Polymorphic Extensions of Simple Type Structures

with an application to a bar recursive minimization

Erik Barendsen
Computing Science Institute
Nijmegen University

Marc Bezem
Department of Philosophy
Utrecht University

Abstract

The technical contribution of this paper is threefold.
First we show how to encode functionals in a 'flat' applicative structure by adding oracles to untyped λ-calculus and mimicking the applicative behaviour of the functional with an impredicatively defined reduction relation. The main achievement here is a Church-Rosser result for the extended reduction relation.
Second, by combining the previous result with the model construction based on partial equivalence relations, we show how to extend a λ-closed simple type structure to a model of the polymorphic λ-calculus.
Third, we specialize the previous result to a counter model against a simple minimization. This minimization is realized by a bar recursive *functional*, which contrasts the results of Spector and Girard which imply that the bar recursive *functions* are exactly those that are definable in the polymorphic λ-calculus. As a spin-off, we obtain directly the non-conservativity of the extensions of Gödel's **T** with bar recursion, fan functional, and Luckhardt's minimization functional, respectively. For the latter two extensions these results are new.

MR Classification (1991): 03B40 (03B15, 03F10).
CR Classification (1991): F.4.1.
Keywords and phrases: lambda calculus, types, polymorphism, recursion, partial equivalence relations.

Contents

Introduction

Gödel (1958) proved the consistency of arithmetic by means of a functional interpretation, called *Dialectica interpretation* after the journal in which the results first appeared. The interpretation involved a system of primitive recursive functionals denoted by $\boldsymbol{\lambda}\mathbf{T}$.

Spector (1962) extended Gödel's functional interpretation to analysis. This involved a new concept in higher order subrecursion theory called *bar recursion*. Bar recursion is essentially a principle of definition through transfinite recursion over a well-founded tree of finite sequences of functionals. Spector's system of bar recursive functionals will be denoted by $\boldsymbol{\lambda}\mathbf{TB}$ in this paper. Spector's results yielded, apart from a consistency proof for analysis, a characterization of the provably total recursive functions (resp. the provable well-orderings) of analysis as those functions (resp. well-orderings) that are definable by bar recursion.

In his thesis, Girard (1972) extended Gödel's results cited above to second- and higher-order intuitionistic arithmetic. This involved a completely new system of typed lambda calculi, namely second- and higher-order (typed) lambda calculus. Girard's system **F**, the second-order or *polymorphic* lambda calculus, will be denoted by $\boldsymbol{\lambda}\mathbf{2T}$ here and below. Apart from consistency proofs for second- and higher-order arithmetic, Girard's results yielded characterizations of the provably total recursive functions of these theories in terms of definability in the corresponding typed lambda calculi.

The metamathematical results from Spector (1962) and Girard (1972) imply that the class of bar recursive functions coincides with the class of functions definable in second order typed lambda calculus. This phenomenon suggests a, possibly deep, relationship between bar recursion and second order lambda calculus, and calls for an explanation. As a special instance of this general problem one could consider the question of definability of *functionals* of higher type (as opposed to *functions*) in both systems. The research reported here has been sparked off by this problem.

The first difficulty that arises concerns the concept of definability itself. E.g., the domain of the above functions is the set of natural numbers, but for functionals it is not clear which (higher-type) objects should be regarded as inputs. For a comparison of $\boldsymbol{\lambda}\mathbf{2T}$ and $\boldsymbol{\lambda}\mathbf{TB}$ we introduce a concept of *specification* of functionals in the language of $\boldsymbol{\lambda}\mathbf{T}$, with a notion of realizability by terms of $\boldsymbol{\lambda}\mathbf{2T}$ and $\boldsymbol{\lambda}\mathbf{TB}$ respectively. In the case of functions (i.e., functionals of type $\mathbf{1}$) this corresponds exactly to standard definability.

We present a specification $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}(\Phi)$, with Φ a type $\mathbf{3}$ variable. The computational interpretation of $\boldsymbol{\Sigma}(\Phi)$ is that Φ is a certain minimization functional. It will be shown that $\boldsymbol{\Sigma}$ has a realization in $\boldsymbol{\lambda}\mathbf{TB}$, but is not realizable in $\boldsymbol{\lambda}\mathbf{2T}$. This shows that (with our notion of realizability) $\boldsymbol{\lambda}\mathbf{TB}$ and $\boldsymbol{\lambda}\mathbf{2T}$ have different classes of functionals (at least of type $\mathbf{3}$). The case of type $\mathbf{2}$ is open. It is also open whether there exists a type $\mathbf{3}$

functional which is definable in $\boldsymbol{\lambda 2T}$ but not in $\boldsymbol{\lambda TB}$, but we consider this unlikely. There is evidence that $\boldsymbol{\lambda TB}$ is in some sense stronger than $\boldsymbol{\lambda 2T}$: this latter system interprets a purely intuitionistic theory, whereas the former also interprets a weak form of the Law of the Excluded Middle (Double Negation Shift, or axiom F from Spector (1962)).

The positive result for $\boldsymbol{\lambda TB}$ is proved by a purely syntactical argument: it turns out that there is a bar recursive term that provably satisfies the specification $\boldsymbol{\Sigma}$.

The negative proof for $\boldsymbol{\lambda 2T}$ is rather involved. It uses a *counter model* containing a specific discontinuous functional. In the standard literature, however, models of second-order lambda calculus are usually based on some continuity principle, e.g. coherence spaces (see Girard et al. (1989)) or complete partial orderings (see Poll (1994)). Also Girard's model HEO2 (see Troelstra (1973)) exhibits an inherent continuity expressed by the Kreisel-Lacombe-Schoenfield Theorem.

We apply a general model construction (inspired by HEO2) for second order systems based on partial equivalence relations (per). This per construction is parametrized by an untyped (partial) applicative structure $\langle A, \cdot \rangle$. Each type is interpreted as a subset of A. Equality in the theory (at type τ, say) is modelled by an equivalence relation on the interpretation of type τ. The objects of type $\tau_1 {\rightarrow} \tau_2$ are those whose applicative behaviour respects the equivalence relations on τ_1 and τ_2.

For the construction of a counter model based on pers (containing the discontinuous functional in question) one is then confronted with another problem: this type-**2** functional should be encoded in a 'flat' applicative structure in such a way that it 'survives' the per construction, i.e. is present in the associated per model of $\boldsymbol{\lambda 2T}$. A method for doing this is provided by Kleene's (1962) concept of λ-definability in higher types, using an extension of untyped λ-calculus with *oracles* originating from a model of first-order typed lambda-calculus (a so-called *type structure*). We use the closed term model associated with the resulting reduction system as the basis of a per construction.

Combining these two techniques yields a construction for extending certain first-order models to second-order ones, which is interesting in itself. For our non-realizability result we apply this construction to the structure of all functionals over $\mathbb{N}$ (the so-called *full type structure* over $\mathbb{N}$). This does *not* contradict Reynolds (1984): 'In this paper, we will prove that the standard set-theoretic model of the ordinary typed lambda calculus cannot be extended to model this [polymorphic] language extension.' In fact the latter quote does not capture the main result of Reynolds (1984), which states that the type constructor $\rightarrow$ in the polymorphic lambda calculus cannot be interpreted as the set-theoretic function space constructor. Our results show that $\rightarrow$ can be interpreted in such a way that its restriction to simple types corresponds (up to isomorphism) to the function space constructor.

The proof of the positive realizability result for $\boldsymbol{\lambda TB}$ results moreover in direct non-conservativity proofs for extensions of $\boldsymbol{\lambda T}$ with bar recursion, fan functional and Luckhardt's minimization functional, respectively.

Acknowledgements. The research in this paper has been supported by the Netherlands Computer Science Research Foundation (SION) with financial support of the Netherlands Organization for Scientific Research (NWO). We thank Henk Barendregt, Marco Hollenberg and Jan Terlouw for comments on earlier versions of this paper.

Chapter 1

Lambda Calculus with Higher-type Oracles

1.1. Syntax

In this section we will extend untyped λ-calculus with higher type objects (typically functionals) from a type structure. The untyped applicative structure thus obtained will be used to construct second-order models using partial equivalence relations (see Section 2.4).

1.1.1. DEFINITION. (i) The set of *first-order types* (notation $\mathbb{T}_1$) is given by the abstract syntax

$$\mathbb{T}_1 = \mathbb{C} \mid \mathbb{T}_1 \to \mathbb{T}_1.$$

Here $\mathbb{C}$ is a set of *type constants*. In this work we take $\mathbb{C} = \{\mathbf{0}\}$.

(ii) The *height* (or *type level*) of $\sigma \in \mathbb{T}_1$ (notation $\mathrm{h}(\sigma)$) is defined inductively by

$$\begin{array}{rcl} \mathrm{h}(\mathbf{0}) & = & 0, \\ \mathrm{h}(\sigma\to\tau) & = & \max(\mathrm{h}(\sigma)+1, \mathrm{h}(\tau)). \end{array}$$

NOTATION. (i) We let $\to$ associate to the right, so $\sigma\to\tau\to\rho$ stands for $\sigma\to(\tau\to\rho)$.

(ii) Note that each first order type is of the form

$$\sigma_1\to\sigma_2\to\cdots\to\sigma_n\to\mathbf{0}.$$

Such an expression will usually be abbreviated by $\vec{\sigma}\to\mathbf{0}$. Note that for each i

$$\mathrm{h}(\sigma_i) < \mathrm{h}(\vec{\sigma}\to\mathbf{0}) \quad (= \max_i \mathrm{h}(\sigma_i) + 1).$$

1.1.2. DEFINITION. (i) A first-order *type structure* is a structure

$$\mathfrak{M} = \langle (\mathfrak{M}_\sigma)_{\sigma\in\mathbb{T}_1},\ (\mathrm{App}_{\sigma,\tau})_{\sigma,\tau\in\mathbb{T}_1}\rangle$$

such that

(1) each $\mathfrak{M}_\sigma$ is a non-empty set;

(2) for each $\sigma, \tau \in \mathbb{T}_1$

$$\mathrm{App}_{\sigma,\tau} : \mathfrak{M}_{\sigma\to\tau} \times \mathfrak{M}_\sigma \to \mathfrak{M}_\tau$$

is an *application function* (we write $a \cdot b$ or simply ab for $\mathrm{App}_{\sigma,\tau}(a, b)$).

(ii) $\mathfrak{M}$ is an ω-*structure* if $\mathfrak{M}_{\mathbf{0}} = \mathbb{N}$.
(iii) a is an $\mathfrak{M}$-*element* if $a \in \bigcup_{\sigma \in \mathbb{T}_1} \mathfrak{M}_\sigma$. This is usually loosely denoted as $a \in \mathfrak{M}$.
(iv) $\mathfrak{M}$ is extensional if for each $a, a' \in \mathfrak{M}_{\sigma\to\tau}$

$$\forall b \in \mathfrak{M}_\sigma \;\; [ab = a'b] \;\Rightarrow\; a = a'.$$

We will freely make use of vector notation like $\vec{b} \in \mathfrak{M}_{\vec{\sigma}}$.

The most common example of an extensional type structure is the structure of functionals over a given set.

1.1.3. Definition. Let X be a non-empty set. The *full type structure over* X (notation $\mathfrak{M}(X)$) is defined by setting

$$\mathfrak{M}(X) = \langle (X_\sigma)_{\sigma \in \mathbb{T}_1},\; (\cdot_{\sigma,\tau})_{\sigma,\tau \in \mathbb{T}_1} \rangle$$

where

$$\begin{aligned} X_{\mathbf{0}} &= X, \\ X_{\sigma\to\tau} &= {X_\tau}^{X_\sigma} \end{aligned}$$

and

$$f \cdot a = f(a).$$

Of course $\mathfrak{M}(\mathbb{N})$ is an extensional ω-type structure.

The idea of adding higher-type oracles to untyped λ-calculus originates from Kleene (1962). He used this idea to prove the equivalence between recursiveness and lambda definability in higher types on $\mathfrak{M}(\mathbb{N})$.

In the sequel, let $\mathfrak{M} = \langle (\mathfrak{M}_\sigma)_{\sigma \in \mathbb{T}_1},\; (\mathrm{App}_{\sigma,\tau})_{\sigma,\tau \in \mathbb{T}_1} \rangle$ be an extensional ω-type structure.

1.1.4. Definition. (i) For each $\mathfrak{M}$-element of higher type $a \in \mathfrak{M}_\sigma$ (with $\sigma \not\equiv \mathbf{0}$), let $\boldsymbol{a}$ be a constant associated with a. The collection of these so-called oracles is denoted by $\mathcal{O}_{\mathfrak{M}}$.

(ii) The set of $\lambda\mathfrak{M}$-*terms* (notation $\Lambda\mathfrak{M}$) is defined by the following abstract syntax.

$$\Lambda\mathfrak{M} = V \mid \mathcal{O}_{\mathfrak{M}} \mid (\Lambda\mathfrak{M}\;\Lambda\mathfrak{M}) \mid (\lambda V.\Lambda\mathfrak{M}).$$

We use the same notational conventions for $\lambda\mathfrak{M}$-terms as we do for λ-terms; see Barendregt (1984). The set of *closed* $\lambda\mathfrak{M}$-terms is denoted by $\Lambda^\circ\mathfrak{M}$.

The principal reduction relation is β-reduction. A second notion of reduction will be added to the system. We suppose the reader is familiar with the concept of reduction relations and induced conversion relations (denoted by $\to_\beta$, $\twoheadrightarrow_\beta$, and $=_\beta$ for β-reduction).

Now we consider the constants in $\Lambda\mathfrak{M}$ as higher-type oracles. Some care is needed: the result of applying a constant of type, say, $(\mathbf{0}\to\mathbf{0})\to\mathbf{0}\to\mathbf{0}$ to a type $\mathbf{0}\to\mathbf{0}$ object is *not* an object of the corresponding type $\mathbf{0}\to\mathbf{0}$. Instead, oracles of type $(\mathbf{0}\to\mathbf{0})\to\mathbf{0}\to\mathbf{0}$ only give a result if supplied with both a type $\mathbf{0}\to\mathbf{0}$ and a type $\mathbf{0}$ object. In view of the type **0** objects being the only 'observables' in calculations, and the oracles giving results of computations in one single step, this is only natural.

As the basic 'observable' objects are natural numbers, we can use some standard representations of natural numbers in untyped λ-calculus. Adding other basic types for observables (e.g., booleans) can also be modelled using the standard representations of the corresponding objects. We will restrict ourselves to the natural numbers. We use the Church numerals to represent these.

1.1.5. DEFINITION. (i) Let $F, M \in \Lambda\mathfrak{M}$. For every $n \in \mathbb{N}$, the n-th *iteration* of F on M (notation $F^n(M)$) is defined inductively by

$$\begin{array}{rcl} F^0(M) & \equiv & M, \\ F^{n+1}(M) & \equiv & F(F^n(M)). \end{array}$$

(ii) For each $n \in \mathbb{N}$, the n-th *Church numeral* is defined by

$$\ulcorner n \urcorner \equiv \lambda f x. f^n(x).$$

It is well known that the system $(\ulcorner n \urcorner)_{n \in \mathbb{N}}$ is adequate with respect to recursive functions: each total recursive function is λ-definable with respect to the Church numerals. Not every adequate numeral system is suitable for the applications in Chapter 2; e.g., the recursor (needed for the interpretation of $\boldsymbol{\lambda}\mathbf{T}$) is not definable with respect to the (adequate) unsolvable numeral system from Barendsen (1991). One can work safely with the Church numerals, however.

1.1.6. DEFINITION. For each $a \in \mathfrak{M}$ the *standard representation* of a (notation $\underline{a}$) is defined by

$$\begin{array}{rcll} \underline{a} & \equiv & \ulcorner a \urcorner & \text{if } a \in \mathfrak{M}_{\mathbf{0}}, \\ & \equiv & \boldsymbol{a} & \text{otherwise.} \end{array}$$

1.1.7. DEFINITION. Let $\rightarrow_R$ be a notion of reduction (on $\Lambda\mathfrak{M}$), $A \in \Lambda\mathfrak{M}$, and $a \in \mathfrak{M}$, say $a \in \mathfrak{M}_{\vec{\sigma} \to \mathbf{0}}$. Then A is said to *R-define* a (notation $A \rhd_R a$) if for all $\vec{b} \in \mathfrak{M}_{\vec{\sigma}}$ one has

$$A\underline{\vec{b}} =_R \underline{a\vec{b}} \quad (\equiv \ulcorner a\vec{b} \urcorner).$$

In particular, $A \rhd_R n$ iff $A =_R \ulcorner n \urcorner$. This notion is extended to sequences ($\vec{A} \rhd_R \vec{a}$) in the obvious way.

The aim is to construct a reduction relation $\rightarrow_{\mathfrak{M}}$ such that for any $a, \vec{b}$ and $\vec{B}$ (of appropriate types)

$$\boldsymbol{a}\vec{B} \rightarrow_{\mathfrak{M}} \ulcorner a\vec{b} \urcorner \qquad \text{if} \qquad \vec{B} \rhd_{\beta\mathfrak{M}} \vec{b}. \tag{$*$}$$

Here $\beta\mathfrak{M}$-reduction is defined by $\rightarrow_{\beta\mathfrak{M}} = \rightarrow_\beta \cup \rightarrow_{\mathfrak{M}}$. Note that if '$\vec{B} \rhd_{\beta\mathfrak{M}} \vec{b}$' would be replaced by '$\vec{B} \rhd_\beta \vec{b}$', the reduction $\rightarrow_{\mathfrak{M}}$ could immediately be obtained by so-called δ-reduction making some external function internal (see Barendregt (1984), Section 15.3). This corresponds to the notion of *oracle* in recursion theory. In ($*$), however, the behaviour of the oracle is specified in terms of itself, since $\boldsymbol{a}$ can occur in $\vec{B}$.

Because of this impredicativity, the question arises whether or not $\rightarrow_{\mathfrak{M}}$ can be well defined. This is indeed the case. The idea is to generate $\rightarrow_{\mathfrak{M}}$ in stages, according to a certain inductive operator.

1.1.8. DEFINITION. Let $R \subseteq \Lambda\mathfrak{M} \times \Lambda\mathfrak{M}$ be a relation. The *compatible closure* of R (notation $\overline{R}$) is defined inductively as follows.

$$\begin{array}{rcl} M\,R\,M' & \Rightarrow & M\,\overline{R}\,M'; \\ M\,\overline{R}\,M' & \Rightarrow & MN\,\overline{R}\,M'N, \\ & & NM\,\overline{R}\,NM', \\ & & \lambda x.M\,\overline{R}\,\lambda x.M'. \end{array}$$

1.1.9. EXAMPLE. $\rightarrow_\beta = \overline{\{((\lambda x.M)N, M[x := N]) \mid M, N \in \Lambda\mathfrak{M}, x \in V\}}$.

1.1.10. DEFINITION. The operator $\Gamma_{\mathfrak{M}} : \wp(\Lambda\mathfrak{M} \times \Lambda\mathfrak{M}) \rightarrow \wp(\Lambda\mathfrak{M} \times \Lambda\mathfrak{M})$ is defined by

$$\Gamma_{\mathfrak{M}}(R) = \overline{\{(\boldsymbol{a}\vec{B}, \ulcorner a\vec{b}\urcorner) \mid a \in \mathfrak{M}_{\vec{\sigma}\rightarrow\mathbf{0}}, \vec{B} \in \Lambda\mathfrak{M}, \vec{b} \in \mathfrak{M}_{\vec{\sigma}}, \vec{B} \rhd_{\beta R} \vec{b}\}}.$$

1.1.11. LEMMA. $\Gamma_{\mathfrak{M}}$ *is monotone. That is,*

$$R \subseteq R' \;\Rightarrow\; \Gamma_{\mathfrak{M}}(R) \subseteq \Gamma_{\mathfrak{M}}(R').$$

PROOF. By monotonicity of $^{-}$ and the fact that $\rhd_{\beta R}$ is monotone in R. $\square$

Now the stages in the inductive definition are defined by setting for each ordinal number ζ (cf. Hinman (1978))

$$\Gamma^{\zeta}_{\mathfrak{M}} = \Gamma_{\mathfrak{M}}(\bigcup\{\Gamma^{\vartheta}_{\mathfrak{M}} \mid \vartheta < \zeta\}). \qquad (**)$$

Using the denotation $\Gamma^{(\zeta)}_{\mathfrak{M}} = \bigcup\{\Gamma^{\vartheta}_{\mathfrak{M}} \mid \vartheta < \zeta\}$, the expression $(**)$ becomes

$$\Gamma^{\zeta}_{\mathfrak{M}} = \Gamma_{\mathfrak{M}}(\Gamma^{(\zeta)}_{\mathfrak{M}}).$$

From the theory of inductive definitions we know that for some minimal ζ_0, $\Gamma^{\zeta_0}_{\mathfrak{M}}$ is the least fixed point of $\Gamma_{\mathfrak{M}}$, so

$$\Gamma^{\zeta_0}_{\mathfrak{M}} = \Gamma^{(\zeta_0)}_{\mathfrak{M}}.$$

We define $\rightarrow_{\mathfrak{M}} = \Gamma^{\zeta_0}_{\mathfrak{M}}$. It is clear that this $\rightarrow_{\mathfrak{M}}$ indeed satisfies

$$\boldsymbol{a}\vec{B} \rightarrow_{\mathfrak{M}} \ulcorner a\vec{b}\urcorner \qquad \text{if} \qquad \vec{B} \rhd_{\beta\mathfrak{M}} \vec{b}.$$

As a consequence, the standard representations behave in the following way, as expected.

1.1.12. LEMMA. *Let* $a \in \mathfrak{M}$. *Then* $\underline{a} \rhd_{\beta\mathfrak{M}} a$.

PROOF. Note that $\underline{a} \rhd_{\beta\mathfrak{M}} a$ iff $\underline{a}\,\underline{\vec{b}} =_{\beta\mathfrak{M}} \underline{a\vec{b}}$. One verifies the property for each $a \in \mathfrak{M}_\sigma$ by induction on the height of σ. $\square$

Recently, Jäger and Strahm (1994) describe a method for generating a reduction relation based on a first-order applicative theory, using similar transfinite induction techniques (but without the underlining technique used in the next section).

The original reduction system of Kleene (1962) used more restrictive contraction rules (requiring for a redex $\boldsymbol{a}M$ that M is closed and in normal form). The resulting reduction system, however, is not suitable for our applications in the next chapter. A system with less restrictive contraction rules (only requiring closedness) is described in van Draanen (1989). Our reduction relation is the most general: there are no syntactical restrictions on the form of an oracle redex. As a consequence, the proof of the Church-Rosser is more complicated than earlier ones. Our proof (see the next section) has been inspired by van Draanen (1989).

1.2. The Church-Rosser Property

In this section we will show that $\beta\mathfrak{M}$-reduction is Church-Rosser, or *confluent*.

1.2.1. Definition. We say that a reduction relation $\to_R$ on $\Lambda\mathfrak{M}$ is *Church-Rosser* if for all $M, M_1, M_2 \in \Lambda\mathfrak{M}$

$$M \twoheadrightarrow_R M_1 \wedge M \twoheadrightarrow_R M_2 \;\Rightarrow\; \exists N\,[M_1 \twoheadrightarrow_R N \wedge M_2 \twoheadrightarrow_R N],$$

in a picture

$$\begin{tikzcd} & M \arrow[dl, "R"'] \arrow[dr, "R"] & \\ N_1 \arrow[dr, dotted, "R"'] & & N_2 \arrow[dl, dotted, "R"] \\ & N & \end{tikzcd}$$

The Church-Rosser property has two important consequences: normal forms are unique, and normal forms can be found by reduction: if M has R-normal form N, then $M \twoheadrightarrow_R N$.

1.2.2. Main Theorem. *$\beta\mathfrak{M}$-reduction is Church-Rosser.*

The Main Theorem will be proved by showing for each ordinal ζ that the reduction relation $\to_\beta \cup \Gamma^\zeta_{\mathfrak{M}}$ is Church-Rosser, using transfinite induction.

To simplify notation, let $\to_\zeta$ abbreviate $\Gamma^\zeta_{\mathfrak{M}}$. Similarly one defines $\to_{\beta\zeta}$, $\to_{(\zeta)}$, $=_{\beta\zeta}$, etcetera.

The proof will occupy the rest of this section. We start with some auxiliary results.

1.2.3. Substitution Lemma. *Let $M, N, L \in \Lambda\mathfrak{M}$ and $x, y \in V$, such that $x \not\equiv y$ and $x \notin \mathrm{FV}(L)$. Then*

$$M[x := N][y := L] \equiv M[y := L][x := N[y := L]].$$

Proof. By an easy induction on M. $\square$

1.2.4. Lemma. *Let $M, N \in \Lambda\mathfrak{M}$. If $M =_{\beta(\zeta)} N$, then for some $\vartheta < \zeta$ one has $M =_{\beta\vartheta} N$.*

Proof. By induction on the generation of $=_{\beta(\zeta)}$. $\square$

1.2.5. Proposition. *The relation $=_{\beta\zeta}$ is substitutive, i.e.*

$$M =_{\beta\zeta} M' \;\Rightarrow\; M[x := N] =_{\beta\zeta} M'[x := N].$$

PROOF. By transfinite induction on ζ, and induction on the generation of $=_{\beta\zeta}$. We only treat the prime cases $\to_\beta$ and $\to_\zeta$. Let L^* denote $L[x := N]$. As to the (β) contraction rule, note that

$$\begin{array}{rcll} ((\lambda y.P)Q)^* & \equiv & (\lambda y.P^*)Q^* & \\ & \to_\beta & P^*[y := Q^*] & \\ & \equiv & (P[y := Q])^*, & \text{by the substitution lemma.} \end{array}$$

Now consider $\boldsymbol{a}\vec{B} \to_\zeta \ulcorner a\vec{b}\urcorner$, where $\vec{B} \rhd_{\beta(\zeta)} \vec{b}$. We claim that $\vec{B}^* \rhd_{\beta(\zeta)} \vec{b}$. As to component i, say $b_i \in \mathfrak{M}_{\vec{\sigma}\to\mathbf{0}}$. Let $\vec{c} \in \mathfrak{M}_{\vec{\sigma}}$. Then

$$B_i\underline{\vec{c}} =_{\beta(\zeta)} \ulcorner b_i\vec{c}\urcorner$$

so by Lemma 1.2.4, for some $\vartheta < \zeta$

$$B_i\underline{\vec{c}} =_{\beta\vartheta} \ulcorner b_i\vec{c}\urcorner.$$

Therefore by the induction hypothesis (of the transfinite induction)

$$(B_i\underline{\vec{c}})^* =_{\beta\vartheta} (\ulcorner b_i\vec{c}\urcorner)^*,$$

which means

$$B_i^*\underline{\vec{c}} =_{\beta\vartheta} \ulcorner b_i\vec{c}\urcorner.$$

This proves the claim. Now $\boldsymbol{a}\vec{B}^* \to_\zeta \ulcorner a\vec{b}\urcorner$ and we are done. $\square$

1.2.6. COROLLARY. *$\rhd_{\beta\zeta}$ is substitutive, i.e.*

$$A \rhd_{\beta\zeta} a \;\Rightarrow\; A[x := N] \rhd_{\beta\zeta} a.$$

Now we can show that $\to_{\beta\zeta}$ is Church-Rosser for each ζ, by a transfinite induction.

In the sequel, let ζ be an ordinal number, and suppose $\beta\vartheta$-reduction is Church-Rosser for all $\vartheta < \zeta$. We will use 'main induction hypothesis' to refer to this assumption.

We use a method due to Hindley (1964) and Rosen (1973).

1.2.7. HINDLEY-ROSEN LEMMA. *Let $\to_1$ and $\to_2$ be reduction relations. Suppose*

(1) *$\to_1$ is Church-Rosser,*
(2) *$\to_2$ is Church-Rosser,*
(3) *$\twoheadrightarrow_1$ and $\twoheadrightarrow_2$ commute, i.e.*

$$\begin{tikzcd} & M \arrow[dl, "1"'] \arrow[dr, "2"] & \\ N_1 \arrow[dr, dotted, "2"'] & & N_2 \arrow[dl, dotted, "1"] \\ & N & \end{tikzcd}$$

Then $\to_1 \cup \to_2$ is Church-Rosser.

PROOF. By an easy diagram chase. $\Box$

1.2.8. LEMMA. *Let* $M \xrightarrow{\equiv}_\beta N$ *express that* $M \to_\beta N$ *or* $M \equiv N$. *Then*

$$\begin{tikzcd}
& M \arrow[dl, "\beta"'] \arrow[dr, "\zeta"] & \\
N_1 \arrow[dr, dotted, "\zeta"'] & & N_2 \arrow[dl, dotted, "\equiv"', "\beta"] \\
& N &
\end{tikzcd}$$

PROOF. Say $M \xrightarrow{\Delta_1}_\beta N_1$, $M \xrightarrow{\Delta_2}_\zeta N_2$ where $\Delta_1 \equiv (\lambda x.P)Q$, $\Delta_2 \equiv \boldsymbol{a}\vec{B}$ with $\vec{B} \rhd_{\beta(\zeta)} \vec{b}$. (This is the 'most complex situation' for $\to_\zeta$; the case that $M \to N_2$ according to (ζ) is treated similarly.) We distinguish cases as to the relative positions of Δ_1 and Δ_2 in M.

Case 1. Δ_1 and Δ_2 are disjoint. Then the result is trivial.

Case 2. $\Delta_1 \subseteq \Delta_2$. Let $\vec{B}'$ be the result of contacting Δ_2 in $\vec{B}$. Then $\vec{B}' \rhd_{\beta(\zeta)} \vec{b}$ since $\vec{B} =_\beta \vec{B}'$ so the situation is as follows.

$$\begin{tikzcd}
& \boxed{\cdots \boldsymbol{a}\vec{B} \cdots} \arrow[dl, "\beta"'] \arrow[dr, "\zeta"] & \\
\boxed{\cdots \boldsymbol{a}\vec{B}' \cdots} \arrow[rr, "\zeta"'] & & \boxed{\cdots \ulcorner a\vec{b} \urcorner \cdots}
\end{tikzcd}$$

Case 3. $\Delta_2 \subseteq \Delta_1$. Then either $\Delta_2 \subseteq P$ or $\Delta_2 \subseteq Q$.

Case 3a. $\Delta_2 \subseteq P$. Let P' be the result of contracting Δ_2 in P. Then

$$\begin{tikzcd}
& \boxed{\cdots (\lambda x.P)Q \cdots} \arrow[dl, "\beta"'] \arrow[dr, "\zeta"] & \\
\boxed{\cdots P[x := Q] \cdots} \arrow[dr, "\zeta"'] & & \boxed{\cdots (\lambda x.P')Q \cdots} \arrow[dl, "\beta"] \\
& \boxed{\cdots P'[x := Q] \cdots} &
\end{tikzcd}$$

is a correct diagram since $\vec{B}[x := Q] \rhd_{\beta(\zeta)} \vec{b}$ by Corollary 1.2.6.

Case 3b. $\Delta_2 \subseteq Q$. Let Q' be the result of contracting Δ_2 in Q. Then one has

$$\begin{tikzcd}
& \boxed{\cdots(\lambda x.P)Q\cdots} \arrow[dl, "\beta"'] \arrow[dr, "\zeta"] & \\
\boxed{\cdots P[x:=Q]\cdots} \arrow[dr, two heads, "\zeta"'] & & \boxed{\cdots(\lambda x.P)Q'\cdots} \arrow[dl, "\beta"] \\
& \boxed{\cdots P[x:=Q']\cdots} &
\end{tikzcd}$$

where the number of steps in the $\twoheadrightarrow_\zeta$ reduction depends on the multiplicity of x in P. □

1.2.9. Proposition. *$\twoheadrightarrow_\beta$ and $\twoheadrightarrow_\zeta$ commute.*

Proof. By Lemma 1.2.8 one has

$$\begin{tikzcd}
& M \arrow[dl, "\beta"'] \arrow[dr, "\zeta"] & \\
N_1 \arrow[dr, dotted, "\zeta"'] & \equiv & N_2 \arrow[dl, dotted, "\beta"] \\
& N &
\end{tikzcd}$$

Now by a diagram chase suggested in the following figure we are done.

$$\begin{tikzcd}
& & & M \arrow[dl, "\beta"'] \arrow[dr, "\zeta"] & & \\
& & \bullet \arrow[dl, "\beta"'] \arrow[dr, dotted] & \equiv & \bullet \arrow[dl, dotted] \arrow[dr, "\zeta"] & \\
& \bullet \arrow[dl, "\beta"'] \arrow[dr, dotted] & \equiv & \bullet \arrow[dl, dotted] \arrow[dr, dotted] & \equiv & N_2 \arrow[dl, dotted, "\beta"] \\
N_1 \arrow[dr, dotted, "\zeta"'] & \equiv & \bullet \arrow[dl, dotted] \arrow[dr, dotted] & \equiv & \bullet \arrow[dl, dotted, "\beta"] & \\
& \bullet \arrow[dr, dotted, "\zeta"'] & \equiv & \bullet \arrow[dl, dotted, "\beta"] & & \\
& & N & & &
\end{tikzcd}$$

□

In the following we will show that $\rightarrow_\zeta$ is Church-Rosser. We first prove some facts about representation of elements in the type structure.

1.2.10. Non-ambiguity Lemma. (i) *Let* $b_1, b_2 \in \mathfrak{M}_{\vec{\sigma}\to\mathbf{0}}$. *Then*

$$B \rhd_{\beta(\zeta)} b_1,\ B \rhd_{\beta(\zeta)} b_2 \ \Rightarrow\ b_1 = b_2.$$

(ii) *Let* $n_1, n_2 \in \mathbb{N}$. *Then*

$$\boldsymbol{a}\vec{B} \to_\zeta \ulcorner n_1 \urcorner,\ \boldsymbol{a}\vec{B} \to_\zeta \ulcorner n_2 \urcorner \ \Rightarrow\ n_1 = n_2.$$

Proof. (i) Suppose $B \rhd_{\beta(\zeta)} b_1$ and $B \rhd_{\beta(\zeta)} b_2$. Let $\vec{c} \in \mathfrak{M}_{\vec{\sigma}}$. Then

$$B\underline{\vec{c}} =_{\beta(\zeta)} \ulcorner b_1\vec{c} \urcorner,$$

so for some $\vartheta_1 < \zeta$ one has

$$B\underline{\vec{c}} =_{\beta\vartheta_1} \ulcorner b_1\vec{c} \urcorner.$$

Similarly one shows that for some $\vartheta_2 < \zeta$

$$B\underline{\vec{c}} =_{\beta\vartheta_2} \ulcorner b_2\vec{c} \urcorner.$$

Now suppose (without loss of generality) $\vartheta_1 \le \vartheta_2$. Then also $B\underline{\vec{c}} =_{\beta\vartheta_2} \ulcorner b_1\vec{c} \urcorner$. Note that numerals are normal forms; hence $\ulcorner b_1\vec{c} \urcorner \equiv \ulcorner b_2\vec{c} \urcorner$ by the Church-Rosser property for $\beta\vartheta_2$-reduction. Now by extensionality of $\mathfrak{M}$ the result follows.

(ii) By (i). $\square$

1.2.11. Corollary. $\rhd_{\beta(\zeta)}$ *can be considered as a partial mapping.*

In order to keep track of specific ζ-redexes during reduction, we introduce a kind of *underlining* similar to the indexing employed by Barendregt (1984) in the proof of the Church-Rosser property for β-reduction.

1.2.12. Definition (Marking). (i) The set of terms with *marked ζ-redexes* (notation $\underline{\Lambda\mathfrak{M}}$) is defined inductively as follows. Below, M^- denotes M after removal of all markings; this is defined simultaneously.

$$\begin{array}{rcl}
x \in V & \Rightarrow & x \in \underline{\Lambda\mathfrak{M}}, \\
a \in \mathfrak{M}\backslash\mathfrak{M}_{\mathbf{0}} & \Rightarrow & \boldsymbol{a} \in \underline{\Lambda\mathfrak{M}}, \\
M, N \in \underline{\Lambda\mathfrak{M}} & \Rightarrow & (MN) \in \underline{\Lambda\mathfrak{M}}, \\
M \in \underline{\Lambda\mathfrak{M}}, x \in V & \Rightarrow & (\lambda x.M) \in \underline{\Lambda\mathfrak{M}}, \\
\left.\begin{array}{r} a \in \mathfrak{M}_{\vec{\sigma}\to\mathbf{0}}, \vec{B} \in \underline{\Lambda\mathfrak{M}} \\ \exists\vec{b} \in \mathfrak{M}_{\vec{\sigma}}\ [\vec{B}^- \rhd_{\beta(\zeta)} \vec{b}] \end{array}\right\} & \Rightarrow & (\underline{\boldsymbol{a}}\vec{B}) \in \underline{\Lambda\mathfrak{M}}.
\end{array}$$

Moreover

$$\begin{array}{rcl}
x^- & \equiv & x, \\
\boldsymbol{a}^- & \equiv & \boldsymbol{a}, \\
(MN)^- & \equiv & M^-N^-, \\
(\lambda x.M)^- & \equiv & \lambda x.M^-, \\
(\underline{\boldsymbol{a}}\vec{B})^- & \equiv & \boldsymbol{a}\vec{B}^-.
\end{array}$$

(ii) For each $\vartheta \leq \zeta$, the notions of reduction $\to_{\underline{\vartheta}}$, $\to_{\underline{\beta\vartheta}}$ are defined in the same way as before, giving $\underline{\boldsymbol{a}}$ exactly the same behaviour as $\boldsymbol{a}$. So the contraction rules are

$$\begin{array}{rclll} \boldsymbol{a}\vec{B} & \to & \ulcorner a\vec{b}\urcorner & \text{if } \vec{B}^{-} \rhd_{\beta(\vartheta)} \vec{b}, \\ \underline{\boldsymbol{a}}\vec{B} & \to & \ulcorner a\vec{b}\urcorner & \text{if } \vec{B}^{-} \rhd_{\beta(\vartheta)} \vec{b}. \end{array}$$

Notice that $\underline{\Lambda\mathfrak{M}}$ is closed under reduction by Corollary 1.2.6.

Observing that $\underline{\boldsymbol{a}}\vec{B}$ and $\boldsymbol{a}\vec{B}$ act the same, it is not hard to believe that $\twoheadrightarrow_{\underline{\beta\vartheta}}$ is Church-Rosser for each $\vartheta < \zeta$. This is made precise in the following technical results. Below, $M \xrightarrow{-} N$ expresses that $M^{-} \equiv N$.

1.2.13. Lemma (Lifting).

$$\begin{tikzcd} M' \arrow[r, dotted, two heads, "\underline{\beta\vartheta}"] \arrow[d, "-"'] & N' \arrow[d, "-"] \\ M \arrow[r, two heads, "\beta\vartheta"'] & N \end{tikzcd} \qquad \begin{array}{l} M, N \in \Lambda\mathfrak{M}, \\ M', N' \in \underline{\Lambda\mathfrak{M}}. \end{array}$$

Proof. First consider a one step reduction. Then N' is obtained by contracting the corresponding redex in M'. The general statement follows by transitivity. $\square$

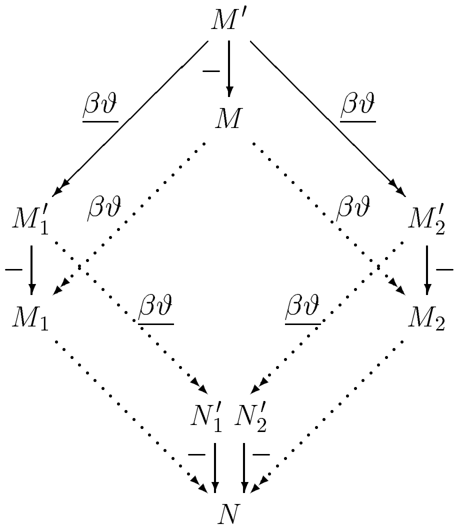
1.2.14. Lemma (Projecting).

$$\begin{tikzcd} M' \arrow[r, two heads, "\underline{\beta\vartheta}"] \arrow[d, "-"'] & N' \arrow[d, "-"] \\ M \arrow[r, dotted, two heads, "\beta\vartheta"'] & N \end{tikzcd} \qquad \begin{array}{l} M', N' \in \underline{\Lambda\mathfrak{M}}, \\ M, N \in \Lambda\mathfrak{M}. \end{array}$$

Proof. Obvious. $\square$

1.2.15. Proposition. *$\underline{\beta\vartheta}$-reduction is Church-Rosser for each $\vartheta < \zeta$.*

Proof. By the main induction hypothesis, $\beta\vartheta$-reduction is Church-Rosser. Now by lifting and projecting reduction sequences we can erect the following diagram.



With a little thought one sees that N_1' and N_2' must be syntactically equal (otherwise trace back differently marked redexes). $\square$

We use denotations $=_{\underline{\beta\vartheta}}$ and $=_{\underline{\beta(\vartheta)}}$ like for the unmarked system. Results for $\rightarrow_{\beta\vartheta}$ can easily be translated into corresponding ones for the marked systems.

1.2.16. Definition. Let $M \in \underline{\Lambda\mathfrak{M}}$. Then $\Phi(M) \in \Lambda\mathfrak{M}$ is defined as follows.

$$\begin{array}{rcl}
\Phi(x) & \equiv & x, \\
\Phi(\boldsymbol{a}) & \equiv & \boldsymbol{a}, \\
\Phi(MN) & \equiv & \Phi(M)\Phi(N), \\
\Phi(\lambda x.M) & \equiv & \lambda x.\Phi(M), \\
\Phi(\underline{\boldsymbol{a}}\vec{B}) & \equiv & \ulcorner a\vec{b}\urcorner, \quad \text{if } \vec{B}^- \rhd_{\beta(\zeta)} \vec{b}.
\end{array}$$

So Φ contracts marked ζ-redexes. Note that Φ is well-defined (use the Non-ambiguity Lemma in the last clause).

1.2.17. Lemma. $\Phi(M[x := N]) \equiv \Phi(M)[x := \Phi(N)]$.

Proof. Induction on M. The cases x, $\boldsymbol{a}$, M_1M_2, and $\lambda x.M_1$ are easy. Now consider $M \equiv \underline{\boldsymbol{a}}\vec{B}$, with $\vec{B}^- \rhd_{\beta(\zeta)} \vec{b}$. By Corollary 1.2.6 also $\vec{B}^-[x := N] \rhd_{\beta(\zeta)} \vec{b}$, so

$$\begin{array}{rcl}
\Phi(\underline{\boldsymbol{a}}\vec{B}[x := N]) & \equiv & \ulcorner a\vec{b}\urcorner \\
& \equiv & \Phi(\underline{\boldsymbol{a}}\vec{B})[x := \Phi(N)]. \ \square
\end{array}$$

In diagrams we use $M \xrightarrow{\Phi} N$ to indicate that $\Phi(M) \equiv N$.

1.2.18. Lemma. *For each $\vartheta < \zeta$*

$$\begin{tikzcd} M \arrow[rr, two heads, "\underline{\beta\vartheta}"] \arrow[d, "\Phi"'] & & N \arrow[d, "\Phi"] \\ \Phi(M) \arrow[rr, dotted, two heads, "\beta\vartheta"'] & & \Phi(N) \end{tikzcd} \qquad M, N \in \underline{\Lambda\mathfrak{M}}.$$

Proof. By transfinite induction on ϑ. Suppose $\vartheta < \zeta$ and the statement holds for all $\vartheta' < \vartheta$. We proceed by induction on the generation of $\twoheadrightarrow_{\underline{\beta\vartheta}}$. First consider a one step reduction $\to_{\underline{\beta\vartheta}}$.

Case 1. $\overline{M} \to_{\underline{\beta\vartheta}} N$ is $(\lambda x.P)Q \to_\beta P[x := Q]$. Then we are done by Lemma 1.2.17.

Case 2. $M \to_{\underline{\beta\vartheta}} N$ is $\boldsymbol{a}\vec{B} \to \ulcorner a\vec{b} \urcorner$ where $\vec{B}^- \rhd_{\beta(\vartheta)} \vec{b}$. Note that $\Phi(\boldsymbol{a}\vec{B}) \equiv \boldsymbol{a}\Phi(\vec{B})$.

Claim. $\Phi(\vec{B}) \rhd_{\beta(\vartheta)} \vec{b}$. Then we are done.

Proof. For all i and $\vec{c}$ (of appropriate types) one has by assumption

$$B_i^- \underline{\vec{c}} =_{\beta(\vartheta)} \ulcorner b_i \vec{c} \urcorner,$$

so for some $\vartheta' < \vartheta$

$$B_i^- \underline{\vec{c}} =_{\beta\vartheta'} \ulcorner b_i \vec{c} \urcorner.$$

Therefore $B_i^- \underline{\vec{c}} \twoheadrightarrow_{\beta\vartheta'} \ulcorner b_i \vec{c} \urcorner$ by the main induction hypothesis, so $B_i \underline{\vec{c}} \twoheadrightarrow_{\underline{\beta\vartheta'}} \ulcorner b_i \vec{c} \urcorner$ by lifting. Now by the hypothesis of the transfinite induction

$$\Phi(B_i \underline{\vec{c}}) \twoheadrightarrow_{\beta\vartheta'} \Phi(\ulcorner b_i \vec{c} \urcorner) \equiv \ulcorner b_i \vec{c} \urcorner.$$

Hence also $\Phi(B_i)\underline{\vec{c}} =_{\beta(\vartheta)} \ulcorner b_i \vec{c} \urcorner$. $\square_{Claim}$

Case 3. $M \to_{\underline{\beta\vartheta}} N$ is $\underline{\boldsymbol{a}}\vec{B} \to \ulcorner a\vec{b} \urcorner$, where $\vec{B}^- \rhd_{\underline{\beta(\vartheta)}} \vec{b}$. Then $\Phi(\underline{\boldsymbol{a}}\vec{B}) \equiv \ulcorner a\vec{b} \urcorner$ and we are done.

The other cases (regarding the compatibility rules) are easy. Moreover, the general statement follows by transitivity. $\square$

1.2.19. Lemma.

$$\begin{tikzcd} & M' \arrow[dd, dotted, two heads, "\zeta"] \\ M \arrow[ur, "-"] \arrow[dr, "\Phi"'] & \\ & \Phi(M) \end{tikzcd} \qquad \begin{array}{l} M \in \underline{\Lambda\mathfrak{M}}, \\ M' \in \Lambda\mathfrak{M}. \end{array}$$

Proof. Induction on the structure of M. The most interesting case is $M \equiv \underline{\boldsymbol{a}}\vec{B}$ where $\vec{B}^- \rhd_{\beta(\zeta)} \vec{b}$. Then $\Phi(M) \equiv \ulcorner a\vec{b} \urcorner$ and $M^- \equiv \boldsymbol{a}\vec{B}^- \to_\zeta \ulcorner a\vec{b} \urcorner$, so we are done. $\square$

1.2.20. Proposition. *Let $\vartheta < \zeta$.*

(i)

$$\begin{tikzcd}
M \arrow[r, two heads, "\beta\vartheta"] \arrow[d, "\zeta"'] & M_2 \arrow[d, dotted, two heads, "\zeta"] \\
M_1 \arrow[r, dotted, two heads, "\beta\vartheta"'] & M_3
\end{tikzcd}$$

(ii)

$$\begin{tikzcd}
M \arrow[r, two heads, "\beta\vartheta"] \arrow[d, two heads, "\zeta"'] & M_2 \arrow[d, dotted, two heads, "\zeta"] \\
M_1 \arrow[r, dotted, two heads, "\beta\vartheta"'] & M_3
\end{tikzcd}$$

PROOF. (i) Say $M \xrightarrow{\Delta}_{\zeta} M_1$. Let $M' \in \underline{\Lambda\mathfrak{M}}$ be the term obtained by marking Δ in M. By the lemmas 1.2.13, 1.2.18 and 1.2.19 one can erect the following diagram.

$$\begin{tikzcd}
 & M \arrow[rr, two heads, "\beta\vartheta"] \arrow[dd, "\zeta"] & & M_2 \arrow[dd, dotted, two heads, "\zeta"] \\
M' \arrow[ur, "-"] \arrow[dr, "\Phi"'] \arrow[rr, dotted, two heads, "\underline{\beta\vartheta}"'] & & M_2' \arrow[ur, "-"] \arrow[dr, "\Phi"'] & \\
 & M_1 \arrow[rr, dotted, two heads, "\beta\vartheta"'] & & M_3
\end{tikzcd}$$

(ii) By (i) and an easy diagram chase. $\square$

1.2.21. LEMMA.

$$\begin{tikzcd}
 & M \arrow[dl, "\zeta"'] \arrow[dr, "\zeta"] & \\
M_1 \arrow[dr, dotted, "\equiv", "\zeta"'] & & M_2 \arrow[dl, dotted, "\equiv"', "\zeta"] \\
 & N &
\end{tikzcd}$$

PROOF. Set $M \xrightarrow{\Delta_1} M_1$, $M \xrightarrow{\Delta_2} M_2$. Distinguish cases as to the relative positions of Δ_1 and Δ_2.

Case 1. Δ_1 and Δ_2 are disjoint. This case is trivial.

Case 2. Δ_1 and Δ_2 coincide. Then we are done by the Non-ambiguity Lemma 1.2.10.

Case 3. $\Delta_1 \subseteq \Delta_2$, say $\Delta_1 \equiv \boldsymbol{a_1}\vec{B}_1$, $\vec{B}_1 \rhd_{\beta(\zeta)} \vec{b}_1$, $\Delta_2 \equiv \boldsymbol{a_2}\vec{B}_2$, $\vec{B}_2 \rhd_{\beta(\zeta)} \vec{b}_2$. Let $\vec{B}'_2$ be the result of replacing Δ_1 in $\vec{B}_2$ by $\ulcorner a_1\vec{b}_1\urcorner$. (In fact, this substitution affects only one B_{2i}.)

Claim. $\vec{B}'_2 \rhd_{\beta(\zeta)} \vec{b}_2$.

Proof. As to B'_{2i}, let $\vec{c}$ be of the appropriate types. Then

$$B_{2i}\underline{\vec{c}} =_{\beta\vartheta} \ulcorner b_{2i}\vec{c}\urcorner$$

for some $\vartheta < \zeta$, so by the main induction hypothesis (see Page 8) $B_{2i}\underline{\vec{c}} \twoheadrightarrow_{\beta\vartheta} \ulcorner b_{2i}\vec{c}\urcorner$. By Proposition 1.2.20 (ii) one has for some P

$$\begin{tikzcd}
B_{2i}\underline{\vec{c}} \arrow[r, twoheadrightarrow, "\beta\vartheta"] \arrow[d, "\zeta"'] & \ulcorner b_{2i}\vec{c}\urcorner \arrow[d, dotted, twoheadrightarrow, "\zeta"] \\
B'_{2i}\underline{\vec{c}} \arrow[r, dotted, twoheadrightarrow, "\beta\vartheta"'] & P
\end{tikzcd}$$

But $\ulcorner b_{2i}\vec{c}\urcorner$ is in ζ-nf so $P \equiv \ulcorner b_{2i}\vec{c}\urcorner$. Therefore $B'_{2i}\underline{\vec{c}} =_{\beta\vartheta} \ulcorner b_{2i}\vec{c}\urcorner$ and hence $B'_{2i}\underline{\vec{c}} =_{\beta(\zeta)} \ulcorner b_{2i}\vec{c}\urcorner$. $\square_{Claim}$

So we have

$$\begin{tikzcd}
& \boxed{\cdots \boldsymbol{a_2}\vec{B}_2 \cdots} \arrow[dl, "\zeta"'] \arrow[dr, "\zeta"] & \\
\boxed{\cdots \boldsymbol{a_2}\vec{B}'_2 \cdots} \arrow[rr, "\zeta"'] & & \boxed{\cdots \ulcorner a_2\vec{b}_2\urcorner \cdots}
\end{tikzcd} \qquad \square$$

1.2.22. Proposition. *ζ-reduction is Church-Rosser.*

Proof. By Lemma 1.2.21 one has

$$\begin{tikzcd}
& M \arrow[dl, "\zeta"'] \arrow[dr, "\zeta"] \arrow[d, phantom, "\equiv\quad\equiv"] & \\
M_1 \arrow[dr, dotted, "\zeta"'] & \equiv\quad\equiv & M_2 \arrow[dl, dotted, "\zeta"] \\
& N &
\end{tikzcd}$$

Now the following diagram chase shows that $\to_\zeta$ is Church-Rosser.

$$\begin{tikzcd}
 & & & M \arrow[dl, "\zeta"'] \arrow[dr, "\zeta"] & & \\
 & & \bullet \arrow[dl, "\zeta"'] \arrow[dr, dotted, "\equiv"] & & \bullet \arrow[dl, dotted, "\equiv"'] \arrow[dr, "\zeta"] & \\
 & \bullet \arrow[dl, "\zeta"'] \arrow[dr, dotted, "\equiv"] & & \bullet \arrow[dl, dotted, "\equiv"'] \arrow[dr, dotted, "\equiv"] & & N_2 \arrow[dl, dotted, "\equiv"', "\zeta"] \\
N_1 \arrow[dr, dotted, "\equiv", "\zeta"'] & & \bullet \arrow[dl, dotted, "\equiv"'] \arrow[dr, dotted, "\equiv"] & & \bullet \arrow[dl, dotted, "\equiv"', "\zeta"] & \\
 & \bullet \arrow[dr, dotted, "\equiv", "\zeta"'] & & \bullet \arrow[dl, dotted, "\equiv"', "\zeta"] & & \\
 & & N & & &
\end{tikzcd}$$

□

1.2.23. Corollary. *$\beta\zeta$-reduction is Church-Rosser.*

Proof. By the Hindley-Rosen Lemma, using the propositions 1.2.9 and 1.2.22, and the fact that β-reduction is Church-Rosser. □

Now we have completed the main transfinite induction.

1.2.24. Theorem. *$\beta\mathfrak{M}$-reduction is Church-Rosser.*

Chapter 2

First- and Second-order Lambda Calculus

2.1. Syntax

In this section we describe the syntax of some systems of first-order lambda calculus and of their second-order extensions.

The first-order system $\boldsymbol{\lambda}^{\boldsymbol{\tau}}$, the so-called *simply typed lambda calculus* was introduced by Church (1940). Its second-order extension $\boldsymbol{\lambda 2}$ of *polymorphic lambda calculus* is due to Girard (1972).

The first and second order systems are distinguished by their respective sets of types, $\mathbb{T}_1$, $\mathbb{T}_2$. The set $\mathbb{T}_1$ has been introduced before (see Section 1.1).

2.1.1. Definition. The sets of first and second order *types* are given by the following abstract syntax.

$$\begin{aligned}
\mathbb{T}_1 &= \mathbb{C} \mid \mathbb{T}_1 {\rightarrow} \mathbb{T}_1, \\
\mathbb{T}_2 &= \mathbb{C} \mid \mathbb{V} \mid \mathbb{T}_2 {\rightarrow} \mathbb{T}_2 \mid \forall \mathbb{V}.\mathbb{T}_2.
\end{aligned}$$

Here $\mathbb{V} = \{\alpha, \beta, \alpha', \ldots\}$ is an infinite set of *type variables*, and $\mathbb{C}$ is a set of *type constants*. In this work we take $\mathbb{C} = \{\mathbf{0}\}$. Types from $\mathbb{T}_1$ are sometimes called *simple* types and types from $\mathbb{T}_2$ *polymorphic* types. Note that $\mathbb{T}_1 \subset \mathbb{T}_2$. In the sequel, $\sigma, \tau, \sigma', \ldots$ range over types. It is convenient to single out some special simple types called *pure types*: $\mathbf{1} \equiv \mathbf{0}{\rightarrow}\mathbf{0}$, $\mathbf{2} \equiv \mathbf{1}{\rightarrow}\mathbf{0}$, and so on.

The terms of the systems are built from typed variables and typed constants (specific for each system), using application and lambda abstraction. Below, $\boldsymbol{\lambda}\square$ ranges over systems.

2.1.2. Definition. (i) For each type σ, let

$$\mathrm{Var}_\sigma = \{v_0^\sigma, v_1^\sigma, \ldots\}$$

be an infinite set of *variables of type* σ. Below, $x, y, z, \ldots$ range over variables; their respective types are indicated by superscripts (x^σ) when necessary.

(ii) For each type σ we assume a set $\mathrm{Cons}_\sigma(\boldsymbol{\lambda}\square)$ of *constants of type* σ to be given. For the base systems $\boldsymbol{\lambda}^{\boldsymbol{\tau}}$ and $\boldsymbol{\lambda 2}$ one takes $\mathrm{Cons}_\sigma(\boldsymbol{\lambda}\square) = \emptyset$ for each σ.

(iii) For each $\sigma \in \mathbb{T}_1, \mathbb{T}_2$ respectively, the set of $\boldsymbol{\lambda}\Box$-*terms of type* σ (notation $\mathrm{Term}_\sigma(\boldsymbol{\lambda}\Box)$) is defined inductively as follows.

$$\begin{array}{rcl}
x^\sigma \in \mathrm{Var}_\sigma & \Rightarrow & x^\sigma \in \mathrm{Term}_\sigma(\boldsymbol{\lambda}\Box), \\
c \in \mathrm{Cons}_\sigma(\boldsymbol{\lambda}\Box) & \Rightarrow & c \in \mathrm{Term}_\sigma(\boldsymbol{\lambda}\Box), \\
M \in \mathrm{Term}_{\sigma\to\tau}(\boldsymbol{\lambda}\Box),\ N \in \mathrm{Term}_\sigma(\boldsymbol{\lambda}\Box) & \Rightarrow & (MN) \in \mathrm{Term}_\tau(\boldsymbol{\lambda}\Box), \\
M \in \mathrm{Term}_\tau(\boldsymbol{\lambda}\Box),\ x^\sigma \in \mathrm{Var}_\sigma & \Rightarrow & (\lambda x^\sigma.M) \in \mathrm{Term}_{\sigma\to\tau}(\boldsymbol{\lambda}\Box).
\end{array}$$

For the second order systems we also include

$$\begin{array}{rcl}
M \in \mathrm{Term}_{\forall\alpha.\sigma}(\boldsymbol{\lambda}\Box),\ \tau \in \mathbb{T}_2 & \Rightarrow & (M\tau) \in \mathrm{Term}_{\sigma[\alpha:=\tau]}(\boldsymbol{\lambda}\Box), \\
M \in \mathrm{Term}_\sigma(\boldsymbol{\lambda}\Box),\ \alpha \in \mathbb{V} & \Rightarrow & (\Lambda\alpha.M) \in \mathrm{Term}_{\forall\alpha.\sigma}(\boldsymbol{\lambda}\Box).
\end{array}$$

(iv) The set of $\boldsymbol{\lambda}\Box$-*terms* is

$$\mathrm{Term}(\boldsymbol{\lambda}\Box) = \bigcup_\sigma \mathrm{Term}_\sigma(\boldsymbol{\lambda}\Box).$$

For $\boldsymbol{\lambda}\Box$-terms we use the denotational conventions of Barendregt (1984).

2.1.3. Example. (i) Typical examples of $\boldsymbol{\lambda}^{\boldsymbol{\tau}}$-terms are

$$\begin{array}{rll}
x^{\mathbf{0}} & \text{of type} & \mathbf{0}, \\
f^{\mathbf{1}} & \text{of type} & \mathbf{1}, \\
\lambda f^{\mathbf{1}} x^{\mathbf{0}}.f(fx) & \text{of type} & \mathbf{1}\to\mathbf{0}\to\mathbf{0}, \\
(\lambda f^{\mathbf{1}} x^{\mathbf{0}}.f(fx))(\lambda y^{\mathbf{0}}.y) & \text{of type} & \mathbf{0}\to\mathbf{0} \quad (\equiv \mathbf{1}).
\end{array}$$

(ii) Typical examples of $\boldsymbol{\lambda}\mathbf{2}$-terms are

$$\begin{array}{rll}
\Lambda\alpha.\lambda x^\alpha.x & \text{of type} & \forall\alpha.\alpha\to\alpha, \\
\Lambda\alpha.\lambda f^{\alpha\to\alpha} x^\alpha.f(fx) & \text{of type} & \forall\alpha.(\alpha\to\alpha)\to\alpha\to\alpha, \\
(\Lambda\alpha.\lambda x^\alpha.x)\mathbf{1} & \text{of type} & \mathbf{1}\to\mathbf{1}.
\end{array}$$

Here and below we assume familiarity with notions such as $\mathrm{FV}(M)$, the set of free variables of M, *hygienic* substitution (expressed by $[x := N]$, $[\alpha := \tau]$ etc.), closed term, and so on. The usual care in dealing with variables should be exercised. For example, in $\Lambda\alpha.M$ we assume that the type variable α does not occur free in any type of a free term variable occurring in M. Furthermore we shall tacitly assume that terms are well-typed and shall reduce type superscripts to a minimum that is needed to reconstruct the types of the constituents of a well-typed term. Below, $F, \ldots, M, M', \ldots, Y$ range over terms.

For the moment, we focus on *equational* $\boldsymbol{\lambda}\Box$-theories. We view these systems in the first place as models of (higher-order, subrecursive) computation. In this view the choice for equational theories is natural. Moreover, equational theories exhibit an attractive conceptual simplicity. In Chapter 3, we will consider extensions with quantifier-free arithmetic and with propositional logic.

2.1.4. DEFINITION. The *formulas* of $\boldsymbol{\lambda}\Box$ are *equations* $M =_\sigma N$ with σ a type of $\boldsymbol{\lambda}\Box$ and M and N terms of $\boldsymbol{\lambda}\Box$ of type σ. Typical examples of such equations can be found in the inference rules that define the theories $\boldsymbol{\lambda}\Box$ below. In the sequel, $E, E', E(x,y), \ldots$ range over equations, and (to smoothen future extensions) $\varphi, \varphi', \ldots$ over arbitrary formulas. We shall omit the type subscript in equations when confusion is not likely.

2.1.5. DEFINITION. The *theories* $\boldsymbol{\lambda}\Box$ are built up from axioms and rules that naturally divide into two groups.

1. *Lambda calculus* axioms and rules. We distinguish the following subgroups.

   (L1) Basic axioms and rules for simply typed lambda calculus (first order).
   Primary axioms:

   $$(\lambda x^\sigma.M)N =_\tau M[x := N] \tag{β_1}$$

   and

   $$\lambda x^\sigma.Mx =_{\sigma\to\tau} M, \tag{η_1}$$

   provided $x \notin \mathrm{FV}(M)$.
   Equality rules:

   $$M =_\sigma M \qquad \frac{M =_\sigma N}{N =_\sigma M} \qquad \frac{M =_\sigma N \qquad N =_\sigma L}{M =_\sigma L}$$

   Compatibility rules:

   $$\frac{M =_{\sigma\to\tau} N}{ML =_\tau NL} \qquad \frac{M =_\sigma N}{FM =_\tau FN} \qquad \frac{M =_\tau N}{\lambda x^\sigma.M =_{\sigma\to\tau} \lambda x^\sigma.N}\,(\xi_1)$$

   (L2) Axioms and rules for polymorphism (second order).
   Primary axioms:

   $$(\Lambda\alpha.M)\tau =_{\sigma[\alpha:=\tau]} M[\alpha := \tau] \tag{β_2}$$

   and

   $$\Lambda\alpha.M\alpha =_{\forall\alpha.\sigma} M, \tag{η_2}$$

   provided that α is *loose* in M, i.e. not free in any type occuring in M (optional rule, not used in this work).
   Equality and compatibility axioms/rules extended to the second order case, including:

   $$\frac{M =_{\forall\alpha.\sigma} N}{M\tau =_{\sigma[\alpha:=\tau]} N\tau} \qquad \frac{M =_\sigma N}{\Lambda\alpha.M =_{\sigma\to\tau} \Lambda\alpha.N}\,(\xi_2)$$

2. Defining equations for *constants*, to be decribed separately for each system.

The corresponding *deduction relations* $\vdash_{\boldsymbol{\lambda}\Box}$ are defined as usual in natural deduction systems.

For some extensions of the systems described in this chapter, it is necessary to allow derivations to depend on *assumptions* ($\Gamma \vdash \varphi$ as opposed to just $\vdash \varphi$), which is no problem in a natural deduction system. This is not standard in lambda calculus; some additional care in dealing with variables should be exercised. The ξ-rule has to be restricted: $\lambda x.M = \lambda x.N$ can only be inferred from $M = N$ when x does not occur free in any assumption on which $M = N$ depends.

2.1.6. Remark. The η-axiom is in fact equivalent to the following *rule of extensionality*.

$$\frac{Mx^\sigma =_\tau Nx^\sigma}{M =_{\sigma\to\tau} N}\ (\textsc{ext})$$

Here x must neither occur in $\mathrm{FV}(M)$, nor in $\mathrm{FV}(N)$, nor in any assumption on which the premiss depends. The η-axiom will mostly be used in the form of the rule (ext).

One of the best known extensions of $\boldsymbol{\lambda^\tau}$ is *Gödel's* $\mathbf{T}$, which results by adding constants for natural numbers and primitive recursion. We refer to this system as $\boldsymbol{\lambda}\mathbf{T}$. Its second order version $\boldsymbol{\lambda}\mathbf{2T}$ is in fact equivalent to Girard's *Système* $\mathbf{F}$.

2.1.7. Definition. (i) The systems $\boldsymbol{\lambda}\mathbf{T}$ (first order) and $\boldsymbol{\lambda}\mathbf{2T}$ (second order) are based on $\boldsymbol{\lambda^\tau}$ (resp. $\boldsymbol{\lambda}\mathbf{2}$), but allow the typed *constants*

$$\boldsymbol{0}, \boldsymbol{S}, \boldsymbol{R}_\sigma,$$

where $\boldsymbol{0}$ is of type $\mathbf{0}$ and $\boldsymbol{S}$ is of type $\mathbf{1}$, with intended interpretation *zero* respectively the *successor function* (the intended interpretation of $\mathbf{0}$ is the set of natural numbers). We use the abbreviations $\overline{0} \equiv \boldsymbol{0}$, $\overline{1} \equiv \boldsymbol{S0}$, $\overline{2} \equiv \boldsymbol{S}(\boldsymbol{S0})$, and so on. The constants $\boldsymbol{R}_\sigma$ of type $\sigma\to(\mathbf{0}\to\sigma\to\sigma)\to\mathbf{0}\to\sigma$ are added for all types σ of the system in question; their intended interpretation is that of *primitive recursor*.

(ii) The *rules* for these constants are the following.

$$\boldsymbol{R}_\sigma MN\boldsymbol{0} =_\sigma M \qquad\qquad \boldsymbol{R}_\sigma MN(\boldsymbol{S}P) =_\sigma NP(\boldsymbol{R}_\sigma MNP) \qquad\qquad (\mathrm{CR})$$

A typical example of a $\boldsymbol{\lambda}\mathbf{T}$-term is $\mathsf{Add} \equiv \lambda x^{\mathbf{0}}y^{\mathbf{0}}.\boldsymbol{R}_{\mathbf{0}}x(\lambda z^{\mathbf{0}}p^{\mathbf{0}}.\boldsymbol{S}p)y$. Then, e.g., $\mathsf{Add}\,\overline{2}\,\overline{3} =_{\mathbf{0}} \overline{5}$ is provable in $\boldsymbol{\lambda}\mathbf{T}$.

Let us expand a bit on the relation between first-order and second-order systems.

2.1.8. Definition. Let $\boldsymbol{\lambda}\Box$ be a first-order system. The *plain polymorphic extension* of $\boldsymbol{\lambda}\Box$ (notation $(\boldsymbol{\lambda}\Box)^2$) is the second order system that has the same constants as $\boldsymbol{\lambda}\Box$, and the rules of $\boldsymbol{\lambda}\Box$ extended with (L2).

Obviously, $\boldsymbol{\lambda}\mathbf{2} = (\boldsymbol{\lambda^\tau})^2$. Note that, however, $\boldsymbol{\lambda}\mathbf{2T} \neq (\boldsymbol{\lambda}\mathbf{T})^2$ since $\boldsymbol{\lambda}\mathbf{2T}$ contains primitive recursors $\boldsymbol{R}_\sigma$ for *all* types $\sigma \in \mathbb{T}_2$. As a consequence, extending $\boldsymbol{\lambda}\mathbf{T}$ to $\boldsymbol{\lambda}\mathbf{2T}$ increases the computation power on the first-order numerals $\overline{0}, \overline{1}, \overline{2}, \ldots$ of type $\mathbf{0}$. This can be seen as follows. Write $\boldsymbol{c}_n \equiv \Lambda\alpha\lambda f^{\alpha\to\alpha}x^\alpha.f^n(x)$ for the n-th *polymorphic Church numeral* of type $\mathbf{PolyNat} \equiv \forall\alpha.(\alpha\to\alpha)\to\alpha\to\alpha$.

2.1.9. Theorem. (i) *The* $\boldsymbol{\lambda}\mathbf{T}$*-representable functions on the first-order numerals are exactly those that are provably total in Peano Arithmetic* ($\mathbf{PA}$).

(ii) *The* $\boldsymbol{\lambda 2}$*-representable functions on the polymorphic Church numerals are exactly the provably total functions of second-order Heyting arithmetic* ($\mathbf{HA_2}$).

PROOF. (i) See Gödel (1958).

(ii) See Girard (1972). In Fortune et al. (1983), some examples of rapidly growing definable functions (e.g., ε_0 recursive) are given. $\square$

Using the 'new' primitive recursors in $\boldsymbol{\lambda 2}\mathbf{T}$ one can transfer the computation power on **PolyNat** to $\mathbf{0}$.

2.1.10. PROPOSITION. *In* $\boldsymbol{\lambda 2}\mathbf{T}$ *there exist terms*

$$\begin{array}{lll} \mathsf{T}_1 & \textit{of type} & \mathbf{PolyNat}\to\mathbf{0}, \\ \mathsf{T}_2 & \textit{of type} & \mathbf{0}\to\mathbf{PolyNat}, \end{array}$$

such that for each $n \in \mathbb{N}$

$$\begin{array}{rcl} \mathsf{T}_1\boldsymbol{c}_n & = & \overline{n}, \\ \mathsf{T}_2\overline{n} & = & \boldsymbol{c}_n. \end{array}$$

PROOF. Take

$$\mathsf{T}_1 \equiv \lambda x^{\mathbf{PolyNat}}.x\,\mathbf{0}\boldsymbol{S0},$$

and

$$\begin{array}{rcl} \mathsf{T}_2 & \equiv & \lambda x^{\mathbf{0}}.\boldsymbol{R}_{\mathbf{PolyNat}}\boldsymbol{c}_0(\lambda y^{\mathbf{0}} z^{\mathbf{PolyNat}}.\mathsf{Succ}\,z)x \\ & = & \boldsymbol{R}_{\mathbf{PolyNat}}\boldsymbol{c}_0(\lambda y^{\mathbf{0}}.\mathsf{Succ}), \end{array}$$

where $\mathsf{Succ} \equiv \lambda m^{\mathbf{PolyNat}}\Lambda\alpha\lambda f^{\alpha\to\alpha}x^{\alpha}.f(m\alpha f x)$ represents the successor function on the polymorphic Church numerals. $\square$

2.1.11. COROLLARY. *The* $\boldsymbol{\lambda 2}\mathbf{T}$*-definable functions on the first-order numerals are those that are provably total in* $\mathbf{HA_2}$.

Plain second-order extensions are investigated in Breazu-Tannen and Meyer (1987). We will return to this in Section 2.6.

2.2. Semantics of First-order Systems

In this section we describe the basic semantical notions for $\boldsymbol{\lambda}^{\boldsymbol{\tau}}$ and its extensions. The exposition is based on work by Friedman (1975). The mathematical basis of first-order semantics is the notion of type structure (see Section 1.1).

Below, $\boldsymbol{\lambda}\square$ ranges over the first order systems.

2.2.1. DEFINITION. Let $\mathfrak{M}$ be a type structure.

(i) A (term) *valuation* in $\mathfrak{M}$ is a map

$$\rho : \mathrm{Var}(\boldsymbol{\lambda}\square) \to \mathfrak{M}$$

such that for all variables x^σ

$$\rho(x^\sigma) \in \mathfrak{M}_\sigma.$$

The set of all valuations is denoted by $\mathrm{Val}(\mathfrak{M})$. In the sequel, $\rho, \rho', \ldots$ range over valuations.

(ii) A $\boldsymbol{\lambda}\Box$-*interpretation* in $\mathfrak{M}$ is a map

$$[\![\ ]\!] : \mathrm{Term}(\boldsymbol{\lambda}\Box) \times \mathrm{Val}(\mathfrak{M}) \to \mathfrak{M}$$

(we write $[\![M]\!]_\rho$ instead of $[\![\ ]\!](M, \rho)$) that is *type correct*, i.e. for all M, ρ

$$M \in \mathrm{Term}_\sigma(\boldsymbol{\lambda}\Box) \ \Rightarrow\ [\![M]\!]_\rho \in \mathfrak{M}_\sigma,$$

and moreover satisfies

$$\begin{array}{rcl} [\![x]\!]_\rho & = & \rho(x), \\ [\![MN]\!]_\rho & = & [\![M]\!]_\rho \cdot [\![N]\!]_\rho, \\ [\![\lambda x^\sigma.M]\!]_\rho \cdot a & = & [\![M]\!]_{\rho(x:=a)} \quad \text{for each } a \in \mathfrak{M}_\sigma, \end{array}$$

and

$$\rho \restriction \mathrm{FV}(M) = \rho' \restriction \mathrm{FV}(M) \ \Rightarrow\ [\![M]\!]_\rho = [\![M]\!]_{\rho'}.$$

2.2.2. Definition. A $\boldsymbol{\lambda}\Box$-*structure* consists of a type structure $\mathfrak{M}$ together with a $\boldsymbol{\lambda}\Box$-interpretation in $\mathfrak{M}$:

$$\mathfrak{M} = \langle (\mathfrak{M}_\sigma)_{\sigma\in\mathbb{T}},\ (\mathrm{App}_{\sigma,\tau})_{\sigma,\tau\in\mathbb{T}},\ [\![\ ]\!]\rangle.$$

Notions for type structures, such as ‘extensionality’ and ‘ω-structure’, carry over to $\boldsymbol{\lambda}\Box$-structures in the obvious way.

Recall the full type structures $\mathfrak{M}(X)$. There is a straightforward way to interpret $\boldsymbol{\lambda}^{\boldsymbol{\tau}}$-terms in this structure, namely as functionals. In the $\mathfrak{M}(\mathbb{N})$ case this interpretation can be extended to $\boldsymbol{\lambda}\mathbf{T}$.

2.2.3. Definition. (i) The $\boldsymbol{\lambda}^{\boldsymbol{\tau}}$-interpretation $[\![\ ]\!]$ in $\mathfrak{M}(X)$ is defined by

$$\begin{array}{rcl} [\![x]\!]_\rho & = & \rho(x), \\ [\![MN]\!]_\rho & = & [\![M]\!]_\rho([\![N]\!]_\rho), \\ [\![\lambda x^\sigma.M]\!]_\rho & = & \boldsymbol{\lambda} a{\in}X_\sigma.[\![M]\!]_{\rho(x:=a)}, \end{array}$$

where $\boldsymbol{\lambda}$ denotes meta lambda abstraction. One easily verifies that $[\![\ ]\!]$ satisfies the requirements in Definition 2.2.1. This gives a $\boldsymbol{\lambda}^{\boldsymbol{\tau}}$-structure, which is also denoted by $\mathfrak{M}(X)$.

(ii) Consider $\mathfrak{M}(\mathbb{N})$. Extend the interpretation in (i) with

$$\begin{array}{rcl} [\![\boldsymbol{0}]\!]_\rho & = & 0, \\ [\![\boldsymbol{S}]\!]_\rho & = & \boldsymbol{\lambda} n{\in}\mathbb{N}.n{+}1, \\ [\![\boldsymbol{R}_\sigma]\!]_\rho & = & f_\sigma, \end{array}$$

where f_σ is such that

$$\begin{array}{rcll} f_\sigma abn & = & a & \text{if } n = 0, \\ & = & b(n{-}1)(f_\sigma ab(n{-}1)) & \text{if } n > 0. \end{array}$$

(Note that such an f_σ exists in $\mathbb{N}_{\sigma\to(\mathbf{0}\to\sigma\to\sigma)\to\mathbf{0}\to\sigma}$.) This gives a $\boldsymbol{\lambda}\mathbf{T}$-interpretation in $\mathfrak{M}(\mathbb{N})$.

The interpretation of types ($[\![\sigma]\!] = \mathfrak{M}_\sigma$) is not mentioned explicitly. The notion of *satisfaction* of equations and sequents is defined in the obvious way.

2.2.4. DEFINITION. Let $\mathfrak{M}$ be a $\boldsymbol{\lambda}\Box$-structure as above.

(i) For $M, N \in \mathrm{Term}_\sigma(\boldsymbol{\lambda}\Box)$ one defines

$$\begin{array}{rcl} \mathfrak{M}, \rho \models M = N & \Leftrightarrow & [\![M]\!]_\rho = [\![N]\!]_\rho, \\ \mathfrak{M} \models M = N & \Leftrightarrow & \text{for all } \rho \quad \mathfrak{M}, \rho \models M = N. \end{array}$$

(ii) $\mathfrak{M}$ *satisfies* $\Gamma \vdash E$ (notation $\Gamma \models_{\mathfrak{M}} E$) if for all ρ

$$\mathfrak{M}, \rho \models E \quad \text{whenever for each } A \text{ in } \Gamma \quad \mathfrak{M}, \rho \models A.$$

(iii) $\mathfrak{M}$ is a *model of* $\boldsymbol{\lambda}\Box$ if

$$\Gamma \vdash_{\boldsymbol{\lambda}\Box} E \;\Rightarrow\; \Gamma \models_{\mathfrak{M}} E.$$

Now we can show that the $\boldsymbol{\lambda}^{\boldsymbol{\tau}}$ structures $\mathfrak{M}(X)$ are in fact models of $\boldsymbol{\lambda}^{\boldsymbol{\tau}}$, and likewise $\mathfrak{M}(\mathbb{N})$ for $\boldsymbol{\lambda}\mathbf{T}$.

We first state and prove some general technical results.

2.2.5. SUBSTITUTION LEMMA. *Let $\mathfrak{M}$ be a $\boldsymbol{\lambda}\Box$-structure. Suppose $\mathfrak{M}$ is extensional. Let $M, N \in \mathrm{Term}(\boldsymbol{\lambda}\Box)$ with $N \in \mathrm{Term}_\sigma(\boldsymbol{\lambda}\Box)$. Then*

$$[\![M[x^\sigma := N]]\!]_\rho = [\![M]\!]_{\rho(x^\sigma := [\![N]\!]_\rho)}.$$

PROOF. Induction on M. $\Box$

2.2.6. REMARK. In fact, the condition that $\mathfrak{M}$ is extensional can be relaxed. As to a more refined approach, call $\mathfrak{M}$ a *ξ-structure* if $\mathfrak{M}$ satisfies the ξ-axiom:

$$\mathfrak{M} \models \forall x^\sigma\; M = N \to \lambda x^\sigma.M = \lambda x^\sigma.N,$$

i.e.

$$\forall a \in \mathfrak{M}_\sigma \;\; [[\![M]\!]_{\rho(x^\sigma := a)} = [\![N]\!]_{\rho(x^\sigma := a)}] \;\Rightarrow\; [\![\lambda x^\sigma.M]\!]_\rho = [\![\lambda x^\sigma.N]\!]_\rho.$$

One can prove the above substitution result for ξ-structures (however with considerable effort like in the untyped case, cf. Barendregt (1984), Lemma 5.3.3). But one also has

$$\mathfrak{M} \text{ is extensional} \;\Rightarrow\; \mathfrak{M} \text{ is a } \xi\text{-structure}$$

(see also Barendsen (1990)). Since our models are extensional we work with extensionality to shorten our proofs.

2.2.7. THEOREM. *Let $\mathfrak{M}$ be extensional. Then $\mathfrak{M}$ is a model of $\boldsymbol{\lambda}^{\boldsymbol{\tau}}$.*

PROOF. By induction on the derivation of $\Gamma \vdash_{\boldsymbol{\lambda}^{\boldsymbol{\tau}}} M = N$. As to the rules and axioms of (L1), for (β_1) note that

$$\begin{array}{rcll} [\![(\lambda x^\sigma.M)N]\!]_\rho & = & [\![\lambda x^\sigma.M]\!]_\rho \cdot [\![N]\!]_\rho & \\ & = & [\![M]\!]_{\rho(x^\sigma := [\![N]\!]_\rho)} & \\ & = & [\![M[x^\sigma := N]]\!]_\rho, & \text{by the Substitution Lemma.} \end{array}$$

The soundness of the (η_1)-axiom is verified as follows. For each $a \in \mathfrak{M}_\sigma$

$$\begin{aligned}[\![\lambda x^\sigma.Mx]\!]_\rho \cdot a &= [\![Mx]\!]_{\rho(x^\sigma:=a)}\\ &= [\![M]\!]_\rho \cdot [\![x]\!]_{\rho(x^\sigma:=a)}, \quad \text{since } x \notin \mathrm{FV}(M)\\ &= [\![M]\!]_\rho \cdot a.\end{aligned}$$

Hence by extensionality $[\![\lambda x^\sigma.Mx]\!]_\rho = [\![M]\!]_\rho$. The validity of the compatibility rules is easily checked. □

2.2.8. COROLLARY. (i) $\mathfrak{M}(X)$ *is a model of* $\boldsymbol{\lambda}^{\boldsymbol{\tau}}$.
(ii) $\mathfrak{M}(\mathbb{N})$ *is a model of* $\boldsymbol{\lambda}\mathbf{T}$.

PROOF. (i) By Theorem 2.2.7.
(ii) By extension of the proof of Theorem 2.2.7. The axioms in (CR) are obviously satisfied by construction of $[\![\boldsymbol{R}_\sigma]\!]$, $[\![\boldsymbol{S}]\!]$ and $[\![\boldsymbol{0}]\!]$. □

2.3. Semantics of Second-order Systems

In this subsection we will describe a generalized version of a well-known model construction for some second-order systems. A general setting for arbitrary second-order models will not be given; the reader is referred to Bruce et al. (1990) and to Breazu-Tannen and Coquand (1988).

A straightforward function-theoretic construction does not work because of the impredicative nature of second-order λ-calculus. E.g., the term $\mathsf{I} \equiv \Lambda\alpha.\lambda x^\alpha.x$ of type $\forall\alpha.\alpha\to\alpha$ can be applied to its own type. Sticking to the function-theoretic setting one is tempted to interpret I as a function. But then $[\![\mathsf{I}]\!] \in [\![\forall\alpha.\alpha\to\alpha]\!]$, whereas at the same time $[\![\forall\alpha.\alpha\to\alpha]\!]$ must be a valid argument for $[\![\mathsf{I}]\!]$, which is impossible by the foundation axiom in set theory.

The model construction described below makes use of the idea of partial equivalence relations on the domain of a certain (untyped) applicative structure.

Since the types may now contain free variables, the type interpretation is no longer fixed but depends on a type valuation dealing with type variables.

The interpretation of terms is very simple: first the type information is erased and then the resulting term is interpreted in the untyped structure mentioned above.

2.3.1. DEFINITION. (i) A *partial applicative structure* is a structure

$$\mathfrak{A} = \langle A, \ \cdot \rangle$$

where A is a non-empty set, called the *domain* of $\mathfrak{A}$, and $\cdot : A \times A \rightharpoonup A$ is a *partial application function*. Again we often write ab for $a \cdot b$.

(ii) By $\simeq$ we denote *partial equality* of expressions: $p \simeq q$ iff either p and q are undefined, or p and q are both defined and equal.

Given a partial applicative structure, we consider the collection of so-called partial equivalence relations on its domain. The second-order types are then interpreted as elements of this collection.

2.3.2. Definition. Let A be a set, and $R \subseteq A \times A$.

(i) R is a *partial equivalence relation* (*per*) if R is symmetric and transitive. By $\mathrm{PER}(A)$ we denote the collection of all such relations.

(ii) The *domain* of R is the set

$$\mathrm{dom}\, R = \{a \in A \mid \exists a' \in A \ \ [a\ R\ a' \text{ or } a'\ R\ a]\}.$$

2.3.3. Lemma. *Let R be a per on A. Then*

$$\mathrm{dom}\, R = \{a \in A \mid a\ R\ a\}.$$

Proof. ($\subseteq$) Let $a \in \mathrm{dom}\, R$, say (without loss of generality) $a\ R\ a'$. Then $a'\ R\ a$ by symmetry, so $a\ R\ a$ by transitivity.

($\supseteq$) Trivial. $\square$

Note that a per R is an equivalence relation on $\mathrm{dom}\, R$.

For the following definitions, fix a partial applicative structure $\mathfrak{A} = \langle A, \ \cdot \rangle$.

2.3.4. Definition. (i) For relations R, S on A, the *function relation* $R{\to}S$ is defined as follows.

$$a\ R{\to}S\ a' \ \Leftrightarrow\ \forall b, b'\ [b\ R\ b' \ \Rightarrow\ ab\ S\ a'b'],$$

where it is understood that $ab\ S\ a'b'$ only holds if both $ab{\downarrow}$ and $a'b'{\downarrow}$.

(ii) If $(R_i)_{i \in I}$ is a collection of relations on A, then the *intersection relation* $\bigwedge_{i \in I} R_i$ is defined by setting

$$a \bigwedge_{i \in I} R_i\ a' \ \Leftrightarrow\ \forall i \in I \ \ [a\ R_i\ a'].$$

2.3.5. Lemma. (i) *If $R, S \in \mathrm{PER}(A)$, then $R{\to}S \in \mathrm{PER}(A)$.*

(ii) *If $(R_i)_{i \in I}$ is a collection in $\mathrm{PER}(A)$, then $\bigwedge_{i \in I} R_i \in \mathrm{PER}(A)$.*

Proof. (i) Suppose $R, S \in \mathrm{PER}(A)$. The symmetry of $R{\to}S$ follows directly from the symmetry of R and S. As to transitivity, let $a, a', a'' \in A$. Suppose

$$a\ R{\to}S\ a', \quad a'\ R{\to}S\ a''.$$

Let $b, b' \in A$ with $b\ R\ b'$. Then $ab\ S\ a'b'$. Moreover $b'\ R\ b'$ by Lemma 2.3.3, so $a'b'\ S\ a''b'$. Therefore by transitivity of S one has $ab\ S\ a''b$. Hence $a\ R{\to}S\ a''$.

(ii) Straightforward. $\square$

If we have an interpretation of the type constants in $\mathrm{PER}(A)$, then each polymorphic type can be mapped into $\mathrm{PER}(A)$. The resulting structure is called a per structure (cf. 'type structure' in the first order case).

2.3.6. Definition. (i) A *per structure* is a structure

$$\mathfrak{P} = \langle A, \ \cdot, \ \mathcal{T} \rangle$$

consisting of a partial applicative structure and a valuation of type constants $\mathcal{T} : \mathbb{C} \to \mathrm{PER}(A)$.

(ii) For each $\sigma \in \mathbb{T}_2$ and type valuation $\xi : \mathbb{V} \to \mathrm{PER}(A)$ the *per interpretation* in $\mathfrak{P}$ of σ under ξ, notation $[\sigma]_\xi = [\sigma]_\xi^{\mathcal{T}} \in \mathrm{PER}(A)$, is defined inductively as follows.

$$\begin{array}{rcl} [c]_\xi & = & \mathcal{T}(c), \\ [\alpha]_\xi & = & \xi(\alpha), \\ [\sigma \to \tau]_\xi & = & [\sigma]_\xi \to [\tau]_\xi, \\ [\forall \alpha.\sigma]_\xi & = & \displaystyle\bigwedge_{R \in \mathrm{PER}(A)} [\sigma]_{\xi(\alpha := R)}. \end{array}$$

This is a sound definition by Lemma 2.3.5.

Since we allow type systems with constants we also consider untyped λ-terms with constants. If C is a given set then we write $\Lambda(C)$ for the set of untyped terms extended with constants from C.

2.3.7. Definition. A *partial* $\lambda(C)$-*interpretation* in $\langle A, \cdot \rangle$ consists of a constant valuation

$$\mathcal{V} : C \to A$$

and a partial map

$$(\![\ ]\!) : \Lambda(C) \times \mathrm{Val}(A) \hookrightarrow A$$

(where $\mathrm{Val}(A)$ is defined in the obvious way), such that

$$\begin{array}{rcl} (\![x]\!)_\rho & \simeq & \rho(x), \\ (\![c]\!)_\rho & \simeq & \mathcal{V}(c) \quad \text{if } c \in C, \\ (\![MN]\!)_\rho & \simeq & (\![M]\!)_\rho \cdot (\![N]\!)_\rho, \\ & (\![\lambda x^\sigma.M]\!)_\rho\downarrow, & \\ (\![\lambda x^\sigma.M]\!)_\rho \cdot a & \simeq & (\![M]\!)_{\rho(x:=a)}, \end{array}$$

and moreover

$$\rho \restriction \mathrm{FV}(M) = \rho' \restriction \mathrm{FV}(M) \ \Rightarrow\ (\![M]\!)_\rho \simeq (\![M]\!)_{\rho'}.$$

2.3.8. Fact. There exists a partial λ-interpretation into $\langle A, \cdot \rangle$ if there exist $k, s \in A$ such that for all $a, b, c \in A$

$$\begin{array}{c} ka\downarrow, \quad kab \simeq a, \\ sa\downarrow, \quad sab\downarrow, \quad sabc \simeq ac(bc). \end{array}$$

Then $(\![\ ]\!)$ is obtained by translating each λ-term to its combinatory variant (using $\boldsymbol{K}, \boldsymbol{S}$) and then using k, s above for the interpretation in A.

An example is Kleene's applicative structure $\langle \mathbb{N}, \cdot \rangle$ with a recursion theoretic interpretation of Λ $(= \Lambda(\emptyset))$.

2.3.9. Example. Set $\mathcal{K} = \langle \mathbb{N}, \cdot \rangle$ where

$$e \cdot x \simeq \{e\}(x).$$

Define $(\![\ ]\!)$ by

$$\begin{array}{rcl} (\![x]\!)_\rho & = & \rho(x), \\ (\![MN]\!)_\rho & \simeq & (\![M]\!)_\rho \cdot (\![N]\!)_\rho, \\ (\![\lambda x.M]\!)_\rho & = & \tilde{\lambda} n.(\![M]\!)_{\rho(x:=n)}, \end{array}$$

where $\tilde{\lambda} n.\psi(n)$ stands for the choice of a (canonical) index for the partial recursive function ψ. One easily verifies that $\boldsymbol{\lambda}\vec{n}.(\![M]\!)_{\rho(\vec{x}:=\vec{n})}$ is partial recursive and indices can be found effectively. Then $(\![\ ]\!)$ is a partial λ-interpretation in $\mathcal{K}$.

Now let $\boldsymbol{\lambda}\Box$ range over the second-order systems.

2.3.10. Definition. A *per structure for* $\boldsymbol{\lambda}\Box$ is a structure

$$\mathfrak{P} = \langle A,\ \cdot,\ \mathcal{T},\ \mathcal{V},\ (\![\ ]\!)\rangle$$

such that

(1) $\langle A,\ \cdot,\ \mathcal{T}\rangle$ is a per structure;

(2) $\mathcal{V}$ and $(\![\ ]\!)$ constitute a partial $\lambda(\mathrm{Cons}(\boldsymbol{\lambda}\Box))$-interpretation in $\langle A, \cdot\rangle$;

(3) $(\mathcal{T}, \mathcal{V})$ is a *constant valuation pair*, i.e. for each $c \in \mathrm{Cons}_\sigma(\boldsymbol{\lambda}\Box)$ and ξ one has $\mathcal{V}(c) \in \mathrm{dom}[\sigma]^{\mathcal{T}}_\xi$.

We omit $\mathcal{V}$ if $\mathrm{Cons}(\boldsymbol{\lambda}\Box) = \emptyset$.

Terms of $\boldsymbol{\lambda}\Box$ are first interpreted in the domain of a per structure $\mathfrak{P}$ by erasing type information and using the interpretation function in $\mathfrak{P}$.

2.3.11. Definition. (i) The *erase-type map* $|\ | : \mathrm{Term}(\boldsymbol{\lambda}\Box) \to \Lambda(\mathrm{Cons}(\boldsymbol{\lambda}\Box))$ is defined inductively as follows.

$$\begin{array}{rcl} |c| & \equiv & c, \\ |x^\sigma| & \equiv & x_\sigma, \\ |MN| & \equiv & |M||N|, \\ |\lambda x^\sigma.M| & \equiv & \lambda x_\sigma.|M|, \\ |M\sigma| & \equiv & |M|, \\ |\Lambda\alpha.M| & \equiv & |M|, \end{array}$$

where x_σ is an untyped variable uniquely determined by x^σ.

(ii) For $M \in \mathrm{Term}(\boldsymbol{\lambda}\Box)$, $\rho : \mathrm{Var}(\boldsymbol{\lambda}\Box) \to A$ one defines the *per interpretation* of M by

$$[M]_\rho = (\![\,|M|\,]\!)_{|\rho|}$$

where $|\rho|$ is such that for each variable x^σ

$$|\rho|(|x^\sigma|) = \rho(x^\sigma).$$

(iii) Let $M \in \mathrm{Term}(\boldsymbol{\lambda}\Box)$. Then (ρ, ξ) is a *valuation pair* for M (notation $(\rho, \xi) \succ M$) if for all $x^\sigma \in \mathrm{FV}(M)$ one has $\rho(x^\sigma) \in \mathrm{dom}[\sigma]_\xi$.

Now we want to show that this gives rise to a type correct interpretation, i.e. each per structure is at least a model of type assignment.

2.3.12. Definition. Let $\mathfrak{P}$ be a per structure for $\boldsymbol{\lambda}\Box$.

(i) Let ρ, ρ' be term valuations in $\mathfrak{P}$, and let ξ be a type valuation. Let $X \subseteq \mathrm{Var}(\boldsymbol{\lambda}\Box)$. Then ρ and ρ' are *ξ-equivalent with respect to X* (notation $\rho \approx_\xi^X \rho'$) if

$$\rho(x^\sigma)\ [\sigma]_\xi\ \rho'(x^\sigma) \quad \text{for all } x^\sigma \in X.$$

(ii) By $\rho \approx_\xi^M \rho'$ we abbreviate $\rho \approx_\xi^{\mathrm{FV}(M)} \rho'$.

Note that if $\rho \approx_\xi^M \rho'$ then both (ρ, ξ) and (ρ', ξ) are valuation pairs for M.

2.3.13. Substitution Lemma. $[\sigma[\alpha := \tau]]_\xi = [\sigma]_{\xi(\alpha := [\tau]_\xi)}$.

Proof. Straightforward induction on the structure of σ. $\Box$

The main technical tool to show type correctness of the per interpretation of terms is the following.

2.3.14. Proposition. *For all $M \in \mathrm{Term}_\sigma(\boldsymbol{\lambda}\Box)$ and all valuations ρ, ρ', ξ*

$$\rho \approx_\xi^M \rho' \ \Rightarrow\ [M]_\rho\ [\sigma]_\xi\ [M]_{\rho'}.$$

Proof. Induction on M. $\Box$

2.3.15. Corollary (Type Correctness). *Let $M \in \mathrm{Term}_\sigma(\boldsymbol{\lambda}\Box)$. Then*

$$(\rho, \xi) \succ M \ \Rightarrow\ [M]_\rho \in \mathrm{dom}[\sigma]_\xi.$$

Proof. Suppose $(\rho, \xi) \succ M$. Then obviously $\rho \approx_\xi^M \rho$, so by Proposition 2.3.14

$$[M]_\rho[\sigma]_\xi[M]_\rho. \ \Box$$

Now a proper interpretation of types and terms in a per structure $\mathfrak{P}$ for $\boldsymbol{\lambda}\Box$, depending on a valuation pair (ρ, ξ), can be given.

2.3.16. Definition. (i) Let $\sigma \in \mathbb{T}_2$. The *interpretation* of σ in $\mathfrak{P}$ under type valuation ξ is

$$[\![\sigma]\!]_\xi = {}^{\mathrm{dom}[\sigma]_\xi}\Big/_{[\sigma]_\xi}\ .$$

(ii) For $M \in \mathrm{Term}_\sigma(\boldsymbol{\lambda}\Box)$ and $(\rho, \xi) \succ M$

$$[\![M]\!]_{\rho,\xi} = [M]_\rho \pmod{[\sigma]_\xi}.$$

By Corollary 2.3.15 this is a sound definition.

We conclude the analysis as follows.

2.3.17. Theorem. *$\mathfrak{P}$ is a model of type assignment, i.e. for all $M \in \mathrm{Term}_\sigma(\boldsymbol{\lambda}\Box)$ and $(\rho,\xi) \succ M$*

$$[\![M]\!]_{\rho,\xi} \in [\![\sigma]\!]_\xi.$$

2.3.18. Definition. The notion of *validity* in $\mathfrak{P}$ is defined in the usual way. Moreover $\mathfrak{P}$ is said to be a *model of* $\boldsymbol{\lambda}\Box$ if

$$\Gamma \vdash_{\boldsymbol{\lambda}\Box} E \;\Rightarrow\; \Gamma \models_{\mathfrak{P}} E.$$

Disregarding the constants, this construction automatically gives a model of $\boldsymbol{\lambda 2}$. This will now be shown.

2.3.19. Substitution Lemma. *For all M, N and appropriate ρ, ξ*

$$[\![M[x^\sigma := N]]\!]_{\rho,\xi} = [\![M]\!]_{\rho(x^\sigma := [\![N]\!]_{\rho,\xi}),\xi}.$$

Proof. By induction on the structure of $M \in \mathrm{Term}_\sigma(\boldsymbol{\lambda}\Box)$ one shows

$$[M[x^\sigma := N]]_\rho \; [\sigma]_\xi \; [M]_{\rho(x^\sigma := [N]_\rho)}. \;\Box$$

Note the similarity with the proof of the Substitution Lemma in the first-order case (2.2.5). In the present case, the built-in extensionality of the per construction is used.

2.3.20. Proposition. *Let $\mathfrak{P}$ be a per structure for $\boldsymbol{\lambda}\Box$. Then the axioms and rules in* (L1) *and* (L2) *are valid in $\mathfrak{P}$.*

Proof. We only treat the principal cases; e.g., the compatibility rules are easily verified.

As to (β_1), note that for each $M \in \mathrm{Term}_\tau(\boldsymbol{\lambda}\Box)$ and each valuation pair (ρ,ξ) for M

$$\begin{array}{rcll} [(\lambda x^\sigma.M)N]_\rho & = & [\lambda x^\sigma.M]_\rho \cdot [N]_\rho & \\ & = & [M]_{\rho(x^\sigma := [N]_\rho)} & \\ & [\tau]_\xi & [M[x^\sigma := N]]_\rho, & \text{by the Substitution Lemma 2.3.19.} \end{array}$$

For (η_1), let $M \in \mathrm{Term}_{\sigma\to\tau}(\boldsymbol{\lambda}\Box)$. Let $a, a' \in A$ such that $a\;[\sigma]_\xi\;a'$. Then

$$\begin{array}{rcll} [\lambda x^\sigma.Mx]_\rho \cdot a & = & [Mx]_{\rho(x^\sigma := a)} & \\ & = & [M]_\rho \cdot [x]_{\rho(x^\sigma := a)} & \\ & [\tau]_\xi & [M]_\rho \cdot [x]_{\rho(x^\sigma := a')}, & \text{by Proposition 2.3.14} \\ & = & [M]_\rho \cdot a'. & \end{array}$$

Hence $[\lambda x^\sigma.Mx]_\rho \; [\sigma\to\tau]_\xi \; [M]_\rho$.

The rules of (L2) are trivially sound by the fact that the type information is erased in the interpretation process. $\Box$

2.3.21. Corollary. *Let $\mathfrak{P}$ be a per structure for $\boldsymbol{\lambda 2}$. Then $\mathfrak{P}$ is a model of $\boldsymbol{\lambda 2}$.*

So the lambda calculus axioms and rules are 'automatically' satisfied in a per structure. This shows that the question whether a per structure $\mathfrak{P}$ is a model of $\boldsymbol{\lambda}\Box$ depends entirely on the constant interpretations $\mathcal{T}, \mathcal{V}$.

The structure $\mathcal{K}$ (see Example 2.3.9) can be extended to a model of $\boldsymbol{\lambda 2 \mathrm{T}}$.

2.3.22. Definition. (i) Using the recursion theorem, determine $r \in \mathbb{N}$ such that in $\mathcal{K}$

$$\begin{array}{rcll} rabn & \simeq & a & \text{if } n = 0, \\ & \simeq & b(n{-}1)(rab(n{-}1)) & \text{if } n > 0. \end{array}$$

(ii) The per structure HEO2 is defined by

$$\mathrm{HEO2} = \langle \mathbb{N},\ \cdot,\ \mathcal{T},\ \mathcal{V},\ (\![\ ]\!) \rangle$$

where

$$\mathcal{T}(\mathbf{0}) = \mathrm{EQ}_{\mathbb{N}} = \{(n,n) \mid n \in \mathbb{N}\}$$

and

$$\begin{array}{rcl} \mathcal{V}(\boldsymbol{0}) & = & 0, \\ \mathcal{V}(\boldsymbol{S}) & = & \tilde{\lambda} n.n{+}1, \\ \mathcal{V}(\boldsymbol{R}_\sigma) & = & r. \end{array}$$

2.3.23. Proposition. HEO2 *is a model of* $\boldsymbol{\lambda 2 \mathrm{T}}$.

Proof. By induction on the derivation showing $\Gamma \vdash_{\boldsymbol{\lambda 2 \mathrm{T}}} \varphi$ one shows the validity of each provable sequent. The rules and axioms in (L1) and (L2) are valid by Proposition 2.3.20. Furthermore for all appropriate M, N, ρ, ξ

$$\begin{array}{rcl} \mathrm{HEO2}, \rho, \xi & \models & \boldsymbol{R}_\sigma M N \boldsymbol{0} =_\sigma M, \\ \mathrm{HEO2}, \rho, \xi & \models & \boldsymbol{R}_\sigma M N (\boldsymbol{S} P) =_\sigma N P (\boldsymbol{R}_\sigma M N P), \end{array}$$

by the above recursion-theoretic construction. $\Box$

If one restrict the types to $\mathbb{T}_1$ we obtains a $\boldsymbol{\lambda^\tau}/\boldsymbol{\lambda \mathrm{T}}$-structure (in the sense of Definition 2.2.2), known as HEO.

2.3.24. Corollary. *Set*

$$\mathrm{HEO} = \langle ([\![\sigma]\!])_{\sigma \in \mathbb{T}_1},\ (\mathrm{App}_{\sigma,\tau})_{\sigma,\tau \in \mathbb{T}_1},\ [\![\ ]\!] \rangle$$

where $\mathrm{App}_{\sigma,\tau}([a],[b]) = [a \cdot b]$, *and* $[\![\ ]\!]$ *is restricted to terms of* $\boldsymbol{\lambda^\tau}/\boldsymbol{\lambda \mathrm{T}}$. *Then* HEO *is a model of* $\boldsymbol{\lambda^\tau}$ *and of* $\boldsymbol{\lambda \mathrm{T}}$.

The structure HEO can be viewed as the first-order fragment of HEO2. In the next section, the relation between first and second order semantical structures will be investigated further.

2.4. Building Per Structures over Type Structures

The rest of this chapter concerns the extension of first-order mathematical structures (type structures) to second-order ones (per structures). The results can be used to extend first-order models to models of polymorphic lambda calculi.

We proceed as follows. We add a type structure to untyped λ-calculus, using the techniques developed in Chapter 1. We consider a per structure based on the applicative theory thus obtained.

This technique can be applied to any type structure. However, some elements of the type structure in question might get lost in the per construction. If the type structure is closed under λ-definability (see below) then the elements survive in the per construction, and one obtains a per structure that is in its simple types isomorphic to the original type structure.

Let $\mathfrak{M}$ be an extensional ω-structure. In this section we will consider $\Lambda^\circ\mathfrak{M}$ (modulo $=_{\beta\mathfrak{M}}$) as an applicative structure. Due to the generality of our Church-Rosser result one can also work with the open term model of $\lambda\mathfrak{M}$, but this is not necessary for our purposes.

2.4.1. Definition. (i) For $M \in \Lambda^\circ\mathfrak{M}$ we write the *$\beta\mathfrak{M}$-equivalence class* of M as

$$\langle\!\langle M \rangle\!\rangle = \{N \in \Lambda^\circ\mathfrak{M} \mid M =_{\beta\mathfrak{M}} N\}.$$

(ii) $\mathcal{L}_{\mathfrak{M}} = \{\langle\!\langle M \rangle\!\rangle \mid M \in \Lambda^\circ\mathfrak{M}\}$.

(iii) *Application* in $\mathcal{L}_{\mathfrak{M}}$ is defined as usual:

$$\langle\!\langle M \rangle\!\rangle \cdot \langle\!\langle N \rangle\!\rangle = \langle\!\langle MN \rangle\!\rangle.$$

Note that this is a sound definition, i.e. the resulting equivalence class is independent of the choice of the representatives of $\langle\!\langle M \rangle\!\rangle$ and $\langle\!\langle N \rangle\!\rangle$.

$\mathcal{L}_{\mathfrak{M}}$ can be extended to a per structure by interpreting $\mathbf{0}$ as a canonical per on the Church numerals.

2.4.2. Definition. The *per structure* over $\mathfrak{M}$ is defined by

$$\mathfrak{P}_{\mathfrak{M}} = \langle \mathcal{L}_{\mathfrak{M}},\ \cdot,\ \mathcal{T} \rangle$$

where

$$\mathcal{T}(\mathbf{0}) = \{(\langle\!\langle \ulcorner n \urcorner \rangle\!\rangle, \langle\!\langle \ulcorner n \urcorner \rangle\!\rangle) \mid n \in \mathfrak{M}_{\mathbf{0}}\}.$$

The rest of this section is devoted to a comparison of $\mathfrak{M}$ and $\mathfrak{P}_{\mathfrak{M}}$.

2.4.3. Definition. Let $\mathfrak{P} = \langle A,\ \cdot,\ \mathcal{T} \rangle$ be a per structure. The *first-order fragment* of $\mathfrak{P}$ (notation $\mathfrak{P}^1$) is the type structure

$$\mathfrak{P}^1 = \langle (\llbracket \sigma \rrbracket)_{\sigma \in \mathbb{T}_1},\ (\cdot_{\sigma,\tau})_{\sigma,\tau \in \mathbb{T}_1} \rangle,$$

where $\llbracket \cdot \rrbracket$ is the per interpretation of types in $\mathfrak{P}$, and $\cdot_{\sigma,\tau}$ is defined from $\cdot$ on the quotient spaces $\mathrm{dom}[\sigma]/[\sigma]$ by setting

$$[a]_{[\sigma\to\tau]} \cdot_{\sigma,\tau} [b]_{[\sigma]} = [a \cdot b]_{[\tau]},$$

where $[x]_R$ denotes the equivalence class of x modulo $R \in \mathrm{PER}(A)$ (provided $x \in \mathrm{dom}\, R$). From the per construction it follows that these application functions are well defined.

In order to formulate the main result of this section we introduce some auxiliary terminology.

2.4.4. Definition. Let $\mathfrak{M}$ be a type structure, and let $\varphi : \mathfrak{M}_{\sigma_1} \times \cdots \times \mathfrak{M}_{\sigma_k} \to \mathfrak{M}_{\mathbf{0}}$ be a map.

(i) Let $F \in \Lambda^o\mathfrak{M}$. Then F is said to $\lambda\mathfrak{M}$-*define* φ if for all $a_1 \in \mathfrak{M}_{\sigma_1}, \ldots, a_k \in \mathfrak{M}_{\sigma_k}$

$$F\underline{a_1} \cdots \underline{a_k} =_{\beta\mathfrak{M}} \underline{\varphi(a_1, \ldots, a_k)},$$

which becomes in vector denotation

$$F\underline{\vec{a}} =_{\beta\mathfrak{M}} \underline{\varphi(\vec{a})}.$$

(ii) φ is $\lambda\mathfrak{M}$-*definable* if φ is $\lambda\mathfrak{M}$-defined by some $F \in \Lambda^o\mathfrak{M}$.

2.4.5. Definition. (i) Let $a \in \mathfrak{M}_{\sigma_1 \to \cdots \to \sigma_k \to \mathbf{0}}$. The *external map* corresponding to a (notation $\widehat{a}$) is defined as follows.

$$\widehat{a} : \mathfrak{M}_{\sigma_1} \times \cdots \times \mathfrak{M}_{\sigma_k} \to \mathfrak{M}_{\mathbf{0}};$$
$$\widehat{a}(b_1, \ldots, b_k) = ab_1 \cdots b_k.$$

(ii) $\mathfrak{M}$ is λ-*closed* if for all $\varphi : \mathfrak{M}_{\vec{\sigma}} \to \mathfrak{M}_{\mathbf{0}}$

$$\varphi \text{ is } \lambda\mathfrak{M}\text{-definable} \;\Rightarrow\; \varphi = \widehat{a} \text{ for some } a \in \mathfrak{M}_{\vec{\sigma} \to \mathbf{0}}.$$

An alternative characterization of λ-closedness can be obtained using Kleene's (1959b) schemata for recursion in higher types.

It will turn out that if $\mathfrak{M}$ is λ-closed then $\mathfrak{P}_{\mathfrak{M}}$ is an extension of $\mathfrak{M}$, i.e., the 'first-order part' of $\mathfrak{P}_{\mathfrak{M}}$ is isomorphic to $\mathfrak{M}$.

We first need a technical result. Recall that $\to_{\mathfrak{M}}$ has been developed in stages. We will show that for any $a \in \mathfrak{M}_\sigma$, the oracle $\boldsymbol{a}$ can be replaced by a term A representing a, in any computation yielding a numeral. We consider $\lambda\mathfrak{M}$-terms with *marked standard representations*, i.e. marked oracles (cf. Section 1.2) and marked numerals, in order to monitor these during computations. (The markings do not affect the reduction relations.) If M, A are terms, then $M[\underline{\underline{a}} := A]$ denotes M with all marked occurrences of $\underline{a}$ replaced by A.

2.4.6. Proposition. *Let $\sigma \in \mathbb{T}_1$ and $a \in \mathfrak{M}_\sigma$. Let $A \in \Lambda\mathfrak{M}$ such that $A \rhd_{\beta\mathfrak{M}} a$. Then for all ζ one has*

(i) $\quad M \rhd_{\beta(\zeta)} b \quad\Rightarrow\quad M[\underline{\underline{a}} := A] \rhd_{\beta\mathfrak{M}} b.$

(ii) $\quad M \twoheadrightarrow_{\beta\zeta} N \quad\Rightarrow\quad M[\underline{\underline{a}} := A] =_{\beta\mathfrak{M}} N[\underline{\underline{a}} := A].$

Proof. We prove (i), (ii) simultaneously by induction on the height of σ. Let M^* denote $M[\underline{\underline{a}} := A]$.

Basis ($\sigma \equiv \mathbf{0}$). Then (i), (ii) are trivial since

$$A \rhd_{\beta\mathfrak{M}} n \;\Leftrightarrow\; A =_{\beta\mathfrak{M}} \ulcorner n \urcorner.$$

Induction step. Let $\sigma \in \mathbb{T}_1$, and suppose the statement holds for all types of smaller height and for every ζ (we refer to this as IH_1). We prove the statement for σ by transfinite induction on ζ. Suppose the result holds for all $\vartheta < \zeta$ (this is IH_2).

As to (i), suppose $M \rhd_{\beta(\zeta)} b$. Let $\vec{c}$ be objects of appropriate types. Then $M\underline{\vec{c}} \twoheadrightarrow_{\beta\vartheta} \ulcorner b\vec{c}\urcorner$ for some $\vartheta < \zeta$. Hence by IH_2(ii) one has

$$M^*\underline{\vec{c}} =_{\beta\mathfrak{M}} \ulcorner b\vec{c}\urcorner.$$

We conclude that $M^* \rhd_{\beta\mathfrak{M}} b$.

As to (ii), we proceed by induction on the generation of $\twoheadrightarrow_{\beta\zeta}$. We only treat the prime cases. The compatibility and transitivity rules are easily verified.

- $M \rightarrow_{\beta\zeta} N$ is

$$(\lambda x.P)Q \rightarrow P[x := Q].$$

Then by substitutivity of $\rightarrow_{\beta\mathfrak{M}}$ one has $(\lambda x.P^*)Q^* \rightarrow_{\beta\mathfrak{M}} P^*[x := Q^*]$ and we are done.

- $M \rightarrow_{\beta\zeta} N$ is

$$\boldsymbol{a}'\vec{B} \rightarrow \ulcorner a'\vec{b}\urcorner \quad \text{since } \vec{B} \rhd_{\beta(\zeta)} \vec{b}.$$

(This includes $\boldsymbol{a}' \equiv \boldsymbol{a}$, i.e. an unmarked occurrence of $\boldsymbol{a}$.) Then $\vec{B}^* \rhd_{\beta(\zeta)} \vec{b}$ by (i), so $\boldsymbol{a}'\vec{B}^* \rightarrow_{\beta\mathfrak{M}} \ulcorner a'\vec{b}\urcorner$.

- $M \rightarrow_{\beta\zeta} N$ is

$$\underline{\boldsymbol{a}}\vec{B} \rightarrow \ulcorner a\vec{b}\urcorner \quad \text{since } \vec{B} \rhd_{\beta(\zeta)} \vec{b}.$$

Note that $A\underline{\vec{b}} \twoheadrightarrow_{\beta\mathfrak{M}} \ulcorner a\vec{b}\urcorner$ because $A \rhd_{\beta\mathfrak{M}} a$. Moreover $\vec{B}^* \rhd_{\beta\mathfrak{M}} \vec{b}$ by (i). Therefore

$$A\vec{B}^* =_{\beta\mathfrak{M}} \ulcorner a\vec{b}\urcorner$$

by repeated application of IH_1(ii), marking the appropriate $\underline{b_i}$. $\square$

Below we will often replace $=_{\beta\mathfrak{M}}$ by $=$ and $\rhd_{\beta\mathfrak{M}}$ by $\rhd$ if there is no danger of confusion.

2.4.7. Corollary. *Let* $a \in \mathfrak{M}_{\vec{\sigma}\rightarrow\mathbf{0}}$.

(i) *Let* $b \in \mathfrak{M}_{\sigma_1}$. *Then*

$$A \rhd a,\ B \rhd b \ \Rightarrow\ AB \rhd ab.$$

(ii) *Let* $\vec{b} \in \mathfrak{M}_{\vec{\sigma}}$. *Then*

$$A \rhd a,\ \vec{B} \rhd \vec{b} \ \Rightarrow\ A\vec{B} = \ulcorner a\vec{b}\urcorner.$$

Proof. (i) Suppose $A \rhd a$. Then for all $b, \vec{c}$ of the appropriate types

$$A\underline{b}\,\underline{\vec{c}} = \ulcorner ab\vec{c}\urcorner,$$

so $A\underline{b} \rhd ab$. Hence $AB \rhd ab$ by Proposition 2.4.6 (i).

(ii) By repeated application of (i). $\square$

2.4.8. CLASSIFICATION THEOREM. *Let $\mathfrak{M}$ be λ-closed. Then for all $\sigma \in \mathbb{T}_1$ one has the following in $\mathfrak{P}_{\mathfrak{M}}$, for each appropriate A and $a, a' \in \mathfrak{M}_\sigma$.*

$$\begin{array}{llcl} (\mathrm{i})_\sigma & \langle\!\langle \underline{a} \rangle\!\rangle \in \mathrm{dom}[\sigma], & & \\ (\mathrm{ii})_\sigma & \langle\!\langle A \rangle\!\rangle \in \mathrm{dom}[\sigma] & \Rightarrow & A \rhd a \textit{ for some } a, \\ (\mathrm{iii})_\sigma & \langle\!\langle A \rangle\!\rangle\ [\sigma]\ \langle\!\langle \underline{a} \rangle\!\rangle & \Leftrightarrow & A \rhd a, \\ (\mathrm{iv})_\sigma & \langle\!\langle \underline{a} \rangle\!\rangle\ [\sigma]\ \langle\!\langle \underline{a'} \rangle\!\rangle & \Rightarrow & a = a'. \end{array}$$

PROOF. By simultaneous induction on the height of σ.

Basis ($\sigma \equiv \mathbf{0}$). The properties $(\mathrm{i})_\mathbf{0}$ and $(\mathrm{ii})_\mathbf{0}$ hold by definition of $\mathcal{T}$ and $\rhd$ respectively. As to $(\mathrm{iii})_\mathbf{0}$, the ($\Rightarrow$) part holds by construction; for ($\Leftarrow$) use closedness of $\langle\!\langle A \rangle\!\rangle$ under $\beta\mathfrak{M}$-conversion. Moreover $(\mathrm{iv})_\mathbf{0}$ holds by the Church-Rosser theorem for $\beta\mathfrak{M}$-reduction.

Induction step ($\sigma \equiv \sigma_1 \to \cdots \to \sigma_k \to \mathbf{0}$). Note that $\underline{a} \equiv \boldsymbol{a}$ in this case.

(i) Let $a \in \mathfrak{M}_{\vec{\sigma} \to \mathbf{0}}$. Then for all $\vec{B} \rhd \vec{b}$

$$\underline{a}\vec{B} \equiv \boldsymbol{a}\vec{B} = \underline{a\vec{b}}.$$

Suppose $\langle\!\langle \vec{B} \rangle\!\rangle\ [\vec{\sigma}]\ \langle\!\langle \vec{B}' \rangle\!\rangle$. Then $\vec{B} \rhd \vec{b}$ for some $\vec{b}$, by $(\mathrm{ii})_{\vec{\sigma}}$, so $\langle\!\langle \vec{B} \rangle\!\rangle\ [\vec{\sigma}]\ \langle\!\langle \underline{\vec{b}} \rangle\!\rangle$ by $(\mathrm{iii})_{\vec{\sigma}}$. Similarly one has $\langle\!\langle \vec{B}' \rangle\!\rangle\ [\vec{\sigma}]\ \langle\!\langle \underline{\vec{b}'} \rangle\!\rangle$ for some $\vec{b}'$. But then $\vec{b} = \vec{b}'$ by $(\mathrm{iv})_{\vec{\sigma}}$. Hence

$$\langle\!\langle \boldsymbol{a} \rangle\!\rangle \cdot \langle\!\langle \vec{B} \rangle\!\rangle = \langle\!\langle \underline{a\vec{b}} \rangle\!\rangle = \langle\!\langle \boldsymbol{a} \rangle\!\rangle \cdot \langle\!\langle \vec{B}' \rangle\!\rangle$$

and we are done.

(ii) Suppose $A \in \mathrm{dom}[\vec{\sigma} \to \mathbf{0}]$. Then for all $\vec{b} \in \mathfrak{M}_{\vec{\sigma}}$ one has by $(\mathrm{i})_{\vec{\sigma}}$

$$\langle\!\langle A \rangle\!\rangle \cdot \langle\!\langle \underline{\vec{b}} \rangle\!\rangle \in \mathrm{dom}[\mathbf{0}].$$

Hence for some $\varphi : \mathfrak{M}_{\vec{\sigma}} \to \mathfrak{M}_\mathbf{0}$ one has

$$A\underline{\vec{b}} = \underline{\varphi(\vec{b})}.$$

Using λ-closedness, determine $a \in \mathfrak{M}_{\vec{\sigma} \to \mathbf{0}}$ such that $\widehat{a} = \varphi$. Then $A \rhd a$.

(iii) ($\Rightarrow$) Suppose $\langle\!\langle A \rangle\!\rangle\ [\sigma]\ \langle\!\langle \boldsymbol{a} \rangle\!\rangle$. Let $\vec{b} \in \mathfrak{M}_{\vec{\sigma}}$. By $(\mathrm{i})_{\vec{\sigma}}$ one has $\langle\!\langle \underline{\vec{b}} \rangle\!\rangle\ [\sigma]\ \langle\!\langle \underline{\vec{b}} \rangle\!\rangle$, so $\underline{\vec{b}} \rhd \vec{b}$ by $(\mathrm{iii})_{\vec{\sigma}}$. Now

$$\begin{array}{rcll} \langle\!\langle A \rangle\!\rangle \cdot \langle\!\langle \underline{\vec{b}} \rangle\!\rangle & = & \langle\!\langle \boldsymbol{a} \rangle\!\rangle \cdot \langle\!\langle \underline{\vec{b}} \rangle\!\rangle, & \text{by definition of } [\sigma] \\ & = & \langle\!\langle \underline{a\vec{b}} \rangle\!\rangle, & \end{array}$$

so $A\underline{\vec{b}} = \underline{a\vec{b}}$ by $(\mathrm{iv})_\mathbf{0}$. Therefore $A \rhd a$.

($\Leftarrow$) Suppose $A \rhd a$. Then for all $\vec{b} \in \mathfrak{M}_{\vec{\sigma}}$

$$A\underline{\vec{b}} = \underline{a\vec{b}}.$$

By Corollary 2.4.7 (ii) one has for each $\vec{B} \rhd \vec{b}$

$$A\vec{B} = \underline{a\vec{b}}.$$

Now suppose $\langle\!\langle \vec{B} \rangle\!\rangle \ [\vec{\sigma}] \ \langle\!\langle \vec{B}' \rangle\!\rangle$. Then it follows by $(\text{ii})_{\vec{\sigma}}$, $(\text{iii})_{\vec{\sigma}}$ and $(\text{iv})_{\vec{\sigma}}$ (cf. the proof of (i)) that $\vec{B}, \vec{B}' \rhd \vec{b}$ for some $\vec{b}$. Therefore

$$\begin{aligned} \langle\!\langle A \rangle\!\rangle \cdot \langle\!\langle \vec{B} \rangle\!\rangle &= \langle\!\langle \underline{a\vec{b}} \rangle\!\rangle \\ &= \langle\!\langle \boldsymbol{a} \rangle\!\rangle \cdot \langle\!\langle \vec{B}' \rangle\!\rangle, \end{aligned}$$

so $\langle\!\langle A \rangle\!\rangle \ [\sigma] \ \langle\!\langle \boldsymbol{a} \rangle\!\rangle$.

(iv) Suppose $\langle\!\langle \boldsymbol{a} \rangle\!\rangle \ [\sigma] \ \langle\!\langle \boldsymbol{a}' \rangle\!\rangle$. Then by $(\text{i})_{\vec{\sigma}}$ one has for all $\vec{b} \in \mathfrak{M}_{\vec{\sigma}}$

$$\boldsymbol{a}\underline{\vec{b}} = \boldsymbol{a}'\underline{\vec{b}}$$

so $a = a'$ by Church-Rosser and extensionality. $\square$

We will now show that each λ-closed $\mathfrak{M}$ is embeddable in $\mathfrak{P}_{\mathfrak{M}}$ (in the obvious sense). The first-order fragment of $\mathfrak{P}_{\mathfrak{M}}$ even corresponds exactly to $\mathfrak{M}$.

2.4.9. Isomorphism Theorem. *Let $\mathfrak{M}$ be λ-closed. Then $\mathfrak{M} \cong \mathfrak{P}^1_{\mathfrak{M}}$.*

Proof. Define for each $\sigma \in \mathbb{T}_1$

$$f_\sigma : \mathfrak{M}_\sigma \to [\![\sigma]\!]$$

by

$$f_\sigma(a) = \langle\!\langle \underline{a} \rangle\!\rangle \quad (\text{mod } [\sigma]).$$

By the Classification Theorem this map is bijective (injectivity follows from $(\text{iv})_\sigma$ and surjectivity from $(\text{ii})_\sigma$). Moreover for all appropriate a, b

$$f_{\sigma\to\tau}(a) \cdot_{\sigma,\tau} f_\sigma(b) = f_\tau(ab)$$

by (iii) of the Classification Theorem, using Corollary 2.4.7 (i). $\square$

2.5. Interpretations in Extended Type Structures

In this section we consider interpretations in the per structures $\mathfrak{P}_{\mathfrak{M}}$. First, untyped terms can be interpreted in this structure.

2.5.1. Definition. Let $M \in \Lambda$ and let $\rho : V \to \mathcal{L}_{\mathfrak{M}}$ be a valuation; say $\text{FV}(M) = \{x_1, \ldots, x_n\}$ and $\rho(x_i) = \langle\!\langle P_i \rangle\!\rangle$. Then define

$$(\![M]\!)_\rho = \langle\!\langle M[\vec{x} := \vec{P}] \rangle\!\rangle.$$

2.5.2. Lemma. $(\![\ ]\!)$ *is a λ-interpretation in $\mathfrak{P}_{\mathfrak{M}}$.*

Proof. Straightforward. $\square$

We will use the denotation $\mathfrak{P}_{\mathfrak{M}}$ also for the per structure on $\mathfrak{M}$ extended with the standard interpretation above. Then we have that

$$\mathfrak{P}_{\mathfrak{M}} = \langle \mathcal{L}_{\mathfrak{M}}, \ \cdot, \mathcal{T}, \ (\![\ ]\!) \rangle$$

is a per structure for $\boldsymbol{\lambda 2}$.

2.5.3. THEOREM. *$\mathfrak{P}_{\mathfrak{M}}$ is a model of $\boldsymbol{\lambda}\mathbf{2}$.*

PROOF. Immmediate by Corollary 2.3.21. $\square$

We will extend $\mathfrak{P}_{\mathfrak{M}}$ to per structures for other second-order systems $\boldsymbol{\lambda}\square$ by extending the interpretation $(\![\ ]\!)$ according to a given constant valuation $\mathcal{V}$. The procedure is as follows.

2.5.4. DEFINITION. Let C be some set of constants, and $\mathcal{V} : C \to \mathcal{L}_{\mathfrak{M}}$ an interpretation of these, such that $(\mathcal{T}, \mathcal{V})$ is a constant valuation pair. Let $M \in \Lambda(C)$; say $c_1, \ldots, c_k$ are the constants appearing in M, and $\mathcal{V}(c_i) = \langle\!\langle Q_i \rangle\!\rangle$. Then $M^{\mathcal{V}}$ is obtained from M by replacing c_i by Q_i (for each i). Then define

$$(\![M]\!)^{\mathcal{V}}_{\rho} = \langle\!\langle M^{\mathcal{V}}[\vec{x} := \vec{P}] \rangle\!\rangle$$

where x_i, P_i are as in Definition 2.5.1. Now set

$$\mathfrak{P}^{\mathcal{V}}_{\mathfrak{M}} = \langle \mathcal{L}_{\mathfrak{M}},\ \cdot,\ \mathcal{T},\ \mathcal{V},\ (\![\ ]\!)^{\mathcal{V}} \rangle.$$

2.6. A Full Per Model

This section presents an application of the construction described in Section 2.4.

First of all, we can extend the $\boldsymbol{\lambda}\mathbf{T}$-model $\mathfrak{M}(\mathbb{N})$ to a model of $\boldsymbol{\lambda}\mathbf{2T}$. Of course, $\mathfrak{M}(\mathbb{N})$ is extensional and λ-closed. Now we focus on

$$\mathfrak{P}_{\mathfrak{M}(\mathbb{N})} = \langle \mathcal{L}_{\mathfrak{M}(\mathbb{N})},\ \cdot,\ \mathcal{T} \rangle.$$

As was pointed out in Definition 2.5.4, we additionally have to specify $\mathcal{V}$.

Let **true**, **false**, S^+, P^-, Z be the usual (untyped) λ-terms defining respectively true, false, successor function, predecessor function and test for zero. We use the denotation $\mathsf{if} \cdots \mathsf{then} \cdots \mathsf{else} \cdots$ for conditional terms.

The constants $\boldsymbol{R}_\sigma$ will be interpreted using a single lambda-calculus recursor.

2.6.1. DEFINITION. Using the fixed point combinator Y, define

$$\mathsf{Rec} \equiv \mathsf{Y}(\lambda rmnx.\mathsf{if}\ \mathsf{Z}\,x\ \mathsf{then}\ m\ \mathsf{else}\ n(\mathsf{P}^-x)(rmn(\mathsf{P}^-x))).$$

Then for all $M, N \in \Lambda\mathfrak{M}$ and $n \in \mathbb{N}$ one has

$$\begin{array}{rcl} \mathsf{Rec}MN\ulcorner 0 \urcorner & = & M, \\ \mathsf{Rec}MN\ulcorner n+1 \urcorner & = & N\ulcorner n \urcorner(\mathsf{Rec}MN\ulcorner n \urcorner). \end{array}$$

2.6.2. DEFINITION. (i) The interpretation $\mathcal{V}$ of $\boldsymbol{\lambda}\mathbf{2T}$-constants is as follows.

$$\begin{array}{rcl} \mathcal{V}(\boldsymbol{0}) & = & \langle\!\langle \ulcorner 0 \urcorner \rangle\!\rangle, \\ \mathcal{V}(\boldsymbol{S}) & = & \langle\!\langle \mathsf{S}^+ \rangle\!\rangle, \\ \mathcal{V}(\boldsymbol{R}_\sigma) & = & \langle\!\langle \mathsf{Rec} \rangle\!\rangle. \end{array}$$

(ii) The resulting structure is denoted by $\mathfrak{P}(\mathbb{N})$. So

$$\begin{aligned}\mathfrak{P}(\mathbb{N}) &= \mathfrak{P}^{\mathcal{V}}_{\mathfrak{M}(\mathbb{N})}\\ &= \langle \mathcal{L}_{\mathfrak{M}(\mathbb{N})},\ \cdot,\ \mathcal{T},\ \mathcal{V},\ (\!\![\ ]\!\!)^{\mathcal{V}}\rangle.\end{aligned}$$

2.6.3. Lemma. *$(\mathcal{T},\mathcal{V})$ is a constant valuation pair.*

Proof. Let ξ be a type valuation in $\mathrm{PER}(\mathcal{L}_{\mathfrak{M}(\mathbb{N})})$. Obviously $\mathcal{V}(\boldsymbol{0}) \in \mathrm{dom}[\mathbf{0}]_\xi$. Moreover note that

$$\mathrm{dom}[\mathbf{0}{\to}\mathbf{0}]_\xi = \{\langle\!\langle F\rangle\!\rangle \mid F \in \Lambda^\circ\mathfrak{M}, F \rhd_{\beta\mathfrak{M}} f \text{ for some } f:\mathbb{N}\to\mathbb{N}\}.$$

Therefore $\mathcal{V}(\boldsymbol{S}) \in \mathrm{dom}[\mathbf{0}{\to}\mathbf{0}]_\xi$. Regarding the recursors $\boldsymbol{R}_\sigma$, suppose $\langle\!\langle M\rangle\!\rangle\ [\sigma]_\xi\ \langle\!\langle M'\rangle\!\rangle$, and $\langle\!\langle N\rangle\!\rangle\ [\mathbf{0}{\to}\sigma{\to}\sigma]_\xi\ \langle\!\langle N'\rangle\!\rangle$, and prove by induction on n that

$$\langle\!\langle \mathsf{Rec}\rangle\!\rangle\langle\!\langle M\rangle\!\rangle\langle\!\langle N\rangle\!\rangle\langle\!\langle \ulcorner n\urcorner\rangle\!\rangle\ [\sigma]_\xi\ \langle\!\langle \mathsf{Rec}\rangle\!\rangle\langle\!\langle M'\rangle\!\rangle\langle\!\langle N'\rangle\!\rangle\langle\!\langle \ulcorner n\urcorner\rangle\!\rangle.$$

It follows that $\mathcal{V}(\boldsymbol{R}_\sigma) \in \mathrm{dom}[\sigma{\to}(\mathbf{0}{\to}\sigma{\to}\sigma){\to}\mathbf{0}{\to}\sigma]_\xi$. $\square$

So we can conclude that $\mathfrak{P}(\mathbb{N})$ is a per structure for $\boldsymbol{\lambda}\mathbf{2T}$.

2.6.4. Theorem. *$\mathfrak{P}(\mathbb{N})$ is a model of $\boldsymbol{\lambda}\mathbf{2T}$.*

Proof. In view of Proposition 2.3.20 we only have to check the validity of (CR). This is verified by translating the equations following Definition 2.6.1 to $\mathcal{L}_{\mathfrak{M}}$. $\square$

In Breazu-Tannen and Meyer (1987), the authors investigate plain second order extensions $(\boldsymbol{\lambda}\square)^2$ of first-order theories $\boldsymbol{\lambda}\square$ (being $\boldsymbol{\lambda}^{\boldsymbol{\tau}}$-theories over a many-sorted algebra). They show that these extensions are conservative with respect to provable equality, using (a refinement of) the property that every closed $(\boldsymbol{\lambda}\square)^2$-term of simple type has a normal form in $\boldsymbol{\lambda}\square$. As a consequence of this property, there is no increase in computational strength in the simple types of the polymorphic extension. In Section 5 of their paper, the authors mention this aspect of their construction, which they consider unfortunate.

As a second result, Breazu-Tannen and Meyer essentially show that every extensional model of a theory $\boldsymbol{\lambda}\square$ can be extended to a model of $(\boldsymbol{\lambda}\square)^2$. At first sight, the construction of e.g. our model $\mathfrak{P}^{\mathcal{V}}_{\mathfrak{M}(\mathbb{N})}$ for $\boldsymbol{\lambda}\mathbf{2T}$ seems to overlap with their approach. The situation here, however, is essentially different.

Breazu-Tannen and Meyer again use the normalization property mentioned above. This does not apply to our case: we have already seen that $\boldsymbol{\lambda}\mathbf{2T}$ is not the plain polymorphic extension of $\boldsymbol{\lambda}\mathbf{T}$; the absence of increase of computational power contrasts Corollary 2.1.11.

Moreover, the construction in Breazu-Tannen and Meyer (1987) is essentially *typed*, whereas our first-order model has to be encoded into an *untyped* applicative structure. The fact that a per construction over an untyped algebra is powerful enough is interesting in itself.

The primary motivation of our construction was to build a certain counter model, to be used in Chapter 3, and not to improve on Breazu-Tannen and Meyer (1987).

However, for our model of $\boldsymbol{\lambda 2}\mathbf{T}$ we use a construction which may be seen as an improvement of the construction of Breazu-Tannen and Meyer (1987) in that it does not suffer from the unfortunate aspect mentioned above. Of course, there is a price to be paid. First, the first-order model has to satisfy strong computational closure conditions. Second, our construction is much more involved than that in Breazu-Tannen and Meyer (1987).

Chapter 3

Bar Recursion versus Polymorphism

3.1. Syntax

In this chapter, we consider extensions of $\boldsymbol{\lambda}\mathbf{T}$ with bar recursion ($\boldsymbol{\lambda}\mathbf{TB}$) and with polymorphism ($\boldsymbol{\lambda}\mathbf{2T}$). To the systems $\boldsymbol{\lambda}\mathbf{T}$, $\boldsymbol{\lambda}\mathbf{TB}$ and $\boldsymbol{\lambda}\mathbf{2T}$ one can add quantifier-free arithmetic, which will be expressed by the denotation $\boldsymbol{\lambda}\square_{\mathbf{A}}$. With $\longrightarrow$ expressing the subsystem relation we can depict the situation as follows.

$$
\begin{tikzcd}
\boldsymbol{\lambda}^{\tau} \arrow[r] \arrow[d] & \boldsymbol{\lambda}\mathbf{T}_{(\mathbf{A})} \arrow[r] \arrow[d] & \boldsymbol{\lambda}\mathbf{TB}_{(\mathbf{A})} & \text{first order systems} \\
\boldsymbol{\lambda}\mathbf{2} \arrow[r] & \boldsymbol{\lambda}\mathbf{2T}_{(\mathbf{A})} & & \text{second order systems}
\end{tikzcd}
$$

Bar Recursion

In this chapter we only consider bar recursion of the lowest possible type. This bar recursion (of type $\mathbf{0}$) is in fact recursion on trees of finite sequences (of natural numbers). See Bezem (1986) for more details on general bar recursion.

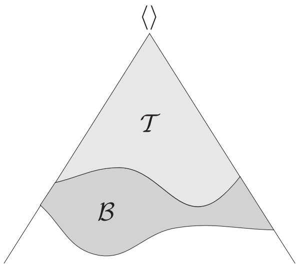
3.1.1. Definition. (i) The collection of *finite sequences* of natural numbers is indicated by Seq. We use the denotation $\langle n_0, \ldots, n_{k-1}\rangle$ ($k \geq 0$) for elements of Seq; the *empty sequence* is denoted by $\langle\,\rangle$. Let $s, s', \ldots$ range over such sequences.

(ii) We presuppose a surjective, primitive recursive *encoding* of such sequences as natural numbers, with $*$ (*concatenation* operator), lh (*length* function) and $\preceq$ (*prefix* relation) primitive recursive.

(iii) If $s = \langle n_0, \ldots, n_{k-1}\rangle$ then $[s]$ is the function assigning n_i to $i < k$ and 0 to $i \geq k$. Note that $[s]$ is primitive recursive in s.

(iv) If $f \in \mathbb{N}^{\mathbb{N}}$ and $n \in \mathbb{N}$, then $\overline{f}(n)$ is the sequence $\langle f(0), \ldots, f(n-1)\rangle$.

For an informal explanation of bar recursion, consider the tree of sequence numbers, ordered by the prefix relation.



Now suppose $\mathcal{B} \subseteq \text{Seq}$ is a *bar*, i.e.

$$\forall f \in \mathbb{N}^{\mathbb{N}} \;\; \exists n \in \mathbb{N} \;\; [\overline{f}(n) \in \mathcal{B}].$$

Then we can specify a function F on the tree

$$\mathcal{T} = \{s \in \text{Seq} \mid \forall s' \prec s \; [s' \notin \mathcal{B}]\}$$

by specifying F on the leaves of $\mathcal{T}$ (or even on all elements of $\mathcal{B}$), and defining F on each inner node s recursively in terms of all values $F(s*\langle x\rangle)$. In particular, F is defined in $\langle\,\rangle$.

Technically, F is said to be *bar recursive in G and H* if Gs gives the value of F at each leaf s, and the value in an inner node s is obtained by applying Hs to $\lambda x.F(s*\langle x\rangle)$ (all predecessors of s in $\mathcal{T}$).

In $\boldsymbol{\lambda}\mathbf{TB}$, a bar is given by a type-$\mathbf{2}$ functional Y, by setting

$$\mathcal{B}_Y = \{s \in \text{Seq} \mid Y[s] < \text{lh}(s)\}.$$

Of course each Y should be such that $\mathcal{B}_Y$ is a proper bar (i.e., the corresponding tree $\mathcal{T}$ is well-founded). This indicates that it is non-trivial to find a model of bar recursion.

Now we can formally introduce $\boldsymbol{\lambda}\mathbf{TB}$.

3.1.2. Definition. The system $\boldsymbol{\lambda}\mathbf{TB}$ is the extension of $\boldsymbol{\lambda}\mathbf{T}$ with the constant

$$\boldsymbol{B} \quad \text{of type} \quad \mathbf{2}{\to}\mathbf{1}{\to}(\mathbf{0}{\to}\mathbf{1}{\to}\mathbf{0}){\to}\mathbf{0}{\to}\mathbf{0}.$$

The intended interpretation is that of *bar recursor*. To the axioms and rules of $\boldsymbol{\lambda}\mathbf{T}$ we add the following.

$$\frac{Y[s] < \text{lh}(s)}{\boldsymbol{B}YGHs = Gs} \qquad\qquad \frac{Y[s] \geq \text{lh}(s)}{\boldsymbol{B}YGHs = Hs(\lambda x^{\mathbf{0}}.\boldsymbol{B}YGH(s*\langle x\rangle))} \tag{CB}$$

The denotations $\text{lh}(s), [s]$ and $*, \langle x\rangle$ in the above rules should be taken as abbreviations of $\boldsymbol{\lambda}\mathbf{T}$-terms representing the corresponding primitive recursive functions. The premisses can be considered equational, using the (primitive recursive) characteristic functions of $<$ and $\geq$, as will be explained below.

Quantifier-free Arithmetic

To increase the proof-theoretic usability of our theories we can extend these with rules for the successor function and for induction, and with rules for propositional logic. This leads to systems arithmetical extensions $\boldsymbol{\lambda}\Box_{\mathbf{A}}$.

3.1.3. DEFINITION. The additional rules for the structure of natural numbers are the following.

$$\frac{\boldsymbol{S}P = \boldsymbol{S}Q}{P = Q}\,\mathrm{SE} \qquad \frac{\boldsymbol{S}P = \boldsymbol{0}}{\varphi}\,\bot\mathrm{R} \qquad \frac{\varphi(\boldsymbol{0}, y) \qquad \begin{array}{c}\cancel{\varphi(x, Fy)}\\ \vdots \\ \varphi(\boldsymbol{S}x, y)\end{array}}{\varphi(P, Q)}\,\mathrm{IR} \qquad \text{(N)}$$

The conclusions in the above rules are schematic in order to avoid a general instantiation rule. It will be proved that full instantiation is derivable in the system (see Corollary 3.1.13).

Our induction rule is a little special, which we shall explain now. Firstly, the second premiss (the induction step) is casted so that it can be added to purely equational systems. This is possible because we allow derivations to depend on assumptions. Some additional care in dealing with variables should be exercised. Of course the induction rule can only be applied when no variable in P or Q occurs free in any assumption (other than the induction hypothesis $\varphi(x, Fy)$) on which the premisses depend.

Secondly, a more usual form of the induction rule is obtained by putting $F \equiv \lambda y.y$ so that $Fy = y$. Since φ may contain other variables than those explicitly shown, we can formulate this induction rule as usual:

$$\frac{\varphi(\boldsymbol{0}) \qquad \begin{array}{c}\cancel{\varphi(x)}\\ \vdots \\ \varphi(\boldsymbol{S}x)\end{array}}{\varphi(P)}\,\mathrm{IR}'$$

In the rule IR the induction hypothesis $\varphi(x, Fy)$ has as special feature the occurrence of the term F. Note that the rule is sound due to the base step $\varphi(\boldsymbol{0}, y)$ (and the fact that y does not occur free in any assumption on which the premiss depends). The reason for the special feature is that it allows us to prove conveniently in our (quantifier-free) theory some lemmas whose usual proofs are by double induction, which cannot be done in a quantifier-free system. Actually, the rule IR is a derived rule in the system with the at first sight weaker rule IR$'$. The tedious proof of this fact (essentially formalizing the semantic argument given in the proof of Theorem 3.2.1, cf. Troelstra (1973), 1.7.8–10) is avoided by postulating the stronger rule.

The ease in use of the quantifier-free arithmetic would be greatly improved by the addition of propositional logic. This should be done with care in order to allow the formulas to be translated into purely equational form. We first give a precise definition of the set Form($\boldsymbol{\lambda}\Box_{\mathbf{A}}$) of (propositional) formulas. Below, $\boldsymbol{\lambda}\Box$ ranges over the first-order theories containing $\boldsymbol{\lambda}\mathbf{T}$.

3.1.4. Definition. The set $\mathrm{Form}_\sigma(\boldsymbol{\lambda}\Box_{\mathbf{A}})$ of *formulas of type* σ is inductively defined as follows.

$$\begin{array}{rcl} M, N \in \mathrm{Term}_\sigma(\boldsymbol{\lambda}\Box) & \Rightarrow & (M =_\sigma N) \in \mathrm{Form}_\sigma(\boldsymbol{\lambda}\Box_{\mathbf{A}}), \\ \varphi, \psi \in \mathrm{Form}_{\mathbf{0}}(\boldsymbol{\lambda}\Box_{\mathbf{A}}) & \Rightarrow & (\varphi \to \psi) \in \mathrm{Form}_{\mathbf{0}}(\boldsymbol{\lambda}\Box_{\mathbf{A}}). \end{array}$$

As before, $\varphi, \psi, \chi, \varphi'$ range over formulas. To economize on parentheses, outermost parentheses are omitted and we take $\to$ to associate to the right. It is important to stress that only type $\mathbf{0}$ equations may be combined into propositional formulas (for reasons that will become clear below). We single out the formula $\overline{0} = \overline{1}$ from $\mathrm{Form}_{\mathbf{0}}(\boldsymbol{\lambda}\Box_{\mathbf{A}})$ and denote it by $\perp$.

3.1.5. Definition. The rules for *propositional logic* read as follows.

$$\dfrac{\begin{array}{c}\cancel{\varphi}\\ \vdots \\ \psi\end{array}}{\varphi \to \psi}\to\mathrm{I} \qquad\qquad \dfrac{\varphi \to \psi \qquad \varphi}{\psi}\to\mathrm{E} \tag{P}$$

3.1.6. Definition. The extension of the theory $\boldsymbol{\lambda}\Box$ with *quantifier-free arithmetic* (notation $\boldsymbol{\lambda}\Box_{\mathbf{A}}$) is obtained by adding the rules (N) and (P). The corresponding (extended) deduction relation is denoted by $\vdash_{\boldsymbol{\lambda}\Box_{\mathbf{A}}}$. Note that the induction rule now also applies to propositional formulas.

The definitions of the theories $\boldsymbol{\lambda}\Box$ are summarized in the following table. The rules L1, L2, and CR have been introduced in Section 2.1.

$\boldsymbol{\lambda}^{\boldsymbol{\tau}}$	L1					
$\boldsymbol{\lambda}\mathbf{T}_{(\mathbf{A})}$	L1		CR		(N	P)
$\boldsymbol{\lambda}\mathbf{TB}_{(\mathbf{A})}$	L1		CR	CB	(N	P)
$\boldsymbol{\lambda}\mathbf{2}$	L1	L2				
$\boldsymbol{\lambda}\mathbf{2T}_{(\mathbf{A})}$	L1	L2	CR		(N	P)

The following definitions and lemmas prepare for a translation from formulas of $\boldsymbol{\lambda}\Box_{\mathbf{A}}$ into equations while preserving provability.

3.1.7. Definition. We introduce the following abbreviations for terms.

$$\begin{array}{rcl}
\mathsf{C} & \equiv & \lambda x^{\mathbf{0}} y^{\mathbf{0}} z^{\mathbf{0}}.\boldsymbol{R}_{\mathbf{0}} z(\lambda n^{\mathbf{0}} m^{\mathbf{0}}.y)x \quad \text{(conditional)}, \\
\mathsf{P} & \equiv & \boldsymbol{R}_{\mathbf{0}}\overline{0}(\lambda x^{\mathbf{0}} y^{\mathbf{0}}.x) \quad \text{(predecessor)}, \\
\boxed{-} & \equiv & \lambda x^{\mathbf{0}} y^{\mathbf{0}}.\boldsymbol{R}_{\mathbf{0}} x(\lambda u^{\mathbf{0}} v^{\mathbf{0}}.\mathsf{P}v)y \quad \text{(cut-off subtraction)}, \\
\boxed{<} & \equiv & \lambda x^{\mathbf{0}} y^{\mathbf{0}}.\boldsymbol{R}_{\mathbf{0}}\overline{0}(\lambda u^{\mathbf{0}} v^{\mathbf{0}}.\overline{1})(y\boxed{-}x) \quad \text{(less than)}, \\
\boxed{=} & \equiv & \lambda x^{\mathbf{0}} y^{\mathbf{0}}.\boldsymbol{R}_{\mathbf{0}}(\boldsymbol{R}_{\mathbf{0}}\overline{1}(\lambda u^{\mathbf{0}} v^{\mathbf{0}}.\overline{0})(x\boxed{-}y))(\lambda u^{\mathbf{0}} v^{\mathbf{0}}.\overline{0})(y\boxed{-}x) \quad \text{(equal)}, \\
\boxed{\to} & \equiv & \lambda x^{\mathbf{0}} y^{\mathbf{0}}.\boldsymbol{R}_{\mathbf{0}}\overline{1}(\lambda u^{\mathbf{0}} v^{\mathbf{0}}.y)x \quad \text{(implication)}.
\end{array}$$

Note that primitive recursion of the lowest type suffices for these operations. Observe that we write $\boxed{-}$ as an infix operator (binding weaker than unary functions).

3.1.8. Definition. We introduce the following abbreviations for formulas.

$$\begin{array}{rcl} M \neq N & \equiv & (M = N) \to \bot, \\ M < N & \equiv & \boxed{<} MN = \overline{1}, \\ M > N & \equiv & \boxed{<} NM = \overline{1}. \end{array}$$

Furthermore, $\Gamma \vdash_{\boldsymbol{\lambda}\Box} \varphi \leftrightarrow \psi$ is shorthand for $\Gamma \vdash_{\boldsymbol{\lambda}\Box} \varphi \to \psi$ and $\Gamma \vdash_{\boldsymbol{\lambda}\Box} \psi \to \varphi$. We abbreviate $\vdash_{\boldsymbol{\lambda}\mathbf{T_A}}$ by $\vdash$.

3.1.9. Lemma. *The following can be proved in* $\boldsymbol{\lambda}\mathbf{T_A}$.

(i) $\vdash x \neq \overline{0} \to x = \boldsymbol{S}(\mathsf{P}\, x)$
(ii) $\vdash (x \neq \overline{0} \to \bot) \to x = \overline{0}$
(iii) $\vdash x \neq \overline{0} \leftrightarrow x > \overline{0}$
(iv) $\vdash \overline{0} \boxed{-} x = x$
(v) $\vdash \boldsymbol{S}M \boxed{-} x = \overline{0} \to \overline{0} < x$
(vi) $\vdash \boldsymbol{S}x \boxed{-} \boldsymbol{S}y = x \boxed{-} y$
(vii) $\vdash (x \neq y \to \bot) \to x = y$
(viii) $\vdash x \boxed{-} y = \overline{0} \to y \boxed{-} x = \overline{0} \to x = y$
(ix) $\vdash x \boxed{-} x = \overline{0}$
(x) $\vdash x = y \leftrightarrow \boxed{=} xy = \overline{1}$
(xi) $\vdash x \neq y \leftrightarrow \boxed{=} xy = \overline{0}$
(xii) $\vdash \boxed{\to} (\boldsymbol{S}x) y = y$
(xiii) $\vdash \boxed{\to} \overline{0} y = \overline{1}$
(xiv) $\vdash \boxed{\to} xy = \overline{0} \to x \neq \overline{0}$
(xv) $\vdash \boxed{\to} xy = \overline{0} \to y = \overline{0}$
(xvi) $\vdash \boxed{\to} xy = \overline{1} \to y \neq \overline{1} \to x = \overline{0}$
(xvii) $y \neq \overline{0} \to y = \overline{1} \vdash \boxed{\to} xy \neq \overline{0} \to \boxed{\to} xy = \overline{1}$

Proof. The proofs are more or less standard and can be found in the literature (cf. Troelstra (1973), 1.7.2–7, where this development of quantifier-free arithmetic is attributed to K. Schütte). We leave them as exercises to the reader, but not without providing a number of hints. Most proofs use the induction rule IR as the essential step. However, not all proofs use IR in the same way. We distinguish between the following cases. First consider the case in which the special feature of our induction rule is not used, so in fact only the weaker rule IR$'$ is applied. If no induction hypothesis is used at all, then we actually do only case distinction (cases $\varphi(\boldsymbol{0})$ and $\varphi(\boldsymbol{S}x)$) and will phrase this accordingly. If we do use the induction hypothesis $\varphi(x)$, then we phrase the case as 'by ordinary induction'. Next consider the case that the force of the induction rule with parameters is indeed used. Then we phrase the proof as 'by induction' and mention the induction hypothesis explicitly. Moreover we mention which variable is the induction variable and which previous parts of the lemma can be profitably used. Hints for the proofs: (i)–(v) by case distinction on x; (vi) by ordinary induction on y; (vii) by induction on y using the induction hypothesis $(\mathsf{P}\, x \neq y \to \bot) \to \mathsf{P}\, x = y$ and part (ii); (viii) by induction on x using the induction hypothesis $x \boxed{-} \mathsf{P}\, y = \overline{0} \to \mathsf{P}\, y \boxed{-} x = \overline{0} \to x = \mathsf{P}\, y$ and the parts (i), (iii) and (v); (ix) by ordinary induction on x using (vi); (x),(xi) by previous parts (not using the induction rule at all); (xii)–(xiv) trivial; (xv)–(xvii) by case distinction on x. $\Box$

3.1.10. Definition. We define a mapping

$$\varphi \mapsto \boxed{\varphi} \quad : \quad \mathrm{Form}_{\mathbf{0}}(\boldsymbol{\lambda}\Box_{\mathbf{A}}) \to \mathrm{Term}_{\mathbf{0}}(\boldsymbol{\lambda}\Box_{\mathbf{A}})$$

by induction on the structure of φ as follows.

$$\begin{array}{rcl} \boxed{M =_{\mathbf{0}} N} & \equiv & \boxed{=}MN, \\ \boxed{\varphi \to \psi} & \equiv & \boxed{\to}\boxed{\varphi}\boxed{\psi}. \end{array}$$

3.1.11. Theorem. *For all* $\varphi \in \mathrm{Form}_{\mathbf{0}}(\boldsymbol{\lambda}\Box_{\mathbf{A}})$ *we have*

(i) $\mathrm{FV}(\varphi) = \mathrm{FV}(\boxed{\varphi})$;

(ii) $\vdash_{\boldsymbol{\lambda}\mathbf{T}_{\mathbf{A}}} \boxed{\varphi} \neq \overline{0} \to \boxed{\varphi} = \overline{1}$.

(iii) $\vdash_{\boldsymbol{\lambda}\mathbf{T}_{\mathbf{A}}} \varphi \leftrightarrow \boxed{\varphi} = \overline{1}$.

Proof. (i) is obvious. (ii) and (iii) are proved by induction on φ, using Lemma 3.1.9. $\Box$

3.1.12. Corollary. $\boldsymbol{\lambda}\Box_{\mathbf{A}}$ *enjoys the full force of classical propositional logic.*

Proof. By Theorem 3.1.11 (iii) and Lemma 3.1.9 (vii). $\Box$

3.1.13. Corollary. *If the variable x does not occur free in Γ, then*

$$\Gamma \vdash_{\boldsymbol{\lambda}\Box_{\mathbf{A}}} \varphi \;\Rightarrow\; \Gamma \vdash_{\boldsymbol{\lambda}\Box_{\mathbf{A}}} \varphi[x := N].$$

Proof. First assume φ is an equation, say $P = Q$. Since x does not occur in Γ it follows by the ξ-rule that $\lambda x.P = \lambda x.Q$. Now we calculate $P[x := N] = (\lambda x.P)N = (\lambda x.Q)N = Q[x := N]$, so $\varphi[x := N]$. If φ is not an equation, then apply Theorem 3.1.11 (iii) and the fact that $\boxed{\varphi[x := N]} \equiv \boxed{\varphi}[x := N]$. $\Box$

3.1.14. Remark. Theorem 3.1.11 above cannot be generalized to formulas $M =_\sigma N$ with $\sigma \neq \mathbf{0}$. The reason is that the higher type equalities are undecidable, whereas equality of type $\mathbf{0}$ is decidable (even primitive recursive). As a consequence, adding propositional logic for equations of arbitrary type would violate the equational character of the theory.

3.1.15. Remark. It is useful to remark that for all $\varphi \in \mathrm{Form}_{\mathbf{0}}(\boldsymbol{\lambda}\Box_{\mathbf{A}})$ there exist $\boldsymbol{\lambda}\Box$-terms $\mu x{\le}y.\ \varphi$, $\forall x{\le}y.\ \varphi$, $\exists x{\le}y.\ \varphi$ encoding, respectively, bounded minimization, bounded universal quantification and bounded existential quantification. More precisely, these terms satisfy, provably in $\boldsymbol{\lambda}\mathbf{T}_{\mathbf{A}}$, the following specifications.

$$\begin{array}{rcl} (\mu x{\le}y.\ \varphi) < \boldsymbol{S}y & \to & \varphi[x := (\mu x{\le}y.\ \varphi)], \\ x < (\mu x{\le}y.\ \varphi) & \to & \varphi \to \perp, \\ (\exists x{\le}y.\ \varphi) = \overline{1} & \to & (\mu x{\le}y.\ \varphi) < \boldsymbol{S}y, \\ (\exists x{\le}y.\ \varphi) = \overline{0} & \to & x \le y \to \varphi \to \perp, \\ (\forall x{\le}y.\ \varphi) = \overline{1} & \to & x \le y \to \varphi, \\ (\forall x{\le}y.\ \varphi) = \overline{0} & \to & (\mu x{\le}y.\ \varphi \to \perp) < \boldsymbol{S}y. \end{array}$$

Our formulation of $\boldsymbol{\lambda}\mathbf{TB}_{\mathbf{A}}$ corresponds to that of Howard (1968). In order to transfer our results to purely equational systems (i.e., without (P)) one should work with the encoded propositional formulas $\boxed{\varphi}$, cf. Spector (1962). This requires an additional conservativity result for the propositional extension, which goes back to Goodstein (1941).

3.2. Semantics

We do not need models for $\boldsymbol{\lambda}\mathbf{TB}$ in this chapter, since the (positive) results concerning that system can be proved syntactically. One usually considers the type structure of extensional continuous functionals (introduced in Kleene (1959a) as 'countable functionals', and in Kreisel (1959)) as the standard model. Another model is obtained by taking the strongly majorizable functionals (introduced by Bezem (1985)), a variant of the heriditarily majorizable functionals from Howard (1973).

We can use the $\boldsymbol{\lambda}\mathbf{T}$-model $\mathfrak{M}(\mathbb{N})$ and the $\boldsymbol{\lambda}\mathbf{2T}$-model $\mathfrak{P}(\mathbb{N})$ also for the extended theories. This will be verified now.

3.2.1. THEOREM. *$\mathfrak{M}(\mathbb{N})$ is a model of $\boldsymbol{\lambda}\mathbf{T_A}$.*

PROOF. The proof of Corollary 2.2.8 (ii) can be extended as follows. The rules (SE) and ($\perp$R) are sound since $[\![\boldsymbol{S}]\!]$ is the successor function and $\mathfrak{M}(\mathbb{N})$ is an ω-model.

As to the induction rule, suppose

$$\mathfrak{M}(\mathbb{N}), \rho \models \varphi(\boldsymbol{0}, y) \tag{1}$$

and

$$\mathfrak{M}(\mathbb{N}), \rho \models \varphi(x, Fy) \ \Rightarrow\ \mathfrak{M}(\mathbb{N}), \rho \models \varphi(\boldsymbol{S}x, y) \tag{2}$$

in order to show

$$\mathfrak{M}(\mathbb{N}), \rho \models \varphi(x, Q).$$

Now reason in $\mathfrak{M}(\mathbb{N})$. Let $n \in \mathbb{N}$. For each $k \leq n$ one has

$$\varphi(n-k, F^k(y)) \ \Rightarrow\ \varphi(n, y)$$

by (2) and induction on k. Hence (choosing $k = n$)

$$\varphi(0, F^k(y)) \ \Rightarrow\ \varphi(n, y).$$

Now by (1) we are done.

Finally, the propositional rules (P) are obviously sound. □

3.2.2. THEOREM. *$\mathfrak{P}(\mathbb{N})$ is a model of $\boldsymbol{\lambda}\mathbf{2T_A}$.*

PROOF. By extension of the proof of Theorem 2.6.4 with the following observations. The rules in (N) are satisfied since $\mathrm{dom}[\mathbf{0}] \cong \mathbb{N}$ and S^+ represents the 'real' successor function; the rule (IR) is verified as in the proof above. Moreover, the propositional extension is obviously sound. □

By a similar argument one can show the following.

3.2.3. THEOREM. HEO2 *is a model of $\boldsymbol{\lambda}\mathbf{2T_A}$.*

It is convenient to formulate a notion of ω-model for arbitrary models of theories containing $\boldsymbol{\lambda}\mathbf{T}$.

3.2.4. DEFINITION. Let $\mathfrak{M}$ be a first-order (or second-order) model of (an extension of) $\boldsymbol{\lambda}\mathbf{T}$. Then $\mathfrak{M}$ is said to be an *ω-model* if

$$\langle [\![\mathbf{0}]\!],\ [\![\boldsymbol{S}]\!],\ [\![\boldsymbol{0}]\!] \rangle \cong \langle \mathbb{N},\ S,\ 0 \rangle.$$

Obviously, the models $\mathfrak{M}(\mathbb{N})$, $\mathfrak{P}(\mathbb{N})$ and HEO2 are ω-models.

3.3. Realizability of Functional Specifications

The metamathematical results from Spector (1962) and Girard (1972) imply that $\boldsymbol{\lambda}\mathbf{TB}$ and $\boldsymbol{\lambda}\mathbf{2T}$ are equally powerful with respect to definability of functions on the natural numbers: the definable functions (on the numerals of type $\mathbf{0}$) are exactly those that are provably total in analysis.

It is not obvious how to compare the above theories with respect to definability in higher types. In the above situation it is clear what the domain of the functions in question should be: the set of natural numbers. Due to the variety of models (containing at type $\mathbf{1}$, e.g., all numerical functions ($\mathfrak{M}(\mathbb{N})$), or just the computable ones (HEO)) it is not clear how to specify a class of type-$\mathbf{2}$ functionals for investigation of definability.

Here we choose for a *syntactical* specification of functions and functionals in the common language of $\boldsymbol{\lambda}\mathbf{TB}$ and $\boldsymbol{\lambda}\mathbf{2T}$, namely the language of $\boldsymbol{\lambda}\mathbf{T}$. The notion of definability is replaced by the concept of realizability of $\boldsymbol{\lambda}\mathbf{T}$-specifications, defined in *semantical* terms.

We will show that the realizability concept is not vacuous: every computable function on the natural numbers can be specified in $\boldsymbol{\lambda}\mathbf{T}$. Then we reformulate the above result about definability of functions in terms of realizability of specifications.

3.3.1. Definition. Let $\sigma \in \mathbb{T}_1$. A *σ-specification* is a formula $\Sigma^\sigma \equiv \Sigma(f^\sigma)$ in the language of $\boldsymbol{\lambda}\mathbf{T}_\mathbf{A}$, containing one free variable of type σ (the *main variable*) and possibly containing other free variables of lower types (the *auxiliary variables*). We write $\Sigma^\sigma \equiv \Sigma(f^\sigma)$, or $\Sigma(f^\sigma, \vec{x})$ if we want to display the auxiliary variables explicitly.

3.3.2. Definition. (i) Let ψ be a formula. By $\boldsymbol{\lambda}\Box \models_\omega \psi$ we denote that ψ is *true in all ω-models* of $\boldsymbol{\lambda}\Box$.

(ii) Let $\boldsymbol{\lambda}\Box$ be an extension of $\boldsymbol{\lambda}\mathbf{T}$, and let $\Sigma(f^\sigma, \vec{x})$ be a σ-specification. Then Σ is said to be *realizable* in $\boldsymbol{\lambda}\Box$ if there exists $F \in \mathrm{Term}_\sigma(\boldsymbol{\lambda}\Box)$ such that

$$\boldsymbol{\lambda}\Box \models_\omega \Sigma(F, \vec{x}),$$

i.e. $\forall \vec{x}\ \Sigma(F, \vec{x})$ holds in all ω-models of $\boldsymbol{\lambda}\Box$.

The following is the traditional syntactical notion of definability at type $\mathbf{1}$.

3.3.3. Definition. Let $F \in \mathrm{Term}_\mathbf{1}(\boldsymbol{\lambda}\Box)$, and $f : \mathbb{N} \to \mathbb{N}$. Then F is said to $\boldsymbol{\lambda}\Box$-*define* f if for all $n \in \mathbb{N}$

$$\boldsymbol{\lambda}\Box \vdash F\overline{n} =_\mathbf{0} \overline{f(n)}.$$

3.3.4. Proposition. *Each computable function f on $\mathbb{N}$ has a $\mathbf{1}$-specification, i.e. a specification Σ that is satisfied only by type-$\mathbf{1}$ objects behaving like f.*

Proof. Let T be the computation predicate and U the result extracting function from Kleene's Normal Form Theorem, such that

$$\{e\}(x) \simeq U(\mu z.T(e, x, z)).$$

Remember that T and U are primitive recursive and hence expressible in $\boldsymbol{\lambda}\mathbf{T}$. Now let $f : \mathbb{N} \to \mathbb{N}$ be computable, say with index e. Now take

$$\begin{aligned} \Sigma_e^\mathbf{1}(f^\mathbf{1}) &\equiv T(\overline{e}, x, z) \to f^\mathbf{1}x =_\mathbf{0} Uz \\ &\equiv \Sigma_e^\mathbf{1}(f^\mathbf{1}, x^\mathbf{0}, z^\mathbf{0}). \end{aligned}$$

Obviously, this completely caputures the behaviour of f. $\Box$

In particular, F is a term realizing the above Σ in $\boldsymbol{\lambda}\square$ (i.e. $\boldsymbol{\lambda}\square \models_\omega \Sigma_e(F,x,z)$) iff F $\boldsymbol{\lambda}\square$-defines f.

3.3.5. Definition. Let $\boldsymbol{\lambda}\square$ and $\boldsymbol{\lambda}\boxtimes$ be theories.

(i) By $\boldsymbol{\lambda}\square \preccurlyeq^\sigma \boldsymbol{\lambda}\boxtimes$ we denote that every $\boldsymbol{\lambda}\square$-realizable σ-specification is realizable in $\boldsymbol{\lambda}\boxtimes$.

(ii) $\boldsymbol{\lambda}\square$ and $\boldsymbol{\lambda}\boxtimes$ are *realization equivalent at type* σ (notation $\boldsymbol{\lambda}\square \approx^\sigma \boldsymbol{\lambda}\boxtimes$) if both $\boldsymbol{\lambda}\square \preccurlyeq^\sigma \boldsymbol{\lambda}\boxtimes$ and $\boldsymbol{\lambda}\boxtimes \preccurlyeq^\sigma \boldsymbol{\lambda}\square$.

Now we can translate the results from Spector (1962) and Girard (1972) into a realization property. First we need some technicalities.

3.3.6. Definition. Let $f:\mathbb{N}\to\mathbb{N}$. Then $\boldsymbol{\lambda}\mathbf{T}\boldsymbol{f}$ is the theory obtained by extending $\boldsymbol{\lambda}\mathbf{T}$ with a constant $\boldsymbol{f}$ and axioms

$$\boldsymbol{f}\,\overline{n} = \overline{f(n)}. \qquad (f)$$

We use $\to$ (and $\twoheadrightarrow$) to refer to the (multistep) *reduction relations* corresponding to the equalities in the theories $\boldsymbol{\lambda}\mathbf{T}$ and $\boldsymbol{\lambda}\mathbf{T}\boldsymbol{f}$.

3.3.7. Lemma. (i) $\to$ *is Church-Rosser on* $\mathrm{Term}(\boldsymbol{\lambda}\mathbf{T}\boldsymbol{f})$.

(ii) $\to$ *is strongly normalizing.*

(iii) *Each closed* $\boldsymbol{\lambda}\mathbf{T}\boldsymbol{f}$*-term of type* $\mathbf{0}$ *reduces to a unique numeral.*

Proof. (i) By the observation that the contraction rules for $\boldsymbol{f}$ ($\boldsymbol{f}\,\overline{n} \to \overline{f(n)}$) define a notion of δ-reduction (see Barendregt (1984), Theorem 15.3.3) and the fact that $\boldsymbol{\lambda}\mathbf{T}$ is Church-Rosser (folklore).

(ii) Straightforward adaptation of Tait's computability argument (see Tait (1967), cf. Barendregt (1984), Theorem A.2.3).

(iii) We show that every closed normal form P of type $\mathbf{0}$ is a numeral. Then we are done by (i) and (ii). We proceed by induction on the length of normal forms. Note that every normal form has the shape $\lambda\vec{x}.aM_1\cdots M_k$, with a atomic, i.e. a variable or a constant, and the M_i again in normal form. Since P has type $\mathbf{0}$ the abstraction $\lambda\vec{x}$ is empty, and a is a constant.

If $a\equiv\boldsymbol{0}$ then we are done.

If $a\equiv\boldsymbol{S}$ then $P\equiv\boldsymbol{S}Q$ for some closed normal form Q of type $\mathbf{0}$, so the induction hypothesis applies.

If $a\equiv\boldsymbol{f}$ then $P\equiv\boldsymbol{f}Q$ for some Q. But then P is a redex by induction hypothesis, contradiction.

If $a\equiv\boldsymbol{R}_\sigma$ then $\sigma\equiv\mathbf{0}$ and $P\equiv\boldsymbol{R}_{\mathbf{0}}MNQ$ with Q closed and of type $\mathbf{0}$. Then again P would be a redex, contradiction. $\square$

3.3.8. Lemma. *Let* $F\in\mathrm{Term}_{\mathbf{1}}(\boldsymbol{\lambda}\square)$*, and* $f:\mathbb{N}\to\mathbb{N}$*. Suppose* f *is* $\boldsymbol{\lambda}\square$*-defined by* F*. Let* $\Sigma^{\mathbf{1}}\equiv\Sigma(f^{\mathbf{1}},\vec{x}^{\mathbf{0}})$ *be a* $\mathbf{1}$*-specification. Then*

$$\boldsymbol{\lambda}\mathbf{T}\boldsymbol{f} \models_\omega \forall\vec{x}\,\Sigma(\boldsymbol{f},\vec{x}) \;\Leftrightarrow\; \boldsymbol{\lambda}\square \models_\omega \forall\vec{x}\,\Sigma(F,\vec{x}).$$

Proof. Note that each ω-model of $\boldsymbol{\lambda}\square$ is a model of $\boldsymbol{\lambda}\square_{\mathbf{A}}$. By Theorem 3.1.11, we can write $\Sigma^{\mathbf{1}}(f,\vec{x})$ as $Gf\vec{x} =_{\mathbf{0}} \overline{1}$ for some closed $\boldsymbol{\lambda}\mathbf{T}$-term G. Assume F $\boldsymbol{\lambda}\square$-defines f. Observe that each ω-model of $\boldsymbol{\lambda}\square$ is also a model of $\boldsymbol{\lambda}\mathbf{T}\boldsymbol{f}$, with $[\![\boldsymbol{f}]\!]=[\![F]\!]$ ('$=f$') by extensionality.

($\Rightarrow$) Suppose $\boldsymbol{\lambda}\mathbf{T}\boldsymbol{f} \models_\omega \forall\vec{x}\,\Sigma(\boldsymbol{f},\vec{x})$. Then for all $\vec{n} \in \mathbb{N}$

$$\begin{aligned} [\![GF\overline{\vec{n}}]\!] &= [\![G]\!]\,[\![F]\!]\,[\![\overline{\vec{n}}]\!] \\ &= [\![G]\!]\,[\![\boldsymbol{f}]\!]\,[\![\overline{\vec{n}}]\!] \\ &= [\![\overline{1}]\!] \end{aligned}$$

and we are done by the above observation.

($\Leftarrow$) Suppose $\boldsymbol{\lambda}\mathbf{T}\boldsymbol{f} \not\models_\omega \forall\vec{x}\,\Sigma(\boldsymbol{f},\vec{x})$. Then $[\![G\boldsymbol{f}\overline{\vec{n}}]\!] = [\![\overline{p}]\!]$ in some ω-model of $\boldsymbol{\lambda}\mathbf{T}\boldsymbol{f}$, for some $\vec{n}, p \in \mathbb{N}$ with $p \neq 1$. Note that $G\boldsymbol{f}\overline{\vec{n}}$ is a closed term of type $\mathbf{0}$. Therefore $G\boldsymbol{f}\overline{\vec{n}} \twoheadrightarrow \overline{p}$ by Lemma 3.3.7 (iii). This reduction sequence can be transformed into a $\boldsymbol{\lambda}\Box$-derivation of $GF\overline{\vec{n}} =_{\mathbf{0}} \overline{p}$, using a subsidiary derivation of $F\overline{k} =_{\mathbf{0}} \overline{f(k)}$ for each reduction step $\boldsymbol{f}\,\overline{k} \rightarrow \overline{f(k)}$. Hence $[\![GF\overline{\vec{n}}]\!] \neq [\![\overline{1}]\!]$, so $\boldsymbol{\lambda}\Box \not\models_\omega \forall\vec{x}\,\Sigma(F,\vec{x})$. $\Box$

3.3.9. Theorem. $\boldsymbol{\lambda}\mathbf{TB_A} \approx^{\mathbf{1}} \boldsymbol{\lambda}\mathbf{2T_A}$.

Proof. By Lemma 3.3.8, using the fact that $\boldsymbol{\lambda}\mathbf{TB_A}$ and $\boldsymbol{\lambda}\mathbf{2T_A}$ define the same f's, by the results of Spector (1962) and Girard (1972). $\Box$

3.4. First-order Non-conservativity

In this section we will consider a $\mathbf{3}$-specification $\mathbf{\Sigma} \equiv \mathbf{\Sigma}(\Phi^{\mathbf{3}})$ and show that this specification is realizable in $\boldsymbol{\lambda}\mathbf{TB_A}$, but not in $\boldsymbol{\lambda}\mathbf{T_A}$. (In the next section we will show that even in $\boldsymbol{\lambda}\mathbf{2T}$ the specification is not realizable.)

The positive result for $\boldsymbol{\lambda}\mathbf{TB_A}$ will be proved by a purely syntactical method. With a little extra effort one can use the specification $\mathbf{\Sigma}$ to construct a formula ψ in the language of $\boldsymbol{\lambda}\mathbf{T_A}$ such that

$$\begin{gathered} \boldsymbol{\lambda}\mathbf{TB_A} \vdash \psi, \\ \boldsymbol{\lambda}\mathbf{T_A} \nvdash \psi, \end{gathered}$$

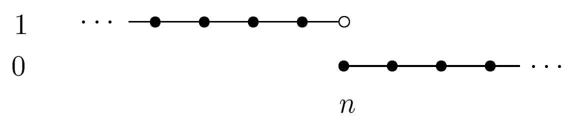
thus showing that $\boldsymbol{\lambda}\mathbf{TB_A}$ is a non-conservative extension of $\boldsymbol{\lambda}\mathbf{T_A}$. This does not really come as a surprise since it follows from Spector (1962) that in the system $\boldsymbol{\lambda}\mathbf{TB_A}$ the consistency of Peano Arithmetic, i.e. $\neg\mathrm{Proof}_{\mathbf{PA}}(x, \ulcorner 0 = 1 \urcorner)$, can be proved (after suitable encoding).

The formula presented here, however, does not refer to any metamathematical properties and can be formulated in a simple way.

3.4.1. Definition. The term $\mathbf{\Delta} \in \mathrm{Term}_{\mathbf{0}\to\mathbf{0}\to\mathbf{0}}(\boldsymbol{\lambda}\mathbf{T})$ is defined by

$$\mathbf{\Delta} \equiv \lambda xy.\mathsf{if}\ y < x\ \mathsf{then}\ \overline{1}\ \mathsf{else}\ \overline{0}.$$

The intuition is that $\mathbf{\Delta}\boldsymbol{n}$ is a so-called *stepfunction* that can be depicted as follows.



3.4.2. Definition. The $\mathbf{3}$-specification $\mathbf{\Sigma} \equiv \mathbf{\Sigma}(\Phi)$, with Φ a type-$\mathbf{3}$ variable, is defined by

$$\mathbf{\Sigma}(\Phi) \;\equiv\; [x < \Phi\varphi \rightarrow \varphi(\mathbf{\Delta}x) \geq x \quad \wedge \quad \varphi(\mathbf{\Delta}(\Phi\varphi)) < \Phi\varphi].$$

Here φ is a type-$\mathbf{2}$ variable and x a type-$\mathbf{0}$ variable. Note that $\mathbf{\Sigma}(\Phi)$ can be written in the form $G\Phi\varphi x = \overline{0}$ or $G\Phi = \lambda\varphi x.\overline{0}$.

3.4.3. Intuition. If Φ satisfies $\mathbf{\Sigma}(\Phi)$ then Φ performs the following minimization.

$$\Phi\varphi \simeq \mu x.\ \varphi(\mathbf{\Delta}x) < x.$$

This minimization is only well defined for φ such that there exists x with $\varphi(\mathbf{\Delta}x) < x$. Since $\mathbf{\Sigma}$ will be shown to have a bar recursive solution Φ, the minimization is well defined for φ taken from a model of bar recursion, such as the countable/continuous functionals, or the (strongly) majorizable functionals. A counter model against $\mathbf{\Sigma}(\Phi)$ must contain a φ for which the minimization is not well defined. Since every continuous functional of type $\mathbf{2}$ is also majorizable, any counter model must contain a non-majorizable functional φ. This is not very problematic for counter models satisfying $\boldsymbol{\lambda}\mathbf{T}_\mathbf{A}$ but turned out to be difficult for counter models satisfying $\boldsymbol{\lambda}\mathbf{2T}_\mathbf{A}$, since all 'standard' models of $\boldsymbol{\lambda}\mathbf{2T}_\mathbf{A}$ seem to exhibit an inherent continuity.

3.4.4. Definition. Let $\boldsymbol{\lambda}\square$, $\boldsymbol{\lambda}\boxtimes$ be two of our systems. We say that $\mathbf{\Sigma}$ is $\boldsymbol{\lambda}\square$-*solvable* in $\boldsymbol{\lambda}\boxtimes$ if for some $\Phi \in \mathrm{Term}_\mathbf{3}(\boldsymbol{\lambda}\square)$ one has

$$\boldsymbol{\lambda}\boxtimes \vdash \mathbf{\Sigma}(\Phi).$$

Obviously, if $\mathbf{\Sigma}$ is $\boldsymbol{\lambda}\square$-solvable in $\boldsymbol{\lambda}\square$ then $\mathbf{\Sigma}$ is realizable in $\boldsymbol{\lambda}\square$. We first show that $\mathbf{\Sigma}$ is not realizable in $\boldsymbol{\lambda}\mathbf{T}_\mathbf{A}$ (so *a fortiori* $\mathbf{\Sigma}$ is not $\boldsymbol{\lambda}\mathbf{T}$-solvable in $\boldsymbol{\lambda}\mathbf{T}_\mathbf{A}$).

3.4.5. Proposition. *There exists no* $\Phi \in \mathrm{Term}_\mathbf{3}(\boldsymbol{\lambda}\mathbf{T})$ *such that*

$$\boldsymbol{\lambda}\mathbf{T}_\mathbf{A} \models_\omega \mathbf{\Sigma}(\Phi).$$

Proof. Suppose, towards a contradiction, $\boldsymbol{\lambda}\mathbf{T}_\mathbf{A} \models_\omega \mathbf{\Sigma}(\Phi)$ with $\Phi \in \mathrm{Term}_\mathbf{3}(\boldsymbol{\lambda}\mathbf{T})$. Consider the ω-model $\mathfrak{M}(\mathbb{N})$. Define $\delta \in \mathbb{N}_{\mathbf{0}\to\mathbf{0}\to\mathbf{0}}$ by

$$\begin{aligned} \delta(n)(x) &= 1 \qquad \text{if } x < n, \\ &= 0 \qquad \text{if } x \geq n. \end{aligned}$$

Note that $[\![\mathbf{\Delta}]\!] = \delta$; denotation: $\delta_n = \delta(n)$. Take $\varphi \in \mathbb{N}_\mathbf{2}$ such that for each $n \in \mathbb{N}$

$$\varphi(\delta_n) = n.$$

Then one has

$$\varphi(\delta([\![\Phi]\!]\varphi)) = [\![\Phi]\!]\varphi,$$

contradicting $\mathfrak{M}(\mathbb{N}) \models \mathbf{\Sigma}(\Phi)$. $\square$

It can easily be seen that $\mathbf{\Sigma}$ is $\boldsymbol{\lambda}\mathbf{TB}$-solvable in $\boldsymbol{\lambda}\mathbf{TB_A}$. Indeed, the minimization stated above can be obtained by constructing a 'searching functional' Ψ of type $\mathbf{2}{\to}\mathbf{0}{\to}\mathbf{0}$ such that

$$\begin{array}{rcll} \Psi\varphi s & = & 0 & \text{if } \varphi[s] < \mathrm{lh}(s), \\ & = & 1 + \Psi\varphi(s * \langle 1\rangle) & \text{otherwise,} \end{array}$$

and finally defining $\Phi \equiv \lambda\varphi.\Psi\varphi\langle\,\rangle$. From the form of the equations for Ψ it can be expected that such a Ψ can be constructed using the bar recursor.

3.4.6. Definition. The term $\mathsf{F}_{\mathrm{B}} \in \mathrm{Term}_{\mathbf{3}}(\boldsymbol{\lambda}\mathbf{TB})$ is defined as follows.

$$\mathsf{F}_{\mathrm{B}} \equiv \lambda\varphi.\boldsymbol{B}\varphi GH\langle\,\rangle,$$

where

$$\begin{array}{rcl} G & \equiv & \lambda\sigma.\overline{0}, \\ H & \equiv & \lambda sf.\boldsymbol{S}(f\overline{1}). \end{array}$$

In order to show that F_{B} is a solution for $\mathbf{\Sigma}$ we reason informally in $\boldsymbol{\lambda}\mathbf{TB_A}$; the argument can easily be formalized.

3.4.7. Definition. Let $P(x,\varphi)$ abbreviate

$$\begin{array}{rcl} x < \mathsf{F}_{\mathrm{B}}\varphi & \to & (\varphi(\boldsymbol{\Delta}x) \geq x \quad \wedge \\ & & \quad \mathsf{F}_{\mathrm{B}}\varphi = x + 1 + \boldsymbol{B}\varphi GH1^{x+1}), \end{array}$$

where $1^x = \langle\underbrace{1,1,\ldots,1}_{x \text{ times}}\rangle$.

3.4.8. Lemma. $\boldsymbol{\lambda}\mathbf{TB_A} \vdash P(x,\varphi)$.

Proof. By induction on x. Note that $[1^x] = \boldsymbol{\Delta}x$, by extensionality.

Basis. $P(0,\varphi)$ holds since trivially $\varphi(\boldsymbol{\Delta}0) \geq 0$ and $\mathsf{F}_{\mathrm{B}}\varphi > 0$ implies $\varphi[\langle\,\rangle] \geq 0 = \mathrm{lh}(\langle\,\rangle)$ and therefore

$$\begin{array}{rcl} \mathsf{F}_{\mathrm{B}}\varphi & = & H\langle\,\rangle(\lambda y.\boldsymbol{B}\varphi GH\langle y\rangle) \\ & = & 1 + \boldsymbol{B}\varphi GH\langle 1\rangle. \end{array}$$

Induction step. Suppose $P(x,\varphi)$ in order to show $P(x{+}1,\varphi)$. Suppose $x{+}1 < \mathsf{F}_{\mathrm{B}}\varphi$. Then $x < \mathsf{F}_{\mathrm{B}}\varphi$ so by induction hypothesis

$$\mathsf{F}_{\mathrm{B}}\varphi = x + 1 + \boldsymbol{B}\varphi GH1^{x+1}.$$

We claim that $\varphi(\boldsymbol{\Delta}(x+1)) \geq x+1$. Indeed, suppose $\varphi(\boldsymbol{\Delta}(x+1)) < x+1$. Then $\varphi[1^{x+1}] < x+1$, so $\boldsymbol{B}\varphi GH1^{x+1} = 0$, so $\mathsf{F}_{\mathrm{B}}\varphi = x+1$, contradicting $\mathsf{F}_{\mathrm{B}}\varphi > x+1$. Now it follows that

$$\begin{array}{rcl} \boldsymbol{B}\varphi GH1^{x+1} & = & H1^{x+1}(\lambda y.\boldsymbol{B}\varphi GH(1^{x+1} * \langle y\rangle)) \\ & = & 1 + \boldsymbol{B}\varphi GH1^{x+2}, \end{array}$$

so

$$\mathsf{F}_{\mathrm{B}}\varphi = x + 2 + \boldsymbol{B}\varphi GH1^{x+2},$$

which completes the induction step. $\square$

3.4.9. PROPOSITION. $\boldsymbol{\lambda}\mathbf{TB_A} \vdash \boldsymbol{\Sigma}(\mathsf{F}_{\mathrm{B}})$.

PROOF. Observe that $\mathsf{F}_{\mathrm{B}}\varphi > 0$. By Lemma 3.4.8 one has

$$P(\mathsf{F}_{\mathrm{B}}\varphi - 1, \varphi),$$

and therefore

$$\mathsf{F}_{\mathrm{B}}\varphi = \mathsf{F}_{\mathrm{B}}\varphi + \boldsymbol{B}\varphi GH1^{\mathsf{F}_{\mathrm{B}}\varphi},$$

so

$$\boldsymbol{B}\varphi GH1^{\mathsf{F}_{\mathrm{B}}\varphi} = 0,$$

which implies

$$\begin{array}{rcl} \varphi(\boldsymbol{\Delta}(\mathsf{F}_{\mathrm{B}}\varphi)) & = & \varphi[1^{\mathsf{F}_{\mathrm{B}}\varphi}] \\ & < & \mathrm{lh}(1^{\mathsf{F}_{\mathrm{B}}\varphi}) \\ & = & \mathsf{F}_{\mathrm{B}}\varphi. \end{array}$$

Combining this with $P(x, \varphi)$ gives

$$\boldsymbol{\Sigma}(\mathsf{F}_{\mathrm{B}}). \quad \square$$

3.4.10. COROLLARY. $\boldsymbol{\lambda}\mathbf{TB_A} \not\preccurlyeq^{\mathbf{3}} \boldsymbol{\lambda}\mathbf{T_A}$.

PROOF. By the propositions 3.4.5 and 3.4.9. $\square$

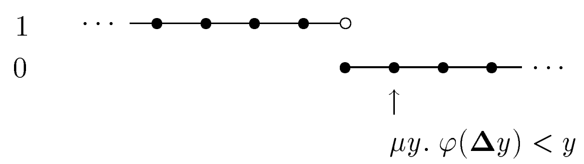
The above corollary is just preparing for the results in the next section. In fact one already has $\boldsymbol{\lambda}\mathbf{TB_A} \not\preccurlyeq^{\mathbf{1}} \boldsymbol{\lambda}\mathbf{T_A}$, by the metamathematical results of Spector (1962) and Gödel (1958).

It will turn out that $\boldsymbol{\Sigma}$ is even $\boldsymbol{\lambda}\mathbf{T}$-solvable in $\boldsymbol{\lambda}\mathbf{TB_A}$. Use is made of a trick due to G. Kreisel, also employed by J.A. Bergstra and J. Terlouw, see Barendregt (1984), p. 581.

3.4.11. DEFINITION. Define $f_\varphi \in \mathrm{Term}_{\mathbf{1}}(\boldsymbol{\lambda}\mathbf{T})$ by

$$f_\varphi \equiv \lambda x.\mathsf{if}\ \forall y{\le}x{+}1.\ \varphi(\boldsymbol{\Delta}y) \geq y\ \mathsf{then}\ \overline{1}\ \mathsf{else}\ \overline{0}.$$

3.4.12. INTUITION. f_φ is a stepfunction which looks as follows.



So $f_\varphi = \boldsymbol{\Delta}((\mu y.\ \varphi(\boldsymbol{\Delta}y) < y) - 1)$ (provided $\exists y\ \varphi(\boldsymbol{\Delta}y) < y$; otherwise '$f_\varphi = \boldsymbol{\Delta}_\infty$'). Therefore

$$\varphi(f_\varphi) \geq (\mu y.\ \varphi(\boldsymbol{\Delta}y) < y) - 1$$

and hence

$$(\mu y.\ \varphi(\boldsymbol{\Delta}y) < y) \ \leq\ \varphi(f_\varphi) + 1.$$

This gives a primitive recursive upperbound of the minimization expressed in $\boldsymbol{\Sigma}$.

3.4.13. Definition. The term $\mathsf{F}_{\mathrm{T}} \in \mathrm{Term}_{\mathbf{3}}(\boldsymbol{\lambda}\mathbf{T})$ is defined as follows.

$$\mathsf{F}_{\mathrm{T}} \equiv \lambda\varphi.\ \mu x{\leq}\varphi(f_\varphi){+}1.\ \varphi(\boldsymbol{\Delta}x) < x.$$

3.4.14. Proposition. $\boldsymbol{\lambda}\mathbf{TB}_{\mathbf{A}} \vdash \mathsf{F}_{\mathrm{T}} = \mathsf{F}_{\mathrm{B}}$.

Proof. Again we reason informally in $\boldsymbol{\lambda}\mathbf{TB}_{\mathbf{A}}$. From Proposition 3.4.9 we know $\boldsymbol{\Sigma}(\mathsf{F}_{\mathrm{B}})$, so

$$x < \mathsf{F}_{\mathrm{B}}\varphi \to \varphi(\boldsymbol{\Delta}x) \geq x, \tag{1}$$

$$\varphi(\boldsymbol{\Delta}(\mathsf{F}_{\mathrm{B}}\varphi)) < \mathsf{F}_{\mathrm{B}}\varphi. \tag{2}$$

From (1) it follows that

$$\forall y \leq \mathsf{F}_{\mathrm{B}}\varphi - 1. \quad \varphi(\boldsymbol{\Delta}y) \geq y \tag{3}$$

and therefore

$$x < \mathsf{F}_{\mathrm{B}}\varphi - 1 \to f_\varphi x = 1, \tag{4}$$

and

$$x \geq \mathsf{F}_{\mathrm{B}}\varphi - 1 \to f_\varphi x = 0. \tag{5}$$

From (4) and (5) one obtains by extensionality

$$f_\varphi = \boldsymbol{\Delta}(\mathsf{F}_{\mathrm{B}}\varphi - 1).$$

Hence again by (1)

$$\varphi(f_\varphi) \geq \mathsf{F}_{\mathrm{B}}\varphi - 1$$

and therefore

$$\mathsf{F}_{\mathrm{B}}\varphi \leq \varphi(f_\varphi) + 1.$$

Combining this with (2) and again (1) (ensuring the minimality of $\mathsf{F}_{\mathrm{B}}\varphi$ with respect to $\varphi(\boldsymbol{\Delta}x) < x$) yields

$$\mathsf{F}_{\mathrm{T}}\varphi = \mathsf{F}_{\mathrm{B}}\varphi$$

and hence by extensionality

$$\mathsf{F}_{\mathrm{T}} = \mathsf{F}_{\mathrm{B}}.\ \square$$

3.4.15. Corollary. $\boldsymbol{\lambda}\mathbf{TB}_{\mathbf{A}} \vdash \boldsymbol{\Sigma}(\mathsf{F}_{\mathrm{T}})$.

Proof. By the propositions 3.4.9 and 3.4.14. $\square$

Now we can formulate the non-conservativity result obtained in this section.

3.4.16. Theorem. *$\boldsymbol{\lambda}\mathbf{TB}_{\mathbf{A}}$ is a non-conservative extension of $\boldsymbol{\lambda}\mathbf{T}_{\mathbf{A}}$.*

Proof. Note that $\boldsymbol{\Sigma}(\mathsf{F}_{\mathrm{T}})$ is an equation in the language of $\boldsymbol{\lambda}\mathbf{T}_{\mathbf{A}}$. By Proposition 3.4.5

$$\boldsymbol{\lambda}\mathbf{T}_{\mathbf{A}} \nvdash \boldsymbol{\Sigma}(\mathsf{F}_{\mathrm{T}}),$$

whereas by Proposition 3.4.15

$$\boldsymbol{\lambda}\mathbf{TB}_{\mathbf{A}} \vdash \boldsymbol{\Sigma}(\mathsf{F}_{\mathrm{T}}).\ \square$$

It is important to stress that our results have little to do with proof theoretic strength in the usual sense. Luckhardt (1975) presents a system $\boldsymbol{\lambda}\mathbf{T}+\boldsymbol{\mu}$ which has the same proof theoretic strength as Heyting's Arithmetic. Our specification $\boldsymbol{\Sigma}$ is in fact a special case of the defining equation for $\boldsymbol{\mu}$. It is not difficult to prove that $\vdash_{\boldsymbol{\lambda}\mathbf{T}+\boldsymbol{\mu}} \boldsymbol{\Sigma}(\mathsf{F}_{\mathrm{T}})$. As a consequence we can complement the results from Luckhardt (1975) with the following theorem.

3.4.17. THEOREM. *$\boldsymbol{\lambda}\mathbf{T}+\boldsymbol{\mu}$ is a non-conservative extension of $\boldsymbol{\lambda}\mathbf{T}_{\mathbf{A}}$.*

Similar results can be obtained for the extension of $\boldsymbol{\lambda}\mathbf{T}$ with a modulus of uniform continuity (a so-called *fan functional*).

3.5. Non-realizability in Second-order Lambda Calculus

In this section it will be shown that $\boldsymbol{\Sigma}$ is not even realizable in $\boldsymbol{\lambda}\mathbf{2T}_{\mathbf{A}}$. This suggests that bar recursion is, in some sense, a more powerful extension of $\boldsymbol{\lambda}\mathbf{T}$ than the concept of polymorphism (see the discussion in the introduction).

The idea of the proof is the same as the proof for $\boldsymbol{\lambda}\mathbf{T}_{\mathbf{A}}$ (Proposition 3.4.5), but the implementation is far more difficult. The presence of a type-**2** functional like φ, introducing a 'fatal discontinuity', would solve the problem.

A first attempt would be to extend Kleene's partial applicative structure with codes for an oracle function, thus obtaining a relative version of the model HEO2. A naive solution like adding an oracle φ such that

$$\begin{aligned}\varphi(e) &= n && \text{if } \{e\}=\delta_n,\\ &= 0 && \text{otherwise}\end{aligned}$$

fails since an index of such a φ will get lost in the PER-construction. (Let $\{\cdot\}^{\varphi}$ denote the relativized version of $\{\cdot\}$. There will be e_1, e_2 such that $\{e_1\}=\delta_n$ and $\{e_2\}\neq\delta_n$ for some $n>0$ but $\{e_1\}^{\varphi}=\{e_2\}^{\varphi}$; φ acts differently on e_1 and e_2.) This is not just a technical accident but has a fundamental reason: one can show that the Kreisel-Lacombe-Schoenfield theorem (see Kreisel et al. (1959)), expressing that all type-**2** elements of HEO2 are continuous, can be relativized. Hence every recursion theoretic oracle is doomed to fail. This indicates the necessity of 'impredicative oracles'. Other well-known model constructions for $\boldsymbol{\lambda}\mathbf{2}$ and $\boldsymbol{\lambda}\mathbf{2T}$ seem to exhibit an inherent continuity.

Using the construction of Section 2.6 we can prove an analogue of Proposition 3.4.5.

3.5.1. PROPOSITION. *There exists no $\Phi\in\mathrm{Term}_{\mathbf{3}}(\boldsymbol{\lambda}\mathbf{2T})$ such that*

$$\boldsymbol{\lambda}\mathbf{2T}_{\mathbf{A}} \models_{\omega} \boldsymbol{\Sigma}(\Phi).$$

PROOF. Suppose $\boldsymbol{\lambda}\mathbf{2T}_{\mathbf{A}} \models_{\omega} \boldsymbol{\Sigma}(\Phi)$. We want to derive a contradiction. We consider the model $\mathfrak{P}(\mathbb{N})$. Take $\varphi:\mathbb{N}^{\mathbb{N}}\to\mathbb{N}$ such that $\varphi(\delta_n)=n$ (like in the proof of Proposition 3.4.5). Construct $\mathsf{D}\in\Lambda^{\circ}\mathfrak{M}$ such that

$$\begin{aligned}\mathsf{D}\ulcorner n\urcorner\ulcorner m\urcorner &= \ulcorner 1\urcorner && \text{if } m<n,\\ &= \ulcorner 0\urcorner && \text{otherwise.}\end{aligned}$$

Note that $\langle\!\langle \mathsf{D} \rangle\!\rangle \in \mathrm{dom}[\mathbf{0}\to\mathbf{0}\to\mathbf{0}]_\xi$; more specifically, $\langle\!\langle \mathsf{D} \rangle\!\rangle \in [\![\boldsymbol{\Delta}]\!]$. Moreover, by the Classification Theorem 2.4.8 (i) one has $\langle\!\langle \boldsymbol{\varphi} \rangle\!\rangle \in \mathrm{dom}[(\mathbf{0}\to\mathbf{0})\to\mathbf{0}]_\xi$, and

$$\langle\!\langle \boldsymbol{\varphi} \rangle\!\rangle(\langle\!\langle \mathsf{D} \rangle\!\rangle([\Phi]\langle\!\langle \boldsymbol{\varphi} \rangle\!\rangle)) = [\Phi]\langle\!\langle \boldsymbol{\varphi} \rangle\!\rangle.$$

This implies

$$[\varphi(\boldsymbol{\Delta}(\Phi))]_{\rho(\varphi:=\langle\!\langle \boldsymbol{\varphi} \rangle\!\rangle)}\ [\mathbf{0}]\ [\Phi\varphi]_{\rho(\varphi:=\langle\!\langle \boldsymbol{\varphi} \rangle\!\rangle)},$$

contradicting $\mathfrak{P}(\mathbb{N}) \models \boldsymbol{\Sigma}(\Phi)$. Note that the Isomorphism Theorem 2.4.9 guarantees that distinct natural numbers correspond to distinct model elements of type $\mathbf{0}$. $\Box$

3.5.2. Corollary. $\boldsymbol{\lambda}\mathbf{TB_A} \not\preccurlyeq^{\mathbf{3}} \boldsymbol{\lambda}\mathbf{2T_A}$.

It is open whether $\boldsymbol{\lambda}\mathbf{2T_A} \preccurlyeq^{\mathbf{3}} \boldsymbol{\lambda}\mathbf{TB_A}$ holds. Moreover the type-**2** case is still unsolved.

The original proof of Proposition 3.5.1 in Barendsen and Bezem (1991) used essentially the same technique but in a more restricted way (aiming just at the non-realizability result above). This involved a per model over untyped λ-calculus with one single oracle (namely for the above φ), with a reduction relation $\to_\varphi$ such that

$$\boldsymbol{\varphi} F \to_\varphi \ulcorner\varphi(f)\urcorner \qquad \text{if} \qquad F \rhd_{\beta\varphi} f.$$

See Barendsen and Bezem (1992) for details.

3.6. Extensions to Higher-order Lambda Calculus

It is possible to extend our results to higher-order lambda calculi. In these ω-order systems one considers *type constructors*. These include all types, but also functions from types to types, like $\lambda\alpha.\alpha\to\alpha$, functionals on these, and so on. In this section we briefly describe the systems and our results.

The (extensions of) the ω-order system $\boldsymbol{\lambda\omega}$ can be described using the notion of *kind*. Kinds are the 'types of constructors'. First choose a constant $*$; the intended meaning of $*$ is the collection of all types. Now the kinds of $\boldsymbol{\lambda\omega}$ are defined by the following abstract syntax.

$$\mathbb{K} = * \mid \mathbb{K}\to\mathbb{K}.$$

Notice the similarity with the *types* of $\boldsymbol{\lambda}^{\boldsymbol{\tau}}$.

For each $\kappa \in \mathbb{K}$, the *constructors* of kind κ (notation Constr_κ) are defined as follows. One easily recognizes the types in $\mathbb{T}_2$ as a subcollection of Constr_*. For each $\kappa \in \mathbb{K}$, let CVar_κ be a set of *constructor variables*.

$$\begin{array}{rcl}
\alpha^\kappa \in \mathrm{CVar}_\kappa & \Rightarrow & \alpha^\kappa \in \mathrm{Constr}_\kappa, \\
C \in \mathbb{C} & \Rightarrow & C \in \mathrm{Constr}_*, \\
\sigma, \tau \in \mathrm{Constr}_* & \Rightarrow & (\sigma\to\tau) \in \mathrm{Constr}_*, \\
\sigma \in \mathrm{Constr}_*,\ \alpha^\kappa \in \mathrm{CVar}_\kappa & \Rightarrow & (\forall\alpha^\kappa.\sigma) \in \mathrm{Constr}_*, \\
\gamma \in \mathrm{Constr}_{\kappa_1\to\kappa_2},\ \delta \in \mathrm{Constr}_{\kappa_1} & \Rightarrow & (\gamma\delta) \in \mathrm{Constr}_{\kappa_2}, \\
\alpha^{\kappa_1} \in \mathrm{CVar}_{\kappa_1},\ \gamma \in \mathrm{Constr}_{\kappa_2} & \Rightarrow & (\lambda\alpha^{\kappa_1}.\gamma) \in \mathrm{Constr}_{\kappa_1\to\kappa_2}.
\end{array}$$

Then, e.g., $\lambda\alpha^*.\alpha\to\alpha \in \mathrm{Constr}_{*\to*}$.

Let $\boldsymbol{\lambda}\Box$ range over the ω-order systems. Now the collection of *terms* of $\boldsymbol{\lambda}\Box$ (inhabitants of types $\sigma \in \mathrm{Constr}_*$) can be defined. For a proper syntactic treatment one is forced to consider variable *contexts* instead of annotated term variables. We refrain from going into the details of this. The system $\boldsymbol{\lambda\omega}$ is the plain ω-order calculus (without term constants), and $\boldsymbol{\lambda\omega}\mathbf{T}$ is the ω-order variant of $\boldsymbol{\lambda}\mathbf{T}$.

The per semantics of the second-order systems (in a per structure $\mathfrak{P} = \langle A, \cdot, \mathcal{T}\rangle$) can be extended to ω-order systems, as follows. The higher-order constructors are interpreted in the full type structure over $\mathrm{PER}(A)$, by setting

$$\begin{aligned} [*] &= \mathrm{PER}(A), \\ [\kappa_1 \to \kappa_2] &= [\kappa_2]^{[\kappa_1]}. \end{aligned}$$

Moreover, elements of Constr_* are interpreted by extending the per interpretation of types into $\mathrm{PER}(A)$ described in Definition 2.3.6. In particular,

$$[\forall \alpha^\kappa . \sigma]_\xi = \bigwedge_{F \in [\kappa]} [\sigma]_{\xi(\alpha^\kappa := F)}.$$

Without proof we mention the following.

3.6.1. Theorem. (i) *Let $\mathfrak{P}$ be a per structure. Then $\mathfrak{P}$ is a model of $\boldsymbol{\lambda\omega}$.*
(ii) *$\mathfrak{P}(\mathbb{N})$ is a model of $\boldsymbol{\lambda\omega}\mathbf{T}_\mathbf{A}$.*

3.6.2. Corollary. $\boldsymbol{\lambda}\mathbf{TB}_\mathbf{A} \not\preccurlyeq^\mathbf{3} \boldsymbol{\lambda\omega}\mathbf{T}_\mathbf{A}$.

Proof. Analogous to the proof of Corollary 3.5.2. $\Box$

Bibliography

Barendregt, H.P. (1984). *The Lambda Calculus: Its Syntax and Semantics*, second, revised edition, North-Holland, Amsterdam.

Barendsen, E. (1990). Semantics of simply typed lambda calculus, Typescript, University of Nijmegen.

Barendsen, E. (1991). An unsolvable numeral system in lambda calculus, *Journal of Functional Programming* **1**, pp. 367–372.

Barendsen, E. and M.A. Bezem (1991). Bar recursion versus polymorphism (extended abstract), *Technical Report 69*, Logic Group, Utrecht Research Institute for Philosophy, Utrecht University.

Barendsen, E. and M.A. Bezem (1992). Bar recursion versus polymorphism, *Technical Report 81*, Logic Group, Utrecht Research Institute for Philosophy, Utrecht University.

Bezem, M.A. (1985). Strongly majorizable functionals of finite type: A model for bar recursion containing discontinuous functionals, *Journal of Symbolic Logic* **50**, pp. 652–660.

Bezem, M.A. (1986). *Bar Recursion and Functionals of Finite Type*, Dissertation, Utrecht University.

Breazu-Tannen, V. and Th. Coquand (1988). Extensional models for polymorphism, *Theoretical Computer Science* **59**, pp. 85–114.

Breazu-Tannen, V. and A. Meyer (1987). Polymorphism is conservative over simple types, *Proceedings of the 2nd Annual Symposium on Logic in Computer Science*, Ithaca, New York, IEEE Computer Society Press, pp. 7–17.

Bruce, K.B., A. Meyer and J.C. Mitchell (1990). The semantics of second-order lambda calculus, *Information and Computation* **85**, pp. 76–134.

Church, A. (1940). A formulation of the simple theory of types, *Journal of Symbolic Logic* **5**, pp. 56–68.

van Draanen, J.P. (1989). *Lambda-definability and partial recursiveness in higher types*, Master's thesis, Department of Computing Science, University of Nijmegen.

Fortune, S., D. Leivant and M. O'Donnell (1983). The expressiveness of simple and second-order type structures, *Journal of the Association for Computing Machinery* **30**, pp. 151–185.

Friedman, H. (1975). Equality between functionals, *in:* R. Parikh (ed.), *Logic Colloquium: Symposium on Logic held at Boston*, Lecture Notes in Mathematics 453, Springer-Verlag, Berlin, pp. 22–37.

Girard, J.-Y. (1972). *Interprétation fonctionelle et élimination des coupures dans l'arithétique d'ordre supérieur*, Dissertation, Université Paris VII.

Girard, J.-Y., P. Taylor and Y. Lafont (1989). *Proofs and Types*, Cambridge Tracts in Theoretical Computer Science 7, Cambridge University Press.

Gödel, K. (1958). Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes, *Dialectica* **12**, pp. 280–287.

Goodstein, R.L. (1941). Function theory in an axiom-free equation calculus, *Proceedings of the London Mathematical Society* **48**, pp. 401–434.

Heyting, A. (ed.) (1959). *Constructivity in Mathematics*, North-Holland, Amsterdam.

Hindley, J.R. (1964). *The Church-Rosser Property and a Result in Combinatory Logic*, Dissertation, University of Newcastle-upon-Tyne.

Hinman, P.G. (1978). *Recursion-theoretic Hierarchies*, North-Holland, Amsterdam.

Howard, W.A. (1968). Functional interpretation of bar induction by bar recursion, *Compositio Mathematica* **20**, pp. 107–124.

Howard, W.A. (1973). Hereditarily majorizable functionals of finite type, *in:* Troelstra (1973), pp. 455–459.

Jäger, G. and Th. Strahm (1994). Totality in applicative theories, To appear in: Annals of Pure and Applied Logic.

Kleene, S.C. (1959a). Countable functionals, *in:* Heyting (1959), pp. 81–100.

Kleene, S.C. (1959b). Recursive functionals and quantifiers of finite type I, *Transactions of the American Mathematical Society* **91**, pp. 1–52.

Kleene, S.C. (1962). Lambda-definable functionals of finite types, *Fundamenta Mathematicae* **L**, pp. 281–303.

Kreisel, G. (1959). Interpretation of analysis by means of constructive functionals of finite type, *in:* Heyting (1959), pp. 101–128.

Kreisel, G., D. Lacombe and J.R. Schoenfield (1959). Partial recursive functionals and effective operations, *in:* Heyting (1959), pp. 290–297.

Luckhardt, H. (1975). The real elements in a consistency proof for simple type theory, *in:* J. Diller and G.H. Müller (eds.), *Proof Theory Symposium Kiel 1974*, Lecture Notes in Mathematics 500, Springer-Verlag, Berlin, pp. 233–256.

Poll, E. (1994). *A Programming Logic Based on Type Theory*, Dissertation, Eindhoven University of Technology.

Reynolds, J.C. (1984). Polymorphism is not set-theoretic, *in:* G. Kahn, D.B. McQueen and G. Plotkin (eds.), *Semantics of Data Types: international symposium*, Sophia-Antipolis, France, Lecture Notes in Computer Science 173, Springer-Verlag, Berlin, pp. 145–156.

Rosen, B.K. (1973). Tree manipulation systems and Church-Rosser theorems, *Journal of the Association for Computing Machinery* **20**, pp. 160–187.

Spector, C. (1962). Provably recursive functionals of analysis: A consistency proof of analysis by an extension of principles formulated in current intuitionistic mathematics, *in:* J.C.E. Dekker (ed.), *Recursive Function Theory*, Proceedings of Symposia in Pure Mathematics V, American Mathematical Society, Providence, pp. 1–27.

Tait, W. (1967). Intensional interpretations of functionals of finite type I, *Journal of Symbolic Logic* **32**, pp. 198–212.

Troelstra, A.S. (ed.) (1973). *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, Lecture Notes in Mathematics 344, Springer-Verlag, Berlin.

Erik Barendsen
Computing Science Institute
Nijmegen University
Toernooiveld 1
6525 ED Nijmegen
The Netherlands
e-mail `erikb@cs.kun.nl`
fax +31.80.652525

Marc Bezem
Department of Philosophy
Utrecht University
Heidelberglaan 8
3584 CS Utrecht
The Netherlands
e-mail `bezem@phil.ruu.nl`
fax +31.30.532816