^{\forall x.A}\ s^t$ corresponds to

$$\text{(}\forall\text{−E)}\ \frac{\overset{r^{\forall x.A}}{\Gamma\Rightarrow\forall x.A}}{\Gamma\Rightarrow A[x := t]}$$

- $r^{\exists x.A}[\lambda x^{\text{nat}}, y^A.s^B]$ corresponds to (where Δ are the formulas in Γ in which x does not occur free)

$$\text{(}\exists\text{−E)}\ \frac{\overset{r^{\exists x.A}}{\Gamma\Rightarrow\exists x.A} \qquad \overset{s^B}{\Delta, A\Rightarrow B}}{\Gamma, \Delta\Rightarrow B}$$

- $r^{\perp}\ \text{efq}_A$ corresponds to

$$\text{(EFQ)}\ \frac{\overset{r^{\perp}}{\Gamma\Rightarrow\perp}}{\Gamma\Rightarrow A}$$

Remark: The above is the first part of the

Curry-Howard isomorphism,

in which to every one assigns to every proof of natural deduction a typed λ-term. The second part will be, that normalization of the typed λ-calculus corresponds to normalization of proofs.
If we identify all prime-formulas except $\perp$, then elements of

$$\sigma(A)$$

in the standard intepretation of the types correspond to proofs in the Brouwer-Heyting-Kolmogorov-interpretation of the logical connectives (here o is interpreted as an arbitrary subset of the natural numbers, even possible empty, corresponding to the set of proofs of this prime-formula, about which BHK does not say anything):

- A BHK-proof of

$$A\rightarrow B$$

is a method transforming an proof of A into a proof of B. An element of

$$\sigma(A)\rightarrow\sigma(B)$$

is an algorithm, mapping elements of $\sigma(A)$ to elements of $\sigma(B)$.

- A BHK-proof of
  $$A \vee B$$
  is a proof of A or a proof of B plus the information, which one it is. An element of
  $$\sigma(A) + \sigma(B)$$
  is either an element of $\sigma(A)$ or of $\sigma(B)$ plus the information which one it is. Etc.

Not every λ-term of type $\sigma(A)$ will correspond now to a proof of A, since not all prime-formulas are equivalent. E.g. if

$$P \not\equiv Q$$

are diffferent prime-formulas, then

$$\lambda x^P.x$$

is of type

$$\sigma(P \to Q)$$

but not a proof of

$$P \to Q \ .$$

However, every λ-term assigned to a proof in natural deduction is of course a proof of the corresponding formula, the set of terms build according the above is a subset of the λ-terms of type $\sigma(A)$.

Definition 2.14.3 (a) In the following we assume, that if $A \not\equiv A'$, $x_i^A \in \mathrm{FV}(r) \cup \mathrm{BV}(r)$ for a proof term t, then $x_i^{A'} \notin \mathrm{FV}(r) \cup \mathrm{BV}(r)$, similarly for A vs. nat.

(b) Two proof terms are
$$\alpha\text{-equivalent},$$
if they are identical up to α-equivalence or replacing of bounded variables x^A by $x^{A'}$ for A and A' not α-equivalent, and their corresponding natural deduction derivations coinicide up to α-conversion of formulas. (A definition of this directly on proof terms becomes quite complicated – the better way of defining it directly is by defining it as α-equivalence of corresponding Martin-Löf type theory derivations).

(c) Substitution
$$r[x^A := s]$$
of an assumption variable x^A by a proof term $s : A$ in a derivation $r : B$ is defined as ordinary substitution, but by renaming of variables in such a way that the result is a proof term, in which all assumptions A indicated by
$$x^A$$

are replaced by the proof s, and other bounded variables possibly renamed s.t. all formulas remain α-equivalent.

Similarly

$$r[x^{\mathrm{nat}} := s]$$

is defined for $s : t$, s.t. in the corresponding derivation, all formulas B, in which x is not later bound, are replaced by

$$B[x := s]$$

up to α-conversion.

Again a formal definition is better given in the context of Martin-Löf type theory.

(d) We identify in the following α-equivalent proof terms and formulas.

Lemma 2.14.4 *(a) If $r : A$ then r is a term of type $\sigma(A)$.*

(b) If

$$r : A, \quad s : t \enspace ,$$

r has assumption variables

$$y_1^{A_1}, \ldots, y_m^{A_m} \enspace ,$$

then

$$r[x := s] : A[x := t]$$

with assumption variables

$$y_1^{A_1[x:=t]}, \ldots, y_m^{A_m[x:=t]} \enspace .$$

(c) If

$$r : A, \quad s : B \enspace ,$$

r has assumption variables

$$y_1^{A_1}, \ldots, y_m^{A_m}$$

and possibly y^B, then

$$r[y^B := s] : A \enspace ,$$

and has assumption variables

$$y_1^{A_1}, \ldots, y_m^{A_m} \enspace .$$

Proof: Immediate (if one wants this precise, one better does it in the context of Martin-Löf type theory).

Theorem 2.14.5 *Assume*

$$r : A, \quad r \longrightarrow s \ ,$$

and that, when substitutions are carried out, renamings of variables are carried out in accordance with Lemma 2.14.4. Then

$$s : A \ .$$

Proof:
Immediate by Lemma 2.14.4.

We will look now at all reductions and see, what proof transformations are carried out. Further we will look at, how to justify them with the BHK-interpretation.

$$(\lambda x.r)\ s \longrightarrow r[x := s] \ .$$

$(\lambda x^A.r^B)\ s^A$ corresponds to the following proof:

$$\dfrac{(\to-\mathrm{I})\ \dfrac{\begin{matrix}[x^A : A] & [x^A : A] & [x^A : A] \\ \vdots & \vdots & \vdots \\ & r : B & \end{matrix}}{\lambda x.r : A{\to}B} \qquad s : A}{(\lambda x.r)\ s : B}\ (\to-\mathrm{E})$$

The proof $r[x := s]$ is as follows:

$$\begin{matrix} s : A & s : A & s : A \\ \vdots & \vdots & \vdots \\ & r[x := s] : B & \end{matrix}$$

So we replace in the proof r of A those assumptions of A denoted by

$$x^A$$

by the direct proof

$$s : A$$

and get therefore a proof of B, which does not make the detour via an introduction and then an elimination.
In the BHK-interpretation this correpends to the following: Take a proof of B from a hypothetical proof of A. We get a proof of

$$A{\to}B$$

as the function, which takes a proof of A and uses the method above to obtain a proof of B.

Instead we can however replace the hypothetical proof of A by the concrete proof of A in the proof of B and get again a proof of

$$B \ .$$

Note that the λ-term and therefore the proof as well might become longer, if the variable x occurs more than once.
The above corresponds to the elimination of a lemma:
If we have a lemma A and have a proof of B using as assumption A, then we get a proof of B without using A by making an $\rightarrow$-introduction w.r.t. A and then an $\rightarrow$-elimination with the proof of the lemma. We can get a direct proof of B by replacing each use of the lemma by a direct proof. If we use the lemma at most once, the resulting proof has the same size or shorter (if the length of the proof of the lemma is taken into consideration). However, if we use the lemma more than once, the direct proof will be longer. The other reductions (except permutative conversions) correspond to a generalization of the elimination of lemmata.

$$\langle r_0^{A_0}, r_1^{A_1}\rangle_0 \ i \longrightarrow r_i \ .$$

$\langle r_0^{A_0}, r_1^{A_1}\rangle_0)$ 0 corresponds to the following proof:

$$\cfrac{\cfrac{r_0 : A_0 \qquad r_1 : A_1}{\langle r_0, r_1\rangle_0 : A_0 \wedge A_1}\,(\wedge\mathrm{-I})}{r_i : A_i}\,(\wedge\mathrm{-E})$$

The reduced proof r_i is as follows:

$$r_i : A_i$$

In the BHK interpretation, one started with a proof of A_0 and of A_1, got a proof of

$$A_0 \wedge A_1$$

by forming the pair of proofs and then a proof of

$$A_i$$

by taking the ith component. However we could have taken directly the ith component.
This reduction always reduces the length of the λ-term, but we might arrive at this reduction only after several other reduction steps.

$$\iota_i(r^{A_i}) \ [\lambda x_0^{A_0}.s_0^D, \lambda x_1^{A_1}.s_1^D] \longrightarrow s_i^D[x_i := r] \ .$$

$\iota_i(r^{A_i})\ [\lambda x_0^{A_0}.s_0^D, \lambda x_1^{A_1}.s_1^D]$ corresponds to the following derivation:

$$\cfrac{\cfrac{r : A_i}{\iota_i(r) : A_0 \vee A_1}\,(\vee\text{-I}) \qquad \begin{matrix} x_0 : A_0 \\ \vdots \\ s_0 : D \end{matrix} \qquad \begin{matrix} x_1 : A_1 \\ \vdots \\ s_1 : D \end{matrix}}{\iota_i(r^{A_i})\ [\lambda x_0^{A_0}.s_0^D, \lambda x_1^{A_1}.s_1^D] : D}\,(\vee\text{-E})$$

$s_i[x_i := r]$ corresponds to

$$\begin{matrix} r : A_i \\ \vdots \\ s_i : D \end{matrix}$$

So we replace in the proof s_i of D all assumptions marked by

$$x_i^{A_i}$$

by the proof

$$r : A_i \ .$$

Note that this proof might be longer than the original proof.
The relationship to the BHK-interpretation is again clear and will be omitted in the following.

$$(\lambda x^{\text{nat}}.r^A)\ s^t \longrightarrow r[x := s]^{A[x:=t]} \ .$$

$(\lambda x^{\text{nat}}.r)\ s$ corresponds to the following derivation:

$$\cfrac{\cfrac{\begin{matrix} x^{\text{nat}} \\ \vdots \\ r : A \end{matrix}}{\lambda x^{\text{nat}}.r : \forall x.A}\,(\forall\text{-I})}{(\lambda x^{\text{nat}}.r)\ s : A[x := t]}\,(\forall\text{-E})$$

$r[x := s]$ corresponds to

$$\begin{matrix} s^t \\ \vdots \\ r[x := s] : A[x := t] \end{matrix}$$

So we replace in the proof of A all occurrences of x by s and obtain a direct proof of $A[x := t]$.

$$\langle r^{\text{nat}}, s^A\rangle_1\ [\lambda x^{\text{nat}}, y^A.t^D] \longrightarrow t[x := r, y := s] \ .$$

$\langle r^t, s_0^{A[x:=t]}\rangle_1\ [\lambda x^{\mathrm{nat}}, y^A.s_1^D]$ corresponds to

$s_1[x := r, y := s_0]$ corresponds to

$$\begin{array}{cc} r^{\mathrm{nat}} & s_0 : A[z := r] \\ & \vdots \\ \multicolumn{2}{c}{s_1[x := r, y := s_0] : D} \end{array}$$

So we replace in the proof s_1 of D, x by r and assumptions of A marked by y by the proof s of $A[x := t]$.

$$r^{A_0\vee A_1}\ [\lambda x_0^{A_0}.s_0^C, \lambda x_1^{A_1}.s_1^C]\ R \longrightarrow r^{A_0\vee A_1}\ [\lambda x_0^{A_0}.(s_0^C\ R), \lambda x_1^{A_1}.(s_1^C\ R)]\ .$$

Let for instance $C \equiv D\to E$, R is a term.

$$r^{A_0\vee A_1}\ [\lambda x_0^{A_0}.s_0^C, \lambda x_1^{A_1}.s_1^C]\ R$$

corresponds to

$$\dfrac{\dfrac{r : A_0 \vee A_1 \qquad \begin{array}{c} x_0 : A_0 \\ \vdots \\ s_0 : D\to E \end{array} \qquad \begin{array}{c} x_1 : A_1 \\ \vdots \\ s_1 : D\to E \end{array}}{r^{A_0\vee A_1}\ [\lambda x_0^{A_0}.s_0^C, \lambda x_1^{A_1}.s_1^C] : D\to E} \qquad R : D}{r^{A_0\vee A_1}\ [\lambda x_0^{A_0}.s_0^C, \lambda x_1^{A_1}.s_1^C]\ R : E}$$

$r^{A_0\vee A_1}\ [\lambda x_0^{A_0}.s_0^C\ R, \lambda x_1^{A_1}.s_1^C\ R]$ corresponds to

$$\dfrac{r : A_0 \vee A_1 \qquad \dfrac{\begin{array}{c} x_0 : A_0 \\ \vdots \\ s_0 : D\to E \end{array} \qquad R : D}{s_0\ R : E} \qquad \dfrac{\begin{array}{c} x_1 : A_1 \\ \vdots \\ s_1 : D\to E \end{array} \qquad R : D}{s_1\ R : E}}{r^{A_0\vee A_1}\ [\lambda x_0^{A_0}.s_0^C\ R, \lambda x_1^{A_1}.s_1^C\ R] : E}$$

This reduction pushes the $\vee$-elimination as much down as possible (till an introduction is reached).

$$r^{\exists x.A}\ [\lambda x^{\mathrm{nat}}, y^A.s^C]\ R \longrightarrow r^{\exists x.A}\ [\lambda x^{\mathrm{nat}}, y^A.s^C\ R]\ .$$

Let for instance $C \equiv \exists z.D$, $R \equiv [\lambda z_0^{\mathrm{nat}}, z_1^D.t^E]$.

$r^{\exists x.A}\ [\lambda x^{\mathrm{nat}}.y^A.s^{\exists z.D}]\ [\lambda z_0^{\mathrm{nat}}, z_1^D.t^E]$ corresponds to

$$\dfrac{\dfrac{r : \exists x.A \qquad \begin{array}{c} x^{\mathrm{nat}} \quad y : A \\ \vdots \\ s : \exists z.D \end{array}}{r\ [\lambda x, y.s] : \exists z.D} \qquad \begin{array}{c} z_0^{\mathrm{nat}} \quad z_1^D \\ \vdots \\ s : E \end{array}}{r\ [\lambda x, y.s]\ [\lambda z_0, z_1.s] : E}$$

$r^{\exists x.A}\ [\lambda x^{\mathrm{nat}}.y^A.s^{\exists z.D}\ [\lambda z_0^{\mathrm{nat}}, z_1^D.t^E]]$ corresponds to

$$\dfrac{r : \exists x.A \qquad \dfrac{\begin{array}{c} x^{\mathrm{nat}} \quad y : A \\ \vdots \\ s : \exists z.D \end{array} \qquad \begin{array}{c} z_0^{\mathrm{nat}} \quad z_1^D \\ \vdots \\ s : E \end{array}}{s\ [\lambda z_0, z_1.s] : E}}{r\ [\lambda x, y.s\ [\lambda z_0, z_1.s]] : \exists z.D}$$

Remark: A derivation in normal form looks now as follows:
Its proof term is the result of application of introductions to a term which is the result of eliminations to a variable, and the same holds with all elimination terms.
Therefore the derivation starts with an assumption (if we have additional axioms, they would be treated as assumptions), then several elimination rules are applied and to the final formal several introductions. Only the last of the elimination rules is an ∃- or ∨-elimination.
A path from the derived formula through formulas proved by introduction rules and then through the main formulas of elimination rules till en assumed formula is called a *main path* (because of $(\wedge - \mathrm{I})$ there are several main path.
Now the derivation of the side-formulas of elimination rules follows the same pattern, starting from assumptions (which can be in case of $(\vee - \mathrm{E})$, $(\exists - \mathrm{I})$ discharged assumption) through elimination rules and then introduction rules.
The paths defined as the main path, but ending at a side-formula of an elimination rule are called *side-paths*.

Remark 2.14.6 *All formulas in a normal derivation are subformulas of the derived formula or of assumptions.*

Proof: Immediate, by the consideration before.

Lemma 2.14.7 *If r is a proof term which derives A from assumption variables $x_i^{A_i}$, after renaming of variables r^η is a proof term deriving A from the same assumption variables*

Proof: First one shows that if

$$r_i : A_i \text{ normal or} r_i \in \{0, 1\} \ ,$$

and

$$x \, r_1 \cdots r_n$$

is a proof term, then

$$\exp(x \, r_1 \cdots r_n)$$

is as well a proof term of the same formula with same assumption variables, by induction on the derived formula. Then one shows the assertion by induction on the length of r.

Remark: The η-expanded derivations is normal and has the property that if the last elimination rule of the path is not ($\exists$-E) or ($\vee$-E), then the resulting formula (or if there is no elimination rule at all the proved formula) is not of the form

$$A \rightarrow B \text{ or } A \wedge B \ .$$

Chapter 3

A Brief Introduction to Martin-Löf's Type Theory

3.1 Motivation

In the last section we attached to a proof term two kinds of types: a simple type and a formula, which was something like a type. This led to some confusion and to technical problems.
It is conceptually clearer, to treat formulas as real types. The first consequence is

- We need dependent types: In the rule ($\mathbf{N}$ will now be the set of natural numbers)

$$\frac{r : \forall x : \mathbf{N}.A(x) \quad s : \mathbf{N}}{r\ s : A(s)}$$

  we see that the type $A(x)$ depends on $x : \mathbf{N}$.

Further we have that the types

$$\forall x : \mathbf{N}.B(x) \text{ and } A \to B$$

have the same rule for building elements (except that in $A \to B$ B does not depend on A). We write instead

$$\Pi x : \mathbf{N}.B(x) \text{ and } \Pi x : A.B \ \ .$$

Consequences

- Types which represent sets and which represents formulas (called in type theory propositions) can be identified.
- However, unless we restrict possibilities for building dependent basic formulas (those corresponding to prime formulas) to those built from elements of some basic types which represent sets only, types can now depend from proof terms.

- Therefore types have to be derived as terms. (E.g. the equality type

$$\mathbf{I}(\forall x : \mathbf{N}.A(x), r, s)$$

which is the proposition

$$r, s \text{ are equal elements of } \forall x : \mathbf{N}.A(x)$$

can only be constructed, if we have proved before

$$\begin{array}{l} r : \forall x : \mathbf{N}.A(x) \\ s : \forall x : \mathbf{N}.A(x) \ \ ) \end{array}$$

- Therefore we have two (so called) judgements:

$$\begin{array}{l} A : \mathbf{type} \\ r : A \end{array}$$

- In order to organize equalities (w.r.t. α, β, η-conversion) we add two more judgements

$$\begin{array}{rll} A = B : \mathbf{type} & \text{for} & A, B \text{ are } (\alpha, \beta, \eta\text{-})\text{equal types} \\ r = s : A & \text{for} & A, B \text{ are } (\alpha, \beta, \eta\text{-})\text{equal elements of the type } A \ . \end{array}$$

The equality above is called judgemental equality.

In order to make sense of

$$\forall x : \mathbf{N}.A(x), \quad A(s)$$

we introduce a dependend type structure on top of what we had before.
The types before are now elements of a new type, called

$$\mathbf{Set} \ ,$$

and A just mentioned is element of the type

$$\mathbf{N} \rightarrow \mathbf{Set} \ .$$

Further because we have dependent types we need dependent judgements of the form

$$x_1 : A_1, \ldots, x_n : A_n \Rightarrow \theta$$

where θ is a judgement as mentioned before.

3.2 The Logical Framework

The dependent λ-calculus on top of **Set** is called the logical framework (or a logical framework, there are variants).
We will in the following first introduce rules of Martin-Löf's type theory.
We will then see and refer to an intuitive understanding what substitution

$$A[x := t]$$

and α conversion means. In the later Section 3.5 will we then give a precise definition of the language of Martin-Löf's type theory and how substitution, α-conversion is defined. We will there distinguish between pre-types, which potentially occur as

$$A : \mathbf{type}$$

(**type** will not be a pre-type) and pre-terms, objects which potentially occur as r in a judgement

$$r : A \enspace .$$

In the following

- $a, b, c, n, f, g, r, s, t$ will be pre-terms,
- A, B, C pre-types,
- x, y, z, u, v, X, Y, Z variables (X, Y, Z stand for variabels which are supposed to be elements of real types whereas x, y, z, u, v are elements of sets),
- Δ, Γ pre-contexts (of the form $x_1 : A_1, \ldots, x_n : A_n$),
- θ pre-nondependent-judgements, i.e. expressions of the form $r : A$, $r = s : A$, $A : \mathbf{type}$, $A = B : \mathbf{type}$.

A judgement is an expression of the form

$$\Gamma \Rightarrow \theta \enspace ,$$

with Γ, θ as before.
We identify in the following α-equivalent pre-terms, pre-types, pre-contexts, pre-nondependent judgements and pre-judgements.
We write

- $\emptyset$ for the empty context,
- θ (if used as a judgement) for $\emptyset \Rightarrow \theta$.

Remark on the premisses of the rules: In the following rules some assumptions can be omitted. For instance it follows from the assumption

$$x : A \Rightarrow B : \mathbf{type}$$

of (→-F)

$$A : \mathbf{type}\ .$$

We take a choice which makes the rules look nice, so esthetic criteria govern our choice. The "official" version demands that all sub-judgements of a judgement have to be assumptions of the rule, where the set of sub-judgements

$$\mathcal{D}(\Gamma \Rightarrow A)$$

of a judgement $\Gamma \Rightarrow A$ is defined as
($\mathcal{D}^+(\Gamma \Rightarrow A) := \mathcal{D}(\Gamma \Rightarrow A) \cup \{\Gamma \Rightarrow A\}$)

- $\mathcal{D}(\Gamma \Rightarrow A = B : \mathbf{type}) := \mathcal{D}^+(\Gamma \Rightarrow A : \mathbf{type}) \cup \mathcal{D}^+(\Gamma \Rightarrow B : \mathbf{type})$.
- $\mathcal{D}(\Gamma \Rightarrow r = s : A : \mathbf{type}) := \mathcal{D}^+(\Gamma \Rightarrow r : A) \cup \mathcal{D}^+(\Gamma \Rightarrow s : A)$.
- $\mathcal{D}(\Gamma \Rightarrow r : A) := \mathcal{D}^+(\Gamma \Rightarrow A : \mathbf{type})$.
- $\mathcal{D}(\Gamma, x : A \Rightarrow B : \mathbf{type}) := \mathcal{D}^+(\Gamma \Rightarrow A\mathbf{type})$.
- $\mathcal{D}(\emptyset \Rightarrow A : \mathbf{type}) := \emptyset$.

But this does not look very nice.

Assumption and Weakening

(Ass) $\dfrac{\Gamma \Rightarrow A : \mathbf{type}}{\Gamma, x : A \Rightarrow x : A}$ (Weak) $\dfrac{\begin{array}{c}\Gamma \Rightarrow A : \mathbf{type} \\ \Gamma \Rightarrow \theta\end{array}}{\Gamma, x : A \Rightarrow \theta}$ $(x \notin \mathrm{FV}(\theta))$

All the following rules, which are not axioms (i.e. no premisses) can be weakened by a context, i.e.

(Rule) $\dfrac{\begin{array}{c}\Delta_1 \Rightarrow \theta_1 \\ \cdots \\ \Delta_n \Rightarrow \theta_n\end{array}}{\Delta \Rightarrow \theta}$

$(n > 0)$ denotes the rule

(Rule) $\dfrac{\begin{array}{c}\Gamma, \Delta_1 \Rightarrow \theta_1 \\ \cdots \\ \Gamma, \Delta_n \Rightarrow \theta_n\end{array}}{\Gamma, \Delta \Rightarrow \theta}$

Judgemental Equality rules

(Ref$_0$) $\dfrac{A : \mathbf{type}}{A = A : \mathbf{type}}$ (Ref$_1$) $\dfrac{r : A}{r = r : A}$

(Sym$_0$) $\dfrac{A = B : \mathbf{type}}{B = A : \mathbf{type}}$ (Sym$_1$) $\dfrac{r = s : A}{s = r : A}$

(Trans$_0$) $\dfrac{\begin{array}{c} A = B : \mathbf{type} \\ B = C : \mathbf{type} \end{array}}{A = C : \mathbf{type}}$ (Trans$_1$) $\dfrac{\begin{array}{c} r = s : A \\ s = t : A \end{array}}{r = t : A}$

(Repl$_0$) $\dfrac{\begin{array}{c} r : A \\ A = B : \mathbf{type} \end{array}}{r : B}$ (Repl$_1$) $\dfrac{\begin{array}{c} r = s : A \\ A = B : \mathbf{type} \end{array}}{r = s : B}$

The Type Set

(**Set**-I) $\mathbf{Set} : \mathbf{type}$

(**Set**-E) $\dfrac{A : \mathbf{Set}}{A : \mathbf{type}}$ (**Set**-E$_=$) $\dfrac{A = B : \mathbf{Set}}{A = B : \mathbf{type}}$

Remark: Sometimes the last two rules are written as

(**Set**-E*) $\dfrac{A : \mathbf{Set}}{\mathbf{El}(A) : \mathbf{type}}$ (**Set**-E$^*_=$) $\dfrac{A = B : \mathbf{Set}}{\mathbf{El}(A) = \mathbf{El}(B) : \mathbf{type}}$

But this leads only to an unnecessary technical overload.

The Dependend Function Type

The formation rules of $\rightarrow$:

(→-F) $\dfrac{\begin{array}{c} A : \mathbf{type} \\ x : A \Rightarrow B : \mathbf{type} \end{array}}{(x : A) \rightarrow B : \mathbf{type}}$

(→-F$_=$) $\dfrac{\begin{array}{c} A = A' : \mathbf{type} \\ x : A \Rightarrow B = B' : \mathbf{type} \end{array}}{(x : A) \rightarrow B = (x : A') \rightarrow B' : \mathbf{type}}$

The introduction rules of $\rightarrow$:

(→-I) $\dfrac{\begin{array}{c} A : \mathbf{type} \\ x : A \Rightarrow B : \mathbf{type} \\ x : A \Rightarrow s : B \end{array}}{(x)s : (x : A) \rightarrow B}$ (→-I$_=$) $\dfrac{\begin{array}{c} A : \mathbf{type} \\ x : A \Rightarrow B : \mathbf{type} \\ x : A \Rightarrow s = s' : B \end{array}}{(x)s = (x)s' : (x : A) \rightarrow B}$

The elimination rules of $\rightarrow$:

$$(\to\text{-E})\quad \frac{\begin{array}{c} A : \mathbf{type} \\ x : A \Rightarrow B : \mathbf{type} \\ r : (x : A) \to B \\ s : A \end{array}}{r\ s : B[x := s]} \qquad (\to\text{-E})\quad \frac{\begin{array}{c} A : \mathbf{type} \\ x : A \Rightarrow B : \mathbf{type} \\ r = r' : (x : A) \to B \\ s = s' : A \end{array}}{r\ s = r'\ s' : B[x := s]}$$

The equality rules of $\to$:

$$(\to\text{-=})\quad \frac{\begin{array}{c} A : \mathbf{type} \\ x : A \Rightarrow B : \mathbf{type} \\ x : A \Rightarrow r : B \\ s : A \end{array}}{((x)r)(s) = r[x := s] : B[x := s]}$$

$$(\to\text{-}\eta)\quad \frac{\begin{array}{c} A : \mathbf{type} \\ x : A \Rightarrow B : \mathbf{type} \\ r : (x : A) \to B \end{array}}{r = (x)(r(x)) : (x : A) \to B}$$

Abbreviations:

- $(x_1 : A_1, \ldots, x_n : A_n) \to B :=$
  $(x_1 : A_1) \to ((x_2 : A_2) \to \cdots \to ((x_n : A_n) \to B) \cdots).$
- $(x_1 : A_1, \ldots, x_{i-1} : A_{i-1}, A_i, x_{i+1} : A_{i+1}, \ldots, x_n : A_n) \to B :=$
  $(x_1 : A_1, \ldots, x_{i-1} : A_{i-1}, x_i : A_i, x_{i+1} : A_{i+1}, \ldots, x_n : A_n) \to B$
  for a fresh variable x_i,
- $A \to B := (A) \to B.$
- $(x_1, \ldots, x_n)r := (x_1)((x_2) \cdots ((x_n)r) \cdots).$

The dependent product. It is useful to have dependent products as part of the logical framework as well, but it will not be needed for what follows. We will give their rules below in Subsection 3.4.

3.3 The Sets in Martin-Löf's Type Theory

The sets of Martin-Löf's type theory can be defined in the precense of the logical framework, except of equality rules, as axioms only. In implementations usually only the logical framework is predefined, the sets have to be defined by the user.
The constructors of the following sets will however have quite a lot of additional parameters which seem to be unnecessary. For instance

$$\text{instead of } \lambda x.t \text{ we have to write } \lambda\ A; B\ (x)t\ \ ,$$

instead of $r(s)$ we have to write $\mathbf{Ap}\ A\ B\ C\ r\ s$,

But it turns out that these additional premisses are necessary, without more statements can be proved. However, these premisses make the proof terms look quite complicated. Therefore, hiding mechanisms have been suggested, but the result still does not look very nice.
One pragmatic approach was taken in the system Half, and the result looks rather harmonic: The dependent function and product type are extended in so far as, if

$$A : \mathbf{Set}, \quad x : A \Rightarrow B : \mathbf{Set} \ ,$$

then

$$(x : A) \rightarrow B, (x : A) \times B : \mathbf{Set} \ .$$

Therefore in the most common constructors

$$\lambda, \quad \mathbf{Ap}$$

we have only the really necessary arguments.
Further the construtors in the introduction rules don't have the additional parameters, and the elimination rules are replaced by definition by case distinction (so called pattern matching), which is more or less definition by recursion as we usually do it, and then only the parameters a function really depends on have to be spelled out.
The resulting type theory looks good (except that unfortunately full recursion is allowed and therefore the type theory is inconsistent), and this seems to be a good pragmatic approach.

The Finite Sets

Let $n \in \mathbb{N}$.

$$\begin{array}{lll}
(\mathbf{N}_n - \mathrm{F}) & \mathbf{N}_n : & \mathbf{Set} \\
(\mathbf{N}_n - \mathrm{I}_k) & \mathbf{A}^n_k : & \mathbf{N}_n \quad (k = 0, \ldots , n-1) \\
(\mathbf{N}_n - \mathrm{E}) & \mathbf{C}_n : & (X : \mathbf{N}_n \rightarrow \mathbf{Set}, \\
 & & \ u_0 : X\ \mathbf{A}^n_0, \\
 & & \ \ldots , \\
 & & \ u_{n-1} : X\ \mathbf{A}^n_{n-1}, \\
 & & \ x : \mathbf{N}_n) \\
 & & \rightarrow X\ x
\end{array}$$

$$(\mathbf{N}_n\text{-}=) \quad \frac{\begin{array}{c} B : \mathbf{N} \rightarrow \mathbf{Set} \\ s_0 : B\ \mathbf{A}^n_0 \\ \cdots \\ s_{n-1} : B\ \mathbf{A}^n_{n-1} \end{array}}{\mathbf{C}_n\ B\ s_0\ \ldots\ s_{n-1}\ \mathbf{A}^n_k = s_k : B\ \mathbf{A}^n_k}$$

The original notation was n_k for $\mathbf{A}^n_k$, but this leads often to confusion.
$\mathbf{N}_2$ are the booleans, and we can define

$$\begin{array}{l} \mathbf{true} := \mathbf{A}^2_0, \quad \mathbf{false} := \mathbf{A}^2_1 \\ \mathbf{if}_A\ a\ \mathbf{then}\ s_0\ \mathbf{else}\ s_1 := \mathbf{C}_2\ A\ s_0\ s_1\ a\ \ . \end{array}$$

$\mathbf{N}_0$ is the falsity: It has no elements, and the elimination rule is

$$\mathbf{C}_0 : (X : \mathbf{N}_0 {\rightarrow} \mathbf{Set}, x : \mathbf{N}_0){\rightarrow} X\ x$$

or ex falsum quodlibet with respect to every set X depending on $\mathbf{N}_0$. Note that there is no equality rule for $\mathbf{N}_0$.
The elimination and equality rules can be derived from the introduction rules: From an element of a set we can define an element of another set, if we have for each way, the element is constructed we have one step function which tells us what to do. In the context of inductive-recursive definitions this derivation of elimination rules from introcuction rules is made precise. Therefore we will not add η-rules for the sets: they follow not in a direct way from the introduction rules but require additional considerations.

The Set of Natural Numbers

$$\begin{array}{lll} (\mathbf{N} - \mathrm{F}) & \mathbf{N}: & \mathbf{Set} \\ (\mathbf{N} - \mathrm{I}_{\mathbf{0}}) & \mathbf{0}: & \mathbf{N} \\ (\mathbf{N} - \mathrm{I}_{\mathbf{S}}) & \mathbf{S}: & \mathbf{N}{\rightarrow}\mathbf{N} \\ (\mathbf{N} - \mathrm{E}) & \mathbf{P}: & (X : \mathbf{N}{\rightarrow}\mathbf{Set}, \\ & & \ u_{\mathbf{0}} : X\ \mathbf{0}, \\ & & \ u_{\mathbf{S}} : (x : \mathbf{N}, X\ x){\rightarrow} X\ (\mathbf{S}\ x), \\ & & \ x : \mathbf{N}){\rightarrow} X\ x \end{array}$$

$$(\mathbf{N}\text{-}{=}_{\mathbf{0}}) \quad \frac{\begin{array}{c} A : \mathbf{N}{\rightarrow}\mathbf{Set} \\ s_{\mathbf{0}} : A\ \mathbf{0} \\ s_{\mathbf{S}} : (x : \mathbf{N}, A\ x){\rightarrow} A\ (\mathbf{S}\ x) \end{array}}{\mathbf{P}\ A\ s_{\mathbf{0}}\ s_{\mathbf{S}}\ \mathbf{0} = s_{\mathbf{0}} : B\ \mathbf{0}}$$

$$(\mathbf{N}\text{-}{=}_{\mathbf{S}}) \quad \frac{\begin{array}{c} A : \mathbf{N}{\rightarrow}\mathbf{Set} \\ s_{\mathbf{0}} : A\ \mathbf{0} \\ s_{\mathbf{S}} : (x : \mathbf{N}, A\ x){\rightarrow} A\ (\mathbf{S}\ x) \\ r : \mathbf{N} \end{array}}{\mathbf{P}\ A\ s_{\mathbf{0}}\ s_{\mathbf{S}}\ (\mathbf{S} r) = s_{\mathbf{S}}\ r\ (\mathbf{P}\ A\ s_{\mathbf{0}}\ s_{\mathbf{S}}\ r) : B\ (\mathbf{S}\ r)}$$

The symbol **P** stands for **p**rimitive recursion.
If $A = (x)\mathbf{N}$ the elimination rule yields ordinary primitive recursion:
Assume

$$a : \mathbf{N}, \quad f : \mathbf{N}{\rightarrow}\mathbf{N}{\rightarrow}\mathbf{N}\ \ .$$

Then

$$g := \mathbf{P}\ ((x)\mathbf{N})\ a\ f : \mathbf{N} \rightarrow \mathbf{N}$$

and we can derive

$$\begin{aligned} &g\ \mathbf{0} = a : \mathbf{N}\ , \\ &g\ (\mathbf{S}n) = f\ n\ (g\ n) : \mathbf{N}\ , \end{aligned}$$

If A is considered as a formula, depending on a natural number, this corresponds to induction:
From proofs of

$$A\ \mathbf{0} \text{ and } x : \mathbf{N}, A\ x \Rightarrow A\ (\mathbf{S}x)\ ,$$

i.e. from

$$r : A\ \mathbf{0}, \quad s : (x : \mathbf{N}, A\ x) \rightarrow A\ (\mathbf{S}x)$$

and from

$$n : \mathbf{N}$$

we can derive

$$A\ n$$

with proof term

$$\mathbf{P}\ A\ r\ s\ n : A\ n\ .$$

The Disjoint Union of Sets

In the following we write $A + B$ instead of $+\ A\ B$.

$$\begin{array}{lll} (+-\mathrm{F}) & + : & (X_0 : \mathbf{Set}, X_1 : \mathbf{Set}) \rightarrow \mathbf{Set} \\ (+-\mathrm{I}_0) & \mathbf{i}_0 : & (X_0 : \mathbf{Set}, X_1 : \mathbf{Set}, x : X_0) \rightarrow X_0 + X_1 \\ (+-\mathrm{I}_1) & \mathbf{i}_1 : & (X_0 : \mathbf{Set}, X_1 : \mathbf{Set}, x : X_1) \rightarrow X_0 + X_1 \\ (+-\mathrm{E}) & \mathbf{D} : & (X_0 : \mathbf{Set}, \\ & & \ X_1 : \mathbf{Set}, \\ & & \ Y : (X_0 + X_1) \rightarrow \mathbf{Set}, \\ & & \ u_0 : (x : X_0) \rightarrow Y\ (\mathbf{i}_0\ x), \\ & & \ u_1 : (x : X_1) \rightarrow Y\ (\mathbf{i}_1\ x), \\ & & \ x : X_0 + X_1) \rightarrow Y\ x \end{array}$$

$$(+\text{-}=_i)\quad \frac{\begin{array}{c} A_0 : \mathbf{Set} \\ A_1 : \mathbf{Set} \\ B : (A_0 + A_1) \rightarrow \mathbf{Set} \\ s_0 : (x : A_0) \rightarrow B\ (\mathbf{i}_0\ x), \\ s_1 : (x : A_1) \rightarrow B\ (\mathbf{i}_1\ x), \\ a : A_i \end{array}}{\mathbf{D}\ A_0\ A_1\ B\ s_0\ s_1\ (\mathbf{i}_i\ a) = s_i\ a : B\ (\mathbf{i}_i\ a)}$$

Note that the rules above are exactly the same as the rules for $+$ in the simple typed λ-calculus, just with the additional type information added. For propositions A, B, $A+B$ is therefore the proposition $A \vee B$, and the above introduction and elimination rules correspond exactly to the introduction and elimination rules for $\vee$, the equality rule is the above mentioned proof transformation.

The Π-set

$$
\begin{array}{lll}
(\Pi - \mathrm{F}) & \Pi : & (X : \mathbf{Set}, Y : X \to \mathbf{Set}) \to \mathbf{Set} \\
(\Pi - \mathrm{I}) & \lambda : & (X : \mathbf{Set}, Y : X \to \mathbf{Set}, y : (x : X) \to Y\ x) \\
 & & \quad \to \Pi\ X\ Y \\
(\Pi - \mathrm{E}) & \mathbf{F} : & (X : \mathbf{Set}, \\
 & & \quad Y : X \to \mathbf{Set}, \\
 & & \quad Z : (\Pi\ X\ Y) \to \mathbf{Set}, \\
 & & \quad u : (y : (x : X) \to Y\ y) \to Z\ (\lambda\ X\ Y\ y), \\
 & & \quad x : (\Pi\ X\ Y)) \to Z\ x
\end{array}
$$

$$
(\Pi\text{-}=) \qquad \frac{\begin{array}{c} A : \mathbf{Set} \\ B : A \to \mathbf{Set} \\ C : (\Pi\ A\ B) \to \mathbf{Set} \\ s : (y : (x : A) \to B\ x) \to C\ (\lambda\ A\ B\ y), \\ f : (x : A) \to B\ x \end{array}}{\mathbf{F}\ A\ B\ C\ s\ (\lambda\ A\ B\ f) = s\ f : C\ (\lambda\ A\ B\ f)}
$$

The term

$$\lambda\ A\ B\ f$$

looks a bit too long for practical purposes, but we have indicated, how to remedy this: Use a version of $\to$ for **Set**.
The elimination rule (**F** is called "Fun-split") is non-standard, but as it stands, it is the exact counterpart of the introduction rule.
We can define from it application

$$
\begin{array}{lll}
\mathbf{Ap} & := & (X, Y, y, x,) \\
 & & \mathbf{F}\ X\ Y\ ((y)\ Y\ x) \\
 & & \quad ((z) z\ x) \\
 & & \quad y \\
 & & : (X : \mathbf{Set}, Y : X \to \mathbf{Set}, y : \Pi\ X\ Y, x : X) \to Y\ x
\end{array}
$$

and can derive, if A, B, f, a have appropriate types,

$$\mathbf{Ap}\ A\ B\ (\lambda\ A\ B\ f)\ a = f\ a : B\ a\ \ .$$

The only other application is that we can show propositional η-equality. If this is not needed, one could take appropriate rules for **Ap** as elimination rules for Π.
We can define now

$$\begin{array}{rcl} \forall x : A.B(x) & := & \Pi(A, B) \\ A{\rightarrow}_{\mathrm{prop}}B & := & \Pi(A, (y)B) \text{ for } y \text{ a new variable} \end{array}$$

where $A{\rightarrow}_{\mathrm{prop}}B$ means the proposition "A implies B". We get, if we had only the rules for **Ap** as elimination rules, the ordinary rules for $\forall$, and propositional implication $\rightarrow$. The rules for **F** cannot be expressed in natural deduction. The would be needed to be written like

$$\frac{a : \forall x : A.B(x) \quad (x : A){\rightarrow}B(x){\Rightarrow}C}{C}$$

i.e. if we have a proof of

$$\forall x.B(x)$$

and,

$$\text{whenever we have } x : A{\Rightarrow}B(x) \text{ then } C$$

then we have C. Such a rule does not make sense in ordinary natural deduction.

The Σ-set

$$\begin{array}{lll} (\Sigma-\mathrm{F}) & \Sigma : & (X : \mathbf{Set}, Y : X{\rightarrow}\mathbf{Set}){\rightarrow}\mathbf{Set} \\ (\Sigma-\mathrm{I}) & \mathbf{p} : & (X : \mathbf{Set}, Y : X{\rightarrow}\mathbf{Set}, x : X, y : Y\ x) \\ & & \quad {\rightarrow}\Sigma\ X\ Y \\ (\Sigma-\mathrm{E}) & \mathbf{E} : & (X : \mathbf{Set}, \\ & & \quad Y : X{\rightarrow}\mathbf{Set}, \\ & & \quad Z : (\Sigma\ X\ Y){\rightarrow}\mathbf{Set}, \\ & & \quad u : (x : X, y : Y\ x){\rightarrow}Z\ (\mathbf{p}\ X\ Y\ x\ y), \\ & & \quad x : (\Sigma\ X\ Y)){\rightarrow}Z\ x \end{array}$$

$$(\Sigma\text{-}=)\qquad \frac{\begin{array}{c} A : \mathbf{Set} \\ B : A{\rightarrow}\mathbf{Set} \\ C : (\Sigma\ A\ B){\rightarrow}\mathbf{Set} \\ s : (x : A, y : B\ x){\rightarrow}C\ (\mathbf{p}\ A\ B\ x\ y), \\ a : A \\ b : B \end{array}}{\begin{array}{c} \mathbf{E}\ A\ B\ C\ s\ (\mathbf{p}\ A\ B\ a\ b) = s\ a\ b \\ : C\ (\mathbf{p}\ A\ B\ a\ b) \end{array}}$$

Again, in order to reduce the complexity, one can make use of the logical framework product.
For a set A and a proposition $B(x)$ depending on $x : A$ we can define now

$$\exists x : A.B(x) := \Sigma\ A\ B$$

and get the usual rules for $\exists$.
We can define as well for propositions A, B

$$A \wedge B := \Sigma\ A\ ((x)B) \quad (x \text{ fresh}) \ .$$

The elimination rule for $\wedge$ corresponds now to a rule

$$\frac{A \wedge B \quad A, B \Rightarrow C}{C}$$

One easily verifies that from this rule one can derive the ordinary elimination rules for $\wedge$ and vice versa.

The Identity Set

$$\begin{array}{lll}
(\mathbf{I}-\mathrm{F}) & \mathbf{I} : & (X : \mathbf{Set}, x : X, y : X) \rightarrow \mathbf{Set} \\
(\mathbf{I}-\mathrm{I}) & \mathbf{r} : & (X : \mathbf{Set}, x : X) \rightarrow \mathbf{I}\ X\ x\ x \\
(\mathbf{I}-\mathrm{E}) & \mathbf{J} : & (X : \mathbf{Set}, \\
 & & \ Y : (x : X, y : X, z : \mathbf{I}\ X\ x\ y) \rightarrow \mathbf{Set}, \\
 & & \ u : (x : X) \rightarrow Y\ x\ x\ (\mathbf{r}\ X\ x), \\
 & & \ x : X, \\
 & & \ y : X, \\
 & & \ z : \mathbf{I}\ X\ x\ y) \rightarrow Y\ x\ y\ z
\end{array}$$

$$(\mathbf{I}\text{-}=) \qquad \frac{\begin{array}{c} A : \mathbf{Set} \\ B : (x : A, y : A, z : \mathbf{I}\ A\ x\ y) \rightarrow \mathbf{Set} \\ s : (x : A) \rightarrow B\ x\ x\ (\mathbf{r}\ A\ x), \\ a : A \end{array}}{\mathbf{J}\ A\ B\ s\ a\ a\ (\mathbf{r}\ A\ a) = s\ a : C\ a\ a\ (\mathbf{r}\ A\ a)}$$

For a set A $\mathbf{I}\ A\ a\ b$ is the set corresponding to the proposition

$$a = b$$

(as elements of A, sometimes written as

$$a =_A b \ )$$

The introduction rule proves reflexivity

$$a = a \ .$$

But, since from $a = b : A$ it follows

$$\mathbf{I}\ A\ a\ a = \mathbf{I}\ A\ a\ b : \mathbf{Set}\ \ ,$$

we can prove $a =_A b$ if we can prove $a = b : A$, i.e. if a, b are α, β, η-equivalent (with respect to the reductions, which correspond to our equational rules).
If we take $A = \mathbf{N}$ and restrict us to terms build from functions for primitive recursive functions, α, β, η-equivalence for these terms means that their equality can be derived from the equality rules and defining equations for primitive recursive functions.
In natural deduction, a proposition B as above cannot depend on a proof for $a =_A b$, therefore the elimination rules correspond to:
If $B(x, y)$ is a proposition, depending on $x, y : A$ then we have the rule

$$\frac{a =_A b \quad B(x, x)}{B(a, b)}$$

One sees easily from this rule we can derive symmetry, transitivity and congruence and vice versa.
The nice thing is that $\mathbf{J}$ can be derived directly. However there is another rule, which was shown by Martin Hofmann in his thesis to be independent of $\mathbf{J}$:

$$
\begin{array}{lll}
(\mathbf{I} - \mathrm{E}_2) & \mathbf{K} : & (X : \mathbf{Set}, \\
 & & \ Y : (x : X, y : \mathbf{I}\ X\ x\ x) \rightarrow \mathbf{Set}, \\
 & & \ u : (x : X) \rightarrow Y\ x\ (\mathbf{r}\ X\ x), \\
 & & \ x : X, \\
 & & \ z : \mathbf{I}\ X\ x\ x) \rightarrow Y\ x\ z
\end{array}
$$

$$
(\mathbf{I}\text{-}{=}_2) \qquad \frac{\begin{array}{c} A : \mathbf{Set} \\ B : (x : A, y : \mathbf{I}\ A\ x\ x) \rightarrow \mathbf{Set} \\ s : (x : A) \rightarrow B\ x\ (\mathbf{r}\ A\ x), \\ a : A \end{array}}{\mathbf{K}\ A\ B\ s\ a\ (\mathbf{r}\ A\ a) = s\ a : C\ a\ (\mathbf{r}\ A\ a)}
$$

We have

$$
\begin{array}{l}
(X, x) \\
\qquad (\mathbf{K}\ X \\
\qquad ((y, v)\mathbf{I}\ (\mathbf{I}\ X\ y\ y)\ v\ (\mathbf{r}\ X\ y)) \\
\qquad ((y)\mathbf{r}\ (\mathbf{I}\ X\ y\ y)\ (\mathbf{r}\ X\ x))) \\
: (X : \mathbf{Set}, \\
\ \ x : X, \\
\ \ v : \mathbf{I}\ X\ x\ x) \\
\rightarrow \quad \mathbf{I}(\mathbf{I}\ X\ x\ x)\ v\ (\mathbf{r}\ X\ x)\ \ ,
\end{array}
$$

i.e. with $\mathbf{K}$ we can prove (as a propositional equality) that there is only one equality proof of $x =_X x$, namely $\mathbf{r}\ X\ x$. This is not provable with $\mathbf{J}$.

We can make now precise that with $\mathbf{F}$ we can prove propositional η-equality for the Π-set:
We have: If

- $X : \mathbf{Set}$,
- $Y : X \rightarrow \mathbf{Set}$,
- $x : (x : X) \rightarrow Y\ x$,
- $x' := \lambda\ X\ Y\ x$

then we can prove (w.r.t. to judgemental equality

$$\lambda\ X\ Y\ ((y)\mathbf{Ap}\ X\ Y\ x'\ y) = \lambda\ X\ Y\ ((y)x(y)) = \lambda\ X\ Y\ x \equiv x'$$

i.e.

$$\lambda\ X\ Y\ ((y)\mathbf{Ap}\ X\ Y\ x'\ y) = x' : \Pi\ X\ Y\ \ .$$

Therefore we can show

$$\begin{array}{l}
(X, Y) \\
\ \mathbf{F}\ X\ Y \\
\ \ ((x)\mathbf{I}\ (\Pi\ X\ Y)\ x\ (\lambda\ X\ Y\ ((y)\mathbf{Ap}\ X\ Y\ x\ y))) \\
\ \ ((x)\mathbf{r}\ (\Pi\ X\ Y)\ (\lambda\ X\ Y\ x)) \\
: (X : \mathbf{Set}, Y : X \rightarrow \mathbf{Set}, x : \Pi\ X\ Y) \rightarrow \mathbf{I}\ (\Pi\ X\ Y)\ x\ (\lambda\ X\ Y\ ((y)\mathbf{Ap}\ X\ Y\ x\ y))
\end{array}$$

Therefore, also x and $\lambda\ X\ Y\ ((y)\mathbf{Ap}\ X\ Y\ x\ y)$ are not equal w.r.t judgemental equality, their equality can be shown.

The W-set

$$\begin{array}{lll}
(\mathbf{W} - \mathrm{F}) & \mathbf{W} : & (X : \mathbf{Set}, Y : X \rightarrow \mathbf{Set}) \rightarrow \mathbf{Set} \\
(\mathbf{W} - \mathrm{I}) & \mathbf{sup} : & (X : \mathbf{Set}, Y : X \rightarrow \mathbf{Set}, \\
 & & \ x : X, y : (Y\ x) \rightarrow \mathbf{W}\ X\ Y) \\
 & & \ \rightarrow \mathbf{W}\ X\ Y \\
(\mathbf{W} - \mathrm{E}) & \mathbf{R} : & (X : \mathbf{Set}, \\
 & & \ Y : X \rightarrow \mathbf{Set}, \\
 & & \ Z : (\mathbf{W}\ X\ Y) \rightarrow \mathbf{Set}, \\
 & & \ u : (x : X, \\
 & & \ \ y : (Y\ x) \rightarrow \mathbf{W}\ X\ Y, \\
 & & \ \ z : (u : Y\ x) \rightarrow Z\ (y\ u)) \rightarrow Z\ (\mathbf{sup}\ X\ Y\ x\ y) \\
 & & \ x : (\mathbf{W}\ X\ Y)) \rightarrow Z\ x
\end{array}$$

$$
(\mathbf{W}\text{-}{=})\quad
\frac{\begin{array}{l}
A : \mathbf{Set} \\
B : A \to \mathbf{Set} \\
C : (\mathbf{W}\ A\ B) \to \mathbf{Set} \\
\ \ s : \quad (x : A, \\
\qquad\quad y : (B\ x) \to \mathbf{W}\ A\ B, \\
\qquad\quad u : (u : B\ x) \to C\ (y\ x)) \to C\ (\mathbf{sup}\ A\ B\ x\ y) \\
a : A \\
b : (B\ a) \to \mathbf{W}\ A\ B
\end{array}}{\begin{array}{l}
\mathbf{R}\ A\ B\ C\ s\ (\mathbf{sup}\ A\ B\ a\ b) = \ s\ a\ b\ ((x)\mathbf{R}\ A\ B\ C\ s\ (b\ x)) \\
\quad : C\ (\mathbf{sup}\ A\ B\ a\ b)
\end{array}}
$$

The elements of $\mathbf{W}\ A\ B$ are well-founded trees with branching degrees $B\ x$ for $x : A$.
The introduction rule constructs from an element

$$a : A$$

and an $B\ a$ indexed collection of trees

$$b : (B\ a) \to \mathbf{W}\ A\ B$$

a new tree

$$\mathbf{sup}\ a\ b$$

with label a and subtrees $b\ x$ for $x : A$.
The elimination rule is induction over sees trees. The step term is crucial: If we can for every $a : A$, $b : (B\ a) \to W\ A\ B$ from the induction hypothesis, which is an element

$$c : (x : B\ a) \to C\ (b\ x)\ \ ,$$

derive an element of

$$C\ (\mathbf{sup}\ A\ B\ a\ b)$$

then we can derive

$$(x : \mathbf{W}\ A\ B)) \to C\ x\ \ .$$

With $\mathbf{W}$ we can simulate inductive definitions, but this is only interesting in order to explore the strength of the closed theory, we are defining in this section. When working in an implemented system, instead of simulating inductive definitions one defines them directly by following essentially the pattern according to which $\mathbf{W}\ A\ B$ is defined.

The Universe

A universe is a collection of sets. More precisely it is a collection of codes for sets, and simultaneously with the universe $\mathbf{U}$ we define its decoding function

$$\mathbf{T} : \mathbf{U} \to \mathbf{Set}\ \ ,$$

which assigns to a code the set it denotes.
We define here a universe closed under all set constructions given before:

$$
\begin{array}{lll}
(\mathbf{U}-\mathrm{F}) & \mathbf{U}: & \mathbf{Set} \\
(\mathbf{T}-\mathrm{F}) & \mathbf{T}: & \mathbf{U}\to\mathbf{Set} \\
(\mathbf{U}-\mathrm{I}_{\widehat{\mathbf{N}}_n}) & \widehat{\mathbf{N}}_n: & \mathbf{U} \quad (n\in\mathbb{N}) \\
(\mathbf{U}-\mathrm{I}_{\widehat{\mathbf{N}}}) & \widehat{\mathbf{N}}: & \mathbf{U} \\
(\mathbf{U}-\mathrm{I}_{\widehat{\Sigma}}) & \widehat{\Sigma}: & (x:\mathbf{U}, y:(\mathbf{T}x)\to\mathbf{U})\to\mathbf{U} \\
(\mathbf{U}-\mathrm{I}_{\widehat{\Pi}}) & \widehat{\Pi}: & (x:\mathbf{U}, y:(\mathbf{T}x)\to\mathbf{U})\to\mathbf{U} \\
(\mathbf{U}-\mathrm{I}_{\widehat{\mathbf{W}}}) & \widehat{\mathbf{W}}: & (x:\mathbf{U}, y:(\mathbf{T}x)\to\mathbf{U})\to\mathbf{U} \\
(\mathbf{U}-\mathrm{I}_{\widehat{+}}) & \widehat{+}: & (x:\mathbf{U}, y:\mathbf{U})\to\mathbf{U} \\
(\mathbf{U}-\mathrm{I}_{\widehat{\mathbf{I}}}) & \widehat{\mathbf{I}}: & (x:\mathbf{U}, y:\mathbf{T}\ x, y:\mathbf{T}\ x)\to\mathbf{U}
\end{array}
$$

$$a\widehat{+}b := \widehat{+}\ a\ b\ .$$

$(\mathbf{T}\text{-}=_{\widehat{\mathbf{N}}_n})\quad \mathbf{T}\ \widehat{\mathbf{N}}_n = \mathbf{N}_n : \mathbf{Set} \qquad (\mathbf{T}\text{-}=_{\widehat{\mathbf{N}}})\quad \mathbf{T}\ \widehat{\mathbf{N}} = \mathbf{N} : \mathbf{Set}$

$$(\mathbf{T}\text{-}=_{\widehat{\Sigma}})\quad \frac{\begin{array}{c} a:\mathbf{U} \\ b:(\mathbf{T}\ a)\to\mathbf{U} \end{array}}{\mathbf{T}(\widehat{\Sigma}\ a\ b) = \Sigma\ (\mathbf{T}\ a)\ ((x)\mathbf{T}\ (b\ x)) : \mathbf{Set}}$$

$$(\mathbf{T}\text{-}=_{\widehat{\Pi}})\quad \frac{\begin{array}{c} a:\mathbf{U} \\ b:(\mathbf{T}\ a)\to\mathbf{U} \end{array}}{\mathbf{T}(\widehat{\Pi}\ a\ b) = \Pi\ (\mathbf{T}\ a)\ ((x)\mathbf{T}\ (b\ x)) : \mathbf{Set}}$$

$$(\mathbf{T}\text{-}=_{\widehat{\mathbf{W}}})\quad \frac{\begin{array}{c} a:\mathbf{U} \\ b:(\mathbf{T}\ a)\to\mathbf{U} \end{array}}{\mathbf{T}(\widehat{\mathbf{W}}\ a\ b) = \mathbf{W}\ (\mathbf{T}\ a)\ ((x)\mathbf{T}\ (b\ x)) : \mathbf{Set}}$$

$$(\mathbf{T}\text{-}=_{\widehat{+}})\quad \frac{\begin{array}{c} a:\mathbf{U} \\ b:\mathbf{U} \end{array}}{\mathbf{T}(a\widehat{+}b) = (\mathbf{T}\ a) + (\mathbf{T}\ b) : \mathbf{Set}}$$

$$(\mathbf{T}\text{-}=_{\widehat{\mathbf{I}}})\quad \frac{\begin{array}{c} a:\mathbf{U} \\ b:\mathbf{T}\ a \\ c:\mathbf{T}\ a \end{array}}{\mathbf{T}(\widehat{\mathbf{I}}\ a\ b\ c) = \mathbf{I}\ (\mathbf{T}\ a)\ b\ c : \mathbf{Set}}$$

An elimination rule as before can not be given for the universe, since we have infinitely many introduction rules (because for every $n\in\mathbb{N}$ we have one introduction rule for

$$\mathbf{N}_n : \mathbf{U}\ .$$

However, one sees that we only need $\mathbf{N}_0$ and $\mathbf{N}_1$. (For $n > 1$ can we define $\mathbf{N}_n$ as

$$\underbrace{\mathbf{N}_1 + \cdots + \mathbf{N}_1}_{n\text{times}} \ .)$$

Then one can define an elimination rule in the spirit of what was before. Applications of the elimination rule are however very limited, so we we do not spell out this rule here. According Martin-Löf, the real elimination rule for the $\mathbf{U}$ is $\mathbf{T} : \mathbf{U} \rightarrow \mathbf{Set}$.
If one wants to prove $\neg(0 = 1)$ i.e. give a term r s.t.

$$r : (\mathbf{I}\ \mathbf{N}\ \mathbf{0}\ (\mathbf{S}\ \mathbf{0})) \rightarrow_{\text{prop}} \mathbf{N}_0$$

one needs at least a microscopic universe, i.e. a universe which has one empty and one nonempty type.

3.4 The Dependent Product of Types

The Binary Product

The formation rules of $\times$:

$$(\times\text{-F}) \qquad \frac{\begin{array}{c} A : \mathbf{type} \\ x : A \Rightarrow B : \mathbf{type} \end{array}}{(x : A) \times B : \mathbf{type}}$$

$$(\times\text{-F}_=) \qquad \frac{\begin{array}{c} A = A' : \mathbf{type} \\ x : A \Rightarrow B = B' : \mathbf{type} \end{array}}{(x : A) \times B = (x : A') \times B' : \mathbf{type}}$$

The introduction rules of $\times$:

$$(\times\text{-I}) \qquad \frac{\begin{array}{c} A : \mathbf{type} \\ x : A \Rightarrow B : \mathbf{type} \\ r : A \\ s : B[x := r] \end{array}}{\langle r, s \rangle : (x : A) \times B} \qquad (\times\text{-I}_=) \qquad \frac{\begin{array}{c} A : \mathbf{type} \\ x : A \Rightarrow B : \mathbf{type} \\ r = r' : A \\ s = s' : B[x := r] \end{array}}{\langle r, s \rangle = \langle r', s' \rangle : (x : A) \times B}$$

The elimination rules of $\times$:

$$(\times\text{-E}_0) \qquad \frac{\begin{array}{c} A : \mathbf{type} \\ x : A \Rightarrow B : \mathbf{type} \\ r : (x : A) \times B \end{array}}{r\ 0 : A} \qquad (\times\text{-E}_{0,=}) \qquad \frac{\begin{array}{c} A : \mathbf{type} \\ x : A \Rightarrow B : \mathbf{type} \\ r = r' : (x : A) \times B \end{array}}{r\ 0 = r'\ 0 : A}$$

$$(\times\text{-E}_1)\quad \frac{\begin{array}{c}A : \mathbf{type}\\ x : A \Rightarrow B : \mathbf{type}\\ r : (x : A) \times B\end{array}}{r\ 1 : B[x := r\ 0]} \qquad (\times\text{-E}_{1,=})\quad \frac{\begin{array}{c}A : \mathbf{type}\\ x : A \Rightarrow B : \mathbf{type}\\ r = r' : (x : A) \times B\end{array}}{r'\ 1 = r'\ 1 : B[x := r\ 0]}$$

The equality rules for $\times$:

$$(\times\text{-}=_0)\quad \frac{\begin{array}{c}A : \mathbf{type}\\ x : A \Rightarrow B : \mathbf{type}\\ r : A\\ s : B[x := r]\end{array}}{\langle r, s\rangle\ 0 = r : A} \qquad (\times\text{-}=_1)\quad \frac{\begin{array}{c}A : \mathbf{type}\\ x : A \Rightarrow B : \mathbf{type}\\ r : A\\ s : B[x := r]\end{array}}{\langle r, s\rangle\ 1 = s : B[x := r\ 0]}$$

$$(\times\text{-}\eta)\quad \frac{\begin{array}{c}A : \mathbf{type}\\ x : A \Rightarrow B : \mathbf{type}\\ r : (x : A) \times B\end{array}}{r = \langle r\ 0, r\ 1\rangle : (x : A) \times B}$$

Abbreviations:

- $(x_1 : A_1, \ldots, x_n : A_n) \times B :=$
  $(x_1 : A_1) \times ((x_2 : A_2) \times \cdots \times ((x_n : A_n) \times B) \cdots)$.
- $(x_1 : A_1, \ldots, x_{i-1} : A_{i-1}, A_i, x_{i+1} : A_{i+1}, \ldots, x_n : A_n) \times B :=$
  $(x_1 : A_1, \ldots, x_{i-1} : A_{i-1}, x_i : A_i, x_{i+1} : A_{i+1}, \ldots, x_n : A_n) \times B$
  for a fresh variable x_i,
- $A \times B := (A) \times B$.
- $\langle r_1, \ldots, r_n\rangle := \langle r_1, \langle r_2, \ldots, \langle r_{n-1}, r_n\rangle\rangle\rangle$.
- If r is introduced as an element of

  $$(x_1 : A_1, \ldots, x_n : A_n) \times B\ \ ,$$

  one sometimes uses x_i as labels and writes

  $$r_{x_i}$$

  for

  $$\begin{cases} r \quad \underbrace{0 \cdots 0}_{n-i+1 \text{ times}} \quad 1 & n > 1 \\ r \quad \underbrace{0 \cdots 0}_{n \text{ times}} & n = 1 \end{cases}\ .$$

  One can write the above type as

  $$(x_1 : A_1, \ldots, x_n : A_n) \times (x : B)$$

and writes then

$$r_x \text{ for } r\,1\ .$$

The problem is that one can then of course no longer replace x_i by different variabels.

The Empty Product

The formation and introduction rule of $\mathbf{1}$:

($\mathbf{1}$-F) $\quad \mathbf{1} : \mathbf{type} \qquad$ ($\mathbf{1}$-I) $\quad \langle\rangle : \mathbf{1}$

The η-equality for $\mathbf{1}$:

($\mathbf{1}$-η) $\quad \dfrac{r : \mathbf{1}}{r = \langle\rangle : \mathbf{1}}$

3.5 The Language of Martin-Löf's Type Theory

Definition 3.5.1 (a) We assume infinitely many variables be given. In the following x, y, z, u, v, X, Y, Z, possibly with subscripts and accents, are variables.

(b) The *type constructors* of Martin-Löf's type theory are **Set**, $\mathbf{1}$.

(c) The *set constructors* of Martin-Löf's type theory are:

- $\mathbf{N}$, $+$, Π, Σ, $\mathbf{I}$, $\mathbf{W}$,
- $\mathbf{U}$, $\mathbf{T}$,
- for $n \in \mathbb{N}$ $\mathbf{N}_n$.

(d) The *term constructors* of Martin-Löf's type theory are

- $\mathbf{0}$, $\mathbf{S}$, $\mathbf{P}$,
- $\mathbf{i}_0$, $\mathbf{i}_1$, $\mathbf{D}$,
- λ, $\mathbf{F}$,
- $\mathbf{p}$, $\mathbf{E}$,
- $\mathbf{r}$, $\mathbf{J}$, $\mathbf{K}$,
- $\mathbf{sup}$, $\mathbf{R}$,
- $\widehat{\mathbf{N}}$, $\widehat{+}$, $\widehat{\Pi}$, $\widehat{\Sigma}$, $\widehat{\mathbf{I}}$, $\widehat{\mathbf{W}}$,
- $\langle\rangle$,
- and for $n \in \mathbb{N}$ $\mathbf{N}_n$, $\mathbf{C}_n$, and

- for $k < n$ $\mathbf{A}^n_k$.

(e) The *pre-terms* are:

- term constructors,
- set constructors and,
- if r, s are pre-terms,
  - $(r\ s)$,
  - $((x)r)$,
  - $\langle r, s\rangle$,
  - $r\ 0$,
  - $r\ 1$.

In the following $a, b, c, n, f, g, r, s, t$, possibly with subscripts and accents,will be pre-terms.

(f) The *pre-sets* are:

- set constructors and,
- if p, q are pre-sets,
  - $(p\ r)$,
  - $(p\ q)$,
  - $((x)p)$,
  - $\langle p, q\rangle$,
  - $p\ 0$,
  - $p\ 1$.

(g) The *pre-types* are:

- type constructors,
- pre-sets, and,
- if A, B are pre-types,
  - $(x : A)\rightarrow B$,
  - $(x : A) \times B$.

In the following A, B, C, with accents or subscripts, denote pre-types.

(h) *Pre-nondependent judgements* are

- $A = B : \mathbf{type}$,
- $A : \mathbf{type}$,
- $r : A$,
- $r = s : A$.

In the following θ, possibly with accents or indices, denote pre-nondependent judgements.

(i) Pre-contexts are

$$x_1 : A_1, \ldots , x_n : A_n$$

where $n \in \mathbb{N}$ (including 0, this pre-context is denoted by $\emptyset$).

In the following Γ, Δ, possibly with accents or indices, denote pre-contexts.

If Γ, Δ are pre-contexts, Γ, Δ is as usual the concatenation of the pre-contexts.

(j) Pre-judgements are $\Gamma \Rightarrow \theta$.

(k) We omit parentheses as usual (where $r\ s\ t := (r\ s)\ t$).

Definition 3.5.2 (a) For pre-terms and -types d we define the set of free variables $\mathrm{FV}(d)$ by

- If $d \in \{\mathbf{Set}, \mathbf{1}\}$ or d is a set- or term-constructor, then $\mathrm{FV}(d) := \emptyset$.
- $\mathrm{FV}(d\ e) := \mathrm{FV}(\langle d, e\rangle) := \mathrm{FV}(d) \cup \mathrm{FV}(e)$.
- $\mathrm{FV}(d\ i) := \mathrm{FV}(d)\ (i = 0, 1)$.
- $\mathrm{FV}((z)d) := \mathrm{FV}(d) \setminus \{z\}$.
- $\mathrm{FV}((x : A) \circ B) := \mathrm{FV}(A) \cup (\mathrm{FV}(B) \setminus \{x\})$. $(\circ \in \{\to, \times\})$.

(b) For pre-terms and pre-types d, f we define the substitution of a variable x by f in d by induction on the d:

- If $d \in \{\mathbf{Set}, \mathbf{1}\}$ or d is a set- or term-constructor, then $d[x := f] := d$.
- $(d\ e)[x := f] := (d[x := f]\ e[x := f])$.
- $\langle d, e\rangle[x := f] := \langle d[x := f], e[x := f]\rangle$,
- $(d\ i)[x := f] := (d[x := f]\ i)\ (i = 0, 1)$.
-

$$((z)d)[x := f] := \begin{cases} (z)d & \text{if } x \equiv z \\ (z)(d[x := f]) & \text{if } x \not\equiv z, (x \notin \mathrm{FV}(d) \vee y \notin \mathrm{FV}(f)) \\ (u)(d[z := u][x := f]) & \text{otherwise, } u \text{ minimal s.t.} \\ & u \notin \mathrm{FV}(f\ d\ x)\ . \end{cases}$$

- If $((z)e)[x := f] = (z')e'$, then for $\circ \in \{\to, \times\}$,

$$((z : d) \circ e)[x := f] = (z : (d[x := f])) \circ e'\ .$$

(c) We define the α-equivalence on pre-terms and pre-types as least transitive and reflexive relations on pre-terms and pre-types s.t.

- $(z)d$, $(y)d[z := y]$ are α-equivalent, if $y \notin \mathrm{FV}(d)$.

- $(z : d) \circ e$, $(y : d) \circ (e[z := y])$ are α-equivalent, if $y \notin \mathrm{FV}(e)$.
- If d, d' are α-equivalent, e, e' are α-equivalent, so are
  - $(d\ e)$ and $(d'\ e')$,
  - $\langle d, e\rangle$ and $\langle d', e'\rangle$,
  - $d\ i$ and $d'\ i$,
  - $(z)d$ and $(z)d'$,
  - $(z : d) \circ e$ and $(z : d') \circ e'$, $\circ \in \{\to, \times\}$.

(d)
- $\mathrm{FV}(A = B : \mathbf{type}) := \mathrm{FV}(A) \cup \mathrm{FV}(B)$,
- $\mathrm{FV}(A : \mathbf{type}) := \mathrm{FV}(A)$,
- $\mathrm{FV}(r = s : A) := \mathrm{FV}(r) \cup \mathrm{FV}(s) \cup \mathrm{FV}(A)$,
- $\mathrm{FV}(r : A) := \mathrm{FV}(r) \cup \mathrm{FV}(A)$.

(e) α-equivalence of pre-nondependent judgements is the least reflexive relation s.t. if A, A', r, r', s, s' are α-equivalent, respectively, so are

- $A = B : \mathbf{type}$ and $A' = B' : \mathbf{type}$,
- $A : \mathbf{type}$ and $A' : \mathbf{type}$,
- $r : A$ and $r' : A'$,
- $r = s : A$ and $r' = s' : A'$.

(f) For pre-nondependent judgements θ and pre-terms and pre-types f we define $\theta[x := f]$ by:

- $(A = B : \mathbf{type})[x := f] := A[x := f] = B[x := f] : \mathbf{type}$,
- $(A : \mathbf{type})[x := f] := (A[x := f]) : \mathbf{type}$,
- $(r : A)[x := f] := r[x := f] : A[x := f]$,
- $(r = s : A)[x := f] := r[x := f] = s[x := f] : A[x := f]$.

(g) For pre-judgements $\Gamma \Rightarrow \theta$ we define $\mathrm{FV}(\Gamma \Rightarrow \theta)$ by:

- $\mathrm{FV}(\emptyset \Rightarrow \theta) := \mathrm{FV}(\theta)$.
- $\mathrm{FV}(x : A, \Gamma \Rightarrow \theta) := \mathrm{FV}(A) \cup (\mathrm{FV}(\Gamma \Rightarrow \theta) \setminus \{x\})$.

(h) For pre-judgements $\Gamma \Rightarrow \theta$ and pre-terms and -types f we define

$$(\Gamma \Rightarrow \theta)[x := f]$$

by:

(i) $(\emptyset \Rightarrow \theta)[x := f] := \emptyset \Rightarrow (\theta[x := f])$.

(j) $(x : A, \Gamma \Rightarrow \theta)[x := f] := x : (A[x := f]), \Gamma \Rightarrow \theta$.

(k) If $x \not\equiv y$, $y \notin \mathrm{FV}(\Gamma \Rightarrow \theta) \vee x \notin \mathrm{FV}(f)$, then

$$(x : A, \Gamma \Rightarrow \theta)[y := f] := x : (A[y := f]), ((\Gamma \Rightarrow \theta)[y := f]) \ .$$

(l) If $x \not\equiv y$, $y \in \mathrm{FV}(\Gamma \Rightarrow \theta)$, $x \in \mathrm{FV}(f)$, then let u minimal s.t. $u \notin \mathrm{FV}(\Gamma \Rightarrow \theta)$, $u \notin \mathrm{FV}(d\, x)$.

$$(x : A, \Gamma \Rightarrow \theta)[y := f] := u : (A[y := f]), ((\Gamma \Rightarrow \theta)[x := u][y := f]) \ .$$

(m) α-equivalence of pre-judgements is the least transitive and reflexive relation on pre-judgements s.t.

- If θ, θ are α-equivalent, so are $\emptyset \Rightarrow \theta$ and $\emptyset \Rightarrow \theta'$.
- If A, A' are α-equivalent and as well $\Gamma \Rightarrow \theta$, $\Gamma' \Rightarrow \theta'$, so are $x : A, \Gamma \Rightarrow \theta$, $x : A', \Gamma' \Rightarrow \theta'$.
- If $y \notin \mathrm{FV}(\Gamma \Rightarrow \theta)$, then $x : A, \Gamma \Rightarrow \theta$ and $y : A, ((\Gamma \Rightarrow \theta)[x := y])$ are α-equivalent.

Bibliography

[CF58] H. B. Curry and R. Feys. *Combinatory Logic, Vol. I.* North-Holland, Amsterdam, 1958.

[HS86] J. R. Hindley and J. P. Seldin. *Introduction to Combinators and λ-calculus.* Cambridge University Press, Cambridge, 1986.

[MJ98] R. Matthes and F. Joachimski. Short proofs of normalization for the simply typed λ-calculus, permutative conversions and Gödel's T. Draft, available via `http://www.tcs.uni-muenchen.de/ matthes/papers/snnew.ps.gz`, 1998.

[Plo74] G. D. Plotkin. The λ-calculus is ω-complete. *J. Symb. Logic*, 39:313 – 317, 1974.

[Set94] A. Setzer. Lösen rekursiver Bereichsgleichungen. Seminar notes, based on some notes of Thomas Streicher, 1994.

[Str94] T. Streicher. Lösen rekursiver Bereichsgleichungen. Handwritten notes, 1994.

[Tak95] M. Takahashi. Parallel reductions in λ-calculus. *Information and Computation*, 118:120 – 127, 1995.

[TD88a] A. Troelstra and D. van Dalen. *Constructivism in Mathematics. An Introduction, Vol. I.* North-Holland, Amsterdam, 1988.

[TD88b] A. Troelstra and D. van Dalen. *Constructivism in Mathematics. An Introduction, Vol. II.* North-Holland, Amsterdam, 1988.