TYPED LAMBDA CALCULUS

Volume I: $\lambda^{\mathbb{A}}_{\rightarrow}$ $\lambda^{\mathcal{A}}_{=}$ $\lambda^{\mathcal{S}}_{\cap}$

Henk Barendregt

Wil Dekkers

Richard Statman

Version: February 21, 2008

Preface

DRAFT. DO NOT DISTRIBUTE.

Bla bla. yghbkjhbkn.
And we love our women.

Nijmegen and Pittsburgh February 21, 2008

Henk Barendregt[1,2]
Wil Dekkers[1]
Rick Statman[2]

[1]Faculty of Mathematics and Computer Science
Catholic University
Nijmegen, The Netherlands
[2]Departments of Mathematics and Computer Science
Carnegie Mellon University
Pittsburgh, USA

Overture

This book about typed lambda terms comes in two volumes: the present one about lambda terms typed using *simple*, *recursive* and *intersection* types and a planned second volume about *higher order*, *dependent* and *inductive* types.

In some sense this book is a sequel to Barendregt [1984]. That book is about untyped lambda calculus. Types give the untyped terms more structure: function applications are allowed only in some cases. In this way one can single out untyped terms having special properties. But there is more to it. The extra structure makes the theory of typed terms quite different from the untyped ones.

The emphasis of the book is on syntax. Models are introduced only in so far they give useful information about terms and types or if the theory can be applied to them.

The writing of the book has been different from that about the untyped lambda calculus. First of all, since many researchers are working on typed lambda calculus, we were aiming at a moving target. Also there was a wealth of material to work with. For these reasons the book has been written by several authors. Several long-term open problems had been solved during the interval these volumes were written, notably the undecidability of lambda definability in finite models, the undecidability of second order typability, the decidability of the unique maximal theory extending $\beta\boldsymbol{\eta}$-conversion and the fact that the collection of closed terms of not every simple type is finitely generated. (One of the remaining open problems is the decidability of matching at arbitrary types higher than order 4.) The book is not written as an encyclopedic volume: many topics are only partially treated. For example reducibility among types is analized for simple types built up from only one atom.

One of the recurring distinctions made in the two volumes is the difference between the implicit typing due to Curry versus the explicit typing due to Church. In the latter case the terms are an enhanced version of the untyped terms, whereas in the Curry theory to some of the untyped terms a collection of types is being assigned. Volume I is mainly about Curry typing, although Part I of it we also treat the for simple types equivalent Church variant in parallel.

The applications of the theory are either within the theory itself, in the theory of programming languages, in proof theory, including the technology of fully formalized proofs used for mechanical verification, or in linguistics. Often the applications are given in an exercise with hints.

We hope that the volumes will inspire readers to pursue the topic.

Contributors

Fabio Alessi	Part III, except §16.3
Henk Barendregt	All parts, except §§5.3, 5.4, 5.5
Marc Bezem	§§5.3, 5.4, 5.5
Felice Cardone	Part II
Mario Coppo	Part II
Wil Dekkers	Parts II, III
Mariangiola Dezani-Ciancaglini	Part III, except §16.3
Gilles Dowek	Chapter 4
Silvia Ghilezan	§6.2
Furio Honsell	Part III, except §16.3
Michael Moortgat	§6.4
Paula Severi	Part III
Richard Statman	Parts I, II
Paweł Urzyczyn	§16.3

Contents in short

Contents

Part I

Simple types $\lambda_{\rightarrow}$ 21.2.2008:897

Lambda calculus is worth studying, because it is a simple formal system that provides a model of computation with a distinctive quality. At first its expressions represent both a program and its data which on their turn evolve by simple rules (β and η-reduction) to instantaneous descriptions of intermediate computational results possibly ending in output. These features, however, are also present in Term Rewriting Systems. Lambda calculus has two extra qualities setting it apart from these other Rewriting Systems. Firstly it is applicative, in that an expression may be applied to another expression, giving them the possibility to act both as function and as argument. Secondly, Lambda Calculus has abstraction built in, meaning that the expressions are closed under explicit definitions.

Lambda calculus as model of computation can be introduced in an untyped fashion: arbitrary expressions may be applied to each other. A similarly flat approach to computing was present in the early assembly languages. Later in imperative programming languages types were introduced to keep order among the code. Exactly the same use of types happened even earlier with lambda calculus as model of computation. Types from a very partial specification of what a program does. In this sense types are somewhat comparable to dimensions in physics that provide—on the basis of meaning—an order in the wealth of quantitative notations. There are certain basic dimensions, like g (gram), m (meter) and s (second) and other dimensions can be expressed using these.

The systems of *simple types* considered in Part I are built up from atomic types $\mathbb{A}$ using as only operator the constructor $\to$ of forming function spaces. For example, from the atoms $\mathbb{A} = \{\alpha, \beta\}$ one can form types $\alpha{\to}\beta$, $(\alpha{\to}\beta){\to}\alpha$, $\alpha{\to}(\alpha{\to}\beta)$ and so on. Two choices of the set of atoms will be made most often are $\mathbb{A} = \{\alpha_0, \alpha_1, \alpha_2, \ldots\}$, an infinite set of type variables, and $\mathbb{A} = \{o\}$, consisting of only one atomic type. Particular atomic types that occur in applications are e.g. Bool, Nat, Real. Even for these simple type systems, the ordering effect is quite powerful.

As a short anthology of what is going to come in Part I we state the following. For an untyped lambda term one can find the collection of its possible types. Similarly, given a simple type, one can find the collection of its possible inhabitants (in normal form). Equality of terms of a certain type can be reduced to equality of terms in a fixed type. The problem of unification

$$\exists X{:}A.MX =_{\beta\boldsymbol{\eta}} NX$$

is for complex enough A undecidable. That of pattern matching

$$\exists X{:}A.MX =_{\beta\boldsymbol{\eta}} N$$

will turn out to be decidable for A simple enough. It is open if this also holds for arbitrary types. The terms of finite type are extended by δ-functions, functionals for primitive and bar recursion. Applications of the theory in computing, proof-checking and linguistics will be discussed.

Chapter 1

The systems $\lambda_{\to}$

1.1. The $\lambda_{\to}$ systems *à la* Curry

Types in this part are syntactic objects built from atomic types using the operator $\to$. In order to classify untyped lambda terms, such types will be assigned to a subset of these terms. The main idea is that if M gets type $A \to B$ and N gets type A, then the application MN is 'legal' (as M is considered as a function from terms of type A to those of type B) and gets type B. In this way types help determining what terms fit together.

1.1.1. Definition. (i) Let $\mathbb{A}$ be a non-empty set of 'atomic types'. The set of *simple types over* $\mathbb{A}$, notation $\mathbb{T} = \mathbb{T}^{\mathbb{A}}$, is inductively defined as follows.

$$\begin{array}{rcll} \alpha \in \mathbb{A} & \Rightarrow & \alpha \in \mathbb{T} & \text{type atoms;} \\ A, B \in \mathbb{T} & \Rightarrow & (A \to B) \in \mathbb{T} & \text{function space types.} \end{array}$$

Such definitions will be used often and for these it is convenient to use the so called *abstract syntax*, see Waite and Goos [1984]. As an example we give the abstract syntax for $\mathbb{T} = \mathbb{T}^{\mathbb{A}}$.

$$\boxed{\mathbb{T} \quad = \quad \mathbb{A} \mid \mathbb{T} \to \mathbb{T}}$$

Figure 1.1: Simple types

(ii) Let $\mathbb{A}_o = \{o\}$. Then we write $\mathbb{T}^o = \mathbb{T}_{\mathbb{A}_o}$.
(iii) Let $\mathbb{A}_\infty = \{\alpha_0, \alpha_1, \alpha_2, \ldots\}$. Then we write $\mathbb{T}_\infty = \mathbb{T}_{\mathbb{A}_\infty}$

We consider that $o = \alpha_0$, hence $\mathbb{T}^o \subseteq \mathbb{T}_\infty$. If we write simply $\mathbb{T}$, then this refers to $\mathbb{T}^{\mathbb{A}}$ for an unspecified $\mathbb{A}$.

1.1.2. Notation. (i) If $A_1, \ldots, A_n \in \mathbb{T}$, then

$$A_1 \to \ldots \to A_n \equiv (A_1 \to (A_2 \to \ldots \to (A_{n-1} \to A_n)..)).$$

That is, we use association to the right (here $\equiv$ denotes syntactic equality).
(ii) $\alpha, \beta, \gamma, \ldots$ denote arbitrary elements of $\mathbb{A}$.
(iii) $A, B, C, \ldots$ denote arbitrary elements of $\mathbb{T}$.

Remember the untyped lambda calculus denoted by λ, see e.g. B[1984][1]. It consists of a set of terms Λ defined by the following abstract syntax.

$$\begin{array}{lcl} \mathsf{V} & = & x \,|\, \mathsf{V}' \\ \Lambda & = & \mathsf{V} \,|\, \lambda \mathsf{V}\, \Lambda \,|\, \Lambda\, \Lambda \end{array}$$

Figure 1.2: Untyped lambda terms

This makes $\mathsf{V} = \{x, x', x'', \ldots\} = \{x_0, x_1, x_2, \ldots\}$.

1.1.3. Notation. (i) $x, y, z, \ldots$ denote arbitrary term variables.
(ii) $M, N, L, \ldots$ denote arbitrary lambda terms.
(iii) $MN_1 \ldots N_k \equiv (..(MN_1) \ldots N_k)$.
(iv) $\lambda x_1 \ldots x_n.M \equiv (\lambda x_1(..(\lambda x_n(M))..))$.

1.1.4. Definition. On Λ the following equational theory $\boldsymbol{\lambda\beta\eta}$ is defined by the usual equality axiom and rules (reflexivity, symmetry, transitivity, congruence), inluding congruence with respect to abstraction:

$$M = N \;\Rightarrow\; \lambda x.M = \lambda x.N,$$

and the following special axiom(schemes)

$$\begin{array}{rcll} (\lambda x.M)N & = & M[x := N] & (\beta\text{-rule}) \\ \lambda x.Mx & = & M, \quad \text{if } x \notin \mathrm{FV}(M) & (\eta\text{-rule}) \end{array}$$

Figure 1.3: The theory $\boldsymbol{\lambda\beta\eta}$

As is know this theory can be analyzed by a notion of reduction.

1.1.5. Definition. On Λ we define the following notions of reduction

$$\begin{array}{rcll} (\lambda x.M)N & \to & M[x{:=}N] & (\boldsymbol{\beta}) \\ \lambda x.Mx & \to & M, \quad \text{if } x \notin \mathrm{FV}(M) & (\boldsymbol{\eta}) \end{array}$$

Figure 1.4: $\beta\boldsymbol{\eta}$-contraction rules

As usual, see B[1984], these notions of reduction generate the corresponding reduction relations $\to_\beta, \twoheadrightarrow_\beta, \to_{\boldsymbol{\eta}}, \twoheadrightarrow_{\boldsymbol{\eta}}, \to_{\boldsymbol{\beta\eta}}$ and $\twoheadrightarrow_{\boldsymbol{\beta\eta}}$. Also there are the corresponding conversion relations $=_\beta, =_{\boldsymbol{\eta}}$ and $=_{\beta\boldsymbol{\eta}}$. Terms in Λ will often be considered modulo $=_\beta$ or $=_{\beta\boldsymbol{\eta}}$. If we write $M = N$, then we mean $M =_{\beta\boldsymbol{\eta}} N$ by default. (In B[1984] the default was $=_\beta$.)

1.1.6. Proposition. *For all $M, N \in \Lambda$ one has*

$$\vdash_{\boldsymbol{\lambda\beta\eta}} M = N \iff M =_{\beta\boldsymbol{\eta}} N.$$

Proof. See B[1984], Proposition 3.3.2. ■

One reason why the analysis in terms of the notion of reduction $\beta\boldsymbol{\eta}$ is useful is that the following holds.

[1] This is an abbreviation fo the reference Barendregt [1984].

1.1.7. THEOREM (Church-Rosser Theorem for $\lambda\beta\boldsymbol{\eta}$). *For the notions of reduction $\twoheadrightarrow_\beta$ and $\twoheadrightarrow_{\beta\boldsymbol{\eta}}$ one has the following.*

(i) *Let $M, N \in \Lambda$. Then*

$$M =_{\beta(\eta)} N \;\Rightarrow\; \exists Z \in \Lambda. M \twoheadrightarrow_{\beta(\eta)} Z \;\&\; N \twoheadrightarrow_{\beta(\eta)} Z.$$

(ii) *Let $M, N_1, N_2 \in \Lambda$. Then*

$$M \twoheadrightarrow_{\beta(\eta)} N_1 \;\&\; M \twoheadrightarrow_{\beta(\eta)} N_2 \;\Rightarrow\; \exists Z \in \Lambda. N_1 \twoheadrightarrow_{\beta(\eta)} Z \;\&\; N_2 \twoheadrightarrow_{\beta(\eta)} Z.$$

PROOF. (i) See Theorems 3.2.8 and 3.3.9 in B[1984].
(ii) By (i). ■

1.1.8. DEFINITION ($\lambda^{\mathrm{Cu}}_{\rightarrow}$). (i) A *(type assignment) statement* is of the form

$$M : A,$$

with $M \in \Lambda$ and $A \in \mathbb{T}$. This statement is pronounced as 'M in A'. The type A is the *predicate* and the term M is the *subject* of the statement.
(ii) A *declaration* is a statement with as subject a term variable.
(iii) A *basis* is a set of declarations with distinct variables as subjects.
(iv) A statement $M{:}A$ is *derivable from* a basis Γ, notation

$$\Gamma \vdash^{\mathrm{Cu}}_{\lambda_{\rightarrow}} M{:}A$$

(or $\Gamma \vdash_{\lambda_{\rightarrow}} M : A$, $\Gamma \vdash^{\mathrm{Cu}} M : A$ or even $\Gamma \vdash M{:}A$ if there is little danger of confusion) if $\Gamma \vdash M{:}A$ can be produced by the following rules.

$$(x{:}A) \in \Gamma \;\Rightarrow\; \Gamma \vdash x : A;$$

$$\Gamma \vdash M : (A \to B),\ \Gamma \vdash N : A \;\Rightarrow\; \Gamma \vdash (MN) : B;$$

$$\Gamma,\ x{:}A \vdash M : B \;\Rightarrow\; \Gamma \vdash (\lambda x.M) : (A \to B).$$

These rules are usually written as follows.

(axiom)	$\Gamma \vdash x : A,$	if $(x{:}A) \in \Gamma$;
(→-elimination)	$\dfrac{\Gamma \vdash M : (A \to B) \quad \Gamma \vdash N : A}{\Gamma \vdash (MN) : B}$;	
(→-introduction)	$\dfrac{\Gamma, x{:}A \vdash M : B}{\Gamma \vdash (\lambda x.M) : (A \to B)}$.	

Figure 1.5: The system $\lambda^{\mathrm{Cu}}_{\rightarrow}$ of type assignment *á la* Curry

This is the modification to the lambda calculus of the system in Curry [1934], as developed in Curry et al. [1958].

NOTATION. Another way of writing these rules is sometimes found in the literature.

Introduction rule	$\dfrac{\begin{array}{c}\cancel{x:A}\\ \vdots \\ M:B\end{array}}{\lambda x.M : (A{\to}B)}$
Elimination rule	$\dfrac{M:(A\to B) \quad N:A}{MN:B}$

$\lambda_{\to}^{\mathrm{Cu}}$ alternative version

In this version the axiom is considered as implicit and is not notated. The notation

$$\begin{array}{c} x:A \\ \vdots \\ M:B \end{array}$$

denotes that $M:B$ can be derived from $x{:}A$. Striking through $x{:}A$ means that for the conclusion $\lambda x.M : A{\to}B$ the assumption $x{:}A$ is no longer needed; it is *discharged*.

1.1.9. DEFINITION. Let $\Gamma = \{x_1{:}A_1, \ldots, x_n{:}A_n\}$. Then
(i) $\mathrm{dom}(\Gamma) = \{x_1, \ldots, x_n\}$.
(ii) $x_1{:}A_1, \ldots, x_n{:}A_n \vdash M : A$ denotes $\Gamma \vdash M : A$.
(iii) In particular $\vdash M : A$ stands for $\emptyset \vdash M : A$.
(iv) $x_1, \ldots, x_n{:}A \vdash M : B$ stands for $x_1{:}A, \ldots, x_n{:}A \vdash M : B$.

1.1.10. EXAMPLE. (i) $\vdash (\lambda xy.x) : (A \to B \to A)$ for all $A, B \in \mathbb{T}$.
We will use the notation of version 1 of $\lambda_{\to}$ for a derivation of this statement.

$$\dfrac{\dfrac{x{:}A, y{:}B \vdash x : A}{x{:}A \vdash (\lambda y.x) : B{\to}A}}{\vdash (\lambda x \lambda y.x) : A{\to}B{\to}A}$$

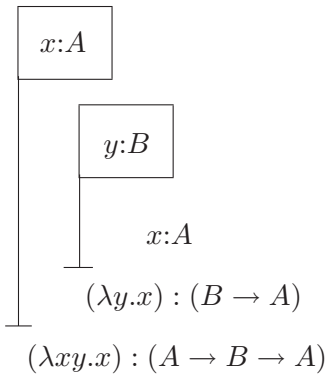
Note that $\lambda xy.x \equiv \lambda x \lambda y.x$ by definition.

(ii) A *natural deduction* derivation (for the alternative version of the system) of the same type assignment is the following.

$$\dfrac{\dfrac{\dfrac{\cancel{x{:}A}\ 2 \qquad \cancel{y{:}B}\ 1}{x{:}A}}{(\lambda y.x) \ : \ (B \to A)}\ 1}{(\lambda xy.x) \ : \ (A \to B \to A)}\ 2$$

The indices 1 and 2 are bookkeeping devices that indicate at which application of a rule a particular assumption is being discharged.

(iii) A more explicit way of dealing with cancellations of statements is the 'flag-notation' used by Fitch (1952) and in the languages AUTOMATH of de Bruijn (1980). In this notation the above derivation becomes as follows.



As one sees, the bookkeeping of cancellations is very explicit; on the other hand it is less obvious how a statement is derived from previous statements in case applications are used.

(iv) Similarly one can show for all $A \in \mathbb{T}$

$$\vdash (\lambda x.x) : (A \to A).$$

(v) An example with a non-empty basis is $y{:}A \vdash (\lambda x.x)y : A$.

In the rest of this chapter and in fact in the rest of this book we usually will introduce systems of typed lambda calculi in the style of the first variant of $\lambda_\to$.

1.1.11. Definition. Let Γ be a basis and $A \in \mathbb{T}$. Then

(i) $\Lambda^\Gamma_\to(A) = \{M \in \Lambda \mid \Gamma \vdash_{\lambda_\to} M : A\}$.

(ii) $\Lambda^\Gamma_\to = \bigcup_{A \in \mathbb{T}} \Lambda^\Gamma_\to(A)$.

(iii) $\Lambda_\to(A) = \Lambda^\emptyset_\to(A)$.

(iv) $\Lambda_\to = \Lambda^\emptyset_\to$

1.1.12. Definition. Let Γ be a basis, $A \in \mathbb{T}$ and $M \in \Lambda$. Then

(i) If $M \in \Lambda_\to(A)$, then we say that

M has type A or *A is inhabited by M*.

(ii) If $M \in \Lambda_\to$, then M is called *typable*.

(iii) If $M \in \Lambda^\Gamma_\to(A)$, then *$M$ has type A relative to Γ*.

(iv) If $M \in \Lambda^\Gamma_\to$, then M is called *typeable relative to Γ*.

(v) If $\Lambda^{(\Gamma)}_\to(A) \neq \emptyset$, then A is *inhabited (relative to Γ)*.

1.1.13. Example. We have

$$\mathsf{K} \in \Lambda^\emptyset_\to(A{\to}B{\to}A);$$
$$\mathsf{K}x \in \Lambda^{\{x:A\}}_\to(B{\to}A).$$

1.1.14. DEFINITION. Let $A \in \mathbb{T}(\lambda_{\to})$.

(i) The *depth* of A, notation $\mathrm{dpt}(A)$, is defined as follows.

$$\begin{array}{rcl} \mathrm{dpt}(\alpha) & = & 0 \\ \mathrm{dpt}(A{\to}B) & = & \max\{\mathrm{dpt}(A), \mathrm{dpt}(B)\} + 1 \end{array}$$

(ii) The *rank* of A, notation $\mathrm{rk}(A)$, is defined as follows.

$$\begin{array}{rcl} \mathrm{rk}(\alpha) & = & 0 \\ \mathrm{rk}(A{\to}B) & = & \max\{\mathrm{rk}(A) + 1, \mathrm{rk}(B)\} \end{array}$$

(iii) The *order* of A, notation $\mathrm{ord}(A)$, is defined as follows.

$$\begin{array}{rcl} \mathrm{ord}(\alpha) & = & 1 \\ \mathrm{ord}(A{\to}B) & = & \max\{\mathrm{ord}(A) + 1, \mathrm{ord}(B)\} \end{array}$$

(iv) The depth (rank or order) of a basis Γ is

$$\max_i\{\mathrm{dpt}(A_i) \mid (x_i{:}A_i) \in \Gamma\},$$

(similarly for the rank and order, respectively). Note that $\mathrm{ord}(A) = \mathrm{rk}(A) + 1$.

1.1.15. DEFINITION. For $A \in \mathbb{T}$ we define $A^k{\to}B$ by recursion on k:

$$\begin{array}{rcl} A^0{\to}B & = & B; \\ A^{k+1}{\to}B & = & A{\to}A^k{\to}B. \end{array}$$

Note that $\mathrm{rk}(A^k{\to}B) = \mathrm{rk}(A{\to}B)$, for all $k > 0$.

Several properties can be proved by induction on the depth of a type. This holds for example for Lemma 1.1.18(i).

The asymmetry in the definition of rank is intended because e.g. a type like $(o{\to}o){\to}o$ is more complex than $o{\to}o{\to}o$, as can be seen by looking to the inhabitants of these types: functionals with functions as arguments versus binary function. Sometimes one uses instead of 'rank' the name *type level*. This notion will turn out to be used most of the times.

In logically motivated papers one finds the notion $\mathrm{ord}(A)$. The reason is that in first-order logic one deals with domains and their elements. In second order logic one deals with functions between first-order objects. In this terminology 0-th order logic can be identified with propositional logic.

The minimal and maximal systems $\lambda^o_{\to}$ and $\lambda^{\infty}_{\to}$

The collection $\mathbb{A}$ of type variables serves as set of base types from which other types are constructed. We have $\mathbb{T}^o = \{o\}$ with just one type atom and $\mathbb{T}_{\infty} = \{\alpha_0, \alpha_1, \alpha_2, \ldots\}$ with infinitely many of them. These two sets of atoms and their resulting type systems play a major role in this Part I of the book.

1.1.16. DEFINITION. We define the following systems of type assignment.

(i) $\lambda^o_\to = \lambda^{\mathbb{T}^o}_\to$. This system is also called λ^τ in the literature.
(ii) $\lambda^\infty_\to = \lambda^{\mathbb{T}}_{\to\infty}$.

If it becomes necessary to distinguish the set of atomic types, will use notations like $\Lambda_o(A) = \Lambda_{\mathbb{T}^o}(A)$ and $\Lambda_\infty(A) = \Lambda_{\mathbb{T}_\infty}(A)$.

Many of the interesting features of the 'larger' $\lambda_\to$ are already present in the minimal version $\lambda^o_\to$. The complexity of $\lambda_\to$ is already present in $\lambda^o_\to$.

1.1.17. DEFINITION. (i) The following types of $\mathbb{T}^o \subseteq \mathbb{T}^{\mathbb{A}}$ are often used.

$$0 = o,\ 1 = 0{\to}0,\ 2 = (0{\to}0){\to}0,\ \ldots.$$

In general

$$0 = o \text{ and } k+1 = k{\to}0.$$

Note that $\mathrm{rk}(n) = n$.
(ii) Define n_k by cases on n.

$$\begin{array}{rcl} o_k & = & o; \\ (n+1)_k & = & n^k{\to}o. \end{array}$$

For example

$$\begin{array}{rcl} 1_2 & = & o{\to}o{\to}o; \\ 2_3 & = & 1{\to}1{\to}1{\to}o. \end{array}$$

Notice that $\mathrm{rk}(n_k) = \mathrm{rk}(n)$, for $k > 0$.

1.1.18. LEMMA. (i) *Every type A of $\lambda^\infty_\to$ is of the form*

$$A = A_1{\to}A_2{\to}\ldots{\to}A_n{\to}\alpha.$$

(ii) *Every type A of $\lambda^o_\to$ is of the form*

$$A = A_1{\to}A_2{\to}\ldots{\to}A_n{\to}o.$$

(iii) $\mathrm{rk}(A_1{\to}A_2{\to}\ldots{\to}A_n{\to}\alpha) = \max\{\mathrm{rk}(A_i)+1 \mid 1 \le i \le n\}$.

PROOF. (i) By induction on the structure (depth) of A. If $A = \alpha$, then this holds for $n = 0$. If $A = B{\to}C$, then by the induction hypothesis one has $C = C_1{\to}\ldots{\to}C_n{\to}\gamma$. Hence $A = B{\to}C_1{\to}\ldots{\to}C_n{\to}\gamma$.
(ii) By (i).
(iii) By induction on n. ■

1.1.19. NOTATION. Let $A \in \mathbb{T}^{\mathbb{A}}$ and suppose $A = A_1{\to}A_2{\to}\ldots{\to}A_n{\to}\alpha$. Then the A_i are called the *components* of A. We write

$$\begin{array}{rcll} \mathtt{arity}(A) & = & n, & \\ A(i) & = & A_i, & \text{for } 1 \le i \le n; \\ \mathtt{target}(A) & = & \alpha. & \end{array}$$

Iterated components are denoted as follows

$$A(i,j) = A(i)(j).$$

Different versions of $\lambda^{\mathbb{A}}_{\to}$

The system $\lambda^{\mathbb{A}}_{\to}$ that was introduced in Definition 1.1.8 assigns types to untyped lambda terms. These system will be referred to as the Curry system and be denoted by $\lambda^{\mathbb{A}}_{\to \mathrm{Cu}}$ or $\lambda^{\mathrm{Cu}}_{\to}$, as the set $\mathbb{A}$ often does not need to be specified. There will be introduced two variants of $\lambda^{\mathbb{A}}_{\to}$.

The first variant of $\lambda^{\mathrm{Cu}}_{\to}$ is the *Church* version of $\lambda^{\mathbb{A}}_{\to}$, denoted by $\lambda^{\mathbb{A}}_{\to \mathrm{Ch}}$ or $\lambda^{\mathrm{Ch}}_{\to}$. In this theory the types are assigned to embellished terms in which the variables (free and bound) come with types attached. For example the Curry style type assignments

$$\begin{array}{rll} & \vdash^{\mathrm{Cu}}_{\lambda_{\to}} & (\lambda x.x) : A{\to}A \qquad (1_{\mathrm{Cu}}) \\ y{:}A & \vdash^{\mathrm{Cu}}_{\lambda_{\to}} & (\lambda x.xy) : (A{\to}B){\to}A{\to}B \qquad (2_{\mathrm{Cu}}) \end{array}$$

now becoming

$$\begin{array}{ll} (\lambda x^A.x^A) \in \Lambda^{\mathrm{Ch}}_{\to}(A{\to}A) & (1_{\mathrm{Ch}}) \\ (\lambda x^{A\to B}.x^{A\to B}y^A) : \Lambda^{\mathrm{Ch}}_{\to}((A{\to}B){\to}A{\to}B) & (2_{\mathrm{Ch}}) \end{array}$$

The second variant of $\lambda^{\mathrm{Cu}}_{\to}$ is the *de Bruijn* version of $\lambda^{\mathbb{A}}_{\to}$, denoted by $\lambda^{\mathbb{A}}_{\to \mathrm{dB}}$ or $\lambda^{\mathrm{dB}}_{\to}$. Now only bound variables get ornamented with types, but only at the binding stage. The examples (1), (2) now become

$$\begin{array}{rll} & \vdash^{\mathrm{dB}}_{\lambda_{\to}} & (\lambda x : A.x) : A{\to}A \qquad (1_{\mathrm{dB}}) \\ y{:}A & \vdash^{\mathrm{dB}}_{\lambda_{\to}} & (\lambda x : (A{\to}B).xy) : (A{\to}B){\to}A{\to}B \qquad (2_{\mathrm{dB}}) \end{array}$$

The reasons to have these variants will be explained later in Section 1.4. In the meantime we will work intuitively.

1.1.20. Notation. Terms like $(\lambda fx.f(fx)) \in \Lambda^{\emptyset}(1{\to}o{\to}o)$ will often be written

$$\lambda f^1 x^0.f(fx)$$

to indicate the types of the bound variables. We will come back to this notational issue in section 1.4.

1.2. Normal inhabitants

In this section we will give an algorithm that enumerates the set of closed terms in normal form of a given type $A \in \mathbb{T}$. Since we will prove in the next chapter that all typable terms do have a nf and that reduction preserves typing, we thus have an enumeration of essentially all closed terms of that given type. We do need to distinguish various kinds of nf's.

1.2.1. Definition. Let $A = A_1{\to}\ldots A_n{\to}\alpha$ and suppose $\Gamma \vdash M : A$.

(i) Then M is in long-nf, notation lnf, if $M \equiv \lambda x_1^{A_1} \ldots x_n^{A_n}.xM_1 \ldots M_n$ and each M_i is in lnf. By induction on the depth of the type of the closure of M one sees that this definition is well-founded.

(ii) M has a lnf if $M =_{\beta\eta} N$ and N is a lnf.

In Exercise 1.5.16 it is proved that if M has a β-nf, which according to Theorem 2.2.4 is always the case, then it also has a unique lnf and will be its unique $\beta\boldsymbol{\eta}^{-1}$ nf. Here η^{-1} is the notion of reduction that is the converse of η.

1.2.2. Examples. (i) Note that $\lambda f^1.f =_{\beta\eta} \lambda f^1\lambda x^o.fx$ and that $\lambda f^1.f$ is a $\beta\boldsymbol{\eta}$-nf but not a lnf.

(ii) $\lambda f^1\lambda x^o.fx$ is a lnf, but not a $\beta\boldsymbol{\eta}$-nf.

(iii) $\lambda x{:}o.x$ is both in $\beta\boldsymbol{\eta}$-nf and lnf.

(iv) The β-nf $\lambda F{:}2_2\lambda f{:}1.Ff(\lambda x{:}o.fx)$ is neither in $\beta\boldsymbol{\eta}$-nf nor lnf.

(v) A variable of atomic type α is a lnf, but of type $A{\to}B$ not.

(vi) A variable $f : 1{\to}1$ has as lnf $\lambda g^1\lambda x^o.f(\lambda y^o.gy)x$.

1.2.3. Proposition. *Every β-nf M has a lnf M^ℓ such that $M^\ell \twoheadrightarrow_{\boldsymbol{\eta}} M$.*

Proof. Define M^ℓ by induction on the depth of the type of the closure of M as follows.

$$M^\ell \equiv (\lambda\vec{x}.yM_1\dots M_n)^\ell = \lambda\vec{x}\vec{z}.yM_1^\ell\dots M_n^\ell\vec{z}^\ell.$$

Then M^ℓ does the job. ■

Now we will define a 2-level grammar for obtaining the collection of all lnf's of a given type A.

1.2.4. Definition. Let $N = \{L(A;\Gamma) \mid A \in \mathbb{T}^{\mathbb{A}}; \Gamma \text{ a context of } \lambda_{\to}\}$. Let Σ be the alphabet of the terms of the $\lambda_{\to}^{\mathrm{Ch}}$. Define the following two-level grammar, see van Wijngaarden et al. [1976], as a notion of reduction over words over $N \cup \Sigma$. The elements of N are the non-terminals (unlike in a context-free language there are now infinitely many of them).

$$\begin{array}{rcll} L(\alpha;\Gamma) & \Rightarrow & xL(B_1;\Gamma)\dots L(B_n;\Gamma), & \text{if } (x{:}\vec{B}{\to}\alpha)\in\Gamma; \\ L(A{\to}B;\Gamma) & \Rightarrow & \lambda x^A.L(B;\Gamma,x{:}A). & \end{array}$$

Typical productions of this grammar are the following.

$$\begin{array}{rcl} L(3;\emptyset) & \Rightarrow & \lambda F^2.L(o;F^2) \\ & \Rightarrow & \lambda F^2.FL(1;F^2) \\ & \Rightarrow & \lambda F^2.F(\lambda x^o.L(o;F^2,x^o)) \\ & \Rightarrow & \lambda F^2.F(\lambda x^o.x). \end{array}$$

But one has also

$$\begin{array}{rcl} L(o;F^2,x^o) & \Rightarrow & FL(1;F^2,x^o) \\ & \Rightarrow & F(\lambda x_1^o.L(o;F^2,x^o,x_1^o)) \\ & \Rightarrow & F(\lambda x_1^o.x_1). \end{array}$$

Hence ($\twoheadRightarrow$ denotes the transitive reflexive closure of $\Rightarrow$)

$$L(3;\emptyset) \twoheadRightarrow \lambda F^2.F(\lambda x^o.F(\lambda x_1^o.x_1)).$$

In fact, $L(3;\emptyset)$ reduces to all possible closed lnf's of type 3. Like in abstract syntax we do not produce parentheses from the $L(A;\Gamma)$, but write them when needed.

1.2.5. PROPOSITION. *Let Γ, M, A be given. Then*

$$L(A,\Gamma) \twoheadrightarrow M \iff \Gamma \vdash M : A \;\&\; M \text{ is in lnf.}$$

Now we will modify the 2-level grammar and the inhabitation machines in order to produce all β-nf's.

1.2.6. DEFINITION. The 2-level grammar N is defined as follows.

$$\begin{array}{rcll} N(A;\Gamma) & \Rightarrow & xN(B_1;\Gamma)\ldots N(B_n;\Gamma), & \text{if } (x{:}\vec{B}{\to}A)\in\Gamma; \\ N(A{\to}B;\Gamma) & \Rightarrow & \lambda x^A.N(B;\Gamma,x{:}A). & \end{array}$$

Now the β-nf's are being produced. As an example we make the following production. Remember that $1 = o{\to}o$.

$$\begin{array}{rcl} L(1{\to}o{\to}o;\emptyset) & \Rightarrow & \lambda f^1.L(o{\to}o;f{:}o{\to}o) \\ & \Rightarrow & \lambda f^1.f. \end{array}$$

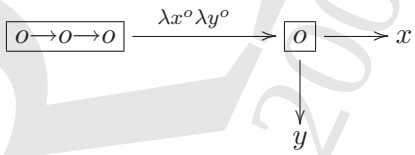
1.2.7. PROPOSITION. *Let Γ, M, A be given. Then*

$$N(A,\Gamma) \twoheadrightarrow M \iff \Gamma \vdash M : A \;\&\; M \text{ is in } \beta\text{-nf.}$$

Inhabitation machines

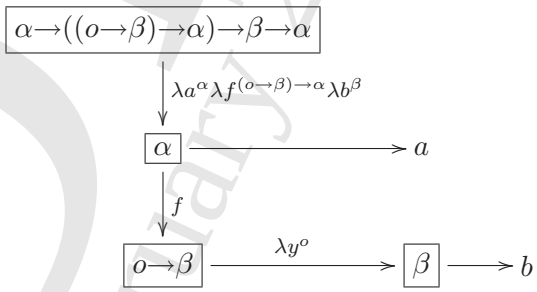
Inspired by this proposition one can introduce for each type A a machine M_A producing the set of closed terms of that type. If one is interested in terms containing variables $x_1^{A_1}, \ldots, x_n^{A_n}$, then one can also find these terms by considering the machine for the type $A_1{\to}\ldots{\to}A_n{\to}A$ and look at the subproduction at node A.

1.2.8. EXAMPLES. (i) $A = o{\to}o{\to}o$. Then M_A is



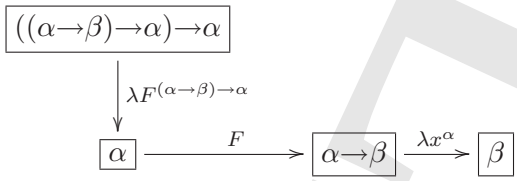
This shows that the type 1_2 has two closed inhabitants: $\lambda xy.x$ and $\lambda xy.y$. We see that the two arrows leaving $\boxed{\text{o}}$ represent a choice.

(ii) $A = \alpha{\to}((o{\to}\beta){\to}\alpha){\to}\beta{\to}\alpha$. Then M_A is



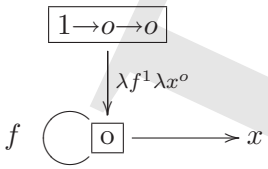
Again there are only two inhabitants, but now the production of them is rather different: $\lambda afb.a$ and $\lambda afb.f(\lambda x^o.b)$.

(iii) $A = ((\alpha{\to}\beta){\to}\alpha){\to}\alpha$. Then M_A is



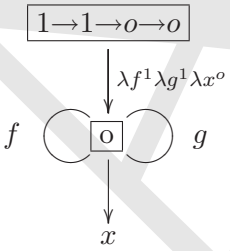
This type, corresponding to Peirce's law, does not have any inhabitants.

(iv) $A = 1{\to}o{\to}o$. Then M_A is



This is the type Nat having the Church's numerals $\lambda f^1 x^o.f^n x$ as inhabitants.

(v) $A = 1{\to}1{\to}o{\to}o$. Then M_A is

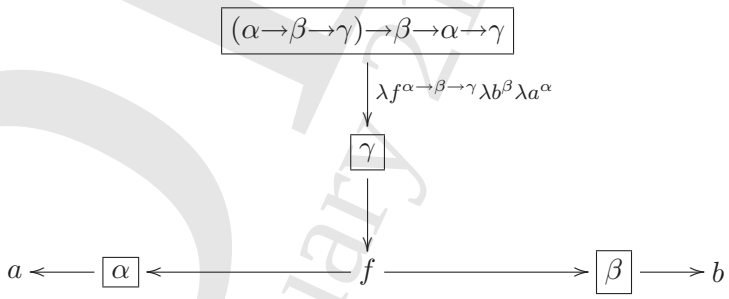


Inhabitants of this type represent words over the alphabet $\Sigma = \{f, g\}$, for example

$$\lambda f^1 g^1 x^o.fgffgfggx,$$

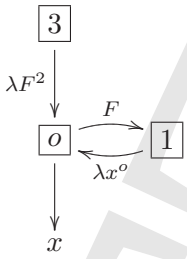
where we have to insert parentheses associating to the right.

(vi) $A = (\alpha{\to}\beta{\to}\gamma){\to}\beta{\to}\alpha{\to}\gamma$. Then M_A is



giving as term $\lambda f^{\alpha\to\beta\to\gamma}\lambda b^\beta\lambda a^\alpha.fab$. Note the way an interpretation should be given to paths going through f: the outgoing arcs (to $\boxed{\alpha}$ and $\boxed{\beta}$) should be completed both separately in order to give f its two arguments.

(vii) $A = 3$. Then M_A is

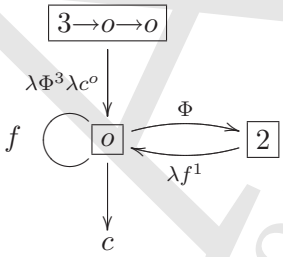


This type 3 has inhabitants having more and more binders:

$$\lambda F^2.F(\lambda x_0^o.F(\lambda x_1^o.F(\cdots(\lambda x_n^o.x_i)))).$$

The novel phenomenon that the binder λx^o may go round and round forces us to give new incarnations λx_0^o, $\lambda x_1^o, \ldots$ each time we do this (we need a counter to ensure freshness of the bound variables). The 'terminal' variable x can take the shape of any of the produced incarnations x_k. As almost all binders are dummy, we will see that this potential infinity of binding is rather innocent and the counter is not yet really needed here.

(viii) $A = 3{\to}o{\to}o$. Then M_A is

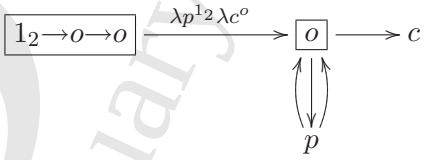


This type, called the *monster* M, does have a potential infinite amount of binding, having as terms e.g.

$$\lambda\Phi^3 c^o.\Phi\lambda f_1^1.f_1\Phi\lambda f_2^1.f_2 f_1\Phi\ldots\lambda f_n^1.f_n\ldots f_2 f_1 c,$$

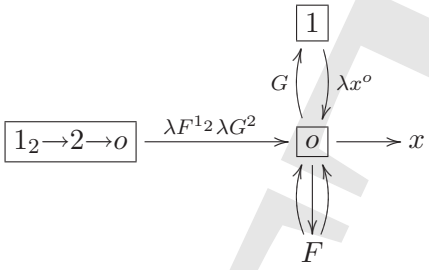
again with inserted parentheses associating to the right. Now a proper bookkeeping of incarnations (of f^1 in this case) becomes necessary, as the f going from $\boxed{o}$ to itself needs to be one that has already been incarnated.

(ix) $A = 1_2{\to}o{\to}o$. Then M_A is



This is the type of binary trees, having as elements, e.g. $\lambda p^{1_2} c^o.c$ and $\lambda p^{1_2} c^o.pc(pcc)$. Again, as in example (vi) the outgoing arcs from p (to $\boxed{o}$) should be completed both separately in order to give p its two arguments.

(x) $A = 1_2{\to}2{\to}o$. Then M_A is



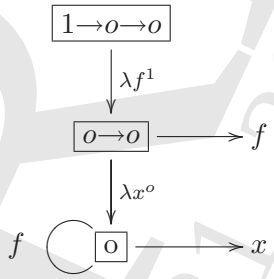
This is the type L corresponding to untyped lambda terms. For example the untyped terms $\omega \equiv \lambda x.xx$ and $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$ can be translated to $(\omega)^t \equiv \lambda F^{1_2}G^2.G(\lambda x^o.Fxx)$ and

$$\begin{array}{rcl}(\Omega)^t & \equiv & \lambda F^{1_2}G^2.F(G(\lambda x^o.Fxx))(G(\lambda x^o.Fxx)) \\ & =_\beta & \lambda FG.F((\omega)^tFG)((\omega)^tFG) \\ & =_\beta & (\omega)^t \cdot_L (\omega)^t,\end{array}$$

where for $M, N \in L$ one defines $M \cdot_L N = \lambda FG.F(MFG)(NFG)$. All features of producing terms inhabiting types (bookkeeping bound variables, multiple paths) are present here.

Following the 2-level grammar N one can make inhabitation machines for β-nf M^β_A.

1.2.9. Example. We show how the production machine for β-nf's differs from the one for lnf's. Let $A = 1{\to}o{\to}o$. Then $\lambda f^1.f$ is the (unique) β-nf of type A that is not a lnf. It will come out from the following machine M^β_A.



So in order to obtain the β-nf's, one has to allow output at types that are not atomic.

1.3. Representing data types

In this section it will be shown that first order algebraic data types can be represented in $\lambda^o_{\to}$. We start with several examples: Booleans, the natural numbers, the free monoid over n generators (words over a finite alphabet with n elements) and trees with at the leafs labels from a type A. The following definitions depend on a given type A. So in fact $\mathsf{Bool} = \mathsf{Bool}_A$ etcetera. Often one takes $A = o$.

Booleans

1.3.1. DEFINITION. Define

$$\begin{array}{rcl} \mathsf{Bool} & \equiv & A{\to}A{\to}A; \\ \mathsf{true} & \equiv & \lambda xy.x; \\ \mathsf{false} & \equiv & \lambda xy.y. \end{array}$$

Then $\mathsf{true} \in \Lambda(\mathsf{Bool})$ and $\mathsf{false} \in \Lambda(\mathsf{Bool})$.

1.3.2. PROPOSITION. *There are terms* not, $\&$, or, imp, iff *with the expected behavior on Booleans. For example* $\mathsf{not} \in \Lambda(\mathsf{Bool}{\to}\mathsf{Bool})$ *and*

$$\begin{array}{rcl} \mathsf{not}\ \mathsf{true} & =_\beta & \mathsf{false}, \\ \mathsf{not}\ \mathsf{false} & =_\beta & \mathsf{true}. \end{array}$$

PROOF. Take $\mathsf{not} \equiv \lambda axy.ayx$ and $\mathsf{or} \equiv \lambda abxy.ax(bxy)$. From these two operations the other Boolean functions can be defined. For example, implication can be represented by

$$\mathsf{imp} \equiv \lambda ab.\mathsf{or}(\mathsf{not}\ a)b.$$

A shorter representation is $\lambda abxy.a(bxy)x$, the normal form of imp. ■

Natural numbers

Following Church the set of natural numbers can be represented as a type

$$\mathsf{Nat} \equiv (A{\to}A){\to}A{\to}A.$$

For each natural number $n \in \mathbb{N}$ we define its representation

$$\mathbf{c}_n \equiv \lambda fx.f^n x,$$

where

$$\begin{array}{rcl} f^0 x & \equiv & x \\ f^{n+1} x & \equiv & f(f^n x). \end{array}$$

1.3.3. PROPOSITION. (i) *There exists a term* $\mathsf{S}^+ \in \Lambda(\mathsf{Nat}{\to}\mathsf{Nat})$ *such that*

$$\mathsf{S}^+\mathbf{c}_n =_\beta \mathbf{c}_{n+1}, \text{ for all } n \in \mathbb{N}.$$

(ii) *There is a term* $\mathsf{zero}_? \in \Lambda(\mathsf{Nat}{\to}\mathsf{Bool})$ *such that*

$$\begin{array}{rcl} \mathsf{zero}_?\,\mathbf{c}_0 & =_\beta & \mathsf{true} \\ \mathsf{zero}_?(\mathsf{S}^+ x) & =_\beta & \mathsf{false}. \end{array}$$

PROOF. (i) Take $\mathsf{S}^+ \equiv \lambda n\lambda fx.f(nfx)$. Then

$$\begin{array}{rcl} \mathsf{S}^+\mathbf{c}_n & =_\beta & \lambda fx.f(\mathbf{c}_n fx) \\ & =_\beta & \lambda fx.f(f^n x) \\ & \equiv & \lambda fx.f^{n+1}x \\ & \equiv & \mathbf{c}_{n+1}. \end{array}$$

(ii) Take $\mathsf{zero}_? \equiv \lambda n\lambda ab.n(\mathsf{K}b)a$. Then

$$\begin{array}{rcl}
\mathsf{zero}_?\,\mathbf{c}_0 & =_\beta & \lambda ab.\mathbf{c}_0(\mathsf{K}b)a \\
& =_\beta & \lambda ab.a \\
& \equiv & \mathsf{true}; \\
\mathsf{zero}_?\,(\mathsf{S}^+x) & =_\beta & \lambda ab.\mathsf{S}^+x(\mathsf{K}b)a \\
& =_\beta & \lambda ab.(\lambda fy.f(xfy))(\mathsf{K}b)a \\
& =_\beta & \lambda ab.\mathsf{K}b(x(\mathsf{K}b)a) \\
& =_\beta & \lambda ab.b \\
& \equiv & \mathsf{false}.\ \blacksquare
\end{array}$$

Addition and multiplication are definable in $\lambda_\to$.

1.3.4. Proposition. (i) *There is a term* $\mathsf{plus} \in \Lambda(\mathsf{Nat}\to\mathsf{Nat}\to\mathsf{Nat})$ *satisfying*

$$\mathsf{plus}\ \mathbf{c}_n\ \mathbf{c}_m =_\beta \mathbf{c}_{n+m}.$$

(ii) *There is a term* $\mathsf{times} \in \Lambda(\mathsf{Nat}\to\mathsf{Nat}\to\mathsf{Nat})$ *such that*

$$\mathsf{times}\ \mathbf{c}_n\ \mathbf{c}_m =_\beta \mathbf{c}_{n\cdot m}$$

Proof. (i) Take $\mathsf{plus} \equiv \lambda nm\lambda fx.nf(mfx)$. Then

$$\begin{array}{rcl}
\mathsf{plus}\ \mathbf{c}_n\ \mathbf{c}_m & =_\beta & \lambda fx.\mathbf{c}_nf(\mathbf{c}_mfx) \\
& =_\beta & \lambda fx.f^n(f^mx) \\
& \equiv & \lambda fx.f^{n+m}x \\
& \equiv & \mathbf{c}_{n+m}.
\end{array}$$

(ii) Take $\mathsf{times} \equiv \lambda nm\lambda fx.m(\lambda y.nfy)x$. Then

$$\begin{array}{rcl}
\mathsf{times}\ \mathbf{c}_n\ \mathbf{c}_m & =_\beta & \lambda fx.\mathbf{c}_m(\lambda y.\mathbf{c}_nfy)x \\
& =_\beta & \lambda fx.\mathbf{c}_m(\lambda y.f^ny)x \\
& =_\beta & \lambda fx.\underbrace{(f^n(f^n(\ldots(f^n}_{m\text{ times}}x)..))) \\
& \equiv & \lambda fxp.f^{n\cdot m}x \\
& \equiv & \mathbf{c}_{n\cdot m}.\ \blacksquare
\end{array}$$

1.3.5. Corollary. *For every polynomial* $p \in \mathbb{N}[x_1,\ldots,x_k]$ *there is a closed term* $M_p{:}\Lambda(\mathsf{Nat}^k\to\mathsf{Nat})$ *such that* $\forall n_1,\ldots,n_k \in \mathbb{N}.M_p\mathbf{c}_{n_1}\ldots\mathbf{c}_{n_k} =_\beta \mathbf{c}_{p(n_1,\ldots,n_k)}$. $\blacksquare$

From the results obtained so far it follows that the polynomials extended by case distinctions (being equal or not to zero) are definable in $\lambda_\to$. In Statman [1976] or Schwichtenberg [1976] it is proved that exactly these so-called extended polynomials are definable in $\lambda_\to$. Hence primitive recursion cannot be defined in $\lambda_\to$; in fact not even the predecessor function, see Proposition 2.4.22.

Words over a finite alphabet

Let $\Sigma = \{a_1, \ldots, a_k\}$ be a finite alphabet. Then Σ^* the collection of words over Σ can be represented in $\lambda_{\to}$.

1.3.6. DEFINITION. (i) The type for words in Σ^* is

$$\mathsf{Sigma}^* \equiv (o \to o)^k \to o \to o.$$

(ii) Let $w = a_{i_1} \ldots a_{i_p}$ be a word. Define

$$\begin{aligned} \underline{w} &\equiv \lambda a_1 \ldots a_k x. a_{i_1}(\ldots(a_{i_p}x)..) \\ &\equiv \lambda a_1 \ldots a_k x. (a_{i_1} \circ \cdots \circ a_{i_p})x. \end{aligned}$$

Note that $\underline{w} \in \Lambda(\mathsf{Sigma}^*)$. If ϵ is the empty word ( ), then naturally

$$\begin{aligned} \underline{\epsilon} &\equiv \lambda a_1 \ldots a_k x.x \\ &\equiv \mathsf{K}^k\mathsf{I}. \end{aligned}$$

Now we show that the operation concatenation can be defined in $\lambda_{\to}$.

1.3.7. PROPOSITION. *There exists a term* $\mathsf{concat} \in \Lambda(\mathsf{Sigma}^* \to \mathsf{Sigma}^* \to \mathsf{Sigma}^*)$ *such that for all* $w, v \in \Sigma^*$

$$\mathsf{concat}\ \underline{w}\ \underline{v} = \underline{wv}.$$

PROOF. Define

$$\mathsf{concat} \equiv \lambda wv.\vec{a}x.w\vec{a}(v\vec{a}x).$$

Then the type is correct and the definition equation holds. ■

1.3.8. PROPOSITION. (i) *There exists a term* $\mathsf{empty} \in \Lambda(\mathsf{Sigma}^*)$ *such that*

$$\begin{aligned} \mathsf{empty}\ \underline{\epsilon} &= \mathsf{true} \\ \mathsf{empty}\ \underline{w} &= \mathsf{false} \qquad \text{if } w \neq \epsilon. \end{aligned}$$

(ii) *Given a (represented) word* $w_0 \in \Lambda(\mathsf{Sigma}^*)$ *and a term* $G \in \Lambda(\mathsf{Sigma}^* \to \mathsf{Sigma}^*)$ *there exists a term* $F \in \Lambda(\mathsf{Sigma}^* \to \mathsf{Sigma}^*)$ *such that*

$$\begin{aligned} F\ \underline{\epsilon} &= w_0; \\ F\ \underline{w} &= G\underline{w}, \qquad \text{if } w \neq \epsilon. \end{aligned}$$

PROOF. (i) Take $\mathsf{empty} \equiv \lambda wpq.w(\mathsf{K}q)^{\sim k}p$.
(ii) Take $F \equiv \lambda w \lambda x \vec{a}.\mathsf{empty}\, w(w_0\vec{a}x)(Gw\vec{a}x)$. ■

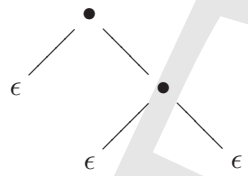
One cannot define a terms 'car' or 'cdr' such that car $\underline{aw} = w$ and *cdr* $\underline{aw} = w$.

Trees

1.3.9. DEFINITION. The set of binary trees, notation T_2, is defined by the following abstract syntax

$$t = \epsilon \,|\, p(t,t)$$

Here ϵ is the 'empty tree' and P is the constructor that puts two trees together. For example $p(\epsilon, p(\epsilon,\epsilon)) \in T_2$ can be depicted as



Now we will represent T_2 as a type in $\lambda_{\rightarrow}$.

1.3.10. DEFINITION. (i) The set T_2 will be represented by the type

$$\top_2 \equiv (o^2 \rightarrow o) \rightarrow o \rightarrow o.$$

(ii) Define for $t \in \mathcal{T}_2$ its representation $\underline{t}$ inductively as follows.

$$\begin{array}{rcl} \underline{\epsilon} & = & \lambda pe.e; \\ \underline{p(t,s)} & = & \lambda p(tpe)(spe). \end{array}$$

(iii) Write

$$\begin{array}{rcl} E & = & \lambda pe.e; \\ P & = & \lambda tspe.p(tpe)(spe). \end{array}$$

Note that Note that for $t \in T_2$ one has $\underline{t} \in \Lambda(\top_2)$

The following follows immediately from this definition.

1.3.11. PROPOSITION. *The map* $\_ : T_2 \rightarrow \top_2$ *can be defined inductively as follows*

$$\begin{array}{rcl} \underline{\epsilon} & = & E; \\ \underline{p(t,s)} & = & P\underline{t}\,\underline{s}. \end{array}$$

Interesting functions, like the one that selects on of the two branches of a tree cannot be defined in $\lambda_{\rightarrow}$.

The type $\top_2$ will play an important role in Section 3.4.

Representing Free algebras with a handicap

Now we will see that all the examples are special cases of a general construction. It turns out that first order algebraic data types $\mathcal{A}$ can be represented in $\lambda^o_{\rightarrow}$. The representations are said to have a handicap because not all primitive recursive functions on $\mathcal{A}$ are representable. Mostly the destructors cannot be represented. In a special cases one can do better. Every finite algebra can be represented with all possible functions on them. Pairing with projections can be represented.

1.3.12. DEFINITION. (i) An *algebra* is a set A with a specific finite set of operators of different arity:

$$\begin{array}{rcll} c_1, c_2, \ldots & \in & A & \text{(constants, we may call these 0-ary functions);} \\ f_1, f_2, \ldots & \in & A{\rightarrow}A & \text{(unary functions);} \\ g_1, g_2, \ldots & \in & A^2{\rightarrow}A & \text{(binary function));} \\ \quad \ldots & & & \\ h_1, h_2, \ldots & \in & A^n{\rightarrow}A & \text{(n-ary functions).} \end{array}$$

(ii) An n-ary function $k : A^n{\rightarrow}A$ is called *algebraic* iff k can be defined explicitly from the constructors. For example

$$k = \lambda\!\!\lambda a_1 a_2.g_1(a_1, (g_2(h_1(a_2), c_2)))$$

is a binary algebraic function.

(iii) An element a of A is called *algebraic* iff a is an algebraic 0-ary function. Algebraic elements of A can be denoted by first-order terms over the algebra.

(iv) The algebra A is *free(ly generated)* if every element of A is algebraic and moreover if for two first-order terms t, s one has

$$t = s \;\Rightarrow\; t \equiv s.$$

In a free algebra the given operators are called *constructors*.

For example $\mathbb{N}$ with constructors $0, s$ (s is the successor) is a free algebra. But $\mathbb{Z}$ with $0, s, p$ (p is the predecessor) is not free. Indeed, $0 = p(s(0))$, but $0 \not\equiv p(s(0))$ as syntactic expressions.

1.3.13. THEOREM. *For a free algebra $\mathcal{A}$ there is a type $\underline{\mathcal{A}} \in \mathbb{T}^o$ and a map $\lambda\!\!\lambda a.\underline{a} : \mathcal{A}{\rightarrow}\Lambda(\underline{\mathcal{A}})$ satisfying the following.*

(i) *$\underline{a}$ is a lnf, for every $a \in \mathcal{A}$.*

(ii) *$\underline{a} =_{\beta\boldsymbol{\eta}} \underline{b} \iff a = b$.*

(iii) *$\Lambda(\underline{\mathcal{A}}) = \{\underline{a} \mid a \in \mathcal{A}\}$, up to $\beta\boldsymbol{\eta}$-conversion.*

(iv) *For k-ary algebraic functions f on $\mathcal{A}$ there is an $\underline{f} \in \Lambda(\underline{\mathcal{A}}{\rightarrow}\underline{\mathcal{A}})$ such that*

$$\underline{f}\,\underline{a}_1 \ldots \underline{a}_k = \underline{f(a_1, \ldots, a_k)}.$$

(v) *There is a representable discriminator distinguishing between elements of the form $c, f_1(a), f_2(a,b), \ldots, f_n(a_1, \ldots, a_n)$. More precisely, there is a term $\mathsf{test} \in \Lambda(\underline{\mathcal{A}}{\rightarrow}\underline{\mathbb{N}})$ such that for all $a, b \in \mathcal{A}$*

$$\begin{array}{rcl} \mathsf{test}\ \underline{c} & = & \mathbf{c}_0; \\ \mathsf{test}\ \underline{f_1(a)} & = & \mathbf{c}_1; \\ \mathsf{test}\ \underline{f_2(a,b)} & = & \mathbf{c}_2 \\ \quad \ldots & & \\ \mathsf{test}\ \underline{f_n(vectan)} & = & \mathbf{c}_n \end{array}$$

PROOF. We show this by a representative example. Let $\mathcal{A}$ be freely generated by, say, the 0-ary constructor c, the 1-ary constructor f and the 2-ary constructor g. Then an element like

$$a = g(c, f(c))$$

is represented by

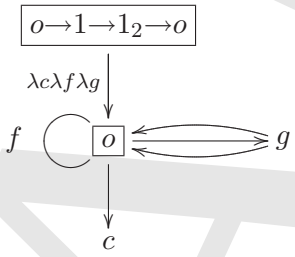
$$\underline{a} = \lambda cfg.gc(fc) \in \Lambda(o{\to}1{\to}1_2{\to}o).$$

Taking $\underline{\mathcal{A}} = o{\to}1{\to}1_2{\to}o$ we will verify the claims. First realize that $\underline{a}$ is constructed from a via $a^\sim = gc(fc)$ and then taking the closure $\underline{a} = \lambda cfg.a^\sim$.

(i) Clearly the $\underline{a}$ are in lnf.

(ii) If a and b are different, then their representations $\underline{a}, \underline{b}$ are different lnfs, hence $\underline{a} \neq_{\beta\boldsymbol{\eta}} \underline{b}$.

(iii) The inhabitation machine $M_{\underline{\mathcal{A}}} = M_{o{\to}1{\to}1_2{\to}o}$ looks like



It follows that for every $M \in \Lambda(\underline{\mathcal{A}})$ one has $M =_{\beta\boldsymbol{\eta}} \lambda cfg.a^\sim = \underline{a}$ for some $a \in \mathcal{A}$. This shows that $\Lambda(\underline{\mathcal{A}}) \subseteq \{\underline{a} \mid a \in \mathcal{A}\}$. The converse inclusion is trivial. In the general case (for other data types $\mathcal{A}$) one has that $\mathrm{rk}(\underline{\mathcal{A}}) = 2$. Hence the lnf inhabitants of $\Lambda(\underline{\mathcal{A}})$ have the form $\lambda cfg.P$, where P is a combination of the variables c, f, g. This means that the corresponding inhabitation machine is similar and the argument goes through in general.

(iv) An algebraic function is explicitly defined from the constructors. We first define representations for the constructors.

$$\begin{array}{lcll} \underline{c} & = & \lambda cfg.c & : \ \underline{\mathcal{A}}; \\ \underline{f} & = & \lambda acfg.f(acfg) & : \ \underline{\mathcal{A}}{\to}\underline{\mathcal{A}}; \\ \underline{g} & = & \lambda abcfg.g(acfg)(bcfg) & : \ \underline{\mathcal{A}}^2{\to}\underline{\mathcal{A}}. \end{array}$$

$$\begin{array}{lcl} \text{Then } \ \underline{f}\,\underline{a} & = & \lambda cfg.f(\underline{a}cfg) \\ & = & \lambda cfg.f(a^\sim) \\ & \equiv & \lambda cfg.(f(a))^\sim, \ \text{(tongue in cheek)}, \\ & \equiv & \underline{f(a)}. \end{array}$$

Similarly one has $\underline{g}\,\underline{a}\,\underline{b} = \underline{g(a,b)}$.

Now if e.g. $h(a,b) = g(a, f(b))$, then we can take

$$\underline{h} \equiv \lambda ab.\underline{g}a(\underline{f}b) : \underline{\mathcal{A}}^2{\to}\underline{\mathcal{A}}.$$

Then clearly $\underline{h}\,\underline{a}\,\underline{b} = \underline{h(a,b)}$.

(v) Take $\mathsf{test} \equiv \lambda afc.a(\mathbf{c}_0 fc)(\lambda x.\mathbf{c}_1 fc)(\lambda xy.\mathbf{c}_2 fc)$. ■

1.3.14. DEFINITION. The notion of free algebra can be generalized to a free *multi-sorted* algebra. We do this by giving an example. The collection of lists of natural numbers, notation $L_{\mathbb{N}}$ can be defined by the 'sorts' $\mathbb{N}$ and $L_{\mathbb{N}}$ and the constructors

$$\begin{array}{rcl} 0 & \in & \mathbb{N}; \\ s & \in & \mathbb{N}{\rightarrow}\mathbb{N}; \\ \text{nil} & \in & \Lambda_{\mathbb{N}}; \\ \text{cons} & \in & \mathbb{N}{\rightarrow}L_{\mathbb{N}}{\rightarrow}L_{\mathbb{N}}. \end{array}$$

In this setting the list $[0,1] \in L_{\mathbb{N}}$ is

$$\text{cons}(0,\text{cons}(s(0),\text{nil})).$$

More interesting multisorted algebras can be defined that are 'mutually recursive', see Exercise 1.5.15.

1.3.15. COROLLARY. *Every freely generated multi-sorted first-order algebra can be represented in a way similar to that in Theorem 1.3.13.*

PROOF. Similar to that of the Theorem. ■

Finite Algebras

For finite algebras one can do much better.

1.3.16. THEOREM. *For every finite set $A = \{a_1, \ldots, a_n\}$ there exists a type $\underline{A} \in \mathbb{T}^o$ and elements $\underline{a}_1, \ldots, \underline{a}_n \in \Lambda(\underline{A})$ such that the following holds.*

(i) $\Lambda(\underline{A}) = \{\underline{a} \mid a \in A\}$.

(ii) *For all k and $f : A^k \rightarrow A$ there exists an $\underline{f} \in \Lambda(\underline{A}^k \rightarrow \underline{A})$ such that*

$$\underline{f}\ \underline{b}_1 \ldots \underline{b}_k = \underline{f(b_1, \ldots, b_k)}.$$

PROOF. Take $\underline{A} = 1_n = o^n \rightarrow o$ and $\underline{a}_i = \lambda b_1 \ldots b_n . b_i \in \Lambda(1_n)$.

(i) By a simple argument using the inhabitation machine M_{1_n}.

(ii) By induction on k. If $k = 0$, then f is an element of A, say $f = a_i$. Take $\underline{f} = \underline{a}_i$. Now suppose we can represent all k-ary functions. Given $f : A^{k+1} \rightarrow A$, define for $b \in A$

$$f_b(b_1, \ldots, b_k) = f(b, b_1, \ldots, b_k).$$

Each f_b is a k-ary function and has a representative $\underline{f_b}$. Define

$$\underline{f} = \lambda b \vec{b} . b(\underline{f_{a_1}}\ \vec{b}) \ldots (\underline{f_{a_n}}\ \vec{b}),$$

where $\vec{b} = b_2, \ldots, b_{k+1}$. Then

$$\begin{array}{rcll} \underline{f}\ \underline{b}_1 \ldots \underline{b}_{k+1} & = & \underline{b}_1\ (\underline{f_{a_1}}\ \underline{\vec{b}}) \ldots (\underline{f_{a_n}}\ \underline{\vec{b}}) & \\ & = & \underline{f_{b_1}}\ \underline{b}_2 \ldots \underline{b}_{k+1} & \\ & = & \underline{f_{b_1}(b_2, \ldots, b_{k+1})}, & \text{by the induction hypothesis ,} \\ & = & \underline{f(b_1, \ldots, b_{k+1})}, & \text{by definition of } f_{b_1}. \ \blacksquare \end{array}$$

One even can faithfully represent the full type structure over A as closed terms of $\lambda^o_{\rightarrow}$, see Exercise **??**.

Examples as free or finite algebras

The examples in the beginning of this section all can be viewed as free or finite algebras. The Booleans form a finite set and its representation is type 1_2. For this reason all Boolean functions can be represented. The natural numbers $\mathbb{N}$ and the trees T are examples of free algebras with a handicapped representation. Words over a finite alphabet $\Sigma = \{a_1, \ldots, a_n\}$ can be seen as an algebra with constant ϵ and further constructors $f_{a_i} = \lambda\!\!\lambda w.a_i w$. The representations given are particular cases of the theorems about free and finite algebras.

Pairing

In the untyped lambda calculus arbitrary two elements can be put in a pair

$$[M, N] \equiv \lambda z.zMN.$$

From the pairs the components can be retrieved:

$$[M_1, M_2](\lambda x_1 x_2.x_i) = M_i.$$

As a handicapped representation this works. But if we want to retrieve the elements from a pair of terms of different types, then this representation does not work for $\lambda^o_{\rightarrow}$, because no proper type can be given to z in order to make the projections work. If, however, M, N have the same type, then they can be paired using only β-conversion. This will enable us to represent also heterogenous pairs.

1.3.17. Lemma. *For every type $A \in \mathbb{T}^o$ there is a type $A \underline{\times} A \in \mathbb{T}^o$ such that there are terms $\mathtt{pair}^A_o, \mathtt{left}^A_o$ and $\mathtt{right}^A_o$ such that* $\mathrm{rk}(A \underline{\times} A) = \mathrm{rk}(A) + 2$ *and*

$$\begin{array}{rcl} \mathtt{pair}^A_o & \in & \Lambda^{\emptyset}_o(A{\rightarrow}A{\rightarrow}A \underline{\times} A); \\ \mathtt{left}^A_o & \in & \Lambda^{\emptyset}_o(A \underline{\times} A{\rightarrow}A); \\ \mathtt{right}^A_o & \in & \Lambda^{\emptyset}_o(A \underline{\times} A{\rightarrow}A), \end{array}$$

and for $M, N \in \Lambda^{\Gamma}_o(A)$ one has

$$\begin{array}{rcl} \mathtt{left}^A_o(\mathtt{pair}^A_o MN) & =_\beta & M; \\ \mathtt{right}^A_o(\mathtt{pair}^A_o MN) & =_\beta & N. \end{array}$$

Proof. Take

$$\begin{array}{rcl} A \times A & = & (A{\rightarrow}A{\rightarrow}A){\rightarrow}A; \\ \mathtt{pair}^A_o & = & \lambda mnz.zmn; \\ \mathtt{left}^A_o & = & \lambda p.p\mathsf{K}; \\ \mathtt{left}^A_o & = & \lambda p.p\mathsf{K}_*. \ \blacksquare \end{array}$$

Using $\beta\boldsymbol{\eta}$-conversion one can define a cartesian product for all pairs of types.

1.3.18. PROPOSITION (Gandy [1950?], Grzegorczyk [1964]). *Let $A, B \in \mathbb{T}^o$ be arbitrary types. Then there is a type $A \times B \in \mathbb{T}^o$ with*

$$\mathrm{rk}(A \times B) = \max\{\mathrm{rk}(A), \mathrm{rk}(B), 2\}$$

such that there are terms $\mathtt{pair}_o^{A,B}, \mathtt{left}_o^{A,B}$ and $\mathtt{right}_o^{A,B}$ such that

$$\begin{array}{rcl} \mathtt{pair}_o^{A,B} & \in & \Lambda_o^{\emptyset}(A{\to}A{\to}A \times B); \\ \mathtt{left}_o^{A,B} & \in & \Lambda_o^{\{z:o\}}(A \times B{\to}A); \\ \mathtt{right}_o^{A,B} & \in & \Lambda_o^{\{z:o\}}(A \times B{\to}B), \end{array}$$

and for $M \in \Lambda_o^{\Delta}(A), N \in \Lambda_o^{\Delta}(B)$ one has

$$\begin{array}{rcl} \mathtt{left}_o^{A,B}(\mathtt{pair}_o^{A,B}MN) & =_{\beta\boldsymbol{\eta}} & M; \\ \mathtt{right}_o^{A,B}(\mathtt{pair}_o^{A,B}MN) & =_{\beta\boldsymbol{\eta}} & N. \end{array}$$

PROOF. Write $n = \mathtt{arity}(A), m = \mathtt{arity}(B)$. Define

$$A \times B = A(1){\to}\ldots{\to}A(n){\to}B(1){\to}\ldots{\to}B(m){\to}o \mathbin{\underline{\times}} o,$$

where $o \mathbin{\underline{\times}} o = (o{\to}o{\to}o){\to}o$. Then

$$\begin{array}{rcl} \mathrm{rk}(A \times B) & = & \max\limits_{i,j}\{\mathrm{rk}(A_i)+1, \mathrm{rk}(B_j)+1, \mathrm{rk}(o^2{\to}o)+1\} \\ & = & \max\{\mathrm{rk}(A), \mathrm{rk}(B), 2\}. \end{array}$$

Define z_A inductively: $z_o = z; z_{A\to B} = \lambda a.z_B$. Then $z_A \in \Lambda_o^{z:o}(A)$. Write $\vec{x} = x_1, \ldots, x_n, \vec{y} = y_1, \ldots, y_m, \vec{z}_A = z_{A(1)}, \ldots, z_{A(n)}$ and $\vec{z}_B = z_{B(1)}, \ldots, z_{B(m)}$. Now define

$$\begin{array}{rcl} \mathtt{pair}_o^{A,B} & = & \lambda mn.\lambda\vec{x}\vec{y}.\mathtt{pair}_o^o(m\vec{x})(n\vec{y}); \\ \mathtt{left}_o^{A,B} & = & \lambda p.\lambda\vec{x}.\mathtt{left}_o^o(p\vec{x}\vec{z}_B); \\ \mathtt{right}_o^{A,B} & = & \lambda p.\lambda\vec{x}.\mathtt{right}_o^o(p\vec{z}_A\vec{y}). \end{array}$$

Then e.g. and

$$\begin{array}{rcl} \mathtt{left}_o^{A,B}(\mathtt{pair}_o^{A,B}MN) & =_\beta & \lambda\vec{x}.\mathtt{left}_o^o(\mathtt{pair}_o^oMN\vec{x}\vec{z}_B) \\ & =_\beta & \lambda\vec{x}.\,\mathtt{left}_o^o[\mathtt{pair}_o^o(M\vec{x})(N\vec{z}_B)] \\ & =_\beta & \lambda\vec{x}.(M\vec{x}) \\ & =_{\boldsymbol{\eta}} & M. \ \blacksquare \end{array}$$

In Barendregt [1974] it is proved that $\boldsymbol{\eta}$-conversion is essential: with β-conversion one can pair only certain combinations of types. Also it is shown that there is no *surjective* pairing in the theory with $\beta\boldsymbol{\eta}$-conversion. Surjectivity states that

$$\mathtt{pair}(\mathtt{left}z)(\mathtt{right}z) =_{\beta\boldsymbol{\eta}} z.$$

In Section 5.2 we will discuss systems extended with surjective pairing. With similar techniques as in mentioned paper it can be shown that in $\lambda_{\to}^{\infty}$ there is no pairing function $\mathtt{pair}_o^{\alpha,\beta}$ for base types (even with $\boldsymbol{\eta}$-conversion). In section 2.3 we will encounter other differences between $\lambda_{\to}^{\infty}$ and $\lambda_{\to}^{o}$.

1.3.19. PROPOSITION. *Let $A_1,\ldots,A_n \in \mathbb{T}^o$. There are closed terms*

$$\begin{array}{rcl} \mathtt{tuple}^n & : & A_1\to\ldots\to A_n\to(A_1\times\ldots\times A_n) \\ \mathtt{proj}^n_k & : & A_1\times\ldots\times A_n\to A_k \end{array}$$

such that for $M_1,\ldots,M_n$ of the right type one has

$$\mathtt{proj}^n_k(\mathtt{tuple}^n M_1\ldots M_n) =_{\beta\boldsymbol{\eta}} M_k.$$

If there is little danger of confusion and the $\vec{M}, N$ are of the right type we write

$$\begin{array}{rcl} \langle M_1,\ldots,M_n\rangle & \equiv & \mathtt{tuple}^n M_1\ldots M_n; \\ N\cdot k & \equiv & \mathtt{proj}^n_k N. \end{array}$$

PROOF. By iterating pairing. ■

1.4. The $\lambda_\to$ systems *à la* Curry, *à la* Church and *à la* de Bruijn

The Curry version of $\lambda_\to$ is called *implicitly typed* because an expression like

$$\lambda x.x\mathsf{K}$$

has a type, but it requires work to find it. In §2.2 we will see that this work is feasible. In systems more complex than $\lambda_\to$ finding types in the implicit version is more complicated and may even not be computable. This will be the case with second and higher order types, like $\lambda 2$ (system F), treated in Volume II.

Therefore there are also versions of the systems that are typed explicitly *à la Church*. The explicitly typed version of $\lambda_\to$ will be denoted by $\lambda^{\mathbb{A}}_{\to\mathrm{Ch}}$ or more often by $\lambda^{\mathrm{Ch}}_{\to}$. In this system one writes for example for the term above

$$\lambda x^{(A\to B\to A)\to C}.x^{(A\to B\to A)\to C}(\lambda y^A z^B.y^A).$$

In the system *á la* de Bruijn this is written as

$$\lambda x{:}((A\to B\to A)\to C).x(\lambda y{:}A\lambda z{:}B.y).$$

So in both cases terms are not just elements of Λ, but versions ornamented with elements of $\mathbb{T}$.

1.4.1. DEFINITION. Let $\mathbb{A}$ be a set of type atoms. The Church version of $\lambda_\to$, notation $\lambda^{\mathbb{A}}_{\to\mathrm{Ch}}$ or $\lambda^{\mathrm{Ch}}_{\to}$ if $\mathbb{A}$ is not emphasized, is defined as follows. The system has the same set of types $\mathbb{T}^{\mathbb{A}}$ as $\lambda^{\mathbb{A}}_{\to\mathrm{Cu}}$.

(i) The set of term variables is different: each such variable is coupled with a unique type. Let

$$\mathsf{V}^{\mathbb{T}} = \mathsf{V}\times\mathbb{T}.$$

(ii) Notation. $\mathbf{x},\mathbf{y},\mathbf{z},\ldots$ range over $\mathsf{V}^{\mathbb{T}}$. If $\mathbf{x}=\langle x,A\rangle$, then we also write

$$\mathbf{x} = x^A = x{:}A.$$

(iii) Terms of type A, notation $\Lambda^{\mathrm{Ch}}_{\to}(A)$, are defined as follows.

$$\begin{array}{rcl} & & x^A \in \Lambda_{\to}(A); \\ M \in \Lambda_{\to}(A{\to}B), N \in \Lambda_{\to}(A) & \Rightarrow & (MN) \in \Lambda_{\to}(B); \\ M \in \Lambda_{\to}(B) & \Rightarrow & (\lambda x^A.M) \in \Lambda_{\to}(A{\to}B). \end{array}$$

Figure 1.6: The system $\lambda^{\mathrm{Ch}}_{\to}$ of typed terms *á la* Church

(iv) The set of terms of $\lambda^{\mathrm{Ch}}_{\to}$, notation $\Lambda^{\mathrm{Ch}}_{\to}$, is defined as

$$\Lambda^{\mathrm{Ch}}_{\to} = \bigcup_{A \in \mathbb{T}} \Lambda^{\mathrm{Ch}}_{\to}(A).$$

For example

$$\begin{array}{rcl} y^{B\to A}x^B & \in & \Lambda^{\mathrm{Ch}}_{\to}(A) \\ \lambda x^A.y^{B\to A} & \in & \Lambda^{\mathrm{Ch}}_{\to}(A{\to}B{\to}A) \\ \lambda x^A.x^A & \in & \Lambda^{\mathrm{Ch}}_{\to}(A{\to}A) \end{array}$$

Substitution of a term N for a typed variable x^B is defined as usual. If also N has type B, then the resulting term keeps its type.

1.4.2. Proposition. (i) *Let* $M \in \Lambda_{\to}(A)$, $N \in \Lambda_{\to}(B)$. *Then*

$$(M[x^B := N]) \in \Lambda_{\to}(A).$$

(ii) *Let* $A, B \in \mathbb{T}$. *Then*

$$M \in \Lambda^{\mathrm{Ch}}_{\to}(A) \;\Rightarrow\; M[\alpha := B] \in \Lambda^{\mathrm{Ch}}_{\to}(A[\alpha := B]).$$

Proof. (i), (ii) By induction on the structure of M. ■

1.4.3. Definition. On $\Lambda^{\mathrm{Ch}}_{\mathbb{A}}$ we define the following notions of reduction.

$$\begin{array}{rcll} (\lambda x^A.M)N & \to & M[x{:=}N] & (\beta) \\ \lambda x{:}A.Mx & \to & M, \quad \text{if } x \notin \mathrm{FV}(M) & (\boldsymbol{\eta}) \end{array}$$

Figure 1.7: $\beta\boldsymbol{\eta}$-contraction rules for $\lambda^{\mathbb{A}}_{\to\mathrm{Ch}}$

As usual, see B[1984], these notions of reduction generate the corresponding reduction relations. Also there are the corresponding conversion relations $=_\beta$, $=_{\boldsymbol{\eta}}$ and $=_{\beta\boldsymbol{\eta}}$. Terms in $\lambda^{\mathrm{Ch}}_{\to}$ will often be considered modulo $=_\beta$ or $=_{\beta\boldsymbol{\eta}}$. The notation $M = N$, means $M =_{\beta\boldsymbol{\eta}} N$ by default.

For every type A the set $\Lambda^{\mathrm{Ch}}_{\to}(A)$ is closed under reduction.

1.4.4. Proposition. (i) $((\lambda x^B.M)N) \in \Lambda^{\mathrm{Ch}}_{\to}(A) \;\Rightarrow\; M^A[x := N^B] \in \Lambda^{\mathrm{Ch}}_{\to}(A)$.

(ii) $\lambda x^B.Mx^B \in \Lambda^{\mathrm{Ch}}_{\to}(A)$ *and* $x^B \notin \mathrm{FV}(M)$, *then* $M \in \Lambda^{\mathrm{Ch}}_{\to}(A)$.

(iii) $M \in \Lambda^{\mathrm{Ch}}_{\to}$ *and* $M \twoheadrightarrow_{\beta\boldsymbol{\eta}} N$. *Then* $N \in \Lambda^{\mathrm{Ch}}_{\to}$.

PROOF. (i) If $(\lambda x^B.M)N \in \Lambda^{\mathrm{Ch}}_{\to}(A)$, then $\lambda x^B.M \in \Lambda^{\mathrm{Ch}}_{\to}(B'\to A)$, so $B = B'$, and $N \in \Lambda^{\mathrm{Ch}}_{\to}(B)$. Now Lemma 1.4.2 applies.

(ii) Similarly. If $(\lambda x^B.Mx^B) \in \Lambda^{\mathrm{Ch}}_{\to}(C)$, then $C = B\to A$ and $Mx^B \in \Lambda^{\mathrm{Ch}}_{\to}(A)$. But then $M \in \Lambda^{\mathrm{Ch}}_{\to}(B\to A)$.

(iii) By induction on the relation $\twoheadrightarrow_{\beta\boldsymbol{\eta}}$, using (i), (ii). ■

The reasoning needed in the proof of this Lemma is made more explicit in the generation lemmas in Section 2.1.

Relating the Curry and Church systems

There are canonical translations between $\lambda^{\mathrm{Ch}}_{\to}$ and $\lambda^{\mathrm{Cu}}_{\to}$.

1.4.5. DEFINITION. There is a 'forgetful' map $|\cdot| : \Lambda_{\mathbb{A}} \to \Lambda$ defined as follows:

$$\begin{array}{rcl} |x^A| & \equiv & x; \\ |MN| & \equiv & |M||N|; \\ |\lambda x{:}A.M| & \equiv & \lambda x.|M|. \end{array}$$

The map $|\cdot|$ just erases all type ornamentations of a term in $\Lambda^{\mathrm{Ch}}_{\to}$. The following result states that ornamented legal terms in the Church version 'project' to legal terms in the Curry version of $\lambda_\to$. Conversely, legal terms in $\lambda^{\mathrm{Cu}}_{\to}$ can be 'lifted' to legal terms in $\lambda^{\mathrm{Ch}}_{\to}$. There is however a condition needed. The term

$$\lambda x^{A\to A\to B}\lambda x^A.x^{A\to B}x^Ax^A \in \Lambda((A\to A\to B)\to(A\to B))$$

projects to $\lambda x\lambda x.xxx$ which does not have a type.

1.4.6. DEFINITION. (i) A collection of type variables $\mathcal{X} \subseteq \mathsf{V}^{\mathsf{T}}$ is called well-named iff $x^A, x^B \in \mathcal{X} \;\Rightarrow\; A = B$. (If we consider x as the first name of x^A and A as its family name, then the notion that $\mathcal{X}$ is well named means that first names uniquely describe the variable.)

(ii) A term is *well-named* if $\mathrm{FV}(M)$ is well-named.

1.4.7. PROPOSITION. (i) *Let $M \in \Lambda_{\mathbb{A}}$ be well-named. Then*

$$M \in \Lambda^{\mathrm{Ch}}_{\mathbb{A}}(A) \;\Rightarrow\; \Gamma_M \vdash^{\mathrm{Cu}}_{\lambda_\to} |M| : A,$$

where $\Gamma_M = \{x{:}A \mid x^A \in \mathrm{FV}(M)\}$

(ii) *Let $M \in \Lambda$. Then*

$$\Gamma \vdash^{\mathrm{Cu}}_{\lambda_\to} M : A \iff \exists M' \in \Lambda^{\mathrm{Ch}}_{\mathbb{A}}(A).|M'| \equiv M \text{ and } M' \text{ is well-named.}$$

PROOF. (i) By induction on the generation of $\Lambda^{\mathrm{Ch}}_{\mathbb{A}}$. The assumption that M is well-named insures that Γ_M is well-defined and that $\Gamma_P \cup \Gamma_Q = \Gamma_{PQ}$.

(ii) ($\Rightarrow$) By induction on the proof of $\Gamma \vdash M : A$ with the induction loading that $\Gamma_{M'} = \Gamma$. ($\Leftarrow$) By (i). ■

Notice that the converse of Proposition 1.4.7(i) is not true: one has

$$\vdash^{\mathrm{Cu}}_{\lambda_{\to}} |\lambda x^A \lambda y^A.x^A y^A| : (A{\to}A){\to}(A{\to}A),$$

but $(\lambda x^A \lambda y^A.x^A y^A) \notin \Lambda^{\mathrm{Ch}}((A{\to}A){\to}(A{\to}A))$.

1.4.8. Corollary. *In particular, for a type $A \in \mathbb{T}$ one has*

$$A \text{ is inhabited in } \lambda^{\mathrm{Cu}}_{\to} \iff A \text{ is inhabited in } \lambda^{\mathrm{Ch}}_{\to}.$$

Proof. Immediate. ■

The translation preserves reduction and conversion.

1.4.9. Proposition. *Let $R = \beta, \boldsymbol{\eta}$ or $\beta\boldsymbol{\eta}$. Then*

(i) *Let $M, N \in \Lambda^{\mathrm{Ch}}_{\to}$. Then*

$$M \to_R N \;\Rightarrow\; |M| \to_R |N|.$$

(ii) *Let $M_1, M_2 \in \Lambda^{\Gamma}_{\to \mathrm{Cu}}(A)$, $M_1 = |M_1{}'|$, with $M_1' \in \Lambda^{\mathrm{Ch}}_{\to}(A)$. Then*

$$M_1 \to_R M_2 \quad \Rightarrow \quad \exists M_2{}' \in \Lambda^{\mathrm{Ch}}_{\to}(A). \; |M_i{}'| \equiv M_i \;\&\; M_1{}' \to_R M_2{}'.$$

(iii) *The same results hold for $\twoheadrightarrow_R$ and R-conversion.*

Proof. Easy. ■

The systems $\lambda_{\to}$ *à la* de Bruijn

There is the following disadvantage about the Church systems. Consider

$$\mathsf{I} \equiv \lambda x^A.x^A.$$

In the next volume we will consider dependent types coming from the Authomath language family, see Nederpelt et al. [1994], designed for formalizing arguments and proof-checking. These are types that depend on a term variable (ranging over another type). An intuitive example is A^n, where n is a variable ranging over natural numbers. A more formal example is Px, where $x : A$ and $P : A{\to}\mathbb{T}$. In this way types may contain redexes and we may have the following reduction

$$\mathsf{I} \equiv \lambda x^A.x^A \to_\beta \lambda x^A.x^{A'},$$

by reducing only the second A to A'. The question now is whether λx^A binds the $x^{A'}$. If we write I as

$$\mathsf{I} \equiv \lambda x{:}A.x,$$

then this problem disappears

$$\lambda x{:}A.x \twoheadrightarrow \lambda x{:}A'.x.$$

In the following system $\lambda^{\mathbb{A}}_{\to \mathrm{dB}}$ this idea is formalized.

1.4.10. DEFINITION. The system $\lambda^{\mathbb{A}}_{\to \mathrm{dB}}$ starts with a collection of *pseudo-terms*, notation $\Lambda^{\mathrm{dB}}_{\to}$, is defined by the following grammar.

$$\Lambda_{\mathbb{A}} \quad = \quad \mathbb{A} \,|\, \Lambda_{\mathbb{A}}\, \Lambda_{\mathbb{A}} \,|\, \lambda\, \mathbb{A}{:}\mathbb{T}.\Lambda_{\mathbb{A}}.$$

For example $\lambda x{:}\alpha.x$ and $(\lambda x{:}\alpha.x)(\lambda y{:}\beta.y)$ are pseudo-terms. As we will see, the first one is a *legal*, i.e. actually typeable term in $\lambda^{\mathbb{A}}_{\to \mathrm{dB}}$, whereas the second one is not.

1.4.11. DEFINITION. (i) Let Γ be a basis consisting of a set of declarations $x{:}A$ with distinct term variables x and types $A \in \mathbb{T}^{\mathbb{A}}$. This is exactly the same as for $\lambda^{\mathbb{A}}_{\to \mathrm{Cu}}$.

(ii) The system of type assignment obtaining statements $\Gamma \vdash M : A$ with Γ a basis, M a pseudoterm and A a type, is defined as follows.

(axiom)	$\Gamma \vdash x : A,$	if $(x{:}A) \in \Gamma$;
($\to$-elimination)	$\dfrac{\Gamma \vdash M : (A \to B) \quad \Gamma \vdash N : A}{\Gamma \vdash (MN) : B}$;	
($\to$-introduction)	$\dfrac{\Gamma, x{:}A \vdash M : B}{\Gamma \vdash (\lambda x{:}A.M) : (A \to B)}$.	

Figure 1.8: $\lambda^{\mathrm{dB}}_{\to}$

Provability in $\lambda^{\mathrm{dB}}_{\to}$ is denoted by $\vdash^{\mathrm{dB}}_{\lambda_\to}$. Thus the legal terms of $\lambda^{\mathrm{dB}}_{\to}$ are defined by making a selection from the context-free language $\Lambda^{\mathrm{dB}}_{\to}$. That $\lambda x{:}\alpha.x$ is legal folows from $x{:}\alpha \vdash^{\mathrm{dB}}_{\lambda_\to} x : \alpha$ using the $\to$-introduction rule. That $(\lambda x{:}\alpha.x)(\lambda y{:}\beta.y)$ is not follows systematically from section 2.3. These legal terms do not form a context-free language, do exercise 1.5.8.

There is a close connection between $\lambda^{\mathrm{Ch}}_{\to}$ and $\lambda^{\mathrm{dB}}_{\to}$. First we need the following.

1.4.12. LEMMA. *Let $\Gamma \subseteq \Gamma'$ be bases of $\lambda^{\mathrm{dB}}_{\to}$. Then*

$$\Gamma \vdash^{\mathrm{dB}}_{\lambda_\to} M : A \;\Rightarrow\; \Gamma' \vdash^{\mathrm{dB}}_{\lambda_\to} M : A.$$

PROOF. By induction on the derivation of the first statement. ■

1.4.13. DEFINITION. (i) Let $M \in \Lambda^{\mathrm{Ch}}_{\to}$ and suppose $\mathrm{FV}(M) \subseteq \mathrm{dom}(\Gamma)$. Define M^Γ inductively as follows.

$$\begin{aligned} x^\Gamma &= x^{\Gamma(x)}; \\ (MN)^\Gamma &= M^\Gamma N^\Gamma; \\ (\lambda x{:}A.M)^\Gamma &= \lambda x^A.M^{\Gamma, x:A}. \end{aligned}$$

(ii) Let $M \in \Lambda^{\mathrm{Ch}}_{\to}(A)$ in $\lambda^{\mathrm{Ch}}_{\to}$. Define M^-, a pseudo-term of $\lambda^{\mathrm{dB}}_{\to}$, as follows.

$$\begin{array}{rcl}(x^A)^- &=& x;\\ (MN)^- &=& M^-N^-;\\ (\lambda x^A.M)^- &=& \lambda x{:}A.M^-.\end{array}$$

(iii) For M a term of $\lambda^{\mathrm{dB}}_{\to}$ define the basis Γ_M as follows.

$$\begin{array}{rcl}\Gamma_{x^A} &=& \{x{:}A\};\\ \Gamma_{MN} &=& \Gamma_M \cup \Gamma_N;\\ \Gamma_{\lambda x^A.M} &=& \Gamma_M \setminus \{x{:}A\}\end{array}$$

1.4.14. Example. To get the (easy) intuition, consider the following.

$$\begin{array}{rcl}(\lambda x{:}A.x)^{\emptyset} &\equiv& (\lambda x^A.x^A);\\ (\lambda x^A.x^A)^- &\equiv& (\lambda x{:}A.x);\\ (\lambda x{:}A{\to}B.xy)^{\{y:A\}} &\equiv& \lambda x^{A\to B}.x^{A\to B}y^A;\\ \Gamma_{(\lambda x^{A\to B}.x^{A\to B}y^A)} &=& \{y{:}A\}.\end{array}$$

1.4.15. Proposition. (i) *Let $M \in \Lambda^{\mathrm{Ch}}_{\mathbb{A}}$ and let M be well-named. Let Γ a basis of $\lambda^{\mathrm{dB}}_{\to}$. Then*

$$M \in \Lambda^{\mathrm{Ch}}_{\mathbb{A}}(A) \iff \Gamma_M \vdash^{\mathrm{dB}}_{\lambda_\to} M^- : A.$$

(ii) $$\Gamma \vdash^{\mathrm{dB}}_{\lambda_\to} M : A \iff M^\Gamma \in \Lambda^{\mathrm{Ch}}_{\to}(A).$$

Proof. (i), (ii)($\Rightarrow$) By induction on the proof or the definition of the LHS.
(i)($\Leftarrow$) By (ii)($\Rightarrow$), using $(M^-)^{\Gamma_M} \equiv M$.
(ii)($\Leftarrow$) By (i)($\Rightarrow$), using $(M^\Gamma)^- \equiv M, \Gamma_{M^\Gamma} \subseteq \Gamma$ and proposition 1.4.12. ■

1.4.16. Corollary. *In particular, for a type $A \in \mathbb{T}$ one has*

$$A \text{ is inhabited in } \lambda^{\mathrm{Ch}}_{\to} \iff A \text{ is inhabited in } \lambda^{\mathrm{dB}}_{\to}.$$

Proof. Immediate. ■

Again the translation preserves reduction and conversion

1.4.17. Proposition. (i) *Let $M, N \in \Lambda^{\Gamma}_{\to\mathrm{Ch}}$. Then*

$$M \to_R N \iff M^\Gamma \to_R N^\Gamma,$$

where $R = \beta, \boldsymbol{\eta}$ or $\beta\boldsymbol{\eta}$.
(ii) *Let $M_1, M_2 \in \Lambda(A)$ and R as in (i). Then*

$$M_1 \to_R M_2 \iff M_1^- \to_R M_2^-.$$

(iii) *The same results hold for conversion.*

Proof. Easy. ■

Comparing $\lambda_\to^{\mathrm{dB}}$ and $\lambda_\to^{\mathrm{Cu}}$

1.4.18. PROPOSITION. (i) *Let $M \in \Lambda_{\mathbb{A}}$ be well-named. Then*

$$M \in \Lambda_\to^{\mathrm{dB}}(A) \;\Rightarrow\; \Gamma_M \vdash_{\lambda_\to}^{\mathrm{Cu}} |M| : A,$$

where $\Gamma_M = \{x{:}A \mid x^A \in \mathrm{FV}(M)\}$

(ii) *Let $M \in \Lambda$. Then*

$$\Gamma \vdash_{\lambda_\to}^{\mathrm{Cu}} M : A \iff \exists M' \in \Lambda_{\mathbb{A}}^{\mathrm{dB}}(A).|M'| \equiv M \text{ and } M' \text{ is well-named.}$$

PROOF. As for Proposition 1.4.7. ■

Again the implication in (i) cannot be reversed.

The three systems compared

Now we can harvest a comparison between the three systems $\lambda_\to^{\mathrm{Ch}}$, $\lambda_\to^{\mathrm{dB}}$ and $\lambda_\to^{\mathrm{Cu}}$.

1.4.19. THEOREM. *Let M be a well-named term in $\Lambda_{\Pi^{\mathbb{A}}}$. Then we have*

$$\begin{aligned} M \in \Lambda_{\mathbb{A}}^{\mathrm{Ch}}(A) \quad &\iff \quad \Gamma_M \vdash_{\lambda_\to}^{\mathrm{dB}} M^- : A \\ &\Rightarrow \quad \Gamma_M \vdash_{\lambda_\to}^{\mathrm{Cu}} |M| : A \\ &\Rightarrow \quad M' \in \Lambda_\to^{\mathrm{Ch}}(A).|M'| \equiv M \;\&\; M' \text{ is well-named.} \end{aligned}$$

PROOF. By Propositions 1.4.7 and 1.4.18 and the fact that $|M^-| = |M|$. ■

Again the second and third arrows cannot be reversed.

It may seem a bit exaggerated to have three versions of the simply typed lambda calculus: $\lambda_\to^{\mathrm{Cu}}$, $\lambda_\to^{\mathrm{Ch}}$ and $\lambda_\to^{\mathrm{dB}}$. But this is indeed necessary.

The Curry version corresponds to implicitly typed programming languages like ML. Since implicit typing makes programming more easy, we want to consider this system.

For extensions of $\lambda_\to^{\mathrm{Cu}}$, like $\lambda 2$ with second order (polymorphic) types, type checking is not decidable, see Wells [1999], and hence one needs the explicit versions. The two explicitly typed systems $\lambda_\to^{\mathrm{Ch}}$ and $\lambda_\to^{\mathrm{dB}}$ are basically isomorphic as shown above. These systems have a very canonical semantics if the version $\lambda_\to^{\mathrm{Ch}}$ is used. Nevertheless we want two versions because the version $\lambda_\to^{\mathrm{dB}}$ can be extended more naturally to more powerful type systems in which there is a notion of reduction on the types (those with 'dependent types' and those with higher order types to be treated in Volume II) generated simultaneously. Also there are important extensions in which there is a reduction relation on types, e.g. in the system $\lambda\omega$ with higher order types. The classical version of $\lambda_\to$ gives problems. For example, if $A \twoheadrightarrow B$, does one have that $\lambda x^A.x^A \twoheadrightarrow \lambda x^A.x^B$? Moreover, is the x^B bound by the λx^A? By denoting $\lambda x^A.x^A$ as $\lambda x{:}A.x$, as is done in $\lambda_\to^{\mathrm{Ch}}$, these problems do not arise. This is so important that for explicitly typed extensions of $\lambda_\to$ we will need to use the Ch-versions, even if for these systems the model theory will be a bit more complicated to express.

The situation is not so bad as it may seem, since the three systems and their differences are easy to memorize. Just look at the following examples.

$$\begin{array}{rcll} \lambda x.xy & \in & \Lambda^{\{y:o\}}_{\to \mathrm{Cu}}((o\to o)\to o)) & \text{(Curry);} \\ \lambda x{:}(o\to o).xy & \in & \Lambda^{\{y:o\}}_{\to \mathrm{dB}}((o\to o)\to o)) & \text{(de Bruijn);} \\ \lambda x^{o\to o}.x^{o\to o}y^{o} & \in & \Lambda_{\to \mathrm{Ch}}((o\to o)\to o)) & \text{(Church).} \end{array}$$

We have chosen to present the three versions of $\lambda_{\to}$ because one finds them like this in the literature.

In this Part I of the book we are interested in untyped lambda terms that can be typed using simple types. We will see that up to substitution this typing is unique. For example

$$\lambda fx.f(fx)$$

can have as type $(o\to o)\to o\to o$, but also $(A\to A)\to A\to A$ for any type A. Moreover there is a simple algorithm to find all possible types for an untyped lambda term, see Section 2.3.

We are interested in typable terms M. This can be terms in the set of untyped lambda terms Λ with Curry typing. Since we are at the same time also interested in the types of the subterms of M, the Church typing is a convenient notation. Moreover, this information is almost uniquely determined once the type A of M is known or required. By this we mean that the Church typing is uniquely determined by A for the nf M^{nf} of M, if M contains some redexes of the form $(\lambda x.M)N$ with $x\notin \mathrm{FV}(M)$ (K-redexes), otherwise for M itself. For example the Church typing of $M\equiv \mathsf{K}\mathsf{I}y$ of type $\alpha\to\alpha$ is $(\lambda x^{\alpha\to\alpha}y^{\beta}.x^{\alpha\to\alpha})(\lambda z^{\alpha}.z^{\alpha})y^{\beta}$. The type β is not determined. But for the $\beta\boldsymbol{\eta}$-nf of M, the term I, the Church typing is is uniquely determined and is $\mathsf{I}_{\alpha}\equiv\lambda z^{\alpha}.z^{\alpha}$. See Exercise 2.5.3.

If A is not available, then the type information for M is schematic in the groundtypes. By this we mean that e.g. the term $\mathsf{I}\equiv\lambda x.x$ has a Church version $\lambda x^{\alpha}.x^{\alpha}$ and type $\alpha\to\alpha$, where one can substitute both in the term and type any $A\in\mathbb{T}^{\mathbb{A}}$ for α. We will study this in greater detail in Section 2.3.

1.5. Exercises

1.5.1. Show that the well-known combinators

$$\begin{array}{rcl} \mathsf{I} & \equiv & \lambda x.x, \\ \mathsf{K} & \equiv & \lambda xy.x, \\ \mathsf{S} & \equiv & \lambda xyz.xz(yz) \end{array}$$

respectively have the following types.

$$\begin{array}{rcl} \mathsf{I} & : & A\to A; \\ \mathsf{K} & : & A\to B\to A; \\ \mathsf{S} & : & (A\to B\to C)\to(A\to B)\to(A\to C). \end{array}$$

1.5.2. Find types for

$$\begin{array}{lcl} \mathsf{B} & \equiv & \lambda xyz.x(yz); \\ \mathsf{C} & \equiv & \lambda xyz.xzy; \\ \mathsf{C}_* & \equiv & \lambda xy.yx; \\ \mathsf{K}_* & \equiv & \lambda xy.y; \\ \mathsf{W} & \equiv & \lambda xy.xyy. \end{array}$$

1.5.3. Find types for SKK, $\lambda xy.y(\lambda z.zxx)x$ and $\lambda fx.f(f(fx))$.

1.5.4. Show that $\mathrm{rk}(A{\to}B{\to}C) = \max\{\mathrm{rk}(A)+1, \mathrm{rk}(B)+1, \mathrm{rk}(C)\}$.

1.5.5. Show that if $M \equiv P[x := Q]$ and $N \equiv (\lambda x.P)Q$, then M may have a type in $\lambda^{\mathrm{Cu}}_{\to}$ but N not. A similar observation can be made for pseudo-terms of $\lambda^{\mathrm{dB}}_{\to}$.

1.5.6. Show the following.
(i) $\lambda xy.(xy)x \notin \Lambda^{\emptyset}_{\to\mathrm{Cu}}$.
(ii) $\lambda xy.x(yx) \in \Lambda^{\emptyset}_{\to\mathrm{Cu}}$.

1.5.7. Find inhabitants in $(A{\to}B{\to}C){\to}B{\to}A{\to}C$ and $(A{\to}A{\to}B){\to}A{\to}B$.

1.5.8. [van Benthem] Show that $\Lambda_{\to\mathrm{Ch}}(A)$ and $\Lambda^{\emptyset}_{\to\mathrm{Cu}}(A)$ is for some $A \in \mathbb{TT}^{\mathbb{A}}$ not a context free language.

1.5.9. Define in $\lambda^o_{\to}$ the 'pseudo-negation' $\sim A \equiv A{\to}o$. Construct an inhabitant of $\sim\sim\sim A{\to}\sim A$.

1.5.10. Let X be a finite set. Give a representation of X in $\lambda^o_{\to}$ such that every function $f{:}X^k \to X$ can be lambda defined in $\lambda^o_{\to}$.

1.5.11. Prove the following, see definition 1.4.13.
(i) Let $M \in \Lambda_{\to\mathrm{Ch}}$ be such that $\mathrm{FV}(M) \subseteq \mathrm{dom}(\Gamma)$, then $(M^{\Gamma})^{-} \equiv M$ and $\Gamma_{M^{\Gamma}} \subseteq \Gamma$.
(ii) Let $M \in \Lambda_{\to\mathrm{dB}}$, then $(M^{-})^{\Gamma_M} \equiv M$.

1.5.12. Construct a term F with $\vdash_{\lambda^o_{\to}} F : T_2{\to}T_2$ such that for trees t one has $F\underline{t} =_\beta \underline{t^{\mathrm{mir}}}$, where t^{mir} is the mirror image of t, defined by

$$\begin{array}{rcl} \epsilon^{\mathrm{mir}} & = & \epsilon; \\ (p(t,s))^{\mathrm{mir}} & = & p(s^{\mathrm{mir}}, t^{\mathrm{mir}}). \end{array}$$

1.5.13. A term M is called *proper* if all λ's appear in the prefix of M, i.e. $M \equiv \lambda\vec{x}.N$ and there is no λ occurring in N. Let A be a type such that $\Lambda^{\emptyset}(A)$ is not empty. Show that

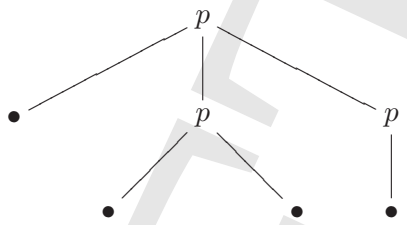
$$\text{Every nf of type } A \text{ is proper} \iff \mathrm{rk}(A) \leq 2.$$

1.5.14. Determine the class of closed inhabitants of the types 4 and 5.

1.5.15. The collection of multi-ary trees can be seen as part of a multi-sorted algebra with sorts MTree and L_{MTree} as follows.

$$\begin{array}{rcl} \mathrm{nil} & \in & L_{\mathrm{Mtree}}; \\ \mathrm{cons} & \in & \mathrm{Mtree}{\to}L_{\mathrm{Mtree}}{\to}L_{\mathrm{Mtree}}; \\ p & \in & \mathrm{L}_{\mathrm{Mtree}}{\to}\mathrm{Mtree}. \end{array}$$

Represent this multi-sorted free algebra in $\lambda^o_{\rightarrow}$. Construct the lambda term representing the tree



1.5.16. In this exercise it will be proved that each term (having a β-nf) has a unique lnf. A term M (typed or untyped) is always of the form $\lambda x_1 \ldots x_n.yM_1 \ldots M_m$ or $\lambda x_1 \ldots x_r.(\lambda x.M_0)M_1 \ldots M_s$. Then $yM_1 \ldots M_m$ (or $(\lambda x.M_0)M_1 \ldots M_m$) is the *matrix* of M and the $(M_0,)M_1, \ldots, M_m$ are its *components*. A typed term $M \in \Lambda^{\Gamma}(A)$ is said to be *fully eta* (f.e.) *expanded* if its matrix is of type o and its components are f.e. expanded. Show the following for typed terms. (For untyped terms there is no finite f.e. expanded form, but the Nakajima tree, see B[1984] Exercise 19.4.4, is the corresponding notion for the untyped terms.)

(i) M is in lnf iff M is a β-nf and f.e. expanded.

(ii) If $M =_{\beta\boldsymbol{\eta}} N_1 =_{\beta\boldsymbol{\eta}} N_2$ and N_1, N_2 are β-nfs, then $N_1 =_{\eta} N_2$. [Hint. Use η-postponement, see B[1984] Proposition 15.1.5.]

(iii) $N_1 =_{\eta} N_2$ and N_1, N_2 are β-nfs, then there exist $N\downarrow$ and $N\uparrow$ such that $N_i \twoheadrightarrow_{\eta} N\downarrow$ and $N\uparrow \twoheadrightarrow_{\eta} N_i$, for $i = 1, 2$. [Hint. Show that both $\rightarrow_{\eta}$ and ${}_{\eta}\leftarrow$ satisfy the diamond lemma.]

(iv) If M has a β-nf, then it has a unique lnf.

(v) If N is f.e. expanded and $N \twoheadrightarrow_{\beta} N'$, then N' is f.e. expanded.

(vi) For all M there is a f.e. expanded M^* such that $M^* \twoheadrightarrow_{\eta} M$.

(vii) If M has a β-nf, then the lnf of M is the β-nf of M^*, its full eta expansion.

1.5.17. Like in B[1984], the terms in this book are *abstract terms*, considered modulo α-conversion. Sometimes it is useful to be explicit about α-conversion and even to violate the variable convention that in a subterm of a term the names of free and bound variables should be distinct. For this it is useful to modify the system of type assignment.

(i) Show that $\vdash_{\lambda\rightarrow}$ is not closed under α-conversion. I.e.

$$\Gamma \vdash M{:}A, M \equiv_{\alpha} M' \not\Rightarrow \Gamma \vdash M'{:}A.$$

[Hint. Consider $M' \equiv \lambda x.x(\lambda x.x)$.]

(ii) Consider the following system of type assignment to untyped terms.

$$\{x{:}A\} \vdash x : A;$$

$$\frac{\Gamma_1 \vdash M : (A{\to}B) \quad \Gamma_2 \vdash N : A}{\Gamma_1 \cup \Gamma_2 \vdash (MN) : B}, \quad \text{provided } \Gamma_1 \cup \Gamma_2 \text{ is a basis;}$$

$$\frac{\Gamma \vdash M : B}{\Gamma - \{x{:}A\} \vdash (\lambda x.M) : (A \to B)}, \quad \text{provided } \Gamma \cup \{x{:}A\} \text{ is a basis.}$$

Provability in this system will be denoted by $\Gamma \vdash' M : A$.

(iii) Show that $\vdash'$ is closed under α-conversion.

(iv) Show that

$$\Gamma \vdash' M : A \iff \exists M' \equiv_\alpha M. \Gamma \vdash M' : A.$$

1.5.18. (i) Let $M = \lambda x_1 \ldots x_n.x_i M_1 \ldots M_m$ be a β-nf. Define by induction on the length of M its Φ-*normal form*, notation $\Phi(M)$, as follows.

$$\Phi(\lambda \vec{x}.x_i M_1 \ldots M_m) := \lambda \vec{x}.x_i (\Phi(\lambda \vec{x}.M_1)\vec{x}) \ldots (\Phi(\lambda \vec{x}.M_m)\vec{x}).$$

(ii) Compute the Φ-nf of $\mathsf{S} = \lambda xyz.xz(yz)$.

(iii) Write $\Phi^{n,m,i} := \lambda y_1 \ldots y_m \lambda x_1 \ldots x_n.x_i(y_1\vec{x}) \ldots (y_m\vec{x})$. Then

$$\Phi(\lambda \vec{x}.x_i M_1 \ldots M_m) = \Phi^{n,m,i}(\Phi(\lambda \vec{x}.M_1)) \ldots (\Phi(\lambda \vec{x}.M_m)).$$

Show that the $\Phi^{n,m,i}$ are typable.

(iv) Show that every closed nf of type A can be written as a product of the $\Phi^{n,m,i}$.

(v) Write S in such a manner.

Chapter 2

Properties

2.1. First properties

In this section we will treat simple properties of the various systems $\lambda_{\to}$. Deeper properties—like strong normalization of typeable terms—will be considered in Section 2.2.

Properties of $\lambda_{\to}^{\mathrm{Cu}}$, $\lambda_{\to}^{\mathrm{Ch}}$ and $\lambda_{\to}^{\mathrm{dB}}$

Unless stated otherwise, properties stated for $\lambda_{\to}$ apply to both systems.

2.1.1. Proposition (Weakening lemma for $\lambda_{\to}$).
Suppose $\Gamma \vdash M : A$ *and* Γ' *is a basis with* $\Gamma \subseteq \Gamma'$. *Then* $\Gamma' \vdash M : A$.

Proof. By induction on the derivation of $\Gamma \vdash M : A$. ■

2.1.2. Lemma (Free variable lemma). (i) *Suppose* $\Gamma \vdash M : A$. *Then* $FV(M) \subseteq \mathrm{dom}(\Gamma)$.

(ii) *If* $\Gamma \vdash M : A$, *then* $\Gamma \restriction \mathrm{FV}(M) \vdash A : M$, *where for a set* X *of variables one has* $\Gamma \restriction \mathrm{FV}(M) = \{x{:}A \in \Gamma \mid x \in X\}$.

Proof. (i), (ii) By induction on the generation of $\Gamma \vdash M : A$. ■

The following result is related to the fact that the system $\lambda_{\to}$ is 'syntax directed', i.e. statements $\Gamma \vdash M : A$ have a unique proof.

2.1.3. Proposition (Generation lemma for $\lambda_{\to}^{\mathrm{Cu}}$).

$$\begin{array}{lrcl}
\text{(i)} & \Gamma \vdash x : A & \Rightarrow & (x{:}A) \in \Gamma. \\
\text{(ii)} & \Gamma \vdash MN : A & \Rightarrow & \exists B \in \mathbb{T}\, [\Gamma \vdash M : B \to A \;\&\; \Gamma \vdash N : B]. \\
\text{(iii)} & \Gamma \vdash \lambda x.M : A & \Rightarrow & \exists B, C \in \mathbb{T}\, [A \equiv B \to C \;\&\; \Gamma, x{:}B \vdash M : C].
\end{array}$$

Proof. (i) Suppose $\Gamma \vdash x : A$ holds in $\lambda_{\to}$. The last rule in a derivation of this statement cannot be an application or an abstraction, since x is not of the right form. Therefore it must be an axiom, i.e. $(x{:}A) \in \Gamma$.

(ii), (iii) The other two implications are proved similarly. ■

2.1.4. PROPOSITION (Generation lemma for $\lambda^{\mathrm{dB}}_{\to}$).

$$\begin{array}{lrcl}
\text{(i)} & \Gamma \vdash x : A & \Rightarrow & (x{:}A) \in \Gamma. \\
\text{(ii)} & \Gamma \vdash MN : A & \Rightarrow & \exists B \in \mathbb{T}\, [\Gamma \vdash M : B{\to}A \;\&\; \Gamma \vdash N : B]. \\
\text{(iii)} & \Gamma \vdash \lambda x{:}B.M : A & \Rightarrow & \exists C \in \mathbb{T}\, [A \equiv B{\to}C \;\&\; \Gamma, x{:}B \vdash M : C].
\end{array}$$

PROOF. Similarly. ■

2.1.5. PROPOSITION (Generation lemma for $\lambda^{\mathrm{Ch}}_{\to}$).

$$\begin{array}{lrcl}
\text{(i)} & x^B \in \Lambda^{\mathrm{Ch}}_{\to}(A) & \Rightarrow & B = A. \\
\text{(ii)} & (MN) \in \Lambda^{\mathrm{Ch}}_{\to}(A) & \Rightarrow & \exists B \in \mathbb{T}.[M \in \Lambda^{\mathrm{Ch}}_{\to}(B{\to}A) \;\&\; N \in \Lambda^{\mathrm{Ch}}_{\to}(B)]. \\
\text{(iii)} & (\lambda x^B.M) \in \Lambda^{\mathrm{Ch}}_{\to}(A) & \Rightarrow & \exists C \in \mathbb{T}.[A = (B{\to}C) \;\&\; M \in \Lambda^{\mathrm{Ch}}_{\to}(C)].
\end{array}$$

PROOF. As before. ■

The following two results hold for $\lambda^{\mathrm{Cu}}_{\to}$ and $\lambda^{\mathrm{dB}}_{\to}$. Variants already have been proved for $\lambda^{\mathrm{Ch}}_{\to}$, Propositions 1.4.2 and 1.4.4(iii).

2.1.6. PROPOSITION (Substitution lemma for $\lambda^{\mathrm{Cu}}_{\to}$ and $\lambda^{\mathrm{dB}}_{\to}$).
(i) $\Gamma, x{:}A \vdash M : B \;\&\; \Gamma \vdash N : A \;\Rightarrow\; \Gamma \vdash M[x{:=}N] : B$.
(ii) $\Gamma \vdash M : A \;\Rightarrow\; \Gamma[\alpha := B] \vdash M : A[\alpha := B]$.

PROOF. The proof will be given for $\lambda^{\mathrm{Cu}}_{\to}$, for $\lambda^{\mathrm{dB}}_{\to}$ it is similar.
(i) By induction on the derivation of $\Gamma, x{:}A \vdash M : B$. Write $P^* \equiv P[x{:=}N]$.

Case 1. $\Gamma, x{:}A \vdash M : B$ is an axiom, hence $M \equiv y$ and $(y{:}B) \in \Gamma \cup \{x{:}A\}$.
Subcase 1.1. $(y{:}B) \in \Gamma$. Then $y \not\equiv x$ and $\Gamma \vdash M^* \equiv y[x{:}N] \equiv y : B$.
Subcase 1.2. $y{:}B \equiv x{:}A$. Then $y \equiv x$ and $B \equiv A$, hence $\Gamma \vdash M^* \equiv N : A \equiv B$.

Case 2. $\Gamma, x{:}A \vdash M : B$ follows from $\Gamma, x{:}A \vdash F : C{\to}B$, $\Gamma, x{:}A \vdash G : C$ and $FG \equiv M$. By the induction hypothesis one has $\Gamma \vdash F^* : C{\to}B$ and $\Gamma \vdash G^* : C$. Hence $\Gamma \vdash (FG)^* \equiv F^*G^* : B$.

Case 3. $\Gamma, x{:}A \vdash M : B$ follows from $\Gamma, x{:}A, y{:}D \vdash G : E$, $B \equiv D{\to}E$ and $\lambda y.G \equiv M$. By the induction hypothesis $\Gamma, y{:}D \vdash G^* : E$, hence $\Gamma \vdash (\lambda y.G)^* \equiv \lambda y.G^* : D{\to}E \equiv B$.
(ii) Similarly. ■

2.1.7. PROPOSITION (Subject reduction property for $\lambda^{\mathrm{Cu}}_{\to}$ and $\lambda^{\mathrm{dB}}_{\to}$).
Suppose $M \twoheadrightarrow_{\beta\boldsymbol{\eta}} M'$. *Then* $\Gamma \vdash M : A \;\Rightarrow\; \Gamma \vdash M' : A$.

PROOF. The proof will be given for $\lambda^{\mathrm{dB}}_{\to}$, for $\lambda^{\mathrm{Cu}}_{\to}$ it is similar. Suppose $\Gamma \vdash M : A$ and $M \to M'$ in order to show that $\Gamma \vdash M' : A$; then the result follows by induction on the derivation of $\Gamma \vdash M : A$.

Case 1. $\Gamma \vdash M : A$ is an axiom. Then M is a variable, contradicting $M \to M'$. Hence this case cannot occur.

Case 2. $\Gamma \vdash M : A$ is $\Gamma \vdash FN : A$ and is a direct consequence of $\Gamma \vdash F : B{\to}A$ and $\Gamma \vdash N : B$. Since $FN \equiv M \to M'$ we can have three subcases.

Subcase 2.1. $M' \equiv F'N$ with $F \to F'$.

Subcase 2.2. $M' \equiv FN'$ with $N \to N'$.
In these two subcases it follows by the induction hypothesis that $\Gamma \vdash M' : A$.

Subcase 2.3. $F \equiv \lambda x{:}B.G$ and $M' \equiv G[x{:=}N]$. Since

$$\Gamma \vdash \lambda x.G : B{\to}A \;\&\; \Gamma \vdash N : B$$

it follows by the generation lemma 2.1.3 for $\lambda_\to$ that

$$\Gamma, x{:}B \vdash G : A \;\&\; \Gamma \vdash N : B.$$

Therefore by the substitution lemma 2.1.6 for $\lambda_\to$ it follows that
$\Gamma \vdash G[x{:=}N] : A$, i.e. $\Gamma \vdash M' : A$.

Case 3. $\Gamma \vdash M : A$ is $\Gamma \vdash \lambda x{:}B.N : B{\to}C$ and follows from $\Gamma, x{:}B \vdash N : C$. Since $M \to M'$ we have $M' \equiv \lambda x{:}B.N'$ with $N \to N'$. By the induction hypothesis one has $\Gamma, x{:}B \vdash N' : C$, hence $\Gamma \vdash \lambda x{:}B.N' : B{\to}C$, i.e. $\Gamma \vdash M' : A$. ■

The following result also holds for $\lambda^{\mathrm{Ch}}_\to$ and $\lambda^{\mathrm{dB}}_\to$, Exercise 2.5.4.

2.1.8. Corollary (Church-Rosser Theorem for $\lambda^{\mathrm{Cu}}_\to$). *On typable terms of $\lambda^{\mathrm{Cu}}_\to$ the Church-Rosser theorem holds for the notions of reduction $\twoheadrightarrow_\beta$ and $\twoheadrightarrow_{\beta\boldsymbol{\eta}}$.*

(i) *Let $M, N \in \Lambda^\Gamma_\to(A)$. Then*

$$M =_{\beta(\boldsymbol{\eta})} N \;\Rightarrow\; \exists Z \in \Lambda^\Gamma_\to(A).M \twoheadrightarrow_{\beta(\boldsymbol{\eta})} Z \;\&\; N \twoheadrightarrow_{\beta(\boldsymbol{\eta})} Z.$$

(ii) *Let $M, N_1, N_2 \in \Lambda^\Gamma_\to(A)$. Then*

$$M \twoheadrightarrow_{\boldsymbol{\beta\eta}} N_1 \;\&\; M \twoheadrightarrow_{\beta(\boldsymbol{\eta})} N_2 \;\Rightarrow\; \exists Z \in \Lambda^\Gamma_\to(A).N_1 \twoheadrightarrow_{\beta(\boldsymbol{\eta})} Z \;\&\; N_2 \twoheadrightarrow_{\beta(\boldsymbol{\eta})} Z.$$

Proof. By the Church-Rosser theorems for $\twoheadrightarrow_\beta$ and $\twoheadrightarrow_{\boldsymbol{\beta\eta}}$ on untyped terms, Theorem 1.1.7, and Proposition 2.1.7. ■

The following property of uniqueness of types only holds for the Church and de Bruijn versions of $\lambda_\to$. It is instructive to find out where the proof brakes down for $\lambda^{\mathrm{Cu}}_\to$ and also that the two contexts in (ii) should be the same.

2.1.9. Proposition (Unicity of types for $\lambda^{\mathrm{Ch}}_\to$ and $\lambda^{\mathrm{dB}}_\to$).

(i) $M \in \Lambda^{\mathrm{Ch}}_\to(A) \;\&\; M \in \Lambda^{\mathrm{Ch}}_\to(B) \;\Rightarrow\; A = B.$
(ii) $\Gamma \vdash^{\mathrm{dB}}_{\lambda_\to} M : A \;\&\; \Gamma \vdash^{\mathrm{dB}}_{\lambda_\to} M : B \;\Rightarrow\; A = B.$

Proof. (i), (ii) By induction on the structure of M, using the generation lemma 2.1.4. ■

Normalization

For several applications, for example for the problem to find all possible inhabitants of a given type, we will need the weak normalization theorem, stating that all typable terms do have a $\beta\boldsymbol{\eta}$-nf (normal form). The result is valid for all versions of $\lambda_{\rightarrow}$ and *a fortiori* for the subsystems $\lambda^o_{\rightarrow}$. The proof is due to Turing and is published posthumously in Gandy [1980]. In fact all typable terms in these systems are $\beta\boldsymbol{\eta}$ strongly normalizing, which means that all $\beta\boldsymbol{\eta}$-reductions are terminating. This fact requires more work and will be proved in §12.2.

The notion of 'abstract reduction system', see Klop [1992], is useful for the understanding of the proof of the normalization theorem.

2.1.10. Definition. (i) An *abstract reduction system* is a pair $(X, \rightarrow_R)$, where X is a set and $\rightarrow_R$ is a binary relation on X.

(ii) An element $x \in X$ is said to be in R-normal form (R-nf) if for no $y \in X$ one has $x \rightarrow_R y$.

(iii) (X, R) is called *weakly normalizing* (R-WN, or simply WN) if every element has an R-nf.

(iv) (X, R) is said to be *strongly normalizing* (R-SN, or simply SN) if every R-reduction path

$$x_0 \rightarrow_R x_1 \rightarrow_R x_2 \rightarrow_R \dots$$

is finite.

2.1.11. Definition. (i) A *multiset over nat* can be thought of as a generalized set S in which each element may occur more than once. For example

$$S = \{3, 3, 1, 0\}$$

is a multiset. We say that 3 occurs in S with multiplicity 2; that 1 has multiplicity 1; etcetera.

More formally, the above multiset S can be identified with a function $f \in \mathbb{N}^{\mathbb{N}}$ that is almost everywhere 0, except

$$f(0) = 1, f(1) = 1, f(3) = 2.$$

This S is finite if f has *finite support*, where

$$\mathtt{support}(f) = \{x \in \mathbb{N} \mid f(x) \neq 0\}.$$

(ii) Let $\mathcal{S}(\mathbb{N})$ be the collection of all finite multisets over $\mathbb{N}$. $\mathcal{S}(\mathbb{N})$ can be identified with $\{f \in \mathbb{N}^{\mathbb{N}} \mid \mathtt{support}(f) \text{ is finite}\}$.

2.1.12. Definition. Let $S_1, S_2 \in \mathcal{S}(\mathbb{N})$. Write

$$S_1 \rightarrow_{\mathcal{S}} S_2$$

if S_2 results from S_1 by replacing some elements (just one occurrence) by finitely many lower elements (in the usual ordering of $\mathbb{N}$). For example

$$\{3, \underline{3}, 1, 0\} \rightarrow_{\mathcal{S}} \{3, \underline{2, 2, 2, 1}, 1, 0\}.$$

2.1.13. LEMMA. *We define a particular (non-deterministic) reduction strategy F on $\mathcal{S}(\mathbb{N})$. A multi-set S is contracted to $F(S)$ by taking a maximal element $n \in S$ and replacing it by finitely many numbers $< n$. Then F is a normalizing reduction strategy, i.e. for every $S \in \mathcal{S}(\mathbb{N})$ the $\mathcal{S}$-reduction sequence*

$$S \to_{\mathcal{S}} F(S) \to_{\mathcal{S}} F^2(S) \to_{\mathcal{S}} \dots$$

is terminating.

PROOF. By induction on the highest number n occuring in S. If $n = 0$, then we are done. If $n = k+1$, then we can successively replace in S all occurrences of n by numbers $\leq k$ obtaining S_1 with maximal number $\leq k$. Then we are done by the induction hypothesis. ■

In fact $(\mathcal{S}(\mathbb{N}), \twoheadrightarrow_{\mathcal{S}})$ is SN. Although we do not strictly need this fact, we will give even two proofs of it. In the first place it is something one ought to know; in the second place it is instructive to see that the result does not imply that $\lambda_{\to}$ satisfies SN.

2.1.14. LEMMA. *The reduction system $(\mathcal{S}(\mathbb{N}), \to_{\mathcal{S}})$ is SN.*

We will give two proofs of this lemma. The first one uses ordinals; the second one is from first principles.

PROOF$_1$. Assign to every $S \in \mathcal{S}(\mathbb{N})$ an ordinal $\#S < \omega^\omega$ as suggested by the following examples.

$$\begin{array}{rcl} \#\{3,3,1,0,0,0\} & = & 2\omega^3 + \omega + 3; \\ \#\{3,2,2,2,1,1,0\} & = & \omega^3 + 3\omega^2 + 2\omega + 1. \end{array}$$

More formally, if S is represented by $f \in \mathbb{N}^{\mathbb{N}}$ with finite support, then

$$\#S = \Sigma_{i \in \mathbb{N}} f(i) \cdot \omega^i.$$

Notice that

$$S_1 \to_{\mathcal{S}} S_2 \;\Rightarrow\; \#S_1 > \#S_2$$

(in the example because $\omega^3 > 3\omega^2 + \omega$). Hence by the well-foundedness of the ordinals the result follows. ■$_1$

PROOF$_2$. Define

$$\begin{array}{rcl} \mathcal{F}_k & = & \{f \in \mathbb{N}^{\mathbb{N}} \mid \forall n \geq k\, f(n) = 0\}; \\ \mathcal{F} & = & \cup_{k \in \mathbb{N}} \mathcal{F}_k. \end{array}$$

The set $\mathcal{F}$ is the set of functions with finite support. Define on $\mathcal{F}$ the relation $>$ corresponding to the relation $\to_{\mathcal{S}}$ for the formal definition of $\mathcal{S}(\mathbb{N})$.

$$\begin{array}{rcl} f > g & \iff & f(k) > g(k), \text{ where } k \in \mathbb{N} \text{ is largest} \\ & & \text{such that } f(k) \neq g(k). \end{array}$$

It is easy to see that $(\mathcal{F}, >)$ is a linear ordering. We will show that it is even a well-ordering, i.e. for every non-empty set $X \subseteq \mathcal{F}$ there is a least element $f_0 \in X$. This implies that there are no infinite descending chains in $\mathcal{F}$.

To show this claim it suffices to prove that each $\mathcal{F}_k$ is well-ordered, since

$$\ldots > (\mathcal{F}_{k+1} \setminus \mathcal{F}_k) > \mathcal{F}_k$$

element-wise. This will be proved by induction on k. If $k = 0$, then this is trivial, since $\mathcal{F}_0 = \{\lambda\!\!\lambda n.0\}$. Now assume (induction hypothesis) that $\mathcal{F}_k$ is well-ordered in order to show the same for $\mathcal{F}_{k+1}$. Let $X \subseteq \mathcal{F}_{k+1}$ be non-empty. Define

$$\begin{array}{rcl} X(k) & = & \{f(k) \mid f \in X\} \subseteq \mathbb{N}; \\ X_k & = & \{f \in X \mid f(k) \text{ minimal in } X(k)\} \subseteq \mathcal{F}_{k+1}; \\ X_k|k & = & \{g \in \mathcal{F}_k \mid \exists f \in X_k\, f|k = g\} \subseteq \mathcal{F}_k, \end{array}$$

where

$$\begin{array}{rcll} f|k(i) & = & f(i), & \text{if } i < k; \\ & = & 0, & \text{else.} \end{array}$$

By the induction hypothesis $X_k|k$ has a least element g_0. Then $g_0 = f_0|k$ for some $f_0 \in X_k$. This f_0 is then the least element of X_k and hence of X. $\blacksquare_2$

2.1.15. Remark. The second proof shows in fact that if $(D, >)$ is a well-ordered set, then so is $(\mathcal{S}(D), >)$, defined analogously to $(\mathcal{S}(\mathbb{N}), >)$. In fact the argument can be carried out in Peano Arithmetic, showing

$$\vdash_{\mathbf{PA}} \mathtt{TI}(\alpha) \to \mathtt{TI}(\alpha^\omega),$$

where $\mathtt{TI}(\alpha)$ is the principle of transfinite induction for the ordinal α. Since $\mathtt{TI}(\omega)$ is in fact ordinary induction we have in PA

$$\mathtt{TI}(\omega),\ \mathtt{TI}(\omega^\omega),\ \mathtt{TI}(\omega^{(\omega^\omega)}), \ldots$$

This implies that the proof of $\mathtt{TI}(\alpha)$ can be carried out in Peano Arithmetic for every $\alpha < \epsilon_0$. Gentzen [1936] shows that $\mathtt{TI}(\epsilon_0)$, where $\epsilon_0 = \omega^{\omega^{\omega^{\cdots}}}$, cannot be carried out in PA.

In order to prove the $\lambda_\to$ is WN it suffices to work with $\lambda^{\mathrm{Ch}}_\to$. We will use the following notation. We write terms with extra type information, decorating each subterm with its type. For example, instead of $(\lambda x^A.M)N \in \mathtt{term}_B$ we write $(\lambda x^A.M^B)^{A\to B}N^A$.

2.1.16. Definition. (i) Let $R \equiv (\lambda x^A.M^B)^{A\to B}N^A$ be a redex. The *depth* of R, notation $\#R$, is defined as follows.

$$\#R = \#(A{\to}B)$$

where $\#$ on types is defined inductively by

$$\begin{array}{rcl} \#\alpha & = & 0; \\ \#(A{\to}B) & = & \max(\#A, \#B) + 1. \end{array}$$

(ii) To each M in $\lambda^{\mathrm{Ch}}_{\to}$ we assign a multi-set S_M as follows

$$S_M = \{\#R \mid R \text{ is a redex occurrence in } M\},$$

with the understanding that the multiplicity of R in M is copied in S_M.

In the following example we study how the contraction of one redex can duplicate other redexes or create new redexes.

2.1.17. Example. (i) Let R be a redex occurrence in a typed term M. Assume

$$M \xrightarrow{R}_\beta N,$$

i.e. N results form M by contracting R. This contraction can duplicate other redexes. For example (we write $M[P]$, or $M[P,Q]$ to display subterms of M)

$$(\lambda x.M[x,x])R_1 \to_\beta M[R_1,R_1]$$

duplicates the other redex R_1.

(ii) (J.J. Lévy [1978]) Contraction of a β-redex may also create new redexes. For example

$$\begin{array}{rcl}
(\lambda x^{A\to B}.M[x^{A\to B}P^A]^C)^{(A\to B)\to C}(\lambda y^A.Q^B) & \to_\beta & M[(\lambda y^A.Q^B)^{A\to B}P^A]^C;\\
(\lambda x^A.(\lambda y^B.M[x^A,y^B]^C)^{B\to C})^{A\to(B\to C)}P^AQ^B & \to_\beta & (\lambda y^B.M[P^A,y^B]^C)^{B\to C}Q^B;\\
(\lambda x^{A\to B}.x^{A\to B})^{(A\to B)\to(A\to B)}(\lambda y^A.P^B)^{A\to B}Q^A & \to_\beta & (\lambda y^A.P^B)^{A\to B}Q^A.
\end{array}$$

2.1.18. Lemma. *Assume $M \xrightarrow{R}_\beta N$ and let R_1 be a created redex in N. Then $\#R > \#R_1$.*

Proof. In Lévy [1978] it is proved that the three ways of creating redexes in example 2.1.17(ii) are the only possibilities. For a proof do exercise 14.5.3 in B[1984]. In each of three cases we can inspect that the statement holds. ■

2.1.19. Theorem (Weak normalization theorem for $\lambda_\to$). *If $M \in \Lambda$ is typable in $\lambda_\to$, then M is $\beta\boldsymbol{\eta}$-WN, i.e. has a $\beta\boldsymbol{\eta}$-nf.*

Proof. By Proposition 1.4.9(ii) it suffices to show this for terms in $\lambda^{\mathrm{Ch}}_{\to}$. Note $\boldsymbol{\eta}$-reductions decreases the length of a term; moreover, for β-normal terms $\boldsymbol{\eta}$-contractions do not create β-redexes. Therefore in order to establish $\beta\boldsymbol{\eta}$-WN it is sufficient to prove that M has a β-nf.

Define the following β-reduction strategy F. If M is in nf, then $F(M) = M$. Otherwise, let R be the *rightmost redex of maximal depth* n in M. Then

$$F(M) = N$$

where $M \xrightarrow{R}_\beta N$. Contracting a redex can only duplicate other redexes that are to the right of that redex. Therefore by the choice of R there can only be redexes of M duplicated in $F(M)$ of depth $< n$. By lemma 2.1.18 redexes created in $F(M)$ by the contraction $M \to_\beta F(M)$ are also of depth $< n$. Therefore in case M is not in β-nf we have

$$S_M \to_{\mathcal{S}} S_{F(M)}.$$

Since $\to_{\mathcal{S}}$ is SN, it follows that the reduction

$$M \to_\beta F(M) \to_\beta F^2(M) \to_\beta F^3(M) \to_\beta \dots$$

must terminate in a β-nf. ■

For β-reduction this weak normalization theorem was first proved by Turing, see Gandy [1980b]. The proof does not really need SN for $\mathcal{S}$-reduction. One may also use the simpler result lemma 2.1.13.

It is easy to see that a different reduction strategy does not yield a $\mathcal{S}$-reduction chain. For example the two terms

$$\begin{array}{l}(\lambda x^A.y^{A\to A\to A}x^A x^A)^{A\to A}((\lambda x^A.x^A)^{A\to A}x^A) \quad \to_\beta \\ y^{A\to A\to A}((\lambda x^A.x^A)^{A\to A}x^A)((\lambda x^A.x^A)^{A\to A}x^A)\end{array}$$

give the multisets $\{1,1\}$ and $\{1,1\}$. Nevertheless, SN does hold for all systems $\lambda_\to$, as will be proved in Section 2.2. It is an open problem whether ordinals can be assigned in a natural and simple way to terms of $\lambda_\to$ such that

$$M \to_\beta N \;\Rightarrow\; \mathtt{ord}(M) > \mathtt{ord}(N).$$

See Howard [1970] and de Vrijer [1987].

Applications of normalization

We will prove that normal terms inhabiting the represented data types (Bool, Nat, Σ^* and T_B) are standard, i.e. correspond to the intended elements. From WN for $\lambda_\to$ and the subject reduction theorem it then follows that all inhabitants of the mentioned data types are standard.

2.1.20. PROPOSITION. *Let $M \in \Lambda$ be in nf. Then $M \equiv \lambda x_1 \cdots x_n.yM_1 \dots M_m$, with $n, m \geq 0$ and the $M_1, \dots, M_m$ again in nf.*

PROOF. By induction on the structure of M. See Barendregt [1984], proposition 8.3.8 for some details if necessary. ■

2.1.21. PROPOSITION. *Let $\mathsf{Bool} \equiv \mathsf{Bool}_\alpha$, with α a type variable. Then for M in nf one has*

$$\vdash M : \mathsf{Bool} \;\Rightarrow\; M \in \{\mathsf{true}, \mathsf{false}\}.$$

PROOF. By repeated use of proposition 2.1.20, the free variable lemma 2.1.2 and the generation lemma for $\lambda_\to^{\mathrm{Cu}}$, proposition 2.1.3, one has the following chain of arguments.

$$\begin{array}{rcl}\vdash M : \alpha\to\alpha\to\alpha & \Rightarrow & M \equiv \lambda x.M_1 \\ & \Rightarrow & x{:}\alpha \vdash M_1 : \alpha\to\alpha \\ & \Rightarrow & M_1 \equiv \lambda y.M_2 \\ & \Rightarrow & x{:}\alpha, y{:}\alpha \vdash M_2 : \alpha \\ & \Rightarrow & M_2 \equiv x \text{ or } M_2 \equiv y.\end{array}$$

So $M \equiv \lambda xy.x \equiv \mathsf{true}$ or $M \equiv \lambda xy.y \equiv \mathsf{false}$. ■

2.1.22. Proposition. *Let* $\mathsf{Nat} \equiv \mathsf{Nat}_\alpha$. *Then for* M *in nf one has*

$$\vdash M : \mathsf{Nat} \;\Rightarrow\; M \in \{\ulcorner n \urcorner \,|\, n \in \mathbb{N}\}.$$

Proof. Again we have

$$\begin{array}{lcl} \vdash M : \alpha{\to}(\alpha{\to}\alpha){\to}\alpha & \Rightarrow & M \equiv \lambda x.M_1 \\ & \Rightarrow & x{:}\alpha \vdash M_1 : (\alpha{\to}\alpha){\to}\alpha \\ & \Rightarrow & M_1 \equiv \lambda f.M_2 \\ & \Rightarrow & x{:}\alpha, f{:}\alpha{\to}\alpha \vdash M_2 : \alpha. \end{array}$$

Now we have

$$\begin{array}{lcl} x{:}\alpha, f{:}\alpha{\to}\alpha \vdash M_2 : \alpha & \Rightarrow & [M_2 \equiv x \vee \\ & & [M_2 \equiv fM_3 \;\&\; x{:}\alpha, f{:}\alpha{\to}\alpha \vdash M_3 : \alpha]]. \end{array}$$

Therefore by induction on the structure of M_2 it follows that

$$x{:}\alpha, f{:}\alpha{\to}\alpha \vdash M_2 : \alpha \;\Rightarrow\; M_2 \equiv f^n(x),$$

with $n \geq 0$. So $M \equiv \lambda x f.f^n(x) \equiv \ulcorner n \urcorner$. ■

2.1.23. Proposition. *Let* $\mathsf{Sigma}^* \equiv \mathsf{Sigma}^*_\alpha$. *Then for* M *in nf one has*

$$\vdash M : \mathsf{Sigma}^* \;\Rightarrow\; M \in \{\underline{w} \,|\, w \in \Sigma^*\}.$$

Proof. Again we have

$$\begin{array}{lcl} \vdash M : \alpha{\to}(\alpha{\to}\alpha)^k{\to}\alpha & \Rightarrow & M \equiv \lambda x.N \\ & \Rightarrow & x{:}\alpha \vdash N : (\alpha{\to}\alpha)^k{\to}\alpha \\ & \Rightarrow & N \equiv \lambda a_1.N_1 \;\&\; x{:}\alpha, a_1{:}\alpha{\to}\alpha \vdash N_1 : (\alpha{\to}\alpha)^{k-1}{\to}\alpha \\ & \cdots & \\ & \Rightarrow & N \equiv \lambda a_1 \cdots a_k.N \;\&\; x{:}\alpha, a_1, \ldots, a_k{:}\alpha{\to}\alpha \vdash N_k : \alpha \\ & \Rightarrow & [N_k \equiv x \vee \\ & & [N_k \equiv a_{i_j} N'_k \;\&\; x{:}\alpha, a_1, \ldots, a_k{:}\alpha{\to}\alpha \vdash N_k{}' : \alpha]] \\ & \Rightarrow & N_k \equiv a_{i_1}(a_{i_2}(\cdots(a_{i_p}x)\cdots)) \\ & \Rightarrow & M \equiv \lambda x a_1 \cdots a_k.a_{i_1}(a_{i_2}(\cdots(a_{i_p}x)\cdots)) \\ & & \equiv \underline{a_{i_1}a_{i_2}\cdots a_{i_p}}. \blacksquare \end{array}$$

Before we can prove that inhabitants of $\mathsf{tree}[\beta]$ are standard, we have to intoduce an auxiliary notion.

2.1.24. Definition. Given $t \in T[b_1, \ldots, b_n]$ define $[t]^{p,l} \in \Lambda$ as follows.

$$\begin{array}{lcl} [b_i]^{p,l} & = & lb_i; \\ [P(t_1, t_2)]^{p,l} & = & p[t_1]^{p,l}[t_2]^{p,l}. \end{array}$$

2.1.25. LEMMA. *For $t \in T[b_1, \ldots, b_n]$ we have*

$$[t] =_\beta \lambda pl.[t]^{p,l}.$$

PROOF. By induction on the structure of t.

$$\begin{array}{rcll} [b_i] & \equiv & \lambda pl.lb_i & \\ & \equiv & \lambda pl.[b_i]^{p,l}; & \\ [P(t_1,t_2)] & \equiv & \lambda pl.p([t_1]pl)([t_2]pl) & \\ & = & \lambda pl.p[t_1]^{p,l}[t_2]^{p.l}, & \text{by the IH,} \\ & \equiv & \lambda pl.[P(t_1,t_2)]^{p,l}. \blacksquare & \end{array}$$

2.1.26. PROPOSITION. *Let $\mathsf{tree}[\beta] \equiv \mathsf{tree}_\alpha[\beta]$. Then for M in nf one has*

$$b_1, \ldots, b_n{:}\beta \vdash M : \mathsf{tree}[\beta] \;\Rightarrow\; M \in \{[t] \mid t \in T[b_1, \ldots, b_n]\}.$$

PROOF. We have $\vec{b}{:}\beta \vdash M : (\alpha{\to}\alpha{\to}\alpha){\to}(\beta{\to}\alpha){\to}\alpha \;\Rightarrow$

$$\begin{array}{ll} \Rightarrow & M \equiv \lambda p.M' \\ \Rightarrow & \vec{b}{:}\beta, p{:}\alpha{\to}\alpha{\to}\alpha \vdash M' : (\beta{\to}\alpha){\to}\alpha \\ \Rightarrow & M' \equiv \lambda l.M'' \\ \Rightarrow & \vec{b}{:}\beta, p{:}(\alpha{\to}\alpha{\to}\alpha), l{:}(\beta{\to}\alpha) \vdash M'' : \alpha \\ \Rightarrow & M'' \equiv lb_i \;\vee\; [M'' \equiv pM_1M_2 \;\&\; \\ & \vec{b}{:}\beta, p{:}(\alpha{\to}\alpha{\to}\alpha), l{:}(\beta{\to}\alpha) \vdash M_j : \alpha], \qquad \text{j=1,2,} \\ \Rightarrow & M'' \equiv [t]^{p,l}, \text{ for some } t \in T[\vec{b}], \\ \Rightarrow & M \equiv \lambda pl.[t]^{p,l} =_\beta [t], \qquad \text{by lemma 2.1.25.} \blacksquare \end{array}$$

2.2. Proofs of strong normalization

We now will give two proofs showing that $\lambda_{\to}$ is strongly normalizing. The first one is the classical proof due to Tait [1967] that needs little technique, but uses set theoretic comprehension. The second proof due to Statman is elementary, but needs results about reduction.

2.2.1. THEOREM (SN for $\lambda^{\mathrm{Ch}}_{\to}$). *For all $A \in \mathbb{T}_\infty$, $M \in \Lambda^{\mathrm{Ch}}_{\to}(A)$ one has $\mathrm{SN}_{\beta\boldsymbol{\eta}}(M)$.*

PROOF. We use an induction loading. First we add to $\lambda_{\to}$ constants $d_\alpha \in \Lambda^{\mathrm{Ch}}_{\to}(\alpha)$ for each atom α, obtaining $\lambda^+_{\to\mathrm{Ch}}$. Then we prove SN for the extended system. It follows *a fortiori* that the system without the constants is SN.

One first defines for $A \in \mathbb{T}_\infty$ the following class $\mathcal{C}_A$ of *computable* terms of type A. We write SN for $\mathrm{SN}_{\beta\boldsymbol{\eta}}$.

$$\begin{array}{rcl} \mathcal{C}_\alpha & = & \{M \in \Lambda^\emptyset_{\to\mathrm{Ch}}(\alpha) \mid \mathrm{SN}(M)\}; \\ \mathcal{C}_{A\to B} & = & \{M \in \Lambda^\emptyset_{\to\mathrm{Ch}}(A{\to}B) \mid \forall P \in \mathcal{C}_A.MP \in \mathcal{C}_B\}. \end{array}$$

Then one defines the classes $\mathcal{C}^*_A$ of terms that are *computable under substitution*

$$\mathcal{C}^*_A = \{M \in \Lambda_{\to\mathrm{Ch}}(A) \mid \forall \vec{Q} \in \mathcal{C}.[M[\vec{x}{:=}\vec{Q}] \in \Lambda^\emptyset_{\to\mathrm{Ch}}(A) \;\Rightarrow\; M[\vec{x}{:=}\vec{Q}] \in \mathcal{C}_A]\}.$$

Write $\mathcal{C}^{(*)} = \bigcup\{\mathcal{C}^{(*)}_A \mid A \in \mathbb{T}(\lambda^+_\to)\}$. For $A = A_1 \to \ldots \to A_n \to \alpha$ define

$$d_A \equiv \lambda x_1{:}A_1 \ldots \lambda x_n{:}A_n.d_\alpha.$$

Then for A one has

$$M \in \mathcal{C}_A \iff \forall \vec{P} \in \mathcal{C}.M\vec{P} \in \mathrm{SN}, \tag{0}$$

$$M \in \mathcal{C}^*_A \iff \forall \vec{P}, \vec{Q} \in \mathcal{C}.M[\vec{x}{:=}\vec{Q}]\vec{P} \in \mathrm{SN}, \tag{1}$$

where the $\vec{P}, \vec{Q}$ should have the right types and $M\vec{P}$ and $M[\vec{x}{:=}\vec{Q}]\vec{P}$ are of type α, respectively. By an easy simultaneous induction on A one can show

$$M \in \mathcal{C}_A \Rightarrow \mathrm{SN}(M); \tag{2}$$

$$d_A \in \mathcal{C}_A. \tag{3}$$

In particular, since $M[\vec{x}{:=}\vec{P}]\vec{Q} \in \mathrm{SN} \Rightarrow M \in \mathrm{SN}$, it follows that

$$M \in \mathcal{C}^* \Rightarrow M \in \mathrm{SN}. \tag{4}$$

Now one shows by induction on M that

$$M \in \Lambda(A) \Rightarrow M \in \mathcal{C}^*_A. \tag{5}$$

We distinguish cases and use (1).

Case $M \equiv x$. Then for $P, \vec{Q} \in \mathcal{C}$ one has $M[x{:=}P]\vec{Q} \equiv P\vec{Q} \in \mathcal{C} \subseteq \mathrm{SN}$, by the definition of $\mathcal{C}$ and (2).

Case $M \equiv NL$ is easy.

Case $M \equiv \lambda x.N$. Now $\lambda x.N \in \mathcal{C}^*$ iff for all $\vec{P}, Q, \vec{R} \in \mathcal{C}$ one has

$$(\lambda x.N[\vec{y}{:=}\vec{P}])Q\vec{R} \in \mathrm{SN}. \tag{6}$$

If $\vec{P}, Q, \vec{R} \in \mathcal{C} \subseteq \mathrm{SN}$, then by the IH one has $N \in \mathcal{C}^* \subseteq \mathrm{SN}$ so

$$N[x{:=}Q, \vec{y}{:=}\vec{P}]\vec{R} \in \mathrm{SN}. \tag{7}$$

Now every maximal reduction path σ starting from the term in (6) passes through a reduct of the term in (7), as reductions within $N, \vec{P}, Q, \vec{R}$ are finite, hence σ is finite. Therefore we have (6).

Finally by (5) and (4), every typable term of $\lambda^+_\to$, hence of $\lambda_\to$, is SN. ■

The idea of the proof is that one would have liked to prove by induction on M that it is SN. But this is not directly possible. One needs the *induction loading* that $M\vec{P} \in \mathrm{SN}$. For a typed system with only combinators this is sufficient and this was the original argument of Tait [1967]. For lambda terms one needs the extra induction loading of being computable under substitution. This argument was first presented by Prawitz [1971] and Girard [1971].

2.2.2. Corollary (SN for $\lambda^{\mathrm{Cu}}_\to$). $\forall M \in \Lambda^{\mathbb{A}_\infty}_{\to\mathrm{Cu}}.\mathrm{SN}_{\beta\boldsymbol{\eta}}(M)$.

Proof. Suppose $M \in \Lambda$ has type A (with respect to Γ) and had an infinite reduction path σ. By repeated use of Proposition 1.4.9(ii) lift M to $M' \in \Lambda^{\mathrm{Ch}}_\to$ with an infinite reduction path (that projects to σ), contradicting the Theorem. ■

An elementary proof of strong normalization

Now the elementary proof of strong normalization of $\lambda^{\mathrm{Cu}}_{\rightarrow}$ due to Statman will be presented. Inspiration came from Nederpelt, Gandy and Klop. The point of this proof is that in this reduction system strong normalizability follows from normalizability by local structure arguements similar to and in many cases identical to those presented for the untyped lambda calculus in Barendregt [1984]. These include analysis of redex creation, permutabilty of head with internal reductions, and permutability of η- with β-redexes. In particular, no special proof technique is needed to obtain strong normalization once normalization has been observed. We assume that the reader is familiar with the untyped lambda calculus

2.2.3. Definition. (i) Let $R \equiv (\lambda x.X)Y$ be a β-redex. Then R is
(1) $\beta\mathsf{I}$ if $x \in \mathrm{FV}(X)$;
(2) $\beta\mathsf{K}$ if $x \notin \mathrm{FV}(X)$;
(3) $\beta\mathsf{K}^-$ if R is $\beta\mathsf{K}$ and $x{:}0$ and $X : 0$;
(4) $\beta\mathsf{K}^+$ if R is a $\beta\mathsf{K}$ and is not a $\beta\mathsf{K}$.
(ii) The term X is said to have the $\lambda\mathsf{K}^-$ *property* if every dummy abstraction $\lambda x.X$ has $x{:}0$ and $X : 0$.

Notation. (i) $\rightarrow_\beta$ is beta reduction.
(ii) $\rightarrow_{\boldsymbol{\eta}}$ is η reduction.
(iii) $\rightarrow_{\beta\mathsf{I}}$ is reduction of $\lambda\mathsf{I}$-redexes.
(iv) $\rightarrow_{\beta\mathsf{I}\mathsf{K}^+}$ is reduction of $\lambda\mathsf{I}$- or $\lambda\mathsf{K}^+$-redexes.
(v) $\rightarrow_{\beta\mathsf{K}^-}$ is reduction of $\lambda\mathsf{K}^-$-redexes.

2.2.4. Theorem. *Every $M \in \Lambda^{\mathrm{Cu}}_{\rightarrow}$ is strongly $\beta\boldsymbol{\eta}$-normalizable.*

Proof. The result is proved in several steps.
(i) *Every term is $\beta(\eta)$-normalizable.* For β this is Theorem 2.1.19.
(ii) *β-head reduction sequences terminate.* By the standardization theorem, B[1984] Theorem 11.4.7, there is a standard reduction to the β-normal form.
(iii) *No β-reduction cycles.* Consider a shortest term M beginning a cyclic reduction. By the standarization theorem there exists a standard reduction from M to M. By choice of M this standard reduction contains a head reduction. By permutability of head reductions with internal reductions for each natural number m, M begins a head reduction with at least m steps contradicting (ii).
(iv) $M \twoheadrightarrow_{\beta\boldsymbol{\eta}} N \;\Rightarrow\; \exists P.M \twoheadrightarrow_\beta P \twoheadrightarrow_\eta N$ (Church). This usually is referred to as η postponement, see B[1984] Corollary 15.1.5, for a proof.
(v) *Strong normalizability of $\rightarrow_\beta$ implies strong normalizability of $\rightarrow_{\beta\boldsymbol{\eta}}$.* Take an infinite $\rightarrow_{\beta\boldsymbol{\eta}}$ sequence and apply η postponement at each step. The result yields an infinite $\rightarrow_\beta$ sequence.
(vi) Let $\beta\mathsf{I}$ be the notion of reduction in which a β-redex $(\lambda x.M)N$ is only allowed to be contracted if $x \in \mathrm{FV}(M)$. Then $\beta\mathsf{I}$ *is weakly normalizing.* Same arguement as for (i).
(vii) $\beta\mathsf{I}$ *is strongly normalizing (Church).* See B[1984], Theorem 11.3.7.

(viii) $M \twoheadrightarrow_\beta N \Rightarrow \exists P.M \twoheadrightarrow_{\beta\mathsf{I}\mathsf{K}^+} P \twoheadrightarrow_{\beta\mathsf{K}^-} N$ ($\beta\mathsf{K}^-$-redex postponement). Contraction of $\beta\mathsf{K}^-$-redexes cannot create new redexes or copy old ones so $\beta\mathsf{K}^-$-redexes can be permuted with $\beta\mathsf{I}$ and and $\beta\mathsf{K}^+$-redexes. This permutation can cause the $\beta\mathsf{K}^-$-redex to be copied several times but it cannot turn an I-redex into a K-redex (indeed the reverse can happen) so no new $\beta\mathsf{K}^-$-redexes are created.

(ix) *If M has the $\lambda\mathsf{K}^-$ property then M β-reduces to only finitely many N.* This follows by (vii) and (viii).

(x) *If M has the $\lambda\mathsf{K}^-$ property then M is strongly β-normalizable.* By (i), (iii) and (ix).

(xi) *If M has the $\lambda\mathsf{K}^-$ property then M is strongly $\beta\eta$-normalizable.* By (v) and (x).

(xii) *For each M there is an N with the $\lambda\mathsf{K}^-$ property such that $N \twoheadrightarrow_{\beta\boldsymbol{\eta}} M$.* First expand M by η expansion so that every subterm of M beginning with a lambda is a lambda prefix followed by a matrix of type 0. Let $a : \alpha$ and $f : 0{\to}(0{\to}0)$ be new variables. For each type $T = T_1{\to}\ldots{\to}T_t{\to}\alpha$ with $T_i = T_{i,1}{\to}\ldots{\to}T_{i,k_i}{\to}\alpha_i$ for $i = 1,\ldots,t$ define terms $U_A : T$ recursively by

$$\begin{array}{rcl} U_0 & = & a; \\ U_T & = & \lambda x_1 \ldots x_t.f(x_1 U_{T_{1,1}} \ldots U_{T_{1,k_1}}) \ldots \\ & & (f(x_{t-1} U_{T_{t-1,1}} \ldots U_{T_{t-1,k_{t-1}}})(x_t U_{T_{t,1}} \ldots U_{T_{t,k_t}}))..). \end{array}$$

Now recursively replace each dummy λx occurring $\lambda x \lambda y \ldots \lambda z.X$ with $x : T$ and $X : 0$ by $\lambda x \lambda y \ldots \lambda z.KX(xU_{T_1} \ldots U_{T_t})$. Clearly the resulting N satisfies $N \twoheadrightarrow_{\beta\boldsymbol{\eta}} M$ and the $\lambda\mathsf{K}^-$ property, since all dummy lambdas appear in $K : 1_2$.

(xiii) *Every typable term is strongly $\beta\eta$ normalizable.* By (xi) and (xii). ■

Still another proof is to be found in de Vrijer [1987] in which for a typed term M a computation is given of the langest reduction path to β-nf.

2.3. Checking and finding types

There are several natural problems concerning type systems.

2.3.1. Definition. (i) The problem of *type checking* consists of determining, given basis Γ, term M and type A whether $\Gamma \vdash M : A$.

(ii) The problem of *typeability* consists of given a term M determining whether M has some type with respect to some Γ.

(iii) The problem of *type reconstruction* ('finding types') consists of finding all possible types A and bases Γ that type a given M.

(iv) The *inhabitation problem* consists of finding out whether a given type A is inhabited by some term M in a given basis Γ.

(v) The *enumeration problem* consists of determining for a given type A and a given context Γ all possible terms M such that $\Gamma \vdash M : A$.

The five problems may be summarized stylistically as follows.

$$\begin{array}{rlll} \Gamma & \vdash_{\lambda_\to} & M : A\,? & \textit{type checking};\\ \exists A, \Gamma\,[\Gamma & \vdash_{\lambda_\to} & M : A]\,? & \textit{typeability};\\ ? & \vdash_{\lambda_\to} & M : \ ? & \textit{type reconstruction};\\ \exists M\,[\Gamma & \vdash_{\lambda_\to} & M : A]\,? & \textit{inhabitation};\\ \Gamma & \vdash_{\lambda_\to} & \ ? : A & \textit{enumeration.} \end{array}$$

In another notation this is the following.

$$\begin{array}{rlll} & M & \in \Lambda^\Gamma_\to(A)\,? & \textit{type checking};\\ \exists A, \Gamma & M & \in \Lambda^\Gamma_\to(A)? & \textit{typeability};\\ & M & \in \Lambda^?_\to(?) & \textit{type reconstruction};\\ & \Lambda^\Gamma_\to(A) & \neq \emptyset\,? & \textit{inhabitation};\\ & ? & \in \Lambda^\Gamma_\to(A) & \textit{enumeration.} \end{array}$$

In this section we will treat the problems of type checking, typeability and type reconstruction for the three versions of $\lambda_\to$. It turns out that these problems are decidable for all versions. The solutions are essentially simpler for $\lambda^{\mathrm{Ch}}_\to$ and $\lambda^{\mathrm{dB}}_\to$ than for $\lambda^{\mathrm{Cu}}_\to$. The problems of inhabitation and enumeration will be treated in the next section.

One may wonder what is the role of the context Γ in these questions. The problem

$$\exists\Gamma\exists A\ \Gamma \vdash M : A.$$

can be reduced to one without a context. Indeed, for $\Gamma = \{x_1{:}A_1, \ldots, x_n{:}A_n\}$

$$\Gamma \vdash M : A \ \Leftrightarrow\ \vdash (\lambda x_1(:A_1)\ldots\lambda x_n(:A_n).M) : (A_1 \to \ldots \to A_n \to A).$$

Therefore

$$\exists\Gamma\exists A\,[\Gamma \vdash M : A] \iff \exists B\,[\vdash \lambda\vec{x}.M : B].$$

On the other hand the question

$$\exists\Gamma\exists M\,[\Gamma \vdash M : A]\,?$$

is trivial: take $\Gamma = \{x{:}A\}$ and $M \equiv x$. So we do not consider this question.

The solution of the problems like type checking for a fixed context will have important applications for the treatment of constants.

Checking and finding types for $\lambda^{\mathrm{dB}}_\to$ and $\lambda^{\mathrm{Ch}}_\to$

We will see again that the systems $\lambda^{\mathrm{Ch}}_\to$ and $\lambda^{\mathrm{dB}}_\to$ are essentially equivalent. For these systems the solutions to the problems of type checking, typeability and type reconstruction are easy. All of the solutions are computable with an algorithm of linear complexity.

2.3.2. Proposition (Type checking for $\lambda^{\mathrm{dB}}_\to$). *Let Γ be a basis of $\lambda^{\mathrm{dB}}_\to$. Then there is a computable function $\mathtt{type}_\Gamma : \Lambda_{\Pi} \to \Pi \cup \{\mathtt{error}\}$ such that*

$$M \in \Lambda^\Gamma_{\to\mathrm{Ch}}(A) \iff \mathtt{type}_\Gamma(M) = A.$$

PROOF. Define

$$\begin{array}{rcll} \mathtt{type}_\Gamma(x) & = & \Gamma(x); & \\ \mathtt{type}_\Gamma(MN) & = & B, & \text{if } \mathtt{type}_\Gamma(M) = \mathtt{type}_\Gamma(N){\to}B, \\ & = & \mathtt{error}, & \text{else;} \\ \mathtt{type}_\Gamma(\lambda x{:}A.M) & = & A{\to}\mathtt{type}_{\Gamma\cup\{x:A\}}(M), & \text{if } \mathtt{type}_{\Gamma\cup\{x:A\}}(M) \neq \mathtt{error}, \\ & = & \mathtt{error}, & \text{else.} \end{array}$$

Then the statement follows by induction on the structure of M. ■

2.3.3. COROLLARY. *Typeability and type reconstruction for $\lambda_{\to}^{\mathrm{dB}}$ are computable. In fact one has the following.*

(i) $M \in \Lambda_{\to\mathrm{dB}}^{\Gamma} \iff \mathtt{type}_\Gamma(M) \neq \mathtt{error}$.

(ii) *Each $M \in \Lambda_{\to\mathrm{dB}}^{\Gamma}(\mathtt{type}_\Gamma)$ has a unique type; in particular*

$$M \in \Lambda_{\to\mathrm{dB}}^{\Gamma}(\mathtt{type}_\Gamma(M)).$$

PROOF. By the proposition. ■

For $\lambda_{\to}^{\mathrm{Ch}}$ things are essentially the same, except that there are no bases needed, since variables come with their own types.

2.3.4. PROPOSITION (Type checking for $\lambda_{\to}^{\mathrm{Ch}}$). *There is a computable function* $\mathtt{type} : \Lambda_{\to\mathrm{Ch}} \to \mathbb{T} \cup \{\mathtt{error}\}$ *such that*

$$M \in \Lambda_{\to\mathrm{Ch}}(A) \iff \mathtt{type}(M) = A.$$

PROOF. Define

$$\begin{array}{rcll} \mathtt{type}(x^A) & = & A; & \\ \mathtt{type}(MN) & = & B, & \text{if } \mathtt{type}(M) = \mathtt{type}(N){\to}B, \\ & = & \mathtt{error}, & \text{else;} \\ \mathtt{type}(\lambda x^A.M) & = & A{\to}\mathtt{type}(M), & \text{if } \mathtt{type}(M) \neq \mathtt{error}, \\ & = & \mathtt{error}, & \text{else.} \end{array}$$

Then the statement follows again by induction on the structure of M. ■

2.3.5. COROLLARY. *Typeability and type reconstruction for $\lambda_{\to}^{\mathrm{Ch}}$ are computable. In fact one has the following.*

(i) $M \in \Lambda_{\to\mathrm{Ch}} \iff \mathtt{type}(M) \neq \mathtt{error}$.

(ii) *Each $M \in \Lambda_{\to\mathrm{Ch}}$ has a unique type; in particular $M \in \Lambda_{\to\mathrm{Ch}}(\mathtt{type}(M))$.*

PROOF. By the proposition. ■

Checking and finding types for $\lambda_{\to}^{\mathrm{Cu}}$

We now will show the computability of the three questions for $\lambda_{\to}^{\mathrm{Cu}}$. This occupies 2.3.6 - 2.3.16 and in these items $\vdash$ stands for $\vdash^{\mathbb{T}^\infty}_{\lambda_{\to}^{\mathrm{Cu}}}$.

Let us first make the easy observation that in $\lambda_{\to}^{\mathrm{Cu}}$ types are not unique. For example $\mathsf{I} \equiv \lambda x.x$ has as possible type $\alpha{\to}\alpha$, but also $(\beta{\to}\beta){\to}(\beta{\to}\beta)$ and $(\alpha{\to}\beta{\to}\beta){\to}(\alpha{\to}\beta{\to}\beta)$. Of these types $\alpha{\to}\alpha$ is the 'most general' in the sense that the other ones can be obtained by a substitution in α.

2.3.6. Definition. (i) A substitutor is an operation $* : \mathbb{T} \to \mathbb{T}$ such that

$$*(A \to B) \equiv *(A) \to *(B).$$

(ii) We write A^* for $*(A)$.

(iii) Usually a substitution $*$ has a finite support, that is, for all but finitely many type variables α one has $\alpha^* \equiv \alpha$ (the support of $*$ being

$$\sup(*) = \{\alpha \mid \alpha^* \not\equiv \alpha\}).$$

In that case we write

$$*(A) = A[\alpha_1 := \alpha_1^*, \ldots, \alpha_n := \alpha_n^*],$$

where $\{\alpha_1, \ldots, \alpha_n\} \supseteq \sup(*)$. We also write

$$* = [\alpha_1 := \alpha_1^*, \ldots, \alpha_n := \alpha_n^*]$$

and

$$* = [\,]$$

for the identity substitution.

2.3.7. Definition. (i) Let $A, B \in \mathbb{T}$. A *unifier* for A and B is a substitutor $*$ such that $A^* \equiv B^*$.

(ii) The substitutor $*$ is a *most general unifier for A and B if*

- $A^* \equiv B^*$
- $A^{*_1} \equiv B^{*_1} \;\Rightarrow\; \exists\, *_2 \,.\, *_1 \equiv *_2 \circ *$.

(iii) *Let $E = \{A_1 = B_1, \ldots, A_n = B_n\}$ be a finite set of equations between types. The equations do not need to be valid. A* unifier *for E is a substitutor $*$ such that $A_1^* \equiv B_1^* \;\&\; \cdots \;\&\; A_n^* \equiv B_n^*$. In that case one writes $* \models E$. Similarly one defines the notion of a most general unifier for E.*

2.3.8. Examples. The types $\beta \to (\alpha \to \beta)$ and $(\gamma \to \gamma) \to \delta$ have a unifier. For example $* = [\beta := \gamma \to \gamma,\ \delta := \alpha \to (\gamma \to \gamma)]$ or $*_1 = [\beta := \gamma \to \gamma,\ \alpha := \varepsilon \to \varepsilon,\ \delta := \varepsilon \to \varepsilon \to (\gamma \to \gamma)]$. The unifier $*$ is most general, $*_1$ is not.

2.3.9. Definition. A is a *variant* of B if for some $*_1$ and $*_2$ one has

$$A = B^{*_1} \text{ and } B = A^{*_2}.$$

2.3.10. EXAMPLE. $\alpha \to \beta \to \beta$ is a variant of $\gamma \to \delta \to \delta$ but not of $\alpha \to \beta \to \alpha$.

Note that if $*_1$ and $*_2$ are both most general unifiers of say A and B, then A^{*_1} and A^{*_2} are variants of each other and similarly for B.

The following result due to Robinson (1965) states that unifiers can be constructed effectively.

2.3.11. THEOREM (Unification theorem). (i) *There is a recursive function U having (after coding) as input a pair of types and as output either a substitutor or* `fail` *such that*

$$\begin{array}{rcl} A \text{ and } B \text{ have a unifier} & \Rightarrow & U(A,B) \text{ is a most general unifier} \\ & & \text{for } A \text{ and } B; \\ A \text{ and } B \text{ have no unifier} & \Rightarrow & U(A,B) = \mathsf{fail}. \end{array}$$

(ii) *There is (after coding) a recursive function U having as input finite sets of equations between types and as output either a substitutor or* `fail` *such that*

$$\begin{array}{rcl} E \text{ has a unifier} & \Rightarrow & U(E) \text{ is a most general unifier for } E; \\ E \text{ has no unifier} & \Rightarrow & U(E) = \texttt{fail}. \end{array}$$

PROOF. Note that $A_1 \to A_2 \equiv B_1 \to B_2$ holds iff $A_1 \equiv B_1$ and $A_2 \equiv B_2$ hold.

(i) Define $U(A,B)$ by the following recursive loop, using case distinction.

$$\begin{array}{rcll} U(\alpha, B) & = & [\alpha := B], & \text{if } \alpha \notin \mathrm{FV}(B), \\ & = & [\,], & \text{if } B = \alpha, \\ & = & \texttt{fail}, & \text{else;} \end{array}$$

$$\begin{array}{rcl} U(A_1 \to A_2, \alpha) & = & U(\alpha, A_1 \to A_2); \\ U(A_1 \to A_2, B_1 \to B_2) & = & U(A_1^{U(A_2,B_2)}, B_1^{U(A_2,B_2)}) \circ U(A_2, B_2), \end{array}$$

where this last expression is considered to be `fail` if one of its parts is. Let $\#_{var}(A,B)$ ='the number of variables in $A \to B$' and $\#_{\to}(A,B)$='the number of arrows in $A \to B$'. By induction on $(\#_{var}(A,B), \#_{\to}(A,B))$ ordered lexicographically one can show that $U(A,B)$ is always defined. Moreover U satisfies the specification.

(ii) If $E = \{A_1 = B_1, \ldots, A_n = B_n\}$, then define $U(E) = U(A,B)$, where $A = A_1 \to \cdots \to A_n$ and $B = B_1 \to \cdots \to B_n$. ■

See [???] for more on unification. The following result due to Parikh [1973] for propositional logic (interpreted by the propositions-as-types interpretation) and Wand [1987] simplifies the proof of the decidability of type checking and typeability for $\lambda_{\to}$.

2.3.12. PROPOSITION. *For every basis* Γ*, term* $M \in \Lambda$ *and* $A \in \mathbb{T}$ *such that* $\mathrm{FV}(M) \subseteq \mathrm{dom}(\Gamma)$ *there is a finite set of equations* $E = E(\Gamma, M, A)$ *such that for all substitutors* $*$ *one has*

$$
\begin{array}{rcll}
* \models E(\Gamma, M, A) & \Rightarrow & \Gamma^* \vdash M : A^*, & (1)\\
\Gamma^* \vdash M : A^* & \Rightarrow & *_1 \models E(\Gamma, M, A), & (2)\\
 & & \text{for some } *_1 \text{ such that } * \text{ and } *_1 \text{ have the same} & \\
 & & \text{effect on the type variables in } \Gamma \text{ and } A. &
\end{array}
$$

PROOF. Define $E(\Gamma, M, A)$ by induction on the structure of M:

$$
\begin{array}{rcl}
E(\Gamma, x, A) & = & \{A = \Gamma(x)\};\\
E(\Gamma, MN, A) & = & E(\Gamma, M, \alpha{\to}A) \ \cup \ E(\Gamma, N, \alpha),\\
 & & \text{where } \alpha \text{ is a fresh variable;}\\
E(\Gamma, \lambda x.M, A) & = & E(\Gamma \cup \{x{:}\alpha\}, M, \beta) \ \cup \ \{\alpha{\to}\beta = A\},\\
 & & \text{where } \alpha, \beta \text{ are fresh.}
\end{array}
$$

By induction on M one can show (using the generation lemma (2.1.3)) that (1) and (2) hold. ■

2.3.13. DEFINITION. (i) Let $M \in \Lambda$. Then (Γ, A) is a *principal pair* (pp) for M if

(1) $\Gamma \vdash M : A$.

(2) $\Gamma' \vdash M : A' \ \Rightarrow \ \exists * \, [\Gamma^* \subseteq \Gamma' \ \& \ A^* \equiv A']$.

Here $\{x_1{:}A_1, \ldots\}^* = \{x_1{:}A_1^*, \ldots\}$.

(ii) Let $M \in \Lambda$ be closed. Then A is a *principal type* (pt) for M if

(1) $\vdash M : A$

(2) $\vdash M : A' \ \Rightarrow \ \exists * \ [A^* \equiv A']$.

Note that if (Γ, A) is a *pp* for M, then every variant (Γ', A') of (Γ, A), in the obvious sense, is also a *pp* for M. Conversely if (Γ, A) and (Γ', A') are *pp*'s for M, then (Γ', A') is a variant of (Γ, A). Similarly for closed terms and *pt*'s. Moreover, if (Γ, A) is a *pp* for M, then $\mathrm{FV}(M) = \mathrm{dom}(\Gamma)$.

The following result is independently due to Curry (1969), Hindley (1969) and Milner (1978). It shows that for $\lambda{\to}$ the problems of type checking and typeability are decidable.

2.3.14. THEOREM (Principal type theorem for $\lambda_{\to}^{\mathrm{Cu}}$). (i) *There exists a computable function pp such that one has*

$$
\begin{array}{rcl}
M \text{ has a type} & \Rightarrow & pp(M) = (\Gamma, A), \text{ where } (\Gamma, A) \text{ is a pp for } M;\\
M \text{ has no type} & \Rightarrow & pp(M) = \mathtt{fail}.
\end{array}
$$

(ii) *There exists a computable function pt such that for closed terms* M *one has*

$$
\begin{array}{rcl}
M \text{ has a type} & \Rightarrow & pt(M) = A, \text{ where } A \text{ is a pt for } M;\\
M \text{ has no type} & \Rightarrow & pt(M) = \mathsf{fail}.
\end{array}
$$

PROOF. (i) Let $\mathrm{FV}(M) = \{x_1, \ldots, x_n\}$ and set $\Gamma_0 = \{x_1{:}\alpha_1, \ldots, x_n{:}\alpha_n\}$ and $A_0 = \beta$. Note that

$$\begin{array}{rcl} M \text{ has a type} & \Rightarrow & \exists \Gamma\, \exists A\ \ \Gamma \vdash M : A \\ & \Rightarrow & \exists *\ \ \Gamma_0^* \vdash M : A_0^* \\ & \Rightarrow & \exists *\ \ * \models E(\Gamma_0, M, A_0). \end{array}$$

Define

$$\begin{array}{rcll} pp(M) & = & (\Gamma_0^*, A_0^*), & \text{if } U(E(\Gamma_0, M, A_0)) = *; \\ & = & \mathtt{fail}, & \text{if } U(E(\Gamma_0, M, A_0)) = \mathtt{fail}. \end{array}$$

Then $pp(M)$ satisfies the requirements. Indeed, if M has a type, then

$$U(E(\Gamma_0, M, A_0)) = *$$

is defined and $\Gamma_0^* \vdash M : A_0^*$ by (1) in proposition 2.3.12. To show that (Γ_0^*, A_0^*) is a pp, suppose that also $\Gamma' \vdash M : A'$. Let $\widetilde{\Gamma} = \Gamma' \restriction \mathrm{FV}(M)$; write $\widetilde{\Gamma} = \Gamma_0^{*_0}$ and $A' = A_0^{*_0}$. Then also $\Gamma_0^{*_0} \vdash M : A_0^{*_0}$. Hence by (2) in proposition 2.3.12 for some $*_1$ (acting the same as $*_0$ on Γ_0, A_0) one has $*_1 \models E(\Gamma_0, M, A_0)$. Since $*$ is a most general unifier (proposition 2.3.11) one has $*_1 = *_2 \circ *$ for some $*_2$. Now indeed

$$(\Gamma_0^*)^{*_2} = \Gamma_0^{*_1} = \Gamma_0^{*_0} = \widetilde{\Gamma} \subseteq \Gamma'$$

and

$$(A_0^*)^{*_2} = A_0^{*_1} = A_0^{*_0} = A'.$$

If M has no type, then $\neg \exists *\ * \models E(\Gamma_0, M, A_0)$ hence

$$U(\Gamma_0, M, A_0) = \mathtt{fail} = pp(M).$$

(ii) Let M be closed and $pp(M) = (\Gamma, A)$. Then $\Gamma = \emptyset$ and we can put $pt(M) = A$. ■

2.3.15. COROLLARY. *Type checking and typeability for $\lambda_\to$ are decidable.*

PROOF. As to type checking, let M and A be given. Then

$$\vdash M : A \iff \exists * [A = pt(M)^*].$$

This is decidable (as can be seen using an algorithm—*pattern matching*—similar to the one in Theorem 2.3.11).

As to the question of typeability, let M be given. Then M has a type iff $pt(M) \neq \mathtt{fail}$. ■

The following result is due to Hindley [1969].

2.3.16. THEOREM (Second principal type theorem for $\lambda_\to^{\mathrm{Cu}}$). (i) *For every type $A \in \mathbb{T}$ one has*

$$\vdash M : A \ \Rightarrow\ \exists M' [M' \twoheadrightarrow_{\beta\boldsymbol{\eta}} M \ \&\ \mathtt{pt}(M') = A].$$

(ii) *For every type* $A \in \mathbb{TT}$ *there exists a basis* Γ *and term* $M \in \Lambda$ *such that* (Γ, A) *is a pp for* M.

PROOF. (i) We present a proof by examples. We choose three situations in which we have to construct an M' that are representative for the general case. Do exercise ?? for the general proof.

Case $M \equiv \lambda x.x$ and $A \equiv (\alpha\to\beta)\to\alpha\to\beta$. Then $\mathtt{pt}(M) \equiv \alpha\to\alpha$. Take $M' \equiv \lambda xy.xy$. The η-expansion of $\lambda x.x$ to $\lambda xy.xy$ makes subtypes of A correspond to unique subterms of M'.

Case $M \equiv \lambda xy.y$ and $A \equiv (\alpha\to\gamma)\to\beta\to\beta$. Then $\mathtt{pt}(M) \equiv \alpha\to\beta\to\beta$. Take $M' \equiv \lambda xy.\mathsf{K}y(\lambda z.xz)$. The β-expansion forces x to have a functional type.

Case $M \equiv \lambda xy.x$ and $A \equiv \alpha\to\alpha\to\alpha$. Then $\mathtt{pt}(M) \equiv \alpha\to\beta\to\alpha$. Take $M' \equiv \lambda xy.\mathsf{K}x(\lambda f.[fx, fy])$. The β-expansion forces x and y to have the same types.

(ii) Let A be given. We know that $\vdash \mathsf{I} : A\to A$. Therfore by (i) there exists an $\mathsf{I}' \twoheadrightarrow_{\beta\boldsymbol{\eta}} \mathsf{I}$ such that $\mathtt{pt}(\mathsf{I}') = A\to A$. Then take $M \equiv \mathsf{I}'x$. We have $\mathtt{pp}(\mathsf{I}'x) = (\{x{:}A\}, A)$. ■

Complexity

The space and time complexity of finding a type for a typable term is exponential, see exercise 2.5.18.

In order to decide whether for two typed terms $M, N \in \Lambda_{\to}(A)$ one has

$$M =_{\beta\eta} N,$$

one can normalize both terms and see whether the results are syntactically equal (up to α-conversion). In exercise 2.5.17 it will be shown that the time and space costs of doing this is at least hyper-exponential (in the size of MN). The reason is that the type-free application of Church numerals

$$\mathbf{c}_n\mathbf{c}_m = \mathbf{c}_{m^n}$$

can be typed, even when applied iteratively

$$\mathbf{c}_{n_1}\mathbf{c}_{n_2}\ldots\mathbf{c}_{n_k}.$$

In exercise 2.5.16 it is shown that the costs are also at most hyper-exponential. The reason is that Turing's proof of normalization for terms in $\lambda_{\to}$ uses a succesive development of redexes of 'highest' type. Now the length of each such development depends exponentially on the length of the term, whereas the length of a term increases at most quadratically at each reduction step. The result even holds for typable terms $M, N \in \Lambda_{\to\mathrm{Cu}}(A)$, as the cost of finding types only ads a simple exponential to the cost.

One may wonder whether there is not a more efficient way to decide $M =_{\beta\eta} N$, for example by using memory for the reduction of the terms, rather than a pure reduction strategy that only depends on the state of the term reduced so far. The sharpest question is whether there is any Turing computable method, that has a better complexity class. In Statman [1979] it is shown that this is

not the case, by showing that every elementary time bounded Turing machine computation can be coded as a a convertibility problem for terms of some type in $\lambda^o_\to$. A shorter proof of this result can be found in Mairson [1992].

2.4. Finding inhabitants

In this section we study for $\lambda_\to$ the problem of finding inhabitants. That is, given a type A, we look for terms M such that in the empty context $\vdash_{\lambda_\to} M : A$. By Corollaries 1.4.8 and 1.4.16 it does not matter whether we work in the system *à la* Curry, Chrurch or de Bruijn. Therefore we will focus on $\lambda^{\mathrm{Cu}}_\to$. Note that by proposition 2.1.2 the term M must be closed.

For example, if $A = \alpha{\to}\alpha$, then we can take $M \equiv \lambda x(:\alpha).x$. In fact we will see later that this M is modulo β-conversion the only choice. For $A = \alpha{\to}\alpha{\to}\alpha$ there are two inhabitants: $M_1 \equiv \lambda x_1 x_2.x_1 \equiv \mathsf{K}$ and $M_2 \equiv \lambda x_1 x_2.x_2 \equiv \mathsf{K}_*$. Again we have exhausted all inhabitants. If $A = \alpha$, then there are no inhabitants, as we will see soon.

Various interpretations will be useful to solve inhabitation problems.

The Boolean model

Type variables can be interpreted as ranging over $\mathbf{B} = \{0, 1\}$ and $\to$ as the two-ary function on $\mathbf{B}$ defined by

$$x{\to}y = 1 - x + xy$$

(classical implication). This makes every type A into a Boolean function. More formally this is done as follows.

2.4.1. Definition. (i) A *Boolean valuation* is a map $\rho : \mathbb{T}$ var$\to\mathbf{B}$.

(ii) Let ρ be a Boolean valuation. The *Boolean interpretation under* ρ of a type $A \in \mathbb{T}(\lambda_\to)$, notation $[\![A]\!]_\rho$, is defined inductively as follows.

$$\begin{array}{rcl} [\![\alpha]\!]_\rho & = & \rho(\alpha); \\ [\![A_1{\to}A_2]\!]_\rho & = & [\![A_1]\!]_\rho{\to}[\![A_2]\!]_\rho. \end{array}$$

(iii) A Boolean valuation ρ *satisfies a type* A, notation $\rho \models A$, if $[\![A]\!]_\rho = 1$. Let $\Gamma = \{x_1 : A_1, \ldots, x_n : A_n\}$, then ρ *satisfies* Γ, notation $\rho \models \Gamma$, if

$$\rho \models A_1 \ \& \ \ldots \ \& \ \rho \models A_n.$$

(iv) A type A is *classically valid*, notation $\models A$, iff for all Boolean valuations ρ one has $\rho \models A$.

2.4.2. Proposition. *Let $\Gamma \vdash_{\lambda_\to} M{:}A$. Then for all Boolean valuations ρ one has*

$$\rho \models \Gamma \ \Rightarrow \ \rho \models A.$$

Proof. By induction on the derivation in $\lambda_\to$. ■

From this it follows that inhabited types are classically valid. This in turn implies that the type α is not inhabited.

2.4.3. COROLLARY. (i) *If A is inhabited, then $\models A$.*
(ii) *A type variable α is not inhabited.*

PROOF. (i) Immediate by proposition 2.4.2, by taking $\Gamma = \emptyset$.
(ii) Immediate by (i), by taking $\rho(\alpha) = 0$. ■

One may wonder whether the converse of 2.4.3(i), i.e.

$$\models A \;\Rightarrow\; A \text{ is inhabited} \tag{1}$$

holds. We will see that in $\lambda_{\to}$ this is not the case. For the subsystem $\lambda^o_{\to}$(having only one base type o), however, the implication (1) is valid.

2.4.4. PROPOSITION (Statman [1982]). *Let $A = A_1 \to \ldots \to A_n \to o$, with $n \geq 1$ be a type of $\lambda^o_{\to}$. Then*

$$\begin{array}{rcl} A \textit{ is inhabited} & \iff & \textit{for some } i \textit{ with } 1 \leq i \leq n \textit{ the type} \\ & & A_i \textit{ is not inhabited.} \end{array}$$

PROOF. ($\Rightarrow$) Assume $\vdash_{\lambda^o_{\to}} M : A$. Suppose towards a contradiction that all A_i are inhabited, i.e. $\vdash_{\lambda^o_{\to}} N_i : A_i$. Then $\vdash_{\lambda^o_{\to}} MN_1 \ldots N_n : o$, contradicting 2.4.3(ii).

($\Leftarrow$) By induction on the structure of A. Assume that A_i with $1 \leq i \leq n$ is not inhabited.

Case 1. $A_i = o$. Then

$$x_1 : A_1, \ldots, x_n : A_n \vdash x_i : o$$

so

$$\vdash (\lambda x_1 \ldots x_n . x_i) : A_1 \to \ldots \to A_n \to o,$$

i.e. A is inhabited.

Case 2. $A_i = B_1 \to \ldots \to B_m \to o$. By (the contrapositive of) the induction hypothesis applied to A_i it follows that all B_j are inhabited, say $\vdash M_j : B_j$. Then

$$\begin{array}{ll} & x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i = B_1 \to \ldots \to B_m \to o \\ \Rightarrow & x_1 : A_1, \ldots, x_n : A_n \vdash x_i M_1 \ldots M_m : o \\ \Rightarrow & \vdash \lambda x_1 \ldots x_n . x_i M_1 \ldots M_m : A_1 \to \ldots \to A_n \to o = A. \blacksquare \end{array}$$

From the proposition it easily follows that inhabitation of terms in $\lambda^o_{\to}$ is decidable with a linear time algorithm.

2.4.5. COROLLARY. *In $\lambda^o_{\to}$ one has for all types A*

$$A \textit{ is inhabited} \iff \models A.$$

PROOF. ($\Rightarrow$) By proposition 2.4.3(i). ($\Leftarrow$) Assume $\models A$ and that A is not inhabited. Then $A = A_1 \to \ldots \to A_n \to o$ with each A_i inhabited. But then for $\rho_0(o) = 0$ one has

$$\begin{array}{rcl} 1 & = & [\![A]\!]_{\rho_0} \\ & = & [\![A_1]\!]_{\rho_0} \to \ldots \to [\![A_n]\!]_{\rho_0} \to o \\ & = & 1 \to \ldots \to 1 \to 0, \text{ since } \models A_i \text{ for all } i, \\ & = & 0, \text{ since } 1 \to 0 = 0, \end{array}$$

contradiction. ■

Corollary 2.4.5 does not hold for $\lambda^\infty_\to$. In fact the type $((\alpha \to \beta) \to \alpha) \to \alpha$ (corresponding to Peirce's law) is a valid type that is not inhabited, as we will see soon.

2.4.6. EXERCISE. Each type A of $\lambda^o_\to$ can be interpreted as an element $[\![A]\!] \in \mathbf{B}^\mathbf{B}$ as follows.

$$[\![A]\!](i) = [\![A]\!]_{\rho_i},$$

where $\rho_i(o) = i$. There are four elements in $\mathbf{B}^\mathbf{B}$

$$\{\lambda\!\!\lambda x \in \mathbf{B}.0, \lambda\!\!\lambda x \in \mathbf{B}.1, \lambda\!\!\lambda x \in \mathbf{B}.x, \lambda\!\!\lambda x \in \mathbf{B}.1 - x\}.$$

Prove that $[\![A]\!] = \lambda\!\!\lambda x \in \mathbf{B}.1$ iff A is inhabited and $[\![A]\!] = \lambda\!\!\lambda x \in \mathbf{B}.x$ iff A is not inhabited.

Intuitionistic propositional logic

Although inhabited types correspond to Boolean tautologies, not all such tautologies correspond to inhabited types. Intuitionistic logic provides a precise characterization of inhabited types. The underlying idea, the *propositions-as-types* correspondence will be explaned in more detail in §5.2 and then in §30.1.

2.4.7. DEFINITION. (i) The set of formulas of the implicational fragment of propositional logic, notation $\mathsf{form}(\mathrm{PROP})$, is defined by the following abstract syntax. Write $\mathsf{form} = \mathsf{form}(\mathrm{PROP})$.

$$\boxed{\begin{array}{rcl} \mathsf{form} & = & \mathsf{var} \mid \mathsf{form} \supset \mathsf{form} \\ \mathsf{var} & = & p \mid \mathsf{var}' \end{array}}$$

For example $p', p' \supset p, p' \supset (p' \supset p)$ are formulas.

(ii) Let Γ be a set of formulas and let A be a formula. Then A is derivable from Γ, notation $\Gamma \vdash_{PROP} A$, if $\Gamma \vdash A$ can be produced by the following formal system.

$$\boxed{\begin{array}{rcl} A \in \Gamma & \Rightarrow & \Gamma \vdash A \\ \Gamma \vdash A \supset B,\ \Gamma \vdash A & \Rightarrow & \Gamma \vdash B \\ \Gamma, A \vdash B & \Rightarrow & \Gamma \vdash A \supset B \end{array}}$$

NOTATION. (i) $q, r, s, t, \ldots$ stand for arbitrary propositional variables.

(ii) As usual $\Gamma \vdash A$ stands for $\Gamma \vdash_{PROP} A$ if there is little danger for confusion. Morover, $\Gamma \vdash A$ stands for $\emptyset \vdash A$.

2.4.8. EXAMPLE. (i) $\vdash A \supset A$;

(ii) $B \vdash A \supset B$;

(iii) $\vdash A \supset (B \supset A)$;

(iv) $A \supset (A \supset B) \vdash A \supset B$.

2.4.9. DEFINITION. Let $A \in \mathsf{form}(\text{PROP})$ and $\Gamma \subseteq \mathsf{form}(\text{PROP})$.

(i) We define $[A] \in \mathbb{TT}$ and $\Gamma_A \subseteq \mathbb{TT}$ as follows.

A	$[A]$	Γ_A
p	α	$\emptyset$
$P \supset Q$	$[P] \to [Q]$	$\Gamma_P \cup \Gamma_Q$

It so happens that $\Gamma_A = \emptyset$ and $A \equiv [A]$ for all A. But the setup will be needed for more complex logics and type theories.

(ii) Moreover, we set $[\Gamma] = \{x_A{:}A \mid A \in \Gamma\}$.

2.4.10. PROPOSITION. *Let $A \in \mathsf{form}(\text{PROP})$ and $\Delta \subseteq \mathsf{form}(\text{PROP})$. Then*

$$\Delta \vdash_{\text{PROP}} A \;\Rightarrow\; [\Delta] \vdash_{\lambda_\to} M : [A], \text{ for some } M.$$

PROOF. By induction on the generation of $\Delta \vdash A$.

Case 1. $\Delta \vdash A$ because $A \in \Delta$. Then $(x_A{:}[A]) \in [\Delta]$ and hence $[\Delta] \vdash x_A : [A]$. So we can take $M \equiv x_A$.

Case 2. $\Delta \vdash A$ because $\Delta \vdash B \supset A$ and $\Delta \vdash B$. Then by the induction hypothesis $[\Delta] \vdash P : [B] \to [A]$ and $[\Delta] \vdash Q : [B]$. Therefore, $[\Delta] \vdash PQ : [A]$.

Case 3. $\Delta \vdash A$ because $A \equiv B \supset C$ and $\Delta, B \vdash C$. By the induction hypothesis $[\Delta], x_B{:}[B] \vdash M : [C]$. Hence $[\Delta] \vdash (\lambda x_B.M) : [B] \to [C] \equiv [B \supset C] \equiv [A]$. ■

Conversely we have the following.

2.4.11. PROPOSITION. *Let $\Delta, A \subseteq \mathsf{form}(\text{PROP})$. Then*

$$[\Delta] \vdash_{\lambda_\to} M : [A] \;\Rightarrow\; \Delta \vdash_{\text{PROP}} A.$$

PROOF. By induction on the structure of M.

Case 1. $M \equiv x$. Then by the generation lemma 2.1.3 one has $(x{:}[A]) \in [\Delta]$ and hence $A \in \Delta$; so $\Delta \vdash_{\text{PROP}} A$.

Case 2. $M \equiv PQ$. By the generation lemma for some $C \in \Lambda$ one has $[\Delta] \vdash P : C \to [A]$ and $[\Delta] \vdash Q : C$. Clearly, for some $C' \in \mathsf{form}$ one has $C \equiv [C]$. Then $C \to [A] \equiv [C' \supset A]$. By the induction hypothesis one has $\Delta \vdash C' \to A$ and $\Delta \vdash C'$. Therefore $\Delta \vdash A$.

Case 3. $M \equiv \lambda x.P$. Then $[\Delta] \vdash \lambda x.P : [A]$. By the generation lemma $[A] \equiv B \to C$ and $[\Delta], x{:}B \vdash P : C$, so that $[\Delta], x{:}[B'] \vdash P : [C']$, with $B'] \equiv B, [C'] \equiv C$ (hence $[A] \equiv [B' \supset C']$). By the induction hypothesis it follows that $\Delta, B \vdash C$ and therefore $\Delta \vdash B \to C \equiv A$. ■

Although intuitionistic logic gives a complete characterization of those types that are inhabited, this does not answer immediately the question whether a type like $((\alpha\to\beta)\to\alpha)\to\alpha$ corresponding to Peirce's law is inhabited.

Kripke models

Remember that a type $A\in\mathbb{T}(\lambda_\to)$ is inhabited iff $[A]$, the translation of A into propositional calculus is intuitionistically provable. This explains why

$$A \text{ inhabited } \Rightarrow \models A,$$

but not conversely, since $\models A$ correspond to classical validity. A common tool to prove that types are not inhabited or that formulas are not intuitionistically derivable consist of so called *Kripke models* that we will introduce now.

2.4.12. Definition. (i) A *Kripke model* consists of a tuple $\mathcal{K} =< K, \leq, \bot, F >$, such that

(1) $< K, \leq, \bot >$ is a partially ordered set with least element $\bot$;

(2) $F : K\to\wp(\mathsf{var})$ is a monotonic map from K to the powerset of the set of type-variables: $\forall k, k' \in K\ [k \leq k' \ \Rightarrow\ F(k) \subseteq F(k')]$.

We often just write $\mathcal{K} =< K, F >$.

(ii) Let $\mathcal{K} =< K, F >$ be a Kripke model. For $k\in K$ we define by induction on the structure of $A\in\mathbb{T}(\lambda_\to)$ the notion k *forces* A, notation $k \Vdash_{\mathcal{K}} A$. We often omit the subscript.

$$\begin{aligned} k \Vdash \alpha &\iff \alpha \in F(k);\\ k \Vdash A_1\to A_2 &\iff \forall k' \geq k\ [k' \Vdash A_1 \ \Rightarrow\ k' \Vdash A_2]. \end{aligned}$$

(iii) $\mathcal{K}$ forces A, notation $\mathcal{K} \Vdash A$, is defined as $\bot \Vdash_{\mathcal{K}} A$.

(iv) If $\Gamma = \{x_1{:}A_1, \ldots, x_n{:}A_n\}$, then we write $\mathcal{K} \Vdash \Gamma$, pronounce $\mathcal{K}$ forces Γ, for $\mathcal{K} \Vdash A_1$ & $\ldots$ & $\mathcal{K} \Vdash A_n$. Define $\Gamma \Vdash A$, pronounce 'Γ forces A', iff for all Kripke models $\mathcal{K}$ one has

$$\mathcal{K} \Vdash \Gamma \ \Rightarrow\ \mathcal{K} \Vdash A.$$

In particular $\Vdash A$, pronounce 'forced A', if $\mathcal{K} \Vdash A$ for all Kripke models $\mathcal{K}$.

2.4.13. Lemma. *Let $\mathcal{K}$ be a Kripke model. Then for all $A\in\mathbb{T}(\lambda_\to)$ one has*

$$k \leq k' \ \&\ k \Vdash_{\mathcal{K}} A \ \Rightarrow\ k' \Vdash_{\mathcal{K}} A.$$

Proof. By induction on the structure of A. ■

2.4.14. Proposition. $\Gamma \vdash_{\lambda_\to} M : A \ \Rightarrow\ \Gamma \Vdash A$.

Proof. By induction on the derivation of $M : A$ from Γ. If $M : A$ is $x : A$ and is in Γ, then this is trivial. If $\Gamma \vdash M : A$ is $\Gamma \vdash FP : A$ and is a direct consequence of $\Gamma \vdash F : B\to A$ and $\Gamma \vdash P : B$, then the conclusion follows from the induction hypothesis and the fact that $k \Vdash B\to A \ \&\ k \Vdash B \ \Rightarrow\ k \Vdash A$. In the case that $\Gamma \vdash M : A$ is $\Gamma \vdash \lambda x.N : A_1\to A_2$ and follows directly from

$\Gamma, x{:}A_1 \vdash N : A_2$ we have to do something. By the induction hypothesis we have for all $\mathcal{K}$

$$\mathcal{K} \Vdash \Gamma, A_1 \;\Rightarrow\; \mathcal{K} \Vdash A_2. \tag{2}$$

We must show $\Gamma \Vdash A_1 \to A_2$, i.e. $\mathcal{K} \Vdash \Gamma \;\Rightarrow\; \mathcal{K} \Vdash A_1 \to A_2$ for all $\mathcal{K}$.

Given $\mathcal{K}$ and $k \in K$, define

$$\mathcal{K}_k =< \{k' \in K \mid k \leq k'\}, \leq, k, K >,$$

(where $\leq$ and F are in fact the appropriate restrictions to the subset $\{k' \in K \mid k \leq k'\}$ of K). Then it is easy to see that also $\mathcal{K}_k$ is a Kripke model and

$$k \Vdash_{\mathcal{K}} A \iff \mathcal{K}_k \Vdash A. \tag{3}$$

Now suppose $\mathcal{K} \Vdash \Gamma$ in order to show $\mathcal{K} \Vdash A_1 \to A_2$, i.e. for all $k \in \mathcal{K}$

$$k \Vdash_{\mathcal{K}} A_1 \;\Rightarrow\; k \Vdash_{\mathcal{K}} A_2.$$

Indeed,

$$\begin{array}{lcll} k \Vdash_{\mathcal{K}} A_1 & \Rightarrow & \mathcal{K}_k \Vdash A_1, & \text{by (3)} \\ & \Rightarrow & \mathcal{K}_k \Vdash A_2, & \text{by (2), since by lemma 2.4.13 also } \mathcal{K}_k \Vdash \Gamma, \\ & \Rightarrow & k \Vdash_{\mathcal{K}} A_2. \blacksquare & \end{array}$$

2.4.15. Corollary. *Let $A \in \mathbb{T}(\lambda_\to)$. Then*

$$A \textit{ is inhabited} \;\Rightarrow\; \Vdash A.$$

Proof. Take $\Gamma = \emptyset$. ■

Now it can be proved, see exercise 2.4.16, that (the type corresponding to) Peirce's law $P = ((\alpha \to \beta) \to \alpha) \to \alpha$ is not forced in some Kripke model. Since $\not\Vdash P$ it follows that P is not inhabited, in spite of the fact that $\models P$.

2.4.16. Exercise. Show that Peirce's law $P = ((\alpha \to \beta) \to \alpha) \to \alpha$ is not forced in the Kripke model $\mathcal{K} =< K, \leq, 0, F >$ with $K = \{0, 1\}$, $0 \leq 1$ and $F(0) = \emptyset, F(1) = \{\alpha\}$.

We also have a converse to corollary 2.4.15 which theoretically answers the inhabitation question for $\lambda_\to$, namely the completeness theorem, Kripke [ ], (usually formulated for provability in intuitionistic logic):

$$A \text{ is inhabited} \iff \Vdash A.$$

This is done by constructing for a type that is not inhabited a Kripke 'countermodel' $\mathcal{K}$, i.e. $\mathcal{K} \not\Vdash A$. In [ ] it is shown that these Kripke countermodels can be taken to be finite. This solves the decision problem for inhabitation in $\lambda_\to$. In Statman [1979] the decision problem is shown to be PSPACE complete, so that further analysis of the complexity of the decision problem appears to be very difficult.

Set-theoretic models

Now we will prove using set-theoretic models that there do not exists terms satisfying certain properties.

2.4.17. DEFINITION. An $A \times A \to A$ *pairing* is a triple (P, L, R) such that in $\lambda_\to$ one has

$$\begin{aligned} &\vdash P : A \to A \to A; \\ &\vdash L : A \to A \ \& \ \vdash R : A \to A; \\ &L(Pxy) =_{\beta\boldsymbol{\eta}} x \ \& \ L(Pxy) =_{\beta\boldsymbol{\eta}} y. \end{aligned}$$

It can be shown that there does not exists a $A \times A \to A$ pairing for arbitrary types A, see Barendregt [1974].

2.4.18. DEFINITION. Let X be a set. The *full typed structure* (for $\lambda^o_\to$ types) over X, notation $\mathcal{M}_X = \{X(A)\}_{A \in \mathbb{T}^o}$, is defined as follows. For $A \in \mathbb{T}^o$ let $X(A)$ be defined inductively as follows.

$$\begin{aligned} X(o) &= X; \\ X(A \to B) &= X(B)^{X(A)}, \text{ the set of functions from } X(A) \text{ into } X(B). \end{aligned}$$

The reason that $\lambda^o_\to$-classic was introduced can be seen now from the way terms of that system can be interpreted easily into $\mathcal{M}_X$.

2.4.19. DEFINITION. (i) A *valuation* in $\mathcal{M}_X$ is a map ρ from typed variables into $\cup_A X(A)$ such that $\rho(x^A) \in X(A)$ for all $A \in \mathbb{T}^o$.

(ii) Let ρ be a valuation in $\mathcal{M}_X$. The *interpretation under* ρ of a $\lambda^o_\to$-classic term into $\mathcal{M}_X$, notation $[\![M]\!]_\rho$, is defined as follows.

$$\begin{aligned} [\![x^A]\!]_\rho &= \rho(x^A); \\ [\![MN]\!]_\rho &= [\![M]\!]_\rho [\![N]\!]_\rho; \\ [\![\lambda x^A.M]\!]_\rho &= \boldsymbol{\lambda} d \in X(A).[\![M]\!]_{\rho(x^A := d)}, \end{aligned}$$

where $\rho(x^A := d) = \rho'$ with $\rho'(x^A) = d$ and $\rho'(y^B) = \rho(y^B)$ if $y^B \not\equiv x^A$.[1]

(iii) Define

$$\mathcal{M}_X \models M = N \iff \forall \rho \, [\![M]\!]_\rho = [\![N]\!]_\rho.$$

Before proving properties about the models it is good to do exercises 2.5.8 and 2.5.9.

2.4.20. PROPOSITION. (i) $M \in \mathtt{term}_A \Rightarrow [\![M]\!]_\rho \in X(A)$.

(ii) $M =_{\beta\boldsymbol{\eta}} N \Rightarrow \mathcal{M}_X \models M = N$.

PROOF. (i) By induction on the structure of M.

[1] Sometimes it is prefered to write $[\![\lambda x^A.M]\!]_\rho$ as $\boldsymbol{\lambda} d \in X(A).[\![M[x^A := \underline{d}]]\!]_\rho$, where $\underline{d}$ is a constant to be interpreted as d. Although this notation is perhaps more intuitive, we will not use it, since it also has technical drawbacks.

(ii) By induction on the 'proof' of $M =_{\beta\boldsymbol{\eta}} N$, using

$[\![M[x:=N]]\!]_\rho = [\![M]\!]_{\rho(x:=[\![N]\!]_\rho)}$, for the β-rule;

$\rho \upharpoonright \mathrm{FV}(M) = \rho' \upharpoonright \mathrm{FV}(M) \;\Rightarrow\; [\![M]\!]_\rho = [\![M]\!]_{\rho'}$, for the $\boldsymbol{\eta}$-rule;

$[\forall d \in X(A)\ [\![M]\!]_{\rho(x:=d)} = [\![N]\!]_{\rho(x:=d)}] \;\Rightarrow\; [\![\lambda x^A.M]\!]_\rho = [\![\lambda x^A.N]\!]_\rho$, for the

ξ-rule. ■

Now we will give applications of the notion of typed structure.

2.4.21. Proposition. *Let A be a type in $\lambda^o_\to$. Then there does not exists an $A \times A \to A$ pairing.*

Proof. Take $X = \{0,1\}$. Then for every type A the set $X(A)$ is finite. Therefore by a cardinality argument there cannot be an $A \times A \to A$ pairing, for otherwise f defined by

$$f(x,y) = [\![P]\!]xy$$

would be an injection from $X(A) \times X(A)$ into $X(A)$, do exercise 2.5.9. ■

2.4.22. Proposition. *There does not exists a term* $\mathtt{pred}$ *such that* $\vdash_{\lambda^o_\to} \mathtt{pred} : \mathsf{Nat} \to \mathsf{Nat}$ *and*

$$\begin{aligned} \mathtt{pred}\,\ulcorner 0 \urcorner &= \ulcorner 0 \urcorner; \\ \mathtt{pred}\,\ulcorner n+1 \urcorner &= \ulcorner n \urcorner. \end{aligned}$$

Proof. As before for $X = \{0,1\}$ the set $X(\mathsf{Nat})$ is finite. Therefore

$$\mathcal{M}_X \models \ulcorner n \urcorner = \ulcorner m \urcorner$$

for some $n \neq m$. If $\mathtt{pred}$ did exist, then it follows easily that $\mathcal{M}_X \models \ulcorner 0 \urcorner = \ulcorner 1 \urcorner$. But this implies that $X(o)$ has cardinality 1, since $\ulcorner 0 \urcorner(\mathsf{K}x)y = y$ but $\ulcorner 1 \urcorner(\mathsf{K}x)y = \mathsf{K}xy = x$, a contradiction. ■

Another application of semantics is that there are no fixed-point operators in $\lambda^o_\to$.

2.4.23. Definition. Let A be a type of $\lambda^o_\to$. A term Y is a *fixed-point operator of type A* iff $\vdash_\tau Y : (A \to A) \to A$ and

$$Y =_{\beta\boldsymbol{\eta}} \lambda f{:}A \to A.f(\mathsf{Y}f).$$

2.4.24. Proposition. *For no type A there exists in $\lambda^o_\to$ a fixed-point combinator.*

Proof. Take $X = \{0,1\}$. Then for every A the set $X(A)$ has at least two elements, say $x, y \in X(A)$ with $x \neq y$. Then there exists an $f \in X(A \to A)$ without a fixed-point:

$$\begin{aligned} f(z) &= x, \quad \text{if } z \neq x; \\ f(z) &= y, \quad \text{else.} \end{aligned}$$

Suppose a fixed-point combinator of type A did exist, then in $\mathcal{M}_X$ one has

$$[\![\mathsf{Y}]\!]f = f([\![\mathsf{Y}]\!]f),$$

contradicting that f has no fixed-point. ■

The results can easily be generalized to $\lambda_{\rightarrow}$, do exercise 2.5.10.

2.5. Exercises

2.5.1. Find out which of the following terms are typeable and determine for those that are the principal type.

$$\lambda xyz.xz(yz);$$
$$\lambda xyz.xy(xz);$$
$$\lambda xyz.xy(zy).$$

2.5.2. (i) Let $A = (\alpha{\rightarrow}\beta){\rightarrow}((\alpha{\rightarrow}\beta){\rightarrow}\alpha){\rightarrow}\alpha$ Construct a term M such that $\vdash M : A$. What is the principal type of M?
(ii) Find an expansion of M such that it has A as principal type.

2.5.3. (Uniqueness of Type Assignments) A K-redex is an $R \equiv (\lambda x.M)N$ with $x \notin \mathrm{FV}(M)$; if $x \in \mathrm{FV}(M)$, then R is an I-*redex*. In the following we consider a Church typing of an untyped term M and all of its subterms so that $M \in \Lambda_{\rightarrow \mathrm{Ch}}(A)$.
(i) Use the principal type theorem to show that if more than one such typing exists then there is one which contains an atomic type not in A.
(ii) Show that if M only has lambda-I redexes, then every atomic type in such a typing must occur in A [Hint: Show this for $\beta\boldsymbol{\eta}$-nf and show that atomic types are preserved by $\beta\boldsymbol{\eta}$-reduction of $\lambda\mathsf{I}$-terms].
(iii) Conclude that if a term $M \in \Lambda^{\mathrm{Cu}}_{\rightarrow}(A)$ has only I-redexes, then it has exactly one typing: if $M_1, M_2 \in \Lambda^{\mathrm{Ch}}_{\rightarrow}(A)$ such that $|M_1| \equiv |M_2| \equiv M$, then $M_1 \equiv M_2$.
(iv) Give an example of a $\lambda\mathsf{K}$-term $M \in \Lambda$ of type A with a non-unique Church typing of its subterms.
(v) Explain why an untyped term with only I-redexes may have more than one type.

2.5.4. (i) Formulate and prove an appropriate version of the Church-Rosser theorem for $\lambda^{\mathrm{Ch}}_{\rightarrow}$. [Hint. One possibility is to redo the proof of this result for untyped terms, as e.g. in B[1984]. An alternative is to use Proposition 1.4.9, Theorem 2.1.8 for $\lambda^{\mathrm{Cu}}_{\rightarrow}$, the normalization theorem for $\lambda_{\rightarrow}$ 2.1.19 and Exercise 2.5.3.
(ii) Show that $\lambda^{\mathrm{dB}}_{\rightarrow}$ satisfies the Church-Rosser Theorem. [Hint. Use (i) and translations between $\lambda^{\mathrm{dB}}_{\rightarrow}$ and $\lambda^{\mathrm{Ch}}_{\rightarrow}$.]

2.5.5. (Hindley) Show that if $\vdash^{\mathrm{Cu}}_{\lambda_{\rightarrow}} M : A$, then there is a M' such that

$$M' \twoheadrightarrow_{\beta\boldsymbol{\eta}} M \;\&\; \mathtt{pt}(M') = A.$$

[Hints. 1. First make an $\boldsymbol{\eta}$-expansion of M in order to obtain a term with a principal type having the same tree as A. 2. Show that for any type B with a subtype B_0 there exists a context $C[\,]$ such that

$$z{:}B \vdash C[z] : B_0.$$

3. Use 1,2 and a term like $\lambda fz.z(fP)(fQ)$ to force identification of the types of P and Q. (For example one may want to identify α and γ in $(\alpha{\to}\beta){\to}\gamma{\to}\delta$.)]

2.5.6. Prove that $\Lambda^o_{\to}(o) = \emptyset$ by applying the normalization and subject reduction theorems.

2.5.7. Let X be a set and consider the model $\mathcal{M}_X$ of $\lambda^o_{\to}$. Notice that every permutation $\pi = pi_o$ (bijection) of X can be lifted to all levels $X(A)$ by defining

$$\pi_{A\to B}(f) = \pi_B \circ f \circ \pi_A^{-1}.$$

Prove that every lambda definable element $f \in X(A)$ in $\mathcal{M}(X)$ is invariant under all lifted permutations; i.e. $\pi_A(f) = f$. [Hint. Use the fundamental theorem for logical relations.]

2.5.8. (i) Show that $\mathcal{M}_X \models (\lambda x^A.x^A)y^A = y^A$.
(ii) Show that $\mathcal{M}_X \models (\lambda x^{A\to A}.x^{A\to A}) = (\lambda x^{A\to A}\lambda y^A.x^{A\to A}y^A)$.
(iii) Show that $[\![\ulcorner 2 \urcorner(\mathsf{K}x^o)y^o]\!]_\rho = \rho(x)$.

2.5.9. Let P, L, R be an $A \times B{\to}C$ pairing. Show that in every structure $\mathcal{M}_X$ one has

$$[\![P]\!]xy = [\![P]\!]x'y' \;\Rightarrow\; x = x' \;\&\; y = y',$$

hence $\mathrm{card}(A).\mathrm{card}(B){\leq}\mathrm{card}(C)$.

2.5.10. Show that propositions 2.4.21, 2.4.22 and 2.4.24 can be generalized to $\lambda_{\to}$ by modifying the notion of typed structure.

2.5.11. Prove that $\Lambda^o_{\to}(o) = \emptyset$ by applying models and the fact shown in the previous exercise that lambda definable elements are invariant under lifted permutations.

2.5.12. Let ${\sim}A \equiv A{\to}o$. Show that if o does not occur in A, then ${\sim}{\sim}({\sim}{\sim}A{\to}A)$ is not inhabited. Why is the condition about o necessary?

2.5.13. We say that the structure of the rational numbers can be represented in $\lambda_{\to}$ if there is a type $Q \in \mathbb{TT}(\lambda_{\to})$ and closed lambda terms:

$$\begin{aligned} &0, 1 : Q;\\ &+, \cdot : Q{\to}Q{\to}Q;\\ &-, {}^{-1} : Q{\to}Q; \end{aligned}$$

such that $(Q, +, \cdot, -, {}^{-1}, 0, 1)$ satisfies the axioms of a field of characteristic 0. Show that the rationals cannot be represented in $\lambda_{\to}$.

2.5.14. Show that there is no closed term

$$P : \mathsf{Nat}{\to}\mathsf{Nat}{\to}\mathsf{Nat}$$

such that P is a bijection in the sense that

$$\forall M{:}\mathsf{Nat}\exists!N_1, N_2{:}\mathsf{Nat}\ PN_1N_2 =_{\beta\boldsymbol{\eta}} M.$$

2.5.15. Show that every closed term of type $(0{\to}0{\to}0){\to}0{\to}0$ is $\beta\boldsymbol{\eta}$-convertible to $\lambda f^{0\to0\to0}.\lambda x^0.t$ with t given by the grammar

$$t := x \mid ftt.$$

The next two exercises show that the minimal length of a reduction-path of a term to normal form is non-elementary in the length of the term[2]. See Péter [1967] for the definition of the class of (Kalmar) elementary functions. This class is the same as $\mathcal{E}_3$ in the Grzegorczyk hierarchy. To get some intuition for this class, define the family of functions $2_n{:}\mathbb{N}{\to}\mathbb{N}$ as follows.

$$\begin{aligned} 2_0(x) &= x; \\ 2_{n+1}(x) &= 2^{2_n(x)}. \end{aligned}$$

Then every elementary function f is eventually bounded by some 2_n:

$$\exists n, m\forall x{>}m\ f(x) \leq 2_n(x).$$

2.5.16. (i) Define the function $\mathtt{gk} : \mathbb{N}{\to}\mathbb{N}$ by

$$\begin{aligned} \mathtt{gk}(m) &= \#F_{\mathtt{GK}}(M), && \text{if } m = \#(M) \text{ for some untyped} \\ & && \text{lambda term } M; \\ &= 0, && \text{else.} \end{aligned}$$

Here $\#M$ denotes the Gödelnumber of the term M and $F_{\mathtt{GK}}$ is the Gross-Knuth reduction strategy defined by completely developing all present redexes in M, see B[1984]. Show that $\mathtt{gk}$ is Kalmar elementary.

(ii) For a type $A \in \mathbb{T}_\infty$ define its *depth* $D(A) \in \mathbb{N}$ by

$$\begin{aligned} D(\alpha) &= 0; \\ D(A{\to}B) &= \mathtt{max}(D(A), D(B)) + 1. \end{aligned}$$

For a term $M \in \Lambda^{\mathrm{Ch}}_{\to}$ define

$$D(M) = \mathtt{max}\{D(A{\to}B) \mid (\lambda x^A.P)^{A\to B}Q \text{ is a redex in } M\}$$

Show that

$$F_{\mathtt{GK}}(|M|) = |N| \ \Rightarrow\ D(M) > D(N).$$

[Hint. Use Lévy's analysis of redex creation, see Barendregt [1984], exercise 14.5.3.]

[2] In Gandy [1980a] this is also proved for arbitrary reduction paths starting from typable terms. In de Vrijer [1987] an exact calculation is given for the longest reduction paths to normal form.

(iii) If M is a term, then its *length*, notation $\mathtt{lth}(M)$, is the number of symbols in M. If a closed term $M \in \Lambda$ has a type, then its principal type $\mathtt{pt}(M)$ has a depth bounded by its length:

$$D(\mathtt{pt}(M)) \leq \mathtt{lth}(M).$$

See the proof of theorem 2.3.14.

(iv) Write $\sigma{:}M \to M^{\mathtt{nf}}$ if σ is some reduction path of M to normal form $M^{\mathtt{nf}}$. Let $\$\sigma$ be the number of reduction steps in σ. Define

$$\$(M) = \mathtt{min}\{\$\sigma \mid \sigma : M \to M^{\mathtt{nf}}\}.$$

Show that $\$M \leq g(\mathtt{lth}(M))$, for some function $g \in \mathcal{E}_4$. [Hint. Take $g(m) = gk^m(m)$.]

2.5.17. (i) Define $\mathbf{2}_1 = \lambda f^{o\to o} x^o . f(fx)$ and $\mathbf{2}_{n+1} = ([o{:=}o\to o])\mathbf{2}_n)\mathbf{2}$. Then the type of $\mathbf{2}_n$ is principal.

(ii) [Church] Show $([o{:=}o\to o]\mathbf{c}_n)\mathbf{c}_m =_\beta \mathbf{c}_{m^n}$.

(iii) Show $\mathbf{2}_n =_\beta \mathbf{c}_{2_n(1)}$.

(iv) Let $M, N \in \Lambda$ be untyped terms. Show that if $M \twoheadrightarrow_\beta N$, then

$$\mathtt{length}(M) \leq \mathtt{length}(N)^2.$$

(v) Conclude that the shortest length of a reduction path starting with a typed term M to normal form is in the worst case non-elementary in the length of M.

2.5.18. It will be shown that the length of the principal type of a typable term is at least exponential in the length of the term. This means that if $f(m) = \mathtt{max}\{\mathtt{length}(\mathtt{pt}(M)) \mid \mathtt{length}(M) \leq m\}$, then for some real number $c_1 > 1$ one has $c_1^n \leq f(n)$, for sufficiently large n. It will be shown also that the depth of the principal type of a typable term M is linear in the length of M. It follows that the length of the principal type of M is also at most exponential in the length of M.

(i) Define M_n to be the following term:

$$\lambda x_n \ldots x_1 . (x_n(x_n(x_{n-1}))(x_{n-1}(x_{n-1}x_{n-2})) \ldots (x_2(x_2x_1)).$$

Show that the principal type of M_n has length $> 2^n$. Conclude that the length of the principle type of a term is at least exponential in the length of the term.

(ii) Show that there is a constant c_2 such that for typable lambda terms M one has for sufficiently long M

$$\mathtt{depth}(\mathtt{pt}(M)) \leq c_2(\mathtt{length}(M)).$$

[Hint Use exercise 2.5.16(iii).]

(iii) Conclude that the length of the principle type of a term is also at most exponential.

2.5.19. (Statman) In this exercise $\mathbb{T} = \mathbb{T}^o$. Let $\mathcal{M}_n = \mathcal{M}(\{0, 1, 2, \ldots, n\})$. The purpose of this exercise is to show that this structure can be isomorphically embedded into $\mathcal{M}(\mathbb{N})$.

(i) (Church's δ) We add to the language $\lambda_\to$ constants $\underline{k} : o$ for $k \leq n$ and a constant $\underline{\delta} : o^4 \to o$. The intended interpretation of $\underline{\delta}$ is the function δ defined by

$$\begin{array}{rcll} \delta xyuv & = & u & \text{if } x = y; \\ & = & v & \text{else.} \end{array}$$

We define the notion of reduction δ by the contraction rules

$$\begin{array}{rcll} \underline{\delta}\,\underline{i}\,\underline{j}\,\underline{k}\,\underline{l} & \to_\delta & \underline{k} & \text{if } i = j; \\ & \to_\delta & \underline{l}, & \text{if } i \neq j. \end{array}$$

The resulting language of terms is called Λ_δ and on this we consider the notion of reduction $\to_{\beta\boldsymbol{\eta}\delta}$.

(ii) Show that every $M \in \Lambda_\delta$ satisfies $\mathrm{SN}_{\beta\boldsymbol{\eta}\delta}(M)$.

(iii) Show that $\to_{\beta\boldsymbol{\eta}\delta}$ is Church-Rosser.

(iv) Let $M \in \Lambda_\delta^\emptyset(o)$ be a closed term of type o. Show that the normal form of M is one of the constants $\underline{0}, \ldots, \underline{n}$.

(v) (Church's theorem) Show that every element $\Phi \in \mathcal{M}_n$ can be defined by a closed term $M_\Phi \in \Lambda_\delta$. [Hint. For each $A \in \mathbb{T}$ define simultaneously the map $\Phi \mapsto M_\Phi : \mathcal{M}_n(A) \to \Lambda_\delta(A)$ and $\delta_A \in \Lambda_\delta(A \to A \to o \to o)$ such that for any closed M, N of type A one has

$$\begin{array}{rcll} \delta_A MNuv & =_{\beta\boldsymbol{\eta}\delta} & u & \text{if } [\![M]\!]^{\mathcal{M}_n} = [\![N]\!]^{\mathcal{M}_n}; \\ & =_{\beta\boldsymbol{\eta}\delta} & v & \text{else.} \end{array}$$

For $A = o$ take $M_i = \underline{i}$ and $\delta_o = \delta$. For $A = B \to C$, let $\mathcal{M}_n(B) = \{\Phi_1, \ldots, \Phi_t\}$ and let $B = B_1 \to \ldots B_k \to o$. Now set

$$\begin{array}{rcl} \delta_A & \equiv & \lambda xyuv.\delta_B(xM_{\Phi_1}(yM_{\Phi_1})(\ldots(\delta_A(xM_{\Phi_t}(y\Phi_t))uv)..))v. \\ M_\Phi & \equiv & \lambda x\vec{y}. \\ & & \delta_B x M_{\Phi_1}(M_{\Phi\Phi_1}\vec{y}\,) \\ & & (\delta_B x M_{\Phi_2}(M_{\Phi\Phi_2}\vec{y}\,) \\ & & (\ldots \\ & & (\delta_B x M_{\Phi_t}(M_{\Phi\Phi_t}\vec{y}\,)\underline{0})..)). \end{array}$$

(vi) Show that $\Phi \mapsto [\![M_\Phi]\!]^{\mathcal{M}(\mathbb{N})}$ is the required isomorphic embedding of $\mathcal{M}_n$ into $\mathcal{M}(\mathbb{N})$.

(vii) (To be used later.) Let $\pi_i^n \equiv (\lambda x_1 \ldots x_n.x_i) : (o^n \to o)$. Define

$$\begin{array}{rcl} \Delta_n & \equiv & \lambda abuv\vec{x}.a \\ & & (b(u\vec{x})(v\vec{x})(v\vec{x})\ldots(v\vec{x})) \\ & & (b(v\vec{x})(u\vec{x})(v\vec{x})\ldots(v\vec{x})) \\ & & \ldots \\ & & (b(v\vec{x})(v\vec{x})\ldots(v\vec{x})(u\vec{x})). \end{array}$$

Then

$$\begin{array}{lll} \Delta^n\pi_i^n\pi_j^n\pi_k^n\pi_l^n & =_{\beta\boldsymbol{\eta}\delta} & \pi_k^n, \quad \text{if } i=j; \\ & =_{\beta\boldsymbol{\eta}\delta} & \pi_l^n, \quad \text{else.} \end{array}$$

Show that for $i \in \{1,\dots,n\}$ one has for all $M : o$

$$\begin{array}{l} M =_{\beta\boldsymbol{\eta}\delta} \underline{i} \Rightarrow \\ M[o := o^n{\to}o][\delta := \Delta^n][\underline{1} := \pi_1^n]\dots[\underline{n} := \pi_n^n] =_{\beta\boldsymbol{\eta}} \pi_i^n. \end{array}$$

2.5.20. (Th. Joly)

(i) Let$M = \langle Q, q_0, F, \delta\rangle$ be a deterministic finite automaton over the finite alphabet $\Sigma = \{a_1,\dots,a_n\}$. That is, Q is the finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final states and $\delta : \Sigma \times Q{\to}Q$ is the transition function. Let $L^r(M)$ be the (regular) language consisting of words in Σ^* accepted by M by reading the words from right to left. Let $\mathcal{M} = \mathcal{M}(Q)$ be the model of $\lambda^o_{\to}$ over Q. Show that

$$w \in L^r(M) \iff [\![\underline{w}]\!]^{\mathcal{M}}\delta_{a_1}\dots\delta_{a_n}q_0 \in F,$$

where $\delta_a(q) = \delta(a,q)$ and $\underline{w}$ is defined in 1.3.6.

(ii) Similarly represent classes of trees (with at the nodes elements of Σ) accepted by a frontier-to-root tree automaton, see Thatcher [1973], by the model $\mathcal{M}$ at the type $\top_n = (0^2{\to}0)^n{\to}0{\to}0$.

Chapter 3

Tools

3.1. Typed lambda Models

In this chapter we work with $\lambda^o_\to$, having one atomic type o, rather than with $\lambda_\to$, having infinitely many atomic types. The reader is encouraged to investigate which results do generalize. We will write $\mathbb{T} = \mathbb{T}(\lambda^o_\to)$.

In this section we will treat models, consistent sets of equations and their term models. We develop the theory for $\lambda^o_\to$, the reader being invited to see how much can be generalized to $\lambda_\to$.

3.1.1. Definition. Let $\mathcal{M} = \{\mathcal{M}(A)\}_{A \in \mathbb{T}^o}$ be a family of non-empty sets indexed by types.

(i) $\mathcal{M}$ is called a *typed structure* for $\lambda^o_\to$ if

$$\mathcal{M}(A{\to}B) \subseteq \mathcal{M}(B)^{\mathcal{M}(A)}.$$

(ii) Let $\mathcal{M}$ be provided with application operators

$$(\mathcal{M}, \cdot) = (\{\mathcal{M}(A)\}_{A \in \mathbb{T}^o}, \{\cdot_{A,B}\}_{A,B \in \mathbb{T}^o})$$

with $\cdot_{A,B} : \mathcal{M}(A{\to}B) \times \mathcal{M}(A) \to \mathcal{M}(B)$. Such a structure we call an *applicative typed structure* if the following property of *extensionality* holds:

$$\forall f, g \in \mathcal{M}(A{\to}B)\, [[\forall a \in \mathcal{M}(A)\ f \cdot_{A,B} a = g \cdot_{A,B} a] \ \Rightarrow\ f = g].$$

(iii) $\mathcal{M}$ is called *trivial* if $\mathcal{M}(o)$ is a singleton.

3.1.2. Notation. For applicative typed structures we use the infix notation $f \cdot_{A,B} x$ for $\cdot_{A,B}(f, x)$. Often we will be even more brief, extensionality becoming

$$\forall f, g \in \mathcal{M}(A{\to}B)\, [[\forall a \in \mathcal{M}_A\ fa = ga] \ \Rightarrow\ f = g]$$

or simply (if A, B are implicitly known)

$$\forall f, g \in \mathcal{M}\, [[\forall a\ fa = ga] \ \Rightarrow\ f = g].$$

3.1.3. Proposition. *The notions of typed structure and applicative typed structure are equivalent.*

PROOF. In a typed structure $\mathcal{M}$ define $f \cdot a = f(a)$; extensionality is obious. Conversely, let $\langle \mathcal{M}, \cdot \rangle$ be an applicative typed structure. Define the typed structure $\mathcal{M}'$ and $\Phi_A : \mathcal{M}(A) \to \mathcal{M}'(A)$ as follows.

$$\begin{aligned} \mathcal{M}'(o) &= \mathcal{M}(o); \\ \Phi_o(a) &= a; \\ \mathcal{M}'(A \to B) &= \{\Phi_{A\to B}(f) \in \mathcal{M}'(B)^{\mathcal{M}'(A)} \mid f \in \mathcal{M}(A \to B)\}; \\ \Phi_{A\to B}(f)(\Phi_A(a)) &= \Phi_B(f \cdot a). \end{aligned}$$

By definition Φ is surjective. By extensionality of the applicative typed structure it is also injective. Hence $\Phi_{A\to B}(f)$ is well defined. Clearly one has $\mathcal{M}'(A \to B) \subseteq \mathcal{M}'(B)^{\mathcal{M}'(A)}$. ■

3.1.4. DEFINITION. Let $\mathcal{M}, \mathcal{N}$ be two applicative typed structures. F is a *morphism* iff $F = \{F_A\}_{A \in \mathbb{T}^o}$ such that for each $A, B \in \mathbb{T}^o$ one has

$F_A : \mathcal{M}(A) \to \mathcal{N}(A)$;
$F_{A\to B}(f) \cdot F_A(a) = F_B(f \cdot a)$.

From now on we will not make a distinction between the notions of type and applicative typed structure.

3.1.5. PROPOSITION. *Let $\mathcal{M}$ be a typed structure. Then*

$$\mathcal{M} \text{ is trivial} \iff \forall A \in \mathbb{T}^o . \mathcal{M}(A) \text{ is a singleton.}$$

PROOF. ($\Leftarrow$) By definition. ($\Rightarrow$) We will show this for $A = 1 = o \to o$. If $\mathcal{M}(o)$ is a singleton, then for all $f, g \in \mathcal{M}(1)$ one has $\forall x : \mathcal{M}(o).(fx) = (gx)$, hence $f = g$. Therefore $\mathcal{M}(o)$ is a singleton. ■

3.1.6. EXAMPLE. (i) The full typed structure $\mathcal{M}_X = \{X(A)\}_{A \in \mathbb{T}^o}$ over a non-empty set X, see definition 2.4.18, is an applicative typed structure.

(ii) Let $(X, \leq)$ be a non-empty partially ordered set. Let $D(o) = X$ and $D(A \to B)$ consist of the monotone elements of $D(B)^{D(A)}$, where we order this set pointwise: for $f, g \in D(A \to B)$ define

$$f \leq g \iff \forall a \in D(A)\, fa \leq ga.$$

The elements of the applicative typed structure $D_X = \{D(A)\}_{A \in \mathbb{T}^o}$ are called the *hereditarily monotonic functions*, see the chapter by Howard in Troelstra [1973].

(iii) Let $\mathcal{M}$ be an applicative typed structure. A *layered non-empty subfamily* of $\mathcal{M}$ is a family $\Delta = \{\Delta(A)\}_{A \in \mathbb{T}^o}$ of sets, such that the following hold

$$\forall A \in \mathbb{T}^o . \emptyset \neq \Delta(A) \subseteq \mathcal{M}(A)$$

Δ is called *closed under application* if

$$f \in \Delta(A \to B), g \in \Delta(A) \;\Rightarrow\; fg \in \Delta(B).$$

Δ is called *extensional* if

$$\forall A,B\in\mathbb{T}^o\forall f,g\in\Delta(A{\to}B).[[\forall a\in\Delta(A).fa=ga]\ \Rightarrow\ f=g].$$

If Δ satisfies all these conditions, then $\mathcal{M}{\restriction}\Delta=(\Delta,\cdot{\restriction}\Delta)$ is an applicative typed structure.

3.1.7. Definition. (i) Let $\mathcal{M}$ be an applicative type structure. Then a (*partial*) *valuation in* $\mathcal{M}$ is a family of (partial) maps $\rho=\{\rho_A\}_{A\in\mathbb{T}^o}$ such that $\rho_A:\mathtt{Var}(A){\to}\mathcal{M}(A)$.

(ii) Given an applicative typed structure $\mathcal{M}$ and a partial valuation ρ in $\mathcal{M}$ one can define for $M\in\Lambda_o(A)$ the partial semantics $[\![M]\!]^{\mathcal{M}}_\rho\in\mathcal{M}(A)$ as follows

$$\begin{array}{rcl}[\![x^A]\!]^{\mathcal{M}}_\rho &=& \rho_A(x);\\ [\![PQ]\!]^{\mathcal{M}}_\rho &=& [\![P]\!]^{\mathcal{M}}_\rho[\![Q]\!]^{\mathcal{M}}_\rho;\\ [\![\lambda x.P]\!]^{\mathcal{M}}_\rho &=& \lambda\!\!\lambda d.[\![P]\!]^{\mathcal{M}}_{\rho[x:=d]}.\end{array}$$

We often write $[\![M]\!]_\rho$ for $[\![M]\!]^{\mathcal{M}}_\rho$. The expression $[\![M]\!]_\rho$ may not always be defined, even if ρ is total. The problem arises with $[\![\lambda x.P]\!]_\rho$. Although the function $\lambda\!\!\lambda d\in A.[\![P]\!]_{\rho[x:=d]}\in\mathcal{M}(B)^{\mathcal{M}(A)}$ is uniquely determined by $[\![\lambda x.P]\!]_\rho d=[\![P]\!]_{\rho[x:=d]}$, it may fail to be an element of $\mathcal{M}(A{\to}B)$ which is only a subset of $\mathcal{M}(B)^{\mathcal{M}(A)}$. We write $[\![M]\!]_\rho{\downarrow}$ if $[\![M]\!]_\rho\in\mathcal{M}(A)$ is well defined. We will write $[\![M]\!]_\rho=[\![M]\!]^{\mathcal{M}}_\rho$ if there is little danger of confusion.

3.1.8. Definition. (i) A typed structure $\mathcal{M}$ is called a $\lambda^o_{\to}$*-model* or simply a *(type) model* if for every partial valuation $\rho=\{\rho_A\}_A$ and every $A\in\mathbb{T}^o$ and $M\in\Lambda_o(A)$ such that $\mathrm{FV}(M)\subseteq\mathrm{dom}(\rho)$ one has $[\![M]\!]_\rho{\downarrow}$.

(ii) Let $\mathcal{M}$ be a model and ρ a partial evaluation. Then we define

$$\mathcal{M},\rho\models M=N\iff[\![M]\!]^{\mathcal{M}}_\rho=[\![N]\!]^{\mathcal{M}}_\rho.$$

Here and later we assume implicitly that M and N have the same type.

(iii) Let $\mathcal{M}$ be a model. Then we say that $\mathcal{M}$ satisfies $M=N$, notation

$$\mathcal{M}\models M=N$$

iff for all partial ρ with $\mathrm{FV}(MN)\subseteq\mathrm{dom}(\rho)$ one has $\mathcal{M},\rho\models M=N$.

(iv) Let $\mathcal{M}$ be a model. The *theory of* $\mathcal{M}$ is

$$\mathrm{Th}(\mathcal{M})=\{M=N\mid M,N\in\Lambda^\emptyset_o\ \&\ \mathcal{M}\models M=N\}.$$

3.1.9. Notation. Let E_1,E_2 be partial (i.e. possibly undefined) expressions.

(i) Write $E_1\rightrightarrows E_2$ iff $E_1{\downarrow}\ \Rightarrow\ [E_2{\downarrow}\ \&\ E_1=E_2]$.

(ii) Write $E_1\simeq E_2$ iff $E_1\rightrightarrows E_2\ \&\ E_2\rightrightarrows E_1$.

3.1.10. Lemma. (i) *Let $M\in\Lambda_o(A)$ and N be a subterm of M. Then*

$$[\![M]\!]_\rho{\downarrow}\ \Rightarrow\ [\![N]\!]_\rho{\downarrow}.$$

(ii) *Let $M \in \Lambda_o(A)$. Then*

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

(iii) *Let $M \in \Lambda_o(A)$ and ρ_1, ρ_2 be such that $\rho_1 \restriction \mathrm{FV}(M) = \rho_2 \restriction \mathrm{FV}(M)$. Then*

$$[\![M]\!]_{\rho_1} \simeq [\![M]\!]_{\rho_2}.$$

PROOF. (i) By induction on the structure of M.
(ii) Similarly.
(iii) By (ii). ■

3.1.11. LEMMA. *Let $\mathcal{M}$ be an applicative structure. Then*
(i) *For $M \in \Lambda_o(A)$, $x, N \in \Lambda_o(B)$ one has*

$$[\![M[x{:=}N]]\!]^{\mathcal{M}}_\rho \simeq [\![M]\!]^{\mathcal{M}}_{\rho[x:=[\![N]\!]^{\mathcal{M}}_\rho]}.$$

(ii) *For $M, N \in \Lambda_o(A)$ one has*

$$M \twoheadrightarrow_{\boldsymbol{\beta\eta}} N \;\Rightarrow\; [\![M]\!]^{\mathcal{M}}_\rho \beth [\![N]\!]^{\mathcal{M}}_\rho.$$

PROOF. (i) By induction on the structure of M. Write $M^\bullet \equiv M[x{:=}N]$. We only treat the case $M \equiv \lambda y.P$. By the variable convention we may assume that $y \notin \mathrm{FV}(N)$. We have

$$\begin{array}{lcll}
[\![(\lambda y.P)^\bullet]\!]_\rho & \simeq & [\![\lambda y.P^\bullet]\!]_\rho & \\
& \simeq & \lambda\!\!\lambda d.[\![P^\bullet]\!]_{\rho[y:=d]} & \\
& \simeq & \lambda\!\!\lambda d.[\![P]\!]_{\rho[y:=d][x:=[\![N]\!]_{\rho[y:=d]}]}, & \text{by the IH,} \\
& \simeq & \lambda\!\!\lambda d.[\![P]\!]_{\rho[y:=d][x:=[\![N]\!]_\rho]}, & \text{by Lemma 3.1.10,} \\
& \simeq & \lambda\!\!\lambda d.[\![P]\!]_{\rho[x:=[\![N]\!]_\rho][y:=d]} & \\
& \simeq & [\![\lambda y.P]\!]_{\rho[x:=[\![N]\!]_\rho]}. &
\end{array}$$

(ii) By induction on the generation of $M \twoheadrightarrow_{\boldsymbol{\beta\eta}} N$.
Case $M \equiv (\lambda x.P)Q$ and $N \equiv P[x{:=}Q]$. Then

$$\begin{array}{lcll}
[\![(\lambda x.P)Q]\!]_\rho & \beth & (\lambda\!\!\lambda d.[\![P]\!]_{\rho[x:=d]})([\![Q]\!]_\rho) & \\
& \beth & [\![P]\!]_{\rho[x:=[\![Q]\!]_\rho]} & \\
& \simeq & [\![P[x{:=}Q]]\!]_\rho, & \text{by (i).}
\end{array}$$

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

$$\begin{array}{lcl}
[\![\lambda x.Nx]\!]_\rho & \beth & \lambda\!\!\lambda d.[\![N]\!]_\rho(d) \\
& \simeq & [\![N]\!]_\rho.
\end{array}$$

Cases $M \twoheadrightarrow_{\boldsymbol{\beta\eta}} N$ is $PZ \twoheadrightarrow_{\boldsymbol{\beta\eta}} QZ$, $ZP \twoheadrightarrow_{\boldsymbol{\beta\eta}} ZQ$ or $\lambda x.P \twoheadrightarrow_{\boldsymbol{\beta\eta}} \lambda x.Q$, and follows directly from $P \twoheadrightarrow_{\boldsymbol{\beta\eta}} Q$, then the result follows from the IH.

The cases where $M \twoheadrightarrow_{\boldsymbol{\beta\eta}} N$ follows via reflexivity or transitivity are easy to treat. ■

3.1.12. Definition. Let $\mathcal{M},\mathcal{N}$ be typed lambda models and let $A\in\mathbb{T}^o$.

(i) $\mathcal{M}$ and $\mathcal{N}$ are *elementary equivalent at* A, notation $\mathcal{M}\equiv_A\mathcal{N}$, iff

$$\forall M,N\in\Lambda^\emptyset_o(A).[\mathcal{M}\models M=N \iff \mathcal{N}\models M=N].$$

(ii) $\mathcal{M}$ and $\mathcal{N}$ are *elementary equivalent*, notation $\mathcal{M}\equiv\mathcal{N}$, iff

$$\forall A\in\mathbb{T}^o.\mathcal{M}\equiv_A\mathcal{N}.$$

3.1.13. Proposition. *Let $\mathcal{M}$ be typed lambda model. Then*

$$\mathcal{M} \textit{ is non-trivial} \iff \forall A\in\mathbb{T}^o.\mathcal{M}(A) \textit{ is not a singleton.}$$

Proof. ($\Leftarrow$) By definition. ($\Rightarrow$) We will show this for $A=1=o\to o$. Let c_1,c_2 be distinct elements of $\mathcal{M}(o)$. Consider $M\equiv\lambda x^o.y^o\in\Lambda^\emptyset_o(1)$. Let ρ_i be the partial valuation with $\rho_i(y^o)=c_i$. Then $[\![M]\!]_{\rho_i}\downarrow$ and $[\![M]\!]_{\rho_1}c_1=c_1,[\![M]\!]_{\rho_2}c_1=c_2$. Therefore $[\![M]\!]_{\rho_1},[\![M]\!]_{\rho_2}$ are different elements of $\mathcal{M}(1)$. ■

3.1.14. Proposition. *Let $\mathcal{M},\mathcal{N}$ be models and $F{:}\mathcal{M}\to\mathcal{N}$ a surjective morphism. Then the following hold.*

(i) $F([\![M]\!]^{\mathcal{M}}_\rho)=[\![M]\!]^{\mathcal{N}}_{F\circ\rho}$, *for all* $M\in\Lambda_o(A)$.

(ii) $F([\![M]\!]^{\mathcal{M}})=[\![M]\!]^{\mathcal{N}}$, *for all* $M\in\Lambda^\emptyset_o(A)$.

Proof. (i) By induction on the structure of M.

(ii) By (i). ■

3.1.15. Proposition. *Let $\mathcal{M}$ be a typed lambda model.*

(i) $\mathcal{M}\models(\lambda x.M)N=M[x:=N]$.

(ii) $\mathcal{M}\models\lambda x.Mx=M$, *if* $x\notin\mathrm{FV}(\mathcal{M})$.

Proof. (i)
$$\begin{aligned}[\![(\lambda x.M)N]\!]_\rho &= [\![\lambda x.M]\!]_\rho[\![N]\!]_\rho\\ &= [\![M]\!]_{\rho[x:=[\![N]\!]_\rho]}, \quad \text{by Lemma 3.1.11,}\\ &= [\![M[x:=N]]\!]_\rho.\end{aligned}$$

(ii)
$$\begin{aligned}[\![\lambda x.Mx]\!]_\rho d &= [\![Mx]\!]_{\rho[x:=d]}\\ &= [\![M]\!]_{\rho[x:=d]}d\\ &= [\![M]\!]_\rho d, \quad \text{as } x\notin\mathrm{FV}(M).\end{aligned}$$

Therefore by extensionality $[\![\lambda x.Mx]\!]_\rho=[\![M]\!]_\rho$. ■

3.1.16. Lemma. *Let $\mathcal{M}$ be a typed lambda model. Then*

$$\mathcal{M}\models M=N \iff \mathcal{M}\models\lambda x.M=\lambda x.N.$$

Proof.
$$\begin{aligned}\mathcal{M}\models M=N &\iff \forall\rho. && [\![M]\!]_\rho &=& [\![N]\!]_\rho\\ &\iff \forall\rho,d. && [\![M]\!]_{\rho[x:=d]} &=& [\![N]\!]_{\rho[x:=d]}\\ &\iff \forall\rho,d. && [\![\lambda x.M]\!]_\rho d &=& [\![\lambda x.N]\!]_\rho d\\ &\iff \forall\rho. && [\![\lambda x.M]\!]_\rho &=& [\![\lambda x.N]\!]_\rho\\ &\iff && \mathcal{M}\models\lambda x.M &=& \lambda x.N.\end{aligned}$$
■

3.1.17. Proposition. (i) *For every non-empty set X the typed structure $\mathcal{M}_X$ is a $\lambda^o_\to$-model.*

(ii) *Let X be a poset. Then D_X is a $\lambda^o_\to$-model.*

(iii) *Let $\mathcal{M}$ be an applicative typed structure. Assume that $[\![\mathsf{K}_{A,B}]\!]^{\mathcal{M}}\downarrow$ and $[\![\mathsf{S}_{A,B,C}]\!]^{\mathcal{M}}\downarrow$. Then $\mathcal{M}$ is a $\lambda^o_\to$-model.*

(iv) *Let Δ be a layered non-empty subfamily of an applicative structure $\mathcal{M}$ that is extensional and closed under application. Suppose $[\![\mathsf{K}_{A,B}]\!], [\![\mathsf{S}_{A,B,C}]\!]$ are defined and in Δ. Then $\mathcal{M}{\restriction}\Delta$, see Example 3.1.6(iii), is a $\lambda^o_\to$-model.*

Proof. (i) Since $\mathcal{M}_X$ is the full typed structure, $[\![M]\!]_\rho$ always exists.

(ii) By induction on M one can show that $\lambda\!\!\lambda d.[\![M]\!]_{\rho(x:=d)}$ is monotonic. It then follows by induction on M that $[\![M]\!]_\rho \in D_X$.

(iii) For every λ-term M there exists a typed applicative expression P consisting only of Ks and Ss such that $P \twoheadrightarrow_{\beta\eta} M$. Now apply Lemma 3.1.11.

(iv) By (iii). ■

Operations on models

Now we will introduce two operations on models: $\mathcal{M}, \mathcal{N} \mapsto \mathcal{M}\times\mathcal{N}$, the cartesian product, and $\mathcal{M} \mapsto \mathcal{M}^*$, the polynomial model. The relationship between $\mathcal{M}$ and $\mathcal{M}^*$ is similar to that of a ring R and its ring of multivariate polynomials $R[\vec{x}]$.

Cartesian products

3.1.18. Definition. If $\mathcal{M}, \mathcal{N}$ are applicative structures, then $\mathcal{M} \times \mathcal{N}$ is the structure defined by

$$\begin{aligned}(\mathcal{M} \times \mathcal{N})(A) &= \mathcal{M}(A) \times \mathcal{N}(A)\\ (M_1, M_2)(N_1, N_2) &= (M_1N_1, M_2N_2).\end{aligned}$$

3.1.19. Proposition. *Let $\mathcal{M}, \mathcal{N}$ be models. For a partial valuation ρ in $\mathcal{M}\times\mathcal{N}$ write $\rho(x) = (\rho_1(x), \rho_2(x))$.*

(i) $[\![M]\!]^{\mathcal{M}\times\mathcal{N}}_\rho = ([\![M]\!]^{\mathcal{M}}_{\rho_1}, [\![M]\!]^{\mathcal{N}}_{\rho_2})$.

(ii) *$\mathcal{M} \times \mathcal{N}$ is a model.*

(iii) $\mathrm{Th}(\mathcal{M} \times \mathcal{N}) = \mathrm{Th}(\mathcal{M}) \cap \mathrm{Th}(\mathcal{N})$.

Proof. (i) By induction on M.

(ii) By (i).

(iii)
$$\begin{aligned}\mathcal{M} \times \mathcal{N}, \rho \models M = N &\iff [\![M]\!]_\rho = [\![N]\!]_\rho\\ &\iff ([\![M]\!]^{\mathcal{M}}_{\rho_1}, [\![M]\!]^{\mathcal{N}}_{\rho_2}) = ([\![N]\!]^{\mathcal{M}}_{\rho_1}, [\![N]\!]^{\mathcal{N}}_{\rho_2})\\ &\iff [\![M]\!]^{\mathcal{M}}_{\rho_1} = [\![N]\!]^{\mathcal{M}}_{\rho_1} \;\&\; [\![M]\!]^{\mathcal{M}}_{\rho_2} = [\![N]\!]^{\mathcal{M}}_{\rho_2}\\ &\iff \mathcal{M}, \rho \models M = N \;\&\; \mathcal{N}, \rho \models M = N,\end{aligned}$$

by (i). Hence for closed terms M, N

$$\mathcal{M} \times \mathcal{N} \models M = N \iff \mathcal{M} \models M = N \;\&\; \mathcal{N} \models M = N. \blacksquare$$

Polynomial models

3.1.20. DEFINITION. (i) We introduce a new constant c_m for each $m \in \mathcal{M}$ and we let $\underline{\mathcal{M}}$ be the set of all typed applicative combinations of variables and constants c_m.

(ii) For each valuation $\rho : \mathtt{Var} \to \mathcal{M}$ define the interpretation $(\!(-)\!)_\rho : \underline{\mathcal{M}} \to \mathcal{M}$ by

$$\begin{aligned} (\!(x)\!)_\rho &= \rho(x); \\ (\!(c_m)\!)_\rho &= m; \\ (\!(PQ)\!)_\rho &= (\!(P)\!)_\rho (\!(Q)\!)_\rho. \end{aligned}$$

(iii) Write

$$P =_{\mathcal{M}} Q \iff \forall \rho\, (\!(P)\!)_\rho = (\!(Q)\!)_\rho,$$

where ρ ranges over valuations in $\mathcal{M}$.

3.1.21. LEMMA. (i) *$=_{\mathcal{M}}$ is an equivalence relation satisfying $c_d \cdot c_e =_{\mathcal{M}} c_{de}$.*

(ii) *For all $P, Q \in \underline{\mathcal{M}}$ one has*

$$P_1 =_{\mathcal{M}} P_2 \iff \forall Q_1, Q_2 \in \underline{\mathcal{M}}\, [Q_1 =_{\mathcal{M}} Q_2 \;\Rightarrow\; P_1 Q_1 =_{\mathcal{M}} P_2 Q_2].$$

PROOF. Easy. ■

3.1.22. DEFINITION. Let $\mathcal{M}$ be an applicative structure. The *polynomial structure* over $\mathcal{M}$ is $\mathcal{M}^* = (|\mathcal{M}^*|, \mathtt{app})$ defined by

$$\begin{aligned} &|\mathcal{M}^*| = \underline{\mathcal{M}}/{=_{\mathcal{M}}} = \{[P]_{=_{\mathcal{M}}} \mid P \in \underline{\mathcal{M}}\}, \\ &\mathtt{app}\ [P]_{=_{\mathcal{M}}} [Q]_{=_{\mathcal{M}}} = [PQ]_{=_{\mathcal{M}}}. \end{aligned}$$

By lemma 3.1.21(ii) this is well defined.

3.1.23. PROPOSITION. (i) *$\mathcal{M} \subseteq \mathcal{M}^*$ by the embedding morphism $d \mapsto c_d$.*

(ii) *$\mathcal{M}^* \cong \mathcal{M}^{**}$.*

PROOF. (i) Define $i(d^A) = [c_{d^A}]$. Then $i : \mathcal{M}(A) \to \mathcal{M}^*(A)$ and

$$\begin{aligned} i(d \cdot_{\mathcal{M}} e) &= [c_{d \cdot_{\mathcal{M}} e}] \\ &= [c_d \cdot_{\underline{\mathcal{M}}} c_e], && \text{by definition of } =_{\mathcal{M}}, \\ &= [c_d] \cdot_{\mathcal{M}^*} [c_e], && \text{by definition of } \mathtt{app} \\ &= i(d) \cdot_{\mathcal{M}^*} i(e). \end{aligned}$$

We will not always be so explicit about where application is taken.

(ii) Adding twice an infinite set of variables is the same as adding one infinite set, since $2.\aleph_0 = \aleph_0$. ■

Working with $\mathcal{M}^*$ it is often convenient to use as elements those of $\underline{\mathcal{M}}$ and reason about them modulo $=_{\mathcal{M}}$.

3.1.24. DEFINITION. Let $P \in \underline{\mathcal{M}}$ and let x be a variable. We say that

$$P \text{ does not depend on } x$$

if whenever ρ_1, ρ_2 satisfy $\rho_1(y) = \rho_2(y)$ for $y \not\equiv x$, we have $[\![P]\!]_{\rho_1} = [\![P]\!]_{\rho_2}$.

3.1.25. LEMMA. *If P does not depend on x, then $P =_{\mathcal{M}} P[x{:=}Q]$ for all $Q \in \underline{\mathcal{M}}$.*

PROOF. First show that $(\!(P[x := Q])\!)_\rho = [\![P]\!]_{\rho[x:=(\!(Q)\!)_\rho]}$, in analogy to Lemma 3.1.11(i). Now suppose P does not depend on x. Then

$$\begin{array}{lcll} (\!(P[x{:=}Q])\!)_\rho & = & [\![P]\!]_{\rho[x:=(\!(Q)\!)_\rho]} & \\ & = & (\!(P)\!)_\rho, & \text{as } P \text{ does not depend on } x. \ \blacksquare \end{array}$$

3.1.26. PROPOSITION. *Let $\mathcal{M}$ be an applicative structure. Then*

(i) *$\mathcal{M}$ is a model $\iff$ for each $P \in \mathcal{M}^*$ and variable x there exists an $F \in \mathcal{M}^*$ not depending on x such that $Fx = P$.*

(ii) *$\mathcal{M}$ is a model $\Rightarrow$ $\mathcal{M}^*$ is a model.*

PROOF. (i) It suffices to show that

$\mathcal{M}$ is a model $\Leftrightarrow$ for each $P \in \underline{\mathcal{M}}$ and variable x there exists an $F \in \underline{\mathcal{M}}$ not depending on x such that $Fx =_{\mathcal{M}} P$.

($\Rightarrow$) Let $\mathcal{M}$ be a model and let P be given. We treat an illustrative example, e.g. $P \equiv c_f x^o y^o$, with $f \in \mathcal{M}(1_2)$. We take $F \equiv c_{[\![\lambda y z_f . z_f x y]\!]} y c_f$. Then

$$(\!(Fx)\!)_\rho = [\![\lambda y z_f . z_f x y]\!]\rho(y) f\rho(x) = f\rho(x)\rho(y) = (\!(c_f x y)\!)_\rho,$$

hence indeed $Fx =_{\mathcal{M}} c_f x y$. In general for each constant c_d in P we take a variable z_d and define $F \equiv [\![\lambda \vec{y} \vec{z_d} x . P]\!] \vec{y} \vec{c_f}$.

($\Leftarrow$) We show by induction on M that $\forall M \in \Lambda_o \exists P_M \in \underline{\mathcal{M}} \forall \rho . [\![M]\!]_\rho = (\!(P_M)\!)_\rho$. For M being a variable, constant or application this is trivial. For $M = \lambda x.N$, we know by the induction hypothesis that $[\![N]\!]_\rho = (\!(P_N)\!)_\rho$ for all ρ. By assumption there is an F not depending on x such that $Fx =_{\mathcal{M}} P_N$. Then

$$(\!(F)\!)_\rho d = (\!(Fx)\!)_{\rho[x:=d]} = (\!(P_N)\!)_{\rho[x:=d]} =_{\text{IH}} [\![N]\!]_{\rho[x:=d]}.$$

Hence $[\![\lambda x.N]\!]_\rho \downarrow = (\!(F)\!)_\rho$. It follows that $\mathcal{M}$ is a model.

(ii) By (i) $\mathcal{M}^*$ is a model if a certain property holds for $\mathcal{M}^{**}$. But $\mathcal{M}^{**} \cong \mathcal{M}^*$ and the property does hold here, since $\mathcal{M}$ is a model. [To make matters concrete, one has to show for example that for all $M \in \mathcal{M}^{**}$ there is an N not depending on y such that $Ny =_{\mathcal{M}^*} M$. Writing $M \equiv M[x_1, x_2][y]$ one can obtain N by rewriting the y in M obtaining $M' \equiv M[x_1, x_2][x] \in \mathcal{M}^*$ and using the fact that $\mathcal{M}$ is a model: $M' = Nx$, so $Ny = M$]. $\blacksquare$

3.1.27. PROPOSITION. *If $\mathcal{M}$ is a model, then* $\mathrm{Th}(\mathcal{M}^*) = \mathrm{Th}(\mathcal{M})$.

PROOF. Do exercise 3.6.2. $\blacksquare$

3.1.28. REMARK. In general for typed structures $\mathcal{M}^* \times \mathcal{N}^* \ncong (\mathcal{M} \times \mathcal{N})^*$, but the isomorphism holds in case $\mathcal{M}, \mathcal{N}$ are typed lambda models.

3.2. Lambda Theories and Term Models

3.2.1. Definition. (i) A *constant* (of type A) is a variable (of the same type) that we promise not to bind by a λ. Rather than $x, y, z, \ldots$ we write constants as $c, d, e, \ldots$. The letters $\mathcal{C}, \mathcal{D}, \ldots$ range over sets of constants (of varying types).

(ii) Let $\mathcal{D} = \{c_1^{A_1}, \ldots, c_n^{A_n}\}$ be a set of constants with types in $\mathbb{T}^o$. Write $\Lambda_o[\mathcal{D}](A)$ for the set of open terms of type A, possibly containing the constants in $\mathcal{D}$. Moreover $\Lambda_o[\mathcal{D}] = \cup_{A \in \mathbb{T}} \Lambda_o[\mathcal{D}](A)$.

(iii) Similarly $\Lambda_o^\emptyset[\mathcal{D}](A)$ and $\Lambda_o^\emptyset[\mathcal{D}]$ consist of closed terms possibly containing the constants in $\mathcal{D}$.

(iv) An equation over D is of the form $M = N$ with $M, N \in \Lambda_o^\emptyset[D]$ of the same type.

In this subsection we will consider sets of equations over D. When writing $M = N$, we implicitly assume that M, N have the same type.

3.2.2. Definition. Let $\mathcal{E}$ be a set of equations over the constants. $\mathcal{D}$.

(i) $P = Q$ is derivable from $\mathcal{E}$, notation $\mathcal{E} \vdash P = Q$ if $P = Q$ can be proved in the equational theory axiomatized as follows

$(\lambda x.M)N = M[x := N]$	(β)
$(\lambda x.Mx = M),\ x \notin \mathrm{FV}(M)$	(η)
$\{M = N \mid (M = N) \in \mathcal{E}\}$	
$M = M$	(reflexivity)
$\dfrac{M = N}{N = M}$	(symmetry)
$\dfrac{M = N \quad N = L}{M = L}$	(transitivity)
$\dfrac{M = N}{MZ = NZ}$	(R-congruence)
$\dfrac{M = N}{ZM = ZN}$	(L-congruence)
$\dfrac{M = N}{\lambda x.M = \lambda x.N}$	(ξ-rule)

We write $M =_{\mathcal{E}} N$ for $\mathcal{E} \vdash M = N$.

(ii) $\mathcal{E}$ is *consistent*, if not all equations are derivable from it.

(iii) $\mathcal{E}$ is a *typed lambda theory* iff $\mathcal{E}$ is consistent and closed under derivability.

3.2.3. Notation. (i) $\mathcal{E}^+ = \{M = N \mid \mathcal{E} \vdash M = N\}$.

(ii) For $A \in \mathbb{T}^o$ write $\mathcal{E}(A) = \{M = N \mid (M = N) \in \mathcal{E}\ \&\ M, N \text{ are of type } A\}$.

(iii) $\mathcal{E}_{\beta\boldsymbol{\eta}} = \emptyset^+$.

3.2.4. Proposition. *If $Mx =_{\mathcal{E}} Nx$, with $x \notin \mathrm{FV}(M) \cup \mathrm{FV}(N)$, then $M =_{\mathcal{E}} N$.*

PROOF. Use (ξ) and (η). ■

3.2.5. DEFINITION. Let $\mathcal{M}$ be a type model and $\mathcal{E}$ a set of equations.
(i) We say that $\mathcal{M}$ *satisfies* (or is a *model* of) $\mathcal{E}$, notation $\mathcal{M} \models \mathcal{E}$, iff

$$\forall (M{=}N) \in \mathcal{E}. \mathcal{M} \models M = N.$$

(ii) We say that $\mathcal{E}$ satisfies $M = N$, notation $\mathcal{E} \models M = N$, iff

$$\forall \mathcal{M}. [\mathcal{M} \models \mathcal{E} \ \Rightarrow \ \mathcal{M} \models M = N].$$

3.2.6. PROPOSITION. *(Soundness)* $\mathcal{E} \vdash M = N \ \Rightarrow \ \mathcal{E} \models M = N$.

PROOF. By induction on the derivation of $\mathcal{E} \vdash M = N$. Assume that $\mathcal{M} \models \mathcal{E}$ for a model $\mathcal{M}$ towards $\mathcal{M} \models M = N$. If $M = N \in \mathcal{E}$, then the conclusion follows from the assumption. The cases that $M = N$ falls under the axioms β or η follow from Proposition 3.1.15. The rules reflexivity, symmetry, transitivity and L,R-congruence are trivial to treat. The case falling under the ξ-rule follows from Lemma 3.1.16. ■

From non-trivial models one can obtain typed lambda theories.

3.2.7. PROPOSITION. *Let $\mathcal{M}$ be a non-trivial model.*
(i) $\mathcal{M} \models \mathcal{E} \ \Rightarrow \ \mathcal{E}$ *is consistent.*
(ii) $\mathrm{Th}(\mathcal{M})$ *is a lambda theory.*

PROOF. (i) Suppose $\mathcal{E} \vdash \lambda xy.x = \lambda xy.y$. Then $\mathcal{M} \models \lambda xy.x = \lambda xy.y$. It follows that $d = (\lambda xy.x)de = (\lambda xy.y)de$ for arbitrary d, e. Hence $\mathcal{M}$ is trivial.
(ii) Clearly $\mathcal{M} \models \mathrm{Th}(\mathcal{M})$. Hence by (i) $\mathrm{Th}(\mathcal{M})$ is consistent. If $\mathrm{Th}(\mathcal{M}) \vdash M = N$, then by soundness $\mathcal{M} \models M = N$, and therefore $(M = N) \in \mathrm{Th}(\mathcal{M})$. ■

The full type model over a finite set yields an interesting λ-theory.

3.2.8. EXERCISE. Let $\mathcal{M}_n = \mathcal{M}_{\{1,\ldots,n\}}$. Write $c_i = \lambda f x. f^i x$ for $i \in \mathbb{N}$, the Church numerals of type $1{\to}o{\to}o$.
(i) Show that for $i, j \in \mathbb{N}$ one has

$$\mathcal{M}_n \models c_i = c_j \iff i = j \ \vee \ [i, j \geq n{-}1 \ \& \ \forall k_{1 \leq k \leq n}. i \equiv j (\mathrm{mod}\ k)].$$

[Hint. For $a \in \mathcal{M}_n(o), f \in \mathcal{M}_n(1)$ define the *trace of a under f* as $\{f^i(a) \mid i \in \mathbb{N}\}$, directed by $G_f = \{(a, b) \mid f(a) = b\}$, which by the pigeonhole principle is 'lasso-shaped'. Consider the traces of 1 under the functions f_n, g_m with $1 \leq m \leq n$

$$\begin{array}{rcll} f_n(k) = & k+1, & \text{if } k < n, & \text{and} \\ = & n, & \text{if } k = n, & \end{array} \qquad \begin{array}{rcll} g_m(k) = & k+1, & \text{if } k < m, & \text{Conclude that} \\ = & 1, & \text{if } k = m, & \\ = & k, & \text{else.} & \end{array}$$

e.g. $\mathcal{M}_5 \models c_4 = c_{64}$, $\mathcal{M}_6 \not\models c_4 = c_{64}$ and $\mathcal{M}_6 \models c_5 = c_{65}$.
(ii) Conclude that $\mathcal{M}_n \equiv_{1 \to o \to o} \mathcal{M}_m \iff n = m$.
(iii) Show that $\bigcap_n \mathrm{Th}(\mathcal{M}_n)(1) = \mathcal{E}_{\beta\eta}(1)$.
It will be shown in the section 3.4 that $\bigcap_n \mathrm{Th}(\mathcal{M}_n) = \mathcal{E}_{\beta\eta}$ and by contrast $\mathrm{Th}(\mathcal{M}_{\mathbb{N}}) = \mathcal{E}_{\beta\boldsymbol{\eta}}$.

Term Models

3.2.9. DEFINITION. Let D be a set of constants and let $\mathcal{E}$ be a set of closed equations between closed terms with constants from D. Define the applicative structure $\mathcal{M}_\mathcal{E}$ by

$$\mathcal{M}_\mathcal{E}(A) = \{[M]_\mathcal{E} \mid M \in \Lambda_o[D](A)\},$$

where $[M]_\mathcal{E}$ is the equivalence class modulo the congruence relation $=_\mathcal{E}$, with

$$[M]_\mathcal{E}[N]_\mathcal{E} = [MN]_\mathcal{E}$$

as application operator. This is well-defined, because $=_\mathcal{E}$ is a congruence.

3.2.10. PROPOSITION. (i) *$\mathcal{M}_\mathcal{E}$ is an applicative type structure.*

(ii) *The semantic interpretation of M in $\mathcal{M}_\mathcal{E}$ is determined by*

$$[\![M]\!]_\rho = [M[\vec{x}{:=}\vec{N}]]_\mathcal{E},$$

where the $\vec{N}$ are determined by $\rho(x_i) = [N_i]_\mathcal{E}$.

(iii) *$\mathcal{M}_\mathcal{E}$ is a type model, called the* open term model.

PROOF. (i) We need to verify extensionality.

$$\begin{array}{lcll} \forall d \in \mathcal{M}_\mathcal{E}.[M]d = [N]d & \Rightarrow & [M][x] = [N][x], & \text{for a fresh } x, \\ & \Rightarrow & [Mx] = [Nx] & \\ & \Rightarrow & Mx =_\mathcal{E} Nx & \\ & \Rightarrow & M =_\mathcal{E} N & \\ & \Rightarrow & [M] = [N]. & \end{array}$$

(ii) We show that $[\![M]\!]_\rho$ satisfies the conditions in definition 17(ii).

$$\begin{array}{rcll} [\![x]\!]_\rho & = & [x[x{:=}N]]_\mathcal{E}, & \text{with } \rho(x) = [N]_\mathcal{E}, \\ & = & [N]_\mathcal{E} & \\ & = & \rho(x); & \\ [\![PQ]\!]_\rho & = & [(PQ)[\vec{x}{:=}\vec{N}]]_\mathcal{E} & \\ & = & [P[\vec{x}{:=}\vec{N}]Q[\vec{x}{:=}\vec{N}]]_\mathcal{E} & \\ & = & [P[\vec{x}{:=}\vec{N}]]_\mathcal{E}[[Q[\vec{x}{:=}\vec{N}]]_\mathcal{E} & \\ & = & [\![P]\!]_\rho[\![Q]\!]_\rho; & \\ [\![\lambda y.P]\!]_\rho[Q]_\mathcal{E} & = & [(\lambda y.P)[\vec{x}{:=}\vec{N}]]_\mathcal{E}[Q]_\mathcal{E} & \\ & = & [\lambda y.P[\vec{x}{:=}\vec{N}]]_\mathcal{E}[Q]_\mathcal{E} & \\ & = & [P[\vec{x}{:=}\vec{N}][y{:=}Q]]_\mathcal{E} & \\ & = & [P[\vec{x}, y{:=}\vec{N}, Q]]_\mathcal{E}, & \text{because } y \notin \mathrm{FV}(\vec{N}) \text{ by the} \\ & & & \text{variable convention,} \\ & = & [\![P]\!]_{\rho(y:=[Q]_\mathcal{E})} & \\ & = & [\![P]\!]_{\rho(y:=[Q]_\mathcal{E})}. & \end{array}$$

(iii) As $[\![M]\!]_\rho$ is always defined by (ii). ■

3.2.11. Corollary. (i) $\mathcal{M}_\mathcal{E} \models M = N \iff M =_\mathcal{E} N$.
(ii) $\mathcal{M}_\mathcal{E} \models \mathcal{E}$.

Proof. (i) ($\Rightarrow$) Suppose $\mathcal{M}_\mathcal{E} \models M = N$. Then $[\![M]\!]_\rho = [\![N]\!]_\rho$ for all ρ. Choosing $\rho(x) = [x]_\mathcal{E}$ one obtains $[\![M]\!]_\rho = [M[\vec{x} := \vec{x}]]_\mathcal{E} = [M]_\mathcal{E}$, and similarly for N, hence $[M]_\mathcal{E} = [N]_\mathcal{E}$ and therefore $M =_\mathcal{E} N$.

$$\begin{array}{lll} (\Leftarrow)\ M =_\mathcal{E} N & \Rightarrow & M[\vec{x} := \vec{P}] =_\mathcal{E} N[\vec{x} := \vec{P}] \\ & \Rightarrow & [M[\vec{x} := \vec{P}]]_\mathcal{E} = [N[\vec{x} := \vec{P}]]_\mathcal{E} \\ & \Rightarrow & [\![M]\!]_\rho = [\![N]\!]_\rho \\ & \Rightarrow & \mathcal{M}_\mathcal{E} \models M = N. \end{array}$$

(ii) If $M = N \in \mathcal{E}$, then $M =_\mathcal{E} N$, hence $\mathcal{M} \models M = N$, by (i). ■

Using this Corollary we obatin completeness in a simple way.

3.2.12. Theorem (Completeness). $\mathcal{E} \vdash M = N \iff \mathcal{E} \models M = N$.

Proof. ($\Rightarrow$) By soundness, Proposition 3.2.6.

$$\begin{array}{lll} (\Leftarrow)\ \mathcal{E} \models M = N & \Rightarrow & \mathcal{M}_\mathcal{E} \models M = N, \quad \text{as } \mathcal{M}_\mathcal{E} \models \mathcal{E}, \\ & \Rightarrow & M =_\mathcal{E} N \\ & \Rightarrow & \mathcal{E} \vdash M = N. \ \blacksquare \end{array}$$

3.2.13. Corollary. *Let $\mathcal{E}$ be a set of equations. Then*

$$\mathcal{E} \text{ has a non-trivial model} \iff \mathcal{E} \text{ is consistent.}$$

Proof. ($\Rightarrow$) By Proposition 3.2.7. ($\Leftarrow$) Suppose that $\mathcal{E} \nvdash x^o = y^o$. Then by the Theorem one has $\mathcal{E} \not\models x^o = y^o$. Then for some model $\mathcal{M}$ one has $\mathcal{M} \models \mathcal{E}$ and $\mathcal{M} \not\models x = y$. It follows that $\mathcal{M}$ is non-trivial. ■

If $\mathcal{D}$ contains enough constants, then one can similarly define the applicative structure $\mathcal{M}^\emptyset_{\mathcal{E}[\mathcal{D}]}$ by restricting $\mathcal{M}_\mathcal{E}$ to closed terms. See section 3.3.

Constructing Theories

The following result is due to Jacopini [1975].

3.2.14. Proposition. *Let $\mathcal{E}$ be a set of equations between closed terms in $\Lambda^\emptyset_o[\mathcal{D}]$. Then $\mathcal{E} \vdash M = N$ iff if for some $n \in \mathbb{N}$ and terms $F_1, \ldots, F_n$ and equations $P_1 = Q_1, \ldots, P_n = Q_n \in \mathcal{E}$ one has*

$$\left.\begin{array}{rcl} M & =_{\beta\eta} & F_1P_1Q_1 \\ F_1Q_1P_1 & =_{\beta\eta} & F_2P_2Q_2 \\ & \ldots & \\ F_{n-1}Q_{n-1}P_{n-1} & =_{\beta\eta} & F_nP_nQ_n \\ F_nQ_nP_n & =_{\beta\eta} & N. \end{array}\right\} \qquad (1)$$

This scheme (1) is called a Jacopini tableau and the sequence $F_1, \ldots, F_n$ is called the list of witnesses.

PROOF. ($\Leftarrow$) Obvious, since clearly $\mathcal{E} \vdash FPQ = FQP$ if $P = Q \in \mathcal{E}$.

($\Rightarrow$) By induction on the derivation of $M = N$ from the axioms. If $M = N$ is a $\beta\boldsymbol{\eta}$-axiom or the axiom of reflexivity, then we can take as witnesses the empty list. If $M = N$ is an axiom in $\mathcal{E}$, then we can take the singleton list K. If $M = N$ follows from $M = L$ and $L = N$, then we can concatenate the lists that exist by the induction hypothesis. If $M = N$ is $PZ = QZ$ (respectively $ZP = ZQ$) and follows from $P = Q$ with list $F_1, \ldots, F_n$, then the list for $M = N$ is $F_1', \ldots, F_n'$ with $F_i' \equiv \lambda ab.F_iabZ$ (respectively $F_i' \equiv \lambda ab.Z(F_iab)$. If $M = N$ follows from $N = M$, then we have to reverse the list. If $M = N$ is $\lambda x.P = \lambda x.Q$ and follows from $P = Q$ with list $F_1, \ldots, F_n$, then the new list is $F_1', \ldots, F_n'$ with $F_i' \equiv \lambda pqx.F_ipq$. Here we use that the equations in $\mathcal{E}$ are between closed terms. ■

Remember that $\mathsf{true} \equiv \lambda xy.x$, $\mathsf{false} \equiv \lambda xy.y$ both having type $1_2 = o \to o \to o$.

3.2.15. LEMMA. *Let $\mathcal{E}$ be a set of equations over D. Then*

$$\mathcal{E} \text{ is consistent} \iff \mathcal{E} \nvdash \mathsf{true} = \mathsf{false}.$$

PROOF. ($\Leftarrow$) By definition. ($\Rightarrow$) Suppose $\mathcal{E} \vdash \lambda xy.x = \lambda xy.y$. Then $\mathcal{E} \vdash P = Q$ for arbitrary $P, Q \in \Lambda_o(o)$. But then for arbitrary terms M, N of the same type $A = A_1 \to \ldots \to A_n \to o$ one has $\mathcal{E} \vdash M\vec{z} = N\vec{Z}$ for for fresh $\vec{z} = z_1, \ldots, z_n$ of the right type, hence $\mathcal{E} \vdash M = N$, by Proposition 3.2.4. ■

3.2.16. DEFINITION. Let $M, N \in \Lambda_o^{\emptyset}[D](A)$ be closed terms of type A.

(i) M is *inconsistent with* N, notation $M \# N$, if

$$\{M = N\} \vdash \mathsf{true} = \mathsf{false}.$$

(ii) M is *separable from* N, notation $M \perp N$, iff for some $F \in \Lambda_o^{\emptyset}[D](A \to 1_2)$

$$FM = \mathsf{true} \;\&\; FN = \mathsf{false}.$$

The following result is not true for the untyped lambda calculus: the equation $\mathsf{K} = \mathsf{YK}$ is inconsistent, but K and YK are not separable, as follows from the genericity lemma, see Barendregt [1984].

3.2.17. PROPOSITION. *Let $M, N \in \Lambda_o^{\emptyset}[D](A)$ be closed terms of type A. Then*

$$M \# N \iff M \perp N.$$

PROOF. ($\Leftarrow$) Trivially separability implies inconsistency.

($\Rightarrow$) Suppose $\{M = N\} \vdash \mathsf{true} = \mathsf{false}$. Then also $\{M = N\} \vdash x = y$. Hence by proposition 3.2.14 one has

$$\begin{aligned} x &=_{\beta\eta} F_1MN \\ F_1NM &=_{\beta\eta} F_2MN \\ &\ldots \\ F_nNM &=_{\beta\eta} y. \end{aligned}$$

Let n be minimal for which this is possible. We can assume that $\mathrm{FV}(F_i) \subseteq \{x, y\}$. The normal form of F_1NM must be either x or y. Hence by the minimality of n it must be y, otherwise there is a shorter list of witnesses. Now consider F_1MM. Its normal form must be either x or y.

Case 1: $F_1MM =_{\beta\eta} x$. Then set $F \equiv \lambda axy.F_1aM$ and we have $FM =_{\beta\eta} \mathsf{true}$ and $FN =_{\beta\eta} \mathsf{false}$.

Case 2: $F_1MM =_{\beta\eta} y$. Then set $F \equiv \lambda axy.F_1Ma$ and we have $FM =_{\beta\eta} \mathsf{false}$ and $FN =_{\beta\eta} \mathsf{true}$. ■

3.2.18. Corollary. *Let $\mathcal{E}$ be a set of equations over D. If $\mathcal{E}$ is inconsistent, then for some equation $M{=}N \in \mathcal{E}$ the terms M and N are separable.*

Proof. By the same reasoning. ■

3.2.19. Theorem. *Let*

$$\mathcal{E}_{max} = \{M{=}N \mid M, N \in \Lambda_o^{\emptyset}[D] \text{ and } M, N \text{ are not separable}\}.$$

Then this is the unique maximally consistent set of equations.

Proof. By the corollary this set is consistent. By Proposition 3.2.17 it contains all consistent equations. Therefore the set is maximally consistent. Moreover it is the unique such set. ■

It will be shown in Chapter 4 that $\mathcal{E}_{\max}$ is decidable.

3.3. Syntactic and Semantic logical relations

In this section we will introduce the well-known method of *logical relations* in two ways: one on the terms and one on elements of a model. Applications of the method will be given and it will be shown how the two methods are related.

Syntactic Logical Relations

3.3.1. Definition. Let n be a fixed natural number and let $\vec{\mathcal{D}} = \mathcal{D}_1, \ldots, \mathcal{D}_n$ be sets of constants.

(i) R is called an (n-ary) *family of relations* (or sometimes just a *relation*) on $\Lambda_o[\vec{D}]$, if $R = \{R_A\}_{A \in \mathbb{T}}$ and for $A \in \mathbb{T}$

$$R_A \subseteq \Lambda_o[\mathcal{D}_1](A) \times \ldots \times \Lambda_o[\mathcal{D}_n](A).$$

If we want to make the sets of constants explicit, we say that R is a relation *on terms from* $\mathcal{D}_1, \ldots, \mathcal{D}_n$.

(ii) Such an R is called a *logical relation* if for all $A, B \in \mathbb{T}$ and for all $M_1 \in \Lambda_o[\mathcal{D}_1](A{\to}B), \ldots, M_n \in \Lambda_o[\mathcal{D}_n](A{\to}B)$ one has

$$\begin{aligned} R_{A\to B}(M_1, \ldots, M_n) \iff & \forall N_1 \in \Lambda_o[\mathcal{D}_1](A) \ldots N_n \in \Lambda_o[\mathcal{D}_n](A) \\ & [R_A(N_1, \ldots, N_n) \Rightarrow R_B(M_1N_1, \ldots, M_nN_n)]. \end{aligned}$$

(iii) R is called empty if $R(o) = \emptyset$.

Obviously, a logical family $\{R_A\}$ is completely determined by R_o; the higher R_A do depend on the choice of the $\mathcal{D}_i$.

3.3.2. LEMMA. *If R is a non-empty logical relation, then $\forall A \in \mathbb{T}^o.R_A \neq \emptyset$.*

PROOF. (For R unary.) By induction on A. Case $A = o$. By assumption. Case $A = B{\to}C$. Then $R_{B\to C}(M) \iff \forall P \in \Lambda_o(B).[R_B(P) \Rightarrow R_C(MP)]$. By the induction hypothesis one has $R_C(N)$. Then $M \equiv \lambda p.N \in \Lambda_o(B{\to}C)$ is in R_A. ■

Even the empty logical relation is interesting.

3.3.3. PROPOSITION. *Let $n = 1, D_1 = \emptyset$ and R let be the logical relation determined by $R_o = \emptyset$. Then*

$$\begin{array}{rcll} R_A & = & \Lambda_o(A) & \text{if } \Lambda_o^{\emptyset}(A) \neq \emptyset; \\ & = & \emptyset & \text{else.} \end{array}$$

PROOF. By induction on A. If $A = o$, then we are done, as $R_o = \emptyset$ and $\Lambda_o^{\emptyset}(o) = \emptyset$. If $A = A_1, \ldots, A_n{\to}o$, then

$$\begin{array}{rcl} R_A(M) & \iff & \forall P_i \in R_{A_i}.R_o(M\vec{P}) \\ & \iff & \forall P_i \in R_{A_i}.\bot, \end{array}$$

seeing R both as a relation and as a set. This last statement either is always the case, namely iff

$$\begin{array}{rcll} \exists i.R_{A_i} = \emptyset & \iff & \exists i.\Lambda_o^{\emptyset}(A_i) = \emptyset & \text{by the induction hypothesis ,} \\ & \iff & \Lambda_o^{\emptyset}(A) \neq \emptyset, & \text{by proposition 2.4.4.} \end{array}$$

Or else, namely iff $\Lambda_o^{\emptyset}(A) = \emptyset$, it is never the case, by the same reasoning. ■

3.3.4. EXAMPLE. Let $n = 2$ and set $R_o(M, N) \iff M =_{\beta\boldsymbol{\eta}} N$. Let R be the logical relation determined by R_o. The it is easily seen that for all A and $M, N \in \Lambda_o[D](A)$ one has $R_A(M, N) \iff M =_{\beta\boldsymbol{\eta}} N$.

3.3.5. DEFINITION. (i) Let M, N be lambda terms. Then M is a *weak head expansion* of N, notation $M \to_{wh} N$, if $M \equiv (\lambda x.P)Q\vec{R}$ and $N \equiv P[x := Q]\vec{R}$.

(ii) A family R on $\Lambda_o[\mathcal{D}]$ is called *expansive* if R_o is closed under coordinatewise weak head expansion, i.e. if $M_i' \to_{wh} M_i$ for $1 \leq i \leq n$, then

$$R_o(M_1, \ldots, M_n) \Rightarrow R_o(M_1', \ldots, M_n').$$

3.3.6. LEMMA. *If R is logical and expansive, then each R_A is closed under coordinatewise weak head expansion.*

PROOF. Immediate by induction on the type A and the fact that

$$M' \to_{wh} M \;\Rightarrow\; M'N \to_{wh} MN. \blacksquare$$

3.3.7. EXAMPLE. (i) Let M be a term. We say that $\beta\boldsymbol{\eta}$ *confluence holds from* M, notation $\downarrow_{\beta\boldsymbol{\eta}} M$, if whenever $N_1 \;{}_{\beta\boldsymbol{\eta}}\!\twoheadleftarrow M \twoheadrightarrow_{\beta\boldsymbol{\eta}} N_2$, then there exists a term L such that $N_1 \twoheadrightarrow_{\beta\boldsymbol{\eta}} L \;{}_{\beta\boldsymbol{\eta}}\!\twoheadleftarrow N_2$. Define R_o by

$$R_o(M) \iff \beta\boldsymbol{\eta} \text{ confluence holds from } M.$$

Then R_o determines a logical R which is expansive by the permutability of head contractions with internal ones.

(ii) Let R be the logical relation generated from

$$R_o(M) \iff \downarrow_{\beta\boldsymbol{\eta}} M.$$

Then for arbitrary type $A \in \mathbb{T}$ one has

$$R_A(M) \;\Rightarrow\; \downarrow_{\beta\boldsymbol{\eta}} M.$$

[Hint. Write $M \downarrow_{\beta\boldsymbol{\eta}} N$ if $\exists Z\, [M \twoheadrightarrow_{\boldsymbol{\beta\eta}} Z \;{}_{\beta\boldsymbol{\eta}}\!\twoheadleftarrow N]$. First show that for arbitrary variable of some type B one has $R_B(x)$. Show also that if x is fresh, then by distinguishing cases whether x gets eaten or not

$$N_1 x \downarrow_{\beta\boldsymbol{\eta}} N_2 x \;\Rightarrow\; N_1 \downarrow_{\beta\boldsymbol{\eta}} N_2.$$

Then use induction on A.]

3.3.8. DEFINITION. Let R be n-ary and $*_1, \ldots, *_n$ be substitutors satisfying $*_i : \mathtt{Var}(A) \to \Lambda_o[\mathcal{D}_i](A)$.

(i) Write $R(*_1, \ldots, *_n)$ if $R_A(x^{*_1}, \ldots, x^{*_n})$ for each variable x of type A.

(ii) Define R^* by

$$R^*_A(M_1, \ldots, M_n) \iff \forall *_1 \ldots *_n\; [R(*_1, \ldots, *_n) \;\Rightarrow\; R_A(M_1^{*_1}, \ldots, M_n^{*_1})].$$

(iii) R is called *substitutive* if $R = R^*$.

3.3.9. LEMMA. (i) *Let R be logical and $M_1 \in \Lambda_o^{\emptyset}[\mathcal{D}_1], \ldots, M_n \in \Lambda_o^{\emptyset}[\mathcal{D}_n]$ be arbitrary closed terms. Then one has*

$$R(M_1, \ldots, M_n) \iff R^*(M_1, \ldots, M_n).$$

(ii) *For a substitutive R one has for arbitrary open $M_1, \ldots, M_n, N_1, \ldots, N_n$*

$$R(M_1, \ldots, M_n)\ \&\ R(N_1, \ldots, N_n) \;\Rightarrow\; R(M_1[x{:=}N_1], \ldots, M_n[x{:=}N_n]).$$

PROOF. (i) Clearly one has $R(M_1, \ldots, M_n) \;\Rightarrow\; R^*(M_1, \ldots, M_n)$. For the converse, note that $R^*(M_1, \ldots, M_n) \;\Rightarrow\; R(M_1, \ldots, M_n)$ in case $R_o \neq \emptyset$. But for $R_o = \emptyset$ the result follows from example 3.3.4.

(ii) Since R is substitutive we have $R^*(M_1,\dots,M_n)$. Let $*_i = [x{:=}N_i]$. Then $R(*_1,\dots,*_n)$ and hence $R(M_1[x{:=}N_1],\dots,M_n[x{:=}N_n])$. ■

3.3.10. EXERCISE. Let R be the logical relation generated by $R_o(M)$ iff $\downarrow_{\beta\boldsymbol{\eta}} M$. Show by induction on M that $R^*(M)$ for all M. [Hint. Use that R is expansive.] Conclude that for closed M one has $R(M)$ and hence $\downarrow_{\beta\boldsymbol{\eta}} M$. The same holds for arbitrary open terms N: let $\{\vec{x}\} = \mathrm{FV}(M)$, then

$$\begin{array}{lll} \lambda\vec{x}.N \text{ is closed} & \Rightarrow & R(\lambda\vec{x}.N) \\ & \Rightarrow & R((\lambda\vec{x}.N)\vec{x}), \qquad \text{since } R(x_i), \\ & \Rightarrow & R(N), \qquad \text{since } R \text{ is closed under } \twoheadrightarrow_\beta, \\ & \Rightarrow & \downarrow_{\beta\boldsymbol{\eta}} N. \end{array}$$

Thus the Church-Rosser property holds for $\twoheadrightarrow_{\boldsymbol{\beta\eta}}$.

3.3.11. PROPOSITION. *Let R be an arbitrary n-ary family on $\Lambda_o[\mathcal{D}]$. Then*

(i) *$R^*(x,\dots,x)$ for all variables.*
(ii) *If R is logical, then so is R^*.*
(iii) *If R is expansive, then so is R^*.*
(iv) *$R^{**} = R^*$, so R^* is substitutive.*
(v) *If R is logical and expansive, then*

$$R^*(M_1,\dots,M_n) \;\Rightarrow\; R^*(\lambda x.M_1,\dots,\lambda x.M_n).$$

PROOF. For notational simplicity we assume $n = 1$.

(i) If $R(*)$, then by definition $R(x^*)$. Therfore $R^*(x)$.

(ii) We have to prove

$$R^*(M) \iff \forall N \in \Lambda_o[\mathcal{D}][R^*(M) \;\Rightarrow\; R^*(MN)].$$

($\Rightarrow$) Assume $R^*(M)$ & R^*N) in order to show $R^*(MN)$. Let $*$ be a substitutor such that $R(*)$. Then

$$\begin{array}{lll} R^*(M) \;\&\; R^*(N) & \Rightarrow & R(M^*) \;\&\; R(N^*) \\ & \Rightarrow & R(M^*N^*) \equiv R((MN)^*) \\ & \Rightarrow & R^*(MN). \end{array}$$

($\Leftarrow$) By the assumption and (i) we have

$$R^*(Mx), \tag{1}$$

where we choose x to be fresh. In order to prove $R^*(M)$ we have to show $R(M^*)$, whenever $R(*)$. In order to do this it suffices to assume $R(N)$ and show $R(M^*N)$. Choose $*' = *(x{:=}N)$, then also $R(*')$. Hence by (1) and the freshness of x we have $R((Mx)^{*'}) \equiv R(M^*N)$ and we are done.

(iii) First observe that weak head reductions permute with substitution:

$$((\lambda x.P)Q\vec{R})^* \equiv (P[x{:=}Q]\vec{R})^*.$$

Now let $M \to_{wh} M^w$ be a weak head reduction step. Then

$$\begin{array}{rcl} R^*(M^w) & \Rightarrow & R(M^{w*}) \equiv R(M^{*w}) \\ & \Rightarrow & R(M^*) \\ & \Rightarrow & R^*(M). \end{array}$$

(iv) For substitutors $*_1, *_2$ write $*_1*_2$ for $*_2 \circ *_1$. This is convenient since

$$M^{*_1*_2} \equiv M^{*_2\circ*_1} \equiv (M^{*_1})^{*_2}.$$

Assume $R^{**}(M)$. Let $*_1(x) = x$ for all x. Then $R^*(*_1)$, by (i), and hence $R^*(M^{*_1}) \equiv R^*(M)$. Conversely, assume $R^*(M)$, i.e.

$$\forall * \, [R(*) \;\Rightarrow\; R(M^*)], \tag{2}$$

in order to show $\forall *_1 \, [R^*(*_1) \;\Rightarrow\; R(M^{*_1})]$. Now

$$\begin{array}{rcl} R^*(*_1) & \iff & \forall *_2 \, [R(*_2) \;\Rightarrow\; R(*_1*_2)], \\ R^*(M^{*_1)}) & \iff & \forall *_2 \, [R(*_2) \;\Rightarrow\; R(M^{*_1*_2})]. \end{array}$$

Therefore by (2) applied to $*_1*_2$ we are done.

(v) Let R be logical and expansive. Assume $R^*(M)$. Then

$$\begin{array}{rcll} R^*(N) & \Rightarrow & R^*(M[x{:=}N]), & \text{since } R^* \text{ is substitutive,} \\ & \Rightarrow & R^*((\lambda x.M)N), & \text{since } R^* \text{ is expansive.} \end{array}$$

Therefore $R^*(\lambda x.M)$ since R^* is logical. ■

3.3.12. Theorem (Fundamental theorem for syntactic logical relations). *Let R be logical, expansive and substitutive. Then for all $A \in \mathbb{T}$ and all pure terms $M \in \Lambda_o(A)$ one has*

$$R_A(M, \ldots, M).$$

Proof. By induction on M we show that $R_A(M, \ldots, M)$.

Case $M \equiv x$. Then the statement follows from the assumption $R = R^*$ (substitutivity) and propoition 3.3.11 (i).

Case $M \equiv PQ$. By the induction hypothesis and the assumption that R is logical.

Case $M \equiv \lambda x.P$. By the induction hypothesis and proposition 3.3.11 (v). ■

3.3.13. Corollary. *Let R be an n-ary expansive logical relation. Then for all closed $M \in \Lambda_o^{\emptyset}[\mathcal{D}]$ one has $R(M, \ldots, M)$.*

Proof. By the theorem applied to R^* and lemma 3.3.9. ■

The proof in exercise 3.3.10 was in fact an application of this corollary.

3.3.14. Example. Let R be the logical relation determined by

$$R_o(M) \iff M \text{ is normalizable.}$$

Then R is expansive. Note that if $R_A(M)$, then M is normalizable. [Hint. Use $R_B(x)$ for arbitrary B and x and the fact that if $M\vec{x}$ is normalizable, then so is M.] It follows from corollary 3.3.13 that each closed term is normalizable. By taking closures it follows that all terms are normalizable. This is the proof of weak normalization in Prawitz [1965].
For strong normalization a similar proof brakes down. The correspnding R is not expansive.

3.3.15. Example. A family of relations $S_A \subseteq \Lambda_o^\emptyset[\mathcal{D}_1](A)\times\ldots\times\Lambda_o^\emptyset[\mathcal{D}_n](A)$ which satisfies

$$\begin{aligned} S_{A\to B}(M_1,\ldots,M_n) \iff\ & \forall N_1\in\Lambda_o^\emptyset[\mathcal{D}_1](A)\ldots N_n\in\Lambda_o^\emptyset[\mathcal{D}_n](A) \\ & [S_A(N_1,\ldots,N_n) \Rightarrow S_B(M_1N_1,\ldots,M_nN_n)] \end{aligned}$$

can be lifted to a substitutive logical relation S^* on $\Lambda_o[\mathcal{D}_1]\times\ldots\times\Lambda_o[\mathcal{D}_n]$ as follows. Define for substitutors $*_i : \mathtt{Var}(A)\to\Lambda_o^\emptyset[\mathcal{D}_i](A)$

$$S_A(*_1,\ldots,*_n) \iff \forall x{:}A\ S_A(x^{*_1},\ldots,x^{*_n}).$$

Now define S^* as follows: for $M_i\in\Lambda_o[\mathcal{D}_i](A)$

$$S_A^*(M_1,\ldots,M_n) \iff \forall *_1\ldots *_n\,[S_A(*_1,\ldots,*_n) \Rightarrow S_A(M_1^{*_1},\ldots,M_n^{*_n})].$$

Show that if S is closed under coordinatewise weak head expansions, then S^* is expansive.

The following definition is needed on order to relate the syntactic and the semantic notions of logical relation.

3.3.16. Definition. Let R be an $n{+}1$-ary family. The *projection* of R, notation $\exists R$, is the n-ary family defined by

$$\exists R(M_1,\ldots,M_n) \iff \exists M_{n+1}\in\Lambda_o[\mathcal{D}_{n+1}]\ R(M_1,\ldots,M_{n+1}).$$

3.3.17. Proposition. (i) *The universal n-ary relation is defined by*

$$R_A = \Lambda_o[\mathcal{D}_1](A)\times\ldots\times\Lambda_o[\mathcal{D}_n](A).$$

This relation is logical, expansive and substitutive.

(ii) *Let $R\subseteq\mathcal{M}_1\times\ldots\times\mathcal{M}_n$ and $S\subseteq\mathcal{N}_1\times\ldots\times\mathcal{N}_m$ be non-empty logical relations. Then $R\times S\subseteq\mathcal{M}_1,\ldots,\mathcal{M}_n,\mathcal{N}_1,\ldots,\mathcal{N}_m$ is a non-empty logical relation. If moreover both R and S are substitutive, then so is $R\times S$.*

(iii) *If R is an n-ary family and π is a permutation of $\{1,\ldots,n\}$, then R^π defined by*

$$R^\pi(M_1,\ldots,M_n) \iff R(M_{\pi(1)},\ldots,M_{\pi(n)})$$

is logical if R is logical, is expansive if R is expansive and is substitutive if R is substitutive.

(iv) *Let R be an n-ary susbstitutive logical relation on terms from $\mathcal{D}_1, \ldots, \mathcal{D}_n$ and let $\mathcal{D} \subseteq \cap_i \mathcal{D}_i$. Then the diagonal of R, notation R^Δ, defined by*

$$R^\Delta(M) \iff R(M, \ldots, M)$$

is a substitutive logical (unary) relation on terms from $\mathcal{D}$, which is expansive if R is expansive.

(v) *If $\mathcal{R}$ is a class of n-ary substitutive logical relations, then $\cap\mathcal{R}$ is an n-ary substitutive logical relation, which is expansive if each member of $\mathcal{R}$ is expansive.*

(vi) *If R is an n-ary substitutive, expansive and logical relation, then $\exists R$ is a substitutive, expansive and logical relation.*

PROOF. (i) Trivial.

(ii) Suppose that R, S are logical. We show for $n = m = 1$ that $R \times S$ is logical.

$$\begin{array}{lcl}(R \times S)_{A\to B}(M,N) & \iff & R_{A\to B}(M)\ \&\ S_{A\to B}(N)\\ & \iff & [\forall P.R_A(P) \Rightarrow R_B(MP)]\ \&\\ & & [\forall Q.R_A(Q) \Rightarrow R_B(NQ)]\\ & \iff & \forall (P,Q).(R\times S)_A(P,Q) \ \Rightarrow\ (R\times S)_B(MP,NQ).\end{array}$$

For the last ($\Leftarrow$) one needs that the R, S are non-empty, and Lemma 3.3.2. If both R, S are expansive, then trivially so is $R \times S$.

(iii) Trivial.

(iv) We have

$$\begin{array}{lcl}R^\Delta(M) & \iff & R(M,M)\\ & \iff & \forall N_1,N_2.R(N_1,N_2) \ \Rightarrow\ R(MN_1,MN_2)\end{array}$$

and must show that the chain of equivalences continuous

$$\iff \forall N.R(N,N) \ \Rightarrow\ R(MN,MN). \qquad (1)$$

Now ($\Rightarrow$) is trivial. As to ($\Leftarrow$), suppose (1) and $R(N_1, N_2)$, in order to show $R(MN_1, MN_2)$. By Lemma 3.3.11(i) one has $R(x, x)$. Hence $R(Mx, Mx)$ by (1). Therefore $R^*(Mx, Mx)$, as R is substitutive. Now taking $*_i = [x := N_i]$, one obtains $R(MN_1, MN_2)$.

(v) Trivial.

(vi) Like in (iv) it suffices to show that

$$\forall P.[\exists R(P) \ \Rightarrow\ \exists R(MP)] \qquad (2)$$

implies $\exists N \forall P, Q.[R(P, Q) \ \Rightarrow\ R(MP, NQ)]$. Again we have $R(x, x)$. Therefore $\exists N_1.R^*(Mx, N_1)$ by (2). Writing $N \equiv \lambda x.N_1$, we get $R^*(Mx, Nx)$. Then $R(P, Q)$ implies as in (iv) that $R(MP, NQ)$. ■

The following property R states that an M essentially does not contain the constants from D. A term $M \in \Lambda_o[D]$ is called *pure* iff $M \in \Lambda_o$. The property $R(M)$ states that M is convertible to a pure term.

3.3.18. Proposition. *Define for* $M \in \Lambda_o[D](A)$

$$R_A(M) \iff \exists N \in \Lambda_o(A)\, M =_{\beta\eta} N.$$

Then

(i) R *is logical.*
(ii) R *is expansive.*
(iii) R *is substitutive.*

Proof. (i) If $R(M)$ and $R(N)$, then clearly $R(MN)$. Conversely, suppose $\forall N\,[R(N) \Rightarrow R(MN)]$. Since obviously $R(x)$ it follows that $R(Mx)$ for fresh x. Hence there exists a pure $L =_{\beta\eta} Mx$. But then $\lambda x.L =_{\beta\eta} M$, hence $R(M)$.

(ii) Trivial as $P \to_{\mathrm{wh}} Q \Rightarrow P =_{\beta\boldsymbol{\eta}} Q$.

(iii) We must show $R = R^*$. Suppose $R(M)$ and $R(*)$. Then $M = N$, with N pure and hence $M^* = N^*$ is pure, so $R^*(M)$. Conversely, suppose $R^*(M)$. Then for $*$ with $x^* = x$ one has $R(*)$. Hence $R(M^*)$. But this is $R(M)$. ■

3.3.19. Proposition. *Let* S *be an* n*-ary logical, expansive and substitutive relation on terms from* $\mathcal{D}_1, \ldots, \mathcal{D}_n$*. Define the restriction to pure terms* $S \restriction \Lambda$*, again a relation on terms from* $\mathcal{D}_1, \ldots, \mathcal{D}_n$*, by*

$$(S\restriction\Lambda)_A(M_1, \ldots, M_n) \iff R(M_1)\ \&\ \ldots\ \&\ R(M_n)\ \&\ S_A(M_1, \ldots, M_n),$$

where R *is as in proposition 3.3.18. Then* $S\restriction\Lambda$ *is logical, expansive and substitutive.*

Proof. Intersection of relations preserves the notion logical, expansive and substitutive. ■

3.3.20. Proposition. *Given* $\mathcal{E}$*, define* $R = R_{\mathcal{E}}$ *by*

$$R(M, N) \iff \mathcal{E} \vdash M = N.$$

Then

(i) R *is logical.*
(ii) R *is expansive.*
(iii) R *is substitutive.*
(iv) R *is a congruence relation.*

Proof. (i) This is proved in several steps.

(1) If $R(M, N)$ with witness list $F_1, \ldots, F_n$, then also $F_1, \ldots, F_n, \mathsf{K}(\mathsf{K}N)$ is a witness list of this fact.

(2) If $R(M_1, M_2)$ with witness list $F_1, \ldots, F_n$ and $R(N_1, N_2)$ with list $G_1, \ldots, G_m$, then we may suppose by (1) that $n = m$. But then $R(M_1N_1, M_2N_2)$ with list

$$\lambda xy.F_1xy(G_1xy), \ldots, F_nxy(G_nxy).$$

(3) $R(x, x)$ holds with witness list $\mathsf{K}(\mathsf{K}x)$.

(4) Suppose that whenever we have $R(N_1, N_2)$ we also have $R(M_1N_1, M_2N_2)$. Then for fresh z by (3) we have $R(M_1z, M_2z)$ with witnesses list, say, $F_1, \ldots, F_n$. Then $R(M_1, M_2)$ with list $\lambda xyz.F_1xy, \ldots, \lambda xyz.F_nxy$.

(ii) Obvious, since provability from $\mathcal{E}$ is closed under β-conversion, hence *a fortiori* under weak head expansion.

(iii) Assume that $R(M, N)$ in order to show $R^*(M, N)$. So suppose $R(x^{*1}, x^{*2})$. We must show $R(M^{*1}, N^{*2})$. Now going back to the definition of R this means that we have $\mathcal{E} \vdash M = N$ and $\mathcal{E} \vdash x^{*1} = x^{*2}$ and we must show $\mathcal{E} \vdash M^{*1} = N^{*2}$. Now if $\mathrm{FV}(MN) \subseteq \{\vec{x}\}$, then

$$\begin{array}{rll} M^{*1} & =_\beta & (\lambda\vec{x}.M)\vec{x}^{*1} \\ & =_\mathcal{E} & (\lambda\vec{x}.N)\vec{x}^{*2} \\ & =_\beta & N^{*2}. \end{array}$$

(iv) Obvious. ■

Semantic logical relations

3.3.21. Definition. Let $\mathcal{M}_1, \ldots, \mathcal{M}_n$ be applicative typed structures.

(i) R is an n-ary *family of relations* or just a *relation* on $\mathcal{M}_1 \times \ldots \times \mathcal{M}_n$ iff $R = \{R_A\}_{A \in \mathbb{T}}$ and for all A

$$R_A \subseteq \mathcal{M}_1(A) \times \ldots \times \mathcal{M}_n(A).$$

(ii) R is a *logical relation* iff

$$\begin{array}{rll} R_{A\to B}(d_1, \ldots, d_n) & \iff & \forall e_1 \in \mathcal{M}_1(A) \ldots e_n \in \mathcal{M}_n(A) \\ & & [R_A(e_1, \ldots, e_n) \;\Rightarrow\; R_B(d_1e_1, \ldots, d_ne_n)]. \end{array}$$

for all A, B and all $d_1 \in \mathcal{M}_1(A{\to}B), \ldots, d_n \in \mathcal{M}_n(A{\to}B)$.

Note that R is an n-ary relation on $\mathcal{M}_1 \times \ldots \times \mathcal{M}_n$ iff R is a unary relation on the single structure $\mathcal{M}_1 \times \ldots \times \mathcal{M}_n$.

3.3.22. Example. Define R on $\mathcal{M} \times \mathcal{M}$ by $R(d_1, d_2) \iff d_1 = d_2$. Then R is logical.

3.3.23. Example. Let $\mathcal{M}$ be a model and let $\pi = \pi_o$ be a permutation of $\mathcal{M}(o)$ which happens to be an element of $\mathcal{M}(o{\to}o)$. Then π can be lifted to higher types by defining

$$\pi_{A\to B}(d) = \lambda\!\!\lambda e \in \mathcal{M}(A).\pi_B(d(\pi_A^{-1}(e))).$$

Now define R_π (the graph of π)

$$R_\pi(d_1, d_2) \iff \pi(d_1) = d_2.$$

Then R is logical.

3.3.24. Example. (Friedman [1975]) Let $\mathcal{M},\mathcal{N}$ be typed structures. A *partial surjective homomorphism* is a family $h = \{h_A\}_{A\in}$ of surjections

$$h_A : \mathcal{M}(A)\to\mathcal{N}(A)$$

such that h_o is a surjection and

$$h_{A\to B}(d) = e \iff \begin{array}{l} e\in\mathcal{N}(A\to B) \text{ is the unique element (if it exists)} \\ \text{such that } \forall f\in \mathtt{dom}(h_A)\, e(h_A(f)) = h_B(d\cdot f). \end{array}$$

This implies that, if all elements involved exist, then

$$h_{A\to B}(d)h_A(f) = h_B(df).$$

Note that $h(d)$ can fail to be defined if one of the following conditions hold

1. for some $f\in\mathtt{dom}(h_A)$ one has $df \notin \mathtt{dom}(h_B)$;
2. the correspondence $h_A(f)\mapsto h_B(df)$ fails to be single valued;
3. the map $h_A(f)\mapsto h_B(df)$ fails to be in $\mathcal{N}_{A\to B}$.

Of course, 3 is the basic reason for partiality, whereas 1 and 2 are derived reasons. A partial surjective homomorphism h is completely determined by its h_o. If we take $\mathcal{M} = \mathcal{M}_X$ and h_o is any surjection $X\to\mathcal{N}_o$, then h_A is, although partial, indeed surjective for all A. Define $R_A(d,e) \iff h_A(d) = e$, the graph of h. Then R is logical. Conversely, if R_o is the graph of a partial surjective map $h_o : \mathcal{M}(o)\to\mathcal{N}(o)$, and the logical relation R induced by this R_o satisfies

$$\forall e\in\mathcal{N}(B)\exists d\in\mathcal{M}(A)\ R_B(d,e),$$

then R is the graph of a partial homomorphism from $\mathcal{M}$ to $\mathcal{N}$.

3.3.25. Definition. Let $\mathcal{M}_1,\dots,\mathcal{M}_n$ be typed structures.

(i) Let R be an n-ary relation on $\mathcal{M}_1,\dots,\mathcal{M}_n$. For valuations $\rho_1,\dots,\rho_n$ with $\rho_i : \mathtt{Var}\to\mathcal{M}_i$ we define

$$R(\rho_1,\dots,\rho_n) \iff R(\rho_1(x),\dots,\rho_n(x)), \text{ for all variables } x.$$

(ii) Let R be an n-ary relation on $\mathcal{M}_1\times\ldots\times\mathcal{M}_n$. The *lifting of* R to $\mathcal{M}_1^*,\dots,\mathcal{M}_n^*$, notation R^*, is defined by

$$R^*(d_1,\dots,d_n) \iff \forall\rho_1,\dots\rho_n\,[R(\rho_1,\dots,\rho_n) \Rightarrow R([\![d_1]\!]_{\rho_1},\dots,[\![d_n]\!]_{\rho_n})].$$

Here the ρ_i range over valuations in $\mathcal{M}_i$.

(iii) Let now R be an n-ary relation on $\mathcal{M}_1^*,\dots,\mathcal{M}_n^*$. We say that R is *substitutive* iff $R = R^*$.

3.3.26. Example. (i) Let R be the equality relation on $\mathcal{M}\times\mathcal{M}$. Then R^* is the equality relation on $\mathcal{M}^*\times\mathcal{M}^*$.

(ii) If R is the graph of a surjective homomorphism, then R^* is the graph of a partial surjective homomorphism whose restriction to $\mathcal{M}$ is R and which fixes each indeterminate x. [[Restriction in the literal sense, not the analogue of 3.3.19.]]

3.3.27. THEOREM. *(Fundamental Theorem for semantic logical relations) Let $\mathcal{M}_1,\ldots,\mathcal{M}_n$ be models and let R be a logical relation on $\mathcal{M}_1,\ldots,\mathcal{M}_n$. Then for each closed and pure term $M \in \Lambda_o^\emptyset$ one has*

$$R([\![M]\!]^{\mathcal{M}_1},\ldots,[\![M]\!]^{\mathcal{M}_n}).$$

PROOF. We treat the case $n = 1$. Let $R \subseteq \mathcal{M}$ be logical. We claim that for all $M \in \Lambda_o$ and all partial valuations ρ such that $\mathrm{dom}(\rho) \subseteq \mathrm{FV}(M)$ one has

$$R(\rho) \;\Rightarrow\; R([\![M]\!]_\rho).$$

This follows by an easy induction on M. In case $M \equiv \lambda x.N$ one should show $R([\![\lambda x.N]\!]_\rho)$, assuming $R(\rho)$. This means that for all d of the right type with $R(d)$ one has $R([\![\lambda x.N]\!]_\rho d)$. This is the same as $R([\![N]\!]_{\rho[x:=d]})$, which holds by the induction hypothesis.

The statement now follows immediately from the claim, by taking as ρ the empty function. ■

Relating syntactic and semantic logical relations

One may wonder whether the Fundamental Theorem for semantic logical relations follows from the syntactic version. This indeed is the case. The notion $\mathcal{M}^*$ for a model $\mathcal{M}$ will be useful for expressing the relation between syntactic and semantic logical relations.

3.3.28. PROPOSITION. (i) *The universal relation $R = \mathcal{M}_1^* \times \ldots \times \mathcal{M}_n^*$ is substitutive and logical on $\mathcal{M}_1^*,\ldots,\mathcal{M}_n^*$.*

(ii) *If R and S are respectively an n-ary relation on $\mathcal{M}_1,\ldots,\mathcal{M}_n$ and an m-ary relation on $\mathcal{N}_1,\ldots,\mathcal{N}_m$ and are both non-empty logical, then $R \times S$ is a $n+m$-ary non-empty logical relation on $\mathcal{M}_1,\ldots,\mathcal{M}_n,\mathcal{N}_1,\ldots,\mathcal{N}_m$. If moreover both R and S are substitutive, then so is $R \times S$.*

(iii) *If R is an n-ary logical relation on $\mathcal{M}^n$ and π is a permutation of $\{1,\ldots,n\}$, then R^π defined by*

$$R^\pi(d_1,\ldots,d_n) \iff R(d_{\pi(1)},\ldots,d_{\pi(n)})$$

is a logical relation. If moreover R is substitutive, then so is R^π.

(iv) *If R is an n-ary substitutive logical relation on $\mathcal{M}^* \times \ldots \times \mathcal{M}^*$, then the diagonal R^Δ defined by*

$$R^\Delta(d) \iff R(d,\ldots,d)$$

is a unary substitutive logical relation on $\mathcal{M}^*$.

(v) *If $\mathcal{R}$ is a class of n-ary substitutive logical relations, then $\cap\mathcal{R}$ is a substitutive logical relation.*

(vi) *If R is an $(n+1)$-ary substitutive logical relation on $\mathcal{M}_1^*,\ldots,\mathcal{M}_{n+1}^*$ and $\mathcal{M}_{n+1}^*$ is a model, then $\exists R$ defined by*

$$\exists R(d_1,\ldots,d_n) \iff \exists d_{n+1}\, R(d_1,\ldots,d_{n+1})$$

is an n-ary substitutive logical relation.

(vii) *If R is an n-ary substitutive logical relation on $\mathcal{M}_1,\ldots,\mathcal{M}_n$, then R^* is an n-ary substitutive logical relation on $\mathcal{M}^*\times\ldots\times\mathcal{M}^*$.*

(viii) *If R is an n-ary substitutive logical relation on $\mathcal{M}_1,\ldots,\mathcal{M}_n$ and each $\mathcal{M}_i$ is a model, then*

$$R^*(d_1,\ldots,d_n) \iff R(\lambda\vec{x}.d_1,\ldots,\lambda\vec{x}.d_n),$$

where the variables on which the $\vec{d}$ depend are included in the list $\vec{x}$.

PROOF. All items are easy. As an example we do the last one for $n=1$.

$$\begin{array}{lll} R^*(d) & \iff & \forall\rho.[R(\rho) \Rightarrow R([\![d]\!]_\rho)] \\ & \iff & \forall\rho.[R(\rho) \Rightarrow R([\![(\lambda\vec{x}.d)\vec{x}]\!]_\rho)] \\ & \iff & \forall\vec{e}.[R(e_1) \Rightarrow \ldots \Rightarrow R(e_n) \Rightarrow R((\lambda\vec{x}.d)e_1\ldots e_n)] \\ & \iff & R(\lambda\vec{x}.d). \blacksquare \end{array}$$

3.3.29. EXAMPLE. Consider $\mathcal{M}_{\mathbb{N}}$ and define

$$R_o(n,m) \iff n\leq m,$$

where $\leq$ is the usual ordering on $\mathbb{N}$. Then $R^*\cap =^*$ is [[related to]] the set of hereditarily monotone functionals. Moreover $\exists(R^*)$ is [[related to]] the set of hereditarily majorizable functionals, see the section by Howard in Troelstra [1973].

Now we want to link the notions of syntactical and semantical logical relation.

3.3.30. NOTATION. Let $\mathcal{M}$ be an applicative structure. Write

$$\Lambda_o[\mathcal{M}] \;=\; \Lambda_o[\{\mathbf{c}_d \mid d\in\mathcal{M}\}].$$

3.3.31. DEFINITION. Define the binary relation $D\subseteq\mathcal{M}^\Lambda\times\mathcal{M}$ by

$$D(M,d) \iff \forall\rho\, [\![M^{\beta\boldsymbol{\eta}}]\!]_\rho = d,$$

where $M^{\beta\boldsymbol{\eta}}$ is the $\beta\boldsymbol{\eta}$-nf of M.

3.3.32. PROPOSITION. *D is a substitutive logical relation.*

PROOF. Substitutivity is easy. Moreover, we have

$$\begin{array}{lll} [\![M^{\beta\boldsymbol{\eta}}]\!] = d & \Rightarrow & \forall N, e.[\![N^{\beta\boldsymbol{\eta}}]\!] = e \ \Rightarrow \ [\![(MN)^{\beta\boldsymbol{\eta}}]\!] = [\![M^{\beta\boldsymbol{\eta}}]\!][\![N^{\beta\boldsymbol{\eta}}]\!] = de \\ & \Rightarrow & [\![(Mx)^{\beta\boldsymbol{\eta}}]\!] = dx, \text{ with } x \text{ fresh}, \\ & \Rightarrow & [\![M^{\beta\boldsymbol{\eta}}]\!] = d. \end{array}$$

Therefore

$$\begin{array}{lll} D_{A\to B}(M,d) & \iff & [\![M^b e]\!] = d \\ & \iff & \forall N, e.[\![N^{\beta\boldsymbol{\eta}}]\!] = e \ \Rightarrow \ [\![(MN)^{\beta\boldsymbol{\eta}}]\!] = de \\ & \iff & \forall N, e.D_A(N,e) \ \Rightarrow \ D_B(MN, de). \ \blacksquare \end{array}$$

D is the connection between the two notions of logical relation.

3.3.33. DEFINITION. Let $\mathcal{M}_1, \ldots, \mathcal{M}_n$ be typed structures.

(i) Let R be a logical relation on $\mathcal{M}_1, \ldots, \mathcal{M}_n$. Let $R^\wedge$ be the relation on $\Lambda[\mathcal{M}_1], \ldots, \Lambda[\mathcal{M}_n]$ defined by

$$\begin{array}{lll} R^\wedge(M_1, \ldots, M_n) & \iff & \exists d_1 \in \mathcal{M}_1 \ldots d_n \in \mathcal{M}_n \\ & & [D^*(M_1, d_1) \ \& \ \ldots D^*(M_n, d_n) \ \& \ R^*(d_1, \ldots, d_n)]. \end{array}$$

(ii) Let S be a logical relation on $\Lambda[\mathcal{M}_1], \ldots, \Lambda[\mathcal{M}_n]$. Let $S^\vee$ be the relation on $\mathcal{M}_1, \ldots, \mathcal{M}_n$ defined by

$$\begin{array}{lll} S^\vee(d_1, \ldots, d_n) & \iff & \exists M_1 \in \Lambda[\mathcal{M}_1] \ldots M_n \in \Lambda[\mathcal{M}_n] \\ & & [D^*(M_1, d_1) \ \& \ \ldots D^*(M_n, d_n) \ \& \ S^*(M_1, \ldots, M_n)]. \end{array}$$

3.3.34. LEMMA. (i) *If R on $\mathcal{M}_1, \ldots, \mathcal{M}_n$ is logical, then on $\Lambda(\mathcal{M}_1), \ldots, \Lambda(\mathcal{M}_n)$ the relation $R^\wedge$ is expansive and logical.*

(ii) *If S on $\Lambda(\mathcal{M}_1), \ldots, \Lambda(\mathcal{M}_n)$ is expansive and logical, then $S^\vee$ is logical.*

PROOF. (i) First we show that $R^\wedge$ is logical. For notational simplicity we take $n = 1$. We must show

$$R^\wedge(M) \iff \forall N.[R^\wedge(N) \ \Rightarrow \ R^\wedge(MN)]$$

i.e.

$$\forall \rho.R([\![M^{\beta\boldsymbol{\eta}}]\!]_\rho) \iff \forall N.[(\forall \rho.R([\![N^{\beta\boldsymbol{\eta}}]\!]_\rho)) \ \Rightarrow \ (\forall \rho.R([\![(MN)^{\beta\boldsymbol{\eta}}]\!]_\rho))].$$

Now ($\Rightarrow$) is trivial. As to ($\Leftarrow$), suppose that $[\![M]\!]_\rho = d$ and that for all e with $R(e)$ one has $R(de)$ in order to show $R(d)$. Take $N = \underline{e}$. Then $R([\![M\underline{e}]\!]_\rho)$, hence $R(de)$. Therefore $R(d)$, as R is logical. This shows that $R^\wedge$ is logical. This relation is expansive by the fact that in the definition of D one takes the $\beta\boldsymbol{\eta}$-nf. [[Make notation c_d vs $\underline{d}$ consistent.]] $\blacksquare$

3.3.35. PROPOSITION. *Let $\mathcal{M}_1, \ldots, \mathcal{M}_n$ be models. Let R on $\mathcal{M}_1, \ldots, \mathcal{M}_n$ be logical and let S on $\Lambda[\mathcal{M}]_1, \ldots, \Lambda[\mathcal{M}]_n$ be expansive and logical. Then*

(i) *$R^\wedge$ is a substitutive logical relation.*

(ii) *$R^{\wedge\vee} = R^*$.*

(iii) $S^{\vee\wedge} = S^*$.

PROOF. (i) In order to show that $R^\wedge$ is substitutive, assume $R^\wedge(M)$, $R^\wedge(\vec{N})$ towards $R^\wedge(M[\vec{x}:=\vec{N}])$. Now $R^\wedge(M)$ means that $R([\![M^{\beta\boldsymbol{\eta}}]\!]_\rho)$, where $[\![M^{\beta\boldsymbol{\eta}}]\!]_\rho$ is independent of ρ. Since the $\vec{\mathcal{M}}$ are models we can leave out the ${}^{\beta\boldsymbol{\eta}}$. Therefore

$$[\![M[\vec{x}:=\vec{N}]]\!]_\rho = [\![M]\!]_{\rho[\vec{x}:=[\![\vec{N}]\!]_\rho]} = [\![M]\!],$$

hence $R^\wedge(M[\vec{x}:=\vec{N}])$.

(ii) Write $T = R^\wedge$. Then (taking $n=1$)

$$\begin{aligned} T^\vee(d) &\iff \exists M\in\Lambda[\mathcal{M}].D^*(M,d)\ \&\ T(M),\\ &\iff \exists M\in\Lambda[\mathcal{M}].D^*(M,d)\ \&\ \exists d'.D^*(M,d')\ \&\ R^*(d'),\\ &\iff \exists M,d'.[\![M]\!]=d\ \&\ [\![M]\!]=d'\ \&\ R^*(d'),\\ &\iff R^*(d). \end{aligned}$$

(iii) Similarly. ■

Using this result, the Fundamental Theorem for syntactical logical relations can be derived from the syntactic version.

We give two applications.

3.3.36. EXAMPLE. Let R be the graph of a partial surjective homomorphism $h:\mathcal{M}\to\mathcal{N}$. The fundamental theorem just shown implies that for closed pure terms one has $h(M)=M$, which is lemma 15 of Friedman [1975]. From this it is derived in this paper that for infinite X one has

$$\mathcal{M}_X \models M=N \iff M=_{\beta\eta} N.$$

We have derived this in another way.

3.3.37. EXAMPLE. Let $\mathcal{M}$ be a typed structure. Let $\Delta\subseteq\mathcal{M}$. Write $\Delta(A)=\Delta\cap\mathcal{M}(A)$. Assume that $\Delta(A)\neq\emptyset$ for all $A\in$ and

$$d\in\Delta(A\to B), e\in\Delta(A)\ \Rightarrow\ de\in\Delta(B).$$

Then Δ may fail to be a typed structure because it is not extensional. Equality as binary relation E_o on $\Delta(o)\times\Delta(o)$ induces a binary logical relation E on $\Delta\times\Delta$. Let $\Delta^E=\{d\in\Delta\mid E(d,d)\}$. Then the restriction of E to Δ^E is an applicative congruence and the equivalence classes form a structure. In particular, if $\mathcal{M}$ is a model, then write

$$\Delta^+=\{d\in\mathcal{M}\mid\exists M\Lambda_o^\emptyset\exists d_1\dots d_n\ [\![M]\!]d_1\dots d_n=d\}$$

for the applicative closure of Δ. The *Gandy hull* of Δ in $\mathcal{M}$ is the set Δ^{+E}. From the fundamental theorem for logical relations it can be derived that

$$\mathcal{M}_\Delta=\Delta^{+E}/E$$

is a model. This model will be also called the Gandy hull of Δ in $\mathcal{M}$.

3.4. Type reducibility

Remember that a type A is reducible to type B, notation $A \leq_{\beta\eta} B$ if for some closed term $\Phi{:}A{\to}B$ one has for all closed $M_1, M_2{:}A$

$$M_1 =_{\beta\eta} M_2 \iff \Phi M_1 =_{\beta\eta} \Phi M_2.$$

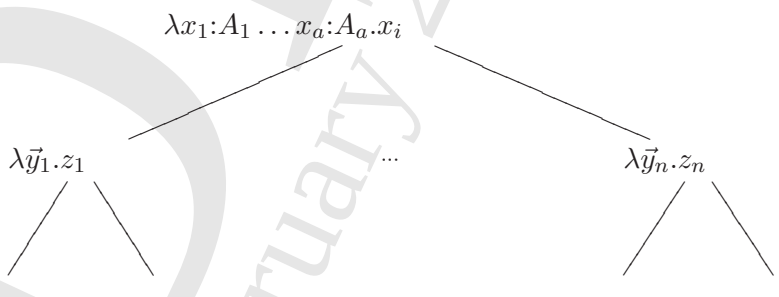
3.4.1. Definition. Write $A \sim_{\beta\boldsymbol{\eta}} B$ iff $A \leq_{\beta\eta} B$ & $B \leq_{\beta\eta} A$.

The reducibility theorem, Statman [1980a], states that there is one type to which all types of $\mathbb{T}(\lambda_{\to})$ can be reduced. At first this may seem impossible. Indeed, in a full typed structure $\mathcal{M}$ the cardinality of the sets of higher type increase arbitrarily. So one cannot always have an injection $\mathcal{M}_A{\to}\mathcal{M}_B$. But reducibility means that one restricts oneself to definable elements (modulo $=_{\beta\eta}$) and then the injections are possible. The proof will occupy[1] 3.4.2-3.4.7. There are four main steps. In order to show that $\Phi M_1 =_{\beta\eta} \Phi M_2 \Rightarrow M_1 =_{\beta\eta} M_2$ in all cases a (pseudo) inverse Φ^{-1} is used. Pseudo means that sometimes the inverse is not lambda definable, but this is no problem for the implication. Sometimes Φ^{-1} is definable, but the property $\Phi^{-1}(\Phi M) = M$ only holds in an extension of the theory; because the extension will be conservative over $=_{\beta\eta}$ the reducibility follows. Next the type hierarchy theorem, also due to Statman [1980a], will be given. Rather unexpectedly it turns out that under $\leq_{\beta\eta}$ types form a well-ordering of length $\omega+3$. Finally some consequences of the reducibility theorem will be given, including the 1-section and finite completeness theorems.

In the first step towards the reducibility theorem it will be shown that every type is reducible to one of rank ≤ 3. The proof is rather syntactic. In order to show that the definable function Φ is 1-1, a non-definable inverse is needed. A warm-up exercise for this is 3.6.4.

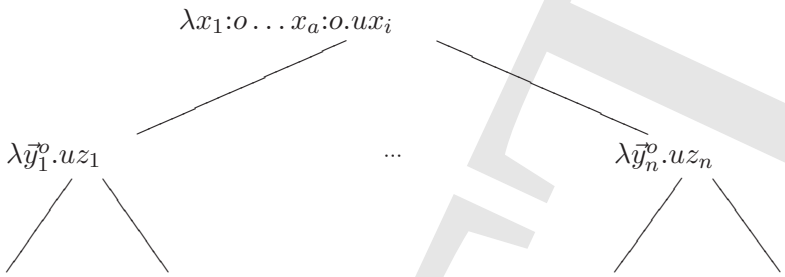
3.4.2. Proposition. *For every type A there is a type B such that* $\mathtt{rank}(B) \leq 3$ *and $A \leq_{\beta\eta} B$.*

Proof. [The intuition behind the construction of the the term Φ responsible for the reducibility is as follows. If M is a term with Böhmtree (see Barendregt [1984])



[1] A simpler alternative route disocvered later by Joly is described in the exercises 3.6.9 and 3.6.11, needing also exercise 3.6.10.

Now let UM be a term with "Böhmtree" of the form



where all the typed variables are pushed down to type o and the variables u (each occurrence possibly different) takes care that the new term remains typable. From this description it is clear that the u can be chosen in such way that the result has rank ≤ 1. Also that M can be reconstructed from UM so that U is injective. ΦM is just UM with the auxiliary variables bound. This makes it of type with rank ≤ 3. What is less clear is that U and hence Φ are lambda-definable.]

Define inductively for any type A the types $A^\sharp$ and $A^\flat$.

$$\begin{array}{rcl} o^\sharp & = & o; \\ o^\flat & = & o; \\ (A_1 \to \ldots \to A_a \to o)^\sharp & = & o \to A_1^\flat \to \ldots \to A_a^\flat \to o; \\ (A_1 \to \ldots \to A_a \to o)^\flat & = & (o^a \to o). \end{array}$$

Notice that $\mathtt{rank}(A^\sharp) \leq 2$.

In the potentially infinite context

$$\{u_A{:}A^\sharp \mid A \in \mathbb{T}\}$$

define inductively for any type A terms $V_A : o \to A, U_A : A \to A^\flat$.

$$\begin{array}{rcl} U_o & = & \lambda x{:}o.x; \\ V_o & = & \lambda x{:}o.x; \\ U_{A_1 \to \ldots \to A_a \to o} & = & \lambda z{:}A \lambda x_1 \ldots x_a{:}o.z(V_{A_1}x_1)\ldots(V_{A_a}x_a); \\ V_{A_1 \to \ldots \to A_a \to o} & = & \lambda x{:}o \lambda y_1{:}A_1 \ldots y_a{:}A_a.u_A x(U_{A_1}y_1)\ldots(U_{A_a}y_a), \end{array}$$

where $A = A_1 \to \ldots \to A_a \to o$. Write $A_i = A_{i1} \to \ldots \to A_{in} \to o$.

Remark that for $C = A_1 \to \ldots \to A_a \to B$ one has

$$U_C = \lambda z{:}C \lambda x_1 \ldots x_a{:}o.U_B(z(V_{A_1}x_1)\ldots(V_{A_a}x_a)). \tag{1}$$

Indeed, both sides are equal to

$$\lambda z{:}C \lambda x_1 \ldots x_a y_1 \ldots y_b{:}o.z(V_{A_1}x_1)\ldots(V_{A_a}x_a)(V_{B_1}y_1)\ldots(V_{B_b}y_b),$$

with $B = B_1 \to \ldots \to B_b \to o$.

Notice that for a closed term M of type $A = A_1 \to \ldots \to A_a \to o$ one can write

$$M =_\beta \lambda y_1{:}A_1 \ldots y_a{:}A_a.y_i(M_1 y_1 \ldots y_a)\ldots(M_n y_1 \ldots y_a),$$

with the $M_1, \ldots, M_n$ closed. Now verify that

$$\begin{aligned} U_A M &= \lambda x_1 \ldots x_a{:}o.M(V_{A_1}x_1)\ldots(V_{A_a}x_a) \\ &= \lambda\vec{x}.(V_{A_i}x_i)(M_1(V_{A_1}x_1)\ldots(V_{A_a}x_a))\ldots(M_n(V_{A_1}x_1)\ldots(V_{A_a}x_a)) \\ &= \lambda\vec{x}.u_{A_i}x_i(U_{A_{i1}}(M_1(V_{A_1}x_1)\ldots(V_{A_a}x_a)))\ldots(U_{A_{in}}(M_n(V_{A_1}x_1)\ldots(V_{A_a}x_a))) \\ &= \lambda\vec{x}.u_{A_i}x_i(U_{B_1}M_1\vec{x})\ldots(U_{B_n}M_n\vec{x}), \end{aligned}$$

using (1), where $B_j = A_1{\to}\ldots{\to}A_a{\to}A_{ij}$ for $1 \leq j \leq n$ is the type of M_j. Hence we have that if $U_A M =_{\beta\boldsymbol{\eta}} U_A N$, then for $1 \leq j \leq n$

$$U_{B_j} M_j =_{\beta\boldsymbol{\eta}} U_{B_j} N_j.$$

Therefore it follows by induction on the complexity of M that if $U_A M =_{\beta\boldsymbol{\eta}} U_A N$, then $M =_{\beta\boldsymbol{\eta}} N$.

Now take as term for the reducibility $\Phi \equiv \lambda m{:}A\lambda u_{B_1}\ldots u_{B_k}.U_A m$, where the $\vec{u}$ are all the ones occurring in the construction of U_A. It follows that

$$A \leq_{\beta\eta} B_1^\sharp{\to}\ldots{\to}B_k^\sharp{\to}A^\flat.$$

Since $\mathtt{rank}(B_1^\sharp{\to}\ldots{\to}B_k^\sharp{\to}A^\sharp) \leq 3$, we are done. ■

For an alternative proof, see Exercise 3.6.9.

In the following proposition it will be proved that we can further reduce types to one particular type of rank 3. First do exercise 3.6.5 to get some intuition. We need the following notation.

3.4.3. Notation. (i) For $k \geq 0$ write

$$1_k = o^k{\to}o,$$

where in general $A^0{\to}o = o$ and $A^{k+1}{\to}o = A{\to}(A^k{\to}o)$.

(ii) For $k_1, \ldots, k_n \geq 0$ write

$$(k_1, \ldots, k_n) = 1_{k_1}{\to}\ldots{\to}1_{k_n}{\to}o.$$

(iii) For $k_{11}, \ldots, k_{1n_1}, \ldots, k_{m1}, \ldots, k_{mn_m} \geq 0$ write

$$\begin{pmatrix} k_{11} & \ldots & k_{1n_1} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ k_{m1} & \ldots & k_{mn_m} \end{pmatrix} = (k_{11}, \ldots, k_{1n_1}){\to}\ldots{\to}(k_{m1}, \ldots, k_{mn_m}){\to}o.$$

Note the "matrix" has a dented right side (the n_i are unequal in general).

3.4.4. Proposition. *Every type A of rank ≤ 3 is reducible to*

$$1_2{\to}1{\to}1{\to}2{\to}o.$$

PROOF. Let A be a type of rank ≤ 3. It is not difficult to see that A is of the form

$$A = \begin{pmatrix} k_{11} & \dots & k_{1n_1} \\ . & & . \\ . & & . \\ k_{m1} & \dots & k_{mn_m} \end{pmatrix}$$

We will first reduce A to type $3 = 2{\to}o$ using a term Φ containing free variables of type $1_2, 1, 1$ respectively acting as a 'pairing'. Consider the context

$$\{p{:}1_2, p_1{:}1, p_2{:}1\}.$$

Consider the notion of reduction p defined by the contraction rules

$$p_i(pM_1M_2){\to}_p M_1.$$

[There now is a choice how to proceed: if you like syntax, then proceed; if you prefer models omit paragraphs starting with ♣ and jump to those starting with ♠.]

♣ This notion of reduction satisfies the subject reduction property. Moreover $\beta\eta p$ is Church-Rosser, see Pottinger [1981]. This can be used later in the proof. [Extending the notion of reduction by adding

$$p(p_1M)(p_2M){\to}_s M$$

preserves the CR property. In the untyped calculus this is not the case, see Klop [1980] or Barendregt [1984], ch. 14.] Goto ♠.

♠ Given the pairing p, p_1, p_2 one can extend it as follows. Write

$$\begin{array}{rcll} p^1 & = & \lambda x{:}o.x; & \\ p^{k+1} & = & \lambda x_1 \dots x_n x_{n+1}{:}o.p(p^k x_1 \dots x_n)x_{n+1}; & \\ p_1^1 & = & \lambda x{:}o.x; & \\ p_{k+1}^{k+1} & = & p_2; & \\ p_i^{k+1} & = & \lambda z{:}o.p_i^k(p_1 z), & \text{for } i \leq k; \\ P^k & = & \lambda f_1 \dots f_k{:}1\lambda z{:}o.p^k(f_1 z)\dots(f_k z); & \\ P_i^k & = & \lambda g{:}1\lambda z{:}o.p_i^k(gz), & \text{for } i \leq k. \end{array}$$

We have that p^k acts as a coding for k-tuples of elements of type o with projections p_i^k. The P^k, P_i^k do the same for type 1. In context containing $\{f{:}1_k, g{:}1\}$ write

$$\begin{array}{rcl} f^{k\to 1} & = & \lambda z{:}o.f(p_1^k z)\dots(p_k^k z); \\ g^{1\to k} & = & \lambda z_1 \dots z_k{:}o.g(p^k z_1 \dots z_k). \end{array}$$

Then $f^{k\to 1}$ is f moved to type 1 and $g^{1\to k}$ is g moved to type 1_k.

Using $\beta\boldsymbol{\eta}p$-convertibility one can show

$$\begin{array}{rcl} p_i^k(p^k z_1 \dots z_k) & = & z_i; \\ P_i^k(P^k f_1 \dots f_k) & = & f_i; \\ f^{k\to 1,1\to k} & = & f. \end{array}$$

For $g^{1\to k,k\to 1} = g$ one needs s, the surjectivity of the pairing.

In order to define the term required for the reducibility start with the term $\Psi{:}A{\to}3$ (containing p, p_1, p_2 as only free variables). We need an auxiliary term Ψ^{-1}, acting as an inverse for Ψ in the presence of a "true pairing".

$$\begin{array}{rcl}
\Psi & \equiv & \lambda M\lambda F{:}2.M \\
 & & [\lambda f_{11}{:}1_{k_{11}} \dots f_{1n_1}{:}1_{k_{1n_1}}.p_1(F(P^{n_1} f_{11}^{k_{11}\to 1} \dots f_{1n_1}^{k_{1n_1}\to 1})] \dots \\
 & & [\lambda f_{m1}{:}1_{k_{m1}} \dots f_{mn_m}{:}1_{k_{mn_m}}.p_m(F(P^{n_m} f_{m1}^{k_{m1}\to 1} \dots f_{mn_m}^{k_{mn_m}\to 1})]; \\
\Psi^{-1} & \equiv & \lambda N{:}(2{\to}o)\lambda K_1{:}(k_{11},\dots,k_{1n_1}) \dots \lambda K_m{:}(k_{m1},\dots,k_{mn_m}). \\
 & & N(\lambda f{:}1.p^m[K_1(P_1^{n_1} f)^{1\to k_{11}} \dots (P_{n_1}^{n_1} f)^{1\to k_{1n_1}}] \dots \\
 & & \qquad [K_m(P_1^{n_m} f)^{1\to k_{m1}} \dots (P_{n_m}^{n_m} f)^{1\to k_{1n_m}}]).
\end{array}$$

Claim. For closed terms M_1, M_2 of type A we have

$$M_1 =_{\beta\boldsymbol{\eta}} M_2 \iff \Psi M_1 =_{\beta\boldsymbol{\eta}} \Psi M_2.$$

It then follows that for the reduction $A \leq_{\beta\eta} 1_2{\to}1{\to}1{\to}3$ we can take

$$\Phi = \lambda M{:}A.\lambda p{:}1_2\lambda p_1, p_2{:}1.\Psi M.$$

It remains to show the claim. The only interesting direction is ($\Leftarrow$). This follows in two ways. We first show that

$$\Psi^{-1}(\Psi M) =_{\beta\boldsymbol{\eta}p} M. \tag{1}$$

We will write down the computation for the "matrix"

$$\begin{pmatrix} k_{11} & \\ k_{21} & k_{22} \end{pmatrix}$$

which is perfectly general.

$$\begin{array}{rcl}
\Psi M & =_\beta & \lambda F{:}2.M[\lambda f_{11}{:}1_{k_{11}}.p_1(F(P^1 f_{11}^{k_{11}\to 1}))] \\
 & & [\lambda f_{21}{:}1_{k_{21}}\lambda f_{22}{:}1_{k_{22}}.p_2(F(P^2 f_{21}^{k_{21}\to 1} f_{22}^{k_{22}\to 1}))]; \\
\Psi^{-1}(\Psi M) & =_\beta & \lambda K_1{:}(k_{11})\lambda K_2{:}(k_{21},k_{22}). \\
 & & \Psi M\underline{(\lambda f{:}1.p^1[K_1(P_1^1 f)^{1\to k_{11}}][K_2(P_1^2 f)^{1\to k_{21}}(P_2^2 f)^{1\to k_{22}}])} \\
 & \equiv & \lambda K_1{:}(k_1 1)\lambda K_2{:}(k_{21},k_{22}).\Psi M\underline{H}, \text{ say,} \\
 & =_\beta & \lambda K_1K_2.M[\lambda f_{11}.p_1(H(P^1 f_{11}^{k_{11}\to 1}))] \\
 & & [\lambda f_{21}\lambda f_{22}.p_2(H(P^2 f_{21}^{k_{21}\to 1} f_{22}^{k_{22}\to 1}))]; \\
 & =_{\beta p} & \lambda K_1K_2.M[\lambda f_{11}.p_1(p^2[K_1 f_{11}][..\text{`junk'}..])] \\
 & & [\lambda f_{21}\lambda f_{22}.p_2(p^2[..\text{`junk'}..][K_2 f_{21} f_{22}])]; \\
 & =_p & \lambda K_1K_2.M(\lambda f_{11}.K_1 f_{11})(\lambda f_{21} f_{22}.K_2 f_{21} f_{22}) \\
 & =_\eta & \lambda K_1K_2.MK_1K_2 \\
 & =_\eta & M,
\end{array}$$

since

$$\begin{array}{rcl}
H(P^1 f_{11}) & =_{\beta p} & p^2[K_1 f_{11}][..\text{`junk'}..] \\
H(P^2 f_{21}^{k_{21}\to 1} f_{22}^{k_{22}\to 1}) & =_{\beta p} & p^2[..\text{`junk'}..][K_2 f_{21} f_{22}].
\end{array}$$

The argument now can be finished in a model theoretic or syntactic way.

♣ If $\Psi M_1 =_{\beta\boldsymbol{\eta}} \Psi M_2$, then $\Psi^{-1}(\Psi M_1) =_{\beta\boldsymbol{\eta}} \Psi^{-1}(\Psi M_2)$. But then by (1) $M_1 =_{\beta\boldsymbol{\eta}p} M_2$. It follows from the Church-Rosser theorem for $\beta\boldsymbol{\eta}p$ that $M_1 =_{\beta\boldsymbol{\eta}} M_2$, since these terms do not contain p. Goto ■.

♠ If $\Psi M_1 =_{\beta\boldsymbol{\eta}} \Psi M_2$, then

$$\lambda p{:}1_2\lambda p_1p_2{:}1.\Psi^{-1}(\Psi M_1) =_{\beta\boldsymbol{\eta}} \lambda p{:}1_2\lambda p_1p_2{:}1.\Psi^{-1}(\Psi M_2).$$

Hence

$$\mathcal{M}(\omega) \models \lambda p{:}1_2\lambda p_1p_2{:}1.\Psi^{-1}\Psi(M_1) = \lambda p{:}1_2\lambda p_1p_2{:}1.\Psi^{-1}(\Psi M_2).$$

Let $\mathbf{q}$ be an actual pairing on ω with projections $\mathbf{q_1}, \mathbf{q_2}$. Then in $\mathcal{M}(\omega)$

$$(\lambda p{:}1_2\lambda p_1p_2{:}1.\Psi^{-1}(\Psi M_1))\mathbf{q}\mathbf{q_1}\mathbf{q_2} = \lambda p{:}1_2\lambda p_1p_2{:}1.\Psi^{-1}(\Psi M_2)\mathbf{q}\mathbf{q_1}\mathbf{q_2}.$$

Since $(\mathcal{M}(\omega), \mathbf{q}, \mathbf{q_1}, \mathbf{q_2})$ is a model of $\beta\boldsymbol{\eta}p$ conversion it follows from (1) that

$$\mathcal{M}(\omega) \models M_1 = M_2.$$

But then $M_1 =_{\beta\boldsymbol{\eta}} M_2$, by a result of Friedman [1975]. ■

We will see below, corollary 3.4.23 (i), that Friedman's result will follow from the reducibility theorem. Therefore the syntactic approach is preferable.

The proof of the next proposition is again syntactic. A warm-up is exercise 3.6.7.

3.4.5. PROPOSITION. *Let A be a type of rank ≤ 2. Then*

$$2{\to}A \leq_{\beta\eta} 1{\to}1{\to}o{\to}A.$$

PROOF. Let $A \equiv (1^{k_1}, \ldots, 1^{k_n}) = 1_{k_1}{\to}\ldots 1_{k_n}{\to}o$. The term that will perform the reduction is relatively simple

$$\Phi \equiv \lambda M{:}(2{\to}A)\lambda f, g{:}1\lambda z{:}o\lambda b_1{:}1_{k_1}\ldots\lambda b_n{:}1_{k_n}.M(\lambda h{:}1.f(h(g(hz)))).$$

In order to show that for all $M_1, M_2{:}2{\to}A$ one has

$$\Phi M_1 =_{\beta\eta} \Phi M_2 \;\Rightarrow\; M_1 =_{\beta\eta} M_2,$$

we may assume w.l.o.g. that $A = 1_2{\to}o$. A typical element of $2{\to}1_2{\to}o$ is

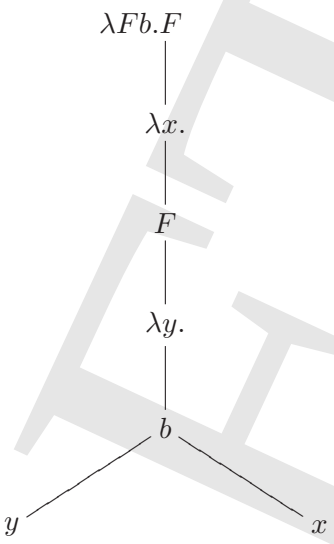
$$M \equiv \lambda F{:}2\lambda b{:}1_2.F(\lambda x.F(\lambda y.byx)).$$

Note that its translation has the following long $\beta\boldsymbol{\eta}$-nf

$$\begin{aligned}\Phi M &= \lambda f, g{:}1\lambda z{:}o\lambda b{:}1_2.f(N_x[x{:}=g(N_x[x{:}=z]])),\\ &\quad \text{where } N_x \equiv f(b(g(bzx))x),\\ &\equiv \lambda f, g{:}1\lambda z{:}o\lambda b{:}1_2.f(f(b(g(bz[g(f(b(g(bzz))z))]))[g(f(b(g(bzz))z))]).\end{aligned}$$

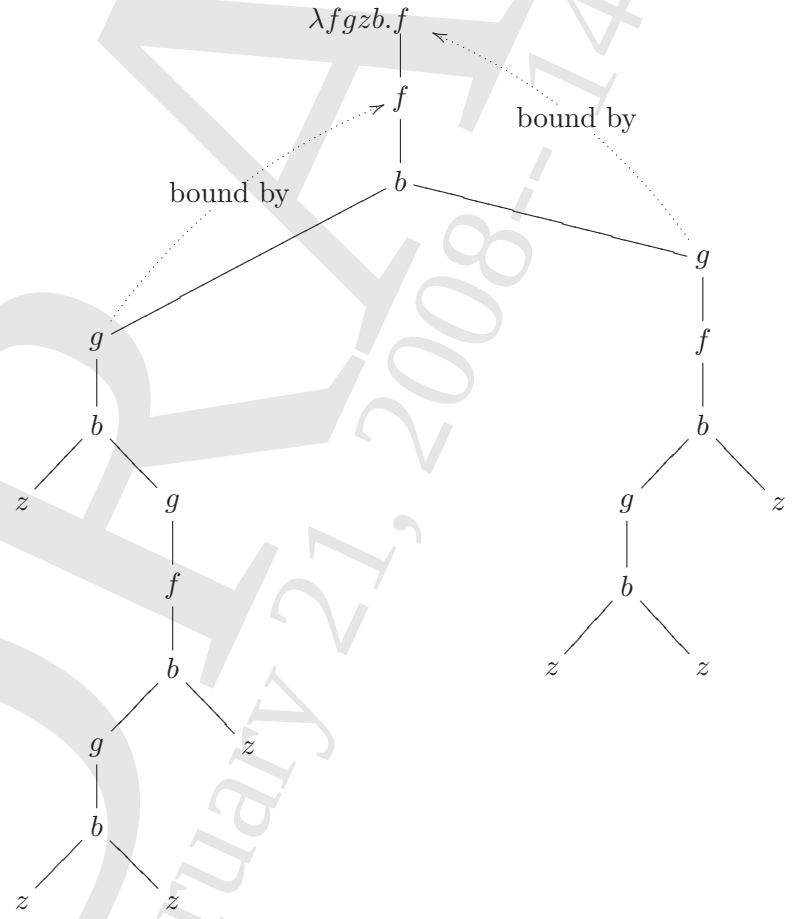
This term M and its translation have the following trees.

BT(M)



and

BT(ΦM)



Note that if we can 'read back' M from its translation ΦM, then we are done. Let $\texttt{Cut}_{g \to z}$ be a syntactic operation on terms that replaces maximal subterms

of the form gP by z. For example (omitting the abstraction prefix)

$$\mathtt{Cut}_{g\to z}(\Phi M) = f(f(bzz)).$$

Note that this gives us back the 'skeleton' of the term M, by reading f as $F(\lambda\odot$. The remaining problem is how to reconstruct the binding effect of each occurrence of the $\lambda\odot$. Using the idea of counting upwards lambda's, see de Bruijn [1972], this is accomplished by a realizing that the occurrence z coming from $g(P)$ should be bound at the position f just above where $\mathtt{Cut}_{g\to z}(P)$ matches in $\mathtt{Cut}_{g\to z}(\Phi M)$ above that z. For a precise inductive argument for this fact, see Statman [1980a], Lemma 5, or do exercise 3.6.10. ■

The following simple proposition brings almost to an end the chain of reducibility of types.

3.4.6. PROPOSITION.

$$1^4\to 1_2\to o\to o \leq_{\beta\eta} 1_2\to o\to o.$$

PROOF. As it is equally simple, let us prove instead

$$1\to 1_2\to o\to o \leq_{\beta\eta} 1_2\to o\to o.$$

Define $\Phi : (1\to 1_2\to o\to o)\to 1_2\to o\to o$ by

$$\Phi \equiv \lambda M{:}(1\to 1_2\to o\to o)\lambda b{:}1_2\lambda c{:}o.M(f^+)(b^+)c,$$

where

$$\begin{array}{rcl} f^+ & = & \lambda t{:}o.b(\#f)t; \\ b^+ & = & \lambda t_1, t_2{:}o.b(\#b)(bt_1t_2); \\ \#f & = & bcc; \\ \#b & = & bc(bcc). \end{array}$$

The terms $\#f, \#b$ serve as recognizers ("Gödel numbers"). Notice that M of type $1\to 1_2\to o\to o$ has a closed long $\beta\boldsymbol{\eta}$-nf of the form

$$M^{\mathrm{nf}} \equiv \lambda f{:}1\lambda b{:}1_2\lambda c{:}o.t$$

with t an element of the set T generated by the grammar

$$T ::= c \mid fT \mid b\ T\ T.$$

Then for such M one has $\Phi M =_{\beta\eta} \Phi(M^{\mathrm{nf}}) \equiv M^+$ with

$$M^+ \equiv \lambda f{:}1\lambda b{:}1_2\lambda c{:}o.t^+,$$

where t^+ is inductively defined by

$$\begin{array}{rcl} c^+ & = & c; \\ (ft)^+ & = & b(\#f)t^+; \\ (bt_1t_2)^+ & = & b(\#b)(bt_1^+t_2^+). \end{array}$$

It is clear that $M^{\rm nf}$ can be constructed back from M^+. Therefore

$$\begin{array}{lcl} \Phi M_1 =_{\beta\eta} \Phi M_2 & \Rightarrow & M_1^+ =_{\beta\eta} M_2^+ \\ & \Rightarrow & M_1^+ \equiv M_2^+ \\ & \Rightarrow & M_1^{\rm nf} \equiv M_2^{\rm nf} \\ & \Rightarrow & M_1 =_{\beta\eta} M_2. \ \blacksquare \end{array}$$

By the same method one can show that any type of rank ≤ 2 is reducible to $\top$, do exercise 3.6.13

Combining propositions 3.4.2-3.4.6 we can complete the proof of the reducibility theorem.

3.4.7. THEOREM (Reducibility theorem, Statman [1980a]). *Let*

$$\top = 1_2 \to o \to o.$$

Then

$$\forall A \in \mathbb{T} \ A \leq_{\beta\eta} \top.$$

PROOF. Let A be any type. Harvesting the results we obtain

$$\begin{array}{lll} A \leq_{\beta\eta} B, & & \text{with } \mathtt{rank}(B) \leq 3, \text{ by 3.4.2,} \\ \leq_{\beta\eta} 1_2 \to 1^2 \to 2 \to o, & & \text{by 3.4.4,} \\ \leq_{\beta\eta} 2 \to 1_2 \to 1^2 \to o, & & \text{by simply permuting arguments,} \\ \leq_{\beta\eta} 1^2 \to o \to 1_2 \to 1^2 \to o, & & \text{by 3.4.5,} \\ \leq_{\beta\eta} 1_2 \to o \to o, & & \text{by an other permutation and 3.4.6} \ \blacksquare \end{array}$$

Now we turn attention to the type hierarchy, Statman [1980a].

3.4.8. DEFINITION. For the ordinals $\alpha \leq \omega + 3$ define the type $A_\alpha \in \mathbb{T}(\lambda^o_\to)$ as follows.

$$\begin{array}{rcl} A_0 & = & o; \\ A_1 & = & o \to o; \\ & \cdots & \\ A_k & = & o^k \to o; \\ & \cdots & \\ A_\omega & = & 1 \to o \to o; \\ A_{\omega+1} & = & 1 \to 1 \to o \to o; \\ A_{\omega+2} & = & 3 \to o \to o; \\ A_{\omega+3} & = & 1_2 \to o \to o. \end{array}$$

3.4.9. PROPOSITION. *For $\alpha, \beta \leq \omega + 3$ one has*

$$\alpha \leq \beta \ \Rightarrow \ A_\alpha \leq_{\beta\eta} A_\beta.$$

PROOF. For all finite k one has $A_k \leq_{\beta\eta} A_{k+1}$ via the map

$$\Phi_{k,k+1} \equiv \lambda m{:}A_k \lambda z x_1 \dots x_k{:}o.m x_1 \dots x_k =_{\beta\eta} \lambda m{:}A_k.\mathsf{K} m.$$

Moreover, $A_k \leq_{\beta\eta} A_\omega$ via

$$\Phi_{k,\omega} \equiv \lambda m{:}A_k \lambda f{:}1 \lambda x{:}o.m(\mathbf{c}_1 f x) \dots (\mathbf{c}_k f x).$$

Then $A_\omega \leq_{\beta\eta} A_{\omega+1}$ via

$$\Phi_{\omega,\omega+1} \equiv \lambda m{:}A_\omega \lambda f, g{:}1 \lambda x{:}o.m f x.$$

Now $A_{\omega+1} \leq_{\beta\eta} A_{\omega+2}$ via

$$\Phi_{\omega+1,\omega+2} \equiv \lambda m{:}A_{\omega+1} \lambda H{:}3 \lambda x{:}o.H(\lambda f{:}1.H(\lambda g{:}1.m f g x)).$$

Finally, $A_{\omega+2} \leq_{\beta\eta} A_{\omega+3} = \top$ because of the reducibility theorem 3.4.7. Do Exercise 3.6.12 that asks for a concrete term $\Phi_{\omega+2,\omega+3}$. ■

3.4.10. PROPOSITION. *For $\alpha, \beta \leq \omega + 3$ one has*

$$\alpha \leq \beta \Leftarrow A_\alpha \leq_{\beta\eta} A_\beta.$$

PROOF. This will be proved in 3.5.32. ■

3.4.11. COROLLARY. *For $\alpha, \beta \leq \omega + 3$ one has*

$$A_\alpha \leq_{\beta\eta} A_\beta \iff \alpha \leq \beta.$$

For a proof that these types $\{A_\alpha\}_{\alpha \leq \omega+3}$ are a good representation of the reducibility classes we need some syntactic notions.

3.4.12. DEFINITION. A type $A \in \mathbb{T}(\lambda^o_{\to})$ is called *large* if it has a negative subterm occurrence of the form $B_1 \to \dots \to B_n \to o$, with $n \geq 2$; A is *small* otherwise.

3.4.13. EXAMPLE. $1_2 \to o \to o$ is large; $(1_2 \to o) \to o$ and $3 \to o \to o$ are small.

Now we will partition the types $\mathbb{T} = \mathbb{T}(\lambda^o_{\to})$ in the following classes.

3.4.14. DEFINITION. Define the following sets of types.

$$\begin{array}{lcl}
\mathbb{T}_{-1} & = & \{A \mid A \text{ has no closed inhabitant}\};\\
\mathbb{T}_0 & = & \{o \to o\};\\
\mathbb{T}_1 & = & \{o^k \to o \mid k > 1\};\\
\mathbb{T}_2 & = & \{1 \to o^q \to o \mid q > 0\} \cup \{(1_p \to o) \to o^q \to o \mid p > 0, q \geq 0\};\\
\mathbb{T}_3 & = & \{A \mid A \text{ is small, } \mathtt{rank}(A) \in \{2,3\} \text{ and } A \notin \mathbb{T}_2\};\\
\mathbb{T}_4 & = & \{A \mid A \text{ is small and } \mathtt{rank}(A) > 3\};\\
\mathbb{T}_5 & = & \{A \mid A \text{ is large}\}.
\end{array}$$

It is clear that the $\mathbb{T}_i$ form a partition of $\mathbb{T}$. A typical element of $\mathbb{T}_{-1}$ is o. This class we will not consider much.

3.4.15. THEOREM (Hierarchy theorem, Statman [1980a]).

$$\begin{array}{lcll} A\in\mathbb{T}_5 & \iff & A=_{\beta\eta} 1_2{\to}o{\to}o; & \\ A\in\mathbb{T}_4 & \iff & A=_{\beta\eta} 3{\to}o{\to}o; & \\ A\in\mathbb{T}_3 & \iff & A=_{\beta\eta} 1{\to}1{\to}o{\to}o; & \\ A\in\mathbb{T}_2 & \iff & A=_{\beta\eta} 1{\to}o{\to}o; & \\ A\in\mathbb{T}_1 & \iff & A=_{\beta\eta} o^k{\to}o, & \text{for some } k>1; \\ A\in\mathbb{T}_0 & \iff & A=_{\beta\eta} o{\to}o; & \\ A\in\mathbb{T}_{-1} & \iff & A=_{\beta\eta} o. & \end{array}$$

PROOF. Since the $\mathbb{T}_i$ form a partition, it is sufficient to show just the $\Rightarrow$'s.

As to $\mathbb{T}_5$, it is enough to show that $1_2{\to}o{\to}o \leq_{\beta\eta} A$, for every large type A, since we know already the converse. For this see Statman [1980a], lemma 7. As a warm-up exercise do 3.6.21.

As to $\mathbb{T}_4$, it is shown in Statman [1980a], proposition 2, that if A is small, then $A \leq_{\beta\eta} 3{\to}o{\to}o$. It remains to show that for any small type A of rank >3 one has $3{\to}o{\to}o \leq_{\beta\eta} A$. Do exercise 3.6.28.

As to $\mathbb{T}_3$, the implication is shown in Statman [1980a], lemma 12. The condition about the type in that lemma is equivalent to belonging to $\mathbb{T}_3$.

As to $\mathbb{T}_2$, do exercise 3.6.23(ii).

As to $\mathbb{T}_i$, with $i=1,0,-1$, notice that $\Lambda^{\emptyset}(o^k{\to}o)$ contains exactly k closed terms for $k\geq 0$. This is sufficient. ■

For an application in the next section we need a refinement of the hierarchy theorem.

3.4.16. DEFINITION. Let A, B be types.

(i) A is *head-redicible to* B, notation $A \leq_h B$, iff for some closed term $\Phi\in\Lambda^{\emptyset}(A{\to}B)$ one has

$$\forall M_1, M_2{:}A\,[M_1 =_{\beta\eta} M_2 \iff \Phi M_1 =_{\beta\eta} \Phi M_2],$$

and moreover Φ is of the form

$$\Phi = \lambda m{:}A\lambda x_1\ldots x_b{:}B.mP_1\ldots P_a, \qquad (+)$$

with $m\notin \mathrm{FV}(P_1,\ldots,P_a)$.

(ii) A is *multi head-reducible to* B, notation $A\leq_{h^+} B$, iff there are closed terms $\Phi_1,\ldots,\Phi_m\in\Lambda^{\emptyset}(A{\to}B)$ each of the form $(+)$ such that

$$\forall M_1, M_2{:}A\,[M_1 =_{\beta\eta} M_2 \iff \Phi_1M_1 =_{\beta\eta} \Phi_1M_2 \,\&\ldots\&\, \Phi_mM_1 =_{\beta\eta} \Phi_mM_2.$$

(iii) Write $A\sim_h B$ iff $A\leq_h B\leq_h A$ and similarly $A\sim_{h^+} B$ iff $A\leq_{h^+} B\leq_{h^+} A$.

3.4.17. PROPOSITION. (i) $A\leq_h B \Rightarrow A\leq_{\beta\eta} B$.

(ii) *Let* $A,B\in\mathbb{T}_i$, *with* $i\neq 2$. *Then* $A\sim_h B$.

(iii) *Let* $A,B\in\mathbb{T}_2$. *Then* $A\sim_{h^+} B$.

(iv) $A\leq_{\beta\eta} B \Rightarrow A\leq_{h^+} B$.

PROOF. (i) Trivial.

(ii) Suppose $A \leq_{\beta\eta} B$. By inspection of the proof of the hierarchy theorem in all cases except for $A \in \mathbb{T}_2$ one obtains $A \leq_h B$. Do exercise 3.6.25.

(iii) In the exceptional case one obtains $A \leq_{h^+} B$, see exercise 3.6.24. ■

(iv) By (ii) and (iii), using the hierarchy theorem.

3.4.18. COROLLARY (Hierarchy theorem (revisited), Statman [1980b]).

$$\begin{array}{lll}
A \in \mathbb{T}_5 & \iff & A \sim_h 1_2 \to o \to o; \\
A \in \mathbb{T}_4 & \iff & A \sim_h 3 \to o \to o; \\
A \in \mathbb{T}_3 & \iff & A \sim_h 1 \to 1 \to o \to o; \\
A \in \mathbb{T}_2 & \iff & A \sim_{h^+} 1 \to o \to o; \\
A \in \mathbb{T}_1 & \iff & A \sim_{h^+} o^2 \to o; \\
A \in \mathbb{T}_0 & \iff & A \sim_h o \to o; \\
A \in \mathbb{T}_{-1} & \iff & A \sim_h o.
\end{array}$$

PROOF. The only extra fact to verify is that $o^k \to o \leq_{h^+} o^2 \to o$. ■

Applications of the reducibility theorem

The reducibility theorem has several consequences.

3.4.19. DEFINITION. Let $\mathcal{C}$ be a class of $\lambda_{\to}$ models. $\mathcal{C}$ is called *complete* iff

$$\forall M, N \in \Lambda^{\emptyset}[\mathcal{C} \models M = N \iff M =_{\beta\boldsymbol{\eta}} N].$$

3.4.20. DEFINITION. (i) $\mathcal{T} = \mathcal{T}_{b,c}$ is the algebraic structure of trees inductively defined as follows.

$$\mathcal{T} = c \mid b\ \mathcal{T}\ \mathcal{T}$$

(ii) For a $\lambda_{\to}$ model $\mathcal{M}$ we say that $\mathcal{T}$ *can be embedded into* $\mathcal{M}$, notation $\mathcal{T} \hookrightarrow M$, iff there exist $b_0 \in \mathcal{M}(o \to o \to o), c_0 \in \mathcal{M}(o)$ such that

$$\forall t, s \in \mathcal{T}[t \neq s \ \Rightarrow\ \mathcal{M} \models t^{\mathrm{cl}} b_0 c_0 \neq s^{\mathrm{cl}} b_0 c_0],$$

where $u^{\mathrm{cl}} = \lambda b{:}o \to o \to o \lambda c{:}o.u$, is the closure of $u \in \mathcal{T}$.

The elements of $\mathcal{T}$ are binary trees with c on the leaves and b on the connecting nodes. Typical examples are $c, bcc, bc(bcc)$ and $b(bcc)c$. The existence of an embeding using b_0, c_0 implies for example that $b_0c_0(b_0c_0c_0), b_0c_0c_0$ and c_0 are mutually different in $\mathcal{M}$.

Note that $\mathcal{T} \not\hookrightarrow \mathcal{M}(2)$. To see this, write $gx = bxx$. One has $g^2(c) \neq g^4(c)$, but $\mathcal{M}(2) \models \forall g{:}o \to o \forall c{:}o.g^2(c) = g^4(c)$, do exercise 3.6.14.

3.4.21. LEMMA. (i) $\Pi_{i \in I}\mathcal{M}_i \models M = N \iff \forall i \in I.\mathcal{M}_i \models M = N$.

(ii) $M \in \Lambda^{\emptyset}(\top) \iff \exists s \in \mathcal{T}.M =_{\beta\boldsymbol{\eta}} s^{\mathrm{cl}}$.

PROOF. (i) Since $[\![M]\!]^{\Pi_{i \in I}\mathcal{M}_i} = \lambda i \in I.[\![M]\!]^{\mathcal{M}_i}$.

(ii) By an analysis of the possible shape of the normal forms of terms of type $\top = 1_2 \to o \to o$. ■

3.4.22. THEOREM (1-section theorem, Statman [1985]). *$\mathcal{C}$ is complete iff there is an (at most countable) family $\{\mathcal{M}_i\}_{i\in I}$ of structurs in $\mathcal{C}$ such that*

$$\mathcal{T} \hookrightarrow \Pi_{i\in I}\mathcal{M}_i.$$

PROOF. ($\Rightarrow$) Suppose $\mathcal{C}$ is complete. Let $t, s \in \mathcal{T}$. Then

$$\begin{array}{lll} t \neq s & \Rightarrow & t^{\mathrm{cl}} \neq_{\beta\boldsymbol{\eta}} s^{\mathrm{cl}} \\ & \Rightarrow & \mathcal{C} \not\models t^{\mathrm{cl}} = s^{\mathrm{cl}}, \qquad \text{by completeness,} \\ & \Rightarrow & \mathcal{M}_{ts} \models t^{\mathrm{cl}} \neq s^{\mathrm{cl}}, \qquad \text{for some } \mathcal{M}_{st} \in \mathcal{C}, \\ & \Rightarrow & \mathcal{M}_{ts} \models t^{\mathrm{cl}} b_{ts} c_{ts} \neq s^{\mathrm{cl}} b_{ts} c_{ts}, \end{array}$$

for some $b_{ts} \in \mathcal{M}(o{\to}o{\to}o), c_{ts} \in \mathcal{M}(o)$ by extensionality. Note that in the third implication the axiom of (countable) choice is used.

It now follows by lemma 3.4.21(i) that

$$\Pi_{t\neq s}\mathcal{M}_{ts} \models t^{\mathrm{cl}} \neq s^{\mathrm{cl}},$$

since they differ on the pair b_0c_0 with $b_0(ts) = b_{ts}$ and similarly for c_0.

($\Leftarrow$) Suppose $\mathcal{T} \hookrightarrow \Pi_{i\in I}\mathcal{M}_i$ with $\mathcal{M}_i \in \mathcal{C}$. Let M, N be closed terms of some type A. By soundness one has

$$M =_{\beta\boldsymbol{\eta}} N \;\Rightarrow\; \mathcal{C} \models M = N.$$

For the converse, let by the reducibility theorem $F : A{\to}\top$ be such that

$$M =_{\beta\boldsymbol{\eta}} N \iff FM =_{\beta\boldsymbol{\eta}} FN,$$

for all $M, N \in \Lambda^{\emptyset}$. Then

$$\begin{array}{lll} \mathcal{C} \models M = N & \Rightarrow & \Pi_{i\in I}\mathcal{M}_i \models M = N, \qquad \text{by the lemma,} \\ & \Rightarrow & \Pi_{i\in I}\mathcal{M}_i \models FM = FN, \\ & \Rightarrow & \Pi_{i\in I}\mathcal{M}_i \models t^{\mathrm{cl}} = s^{\mathrm{cl}}, \end{array}$$

where t, s are such that

$$FM =_{\beta\boldsymbol{\eta}} t^{\mathrm{cl}}, FN =_{\beta\boldsymbol{\eta}} s^{\mathrm{cl}}, \qquad (*)$$

noting that every closed term of type $\top$ is $\beta\boldsymbol{\eta}$-convertible to some u^{cl} with $u \in \mathcal{T}$. Now the chain of arguments continues as follows

$$\begin{array}{lll} & \Rightarrow & t \equiv s, \qquad \text{by the embedding property,} \\ & \Rightarrow & FM =_{\beta\boldsymbol{\eta}} FN, \qquad \text{by (*),} \\ & \Rightarrow & M =_{\beta\boldsymbol{\eta}} N, \qquad \text{by reducibility. } \blacksquare \end{array}$$

3.4.23. COROLLARY. (i) [Friedman [1975]] $\{\mathcal{M}_{\mathbb{N}}\}$ is complete.
(ii) [Plotkin [1985?]] $\{\mathcal{M}_n\}_{n\in\mathbb{N}}$ is complete.
(iii) $\{\mathcal{M}_{\mathbb{N}_\perp}\}$ is complete.
(iv) $\{\mathcal{M}_D\}_{D \text{ a finite cpo}}$ is complete.

PROOF. Immediate from the theorem. ■

The completeness of the collection $\{\mathcal{M}_n\}_{n\in\mathbb{N}}$ essentially states that for every pair of terms M, N of a given type A there is a number $n = n_{M,N}$ such that $\mathcal{M}_n \models M = N \Rightarrow M =_{\beta\eta} N$. Actually one can do better, by showing that n only depends on M.

3.4.24. PROPOSITION (Finite completeness theorem, Statman [1982]). *For every type A in $\mathbb{T}(\lambda^o_\to)$ and every $M \in \Lambda^\emptyset(A)$ there is a number $n = n_M$ such that for all $N \in \Lambda^\emptyset(A)$*

$$\mathcal{M}_n \models M = N \iff M =_{\beta\eta} N.$$

PROOF. By the reduction theorem 3.4.7 it suffices to show this for $A = \top$. Let M a closed term of type $\top$ be given. Each closed term N of type $\top$ has as long $\beta\boldsymbol{\eta}$-nf

$$N = \lambda b{:}1_2 \lambda c{:}o.s_N,$$

where $s_N \in \mathcal{T}$. Let $\mathbf{p} : \mathbb{N}\to\mathbb{N}\to\mathbb{N}$ be an injective pairing on the integers such that $\mathbf{p}(k_1, k_2) > k_i$. Take

$$n_M = [\![M]\!]^{\mathcal{M}_\omega}\mathbf{p}\,0 + 1.$$

Define $\mathbf{p}'{:}X^2_{n+1}\to X_{n+1}$, where $X_{n+1} = \{0, \ldots, n+1\}$, by

$$\begin{array}{rcll} \mathbf{p}'(k_1, k_2) & = & \mathbf{p}(k_1, k_2), & \text{if } k_1, k_2 \le n \mathbf{p}(k_1, k_2) \le n; \\ & = & n+1 & \text{else.} \end{array}$$

Suppose $\mathcal{M}_n \models M = N$. Then $[\![M]\!]^{\mathcal{M}_n}\mathbf{p}'\,0 = [\![N]\!]^{\mathcal{M}_n}\mathbf{p}'\,0$. By the choice of n it follows that $[\![M]\!]^{\mathcal{M}_n}\mathbf{p}\,0 = [\![N]\!]^{\mathcal{M}_n}\mathbf{p}\,0$ and hence $s_M = s_N$. Therefore $M =_{\beta\eta} N$. ■

3.5. The five canonical term-models

The open terms of $\lambda^o_\to$ form an extensional model, the term-model $\mathcal{M}_{\Lambda_o}$. One may wonder whether there are also closed term-models, like in the untyped lambda calculus. If no constants are present, then this is not the case, since there are e.g. no closed terms of ground type o. In the presence of constants matters change. We will first show how a set of constants $\mathcal{D}$ gives rise to an extensional equivalence relation on $\Lambda^\emptyset_o[\mathcal{D}]$, the set of closed terms with constants from $\mathcal{D}$. Then we define canonical sets of constants and prove that for these the resulting equivalence relation is also a congruence, i.e. determines a term-model. After that it will be shown that for all sets $\mathcal{D}$ of constants with enough closed terms the extensional equivalence determines a term-model. Up to elementary equivalence (satisfying the same set of equations between closed pure terms, i.e. closed terms without any constants) all models, for which the equality on type o coincides with $=_{\beta\eta}$, can be obtained in this way. From now on $\mathcal{D}$ will range over sets of constants such that there are closed terms for every type A (i.e. in $\Lambda^\emptyset_o[\mathcal{D}](A)$).

3.5.1. DEFINITION. Let $M, N \in \Lambda^{\emptyset}_o[\mathcal{D}](A)$ with $A = A_1 \to \ldots \to A_a \to o$.

(i) M is $\mathcal{D}$-*extensionally equivalent* with N, notation $M \approx^{\mathtt{ext}}_{\mathcal{D}} N$, iff

$$\forall t_1 \in \Lambda^{\emptyset}_o[\mathcal{D}](A_1) \ldots t_a \in \Lambda^{\emptyset}_o[\mathcal{D}](A_a). M\vec{t} =_{\beta\eta} N\vec{t}.$$

[If $a = 0$, then $M, N \in \Lambda^{\emptyset}_o[\mathcal{D}](o)$; in this case $M \approx^{\mathtt{ext}}_{\mathcal{D}} N \iff M =_{\beta\eta} N$.]

(ii) M is $\mathcal{D}$-*observationally equivalent* with N, notation $M \approx^{\mathtt{obs}}_{\mathcal{D}} N$, iff

$$\forall\, F \in \Lambda^{\emptyset}_o[\mathcal{D}](A \to o)\; FM =_{\beta\eta} FN. \blacksquare$$

3.5.2. REMARK. Note that if $M \approx^{\mathtt{obs}}_{\mathcal{D}} N$, then for all $F \in \Lambda^{\emptyset}_o[\mathcal{D}](A \to B)$ one has $FM \approx^{\mathtt{obs}}_{\mathcal{D}} FN$. Similarly, if $M \approx^{\mathtt{ext}}_{\mathcal{D}} N$ of type $A \to B$, then for all $Z \in \Lambda^{\emptyset}_o[\mathcal{D}](A)$ one has $MZ \approx^{\mathtt{ext}}_{\mathcal{D}} NZ$ of type B.

We will show that for all $\mathcal{D}$ and $M, N \in \Lambda^{\emptyset}_o[\mathcal{D}]$ one has

$$M \approx^{\mathtt{ext}}_{\mathcal{D}} N \iff M \approx^{\mathtt{obs}}_{\mathcal{D}} N.$$

Therefore, as soon as this is proved, we can write simply $M \approx_{\mathcal{D}} N$.

Note that in the definition of extensional equivalence the $\vec{t}$ range over closed terms (containing possibly constants). So this notion is not the same as $\beta\boldsymbol{\eta}$-convertibility: M and N may act differently on different variables, even if they act the same on all those closed terms. The relation $\approx^{\mathtt{ext}}_{\mathcal{D}}$ is related to what is called in the untyped calculus the ω-rule, see Barendregt [1984], §17.3.

The intuition behind observational equivalence is that for M, N of higher type A one cannot 'see' that they are equal, unlike for terms of type o. But one can do 'experiments' with M and N, the outcome of which is observational, i.e. of type o, by putting these terms in a context $C[-]$ resulting in two terms of type o. For closed terms it amounts to the same to consider just FM and FN for all $F \in \Lambda^{\emptyset}_o[\mathcal{D}](A \to o)$.

3.5.3. LEMMA. *Let S be an n-ary logical relation on $\Lambda^{\emptyset}_o[\mathcal{D}]$, that is closed under $=_{\beta\eta}$. If $S(\boldsymbol{d}, \ldots, \boldsymbol{d})$ holds for all $\boldsymbol{d} \in \mathcal{D}$, then*

$$S(M, \ldots, M)$$

holds for all $M \in \Lambda^{\emptyset}_o[\mathcal{D}]$.

PROOF. Let $\mathcal{D} = \{\boldsymbol{d}_1^{A_1}, \ldots, \boldsymbol{d}_n^{A_n}\}$. M can be written as

$$M \equiv M[\vec{\mathbf{d}}\,] =_{\beta\eta} (\lambda\vec{x}.M[\vec{x}])\vec{\mathbf{d}} \equiv M^+\vec{\boldsymbol{d}},$$

with M^+ a closed and pure term (i.e. without variables or constants). Then

$$\begin{array}{lll} & S(M^+, \ldots, M^+), & \text{by the fundamental theorem} \\ & & \text{for logical relations} \\ \Rightarrow & S(M^+\vec{\boldsymbol{d}}, \ldots, M^+\vec{\boldsymbol{d}}), & \text{since } S \text{ is logical and } \forall \boldsymbol{d} \in \mathcal{D}.S(\vec{\boldsymbol{d}}), \\ \Rightarrow & S(M, \ldots, M), & \text{since } S \text{ is } =_{\beta\eta} \text{ closed. } \blacksquare \end{array}$$

3.5.4. LEMMA. *Let $S = S^{\mathcal{D}} = \{S_A\}_{A \in \mathbb{T}(\lambda^o_\rightarrow)}$ be the logical relation on $\Lambda^{\emptyset}_o[\mathcal{D}](o)$ determined by*

$$S_o(M,N) \iff M =_{\beta\eta} N,$$

for $M, N \in \Lambda^{\emptyset}_o[\mathcal{D}](o)$.

(i) *Suppose that for all $\boldsymbol{d} \in \mathcal{D}$ one has $S(\boldsymbol{d},\boldsymbol{d})$. Then $\approx^{\mathtt{ext}}_{\mathcal{D}}$ is logical.*

(ii) *Let $\boldsymbol{d} \in \mathcal{D}$ be a constant of type $A \rightarrow o$ with $A = A_1 \rightarrow \cdots \rightarrow A_m \rightarrow o$. Suppose*

- $\forall F, G \in \Lambda^{\emptyset}_o[\mathcal{D}](A)[F \approx^{\mathtt{ext}}_{\mathcal{D}} G \;\Rightarrow\; F =_{\beta\eta} G]$;
- $\forall t_i \in \Lambda^{\emptyset}_o[\mathcal{D}](A_i)\; S(t_i,t_i), 1 \leq i \leq m$.

Then $S(\boldsymbol{d},\boldsymbol{d})$.

PROOF. (i) By the assumption and the fact that S is $=_{\beta\eta}$ closed (since S_o is), lemma 3.5.3 implies that

$$S(M,M) \tag{0}$$

for all $M \in \Lambda^{\emptyset}_o[\mathcal{D}]$. Hence S is an equivalence relation on $\Lambda^{\emptyset}_o[\mathcal{D}]$. Claim.

$$S_A(F,G) \iff F \approx^{\mathtt{ext}}_{\mathcal{D}} G,$$

for all $F, G \in \Lambda^{\emptyset}_o[\mathcal{D}](A)$. This is proved by induction on the structure of A. If $A = o$, then this is trivial. If $A = B \rightarrow C$, then we proceed as follows.

$$\begin{array}{lrcll}
(\Rightarrow) & S_{B\rightarrow C}(F,G) & \Rightarrow & S_C(Ft,Gt), & \text{for all } t \in \Lambda^{\emptyset}_o[\mathcal{D}](B).S_B(t,t) \\
 & & & \text{by the IH, since } t \approx^{\mathtt{ext}}_{\mathcal{D}} t \text{ and hence } S_B(t,t), & \\
 & & \Rightarrow & Ft \approx^{\mathtt{ext}}_{\mathcal{D}} Gt, & \text{for all } t \in \Lambda^{\emptyset}_o[\mathcal{D}], \text{by the IH}, \\
 & & \Rightarrow & F \approx^{\mathtt{ext}}_{\mathcal{D}} G, & \text{by definition}. \\
(\Leftarrow) & F \approx^{\mathtt{ext}}_{\mathcal{D}} G & \Rightarrow & Ft \approx^{\mathtt{ext}}_{\mathcal{D}} Gt, & \text{for all } t \in \Lambda^{\emptyset}_o[\mathcal{D}], \\
 & & \Rightarrow & S_C(Ft,Gt) & (1)
\end{array}$$

by the induction hypothesis. Now in order to prove $S_{B\rightarrow C}(F,G)$, assume $S_B(t,s)$ trying to show $S_C(Ft,Gs)$. Well, since also $S_{B\rightarrow C}(G,G)$, by (0), we have

$$S_C(Gt,Gs). \tag{2}$$

It follows from (1) and (2) and the transitivity of S (being by the IH on this type the same as $\approx^{\mathtt{ext}}_{\mathcal{D}}$) that $S_C(Ft,Gs)$ indeed.

So we have proved the claim. Since $\approx^{\mathtt{ext}}_{\mathcal{D}}$ is S and S is logical, it follows that $\approx^{\mathtt{ext}}_{\mathcal{D}}$ is logical.

(ii) Let $\boldsymbol{d}$ be given. Then

$$\begin{array}{lrcl}
S(F,G) & \Rightarrow & F\vec{t} =_{\beta\eta} G\vec{t}, & \text{since } \forall \vec{t} \in \Lambda^{\emptyset}_o[\mathcal{D}]\; S(t_i,t_i), \\
 & \Rightarrow & F \approx^{\mathtt{ext}}_{\mathcal{D}} G, & \\
 & \Rightarrow & F =_{\beta\eta} G, & \text{by assumption}, \\
 & \Rightarrow & \boldsymbol{d}F =_{\beta\eta} \boldsymbol{d}G. &
\end{array}$$

Therefore we have by definition $S(\boldsymbol{d},\boldsymbol{d})$. ■

3.5.5. LEMMA. *Suppose that* $\approx_{\mathcal{D}}^{\mathtt{ext}}$ *is logical. Then*

$$\forall M, N \in \Lambda_o^{\emptyset}[\mathcal{D}]\,[M \approx_{\mathcal{D}}^{\mathtt{ext}} N \iff M \approx_{\mathcal{D}}^{\mathtt{obs}} N].$$

PROOF. ($\Leftarrow$) Assume $M \approx_{\mathcal{D}}^{\mathtt{obs}} N$. Then $M \approx_{\mathcal{D}}^{\mathtt{ext}} N$ by taking $F \equiv \lambda m{:}B.m\vec{t}$.
($\Rightarrow$) Assume $M \approx_{\mathcal{D}}^{\mathtt{ext}} N$. Let $F \in \Lambda_o^{\emptyset}[\mathcal{D}](A{\to}o)$. Then and

$$\begin{array}{rrll} & F & \approx_{\mathcal{D}}^{\mathtt{ext}} & F, \\ \Rightarrow & FM & \approx_{\mathcal{D}}^{\mathtt{ext}} & FN, \quad \text{as by assumption } \approx_{\mathcal{D}}^{\mathtt{ext}} \text{ is logical,} \\ \Rightarrow & FM & =_{\beta\eta} & FN, \quad \text{because the type is } o. \end{array}$$

Therefore $M \approx_{\mathcal{D}}^{\mathtt{obs}} N$. ■

In order to show that for arbitrary $\mathcal{D}$ extensional equivalence is the same as observational equivalence first this wil be done for the following six sets of constants.

3.5.6. DEFINITION. The following sets of constants will play a crucial role in this section.

$$\begin{array}{rcl} \mathcal{C}_0 & = & \{\boldsymbol{c}^o\}; \\ \mathcal{C}_1 & = & \{\boldsymbol{c}^o, \boldsymbol{d}^o\}; \\ \mathcal{C}_2 & = & \{\boldsymbol{f}^1, \boldsymbol{c}^o\}; \\ \mathcal{C}_3 & = & \{\boldsymbol{f}^1, \boldsymbol{g}^1, \boldsymbol{c}^o\}; \\ \mathcal{C}_4 & = & \{\boldsymbol{\Phi}^3, \boldsymbol{c}^o\}; \\ \mathcal{C}_5 & = & \{\boldsymbol{b}^{1^2}, \boldsymbol{c}^o\}. \blacksquare \end{array}$$

From now on in this section $\mathcal{C}$ ranges over the canonical sets of constants $\{\mathcal{C}_0, \ldots, \mathcal{C}_5\}$ and $\mathcal{D}$ over an arbitrary set of constants.

3.5.7. DEFINITION. (i) We say that a type $A = A_1{\to}\ldots{\to}A_a{\to}o$ is *represented in* $\mathcal{D}$ iff there are distinct constants $\boldsymbol{d}_i$ of type A_i in $\mathcal{D}$.

(ii) Let $\mathcal{C}$ be one of the canonical sets of constants. Define the *characteristic type* of $\mathcal{C}$, notation $\nabla(\mathcal{C})$, as follows.

$$\begin{array}{rcl} \nabla(\mathcal{C}_0) & = & o{\to}o; \\ \nabla(\mathcal{C}_1) & = & o{\to}o{\to}o; \\ \nabla(\mathcal{C}_2) & = & 1{\to}o{\to}o; \\ \nabla(\mathcal{C}_3) & = & 1{\to}1{\to}o{\to}o; \\ \nabla(\mathcal{C}_4) & = & 3{\to}o{\to}o; \\ \nabla(\mathcal{C}_5) & = & 1^2{\to}o{\to}o. \end{array}$$

In other words, $\nabla(\mathcal{C}_i)$ is intuitively type of $\lambda\!\!\lambda\vec{\boldsymbol{c}}_i.\boldsymbol{c}^o$, where $\mathcal{C}_i = \{\vec{\boldsymbol{c}}_i\}$ (the order of the abstractions is immaterial). Note that $\nabla(\mathcal{C}_i)$ is represented in $\mathcal{C}_i$. Also one has

$$i \leq j \iff \nabla(\mathcal{C}_i) \leq_{\beta\eta} \nabla(\mathcal{C}_j),$$

as follows form the theory of type reducibility.

(iii) The *class* of $\mathcal{D}$ is

$$\mathtt{max}\{i \mid \nabla(\mathcal{C}_i) \leq_{\beta\eta} D \text{ for some type } D \text{ represented in } \mathcal{D}\}.$$

(iv) The *characteristic type* of $\mathcal{D}$ is $\nabla(\mathcal{C}_i)$, where i is the class of $\mathcal{D}$.

3.5.8. Lemma. *Let A be represented in $\mathcal{D}$. Then for $M, N \in \Lambda^{\emptyset}_o(A)$ pure closed terms of type A, one has*

$$M \approx^{\mathtt{ext}}_{\mathcal{D}} N \iff M =_{\beta\eta} N.$$

Proof. The ($\Leftarrow$) direction is trivial. As to ($\Rightarrow$)

$$\begin{array}{lll} M \approx^{\mathtt{ext}}_{\mathcal{D}} N & \iff & \forall \vec{T} \in \Lambda^{\emptyset}_o[\mathcal{D}].M\vec{T} =_{\beta\eta} N\vec{T} \\ & \Rightarrow & M\vec{\boldsymbol{d}} =_{\beta\eta} N\vec{\boldsymbol{d}}, \text{ for some } \boldsymbol{d} \in \mathcal{D} \text{ since} \\ & & \qquad A \text{ is represented in } \mathcal{D}, \\ & \Rightarrow & M\vec{x} =_{\beta\eta} N\vec{x}, \text{ since a constant may be replaced} \\ & & \qquad \text{by a variable if } M, N \text{ are pure,} \\ & \Rightarrow & M =_{\eta} \lambda\vec{x}.M\vec{x} =_{\beta\eta} \lambda\vec{x}.N\vec{x} =_{\eta} N. \ \blacksquare \end{array}$$

We will need the following combinatorial lemma about $\approx_{\mathcal{C}_4}$.

3.5.9. Lemma. *For every $F, G \in \Lambda[\mathcal{C}_4](2)$ one has*

$$F \approx_{\mathcal{C}_4} G \Rightarrow F =_{\beta\eta} G.$$

Proof. We must show

$$[\forall h \in \Lambda[\mathcal{C}_4](1).Fh =_{\beta\eta} Gh] \Rightarrow F =_{\beta\eta} G. \tag{1}$$

In order to do this, a classification has to be given for the elements of $\Lambda[\mathcal{C}_4](2)$. Define for $A \in \mathbb{T}(\lambda^o_{\to})$

$$A_{\Delta} = \{M \in \Lambda[\mathcal{C}_4](A) \mid \Delta \vdash M : A \ \&\ M \text{ in } \beta\boldsymbol{\eta}\text{-nf}\}.$$

It is easy to show that o_{Δ} and 2_{Δ} are generated by the following 'two-level' grammar, see van Wijngaarden et al. [1976].

$$\begin{array}{lll} 2_{\Delta} & = & \lambda f{:}1.o_{\Delta, f:1} \\ o_{\Delta} & = & \boldsymbol{c} \mid \boldsymbol{\Phi}\, 2_{\Delta} \mid \Delta.1\, o_{\Delta}, \end{array}$$

where $\Delta.A$ consists of $\{v \mid v^A \in \Delta\}$.

It follows that a typical element of $2_{\emptyset}$ is

$$\lambda f_1{:}1.\boldsymbol{\Phi}(\lambda f_2{:}1.f_1(f_2(\boldsymbol{\Phi}(\lambda f_3{:}1.f_3(f_2(f_1(f_3\,\boldsymbol{c})))))))).$$

Hence a general element can be represented by a list of words

$$\langle w_1, \ldots, w_n \rangle,$$

with $w_i \in \Sigma_i^*$ and $\Sigma_i = \{f_1, \ldots, f_i\}$, the representation of the typical element above being $\langle \epsilon, f_1 f_2, f_3 f_2 f_1 f_3 \rangle$.

Let $h_m = \lambda z^o.\mathbf{\Phi}(\lambda g{:}1.g^m(z))$; then $h_n \in 1_\emptyset$. We claim that for some m we have

$$Fh_m =_{\beta\eta} Gh_m \;\Rightarrow\; F =_{\beta\eta} G.$$

For a given $F \in \Lambda[\mathcal{C}_4](2)$ one can find a representation of the $\beta\boldsymbol{\eta}$-nf of Fh_m from the representation of the $\beta\boldsymbol{\eta}$-nf $F^{\mathtt{nf}} \in 2_\emptyset$ of F. It will turn out that if m is large enough, then $F^{\mathtt{nf}}$ can be determined ('read back') from the $\beta\boldsymbol{\eta}$-nf of Fh_m.

In order to see this, let $F^{\mathtt{nf}}$ be represented by the list of words $\langle w_1, \ldots, w_n \rangle$, as above. The occurences of f_1 can be made explicit and we write

$$w_i = w_{i0} f_1 w_{i1} f_1 w_{i2} \ldots f_1 w_{ik_i}.$$

Some of the w_{ij} will be empty (in any case the w_{1j}) and $w_{ij} \in \Sigma_i^{-*}$ with $\Sigma_i^- = \{f_2, \ldots, f_i\}$. Then $F^{\mathtt{nf}}$ can be written as (using for application—contrary to the usual convention—association to the right)

$$\begin{aligned} F^{\mathtt{nf}} \;\equiv\; & \lambda f_1.w_{10} f_1 w_{11} \ldots f_1 w_{1k_1} \\ & \mathbf{\Phi}(\lambda f_2.w_{20} f_1 w_{21} \ldots f_1 w_{2k_2} \\ & \ldots \\ & \mathbf{\Phi}(\lambda f_n.w_{n0} f_1 w_{n1} \ldots f_1 w_{nk_n} \\ & \boldsymbol{c})..). \end{aligned}$$

Now we have

$$\begin{aligned} (Fh_m)^{\mathtt{nf}} \;\equiv\; & w_{10} \\ & \mathbf{\Phi}(\lambda g.g^m w_{11} \\ & \ldots \\ & \mathbf{\Phi}(\lambda g.g^m w_{1k_1} \\ & \mathbf{\Phi}(\lambda f_2.w_{20} \\ & \mathbf{\Phi}(\lambda g.g^m w_{21} \\ & \ldots \\ & \mathbf{\Phi}(\lambda g.g^m w_{2k_2} \\ & \mathbf{\Phi}(\lambda f_3.w_{30} \\ & \mathbf{\Phi}(\lambda g.g^m w_{31} \\ & \ldots \\ & \mathbf{\Phi}(\lambda g.g^m w_{3k_3} \\ & \ldots \\ & \ldots \\ & \mathbf{\Phi}(\lambda f_n.w_{n0} \\ & \mathbf{\Phi}(\lambda g.g^m w_{n1} \\ & \ldots \\ & \mathbf{\Phi}(\lambda g.g^m w_{nk_n} \\ & \boldsymbol{c})..))..)..)))..)))..). \end{aligned}$$

So if $m > \mathtt{max}_{ij}\{\mathtt{length}(w_{ij})\}$ we can read back the w_{ij} and hence $F^{\mathtt{nf}}$ from $(Fh_m)^{\mathtt{nf}}$. Therefore using an m large enough (1) can be shown as follows:

$$\begin{array}{lll} \forall h \in \Lambda[\mathcal{C}_4](1). Fh =_{\beta\eta} Gh & \Rightarrow & Fh_m =_{\beta\eta} Gh_m \\ & \Rightarrow & (Fh_m)^{\mathtt{nf}} \equiv (Gh_m)^{\mathtt{nf}} \\ & \Rightarrow & F^{\mathtt{nf}} \equiv G^{\mathtt{nf}} \\ & \Rightarrow & F =_{\beta\eta} F^{\mathtt{nf}} \equiv G^{\mathtt{nf}} =_{\beta\eta} G. \ \blacksquare \end{array}$$

3.5.10. PROPOSITION. *The relations $\approx_{\mathcal{C}_i}$ are logical, for $1 \leq i \leq 5$.*

PROOF. Let S be the logical relation determined by $=_{\beta\eta}$ on type o. By lemma 3.5.4 (i) we have to check $S(\boldsymbol{c}, \boldsymbol{c})$ for all constants $\boldsymbol{c}$ in $\mathcal{C}_i$. For $i \neq 4$ this is easy (trivial for constants of type o and almost trivial for the ones of type 1 and 1^2; in fact for all terms $h \in \Lambda^{\emptyset}_o[\mathcal{C}]$ of these types one has $S(h, h)$).

For $i = 4$ it suffices by lemma 3.5.4 (ii) to show that

$$F \approx_{\mathcal{C}_4} G \ \Rightarrow \ F =_{\beta\eta} G$$

for all $F, G \in \Lambda^{\emptyset}_o[C_4](2)$. This has been done in lemma 3.5.9. ■

It follows that for the canonical sets $\mathcal{C}$ observational equivalence is the same as extensional equivalence.

3.5.11. THEOREM. *Let $\mathcal{C}$ be one of the canonical classes of constants. Then*

$$\forall M, N \in \Lambda^{\emptyset}_o[\mathcal{C}][M \approx^{\mathtt{obs}}_{\mathcal{C}} N \iff M \approx^{\mathtt{ext}}_{\mathcal{C}} N].$$

PROOF. By the proposition and lemma 3.5.5.

3.5.12. DEFINITION. Let $\mathcal{D}$ be an arbitrary set of constants. Define

$$\mathcal{M}_{\mathcal{D}} = \Lambda^{\emptyset}_o[\mathcal{D}]/\approx^{\mathtt{ext}}_{\mathcal{D}},$$

with application defined by

$$[F]_{\mathcal{D}}[M]_{\mathcal{D}} = [FM]_{\mathcal{D}}.$$

Here $[-]_{\mathcal{D}}$ denotes an equivalence class modulo $\approx_{\mathcal{D}}$.

3.5.13. THEOREM. *Let $\mathcal{C}$ be one of the canonical sets of constants.*

(i) *Application in $\mathcal{M}_{\mathcal{C}}$ is a well-defined.*

(ii) *For all $M, N \in \Lambda^{\emptyset}_o[\mathcal{C}]$ on has*

$$\mathcal{M}_{\mathcal{C}} \models M = N \iff M \approx_{\mathcal{C}} N.$$

(iii) *$\mathcal{M}_{\mathcal{C}}$ is an extensional term-model.*

PROOF. (i) Using theorem 3.5.11 one can show that the definition of the application operator is independent of the choice of representatives:

$$\begin{array}{ll} F \approx_{\mathcal{C}} F' \ \& \ M \approx_{\mathcal{C}} M' & \Rightarrow \\ F \approx_{\mathcal{C}}^{\mathtt{ext}} F' \ \& \ M \approx_{\mathcal{C}}^{\mathtt{obs}} M' & \Rightarrow \\ FM \approx_{\mathcal{C}}^{\mathtt{ext}} F'M \approx_{\mathcal{C}}^{\mathtt{obs}} F'M' & \Rightarrow \\ FM \approx_{\mathcal{C}} F'M'. & \blacksquare \end{array}$$

(ii) Show by induction on M that

$$[\![M]\!]_\rho = [M[\vec{x} := \rho(x_1), \ldots, \rho(x_n)]],$$

the $\approx_{\mathcal{C}}$ equivalence class, satisfies the semantic requirements.

(iii) Use (ii) and the fact that $\approx_{\mathcal{C}}^{\mathtt{ext}}$ the model is extensional. ■

3.5.14. DEFINITION. (i) If $\mathcal{M}$ is a model of $\lambda^o_{\rightarrow}[\mathcal{C}]$, then for a type A its A-section is simply $\mathcal{M}_A$.

(ii) We say that $\mathcal{M}$ is *A-complete* (respectively *A-complete for pure terms*) iff for all closed terms (respectively pure closed terms) M, N of type A one has

$$\mathcal{M} \models M = N \iff M =_{\beta\eta} N.$$

(iii) $\mathcal{M}$ is *complete (for pure terms)* if for all types $A \in \Lambda_o$ it is A-complete (for pure terms).

(iv) A model $\mathcal{M}$ is called *fully abstract* if observational equivalence is the same as equality in $\mathcal{M}$.

Using this terminology lemma 3.5.8 states that if A is represented in $\mathcal{C}$, then $\mathcal{M}_{\mathcal{C}}$ is A-complete for pure terms.

3.5.15. COROLLARY. *Let $\mathcal{C}$ one of the canonical classes of constants and let A be its characteristic type. Then $\mathcal{M}_{\mathcal{C}}$ has the following properties.*

(i) *$\mathcal{M}_{\mathcal{C}}$ is an extensional term-model.*
(ii) *$\mathcal{M}_{\mathcal{C}}$ is fully abstract.*
(iii) *$\mathcal{M}_{\mathcal{C}}$ is 0-complete.*
(iv) *$\mathcal{M}_{\mathcal{C}}$ is $\nabla(\mathcal{C})$-complete for pure terms.*

PROOF. (i) By theorem 3.5.11 the definition of application is well-defined. That extensionality holds follows from the definition of $\approx_{\mathcal{D}}$. Because all combinators $[\mathsf{K}_{AB}]_{\mathcal{C}}, [\mathsf{S}_{ABC}]_{\mathcal{C}}$ are in $\mathcal{M}_{\mathcal{C}}$ the structure is a model.

(ii) Again by theorem 3.5.11 $\mathcal{M}_{\mathcal{C}}$ is fully abstract.

(iii) Since on type o the relation $\approx_{\mathcal{C}}$ is just $=_{\beta\eta}$, the model is o-complete.

(iv) By lemma 3.5.8. ■

3.5.16. PROPOSITION. (i) *Let $i \leq j$. Then for pure closed terms $M, N \in \Lambda^{\emptyset}_o$*

$$\mathcal{M}_{\mathcal{C}_j} \models M = N \ \Rightarrow \ \mathcal{M}_{\mathcal{C}_i} \models M = N.$$

(ii) $\mathtt{Th}(\mathcal{M}_{\mathcal{C}_5}) \subseteq \ldots \subseteq \mathtt{Th}(\mathcal{M}_{\mathcal{C}_1})$.

PROOF. (i) $\mathcal{M}_{\mathcal{C}_i} \not\models M = N \Rightarrow M \not=_{\mathcal{C}_i} N$
$\Rightarrow M\vec{t}[\vec{\boldsymbol{d}}] \neq_{\beta\eta} N\vec{t}[\vec{\boldsymbol{d}}]$, for some $\vec{t}\,[\vec{\boldsymbol{d}}]$,
$\Rightarrow \lambda\vec{d}.M\vec{t}\,[\vec{d}] \neq_{\beta\eta} \lambda\vec{d}.N\vec{t}[\vec{d}]$
$\Rightarrow \Psi(\lambda\vec{d}.M\vec{t}\,[\vec{d}]) \neq_{\beta\eta} \Psi(\lambda\vec{d}.N\vec{t}[\vec{d}])$,
since $\nabla(\mathcal{C}_i) \leq_{\beta\eta} \nabla(\mathcal{C}_j)$ via some Ψ,
$\Rightarrow \Psi(\lambda\vec{d}.M\vec{t}\,[\vec{d}]) \not=_{\mathcal{C}_j} \Psi(\lambda\vec{d}.N\vec{t}[\vec{d}])$
$\Rightarrow \mathcal{M}_{\mathcal{C}_j} \not\models \Psi(\lambda\vec{d}.M\vec{t}\,[\vec{d}]) = \Psi(\lambda\vec{d}.N\vec{t}[\vec{d}])$
$\Rightarrow \mathcal{M}_{\mathcal{C}_j} \not\models M = N$ since $\mathcal{M}_{\mathcal{C}_i}$ is a model.

(ii) By (i). ■

It is known that the inclusions $\mathtt{Th}(\mathcal{M}_{\mathcal{C}_1}) \supseteq \mathtt{Th}(\mathcal{M}_{\mathcal{C}_2})$, $\mathtt{Th}(\mathcal{M}_{\mathcal{C}_2}) \supseteq \mathtt{Th}(\mathcal{M}_{\mathcal{C}_3})$ and $\mathtt{Th}(\mathcal{M}_{\mathcal{C}_4}) \supseteq \mathtt{Th}(\mathcal{M}_{\mathcal{C}_5})$ are proper, do exercise3.6.29. It is not known whether $\mathtt{Th}(\mathcal{M}_{\mathcal{C}_3}) = \mathtt{Th}(\mathcal{M}_{\mathcal{C}_4})$ holds.

3.5.17. LEMMA. *Let A, B be types such that $A \leq_{\beta\eta} B$. Suppose $\mathcal{M}_{\mathcal{D}}$ is B-complete for pure terms. Then $\mathcal{M}_{\mathcal{D}}$ is A-complete for pure terms.*

PROOF. Assume $\Phi : A \leq_{\beta\eta} B$. Then we have for $M, N \in \Lambda^{\emptyset}_o(A)$

$$\begin{array}{ccc} \mathcal{M}_{\mathcal{D}} \models M = N & \Leftarrow & M =_{\beta\eta} N \\ \Downarrow & & \Uparrow \\ \mathcal{M}_{\mathcal{D}} \models \Phi M = \Phi N & \Rightarrow & \Phi M =_{\beta\eta} \Phi N \end{array}$$

by the definition of reducibility. ■

3.5.18. COROLLARY. *$\mathcal{M}_{\mathcal{C}_5}$ is complete for pure terms. In other words, for $M, N \in \Lambda^{\emptyset}_o$*

$$\mathcal{M}_{\mathcal{C}_5} \models M = N \iff M =_{\beta\eta} N.$$

PROOF. By lemma 3.5.17 and the reducibility theorem 3.4.7. ■

So $\mathtt{Th}(\mathcal{M}_{\mathcal{C}_5})$, the smallest theory, is actually just $\beta\boldsymbol{\eta}$-convertibility, which is decidable. At the other end something dual happens.

3.5.19. DEFINITION. $\mathcal{M}_{\min} = \mathcal{M}_{\mathcal{C}_1}$ is called the *minimal model* of $\lambda_{\rightarrow}$ since it equates most terms.

3.5.20. PROPOSITION. *Let $A \in \mathbb{T}^o$ be of the form $A = A_1 \rightarrow \ldots A_n \rightarrow o$ and let $M, N \in \Lambda^{\emptyset}_o(A)$ be pure closed terms. Then the following statements are equivalent.*

1. *$M = N$ is inconsistent.*
2. *For all models $\mathcal{M}$ of $\lambda_{\rightarrow}$ one has $\mathcal{M} \not\models M = N$.*
3. *$\mathcal{M}_{\min} \not\models M = N$.*
4. *$\exists P_1 \in \Lambda^{x,y:o}(A_1) \ldots P_n \in \Lambda^{x,y:o}(A_n).M\vec{P} = x \;\&\; N\vec{P} = y$.*

5. $\exists F \in \Lambda^{x,y:o}(A{\to}o).FM = x \;\&\; FN = y.$

6. $\exists G \in \Lambda^{\emptyset}(A{\to}o^2{\to}o).FM = \lambda xy.x \;\&\; FN = \lambda xy.y.$

PROOF. (1) $\Rightarrow$ (2) By soundness. (2) $\Rightarrow$ (3) Trivial. (3) $\Rightarrow$ (4 Since $\mathcal{M}_{\min}$ consists of $\Lambda^{x,y:o}/\approx_{\mathcal{C}_1}$. (4) $\Rightarrow$ (5) By taking $F \equiv \lambda m.m\vec{P}$. (5) $\Rightarrow$ (6) By taking $G \equiv \lambda mxy.Fm$. (6) $\Rightarrow$ (1) Trivial. ■

3.5.21. COROLLARY. $\mathtt{Th}(\mathcal{M}_{\min})$ *is the unique maximally consistent extension of* $\lambda^o_{\to}$.

PROOF. By taking in the proposition the negations one has $M = N$ is consistent iff $\mathcal{M}_{\min} \models M = N$. Hence $\mathtt{Th}(\mathcal{M}_{\min})$ contains all consistent equations. Moreover this theory is consistent. Therefore the statement follows. ■

In Section 4.4 it will be proved that $\mathtt{Th}(\mathcal{M}_{\mathcal{C}_1})$ is decidable. $\mathcal{M}_{\mathcal{C}_0}$ is the degenerate model consisting of one element at each type, since

$$\forall M, N \in \Lambda^{\emptyset}_o[\mathcal{C}_0](o)\ M = x = N.$$

Therefore its theory is inconsistent and hence decidable. For the other theories, $\mathtt{Th}(\mathcal{M}_{\mathcal{C}_2})$, $\mathtt{Th}(\mathcal{M}_{\mathcal{C}_3})$ and $\mathtt{Th}(\mathcal{M}_{\mathcal{C}_4})$ it is not known whether they are decidable.

Now we turn attention again to arbitrary sets of constants $\mathcal{D}$. It will turn out that the results can be proved for arbitrary sets of constants $\mathcal{D}$.

3.5.22. DEFINITION. The set of types $\mathbb{T}_i$ is called *resource conscious* iff for some $A_1{\to}A_2{\to}\ldots{\to}A_a \in \mathbb{T}_i$ one has $A_1{\to}A_1{\to}A_2{\to}\ldots{\to}A_a \notin \mathbb{T}_i$. ■

Note that only $\mathbb{T}_2$ and $\mathbb{T}_0$ are resource conscious.

3.5.23. LEMMA. *Let $\mathcal{D}$ have class i with $\mathbb{T}_i$ not resource conscious and let $\mathcal{C} = \mathcal{C}_i$. Let $B = B_1{\to}\ldots{\to}B_b{\to}o$ and $P, Q{:}\Lambda^{\emptyset}_o[\mathcal{D}](B)$. Then*

(i) $\forall \vec{t} \in \Lambda^{\emptyset}_o[\mathcal{D}]\ P\vec{t} =_{\beta\eta} Q\vec{t}\,(:o) \;\Rightarrow\; \forall \vec{t} \in \Lambda^{\emptyset}_o[\mathcal{D}\cup\mathcal{C}]\ P\vec{t} =_{\beta\eta} Q\vec{t}$.

(ii) $P\vec{\boldsymbol{c}} \neq_{\beta\eta} Q\vec{\boldsymbol{c}} \;\Rightarrow\; PU[\vec{\boldsymbol{d}}] \neq_{\beta\eta} QU[\vec{\boldsymbol{d}}]$, *for some* $U[\vec{\boldsymbol{d}}] \in \Lambda^{\emptyset}_o[\mathcal{D}]$.

PROOF. (i) Suppose $P\vec{t} \neq_{\beta\eta} Q\vec{t}$ for some $\vec{t} \in \Lambda[\mathcal{D}\cup\mathcal{C}]$, in order to show that $P\vec{s} \neq_{\beta\eta} Q\vec{s}$ for some $\vec{s} \in \Lambda^{\emptyset}_o[\mathcal{D}]$. Write $\vec{t}\,[\vec{\mathbf{c}}, \vec{\mathbf{d}}\,]$, displaying explicitly the constants from $\mathcal{C}$ and $\mathcal{D}$. Then

$$\begin{array}{lrcll}
 & P\vec{t}\,[\vec{\mathbf{c}}, \vec{\mathbf{d}}\,] & \neq_{\beta\eta} & Q\vec{t}\,[\vec{\mathbf{c}}, \vec{\mathbf{d}}\,] & : o, \\
\Rightarrow & P\vec{t}\,[\vec{c}, \vec{\mathbf{d}}\,] & \neq_{\beta\eta} & Q\vec{t}\,[\vec{c}, \vec{\mathbf{d}}\,] & : o, \\
 & \multicolumn{4}{l}{\text{using variables } \vec{c} \text{ corresponding to } \vec{\mathbf{c}} \text{ (same number and types)},} \\
\Rightarrow & \lambda\vec{c}.P\vec{t}\,[\vec{c}, \vec{\mathbf{d}}\,] & \neq_{\beta\eta} & \lambda\vec{c}.Q\vec{t}\,[\vec{c}, \vec{\mathbf{d}}\,] & : C = \nabla(\mathcal{C}) \in \mathbb{T}_i, \\
\Rightarrow & \lambda\vec{d}\vec{c}.P\vec{t}\,[\vec{c}, \vec{d}\,] & \neq_{\beta\eta} & \lambda\vec{d}\vec{c}.Q\vec{t}\,[\vec{d}, \vec{d}\,] & : C' \in \mathbb{T}_i, \\
 & \multicolumn{4}{l}{\text{since } \mathbb{T}_i \text{ is not resource conscious},} \\
\Rightarrow & \Phi_k(\lambda\vec{d}\vec{c}.P\vec{t}\,[\vec{c}, \vec{d}\,]) & \neq_{\beta\eta} & \Phi_k(\lambda\vec{d}\vec{c}.Q\vec{t}\,[\vec{c}, \vec{d}\,]), & : D, \\
 & \multicolumn{4}{l}{\text{for some } D \text{ represented in } \mathcal{D} \text{ with } C' \leq_{h^+} D \text{ via } \Phi_1, \ldots, \Phi_m,} \\
\Rightarrow & (\lambda\vec{d}\vec{c}.P\vec{t}\,[\vec{c}, \vec{d}\,])\vec{M} & \neq_{\beta\eta} & (\lambda\vec{d}\vec{c}.Q\vec{t}\,[\vec{c}, \vec{d}\,])\vec{M} & : D, \\
\Rightarrow & (\lambda\vec{d}\vec{c}.P\vec{t}\,[\vec{c}, \vec{d}\,])\vec{M}\vec{\mathbf{d}} & \neq_{\beta\eta} & (\lambda\vec{d}\vec{c}.Q\vec{t}\,[\vec{c}, \vec{d}\,])\vec{M}\vec{\mathbf{d}} & : o, \\
\Rightarrow & P\vec{s}\,[\vec{\mathbf{d}}\,] & \neq_{\beta\eta} & Q\vec{s}\,[\vec{\mathbf{d}}\,] & : o.
\end{array}$$

(ii) By (i). ■

In exercise 3.6.26 it is shown that this lemma is false for $\mathcal{D}$ of class 0 and 2.

3.5.24. PROPOSITION. *Let $\mathcal{D}$ be of class i with $\mathbb{T}_i$ not resource conscious and canonical set $\mathcal{C} = \mathcal{C}_i$. Let A be an arbitrary type.*

(i) *For all $P[\vec{\mathbf{d}}], Q[\vec{\mathbf{d}}] \in \Lambda^{\emptyset}_o[\mathcal{D}](A)$, one has*

$$P[\vec{\boldsymbol{d}}] \approx^{\mathtt{ext}}_{\mathcal{D}} Q[\vec{\boldsymbol{d}}] \iff \lambda\vec{x}.P[\vec{x}] \approx_{\mathcal{C}} \lambda\vec{x}.Q[\vec{x}]$$

(ii) *In particular, for pure closed terms $P, Q \in \Lambda^{\emptyset}_o(A)$ one has*

$$P \approx^{\mathtt{ext}}_{\mathcal{D}} Q \iff P \approx_{\mathcal{C}} Q.$$

PROOF. (i) Although the proof is given in contrapositive form, it nevertheless can be made constructive.

($\Leftarrow$) We will show $P[\vec{\boldsymbol{d}}] \not\approx^{\mathtt{ext}}_{\mathcal{D}} Q[\vec{\boldsymbol{d}}] \Rightarrow \lambda\vec{x}.P[\vec{x}] \not\approx_{\mathcal{C}} \lambda\vec{x}.Q[\vec{x}]$. Indeed

$$\begin{array}{lll} P[\vec{\boldsymbol{d}}] \not\approx^{\mathtt{ext}}_{\mathcal{D}} Q[\vec{\boldsymbol{d}}] & \Rightarrow & P[\vec{\boldsymbol{d}}]\vec{T}[\vec{\boldsymbol{d}}, \vec{\boldsymbol{e}}] \neq_{\beta\eta} Q[\vec{\boldsymbol{d}}]\vec{T}[\vec{\boldsymbol{d}}, \vec{\boldsymbol{e}}], \text{ for some} \\ & & \vec{T}[\vec{\boldsymbol{d}}, \vec{\boldsymbol{e}}] \in \Lambda^{\emptyset}_o[\mathcal{D}], \text{ each } T_i[\vec{\boldsymbol{d}}, \vec{\boldsymbol{e}}] \text{ of the form } T_i\vec{\boldsymbol{d}}\vec{\boldsymbol{e}} \text{ with} \\ & & \text{the } T_i \text{ pure and closed and the } \vec{\boldsymbol{d}}, \vec{\boldsymbol{e}} \text{ all distinct,} \\ & \Rightarrow & P[\vec{x}]\vec{T}[\vec{x}, \vec{y}] \neq_{\beta\eta} Q[\vec{x}]\vec{T}[\vec{x}, \vec{y}] \ (: o), \\ & \Rightarrow & \lambda\vec{x}.P[\vec{x}] \not\approx_{\mathcal{C}} \lambda\vec{x}.Q[\vec{x}], \text{applying theorem 3.5.11 to} \\ & & P \equiv \lambda z\vec{x}\vec{y}.z\vec{x}\vec{T}[\vec{x}, \vec{y}]. \end{array}$$

($\Rightarrow$) Again we show the contrapositive.

$$\begin{array}{lll} \lambda\vec{x}.P[\vec{x}] \not\approx^{\mathtt{ext}}_{\mathcal{C}} \lambda\vec{x}.Q[\vec{x}] & \Rightarrow & P[\vec{V}[\vec{\boldsymbol{c}}]]\vec{W}[\vec{\boldsymbol{c}}] \neq_{\beta\eta} Q[\vec{V}[\vec{\boldsymbol{c}}]]\vec{W}[\vec{\boldsymbol{c}}], \\ & & \text{where } \vec{V}[\vec{\boldsymbol{c}}], \vec{W}[\vec{\boldsymbol{c}}] \in \Lambda^{\emptyset}_o[\mathcal{C}], \\ & \Rightarrow & P[\vec{\boldsymbol{x}}]\vec{W}[\vec{\boldsymbol{c}}] \neq_{\beta\eta} Q[\vec{\boldsymbol{x}}]\vec{W}[\vec{\boldsymbol{c}}], \\ & & \text{otherwise one could substitute for the } \vec{x}, \\ & \Rightarrow & P[\vec{\boldsymbol{d}}]\vec{W}[\vec{\boldsymbol{c}}] \neq_{\beta\eta} Q[\vec{\boldsymbol{d}}]\vec{W}[\vec{\boldsymbol{c}}], \\ & & \text{with } \vec{\boldsymbol{d}} \in \mathcal{D}, \\ & \Rightarrow & P[\vec{\boldsymbol{d}}]\vec{W}[\vec{U}[\boldsymbol{d}]] \neq_{\beta\eta} Q[\vec{\boldsymbol{d}}]\vec{W}[\vec{U}[\boldsymbol{d}]], \\ & & \text{by lemma 3.5.23(ii),} \\ & \Rightarrow & P[\vec{\boldsymbol{d}}] \not\approx^{\mathtt{ext}}_{\mathcal{D}} Q[\vec{\boldsymbol{d}}]. \end{array}$$

(ii) By (i). ■

3.5.25. COROLLARY. *Let $\mathcal{D}$ be a class that is not resource conscious.*

(i) *The relation $\approx^{\mathtt{ext}}_{\mathcal{D}}$ is logical.*

(ii) *The relations $\approx^{\mathtt{ext}}_{\mathcal{D}}$ and $\approx^{\mathtt{obs}}_{\mathcal{D}}$ on $\Lambda^{\emptyset}_o[\mathcal{D}]$ coincide.*

PROOF. (i) We have to show for all $F, G \in \Lambda^{\emptyset}_o[\mathcal{D}](A{\to}B)$ that $F \approx^{\mathtt{ext}}_{\mathcal{D}} G \iff$

$$\forall M, N \in \Lambda^{\emptyset}_o[\mathcal{D}](A)[M \approx^{\mathtt{ext}}_{\mathcal{D}} N \Rightarrow FM \approx^{\mathtt{ext}}_{\mathcal{D}} GN]. \tag{1}$$

($\Rightarrow$) Assume $F[\vec{\boldsymbol{d}}] \approx^{\mathtt{ext}}_{\mathcal{D}} G[\vec{\boldsymbol{d}}]$ and $M[\vec{\boldsymbol{d}}] \approx^{\mathtt{ext}}_{\mathcal{D}} N[\vec{\boldsymbol{d}}]$, with $F, G \in \Lambda^{\emptyset}_o[\mathcal{D}](B \to C)$ and $M, N \in \Lambda^{\emptyset}_o[\mathcal{D}](B)$. By the proposition one has $\lambda\vec{x}.F[\vec{x}] \approx_{\mathcal{C}} \lambda\vec{x}.G[\vec{x}]$ and $\lambda\vec{x}.M[\vec{x}] \approx_{\mathcal{C}} \lambda\vec{x}.N[\vec{x}]$. Consider the pure closed term

$$H \equiv \lambda f{:}(B \to C)\lambda m{:}B\lambda\vec{x}.f\vec{x}(m\vec{x}).$$

Then $H \approx_{\mathcal{C}} H$, since $\approx_{\mathcal{C}}$ is logical. It follows that

$$\begin{array}{rll} \lambda\vec{x}.F[\vec{x}]M[\vec{x}] & =_{\beta\eta} & H(\lambda\vec{x}.F[\vec{x}])(\lambda\vec{x}.M[\vec{x}]) \\ & \approx_{\mathcal{C}} & H(\lambda\vec{x}.F[\vec{x}])(\lambda\vec{x}.M[\vec{x}]) \\ & =_{\beta\eta} & \lambda\vec{x}.G[\vec{x}]N[\vec{x}]. \end{array}$$

But then again by the proposition

$$F[\vec{\boldsymbol{d}}]M[\vec{\boldsymbol{d}}] \approx^{\mathtt{ext}}_{\mathcal{D}} G[\vec{\boldsymbol{d}}]N[\vec{\boldsymbol{d}}].$$

($\Leftarrow$) Assume $F[\vec{\mathbf{d}}] \not\approx^{\mathtt{ext}}_{\mathcal{D}} G[\vec{\mathbf{d}}]$. Then by the proposition

$$\begin{array}{ll} & \lambda\vec{x}.F[\vec{x}] \not\approx_{\mathcal{C}} \lambda\vec{x}.G[\vec{x}], \\ \Rightarrow & F[\vec{T}[\vec{\mathbf{c}}]]S[\vec{\mathbf{c}}] \neq_{\beta\eta} G[\vec{T}[\vec{\mathbf{c}}]]S[\vec{\mathbf{c}}], \\ \Rightarrow & F[\vec{\mathbf{d}}]S[\vec{\mathbf{c}}] \neq_{\beta\eta} G[\vec{\mathbf{d}}]S[\vec{\mathbf{c}}], \\ & \text{otherwise one could substitute the } \vec{T}[\vec{\mathbf{c}}] \text{ for the } \vec{\mathbf{d}}, \\ \Rightarrow & F[\vec{\mathbf{d}}]S[\vec{U}[\vec{\mathbf{d}}]] \neq_{\beta\eta} G[\vec{\mathbf{d}}]S[\vec{U}[\vec{\mathbf{d}}]], \\ & \text{by lemma 3.5.23(ii),} \end{array}$$

contradicting (1), since of course $S[\vec{U}[\vec{\mathbf{d}}]] \approx^{\mathtt{ext}}_{\mathcal{D}} S[\vec{U}[\vec{\mathbf{d}}]]$.

(ii) That $\approx^{\mathtt{ext}}_{\mathcal{D}}$ is $\approx^{\mathtt{obs}}_{\mathcal{D}}$ on $\Lambda^{\emptyset}_o[\mathcal{D}]$ follows by lemma 3.5.5. ■

3.5.26. Lemma. *Let $\mathcal{D}$ be of class 2. Then one of the following cases holds.*

$$\begin{array}{rcl} \mathcal{D} & = & \{\boldsymbol{F}{:}2, \boldsymbol{x}_1 \ldots, \boldsymbol{x}_n{:}o\},\ n \geq 0; \\ \mathcal{D} & = & \{\boldsymbol{f}{:}1, \boldsymbol{x}_1 \ldots, \boldsymbol{x}_n{:}o\},\ n \geq 1. \end{array}$$

Proof. Do exercise 3.6.20. ■

3.5.27. Proposition. *Let $\mathcal{D}$ be of class 2. Then the following hold.*

(i) *The relation $\approx^{\mathtt{ext}}_{\mathcal{D}}$ is logical.*
(ii) *The relations $\approx^{\mathtt{ext}}_{\mathcal{D}}$ and $\approx^{\mathtt{obs}}_{\mathcal{D}}$ on $\Lambda^{\emptyset}_o[\mathcal{D}]$ coincide.*
(iii) $\forall M, N \in \Lambda^{\emptyset}_o\ [M \approx_{\mathcal{D}} N \iff M \approx_{\mathcal{C}_2} N]$.

Proof. (i) Assume that $\mathcal{D} = \{\boldsymbol{F}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ (the other possibility according lemma 3.5.26 is more easy). By proposition 3.5.4 (i) it suffices to show that for $\boldsymbol{d} \in \mathcal{D}$ one has $S(\boldsymbol{d}, \boldsymbol{d})$. This is easy for the ones of type o. For $\boldsymbol{F}{:}2$ we must show $f \approx^{\mathtt{ext}}_{\mathcal{D}} g \ \Rightarrow\ f =_{\beta\eta} g$ for $f, g \in \Lambda^{\emptyset}_o[\mathcal{D}](1)$. Now elements of $\Lambda^{\emptyset}_o[\mathcal{D}](1)$ are of the form

$$\lambda x_1.\boldsymbol{F}(\lambda x_2.\boldsymbol{F}(\ldots(\lambda x_{m-1}.\boldsymbol{F}(\lambda x_m.c))..)),$$

where $c \equiv x_i$ or $c \equiv \boldsymbol{x}_j$. Therefore if $f \neq_{\beta\eta} g$, then inspecting the various possibilities (e.g. one has

$$\begin{aligned} f &\equiv \lambda x_1.\boldsymbol{F}(\lambda x_2.\boldsymbol{F}(\ldots(\lambda x_{m-1}.\boldsymbol{F}(\lambda x_m.x_n))..)) \equiv \mathsf{K}A \\ g &\equiv \lambda x_1.\boldsymbol{F}(\lambda x_2.\boldsymbol{F}(\ldots(\lambda x_{m-1}.\boldsymbol{F}(\lambda x_m.x_1))..)), \end{aligned}$$

do exercise 3.6.19) one has $f(\boldsymbol{F}f) \neq_{\beta\eta} g(\boldsymbol{F}f)$ or $f(\boldsymbol{F}g) \neq_{\beta\eta} g(\boldsymbol{F}g)$, hence $f \not=^{\mathtt{ext}}_{\mathcal{D}} g$.

(ii) By (i) and lemma 3.5.5.

(iii) Let $M, N \in \Lambda^{\emptyset}_o$. Then, using exercise 3.6.22 (iii),

$$\begin{array}{lll} M \not=^{\mathtt{ext}}_{\mathcal{D}} N & \iff & \\ & \iff & Mt_1[\vec{\boldsymbol{d}}]\ldots t_n[\vec{\boldsymbol{d}}] \neq_{\beta\eta} Nt_1[\vec{\boldsymbol{d}}]\ldots t_n[\vec{\boldsymbol{d}}] \\ & & \text{for some } t_1[\vec{\boldsymbol{d}}]\ldots t_n[\vec{\boldsymbol{d}}] \in \Lambda^{\emptyset}_o[\mathcal{D}] \\ & \iff & \lambda\vec{d}.Mt_1[\vec{d}]\ldots t_n[\vec{d}] \neq_{\beta\eta} \lambda\vec{d}.Nt_1[\vec{d}]\ldots t_n[\vec{d}], \\ & \Rightarrow & \lambda fx.(\lambda\vec{d}.Mt_1[\vec{d}]\ldots t_n[\vec{d}])(Ofx)(O_1fx)\ldots(O_nfx) \neq_{\beta\eta} \\ & & \lambda fx.(\lambda\vec{d}.Nt_1[\vec{d}]\ldots t_n[\vec{d}])(Ofx)(O_1fx)\ldots(O_nfx) \\ & \Rightarrow & Mt_1[O\boldsymbol{F}\boldsymbol{x}, O_1\boldsymbol{F}\boldsymbol{x}, \ldots]\ldots t_n[O\boldsymbol{F}\boldsymbol{x}, O_1\boldsymbol{F}\boldsymbol{x}, \ldots] \neq_{\beta\eta} \\ & & Nt_1[O\boldsymbol{F}\boldsymbol{x}, O_1\boldsymbol{F}\boldsymbol{x}, \ldots]\ldots t_n[O\boldsymbol{F}\boldsymbol{x}, O_1\boldsymbol{F}\boldsymbol{x}, \ldots] \\ & \Rightarrow & M \not=^{\mathtt{ext}}_{\mathcal{C}} N. \end{array}$$

The converse can be proved similarly, using exercise 3.6.22 (iv). ■

3.5.28. Proposition. *Let $\mathcal{D}$ be of class 0. Then the following hold.*

(i) *$=^{\mathtt{ext}}_{\mathcal{D}}$ is a logical relation, hence $=^{\mathtt{ext}}_{\mathcal{D}}$ is $=^{\mathtt{obs}}_{\mathcal{D}}$ on $\Lambda^{\emptyset}_o[\mathcal{D}]$.*

(ii) *$\forall M, N \in \Lambda^{\emptyset}_o\,[M =_{\mathcal{D}} N \iff M =_{\mathcal{C}_0} N \iff \mathsf{I} =_{\beta\eta} \mathsf{I}]$.*

Proof. It is not hard to see that $\mathcal{D}$ is of class 0 iff $\Lambda^{\emptyset}_o[\mathcal{D}](o)$ is a singleton. Therefore for these $\mathcal{D}$ the statements hold trivially.

Harvesting the results we obtain the following.

3.5.29. Theorem. *[Statman [1980b]] Let $\mathcal{D}$ be a set of constants such that there are enough closed terms. Then we have the following.*

(i) *$=^{\mathtt{ext}}_{\mathcal{D}}$ is logical.*

(ii) *$=^{\mathtt{ext}}_{\mathcal{D}}$ is $=^{\mathtt{obs}}_{\mathcal{D}}$.*

(iii) *$\mathcal{M}_{\mathcal{D}}$ is a well defined extensional term-model.*

(iv) *$\mathcal{M}_{\mathcal{D}}$ is fully abstract and o-complete.*

Moreover, if $\mathcal{D}$ is of class i, then

(v) *$\mathcal{M}_{\mathcal{D}} \equiv \mathcal{M}_{\mathcal{C}_i}$, i.e. $\forall M, N \in \Lambda^{\emptyset}_o[M =_{\mathcal{D}} N \iff M =_{\mathcal{C}_i} N]$.*

(vi) *$\mathcal{M}_{\mathcal{D}}$ is $\nabla(\mathcal{C}_i)$-complete for pure terms.*

Proof. (i), (ii) By corollary 3.5.25 and propositions 3.5.27 and 3.5.28.

(iii), (iv) As in theorem 3.5.13 and corollary 3.5.15.

(v) By proposition 3.5.27 (ii).

(vi) By (v) and corollary 3.5.15. ■

3.5.30. Remark. So there are at most five canonical term-models that are not elemantary equivalent (plus the degenerate term-model that does not count).

Applications

3.5.31. LEMMA. *Let $A \in \mathbb{T}(\lambda^o_{\to})$ and suppose $\mathcal{M}_{\mathcal{D}}$ is A-complete for pure terms. Then for $M, N \in \Lambda^{\emptyset}_o(B)$ one has*

$$[\mathcal{M}_{\mathcal{D}} \models M = N \;\&\; M \neq_{\beta\eta} N] \;\Rightarrow\; B \not\leq_{\beta\eta} A.$$

PROOF. Suppose $B \leq_{\beta\eta} A$ via $F{:}B{\to}A$. Then

$$\begin{array}{lll} M \neq_{\beta\eta} N & \Rightarrow & FM \neq_{\beta\eta} FN \\ & \Rightarrow & \mathcal{M}_{\mathcal{D}} \not\models FM = FN, \qquad \text{because of } A\text{-completeness}, \\ & \Rightarrow & \mathcal{M}_{\mathcal{D}} \not\models M = N, \end{array}$$

a contradiction. ■

In the previous section the types A_α were introduced. The next proposition is needed to prove that they form a hierarchy.

3.5.32. PROPOSITION. *For $\alpha, \beta \leq \omega + 3$ one has*

$$\alpha \leq \beta \Leftarrow A_\alpha \leq_{\beta\eta} A_\beta.$$

PROOF. Notice that for $\alpha \leq \omega$ the cardinality of $\Lambda^{\emptyset}_o(A_\alpha)$ equals α: For example $\Lambda^{\emptyset}_o(A_2) = \{\lambda xy{:}o.x, \lambda xy{:}o.y\}$ and $\Lambda^{\emptyset}_o(A_\omega = \{\lambda f{:}1\lambda x{:}o.f^k x \mid k \in \mathbb{N}\}$. Therefore for $\alpha, \alpha' \leq \omega$ one has $A_\alpha \leq_{\beta\eta} A_{\alpha'} \;\Rightarrow\; \alpha = \alpha'$.

It remains to show that $A_{\omega+1} \not\leq_{\beta\eta} A_\omega, A_{\omega+2} \not\leq_{\beta\eta} A_{\omega+1}, A_{\omega+3} \not\leq_{\beta\eta} A_{\omega+2}$.

As to $A_{\omega+1} \not\leq_{\beta\eta} A_\omega$, consider

$$\begin{array}{rcl} M & \equiv & \lambda f, g{:}1\lambda x{:}o.f(g(f(gx))), \\ N & \equiv & \lambda f, g{:}1\lambda x{:}o.f(g(g(fx)). \end{array}$$

Then $M, N \in \Lambda^{\emptyset}_o(A_{\omega+1})$, and $M \neq_{\beta\eta} N$. Note that $\mathcal{M}_{\mathcal{C}_2}$ is A_ω-complete. It is not diffiult to show that $\mathcal{M}_{\mathcal{C}_2} \models M = N$, by analyzing the elements of $\Lambda^{\emptyset}_o[\mathcal{C}_2](1)$. Therefore, by lemma 3.5.31, the conclusion follows.

As to $A_{\omega+2} \not\leq_{\beta\eta} A_{\omega+1}$, this is proved in Dekkers [1988].

As to $A_{\omega+3} \not\leq_{\beta\eta} A_{\omega+2}$, consider

$$\begin{array}{rcl} M & \equiv & \lambda h{:}1_2\lambda x{:}o.h(hx(hxx))(hxx), \\ N & \equiv & \lambda h{:}1_2\lambda x{:}o.h(hxx)(h(hxx)x). \end{array}$$

Then $M, N \in \Lambda^{\emptyset}_o(A_{\omega+3})$, and $M \neq_{\beta\eta} N$. Note that $\mathcal{M}_{\mathcal{C}_4}$ is $A_{\omega+2}$-complete. It is not diffiult to show that $\mathcal{M}_{\mathcal{C}_4} \models M = N$, by analyzing the elements of $\Lambda^{\emptyset}_o[\mathcal{C}_4](1_2)$. Therefore, by lemma 3.5.31, the conclusion follows. ■

3.6. Exercises

3.6.1. The iterated exponential function 2_n is

$$\begin{aligned} 2_0 &= 1, \\ 2_{n+1} &= 2^{2_n}. \end{aligned}$$

One has $2_n = 2_n(1)$, according to the definition before Exercise 2.5.16. Define $s(A)$ to be the number of occurrences of atoms in the type A, i.e.

$$\begin{aligned} s(o) &= 1 \\ s(A \to B) &= s(A) + s(B). \end{aligned}$$

Write $\#X$ for the cardinality of the set X. Show the following.

(i) $2_n \leq 2_{n+p}$.

(ii) $2_{n+2}^{2_{p+1}} \leq 2_{n+p+3}$.

(iii) $2_n^{2_p} \leq 2_{n+p}$.

(iv) If $X = \{0, 1\}$, then $\forall A \in \mathbb{T}.\#(X(A)) \leq 2_{s(A)}$.

(v) For which types A do we have $=$ in (iv)?

3.6.2. Show that if $\mathcal{M}$ is a type model, then for the corresponding polynomial type model $\mathcal{M}^*$ one has $\mathrm{Th}(\mathcal{M}^*) = \mathrm{Th}(\mathcal{M})$.

3.6.3. Show that

$$A_1 \to \ldots \to A_n \to o \leq_{\beta\eta} A_{\pi 1} \to \ldots \to A_{\pi n} \to o,$$

for any prmutation $\pi \in S_n$

3.6.4. Let $A = (2 \to 2 \to o) \to 2 \to o$ and
$B = (o \to 1^2 \to o) \to 1_2 \to (o \to 1 \to o) \to o^2 \to o$. Show that

$$A \leq_{\beta\eta} B.$$

[Hint. Use the term $\lambda z{:}A \lambda u_1{:}(o \to 1^2 \to o) \lambda u_2{:}1_2 \lambda u_3{:}(o \to 2) \lambda x_1 x_2{:}o.$
$z[\lambda y_1, y_2{:}2.u_1 x_1 (\lambda w{:}o.y_1(u_2 w))(\lambda w{:}o.y_2(u_2 w))][u_3 x_2]$.]

3.6.5. Let $A = (1^2 \to o) \to o$. Show that

$$A \leq_{\beta\eta} 1_2 \to 2 \to o.$$

[Hint. Use the term
$\lambda M{:}(A \to o) \lambda p{:}1_2 \lambda F{:}2.M(\lambda f, g{:}1.F(\lambda z{:}o.p(fz)(gz)))$.]

3.6.6. (i) Show that

$$\begin{pmatrix} 2 \\ 3 \quad 4 \end{pmatrix} \leq_{\beta\eta} 1 \to 1 \to \begin{pmatrix} 2 \\ 3 \quad 3 \end{pmatrix}.$$

(ii) Show that

$$\begin{pmatrix} 2 \\ 3 \quad 3 \end{pmatrix} \leq_{\beta\eta} 1 \to 1 \to \begin{pmatrix} 2 \\ 3 \end{pmatrix}.$$

(iii) $*$ Show that

$$\begin{pmatrix} 2 & 2 \\ 3 & 2 \end{pmatrix} \leq_{\beta\eta} 1_2\to \begin{pmatrix} 2 & \\ 3 & 2 \end{pmatrix}.$$

[Hint. Use $\Phi = \lambda M\lambda p{:}1_2\lambda H_1' H_2.M$
$[\lambda f_{11}, f_{12}{:}1_2.H_1'(\lambda xy{:}o.p(f_{12}xy, H_2 f_{11})]$
$[\lambda f_{21}{:}1_3\lambda f_{22}{:}1_2.H_2 f_{21} f_{22}].$]

3.6.7. Show that $2\to o \leq_{\beta\eta} 1\to 1\to o\to o$. [Hint. Use

$$\Phi \equiv \lambda M{:}2\lambda f, g{:}1\lambda z{:}o.M(\lambda h{:}1.f(h(g(hz)))).$$

Typical elements of type 3 are $M_i \equiv \lambda F{:}2.F(\lambda x_1.F(\lambda x_2.x_i))$. Show that Φ acts injectively (modulo $\beta\boldsymbol{\eta}$) on these.]

3.6.8. Give example of $F, G \in \Lambda[C_4]$ such that $Fh_2 =_{\beta\eta} Gh_2$, but $F \neq_{\beta\eta} G$, where $h_2 \equiv \lambda z{:}o.\boldsymbol{\Phi}(\lambda g{:}1.g(gz))$.

3.6.9. (Joly [2001], Lemma 2, p. 981, based on an idea of Dana Scott) Show that any type A is reducible to

$$1_2\to 2\to o = (o\to(o\to o))\to((o\to o)\to o)\to o.$$

[Hint. We regard each closed term of type A as an untyped lambda term and then we retype all the variables as type o replacing applictions XY by $fXY(=: X\bullet Y)$ and abstractions $\lambda x.X$ by $g(\lambda x.X)(=: \lambda^\bullet x.X)$ where $f : 1_2, g : 2$. Scott thinks of f and g as a retract pair satisfying $g \circ f = \mathsf{I}$ (of course in our context they are just variables which we abstract at the end). The exercise is to define terms which 'do the retyping' and insert the f and g, and to prove that they work. For $A \in \mathbb{T}$ define terms $U_A : A\to 0$ and $V_A : 0\to A$ as follows.

$$U_o := \lambda x{:}o.x; \quad V_o := \lambda x{:}o.x.$$

$$\begin{aligned} U_{A\to B} &:= \lambda u.g(\lambda x{:}o.U_B(f(V_A x))); \\ V_{A\to B} &:= \lambda v\lambda y.V_B(fv(U_A y)) \end{aligned}$$

Let $A_i = A_{i_1}\to \ldots A_{ir_i}\to 0$ and write for a closed $M : A$

$$M = \lambda y_1 \ldots y_a.y_i(M_1 y_1 \ldots y_s)\ldots(M_n y_1 \ldots y_s),$$

with the M_i closed (this is the "Φ-nf" if the M_i are written similarly). Then

$$U_A M \twoheadrightarrow \lambda^\bullet \vec{x}.x_i(U_{B_1} M_1 \vec{x})\ldots(U_{B_n} M_n \vec{x}),$$

where $B_j = A_1\to \ldots\to A_a\to A_{ij}$, for $1 \leq j \leq n$, is the type of M_j. Show for all closed M, N by induction on the complexity of M that

$$U_A M =_{\beta\boldsymbol{\eta}} U_A N \;\Rightarrow\; M =_{\beta\boldsymbol{\eta}} N.$$

Conclude that $A \leq_{\beta\boldsymbol{\eta}} 1_2\to 2\to o$ via $\Phi \equiv \lambda bfg.U_A b$.]

3.6.10. In this exercise the combinatorics of the argument needed in the proof of 3.4.5 is analyzed. Let $(\lambda F{:}2.M) : 3$. Define M^+ to be the long $\beta\boldsymbol{\eta}$ nf of $M[F := H]$, where

$$H = (\lambda h{:}1.f(h(g(hz)))) \in \Lambda_{\rightarrow}^{\{f,g:1,z:o\}}(2).$$

Write $\mathtt{cut}_{g\rightarrow z}(P) = P[g := \mathsf{K}z]$.

(i) Show by induction on M that if $g(P) \subseteq M^+$ is maximal (i.e. $g(P)$ is not a proper subterm of a $g(P') \subseteq M^+$), then $\mathtt{cut}_{g\rightarrow z}(P)$ is a proper subterm of $\mathtt{cut}_{g\rightarrow z}(M^+)$.

(ii) Let $M \equiv F(\lambda x{:}o.N)$. Then we know

$$M^+ =_{\beta\eta} f(N^+[x := g(N^+[x := z])]).$$

Show that if $g(P) \subseteq M^+$ is maximal and

$$\mathtt{length}(\mathtt{cut}_{g\rightarrow z}(P)) + 1 = \mathtt{length}(\mathtt{cut}_{g\rightarrow z}(M^+)),$$

then $g(P) \equiv g(N^+[x := z])$ and is substituted for an occurrence of x in N^+.

(iii) Show that the occurrences of $g(P)$ in M^+ that are maximal and satisfy $\mathtt{length}(\mathtt{cut}_{g\rightarrow z}(P)) + 1 = \mathtt{length}(\mathtt{cut}_{g\rightarrow z}(M^+))$ are exactly those that were substituted for the occurrences of x in N^+.

(iv) Show that (up to $=_{\beta\eta}$) M can be reconstructed from M^+.

3.6.11. Show directly that

$$2{\rightarrow}1_2{\rightarrow}o \leq_{\beta\boldsymbol{\eta}} 1^2{\rightarrow}1_2{\rightarrow}o{\rightarrow}o,$$

via $\Phi \equiv \lambda M : L\lambda f g : 1\lambda b : 1_2\lambda x : o.M(\lambda h.f(h(g(hx))))b$.
Finish the alternative proof that $\top = 1_2{\rightarrow}o{\rightarrow}o$ satisfies $\forall A \in \mathbf{\Pi}(\lambda^o_{\rightarrow}).A \leq_{\beta\boldsymbol{\eta}} \top$, by showing in the style of the proof of Proposition 3.4.6 the easy

$$1^2{\rightarrow}1_2{\rightarrow}o{\rightarrow}o \leq_{\beta\boldsymbol{\eta}} 1_2{\rightarrow}o{\rightarrow}o.$$

3.6.12. Show directly that (without the reducibility theorem)

$$3{\rightarrow}o{\rightarrow}o \leq_{\beta\eta} 1_2{\rightarrow}o{\rightarrow}o = \top.$$

3.6.13. Show directly the following.

(i) $1_3{\rightarrow}1_2{\rightarrow}o \leq_{\beta\eta} \top$.

(ii) For any type A of rank ≤ 2 one has $A \leq_{\beta\eta} \top$.

3.6.14. Show that all elements $g \in \mathcal{M}_2(o{\rightarrow}o)$ satisfy $g^2 = g^4$. Conclude that $\mathcal{T} \not\twoheadrightarrow \mathcal{M}_2$.

3.6.15. Show that $\mathcal{M}_n \hookrightarrow \mathcal{M}_\omega$, for $n \in \omega$.

3.6.16. A model $\mathcal{M}$ is called finite iff $\mathcal{M}(A)$ is finite for all types A. Find out which of the five canonical termmodels is finite.

3.6.17. Let $\mathcal{M} = \mathcal{M}_{\min}$.

(i) Determine in $\mathcal{M}(1{\to}o{\to}o)$ which of the three Church's numerals $\mathbf{c}_0, \mathbf{c}_{10}$ and $\mathbf{c}_{100}$ are equal and which not.
(ii) Determine the elements in $\mathcal{M}(1_2{\to}o{\to}o)$.

3.6.18. Let $\mathcal{M}$ be a model and let $|\mathcal{M}_0| \leq \kappa$. By Example 3.3.24 there exists a partial surjective homomorphism $h : \mathcal{M}_\kappa{\to}\mathcal{M}$.
(i) Show that $h^{-1}(\mathcal{M}) \subseteq \mathcal{M}_\kappa$ is closed under λ-definability. [Hint. Use Example 3.3.36.]
(ii) Show that as in Example 3.3.37 one has $h^{-1}(\mathcal{M})^E = h^{-1}(\mathcal{M})$.
(iii) Show that the Gandy Hull $h^{-1}(\mathcal{M})/E$ is isomorphic to $\mathcal{M}$.
(iv) For the 5 canonical models $\mathcal{M}$ construct $h^{-1}(\mathcal{M})$ directly without reference to $\mathcal{M}$.
(v) (Plotkin) Do the same as (iii) for the free open term model.

3.6.19. Let $\mathcal{D} = \{\boldsymbol{F}{:}2, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$.
(i) Give a characterisation of the elements of $\Lambda^\emptyset_o[\mathcal{D}](1)$.
(ii) For $f, g \in \Lambda^\emptyset_o[\mathcal{D}](1)$ show that $f \neq_{\beta\eta} g \;\Rightarrow\; f \not\approx_{\mathcal{D}} g$ by applying both f, g to $\boldsymbol{F}f$ or $\boldsymbol{F}g$.

3.6.20. Let $\mathcal{D}$ be of class 2. Show that either one of the following cases holds.

$$\begin{array}{rcl} \mathcal{D} & = & \{\boldsymbol{F}{:}2, \boldsymbol{x}_1 \ldots, \boldsymbol{x}_n{:}o\},\ n \geq 0; \\ \mathcal{D} & = & \{\boldsymbol{f}{:}1, \boldsymbol{x}_1 \ldots, \boldsymbol{x}_n{:}o\},\ n \geq 1. \end{array}$$

3.6.21. Prove the following.

$$\begin{array}{lll} 1_2{\to}o{\to}o & \leq_{\beta\eta} & ((1_2{\to}o){\to}o){\to}o{\to}o, \text{ via} \\ & & \lambda m\lambda F{:}((1_2{\to}o){\to}o)\lambda x{:}o.F(\lambda h{:}1_2.mhx) \text{ or via} \\ & & \lambda m\lambda F{:}((1_2{\to}o){\to}o)\lambda x{:}o.m(\lambda pq{:}o.F(\lambda h{:}1_2.hpq))x. \\ 1_2{\to}o{\to}o & \leq_{\beta\eta} & (1{\to}1{\to}o){\to}o{\to}o \\ & & \text{via } \lambda mHx.m(\lambda ab.H(\mathsf{K}a)(\mathsf{K}b))x. \end{array}$$

3.6.22. (i) Prove $2{\to}o^n{\to}o \leq_{\beta\eta} 1{\to}o{\to}o$. [Construct first three terms O_1, O_2, O_3 such that if $M, N \in \Lambda^\emptyset(2{\to}o^n{\to}o)$ and $M \neq_{\beta\eta} N$, then one of the inequalities $M(O_i fx) \neq_{\beta\eta} N(O_i fx)$ holds. Complete the construction by taking as reducing term

$$\Phi \equiv \lambda mfx.P_3(m(O_1fx))(m(O_2fx))(m(O_3fx)),$$

where P_3 is a polynomially definable coding of $\mathbb{N}^3{\to}\mathbb{N}$.]
(ii) Prove $1{\to}o{\to}o \leq_{\beta\eta} 2{\to}o^n{\to}o$ using $\Phi \equiv \lambda mFx_1 \ldots x_n.m(F\mathsf{I})$.
(iii) Let $M, N \in \Lambda^\emptyset(2{\to}o^n{\to}o)$. Show, using (i), that if $M \neq_{\beta\eta} N$, then for some $O \in \Lambda^\emptyset$ one has

$$\lambda fx.M(Ofx)(O_1fx)\ldots(O_nfx) \neq_{\beta\eta} \lambda fx.N(Ofx)(O_1fx)\ldots(O_nfx).$$

(iv) Let $M, N \in \Lambda^\emptyset(1{\to}o{\to}o)$. Show, using (ii), that if $M \neq_{\beta\eta} N$, then for some $O \in \Lambda^\emptyset$ one has

$$\lambda Fx_1 \ldots x_n.M(OFx_1 \ldots x_n) \neq_{\beta\eta} \lambda Fx_1 \ldots x_n.N(OFx_1 \ldots x_n).$$

3.6.23. (i) Using exercise 3.6.22 prove that for $p > 0, q \geq 0$ one has

$$1{\to}o{\to}o{\sim}_\nabla(1_p{\to}o){\to}o^q{\to}o.$$

(ii) Prove that for $q > 0$ one has $1{\to}o{\to}o{\sim}_\nabla 1{\to}o^q{\to}o$.

3.6.24. Show that for all $A, B \in \mathbb{T}_2$ one has $A \sim_{h^+} B$.

3.6.25. Show that for all $A, B \notin \mathbb{T}_2$ one has $A \sim_h B$.

3.6.26. (i) Show that lemma 3.5.23 is false for $\mathcal{D} = \{\mathbf{g}^1, \mathbf{d}^o\}$ and $\mathcal{C}_2 = \{\mathbf{f}^2, \mathbf{c}^o\}$. [Hint. Consider $P \equiv \lambda fgc.f(g(fc))$ and $Q \equiv \lambda fgc.f(f(gc))$.]

(ii) Show that that lemma is also false for $\mathcal{D} = \{\mathbf{d}^o\}$ and $\mathcal{C} = \{\mathbf{c}^o\}$.

3.6.27. Show that if $M \in \Lambda^\emptyset(3{\to}o{\to}o)$, then M is of the form

$$\begin{array}{ll} \lambda\Phi{:}3\lambda c\ o. & \Phi(\lambda f_1{:}1.w_{f_1} \\ & \Phi(\lambda f_1, f_2{:}1.w_{f_1,f_2} \\ & \ldots \\ & \Phi(\lambda f_1, \ldots, f_n{:}1.w_{f_1,\ldots,f_n} \\ & c)..)), \end{array}$$

where $w_{f_1,\ldots,f_n}$ stands for a "word" over the alphabet $\Sigma = \{f_1, \ldots, f_n\}$ in the sense that e.g. $f_1f_2f_1$ is to be interpreted as $f_1 \circ f_2 \circ f_1$. [Hint. See the proof of lemma 3.5.9.]

3.6.28. Let A be an inhabited small type of rank > 3. Show that

$$3{\to}o{\to}o \leq_m A.$$

[Hint. For small B of rank ≥ 2 one has $B \equiv B_1{\to}\ldots B_b{\to}o$ with $B_i \equiv B_{i_1}{\to}o$ for all i and $\mathtt{rank}(B_{i_{01}}) = \mathtt{rank}(B) - 2$ for some i_o. Define for such B the term

$$X^B \in \Lambda^\emptyset[F^2](B),$$

where F^2 is a variable of type 2.

$$\begin{array}{lll} X^B & \equiv & \lambda x_1 \ldots x_b.F^2x_{i_0}, \qquad \text{if } \mathtt{rank}(B) = 2; \\ & \equiv & \lambda x_1 \ldots x_b.F^2(\lambda y{:}0.x_{i_0}(\lambda y_1 \ldots y_k.y)), \qquad \text{if } \mathtt{rank}(B) = 3 \text{ and} \\ & & \qquad \text{where } B_{i_0} \text{ having} \\ & & \qquad \text{rank 1 is } o^k{\to}o; \\ & \equiv & \lambda x_1 \ldots x_b.x_{i_0}X^{B_{i_{01}}}, \qquad \text{if } \mathtt{rank}(B) > 3. \end{array}$$

(Here $X^{B_{i_{01}}}$ is well-defined since $B_{i_{01}}$ is also small.) As A is inhabited, take $\lambda x_1 \ldots x_b.N \in \Lambda^\emptyset(A)$. Define $\Psi : (3{\to}o{\to}o){\to}A$ by

$$\Psi(M) = \lambda x_1 \ldots x_b.M(\lambda F^2.x_iX^{A_{i1}})N,$$

where i is such that A_{i1} has rank ≥ 2. Show that Ψ works.]

3.6.29. Consider

1. $\lambda f{:}1\lambda x{:}o.fx = \lambda f{:}1\lambda x{:}o.f(fx)$;

2. $\lambda f,g{:}1\lambda x{:}o.f(g(g(fx))) = \lambda f,g{:}1\lambda x{:}o.f(g(f(gx)))$;
3. $\lambda h{:}1_2\lambda x{:}o.h(hx(hxx))(hxx) = \lambda h{:}1_2\lambda x{:}o.h(hxx)(h(hxx)x)$.

(i) Show that 1 holds in $\mathcal{M}_{\mathcal{C}_1}$, but not in $\mathcal{M}_{\mathcal{C}_2}$.
(ii) Show that 2 holds in $\mathcal{M}_{\mathcal{C}_2}$, but not in $\mathcal{M}_{\mathcal{C}_3}$.
(iii) Show that 3 holds in $\mathcal{M}_{\mathcal{C}_3}$ and $\mathcal{M}_{\mathcal{C}_4}$, but not in $\mathcal{M}_{\mathcal{C}_5}$.

3.6.30. Construct as much as possible (6 at most) pure closed terms in order to show that the five canonical theories are maximally different. I.e. we want terms $M_1,\ldots,M_6$ such that in $\mathtt{Th}(\mathcal{M}_{\mathcal{C}_5})$ the $M_1,\ldots,M_6$ are mutually different; $M_1 = M_2$ in $\mathtt{Th}(\mathcal{M}_{\mathcal{C}_4})$, but different from $M_3,\ldots,M_6$; $M_2 = M_3$ in $\mathtt{Th}(\mathcal{M}_{\mathcal{C}_3})$, but different from $M_4,\ldots,M_6$; $M_3 = M_4$ in $\mathtt{Th}(\mathcal{M}_{\mathcal{C}_2})$, but different from M_5, M_6; $M_4 = M_5$ in $\mathtt{Th}(\mathcal{M}_{\mathcal{C}_1})$, but different from M_6. [Hint. Use the previous exercise and a polynomially defined pairing operator. A complicating factor is that it is an open question whether the theories of $\mathcal{M}_{\mathcal{C}_3}$ and $\mathcal{M}_{\mathcal{C}_4}$ are different.]

3.6.31. (Finite generation, Joly [2002]) Let $A \in \mathbb{T}(\lambda_{\rightarrow})$. Then A is said to be *finitely generated* if there exist types $A_1,\ldots,A_t$ and terms $M_1 : A_1,\ldots,A_t : M_t$ such that for any $M : A$, M is $\beta\boldsymbol{\eta}$ convertible to an applicative combination of $M_1,\ldots,M_t$.
Example. $\mathsf{Nat} = 1{\rightarrow}o{\rightarrow}o$ is finitely generated by $\mathbf{c}_0 \equiv (\lambda fx.x) : \mathsf{Nat}$ and $S \equiv (\lambda nfx.f(nfx)) : (\mathsf{Nat}{\rightarrow}\mathsf{Nat})$.
A slantwise enumerates a type B if there exists a type substitution @ and $F : @A{\rightarrow}B$ such that for each $N : B$ there exists $M : A$ such that $F@M =_{\beta\boldsymbol{\eta}} N$ (F is *surjective*).
A type A is said to be *poor* if every beta-eta normal form of type A can be written with the same finite number of bound variables. Otherwise A is said to be *rich*.
Example. Let $A = (1{\rightarrow}o){\rightarrow}o{\rightarrow}o$ is poor. A typical $\beta\boldsymbol{\eta}$-nf of type A has the shape $\lambda F\lambda x(F(\lambda x(\ldots(F(\lambda y(F(\lambda y\ldots x\ldots)))..))))$. One allows the term to violate the variable convention (to have different bound variables). The monster type $3{\rightarrow}1$ is rich.
The goal of this exercise is to prove that the following are equivalent.

1. A slantwise enumerates the monster type M
2. The lambda definability problem for A is undecidable.
3. A is not finitely generated
4. A is rich.

However, we will not ask the reader to prove (4) $\Rightarrow$ (1) since this involves more knowledge of and practice with slantwise enumerations than one can get from this book. For that proof we refer the reader to Joly's paper. We have already shown that the lambda definability problem for the monster M is undecidable. In addition, we make the following steps.

(i) Show A is rich iff A has rank >3 or A is large of rank 3 (for A inhabited; especially for $\Rightarrow$). Use this to show

$$(2) \Rightarrow (3) \text{ and } (3) \Rightarrow (4).$$

(ii) (Alternative to show (3) ⇒ (4).) Suppose that every closed term of type A beta eta converts to a special one built up from a fixed finite set of variables.Show that it suffices to bound the length of the lambda prefix of any subterm of such a special term in order to conclude finite generation. Suppose that we consider only terms X built up only from the variables $v_1{:}A_1, \ldots, v_m{:}A_m$ both free and bound .We shall transform X using a fixed set of new variables. First we assume the set of Ai is closed under subtype. (a) Show that we can assume that X is fully expanded. For example, if X has the form

$$\lambda x_1 \ldots x_t.(\lambda x.X_0)X_1 \ldots X_s$$

then $(\lambda x.X_0)X_1 \ldots X_s$ has one of the A_i as a type (just normalize and consider the type of the head variable). Thus we can eta expand

$$\lambda x_1 \ldots x_t.(\lambda x.X_0)X_1 \ldots X_s$$

and repeat recursively. We need only double the set of variable to do this. We do this keeping the same notation. (b) Thus given

$$X \;=\; \lambda x_1 \ldots x_t.(\lambda x.X_0)X_1 \ldots X_s$$

we have $X_0 = \lambda y_1 \ldots y_r.Y$, where $Y : o$. Now if $r{>}m$, each multiple occurrence of vi in the prefix $\lambda y_1 \ldots y_r$ is dummy and those that occur in the initial segment $\lambda y_1 \ldots y_s$ can be removed with the corresponding X_j. The remaining variables will be labelled $z_1, \ldots, z_k$. The remaining X_j will be labelled $Z_1, \ldots, Z_l$. Note that $r - s + t < m + 1$. Thus

$$X = \lambda x_1 \ldots x_t.(\lambda z_1 \ldots z_k Y)Z_1 \ldots Z_l,$$

where $k < 2m + 1$. We can now repeat this analysis recursively on Y, and $Z_1, \ldots, Z_l$ observing that the types of these terms must be among the A_i. We have bounded the length of a prefix.

(iii) As to (1) ⇒ (2). We have already shown that the lambda definability problem for the monster M is undecidable. Suppose (1) and ¬(2) towards a contradiction. Fix a type B and let $B(n)$ be the cardinality of B in $P(n)$. Show that for any closed terms $M, N : C$

$$P(B(n)) \models M = N \;\Rightarrow\; P(n) \models [o := B]M = [o := B]N.$$

Conclude from this that lambda definability for M is decidable, which is not the case.

Chapter 4

Unification and Matching

4.1. Undecidability of lambda definability

The finite standard models

Recall that the full type structure over a set X, notation $\mathcal{M}_X$, is defined in Definition 2.4.18 as follows.

$$\begin{array}{rcl} X(o) & = & X, \\ X(A{\to}B) & = & X(B)^{X(A)}; \\ \mathcal{M}_X & = & \{X(A)\}_{A\in\mathbb{T}}. \end{array}$$

Remark that if X is finite then all the $X(A)$ are finite. In that case we can represent each elemement of $\mathcal{M}_X$ by a finite piece of data and hence (through Gödel numbering) by a natural number. For instance for $X = \{0, 1\}$ we can represent the four elements of $X(o{\to}o)$ as follows.

0	0
1	0

0	1
1	1

0	0
1	1

0	1
1	0

Any element of the model can be expressed in a similar way, for instance the following table represents an element of $X((o \to o) \to o)$.

0	0	0
1	0	
0	1	0
1	1	
0	0	0
1	1	
0	1	1
1	0	

We know that $\mathsf{I} \equiv \lambda x.x$ is the only closed $\beta\eta$-nf of type $o \to o$. It is easy to prove from this that identity is the only function of $X(o \to o)$ that is denoted by a closed term.

4.1.1. DEFINITION. Let $\mathcal{M}$ be a type structure and let $d \in \mathcal{M}(A)$. Then d is called λ-*definable* iff $\exists M \in \Lambda^{\emptyset}(A).d = [\![M]\!]^{\mathcal{M}}$.

The main result in this section is the undecidability of λ-definability in $\mathcal{M}_X$, for X of cardinality >6. This means that there is no algorithm deciding whether a table describes a λ-definable element in this model. This result is due to Loader [2001b], and was already proved by him in 1993.

The proof proceeds by reducing the two-letter word rewriting problem, a well-known undecidable problem, to the λ-definability problem. It follows that if the λ-definability problem were decidable, then this also would be the case for the two-letter word rewriting problem, *quod non*.

4.1.2. DEFINITION (Word rewriting problem). Let $\Sigma = \{A, B\}$ be a two letter alphabet.

(i) A *word* is a finite sequence of letters $w = w_1 \dots w_n$ with $w_i \in \Sigma$. The set of words over Σ is denoted by Σ^*.

(ii) If $w = w_1 \dots w_n$, then $\mathtt{lth}(w) = n$ is called the length of w. If $\mathtt{lth}w = 0$, then w is called the *empty word* and is denoted by ϵ.

(iii) A *rewrite rule* is a pair of non empty words v, w denoted as $v \hookrightarrow w$.

(iv) Given a word u and a finite set $\mathcal{R} = \{R_1, \dots, R_r\}$ of rewrite rules $R_i = v_i \hookrightarrow w_i$. Then a *derivation* of a word s is a finite sequence of words starting by u finishing by s and such that each word is obtained from the previous by replacing a subword v_i by w_i for some rule $v_i \hookrightarrow w_i \in \mathcal{R}$.

(v) A word s is said to be *derivable* from u if it has a derivation. In this case we write $u \vdash_{\mathcal{R}} s$.

4.1.3. EXAMPLE. Consider the word AB and the rule $AB \hookrightarrow AABB$. Then $AB \vdash AAABBB$, but $AB \nvdash AAB$.

We will need the following well known result, see e.g. Post [1947].

4.1.4. THEOREM. *There is a word $u_0 \in \Sigma^*$ and a finite set of rewrite rules $v_1 \hookrightarrow w_1, \dots, v_r \hookrightarrow w_r$ such that $\{u \in \Sigma^* \mid u_0 \vdash u\}$ is undecidable.*

4.1.5. DEFINITION. Given the alphabet $\Sigma = \{A, B\}$, define the set

$$X = X_\Sigma = \{*, A, B, L, R, Y, N\}.$$

The objects L and R are suggested to be read *left* and *right* and Y and N *yes* and *no*. We will consider the standard model $\mathcal{M} = \mathcal{M}_X$ built over the set X.

4.1.6. DEFINITION (Word encoding). Remember $1_n = o^n \to o$ and $MN^{\sim n} \equiv MN \dots N$, with n times the same term N. Let $w = (w_1 \dots w_n) \in \Sigma^*$ be a non empty word of length n.

(i) The word w is *encoded* as the object $\underline{w} \in \mathcal{M}(1_n)$ defined as follows.

$$\begin{aligned} \underline{w} *^{\sim(i-1)} w_i *^{\sim(n-i)} &= Y; \\ \underline{w} *^{\sim(i-1)} LR *^{\sim(n-i-1)} &= Y; \\ \underline{w} x_1 \dots x_n &= N, \qquad \text{otherwise.} \end{aligned}$$

(ii) The word w is *weakly encoded* by an object $h \in \mathcal{M}(1_n)$ iff

$$\begin{aligned} h *^{\sim(i-1)} w_i *^{\sim(n-i)} &= Y; \\ h *^{\sim(i-1)} LR *^{\sim(n-i-1)} &= Y. \end{aligned}$$

4.1.7. Definition. (Encoding of a rule) In order to define the encoding of a rule we use the notation $(a_1 \dots a_k \mapsto Y)$ to denote the element $h \in \mathcal{M}(1_k)$ defined by

$$\begin{aligned} h(a_1 \dots a_k) &= Y; \\ h(x_1 \dots x_k) &= N, \qquad \text{otherwise.} \end{aligned}$$

Now a rule $v \hookrightarrow w$ where $\mathtt{lth}v = m$ and $\mathtt{lth}w = n$ is encoded as the object $\underline{v \hookrightarrow w} \in \mathcal{M}(1_m \to 1_n)$ defined as follows.

$$\begin{aligned} \underline{v \hookrightarrow w}(\underline{v}) &= \underline{w}; \\ \underline{v \hookrightarrow w}(*^{\sim m} \mapsto Y) &= (*^{\sim n} \mapsto Y); \\ \underline{v \hookrightarrow w}(R*^{\sim(m-1)} \mapsto Y) &= (R*^{\sim(n-1)} \mapsto Y); \\ \underline{v \hookrightarrow w}(*^{\sim(m-1)} L \mapsto Y) &= (*^{\sim(n-1)} L \mapsto Y); \\ \underline{v \hookrightarrow w}(h) &= \lambda\!\!\lambda x_1 \dots x_n.N, \qquad \text{otherwise.} \end{aligned}$$

As usual we identify a term $M \in \Lambda(A)$ with its denotation $[\![M]\!] \in X(A)$.

4.1.8. Lemma. *Let s, u be two words over Σ and let $v \hookrightarrow w$ be a rule. Then $svu \vdash swu$ and*

$$\underline{swu}\,\vec{s}\,\vec{w}\,\vec{u} = \underline{v \hookrightarrow w}\,(\lambda\vec{v}.\underline{svu}\,\vec{s}\,\vec{v}\,\vec{u}\,)\vec{w}, \tag{1}$$

where the $\vec{s}, \vec{u}, \vec{v}, \vec{w}$ are sequences of elements in X with lengths equal the lengths of the words s, u, v, w, say p, q, m, n, respectively.

Proof. The RHS of (1) is obviously either Y or N. Now RHS$= Y$ iff one of the following holds

- $\lambda\vec{v}.\underline{svu}\,\vec{s}\,\vec{v}\,\vec{u} = \underline{v}$ and $\vec{w} = *^{\sim(i-1)} w_i *^{\sim(n-i)}$
- $\lambda\vec{v}.\underline{svu}\,\vec{s}\,\vec{v}\,\vec{u} = \underline{v}$ and $\vec{w} = *^{\sim(i-1)} LR*^{\sim(n-i-1)}$
- $\lambda\vec{v}.\underline{svu}\,\vec{s}\,\vec{v}\,\vec{u} = (*^{\sim m} \mapsto Y)$ and $\vec{w} = *^{\sim n}$
- $\lambda\vec{v}.\underline{svu}\,\vec{s}\,\vec{v}\,\vec{u} = (R*^{\sim(m-1)} \mapsto Y)$ and $\vec{w} = R*^{\sim(n-1)}$
- $\lambda\vec{v}.\underline{svu}\,\vec{s}\,\vec{v}\,\vec{u} = (*^{\sim(m-1)} L \mapsto Y)$ and $\vec{w} = *^{\sim(n-1)} L$

iff one of the following holds

- $\vec{s} = *^{\sim p}, \vec{u} = *^{\sim q}$ and $\vec{w} = *^{\sim(i-1)} w_i *^{\sim(n-i)}$
- $\vec{s} = *^{\sim p}, \vec{u} = *^{\sim q}$ and $\vec{w} = *^{\sim(i-1)} LR*^{\sim(n-i-1)}$
- $\vec{s} = *^{\sim(i-1)} s_i *^{\sim(p-i)}, \vec{u} = *^{\sim q}$ and $\vec{w} = *^{\sim n}$

- $\vec{s} = *^{\sim(i-1)}LR*^{\sim(p-i-1)}, \vec{u} = *^{\sim q}$ and $\vec{w} = *^{\sim n}$
- $\vec{s} = *^{\sim p}, \vec{u} = *^{\sim(i-1)}u_i*^{\sim(q-i)}$ and $\vec{w} = *^{\sim n}$
- $\vec{s} = *^{\sim p}, \vec{u} = *^{\sim(i-1)}LR*^{\sim(q-i-1)}$ and $\vec{w} = *^{\sim n}$
- $\vec{s} = *^{\sim p}, \vec{u} = R*^{\sim(q-1)}$ and and $\vec{w} = *^{\sim(n-1)}L$
- $\vec{s} = *^{\sim(p-1)}L, \vec{u} = *^{\sim q}$ and $\vec{w} = R*^{\sim(n-1)}$

iff one of the following holds

- $\vec{s}\,\vec{w}\,\vec{u} = *^{\sim(i-1)}a_i*^{\sim(p+n+q-i)}$ and a_i is the i-th letter of swu
- $\vec{s}\,\vec{w}\,\vec{u} = * \ldots * LR * \ldots *$

iff $\underline{swu}\,\vec{s}\,\vec{w}\,\vec{u} = Y$. ■

4.1.9. Proposition. *Let $\mathcal{R} = \{R_1, \ldots, R_r\}$ be a set of rules. Then*

$$s \vdash_{\mathcal{R}} u \;\Rightarrow\; \exists F \in \Lambda^{\emptyset}\ \underline{u} = F\underline{s}\ \underline{R}_1 \ldots \underline{R}_r.$$

In other words, (the code of) a word s that can be produced from u and some rules is definable from the (codes) of u and the rules.

Proof. By induction on the length of the derivation of u, using the previous lemma. ■

We now want to prove the converse of this result. We shall prove a stronger result, namely that if a word has a definable weak encoding then it is derivable.

4.1.10. Convention. For the rest of this subsection we consider a fixed word W and set of rewrite rules $\mathcal{R} = \{R_1, \ldots, R_k\}$ with $R_i = V_i \hookrightarrow W_i$. Moreover we let $w, r_1, \ldots, r_k$ be variables of the thypes of $\underline{W}, \underline{R}_1, \ldots, \underline{R}_k$ respectively. Finally ρ is a valuation such that $\rho(w) = \underline{W}$, $\rho(r_i) = \underline{R}_i$ and $\rho(x^o) = *$ for all variables of type o.

The first lemma classifies the long normal terms whose free variables are among W, $F_1, \ldots, F_r$ and that denote a word weak encoding.

4.1.11. Lemma. *Let M be a long normal form with $\mathrm{FV}(M) \subseteq \{w, r_1, \ldots, r_k\}$. Suppose $[\![M]\!]_\rho = \underline{V}$ for some word $V \in \Sigma^*$. Then M has one of the two following forms*

$$\begin{aligned} M &\equiv \lambda\vec{x}.w\vec{x}_1, \\ M &\equiv \lambda\vec{x}.r_i(\lambda\vec{y}.N)\vec{x}_1, \end{aligned}$$

where $\vec{x}, \vec{x}_1, \vec{y}$ are type o variables and the $\vec{x}_1$ are distinct elements of the $\vec{x}$.

PROOF. Since $[\![M]\!]_\rho$ is a weak encoding for V, the term M is of type 1_n and hence has a long normal form $M = \lambda\vec{x}.P$, with P of type o. The head variable of P is either w, some r_i or a bound variable x_i. It cannot be a bound variable, because then the term M would have the form

$$M = \lambda\vec{x}.x_i,$$

which does not denote a word weak encoding.

If the head variable of P is w then

$$M = \lambda\vec{x}.w\vec{P}.$$

The terms $\vec{P}$ must all be among the $\vec{x}$. This is so because otherwise some P_j would have one of the $w, \vec{r}$ as head variable; for all valuations this term P_j would denote Y or N, the term $w\vec{P}$ would then denote N and consequently M would not denote a weak word encoding. Moreover these variables must be distinct, as otherwise M would not denote a word weak encoding.

If the head variable of M is some r_i then

$$M = \lambda\vec{x}.r_i(\lambda\vec{y}.N)\vec{P}.$$

By the same reasoning as before it follows that the terms $\vec{P}$ must all be among $\vec{x}$ and different. ■

In the next four lemmas, we focus on the terms of the form

$$M = \lambda\vec{x}.r_i(\lambda\vec{y}.N)\vec{x}_1.$$

We prove that if such a term denotes a weak word encoding, then

- the variables $\vec{x}_1$ do not occur in $\lambda\vec{y}.N$,
- $[\![\lambda\vec{y}.N]\!]_\rho = \underline{v}_i$.
- and none of the variables $\vec{x}_1$ is the variable x_k.

4.1.12. LEMMA. *Let t be a long normal term of type o that is not a variable and whose free variables are among W, $F_1, \dots, F_r$ and $x_1, \dots, x_k$ of type o. If x_1 is free in t and there is a valuation φ such that $\varphi(x_1) = A$ or $\varphi(x_1) = B$ and $|t|_\varphi = Y$ then φ takes the value $*$ for all other variables of type o free in t.*

PROOF. By induction over the structure of t.

- If the head variable of t is W then

$$t = (Wt_1 \dots t_n)$$

The terms $t_1, \dots, t_n$ must all be variables otherwise, some t_j would have $W, F_1, \dots$ or F_r as head variable, $|t_j|_\varphi$ would be Y or N and the term $|t|_\varphi$ would then denote N. The variable x_1 is among these variables and if some other variable free in this term were not associated to a $*$, it would not denote Y.

- If the head variable of t is some F_i then

$$t = (F_i(\lambda\vec{w}.t')\vec{t}$$

As above, the terms $\vec{t}$ must all be variables.

If x_1 is equal to some t_j then $|\lambda\vec{w}.t'|_\varphi$ is the word v_i, t' is not a variable and all the other variables in $\vec{t}$ denote $*$. Let l be the first letter of v_i. We have $|\lambda\vec{w}.t'|_\varphi l * \ldots * = Y$ and hence

$$|t'|_{\varphi+\langle w_1,l\rangle,\langle w_2,*\rangle,\ldots,\langle w_m,*\rangle} = Y$$

hence by induction hypothesis $\varphi + \langle w_1, l\rangle, \langle w_2, *\rangle, \ldots, \langle w_m, *\rangle$ takes the value $*$ on all free variables of t' but w_1. Hence φ takes the value $*$ on all free variables of $\lambda\vec{w}.t'$. Therefore φ takes the value $*$ on all free variables of t except x_1.

If x_1 is not among $\vec{t}$ then it is free in $\lambda\vec{w}.t'$. Since $|(F_i(\lambda\vec{w}.t')\vec{t})|_\varphi = Y$ it follows that $|\lambda\vec{w}.t'|_\varphi$ is not the constant function equal to N hence there are objects $a_1, \ldots, a_m$ such that $|\lambda\vec{w}.t'|_\varphi(a_1)\ldots(a_m) = Y$. Therefore

$$|t'|_{\varphi+\langle w_1,a_1\rangle,\ldots,\langle w_m,a_m\rangle} = Y$$

and by induction hypothesis $\varphi + \langle w_1, a_1\rangle, \ldots, \langle w_m, a_m\rangle$ takes the value $*$ on all the variables free in t' but x_1. So φ takes the value $*$ on all the variables free in $\lambda\vec{w}t'$ but x_1. Moreover $a_1 = \ldots = a_m = *$, and thus $|\lambda\vec{w}.t'|_\varphi * \ldots * = Y$. Therefore the function $|\lambda\vec{w}.t'|_\varphi$ can only be the function mapping $*\ldots*$ to Y and the other values to N. Hence $|F_i(\lambda\vec{w}.t')|_\varphi$ is the function mapping $*\ldots*$ to Y and the other values to N and φ takes the value $*$ on $\vec{t}$. Therefore φ takes the value $*$ on all free variables of t except x_1. ■

4.1.13. Lemma. *If the term $t = \lambda\vec{x}(F_i(\lambda\vec{w}t')\vec{y})$ denotes a word weak encoding, then the variables $\vec{y}$ do not occur free in $\lambda\vec{w}.t'$ and $|\lambda\vec{w}.t'|_{\varphi_0}$ is the encoding of the word v_i.*

Proof. Consider a variable y_j. This variable is some $x_{j'}$. Let l be the j'^{th} letter of the word w', we have

$$|t| *^{\sim(j'-1)} l*^{\sim(k-j')} = Y$$

Let $\varphi = \varphi_0 + \langle x_{j'}, l\rangle$. We have

$$f_i(|\lambda\vec{w}.t'|_\varphi) *^{\sim(j-1)} l*^{\sim(m-j)} = Y$$

Hence $|\lambda\vec{w}.t'|_\varphi$ is the encoding of the word v_i. Let l' be the first letter of this word, we have

$$|\lambda\vec{w}.t'|_\varphi(l') * \ldots * = Y$$

and hence

$$|t'|_{\varphi+\langle w_1,l'\rangle,\langle w_2,*\rangle,\ldots,\langle w_m,*\rangle} = Y$$

By lemma 18.2, $\varphi + \langle w_1, l' \rangle, \langle w_2, * \rangle, \ldots, \langle w_m, * \rangle$ takes the value $*$ on all variables free in t' except w_1. Hence y_j is not free in t' nor in $\lambda \vec{w}.t'$.

At last $|\lambda \vec{w}.t'|_\varphi$ is the encoding of v_i and y_j does not occur in it. Thus $|\lambda \vec{w}.t'|_{\varphi_0}$ is the encoding of v_i. ■

4.1.14. Lemma. *Let t be a term of type o that is not a variable and whose free variables are among W, F_1, ..., F_r and $x_1, \ldots, x_k$ of type o. Then there is a variable z such that for all valuations $\varphi(z) = L$ implies $|t|_\varphi = N$ or for all valuations $\varphi(z) = A$ or $\varphi(z) = B$ implies $|t|_\varphi = N$.*

Proof. By induction over the structure of t.

- If the head variable of t is W then $t = (W\vec{t})$ the terms $\vec{t} = t_1, \ldots, t_n$ must be variables and we take $z = t_n$. If $\varphi(z) = L$ then $|t|_\varphi = N$.

- If the head variable of t is F_i then $t = (F_i(\lambda \vec{w}.t')\vec{t})$. By induction hypothesis, there is a variable z' free in t' such that for all valuations $\varphi(z') = L$ implies $|t|_\varphi = N$ or for all valuations $\varphi(z') = A$ or $\varphi(z') = B$ implies $|t|_\varphi = N$.

  If the variable z' is not among $w_1, \ldots, w_n$ we take $z = z'$. Either for all valuations such that $\varphi(z) = L$, $|\lambda \vec{w}.t'|_\varphi$ is the constant function equal to N and thus $|t|_\varphi = N$, or for all valuations such that $\varphi(z) = A$ or $\varphi(z) = B$, $|\lambda \vec{w}.t'|_\varphi$ is the constant function equal to N and thus $|t|_\varphi = N$.

  If the variable $z' = w_j$ $(j \leq m - 1)$ then for all valuations $|\lambda \vec{w}.t'|_\varphi$ is a function taking the value N when applied to any sequence of arguments whose j^{th} element is L or when applied to any sequence of arguments whose j^{th} element is A or B. For all valuations, $|\lambda \vec{w}.t'|_\varphi$ is not the encoding of the word v_i and hence $|F_i(\lambda \vec{w}.t')|_\varphi$ is either the function mapping $* \ldots *$ to Y and other arguments to N, the function mapping $R * \ldots *$ to Y and other arguments to N, the function mapping $* \ldots * L$ to Y and other arguments to N or the function mapping all arguments to N. We take $z = t_n$ and for all valuations such that $\varphi(z) = A$ or $\varphi(z) = B$ we have $|t|_\varphi = N$.

  At last if $z' = w_m$ then for all valuations $|\lambda \vec{w}.t'|_\varphi$ is a function taking the value N when applied to any sequence of arguments whose m^{th} element is L or for all valuations $|\lambda \vec{w}.t'|_\varphi$ is a function taking the value N when applied to any sequence of arguments whose m^{th} element is A or B. In the first case, for all valuations, $|\lambda \vec{w}.t'|_\varphi$ is a not the function mapping $* \ldots * L$ to Y and other arguments to N. Hence $|F_i(\lambda \vec{w}.t')|_\varphi$ is either w_i or the function mapping $* \ldots *$ to Y and other arguments to N the function mapping $R * \ldots *$ to Y and other arguments to N or the function mapping all arguments to N. We take $z = t_n$ and for all valuations such that $\varphi(z) = A$ or $\varphi(z) = B$ we have $|t|_\varphi = N$.

  In the second case, for all valuations, $|\lambda \vec{w}.t'|_\varphi$ is a not the encoding of the word v_i. Hence $|F_i(\lambda \vec{w}.t')|_\varphi$ is either the function mapping $* \ldots *$ to Y and other arguments to N the function mapping $R * \ldots *$ to Y and other

arguments to N, the function mapping $*\ldots*L$ to Y and other arguments to N or the function mapping all arguments to N. We take $z = t_n$ and for all valuations such that $\varphi(z) = L$ we have $|t|_\varphi = N$. ■

4.1.15. Lemma. *If the term $t = \lambda\vec{x}.(F_i(\lambda w_1 \ldots w_m t')\vec{y})$ denotes a word weak encoding, then none of the variables $\vec{y}$ is the variable x_k.*

Proof. By the lemma 4.1.14, we know that there is a variable z such that either for all valuations such that $\varphi(z) = L$ we have

$$|(F_i(\lambda\vec{w}.t')\vec{y})|_\varphi = N$$

or for all valuations such that $\varphi(z) = A$ or $\varphi(z) = B$ we have

$$|(F_i(\lambda\vec{w}.t')\vec{y})|_\varphi = N.$$

Since t denotes a word weak encoding, the only solution is that $z = x_k$ and for all valuations such that $\varphi(x_k) = L$ we have

$$|(F_i(\lambda\vec{w}.t')\vec{y})|_\varphi = N.$$

Then, if y_j were equal to x_k and y_{j+1} to some $x_{j'}$ the object

$$|(F_i(\lambda\vec{w}.t')\vec{y})|_{\varphi_0+\langle x_k,L\rangle,\langle x_{j'},R\rangle}$$

wrould be equal to $f_i(|\lambda\vec{w}.t'|_{\varphi_0})*\ldots*LR*\ldots*$ and, as $|\lambda\vec{w}.t'|_{\varphi_0}$ is the encoding of the word v_i, also to Y, a contradiction.

We are now ready to conclude the proof.

4.1.16. Proposition. *If there is a long normal term t whose free variables are among W, $F_1, \ldots, F_r$ that denotes a word weak encoding w', then w' is derivable.*

Proof. Case $t = \lambda\vec{x}.(W\vec{y})$. Then, as t denotes a word weak encoding, it depends on all its arguments and thus all the variables $x_1, \ldots, x_k$ are among $\vec{y}$. Since $\vec{y}$ are distinct, $\vec{y}$ is a permutation of $x_1, \ldots, x_k$. As t denotes a word weak encoding, $|t| * \ldots * LR * \ldots * = Y$. Hence this permuation is the identity and

$$t = \lambda\vec{x}.(W\vec{x}).$$

The word w' is the word w and hence it is derivable.

Case $t = \lambda\vec{x}.(F_i(\lambda\vec{w}.t')\vec{y})$. We know that $|\lambda\vec{w}.t'|_{\varphi_0}$ is the encoding of the word v_i and thus $|(F_i(\lambda\vec{w}.t'))|_{\varphi_0}$ is the encoding of the word w_i.

Since t denotes a word weak encoding $|t| * \ldots * LR * \ldots * = Y$.

If some y_j ($j \leq n-1$) is the variable $x_{j'}$ then, by lemma 4.1.15, $j' \neq k$ and thus $|t| *^{\sim(j'-1)} LR*^{\sim(k-j'-1)} = Y$ and $y_{i+1} = x_{j'+1}$. Hence the variables $\vec{y}$ are consecutive: $\vec{y} = x_{p+1}, \ldots, x_{p+n}$. Call $\vec{z} = z_1, \ldots, z_q$ the variables $x_{p+n+1}, \ldots, x_k$. We have

$$t = \lambda\vec{x}\vec{y}\vec{z}.(F_i(\lambda\vec{w}.t')\vec{y})$$

We write $w' = u_1 w u_2$ (where u_1 has length p, w length n and u_2 length q).

The variables $\vec{y}$ are not free in $\lambda\vec{w}.t'$, hence the term $\lambda\vec{x}\vec{w}\vec{z}.t'$ is closed. We verify that it denotes a weak encoding of the word $u_1 v_i u_2$.

- First clause.
  - If l be the j^{th} letter of u_1. We have

    $$|\lambda\vec{x}\vec{y}\vec{z}.(F_i(\lambda\vec{w}.t')\vec{y})|\ *^{\sim(j-1)}\ l*^{\sim(p-j+n+q)} = Y$$

    Let $\varphi = \varphi_0 + \langle x_j, l\rangle$. The function $|F_i(\lambda\vec{w}.t')|_\varphi$ maps $*\ldots*$ to Y. Hence, the function $|\lambda\vec{w}.t'|_\varphi$ maps $*\ldots*$ to Y and other arguments to N. Hence

    $$|\lambda\vec{x}\vec{w}\vec{z}.t'|\ *^{\sim(j-1)}\ l*^{\sim(p-j+m+q)} = Y$$

  - We know that $|\lambda\vec{w}.t'|_{\varphi_0}$ is the encoding of the word v_i. Hence if l is the j^{th} letter of the word v_i then

    $$|\lambda\vec{x}\vec{w}\vec{z}.t'|\ *^{\sim(p+j-1)}\ l*^{\sim(n-j+q)} = Y$$

  - In a way similar to the first case, we prove that if l is the j^{th} letter of u_2. We have

    $$|\lambda\vec{x}\vec{w}\vec{z}.t'|\ *^{\sim(p+m+j-1)}\ l*^{\sim(q-j)} = Y$$

- Second clause.
  - If $j \leq p-1$, we have

    $$|\lambda\vec{x}\vec{y}\vec{z}.(F_i(\lambda\vec{w}.t')\vec{y})|\ *^{\sim(j-1)}\ LR*^{\sim(p-j-1+m+q)} = Y$$

    Let φ be φ_0 but x_j to L and x_{j+1} to R. The function $|F_i(\lambda\vec{w}.t')|_\varphi$ maps $*\ldots*$ to Y. Hence, the function $|\lambda\vec{w}.t'|_\varphi$ maps $*\ldots*$ to Y and other arguments to N and

    $$|\lambda\vec{x}\vec{w}\vec{z}.t'|\ *^{\sim(j-1)}\ LR*^{\sim(p-j-1+m+q)} = Y$$

  - We have

    $$|\lambda\vec{x}\vec{y}\vec{z}.(F_i(\lambda\vec{w}.t')\vec{y})|\ *^{\sim(p-1)}\ LR*^{\sim(n-1+q)} = Y$$

    Let φ be φ_0 but x_p to L. The function $|F_i(\lambda\vec{w}.t')|_\varphi$ maps $R*\ldots*$ to Y. Hence, the function $|\lambda\vec{w}.t'|_\varphi$ maps $R*\ldots*$ to Y and other arguments to N and

    $$|\lambda\vec{x}\vec{w}\vec{z}.t'|\ *^{\sim(p-1)}\ LR*^{\sim(m-1+q)} = Y$$

  - We know that $|\lambda\vec{w}.t'|_{\varphi_0}$ is the encoding of the word v_i. Hence if $j \leq m-1$ then

    $$|\lambda\vec{x}\vec{w}\vec{z}.t'|\ *^{\sim(p+j-1)}\ LR*^{\sim(m-j-1+q)} = Y$$

  - In a way similar to the second, we prove that

    $$|\lambda\vec{x}\vec{w}\vec{z}.t'|\ *^{\sim(p+m-1)}\ LR*^{\sim(q-1)} = Y$$

– In a way similar to the first, we prove that if $j \leq q-1$, we have

$$|\lambda\vec{x}\vec{w}\vec{z}.t'| *^{\sim(p+m+j-1)} LR*^{\sim(q-j-1)} = Y$$

Hence the term $\lambda\vec{x}\vec{w}\vec{z}.t'$ denotes a weak encoding of the word $u_1v_iu_2$. By induction hypothesis, the word $u_1v_iu_2$ is derivable and hence $u_1w_iu_2$ is derivable.

At last we prove that $w = w_i$, i.e. that $w' = u_1w_iu_2$. We know that $|F_i(\lambda\vec{w}.t')|_{\varphi_0}$ is the encoding of the word w_i. Hence

$$|\lambda\vec{x}\vec{y}\vec{z}.(F_i(\lambda\vec{w}.t')\vec{y})| *^{\sim(p+j-1)} l*^{\sim(n-j+q)} = Y$$

if and only if l is the j^{th} letter of the word w_i.

Since $|\lambda\vec{x}\vec{y}\vec{z}.(F_i(\lambda\vec{w}.t')\vec{y})|$ is a weak encoding of the word u_1wu_2, if l is the j^{th} letter of the word w, we have

$$|\lambda\vec{x}\vec{y}\vec{z}.(F_i(\lambda\vec{w}.t')\vec{y})| *^{\sim(p+j-1)} l*^{\sim(n-j+q)} = Y$$

and l is the j^{th} letter of the word w_i. Hence $w = w_i$ and $w' = u_1w_iu_2$ is derivable. ■

From proposition 4.1.9 and 4.1.16, we conclude.

4.1.17. Proposition. *The word w' is derivable if and only if there is a term whose free variables are among W, $F_1, \ldots, F_r$ that denotes the encoding of w'.*

4.1.18. Corollary. *Let w and w' be two words and $v_1 \hookrightarrow w_1$, …, $v_r \hookrightarrow w_r$ be rewrite rules. Let h be the encoding of w, h' be the encoding of w', f_1 be the encoding of $v_1 \hookrightarrow w_1$, …, f_n be the encoding of $v_r \hookrightarrow w_r$.*

The word w' is derivable from w with the rules $v_1 \hookrightarrow w_1$, …, $v_r \hookrightarrow w_r$ if and only if there is a definable function that maps $h, f_1, \ldots, f_n$ to h'.

4.1.19. Theorem. *(Loader) λ-definability is undeciable, i.e. there is no algorithm deciding whether a table describes a λ-definable element of the model.*

Proof. If there were a algorithm to decide if a function is definable or not, then a generate and test algorithm would permit to decide if there is a definable function that maps $h, f_1, \ldots, f_n$ to h' and hence if w' is derivable from w with the rules $v_1 \hookrightarrow w_1$, …, $v_r \hookrightarrow w_r$ contradicting the undecidablity of the word rewriting problem. ■

Joly has extended Loader's result in two directions as follows. Let $\mathcal{M}_n = \mathcal{M}_{\{0,\ldots,n-1\}}$. Define for $n \in \mathbb{N}, A \in \mathbb{T}, d \in \mathcal{M}_n(A)$

$$D(n, A, d) \iff d \text{ is } \lambda\text{-definable in } \mathcal{M}_n.$$

Since for a fixed n_0 and A_0 the set $\mathcal{M}_{n_0}(A_0)$ is finite, it follows that $D(n_0, A_0, d)$ as predicate in d is decidable. One has the folowing.

4.1.20. Proposition. *Undecidability of λ-definability is monotonic in the following sense.*

$$\lambda\!\!\lambda Ad.D(n_0, A, d) \text{ undecidable} \,\&\, n_0 \leq n_1 \;\Rightarrow\; \lambda\!\!\lambda Ad.D(n_1, A, d) \text{ undecidable.}$$

Proof. Use Exercise 3.6.18(i). ■

Loader's proof above shows in fact that $\lambda\!\!\lambda Ad.D(7, A, d)$ is undecidable. It was sharpened in Loader [2001a] showing that $\lambda\!\!\lambda Ad.D(3, A, d)$ is undecidable.

4.1.21. Theorem (Joly [2005]). *$\lambda\!\!\lambda Ad.D(2, A, d)$ is undecidable.*

4.1.22. Theorem (Joly [2005]). *$\lambda\!\!\lambda nd.P(n, 3{\to}o{\to}o, d)$ is undecidable.*

Loosely speaking one can say that λ-definability at the monster type $\mathsf{M} = 3{\to}1$ is undecidable. Moreover, Joly also has characterised those types A that are undecidable in this sense.

4.1.23. Definition. A type A is called *finitely generated* iff there are closed terms $M_1, \ldots, M_n$, not necessarily of type A such that every closed term of type A is an applicative product of the $M_1, \ldots, M_n$.

4.1.24. Theorem (Joly [2002]). *Let $A \in \mathbb{T}$. Then $\lambda\!\!\lambda nd.D(n, A, d)$ is decidable iff the closed terms of type A can be finitely generated.*

For a sketch of the proof see Exercise 3.6.31.

4.1.25. Corollary. *The monster type $\mathsf{M} = 3{\to}o{\to}o$ is not finitely generated.*

Proof. By Theorems 4.1.24 and 4.1.22. ■

4.2. Undecidability of Unification

4.2.1. Definition. (i) Let $M, N \in \Lambda^{\emptyset}(A{\to}B)$. A *pure unification problem* is of the form

$$\exists X\ A.MX = NX,$$

where one searches for an $X \in \Lambda^{\emptyset}(A)$ (and the equality is $=_{\beta\boldsymbol{\eta}}$). A is called the *search-type* and B the *output-type* of the problem.

(ii) Let $M \in \Lambda^{\emptyset}(A{\to}B), N \in \Lambda^{\emptyset}(B)$. A *pure matching problem* is of the form

$$\exists X\ A.MX = N,$$

where one searches for an $X \in \Lambda^{\emptyset}(A)$. Again A, B are the search- and output types, respectively.

(iii) Often we write for a unification or matching problem (when the types are known from the context or are not relevant) simply

$$MX = NX$$

or

$$MX = N.$$

and speak about the unification (matching) problem with *unknown* X.

Of course matching problems are a particular case of unification problems: solving the matching problem $MX = N$ amounts to solving the unification problem $MX = (\lambda x.N)X$.

4.2.2. Definition. The *rank* (*order*) of a unification or matching problem is $\mathrm{rk}(A)$ ($\mathrm{ord}(A)$ respectively), where A is the search-type. Remember that $\mathrm{rk}(A) = \mathrm{ord}(A) + 1$.

The rank of the output-type is less relevant. Basically one may assume that it is $\top = 1_2 \to o \to o$. Indeed, if $\Phi : B \leq \top$ then

$$MX = NX : B \iff (\Phi \circ M)X = (\Phi \circ N)X : \top.$$

One has $\mathrm{rk}(\top) = 2$. The unification and matching problems with an output type of rank <2 are decidable, see Exercise 4.5.7.

The main results of this section are that unification in general is undecidable from a low level onward (Goldfarb) and matching up to order 4 is decidable (Padovani). It is an open problem wether matching in general is decidable. As a spin-off of the study of matching problems it will be shown that the maximal theory is decidable.

4.2.3. Example. The following are two examples of pure unification problems.

(i) $\exists X\ (1 \to o).\lambda f\ 1.f(Xf) = X$.
(ii) $\exists X\ (1 \to o \to o).\lambda fa.X(Xf)a = \lambda fa.Xf(Xfa)$.

This is not in the format of the previous Definition, but we mean of course

$$\begin{array}{rcl} (\lambda x\ (1 \to o)\lambda f\ 1.f(xf))X & = & (\lambda x\ (1 \to o)\lambda f\ 1.xf)X; \\ (\lambda x : (1 \to o \to o)\lambda f\ 1 \lambda a\ o.x(xf)a)X & = & (\lambda x : (1 \to o \to o)\lambda f\ 1 \lambda a\ o.xf(xfa))X. \end{array}$$

The most understandable form is as follows (provided we remember the types)

$$\begin{array}{rcl} \text{(i)} \quad \lambda f.f(Xf) & = & X; \\ \text{(ii)} \quad X(Xf)a & = & Xf(Xfa). \end{array}$$

The first problem has no solution, becuase there is no fixed-point combinator in $\lambda^o_{\to}$. The second one does ($\lambda fa.f(fa)$ and $\lambda fa.a$), because $n^2 = 2n$ for $n \in \{2, 4\}$.

4.2.4. Example. The following are two pure matching problems.

$$\begin{array}{rcll} (X(Xf)a & = & f^{10}a & X\ 1 \to o \to o; f\ 1, a\ o; \\ f(X(Xf)a) & = & f^{10}a & X\ 1 \to o \to o; f\ 1, a\ o. \end{array}$$

The first problem is without a solution, because $\sqrt{10} \notin \mathbb{N}$. The second with a solution ($X \equiv \lambda fa.f^3a$), because $3^2 + 1 = 10$.

Now the unification and matching problems will be generalized. First of all we will consider more unknowns. Then more equations. Finally, in the general versions of unification and matching problems one does not require that the

$\vec{M}, \vec{N}, \vec{X}$ are closed but they may contain a fixed finite number of constants (free variables). All these generalized problems will be reducible to the pure case, but (only in the transition from non-pure to pure problems) at the cost of possibly raising the rank (order) of the problem.

4.2.5. Definition. (i) Let M, N be closed terms of the same type. A *pure unification problem with several unknowns*

$$M\vec{X} =_{\beta\boldsymbol{\eta}} N\vec{X} \tag{1}$$

searches for closed terms $\vec{X}$ of the right type satisfying (1). The rank of a problem with several unknowns $\vec{X}$ is

$$\max\{\mathrm{rk}(A_i) \mid 1 \leq i \leq n\},$$

where the A_i are the types of the X_i. The order is defined similarly.

(ii) A *system of pure unification problems* starts with terms $M_1, \ldots, M_n$ and $N_1, \ldots, N_n$ such that M_i, N_i are of the same type for $1 \leq i \leq n$. searching for closed terms $\vec{X}_1, \ldots, \vec{X}_n$ all occuring among $\vec{X}$ such that

$$\begin{array}{rcl} M_1\vec{X}_1 & =_{\beta\boldsymbol{\eta}} & N_1\vec{X}_1 \\ & \ldots & \\ M_n\vec{X}_n & =_{\beta\boldsymbol{\eta}} & N_n\vec{X}_n \end{array}$$

The rank (order) of such a system of problems the maximum of the ranks (orders) of the types of the unknowns.

(iii) In the general (non-pure) case it will also be allowed to have the $M, N, \vec{X}$ range over Λ^Γ rather than $\Lambda^\emptyset$. We call this a unification problem *with constants from* Γ. The rank of a non-pure system of unknowns is defined as the maximum of the rank (orders) of the types of the unknowns.

(iv) The same generalizations are made to the matching problems.

4.2.6. Example. A pure system of matching problem in the unknowns P, P_1, P_2 is the following. It states the existence of a pairing and is solvable depending on the types involved, see Barendregt [1974].

$$\begin{array}{rcl} P_1(Pxy) & = & x \\ P_2(Pxy) & = & y. \end{array}$$

One could add a third equation (for surjectivity of the pairing)

$$P(P_1z)(P_2z) = z,$$

causing this system never to have solutions, see Barendregt [1974].

4.2.7. Example. An example of a unification problem with constants from $\Gamma = \{a\,1, b\,1\}$ is the following. We searh for unknowns $W, X, Y, Z \in \Lambda^\Gamma(1)$ such that

$$\begin{array}{rcl} X & = & Y \circ W \circ Y \\ b \circ W & = & W \circ b \\ W \circ W & = & b \circ W \circ b \\ a \circ Y & = & Y \circ a \\ X \circ X & = & Z \circ b \circ b \circ a \circ a \circ b \circ b \circ Z, \end{array}$$

where $f \circ g = \lambda x.f(gx))$ for f, g 1, having as unique solution $W = b$, $X = a \circ b \circ b \circ a$, $Y = Z = a$. This example will be expanded in Exercise 4.5.6.

4.2.8. PROPOSITION. *All unification (matching) problems reduce to pure ones with just one unknown and one equation. In fact we have the following.*

(i) *A problem of rank k with several unkowns can be reduced to a problem with one unknown with rank* $\mathrm{rk}(A) = \max\{k, 2\}$.

(ii) *Systems of problems can be reduced to one problem, without altering the rank. The rank of the output type will be* $\max\{\mathrm{rk}(B_i), 2\}$, *where B_i are the output types of the respective problems in the system.*

(iii) *Non-pure problems with constants from Γ can be reduced to pure problems. In this process a problem of rank k becomes of rank*

$$\max\{\mathrm{rk}(\Gamma), k\}.$$

PROOF. We give the proof for unification.

(i) In the notation of Definition 1.3.19 we have

$$\begin{aligned} &\exists \vec{X}.M\vec{X} = N\vec{X} && (1)\\ &\iff \exists X.(\lambda x.M(x \cdot 1) \ldots (x \cdot n))X = (\lambda x.N(x \cdot 1) \ldots (x \cdot n))X. && (2) \end{aligned}$$

Indeed, if the $\vec{X}$ work for (1), then $X \equiv \langle \vec{X} \rangle$ works for (2). Conversely, if X works for (2), then $\vec{X} \equiv X \cdot 1, \ldots, X \cdot n$ work for (1). By Proposition 5.2 we have $A = A_1 \times \ldots \times A_n$ is the type of X and $\mathrm{rk}(A) = \max\{\mathrm{rk}(A_1), \ldots, \mathrm{rk}(A_n), 2\}$.

(ii) Similarly for $\vec{X}_1, \ldots, \vec{X}_n$ being subsequences of $\vec{X}$ one has

$$\begin{aligned} \exists \vec{X} \quad M_1\vec{X}_1 &= N_1\vec{X}_1 \\ &\ldots \\ M_n\vec{X}_n &= N_n\vec{X}_n \\ \iff \exists \vec{X} \quad (\lambda \vec{x}.\langle M_1\vec{x}_1, \ldots, M_n\vec{x}_n \rangle)\vec{X} &= (\lambda \vec{x}.\langle N_1\vec{x}_1, \ldots, N_n\vec{x}_n \rangle)\vec{X}. \end{aligned}$$

(iii) Write a non-pure problem with $M, N \in \Lambda^{\Gamma}(A \to B)$, and $\mathrm{dom}(\Gamma) = \{\vec{y}\}$ as

$$\exists X[\vec{y}]\ A.M[\vec{y}]X[\vec{y}] = N[\vec{y}]X[\vec{y}].$$

This is equivalent to the pure problem

$$\exists X\ (\bigwedge \Gamma \to A).(\lambda x\vec{y}.M[\vec{y}](x\vec{y}))X = (\lambda x\vec{y}.N[\vec{y}](x\vec{y}))X. \blacksquare$$

Although the 'generalized' unification and matching problems all can be reduced to the pure case with one unknown and one equation, one usually should not do this if one wants to get the right feel for the question.

Decidable case of unification

4.2.9. PROPOSITION. *Unification with unknowns of type* 1 *and constants of types* $o, 1$ *is decidable.*

PROOF. The essential work to be done is the solvability of Markov's problem by Makanin. See Exercise 4.5.6 for the connection and a reference. ■

Undecidability of unification

The undecidability of unification was first proved by Huet. This was done before the undecidability of Hilbert's 10-th problem (Is it decidable whether an arbitrary Diophantine equation over $\mathbb{Z}$ is solvable?) was established. Huet reduced Post's correspondence problem to the unification problem. The theorem by Matijasevic makes things more easy.

4.2.10. THEOREM (Matijasevič). (i) *There are two polynomials p_1, p_2 over $\mathbb{N}$ (of degree 7 with 13 variables[1]) such that*

$$D = \{\vec{n} \in \mathbb{N} \mid \exists \vec{x} \in \mathbb{N}. p_1(\vec{n}, \vec{x}) = p_2(\vec{n}, \vec{x})\}$$

is undecidable.

(ii) *There is a polynomial $p(\vec{x}, \vec{y})$ over $\mathbb{Z}$ such that*

$$D = \{\vec{n} \in \mathbb{N} \mid \exists \vec{x} \in \mathbb{Z}. p(\vec{n}, \vec{x}) = 0\}$$

is undecidable. Therefore Hilbert's 10-th problem is undecidable.

PROOF. (i) This was done by coding arbitrary RE sets as Diophantine sets of the form D. See Matijasevič [1971], Davis [1973] or Matiyasevič [1993].

(ii) Take $p = p_1 - p_2$ with the p_1, p_2 from (i). Using the theorem of Lagrange

$$\forall n \in \mathbb{N} \exists a, b, c, d \in \mathbb{N}. n = a^2 + b^2 + c^2 + d^2,$$

it follows that for $n \in \mathbb{Z}$ one has

$$n \in \mathbb{N} \iff \exists a, b, c, d \in \mathbb{N}. n = a^2 + b^2 + c^2 + d^2.$$

Finally write $\exists x \in \mathbb{N}. p(x, \ldots) = 0$ as $\exists a, b, c, d \in \mathbb{Z}. p(a^2 + b^2 + c^2 + d^2, \ldots) = 0$. ■

4.2.11. COROLLARY. *The solvability of pure unification problems of order 3 (rank 2) is undecidable.*

PROOF. Take the two polynomials p_1, p_2 and D from (i) of the theorem. Find closed terms M_{p_1}, M_{p_2} representing the polynomials, as in Corollary 1.3.5. Let $U_{\vec{n}} = \{M_{p_1}{}^{\ulcorner}\vec{n}^{\urcorner}\vec{x} = M_{p_2}{}^{\ulcorner}\vec{n}^{\urcorner}\vec{x}\}$. Using that every $X \in \Lambda^{\emptyset}(\mathsf{Nat})$ is a numeral, Proposition 2.1.22, it follows that this unification problem is solvable iff $\vec{n} \in D$. ■

The construction of Matijasevic is involved. The encoding of Post's correspondence problem by Huet is a more natural way to show the undecidability of unification. It has as disadvantage that needs to use unification at variable types. There is a way out. In Davis et al. [1960] it is proved that every RE predicate is of the form $\exists \vec{x} \forall y_1 < t_1 \ldots \forall y_n < t_n. p_1 = p_2$. Using this result and higher types (Nat_A, for some non-atomic A) one can get rid of the bounded

[1] This can be pushed to polynomials of degree 4 and 58 variables or of degree $1.6 * 10^{45}$ and 9 variables, see Jones [1982].

quantifiers. The analogon of Proposition 2.1.22 ($X\ \mathsf{Nat} \Rightarrow X$ a numeral) does not hold but one can filter out the 'numerals' by a unification (with $f\ A{\to}A$):

$$f \circ (Xf) = (Xf) \circ f.$$

This yields without Matijasevic's theorem that unification with for the unknown a fixed type is undecidable.

4.2.12. THEOREM. *Unification of order* 2 *(rank 1) with constants is undecidable.*

PROOF. See Exercise 4.5.4. ■

This implies that pure unification of order 3 is undecidable, something we already saw in Corollary 4.2.11. The interest in this result comes from the fact that unification over order 2 variables plays a role in automated deduction and the undecidability of this problem, being a subcase of a more general situation, is not implied by Corollary 4.2.11.

4.3. Decidability of matching of rank 3

The main result will be that matching of rank 3 (which is the same as order 4) is decidable and is due to Padovani [2000]. On the other hand Loader [2003] has proved that general matching modulo $=_\beta$ is undecidable. The decidability of general matching modulo $=_{\beta\boldsymbol{\eta}}$, which is the intended case, remains open.

The structure of this section is as follows. First the notion of interpolation problem is introduced. Then by using tree automata it is shown that these problems restricted to rank 3 are decidable. Then at rank 3 the problem of matching is reduced to interpolation and hence solvable. At rank 1 matching with several unknowns is already NP-complete.

4.3.1. PROPOSITION. (i) *Matching with unknowns of rank 1 is NP-complete.*
(ii) *Pure matching of rank 2 is NP-complete.*

PROOF. (i) Consider $A = o^2{\to}o = \mathsf{Bool}_o$. Using Theorem 2.1.19, Proposition 1.2.3 and Example 1.2.8 it is easy to show that if $M \in \Lambda^\emptyset(A)$, then $M \in {}_{\beta\boldsymbol{\eta}}\{\mathsf{true}, \mathsf{false}\}$ [We should have somthing better: be able to refer to ONE result]. By Proposition 1.3.2 a Boolean function $p(X_1, \ldots, X_n)$ in the variables $X_1, \ldots, X_n$ is λ-definable by a term $M_p \in \Lambda^\emptyset(A^n{\to}A)$. Therefore

$$p \text{ is satisfiable} \iff M_p X_1 \ldots X_n = \mathsf{true} \text{ is solvable.}$$

This is a matching problem of rank 1.
(ii) By (i) and Proposition 4.2.8. ■

In this chapter, we prove the decidability of fourth-order matching. A higher-order matching problem is a set of equations $t = u$ such that t contains both variables and constants and u contains only constants. A solution of such a problem is a substitution σ such that, for each equation, $\sigma t =_{\beta\eta} u$. A

matching problem is said to be of rank n if all the variables of t have at most rank n.

Following an idea of Statman [1982], the decidability of the matching problem can be reduced to the existence for every term u of a logical relation $\|_u$ on terms $\lambda^o_{\rightarrow}$ such that

- $\|_u$ is an equivalence relation;
- for all types T the quotient $\mathcal{T}_T/\|_u$ is finite;
- there is an algorithm that enumerates $\mathcal{T}_T/\|_u$, i.e. that takes in argument a type T and returns a finite sequence of terms representing all the classes.

Indeed, if such a relation exists, then a simple generate and test algorithm permits to solve the higher-order matching problem.

Similarly the decidability of the matching problem of rank n can be reduced to the existence of a relation such that $\mathcal{T}_T/\|_u$ can be enumerated up to rank n.

The finite completeness theorem yields the existence of a standard model $\mathcal{M}$ such that the relation $\mathcal{M} \models t = u$ meets the two first requirements, but Loader's theorem shows that it does not meet the third.

Padovani has proposed another relation - the *relative observational equivalence* - that is enumerable up to order 4. Like in the construction of the finite completeness theorem, the relative observational equivalence relation identifies terms of type o that are $\beta\eta$-equivalent and also all terms of type o that are not subterms of u. But this relation disregards the result of the application of a term to a non definable element.

Padovani has proved that the enumerability of this relation up to rank n can be reduced to to the decidability of a variant of the matching problem of rank n: the *dual interpolation problem* of rank n. Interpolation problems have been introduced in Dowek [1994] as a first step toward decidability of third-order matching. The decidability of the dual interpolation problem of order 4 has been also proved by Padovani. However, here we shall not present the original proof, but a simpler one proposed by Comon and Jurski Comon and Jurski [1998].

Rank 3 interpolation problems

An *interpolation equation* is a equation is a particular matching problem

$$X\ \vec{t} = u,$$

where $t_1, ..., t_n$ and u are closed terms. That is, the unknown X occurs at the head. A solution of such an equation is a term v such that

$$v\ \vec{t} =_{\beta\eta} u.$$

An *interpolation problem* is a conjunction of such equations with the same unkown. A solution of such a problem is a term v that is a solution for all

the equations simultaneously. A *dual interpolation problem* is a conjunction of equations and negated equations. A solution of such a problem is a term solution of all the equations but solution of none of the negated equations. If a dual interpolation problem has a solution it has also a closed solution in lnf. Hence, without loss of generality, we can restrict the search to such terms.

To prove the decidability of the fourth-order dual interpolation problem, we shall prove that the solutions of an interpolation equation can be recognized by a finite tree automaton. Then, the results will follow from the decidability of the non-emptiness of a set of terms recognized by a finite tree automaton and the closure of recognizable sets of terms by intersection and complement.

Relevant solution

In fact, it is not exactly quite so that the solutions of a fourth-order interpolation equation can be recognized by a finite state automaton. Indeed, a solutions of an interpolation equation may contain an arbitrary number of variables. For instance the equation

$$X\mathsf{K} = a$$

where X is a variable of type $(o \to 1 \to o) \to o$ has all the solutions

$$\lambda f.(fa(\lambda z_1.(fa(\lambda z_2.(fa...(\lambda z_n(fz_1(\lambda y.(fz_2(\lambda y.(fz_3....(fz_n(\lambda y.a))..))))..)))))))).$$

Moreover since each z_i has $z_1, ..., z_{i-1}$ in its scope it is not possible to rename these bound variables so that the variables of all these solutions are in a fixed finite set.

Thus the language of the solution cannot be *a priori* limited. In this example, it is clear however that there is another solution

$$\lambda f.(f\ a\ \Box)$$

where $\Box$ is a new constant of type $o \to o$. Moreover all the solutions above can be retrieved from this one by replacing the constant $\Box$ by an appropriate term (allowing captures in this replacement).

4.3.2. Definition. For each simple type T, we consider a constant $\Box_T$. Let t be a term solution of an interpolation equation. A subterm occurrence of t of type T is *irrelevant* if replacing it by the constant $\Box_T$ yields a solution. A *relevant* solution is a closed solution where all irrelevant subterm occurrences are the constant $\Box_T$.

Now we prove that relevant solutions of an interpolation equations can be recognized by a finite tree automaton.

An example

Consider the problem

$$X\mathbf{c}_1 = ha$$

where X is a variable of type $(1 \to o \to o) \to o$ and a and h are constants of type o and 1_2. A relevant solution of this equation substitutes X by the term $\lambda f.v$ where v is a relevant solution of the equation $v[f := \mathbf{c}_1] = ha$.

Let $\mathcal{Q}_{ha}$ be the set of the relevant solutions v of the equation $v[f := \mathbf{c}_1] = ha$. More generally, let $\mathcal{Q}_w$ be the set of relevant solutions v of the equation $v[f := \mathbf{c}_1] = w$.

Notice that terms in $\mathcal{Q}_w$ can only contain the constants and the free variables that occur in w, plus the variable f and the constants $\Box_T$. We can determine membership of such a set (and in particular to $\mathcal{Q}_{ha}$) by induction over the structure of a term.

- *analysis of membership to $\mathcal{Q}_{ha}$*

  A term is in $\mathcal{Q}_{ha}$ if it has either the form (hv_1) and v_1 is in $\mathcal{Q}_a$ or the form (fv_1v_2) and $(v_1[f := \mathbf{c}_1]v_2[f := \mathbf{c}_1]) = ha$. This means that there are terms v_1' and v_2' such that $v_1[f := \mathbf{c}_1] = v_1'$, $v_2[f := \mathbf{c}_1] = v_2'$ and $(v_1'v_2') = ha$, in other words there are terms v_1' and v_2' such that v_1 is in $\mathcal{Q}_{v_1'}$, v_2 is in $\mathcal{Q}_{v_2'}$ and $(v_1'v_2') = ha$. As $(v_1'v_2') = ha$ there are three possibilities for v_1' and v_2': $v_1' = \mathsf{I}$ and $v_2' = ha$, $v_1' = \lambda z.hz$ and $v_2' = a$ and $v_1' = \lambda z.ha$ and $v_2' = \Box_o$. Hence (fv_1v_2) is in $\mathcal{Q}_{ha}$ if either v_1 is in $\mathcal{Q}_{\mathsf{I}}$ and v_2 in $\mathcal{Q}_{ha}$ or v_1 is in $\mathcal{Q}_{\lambda z.hz}$ and v_2 in $\mathcal{Q}_a$ or v_1 is in $\mathcal{Q}_{\lambda z.ha}$ and $v_2 = \Box_o$.

  Hence, we have to analyze membership to $\mathcal{Q}_a$, $\mathcal{Q}_{\mathsf{I}}$, $\mathcal{Q}_{\lambda z.hz}$, $\mathcal{Q}_{\lambda z.ha}$.

- *analysis of membership to $\mathcal{Q}_a$*

  A term is in $\mathcal{Q}_a$ if it has either the form a or the form (fv_1v_2) and v_1 is in $\mathcal{Q}_{\mathsf{I}}$ and v_2 is in $\mathcal{Q}_a$ or v_1 in $\mathcal{Q}_{\lambda z.a}$ and $v_2 = \Box_o$.

  Hence, we have to analyze membership to $\mathcal{Q}_{\lambda z.a}$,

- *analysis of membership to $\mathcal{Q}_{\mathsf{I}}$*

  A term is in $\mathcal{Q}_{\mathsf{I}}$ if it has the form $\lambda z.v_1$ and v_1 is in $\mathcal{Q}_z$

  Hence, we have to analyze membership to $\mathcal{Q}_z$.

- *analysis of membership to $\mathcal{Q}_{\lambda z.hz}$*

  A term is in $\mathcal{Q}_{\lambda z.hz}$ if it has the form $\lambda z.v_1$ and v_1 is in $\mathcal{Q}_{hz}$

  Hence, we have to analyze membership to $\mathcal{Q}_{hz}$.

- *analysis of membership to $\mathcal{Q}_{\lambda z.ha}$*

  A term is in $\mathcal{Q}_{\lambda z.ha}$ if it has the form $\lambda z.v_1$ and v_1 is in $\mathcal{Q}_{ha}$.

- *analysis of membership to $\mathcal{Q}_{\lambda z.a}$*

  A term is in $\mathcal{Q}_{\lambda z.a}$ if it has the form $\lambda z.v_1$ and v_1 is in $\mathcal{Q}_a$.

- *analysis of membership to $\mathcal{Q}_z$*

  A term is in $\mathcal{Q}_z$ if it has the form z or the form (fv_1v_2) and either v_1 is in $\mathcal{Q}_{\mathsf{I}}$ and v_2 is in $\mathcal{Q}_z$ or v_1 is in $\mathcal{Q}_{\lambda z'.z}$ and $v_2 = \Box_o$.

  Hence, we have to analyze membership to $\mathcal{Q}_{\lambda z'.z}$.

- *analysis of membership to* $\mathcal{Q}_{hz}$

  A term is in $\mathcal{Q}_{hz}$ if it has the form (hv_1) and v_1 is in $\mathcal{Q}_z$ or the form (fv_1v_2) and either v_1 is in $\mathcal{Q}_\text{ı}$ and v_2 is in $\mathcal{Q}_{hz}$ or v_1 is in $\mathcal{Q}_{\lambda z.hz}$ and v_2 is in $\mathcal{Q}_z$ or v_1 is in $\mathcal{Q}_{\lambda z'.hz}$ and $v_2 = \Box_o$.

  Hence, we have to analyze membership to $\mathcal{Q}_{\lambda z'.hz}$.

- *analysis of membership to* $\mathcal{Q}_{\lambda z'.z}$

  A term is in $\mathcal{Q}_{\lambda z'.z}$ if it has the form $\lambda z'.v_1$ and v_1 is in $\mathcal{Q}_z$.

- *analysis of membership to* $\mathcal{Q}_{\lambda z'.hz}$

  A term is in $\mathcal{Q}_{\lambda z'.hz}$ if it has the form $\lambda z'.v_1$ and v_1 is in $\mathcal{Q}_{hz}$.

In this way we can build an automaton that recognizes in q_w the terms of $\mathcal{Q}_w$.

$$(hq_a) \to q_{ha}$$

$$(fq_\text{ı}q_{ha}) \to q_{ha}$$
$$(fq_{\lambda z.hz}q_a) \to q_{ha}$$
$$(fq_{\lambda z.ha}q_{\Box_o}) \to q_{ha}$$
$$a \to q_a$$
$$(fq_\text{ı}q_a) \to q_a$$
$$(fq_{\lambda z.a}q_{\Box_o}) \to q_a$$
$$\lambda z.q_z \to q_\text{ı}$$
$$\lambda z.q_{hz} \to q_{\lambda z.hz}$$
$$\lambda z.q_{ha} \to q_{\lambda z.ha}$$
$$\lambda z.q_a \to q_{\lambda z.a}$$
$$z \to q_z$$
$$(fq_\text{ı}q_z) \to q_z$$
$$(fq_{\lambda z'.z}q_{\Box_o}) \to q_z$$
$$(hq_z) \to q_{hz}$$
$$(fq_\text{ı}q_{hz}) \to q_{hz}$$
$$(fq_{\lambda z.hz}q_z) \to q_{hz}$$
$$(fq_{\lambda z'.hz}q_{\Box_o}) \to q_{hz}$$
$$\lambda z'.q_z \to q_{\lambda z'.z}$$
$$\lambda z'.q_{hz} \to q_{\lambda z'.hz}$$

Then we need a rule that permits to recognize $\Box_o$ in the state $q_{\Box_o}$

$$\Box_o \to q_{\Box_o}$$

and at last a rule that permits to recognize in q_0 the relevant solution of the equation $(X\mathbf{c}_1) = ha$

$$\lambda f.q_{ha} \to q_0$$

Notice that as a spin off we have proved that besides f all relevant solutions of this problem can be expressed with two bound variables z and z'.

The states of this automaton are labeled by the terms ha, a, I, $\lambda z.a$, $\lambda z.hz$, $\lambda z.ha$, z, hz, $\lambda z'.z$ and $\lambda z'.hz$. All these terms have the form

$$u = \lambda y_1 \dots \lambda y_p.C$$

where C is a context of a subterm of ha and the free variables of C are in the set $\{z, z'\}$.

Tree automata for relevant solutions

Let t be a normal term and V be a set of k variables of type o not occurring in t where k is the size of t. A *context* of t is a term C such that there exists a substittion σ mapping the variables of V to terms of type o such that $\sigma C = t$.

Consider an equation

$$X\vec{t} = u$$

where X is a variable of fourth-order type at most. Consider a finite number of constants $\Box_T$ for each type T of a subterm of u. Let k be the size of u. Consider a fixed set V of k variables of type o. Let N be the finite set of term of the form $\lambda y_1 ... \lambda y_p C$ where C is a context of a subterm of u and the free variables of C are in V. We define a tree automaton with the states q_w for w in N and $q_{\Box_T}$ for each constant $\Box_T$, and the transitions

- $(f_i q_{w_1} ... q_{w_n}) \to q_w$ if $(t_i \vec{w}) = w$ and replacing a w_i different from $\Box_T$ by a $\Box_T$ does not yield a solution,
- $(hq_{u_1} ... q_{u_n}) \to q_{(hu_1 ... u_n)}$ (for $u_1, ..., u_n$ in N)
- $\Box_T \to q_{\Box_T}$
- $\lambda z.q_t \to q_{\lambda z.t}$,
- $\lambda f_1 \dots \lambda f_n.q_u \to q_0$

4.3.3. Proposition. *Let a and b be two elements of N and $X_1, ..., X_n$ be variables of order at most two. Let σ be a relevant solution of the second-order matching problem*

$$(aX_1 ... X_n) = b$$

then for each i, either σX_i is in N (modulo alpha-conversion) or is equal to $\Box_T$.

Proof. Let a' be the normal form of $(a\sigma X_1 \dots \sigma X_{i-1} X_i \sigma X_{i+1} \dots \sigma X_n)$. If X_i has no occurrence in a' then as σ is relevant $\sigma X_i = \Box_T$.

Otherwise consider the higher occurrence l of a subterm of type o of a' that has the form $(X_i v_1 ... v_p)$. The terms $v_1, ..., v_p$ have type o. Let b' be the

subterm of b at the same occurrence l. The term b' has type o, it is a context of a subterm of u.

Let v'_i be the normal form of $v_i[\sigma X_i/X_i]$. We have $(\sigma X_i v'_1 ... v'_p) = b'$. Consider p variables $y_1, ..., y_p$ of V that not free in b'. We have $\sigma X_i = \lambda y_1 ... \lambda y_p C$ and $C[v'_1/y_1, ..., v'_p/y_p] = b'$. Hence C is a context of a subterm of u and $\sigma X_i = \lambda y_1 \ldots \lambda y_p.C$ is an element of N. ■

4.3.4. Remark. As a corollary of proposition 4.3.3, we get an alternative proof of the decidability of second-order matching.

4.3.5. Proposition. *Let*

$$X\vec{t} = u$$

be an equation, and $\mathcal{A}$ the associated automaton. Then a term is recognized by $\mathcal{A}$ (in q_0) if and only if it is a relevant solution of this equation.

Proof. We want to prove that a term v is recognized in q_0 if and only if it is a relevant solution of the equation $v\ \vec{t} = u$. It is sufficient to prove that v is recognized in the state q_u if and only if it is a relevant solution of the equation $v[f_1 := t_1, ..., f_n := t_n] = u$. We prove, more generally, that for any term w of N, v is recognized in q_w if and only if $v[f_1 := t_1, ..., f_n := t_n] = w$.

The direct sense is easy. We prove by induction over the structure of v that if v is recognized in q_w then v is a relevant solution of the equation $v[f_1 := t_1, ..., f_n := t_n] = w$. If $v = (f_i\ v_1\ ...\ v_p)$ then the term v_i is recognized in a state q_{w_i} where w_i is either a term of N or $\Box_T$ and $(t_i\ \vec{w}) = w$. In the first case, by induction hypothesis v_i is a relevant solution of the equation $v_i[f_1 := t_1, ..., f_n := t_n] = t_i$ and in the second $v_i = \Box_T$. Thus $(t_i\ v_1[f_1 := t_1, ..., f_n := t_n]\ ...\ v_p[f_1 := t_1, ..., f_n := t_n]) = u$ i.e. $v[f_1 := t_1, ..., f_n := t_n] = u$, and moreover v is relevant. If $v = (h\ v_1\ ...\ v_p)$ then the v_i are recognized in states q_{w_i} with w_i in N. By induction hypothesis v_i are relevant solutions of $v_i[f_1 := t_1, ..., f_n := t_n] = t_i$. Hence $v[f_1 := t_1, ..., f_n := t_n] = u$ and moreover v is relevant. The case where v is an abstraction is similar.

Conversely, assume that v is a relevant solution of the equation $v[f_1 := t_1, ..., f_n := t_n] = w$. We prove, by induction over the structure of v, that v is recognized in q_w.

If $v = (f_i\ v_1\ ...\ v_p)$ then $(t_i\ v_1[f_1 := t_1, ..., f_n := t_n]\ ...\ v_p[f_1 := t_1, ..., f_n := t_n]) = u$. Let $v'_i = v_i[f_1 := t_1, ..., f_n := t_n]$. The v'_i are a relevant solutions of the second-order matching problem $(t_i\ v'_1\ ...\ v'_p) = u$.

Hence, by proposition 4.3.3, each v'_i is either an element of N or the constant $\Box_T$. In both cases v_i is a relevant solution of the equation $v_i[f_1 := t_1, ..., f_n := t_n] = v'_i$ and by induction hypothesis v_i is recognized in q_{w_i}. Thus v is recognized in q_w. If $v = (h\ v_1\ ...\ v_p)$ then $(h\ v_1[f_1 := t_1, ..., f_n := t_n]\ ...\ v_p[f_1 := t_1, ..., f_n := t_n]) = w$. Let $w_i = v_i[f_1 := t_1, ..., f_n := t_n]$. We have $(h\ \vec{w}) = w$ and v_i is a relevant solution of the equation $v_i[f_1 := t_1, ..., f_n := t_n] = w_i$. By induction hypothesis v_i is recognized in q_{w_i}. Thus v is recognized in q_w. The case where w is an abstraction is similar. ■

4.3.6. Proposition. *Fourth-order dual interpolation is decidable.*

PROOF. Consider a system of equations and disequations and the automata associated to all these equations. Let $\mathcal{L}$ be the language containing the union of the languages of these automata and an extra constant of type o. Obviously the system has a solution if and only if it has a solution in the language $\mathcal{L}$. Each automaton recognizing the relevant solutions can be transformed into one recognizing all the solutions in $\mathcal{L}$ (adding a finite number of rules, so that the state $\Box_T$ recognizes all terms of type T in the language $\mathcal{L}$). Then using the fact that languages recognized by a tree automaton are closed by intersection and complement, we build a automaton recognizing all the solutions of the system in the language $\mathcal{L}$. The system has a solution if and only if the language recognized by this automaton is non empty.

Decidability follows from the decidability of the emptiness of a language recognized by a tree automaton. ■

Decidability of rank 3 matching

A particular case

We shall start by proving the decidability of a subcase of fourth-order matching where problems are formulated in a language without any constant and the solutions also must not contain any constant.

Consider a problem $t = u$. The term u contains no constant. Hence, by Theorem 3.4.7 there is a closed term r of type $T \to o$, whose constants have order at most two (i.e. level at most one), such that for each term t of type T

$$t =_{\beta\eta} u \iff \forall i.(r\ t) =_{\beta\boldsymbol{\eta}} (r\ u).$$

The normal form of $(r\ u) \in \Lambda^{\emptyset}(o)$ is a closed term whose constants have order at most two, thus it contains no bound variables. Let U be the set of all subterms of type o of the normal forms of $(r\ u)$. All these terms are closed. Like in the relation defined by equality in the model of the finite completeness theorem, we define a congruence on closed terms of type o that identifies all terms that are not in U. This congruence has $card(U) + 1$ equivalence classes.

4.3.7. DEFINITION. $t =_{\beta\eta u} t' \iff \forall s \in U\ [t =_{\beta\eta} s \iff t' =_{\beta\eta} s]$.

Notice that if $t, t' \in \Lambda^{\emptyset}(o)$ one has the following

$$\begin{array}{lll} t =_{\beta\eta u} t' & \iff & t =_{\beta\eta} t' \text{ or } \forall s \in U\ (t \neq_{\beta\boldsymbol{\eta}} s\ \&\ t \neq_{\beta\boldsymbol{\eta}} s) \\ & \iff & [t =_{\beta\eta} t' \\ & & \text{or neither the normal form of } t \text{ nor that of } t' \text{ is in } U] \end{array}$$

Now we extend this to a logical relation on closed terms of arbitrary types. The following construction could be considered as an application of the Gandy Hull defined in Example 3.3.37. Hovere, we choose to do it explicitly so as to prepare for Definition 4.3.16.

4.3.8. DEFINITION. Let $\|_v$ be the logical relation lifted from $=_{\beta\boldsymbol{\eta}u}$ on closed terms.

4.3.9. LEMMA. (i) $\|_v$ *is head-expansive.*
(ii) *For each constant* F *of type of rank* ≤ 1 *one has* $F \|_v F$.
(iii) *For any* $X \in \Lambda(A)$ *one has* $X \|_v X$.
(iv) $\|_v$ *is an equivalence relation.*
(v) $P \|_v Q \iff \forall R_1, \ldots, R_k . P\vec{R} \|_v Q\vec{R}$.

We want to prove, using the decidability of the dual interpolation problem, that the equivalence classes of this relation can be enumerated up to order four, i.e. that we can compute a set $\mathcal{E}_T$ of closed terms containing a term in each class.

More generally, we shall prove that if dual interpolation of rank n is decidable, then the sets $\mathcal{T}_T/\|_u$ can be enumerated up to rank n. We first prove the following Proposition.

4.3.10. PROPOSITION (Substitution lemma). *Let* t *be a normal term of type* o, *whose free variables are* $x_1, ..., x_n$. *Let* $v_1, ..., v_n$, $v'_1, ..., v'_n$ *be closed terms such that* $v_1 \|_u v'_1$, ..., $v_n \|_u v'_n$. *Let* $\sigma = v_1/x_1, ..., v_n/x_n$ *and* $\sigma' = v'_1/x_1, ..., v'_n/x_n$. *Then*

$$\sigma t =_{\beta\eta u} \sigma' t$$

PROOF. By induction on the pair formed with the length of the longest reduction in σt and the size of t. The term t is normal and has type o, thus it has the form $(f\ w_1\ ...\ w_k)$.

If f is a constant, then let us write $w_i = \overline{\lambda} s_i$ with s_i of type o. We have $\sigma t = (f\ \overline{\lambda}\ \sigma s_1\ ...\ \overline{\lambda}\ \sigma s_k)$ and $\sigma' t = (f\ \overline{\lambda}\ \sigma' s_1\ ...\ \overline{\lambda}\ \sigma' s_k)$. By induction hypothesis (as the s_i's are subterms of a) we have $\sigma s_1 =_{\beta\eta u} \sigma' s_1$, ..., $\sigma s_k =_{\beta\eta u} \sigma' s_k$ thus either for all i, $\sigma s_i =_{\beta\eta} \sigma' s_i$ and in this case $\sigma t =_{\beta\eta} \sigma' t$ or for some i, neither the normal forms of σs_i nor that of $\sigma' s_i$ is an element of U. In this case neither the normal form of σt nor that of $\sigma' t$ is in U and $\sigma t =_{\beta\eta u} \sigma' t$.

If f is a variable x_i and $k = 0$ then $t = x_i$, $\sigma t = v_i$ and $\sigma' t = v'_i$ and v_i and v'_i have type o. Thus $\sigma t =_{\beta\eta u} \sigma' t$.

Otherwise, f is a variable x_i and $k \neq 0$. The term v_i has the form $\lambda z_1 ... \lambda z_k\ s$ and the term v'_i has the form $\lambda z_1 ... \lambda z_k\ s'$. We have

$$\sigma t = (v_i\ \sigma w_1\ ...\ \sigma w_k) =_{\beta\eta} s[\sigma w_1/z_1, ..., \sigma w_k/z_k]$$

and $\sigma' t = (v'_i\ \sigma' w_1\ ...\ \sigma' w_k)$. As $v_i \|_u v'_i$, we get

$$\sigma' t =_{\beta\eta u} (v_i\ \sigma' w_1\ ...\ \sigma' w_k) =_{\beta\eta u} s[\sigma' w_1/z_1, ..., \sigma' w_k/z_k]$$

It is routine to check that for all i, $(\sigma w_i) \|_u (\sigma' w_i)$. Indeed, if the term w_i has the form $\lambda_1\ ...\ \lambda y_p\ a$, then for all closed terms $b_1\ ...\ b_p$, we have

$$\begin{aligned} \sigma w_i\ b_1\ ...\ b_p &= ((b_1/y_1, ..., b_p/y_p) \circ \sigma)a \\ \sigma' w_i\ b_1\ ...\ b_p &= ((b_1/y_1, ..., b_p/y_p) \circ \sigma')a. \end{aligned}$$

Applying the induction hypothesis to a that is a subterm of t, we get

$$(\sigma w_i)\ b_1\ ...\ b_p =_{\beta\eta u} (\sigma' w_i)\ b_1\ ...\ b_p$$

and thus $(\sigma w_i) \parallel_u (\sigma' w_i)$.

As $(\sigma w_i) \parallel_u (\sigma' w_i)$ we can apply the induction hypothesis again (because $s[\sigma w_1/z_1, ..., \sigma w_k/z_k]$ is a reduct of σt) and get

$$s[\sigma w_1/z_1, ..., \sigma w_k/z_k] =_{\beta\eta u} s[\sigma' w_1/z_1, ..., \sigma' w_k/z_k]$$

Thus $\sigma t =_{\beta\eta u} \sigma' t$. ■

The next proposition is a direct corollary.

4.3.11. Proposition (Application lemma). *If $v_1 \parallel_u v'_1$, ..., $v_n \parallel_u v'_n$ then for all term t of type $T_1 \to ... \to T_n \to o$,*

$$(t\ v_1\ ...\ v_n) =_{\beta\eta u} (t\ v'_1\ ...\ v'_n)$$

Proof. Applying proposition 4.3.10 to the term $(t\ x_1\ ...\ x_n)$. ■

We then prove the following lemma that justifies the use of the relations $=_{\beta\eta u}$ and $\parallel_u$.

4.3.12. Proposition (Discrimination lemma). *For every term t, if $t \parallel_u u$ then $t =_{\beta\eta} u$.*

Proof. As $t \parallel_u u$, by proposition 4.3.11, we have for all i, $(r_i\ t) =_{\beta\eta u} (r_i\ u)$. Hence, as the normal form of $(r_i\ u)$ is in U, $(r_i\ t) =_{\beta\eta} (r_i\ u)$. Thus $t =_{\beta\eta} u$. ■

Let us discuss now how we can decide and enumerate the relation $\parallel_u$. If t and t' of type $T_1 \to ... \to T_n \to o$, then, by definition, $t \parallel_u t'$ if and only if

$$\forall w_1 \in \mathcal{T}_{T_1} ... \forall w_n \in \mathcal{T}_{T_n}\ (t\ \vec{w} =_{\beta\eta E} t'\ \vec{w})$$

The fact that $t\ \vec{w} =_{\beta\eta E} t'\ \vec{w}$ can be reformulated

$$\forall s \in U\ (t\ \vec{w} =_{\beta\eta} s \text{ if and only if } t'\ \vec{w} =_{\beta\eta} s)$$

Thus $t \parallel_u t'$ if and only if

$$\forall w_1 \in \mathcal{T}_{T_1} ... \forall w_n \in \mathcal{T}_{T_n}\ \forall s \in U\ (t\ \vec{w} =_{\beta\eta} s \text{ if and only if } t'\ \vec{w} =_{\beta\eta} s)$$

Thus to decide if $t \parallel_u t'$, we should list all the sequences $s, w_1, ..., w_n$ where s is an element of U and $w_1, ..., w_n$ are closed terms of type $T_1, ..., T_n$, and check that the set of sequences such that $t\ \vec{w} =_{\beta\eta} s$ is the same as the set of sequences such that $t'\ \vec{w} =_{\beta\eta} s$.

Of course, the problem is that there is an infinite number of such sequences. But by proposition 4.3.11 the fact that $t\ \vec{w} =_{\beta\eta u} t'\ \vec{w}$ is not affected if we replace the terms w_i by $\parallel_u$-equivalent terms. Hence, if we can enumerate the

sets $\mathcal{T}_{T_1}/\ \|_u$, ..., $\mathcal{T}_{T_n}/\ \|_u$ by sets $\mathcal{E}_{T_1}$, ..., $\mathcal{E}_{T_n}$, then we can decide the relation $\|_u$ for terms of type $T_1 \to ... \to T_n \to o$ by enumerating the sequences in $U \times \mathcal{E}_{T_1} \times ... \times \mathcal{E}_{T_n}$, and checking that the set of sequences such that $t\ \vec{w} =_{\beta\eta} s$ is the same as the set of sequences such that $t'\ \vec{w} =_{\beta\eta} s$.

As class of a term t for the relation $\|_u$ is completely determined, by the set of sequences $s, w_1, ..., w_n$ such that $t\ \vec{w} =_{\beta\eta} s$ and there are a finite number of subsets of the set $E = U \times \mathcal{E}_{T_1} \times ... \times \mathcal{E}_{T_n}$, we get this way that the set $\mathcal{T}_T/\ \|_u$ is finite.

To obtain an enumeration $\mathcal{E}_T$ of the set $\mathcal{T}_T/\ \|_u$ we need to be able to select the subsets A of $U \times \mathcal{E}_{T_1} \times ... \times \mathcal{E}_{T_n}$, such that there is a term t such that $t\ \vec{w} =_{\beta\eta} s$ if and only if the sequence $s, \vec{w}$ is in A. This condition is exactly the decidability of the dual interpolation problem. This leads to the following proposition.

4.3.13. Proposition (Enumeration lemma). *If dual interpolation of rank n is decidable, then the sets $\mathcal{T}_T/\ \|_u$ can be enumerated up to rank n.*

Proof. By induction on the order of $T = T_1 \to ... \to T_n \to o$. By the induction hypothesis, the sets $\mathcal{T}_{T_1}/\ \|_u, ..., \mathcal{T}_{T_n}/\ \|_u$ can be enumerated by sets $\mathcal{E}_{T_1}, ..., \mathcal{E}_{T_n}$.

Let x be a variable of type T. For each subset A of $E = U \times \mathcal{E}_{T_1} \times ... \times \mathcal{E}_{T_n}$ we define the dual interpolation problem containing the equation $x\vec{w} = s$ for $s, w_1, ..., w_p \in A$ and the negated equation $x\vec{w} \neq s$ for $s, w_1, ..., w_p \notin A$. Using the decidability of dual interpolation of rank n, we select those of such problems that have a solution and we chose a closed solution for each problem. We get this way a set $\mathcal{E}_T$.

We prove that this set is an enumeration of $\mathcal{T}_T/\ \|_u$, i.e. that for every term t of type T there is a term t' in $\mathcal{E}_T$ such that $t'\ \|_u\ t$. Let A be the set of sequences $s, w_1, ..., w_p$ such that $(t\ \vec{w}) =_{\beta\eta} s$. The dual interpolation problem corresponding to A has a solution (for instance t). Thus one of its solutions t' is in $\mathcal{E}_T$. We have

$$\forall w_1 \in \mathcal{E}_{T_1} ... \forall w_n \in \mathcal{E}_{T_n} \forall s \in U\ ((t\ \vec{w}) =_{\beta\eta} s \iff (t'\ \vec{w}) =_{\beta\eta} s)$$

Thus

$$\forall w_1 \in \mathcal{E}_{T_1} ... \forall w_n \in \mathcal{E}_{T_n}\ (t\ \vec{w}) =_{\beta\eta u} (t'\ \vec{w})$$

and thus, by proposition 4.3.11

$$\forall w_1 \in \mathcal{T}_{T_1} ... \forall w_n \in \mathcal{T}_{T_n} (t\ \vec{w}) =_{\beta\eta u} (t'\ \vec{w})$$

Thus $t\ \|_u\ t'$. ■

Then, we prove that if the sets $\mathcal{T}_T/\ \|_u$ can be enumerated up to rank n, then matching of rank n is decidable. The idea is that we can restrict the search of solutions to the sets $\mathcal{E}_T$.

4.3.14. Proposition (Matching lemma). *If the sets $\mathcal{T}_T/\ \|_u$ can be enumerated up to order N, then matching problems of rank n whose right hand side is u can be decided.*

PROOF. Let $\vec{X} = X_1, \ldots, X_n$. We prove that if a matching problem $t\vec{X} = u$ has a solution $\vec{v}$, then it has also a solution $\vec{v}'$, such that $v'_i \in \mathcal{E}_T$, for each i.

As $\vec{v}$ is a solution of the problem $t = u$, we have $t\vec{v} =_{\beta\eta} u$.

For all i, let v'_i be a representative in $\mathcal{E}_{T_i}$ of the class of v_i. We have

$$v'_1 \parallel_u v_1, \ldots, v'_n \parallel_u v_n.$$

Thus by proposition 4.3.10

$$t\vec{v} =_{\beta\eta u} \vec{v}',$$

thus

$$t\vec{v}' =_{\beta\eta u} u$$

and by proposition 4.3.12

$$t\vec{v}' =_{\beta\eta} u$$

Thus to check if a problem has a solution it is sufficient to check if it has a solution $\vec{v}'$, such that each v'_i is a member of $\mathcal{E}_T$, and such substitutions can be enumerated. ■

4.3.15. THEOREM. *Fourth-order matching problems whose right hand side contain no constants can be decided.*

PROOF. Dual interpolation of order 4 is decidable, hence, by proposition 4.3.13, if u is a closed term containing no constants, then the sets $\mathcal{T}/\parallel_u$ can be enumerated up to order 4, hence, by proposition 4.3.14, we can decide if a problem of the form $t = u$ has a solution. ■

The general case

We consider now terms formed in a langage containing an infinite number of constants of each type and we want to generalize the result. The difficulty is that we cannot apply Statman's result anymore to eliminate bound variables. Hence we shall define directly the set U as the set of subterms of u of type o. The novelty here is that the bound variables of U may now appear free in the terms of U. It is important here to chose the names $x_1, ..., x_n$ of these variables, once for all.

We define the congruence $t =_{\beta\eta u} t'$ on terms of type o that identifies all terms that are not in U.

4.3.16. DEFINITION. (i) Let $t, t' \in \Lambda(o)$ (not necessarily closed). Define

$$t =_{\beta\boldsymbol{\eta} u} t' \iff \forall s \in U.[t =_{\beta\eta} s \iff t' =_{\beta\boldsymbol{\eta}} s].$$

(ii) Define the logical relation $\parallel_u$ by lifting $=_{\beta\boldsymbol{\eta} U}$ to all open terms at higher types.

4.3.17. LEMMA. (i) $\parallel_v$ *is head-expansive.*

(ii) *For any variable of arbitrary type* A *one has* $x \parallel_v x$.

(iii) *For each constant* $F \in \Lambda(A)$ *one has* $F \parallel_v F$.

(iv) *For any* $X \in \Lambda(A)$ *one has* $X \parallel_v X$.

(v) $\|_v$ *is an equivalence relation at all types.*
(vi) $P \|_v Q \iff \forall R_1, \ldots, R_k. P\vec{R} \|_v Q\vec{R}$.

PROOF. (i) By definition the relation is closed under arbitrary $\beta\boldsymbol{\eta}$ expansion.
(ii) By induction on the generation of the type A.
(iii) Similarly.
(iv) Easy.
(v) Easy.
(vi) Easy. ■

Then we can turn to the enumerability lemma (proposition 4.3.13). Due to the presence of the free variables, the proof of this lemma introduces several novelties. Given a subset A of $E = U \times \mathcal{E}_{T_1} \times ... \times \mathcal{E}_{T_n}$ we cannot define the dual interpolation problem containing the equation $(x\ \vec{w}) = s$ for $s, w_1, \ldots, w_p \in A$ and the negated equation $(x\ \vec{w}) \neq s$ for $s, w_1, ..., w_p \notin A$, because the right hand side of these equations may contain free variables. Thus, we shall replace these variables by fresh constants $c_1, ..., c_n$. Let θ be the substitution $c_1/x_1, ..., c_n/x_n$. To each set of sequences, we associate the dual interpolation problem containing the equation $(x\ \vec{w}) = \theta s$ or its negation.

This introduces two difficulties: first the term θs is not a subterm of u, thus, besides the relation $\|_u$, we shall need to consider also the relation $\|_{\theta s}$, and one of its enumerations, for each term s in U. Then, the solutions of such interpolation problems could contain the constants $c_1, ..., c_n$, and we may have difficulties proving that they represent their $\|_u$-equivalence class. To solve this problem we need to duplicate the constants $c_1, ..., c_n$ with constants $d_1, ..., d_n$. This idea goes back to [Goldfarb].

Let us consider a fixed set of constants $c_1, ..., c_n$, $d_1, ..., d_n$ that do not occur in u, and if t is a term containing constants $c_1, ..., c_n$, but not the constants $d_1, ..., d_n$, we write $\tilde{t}$ for the term t where each constant c_i is replaced by the constant d_i.

Let $T = T_1 \to ... \to T_n \to o$ be a type. We assume that for any closed term s of type o, the sets $\mathcal{T}_{T_i}/\|_s$ can be enumerated up to rank n by sets $\mathcal{E}^s_{T_i}$.

4.3.18. DEFINITION. We define the set of sequences E containing for each term s in U and sequence $w_1, ..., w_n$ in $\mathcal{E}^{\theta s}_{T_1} \times ... \times \mathcal{E}^{\theta s}_{T_n}$, the sequence $\theta s, w_1, ..., w_n$. Notice that the terms in these sequences may contain the constants $c_1, ..., c_n$ but not the constants $d_1, ..., d_n$.

To each subset of A of E we associate a dual interpolation problem containing the equations $x\ \vec{w} = s$ and $x\ \tilde{w}_1\ ...\ \tilde{w}_n = \tilde{s}$ for $s, w_1, ..., w_n \in A$ and the disequations $x\ \vec{w} \neq s$ and $x\ \tilde{w}_1\ ...\ \tilde{w}_n \neq \tilde{s}$ for $s, w_1, ..., w_n \notin A$.

The fist lemma justifies the use of constants duplication.

4.3.19. PROPOSITION. *If an interpolation problem of definition 4.3.18 has a solution t, then it also has a solution t' that does not contain the constants $c_1, ..., c_n, d_1, ..., d_n$.*

PROOF. Assume that the term t contains a constant, say c_1. Then by replacing this constant c_1 by a fresh constant e, we obtain a term t'. As the constant e is fresh, all the disequations that t verify are still verified by t'. If t verifies the equations $x\ \vec{w} = s$ and $x\ \tilde{w}_1\ ...\ \tilde{w}_n = \tilde{s}$ then the constant e does not occur in the normal form of $t'\ \vec{w}$. Otherwise the constant c_1 would occur in the normal form of $t\ \tilde{w}_1\ ...\ \tilde{w}_n$, i.e. in the normal form of $\tilde{s}$ which is not the case. Thus t' also verifies the equations $x\ \vec{w} = s$ and $x\ \tilde{w}_1\ ...\ \tilde{w}_n = \tilde{s}$.

We can replace this way all the constants $c_1, ..., c_n, d_1, ..., d_n$ by fresh constants, obtaining a solution where these constants do not occur. ■

Then, we prove that the interpolation problems of definition 4.3.18 characterize the equivalence classes of the relation $\|_u$.

4.3.20. PROPOSITION. *Every term t of type T not containing the constants $c_1, ..., c_n, d_1, ..., d_n$ is the solution of a unique problem of definition 4.3.18.*

PROOF. Consider the subset A of E formed with sequences $s, w_1, ..., w_n$ such that $t\ \vec{w} = s$. The term t is the solution of the interpolation problem associated to A and A is the only subset of E such that t is a solution to the interpolation problem associated to. ■

4.3.21. PROPOSITION. *Let t and t' be two terms of type T not containing the constants $c_1, ..., c_n, d_1, ..., d_n$. Then t and t' are solutions of the same problem if and only if $t \|_u t'$.*

PROOF. By definition if $t \|_u t'$ then for all $w_1, ..., w_n$ and for all s in U $t\ \vec{w} =_{\beta\eta} s \iff t'\ \vec{w} =_{\beta\eta} s$. Thus for any $s, \vec{w}$ in E, $\theta^{-1}s$ is in U and $t\ \theta^{-1}w_1\ ...\ \theta^{-1}w_n =_{\beta\eta} \theta^{-1}s \iff t'\ \theta^{-1}w_1\ ...\ \theta^{-1}w_n =_{\beta\eta} \theta^{-1}s$. Then as the constants $c_1, ..., c_n, d_1, ..., d_n$ do not appear in t and t' $t\ \vec{w} =_{\beta\eta} s \iff t'\ \vec{w} =_{\beta\eta} s$ and $t\ \tilde{w}_1\ ...\ \tilde{w}_n =_{\beta\eta} s \iff t'\ \tilde{w}_1\ ...\ \tilde{w}_n =_{\beta\eta} \tilde{s}$. Thus t and t' are the solutions of the same problems.

Conversely, assume that $t \not\|_u t'$. Then there exists terms $w_1, ..., w_n$ and a term s in U such that $t\ \vec{w} =_{\beta\eta} s$ and $t'\ \vec{w} \neq_{\beta\eta} s$. Hence $t\ \theta w_1\ ...\ \theta w_n =_{\beta\eta} \theta s$ and $t'\ \theta w_1\ ...\ \theta w_n \neq_{\beta\eta} \theta s$. As the sets $\mathcal{E}^{\theta s}_{T_i}$ are enumeration of the sets $\mathcal{T}_{T_i}/\ \|_{\theta s}$ there exists terms $\vec{r}$ such that the $r_i \|_{\theta s} \theta w_i$ and $\theta s, \vec{r} \in E$. Using proposition 4.3.11 we have $t\ \vec{r} =_{\beta\eta\theta s} t\ \theta w_1\ ...\ \theta w_n =_{\beta\eta} \theta s$ hence $t\ \vec{r} =_{\beta\eta\theta s} \theta s$ i.e. $t\ \vec{r} =_{\beta\eta} \theta s$. Similarly, we have $t'\ \vec{r} =_{\beta\eta\theta s} t'\ \theta w_1\ ...\ \theta w_n \neq_{\beta\eta} \theta s$ hence $t'\ \vec{r} \neq_{\beta\eta\theta s} \theta s$ i.e. $t'\ \vec{r} \neq_{\beta\eta} \theta s$ Hence t and t' are not the solutions of the same problems. ■

Finally, we can prove the enumeration lemma.

4.3.22. PROPOSITION (Enumeration lemma). *If dual interpolation of rank n is decidable, then, for any closed term u of type o, the sets $\mathcal{T}_T/\ \|_u$ can be enumerated up to rank n.*

PROOF. By induction on the order of T. Let $T = T_1 \to ... \to T_n \to o$. By the induction hypothesis, for any closed term s of type o, the sets $\mathcal{T}_{T_i}/\ \|_s$ can be enumerated by sets $\mathcal{E}^s_{T_i}$.

We consider all the interpolation problems of definition 4.3.18. Using the decidability of dual interpolation of rank n, we select those of such problems that have a solution. By proposition 4.3.19, we can construct for each such problem a solution not containing the constants $c_1, ..., c_n, d_1, ..., d_n$ and by proposition 4.3.20 and 4.3.21, these terms form an enumeration of $\mathcal{T}_T/\parallel_u$. ■

To conclude, we prove the matching lemma (proposition 4.3.14) exactly as in the particular case and then the theorem.

4.3.23. THEOREM. *(Padovani) Fourth-order matching problems can be decided.*

PROOF. Dual interpolation of order 4 is decidable, hence, by proposition 4.3.13, if u is a closed term, then the sets $\mathcal{T}/\parallel_u$ can be enumerated up to order 4, hence, by proposition 4.3.14, we can decide if a problem of the form $t = u$ has a solution. ■

4.4. Decidability of the maximal theory

We prove now that the maximal theory is decidable. The original proof of this result is due to Vincent Padovani [1996]. This proof has later been simplified independently by Schmidt-Schauß and Loader [1997], based on Schmidt-Schauß [1999].

Remember that the maximal theory is

$$\mathcal{T}_{\max}\{M = N \mid M, N \in \Lambda_o^{\emptyset}(A), A \in \mathbb{TT}^o \ \&\ \mathcal{M}_{\min}^{\vec{c}} \models M = N\},$$

where

$$\mathcal{M}_{\min}^{\vec{c}} = \Lambda_o^{\emptyset}[\vec{c}]/\approx_{\vec{c}}^{\text{ext}}$$

consists of all terms having the $\vec{c} = c_1, \ldots, c_n$, with $n > 1$, of type o as distinct constants and $M \approx_{\vec{c}}^{\text{ext}} N$ on type $A = A_1 \to \ldots \to A_a \to o$ is defined by

$$M \approx_{\vec{c}}^{\text{ext}} N \iff \forall t_1 \in \Lambda_o^{\emptyset}[\vec{c}](A_1) \ldots t_a \in \Lambda_o^{\emptyset}[\vec{c}](A_a). M\vec{t} =_{\beta\eta} N\vec{t}.$$

Theorem 3.5.29 states that $\approx_{\vec{c}}^{\text{ext}}$ is a congruence which we will denote by $\approx$. Also that theorem implies that $\mathcal{T}_{\max}$ is independent of n.

4.4.1. DEFINITION. For a type $T \in \mathbb{TT}^{\mathbb{A}}$ we define its *degree* $||T||$ as follows.

$$\begin{aligned} ||o|| &= 2, \\ ||T \to U|| &= ||T||!||U||, \quad \text{i.e. } ||T|| \text{ factorial times } ||U||. \end{aligned}$$

4.4.2. PROPOSITION. (i) $||T_1 \to \ldots \to T_n \to o|| = 2||T_1||! \ldots ||T_n||!$.
(ii) $||T_i|| < ||T_1 \to \ldots \to T_n \to o||$.
(iii) $n < ||T_1 \to \ldots \to T_n \to o||$.
(iv) *If* $p < ||T_i||$, $||U_1|| < ||T_i||$, ..., $||U_p|| < ||T_i||$ *then*

$$||T_1 \to \ldots \to T_{i-1} \to U_1 \to \ldots \to U_p \to T_{i+1} \to \ldots \to T_n \to o|| < \\ < ||T_1 \to \ldots \to T_n \to o||$$

4.4.3. Definition. Let $M \in \Lambda^{\emptyset}_o[\vec{c}](T_1 \to \ldots T_n \to o)$ be a lnf. Then either $M \equiv \lambda x_1 \ldots x_n.y$ or $M \equiv \lambda x_1 \ldots x_n.x_i U_1 \ldots U_p$. In the first case, M is called *constant*, in the second it *has index* i.

The following proposition states that for every type T, the terms $t \in \Lambda^{\emptyset}_o[\vec{c}](T)$ with a given index can be enumerated by a term $E : \vec{V} \to T$, where the $\vec{V}$ have degrees lower than T.

4.4.4. Proposition. *Let $\approx$ be the equality in the minimal model (the maximal theory). Then for each type T and each natural number i, there exists a natural number $k < ||T||$, types $V_1, \ldots, V_k$ such that $||V_1|| < ||T||$, ..., $||V_k|| < ||T||$, a term E of type $V_1 \to \ldots \to V_k \to T$ and terms B_1 of type $T \to V_1$, ..., B_k of type $T \to V_k$ such that if t has index i then*

$$t \approx E(B_1 t) \ldots (B_k t)$$

Proof. By induction on $||T||$. Let us write $T = T_1 \to \ldots \to T_n \to o$ and $T_i = U_1 \to \ldots \to U_m \to o$. By induction hypothesis, for each j in $\{1, \ldots, m\}$ there are types $W_{j,1}, \ldots, W_{j,l_j}$, terms $E_j, B_{j,1}, \ldots, B_{j,l_j}$ such that $l_j < ||T_i||$, $||W_{j,1}|| < ||T_i||$, ..., $||W_{j,l_j}|| < ||T_i||$ and if $u \in \Lambda^{\emptyset}_o[\vec{c}](T)_i$ has index j then

$$u \approx E_j(B_{j,1}u) \ldots (B_{j,l_j}u).$$

We take $k = m$,

$$V_1 \equiv T_1 \to \ldots \to T_{i-1} \to W_{1,1} \to \ldots \to W_{1,l_1} \to T_{i+1} \to \ldots \to T_n \to o,$$

$$\ldots$$

$$V_k \equiv T_1 \to \ldots \to T_{i-1} \to W_{k,1} \to \ldots \to W_{k,l_k} \to T_{i+1} \to \ldots \to T_n \to o,$$

$$\begin{aligned}
E &\equiv \lambda f_1 \ldots f_k x_1 \ldots x_n.x_i\,(\lambda \vec{c}.f_1 x_1 \ldots x_{i-1}(d_{1,1}x_i) \ldots (d_{1,l_1}x_i)x_{i+1} \ldots x_n) \\
&\qquad \ldots \\
&\qquad (\lambda \vec{c}.f_k x_1 \ldots x_{i-1}(d_{k,1}x_i) \ldots (d_{k,l_k}x_i)x_{i+1} \ldots x_n), \\
B_1 &\equiv \lambda g x_1 \ldots x_{i-1} \vec{z}_1 x_{i+1} \ldots x_n.g x_1 \ldots x_{i-1}(E_1 \vec{z}_1)x_{i+1} \ldots x_n, \\
&\ldots \\
B_k &\equiv \lambda g x_1 \ldots x_{i-1} \vec{z}_k x_{i+1} \ldots x_n.g x_1 \ldots x_{i-1}(E_k \vec{z}_k)x_{i+1} \ldots x_n,
\end{aligned}$$

where $\vec{z}_i = z_1, \ldots, z_{l_i}$ for $1 \le i \le k$. We have $k < ||T_i|| < ||T||$, $||V_i|| < ||T||$ for $1 \le i \le k$ and for any $t \in \Lambda^{\emptyset}_o[\vec{c}](T)$

$$\begin{aligned}
E(B_1 t) \ldots (B_k t) &= \lambda x_1 \ldots x_n.x_i \\
&\quad (\lambda \vec{c}.t x_1 \ldots x_{i-1}(E_1(B_{1,1}x_i) \ldots (B_{1,l_1}x_i))x_{i+1} \ldots x_n) \\
&\quad \ldots \\
&\quad (\lambda \vec{c}.t x_1 \ldots x_{i-1}(E_k(B_{k,1}x_i) \ldots (B_{k,l_k}x_i))x_{i+1} \ldots x_n)
\end{aligned}$$

We want to prove that if t has index i then this term is equal to t. Consider terms $\vec{w} \in \Lambda^{\emptyset}_o[\vec{c}]$. We want to prove that for the term

$$\begin{aligned}
N &= w_i\,(\lambda \vec{c}.t w_1 \ldots w_{i-1}(E_1(B_{1,1}w_i) \ldots (B_{1,l_1}w_i))w_{i+1} \ldots w_n) \\
&\quad \ldots \\
&\quad (\lambda \vec{c}.t w_1 \ldots w_{i-1}(E_k(B_{k,1}w_i) \ldots (B_{k,l_k}w_i))w_{i+1} \ldots w_n)
\end{aligned}$$

one has $N \approx (tw_1 \dots w_n)$. If w_i is constant then this is obvious. Overwise, it has an index j, say, and N reduces to

$$N' \;=\; tw_1 \dots w_{i-1}(E_j(B_{j,1}w_i) \dots (B_{j,l_j}w_i))w_{i+1} \dots w_n.$$

By the induction hypothesis the term $(E_j(B_{j,1}w_i) \dots (B_{j,l_j}w_i)) \approx w_i$ and hence, by Theorem 3.5.29 one has $N = N' \approx (tw_1 \dots w_n)$. ■

4.4.5. Theorem. *Let $\mathcal{M}$ be the minimal model built over $\vec{c}$:o, i.e.*

$$\mathcal{M} = \mathcal{M}_{\min} = \Lambda^{\emptyset}_o[\vec{c}]/{\approx}.$$

For each type T, we can compute a finite set $\mathcal{R}_T \subseteq \Lambda^{\emptyset}_o[\vec{c}](T)$ that enumerates $\mathcal{M}(T)$, i.e. such that

$$\forall M \in \mathcal{M}(T) \exists N \in \mathcal{R}_T . M \approx N.$$

Proof. By induction on $||T||$. If $T = o$, then we can take $\mathcal{R}_T = \{\vec{c}\}$. Otherwise write $T = T_1 \to \dots \to T_n \to o$. By Proposition 4.4.4 for each $i \in \{1, \dots, n\}$, there exists a natural number k_i, types $V_{i,1}, \dots, V_{i,k_i}$ smaller than T, a term E_i of type $V_{i,1} \to \dots \to V_{i,k_i} \to T$ such that for each term t of index i, there exists terms v_1, …, v_{k_i} such that

$$t \approx (E_i v_1 \dots v_{k_i}).$$

By the induction hypothesis, for each type $V_{i,j}$ we can compute a finite set $R_{V_{i,j}}$ that enumerates $\mathcal{M}(V_{i,j})$. We take for R_T all the terms of the form $(E_i e_1 \dots e_{k_i})$ with e_1 in $R_{V_{i,1}}$, …, e_{k_i} in $R_{V_{i,k_i}}$. ■

4.4.6. Corollary (Padovani). *The maximal theory is decidable.*

Proof. Check equivalence in any minimal model $\mathcal{M}^{\vec{c}}_{\min}$. At type $A = A_1 \to \dots \to A_a \to o$ we have

$$M \approx N \iff \forall P_1 \in \Lambda^{\emptyset}_o[\vec{c}](A_1) \dots P_a \in \Lambda^{\emptyset}_o[\vec{c}](A_a) . M\vec{t} =_{\beta\eta} N\vec{t},$$

where we can now restrict the $\vec{P}$ to the $\mathcal{R}_{A_j}$. ■

4.4.7. Corollary (Decidability of unification in $\mathcal{T}_{\max}$). *For terms*

$$M, N \in \Lambda^{\emptyset}_o[\vec{c}](A \to B),$$

of the same type, the following unification problem is decidable

$$\exists X \in \Lambda^{\emptyset}[\vec{c}](A) . MX \approx NX.$$

Proof. Working in $\mathcal{M}^{\vec{c}}_{\min}$, check the finitely many enumerating terms as candidates. ■

4.4.8. Corollary (Decidability of atomic higher-order matching). (i) *For*

$$M_1 \in \Lambda_o^{\emptyset}[\vec{c}](A_1 \to o), \ldots, M_n \in \Lambda_o^{\emptyset}[\vec{c}](A_n \to o),$$

with $1 \leq i \leq n$, *the following problem is decidable*

$$\begin{aligned} \exists X_1 \in \Lambda_o^{\emptyset}[\vec{c}](A_1), \ldots, X_n \in \Lambda_o^{\emptyset}[\vec{c}](A_n).[M_1 X_1 &=_{\beta\boldsymbol{\eta}} c_1 \\ &\ldots \\ M_n X_n &=_{\beta\boldsymbol{\eta}} c_n]. \end{aligned}$$

(ii) *For* $M, N \in \Lambda_o^{\emptyset}[\vec{c}](A \to o)$ *the following problem is decidable.*

$$\exists X \in \Lambda_o^{\emptyset}[\vec{c}](A).MX =_{\beta\boldsymbol{\eta}} NX.$$

Proof. (i) Since $\beta\boldsymbol{\eta}$-convertibility at type o is equivalent to $\approx$, the previous Corollary applies.

(ii) Similarly to (i) or by reducing this problem to the problem in (i). ■

The non-redundancy of the enumeration

We now prove that the enumeration of terms in Proposition is not redundant. We follow the given construction, but actually the proof does not depend on it, see Exercise 4.5.2. We first prove a converse to Proposition 4.4.4.

4.4.9. Proposition. *Let* $E, B_1, \ldots, B_k$ *be the terms constructed in Proposition 4.4.4. Then for any sequence of terms* $M_1, \ldots, M_k$, *we have*

$$(B_j(EM_1 \ldots M_k)) \approx M_j$$

Proof. By induction on $||T||$ where T is the type of $(EM_1 \ldots M_k)$. The term

$$N \equiv B_j(EM_1 \ldots M_k)$$

reduces to

$$\begin{array}{l} \lambda x_1 \ldots x_{i-1} \vec{z}_j x_{i+1} \ldots x_n . E_j \vec{z}_j \\ (\lambda \vec{c}. M_1 x_1 \ldots x_{i-1} (B_{1,1}(E_j \vec{z}_j)) \ldots (B_{1,l_1}(E_j \vec{z}_j)) x_{i+1} \ldots x_n) \\ \ldots \\ (\lambda \vec{c}. M_k x_1 \ldots x_{i-1} (B_{k,1}(E_j \vec{z}_j)) \ldots (B_{k,l_k}(E_j \vec{z}_j)) x_{i+1} \ldots x_n) \end{array}$$

Then, since E_j is a term of index $l_j + j$, the term N continues to reduce to

$$\lambda x_1 \ldots x_{i-1} \vec{z}_j x_{i+1} \ldots x_n . M_j x_1 \ldots x_{i-1} (B_{j,1}(E_j \vec{z}_j)) \ldots (B_{j,l_j}(E_j \vec{z}_j)) x_{i+1} \ldots x_n.$$

We want to prove that this term is equal to M_j. Consider terms

$$N_1, \ldots, N_{i-1}, \vec{L}_j, N_{i+1}, \ldots, N_n \in \Lambda_o^{\emptyset}[\vec{c}].$$

It suffices to show that

$$\begin{array}{l} M_j N_1 \ldots N_{i-1} (B_{j,1}(E_j \vec{L}_j)) \ldots (B_{j,l_j}(E_j \vec{L}_j)) N_{i+1} \ldots N_n \approx \\ M_j N_1 \ldots N_{i-1} \vec{L}_j N_{i+1} \ldots N_n. \end{array}$$

By the induction hypothesis we have

$$(B_{j,1}(E_j\vec{L}_j)) \approx L_1$$

$$\dots$$

$$(B_{j,l_j}(E_j\vec{L}_j)) \approx L_{l_j}$$

Hence by Theorem 3.5.29 we are done. ■

4.4.10. Proposition. *The enumeration in Theorem 4.4.5 is non-redundant, i.e.*

$$\forall A \in \mathbb{T}^o \forall M, N \in \mathcal{R}_A . M \approx_{\mathcal{C}} N \;\Rightarrow\; M \equiv N.$$

Proof. Consider two terms M and N equal in the enumeration of a type A. We prove, by induction, that these two terms are equal. Since M and N are equal, they must have the same head variables. If this variable is free then they are equal. Otherwise, the terms have the form $M = (E_i M'_1 \dots M'_k)$ and $N = (E_i N'_1 \dots N'_k)$. For all j, we have

$$M'_j \approx (B_j M) \approx (B_j N) \approx N'_j$$

Hence, by induction hypothesis $M'_j = N'_j$ and therefore $M = N$. ■

4.5. Exercises

4.5.1. Let $T_n = 1^n \to o$ and let c_n be the cardinality of the set $\mathcal{M}_{T_n}$.

(i) Prove that

$$c_{n+1} = 2 + (n+1)c_n$$

(ii) Prove that

$$c_n = 2n! \sum_{i=0}^{n} \frac{1}{i!}.$$

The $d_n = n! \sum_{i=0}^{n} \frac{1}{i!}$ "the number of arrangements of n elements" forms a well-known sequence in combinatorics. See, for instance, Flajolet and Sedgewick [1993].

(iii) Can the cardinality of $\mathcal{M}_T$ be bounded by a function of the form $k^{|T|}$ where $|T|$ is the size of T and k a constant?

4.5.2. Let $\mathcal{C} = \{c^o, d^o\}$. Let $\mathcal{E}$ be a computable function that assigns to each type $A \in \mathbb{T}^o$ a finite set of terms $\mathcal{X}_A$ such that for all

$$\forall M \in \Lambda[\mathcal{C}](A) \exists N \in \mathcal{X}_A . M \approx_{\mathcal{C}} N.$$

Show that not knowing the theory of section 4.4 one can effectively make $\mathcal{E}$ non-redundant, i.e. such that

$$\forall A \in \mathbb{T}^o \forall M, N \in \mathcal{E}_A . M \approx_{\mathcal{C}} N \;\Rightarrow\; M \equiv N.$$

4.5.3. (Herbrand's Problem) Consider sets S of universally quantified equations $\forall x_1 \dots x_n.[t1 = t2]$ between first order terms involving constants $f, g, h, \dots$ various aritites. Herbrand's theorem concerns the problem of whether $S \models r = s$ where r, s are closed first order terms. For example the word problem for groups can be represented this way. Now let d be a new quaternary constant i.e. $d : 1_4$ and let a, b be new 0-ary constants i.e. $a, b : o$. We define the set S^+ of simply typed equations by

$$S^+ = \{ \ (\lambda \vec{x}.t_1 = \lambda \vec{x}.t_2) \ \mid \ (\forall \vec{x}[t_1 = t_2]) \in S\}.$$

Show that the following are equivalent

(i) $S \not\models r = s$.

(ii) $S^+ \cup \{\lambda x.dxxab = \lambda x.a, drsab = b\}$ is consistent.

Conclude that the consistency problem for finite sets of equations with constants is Π^0_1-complete (in contrast to the decidability of finite sets of pure equations).

4.5.4. (Undecidability of second-order unification) Consider the unification problem

$$Fx_1 \dots x_n = Gx_1 \dots x_n,$$

where each x_i has type of rank <2. By the theory of reducibility we can assume that $Fx_1 \dots x_n$ has type $(o{\rightarrow}(o{\rightarrow}o)){\rightarrow}(o{\rightarrow}o)$ and so by introducing new constants of types o ,and $o{\rightarrow}(o{\rightarrow}o)$ we can assume $Fx_1 \dots x_n$ has type o. Thus we arrive at the problem (with constants) in which we consider the problem of unifying 1st order terms built up from 1st and 2nd order constants and variables, The aim of this exercise is to show that it is recursively unsolvable by encoding Hilbert's 10-th problem, Goldfarb [1981]. For this we shall need several constants. Begin with constants

$$\begin{array}{ll} a, b & o \\ s & o{\rightarrow}o \\ e & o{\rightarrow}(o{\rightarrow}(o{\rightarrow}o)) \end{array}$$

The nth numeral is $s^n a$.

(i) Let $F\ o{\rightarrow}o$. F is said to be affine if $F = \lambda x.s^n x$. N is a numeral iff there exists an affine F such that $Fa = N$. Show that F is affine $\iff F(sa) = s(Fa)$.

(ii) Next show that $L = N + M$ iff there exist affine F and G such that $N = Fa,\ M = Ga$ & $L = F(Ga)$.

(iii) We can encode a computation of $n * m$ by

$$e(n * m)m(e(n * (m - 1))(m - 1)(...(e(n * 1)11)...)).$$

Finally show that $L = N * M \iff \exists c, d, u, v$ affine and $\exists f, w$

$$\begin{aligned} fab &= e(ua)(va)(wab) \\ f(ca)(sa)(e(ca)(sa)b) &= e(u(ca))(v(sa))(fab) \\ L &= ua \\ N &= ca \\ M &= va \\ &= da. \end{aligned}$$

4.5.5. (Sets of unifiers) The intention of this exercise is to prove the following.

THEOREM. *Let A be a simple type and let $\mathcal{S} \subseteq \Lambda^{\emptyset}(A)$ be recursively enumerable and closed under $=_{\beta\boldsymbol{\eta}}$. Then there exist simple types B and C and $F, G : A \to (B \to C)$ such that*

$$M \in \mathcal{S} \iff \exists N\, B.FMN =_{\beta\boldsymbol{\eta}} GMN.$$

Moreover we can always assume that

$$C = (o \to (o \to o)) \to (o \to o) = 1_2 \to o \to o.$$

In short, every r.e. $\beta\boldsymbol{\eta}$-closed set of combinators is the projection of the set of solutions to a unification problem.

(i) Let A be given. Then by the reducibility theorem for sinple types, there exists $H : A \to (o \to (o \to o)) \to (o \to o)$ such that

$$\forall M, N\, A.[M =_{\beta\boldsymbol{\eta}} N \iff HM =_{\beta\eta} HN].$$

Thus it suffices to prove the theorem for $A = 1_2 \to o \to o$. For similar reasons in the end it will suffice to have $C = 1_2 \to o \to o$. We shall need some special types below

$$\begin{aligned} o^* &= o \to o; \\ 1^* &= o^* \to o^*; \\ A &= (1^* \to (1^* \to 1^*)) \to (1^* \to 1^*); \\ A^+ &= (1^* \to (1^* \to 1^*)) \to ((1^* \to 1^*) \to (1^* \to 1^*)); \\ A^{\#} &= (1^* \to (1^* \to 1^*)) \to ((1^* \to 1^*) \to ((1^* \to 1^*) \to (1^* \to 1^*))). \end{aligned}$$

and we declare the following variables

$$\begin{aligned} f &: 1^* \to (1^* \to 1^*) \\ g &: 1^* \to 1^* \\ h &: 1^* \to 1^* \\ a &: o^* \to o^* \end{aligned}$$

where the reader should recognize $o^* \to o^*$ as the type of Church numerals.

(ii) We define several notions of terms recursively as follows.

- cheap
  a is cheap if r and s are cheap then frs is cheap
- inexpensive of level n
  a is inexpensive of level o
  if r and s are inexpensive of level n and j, k are among h, g then $f(jr)(ks)$ is inexpensive of level $n+1$.

Clearly, X is a cheap term of depth n iff there exists an inexpensive term Y of level n such that $X =_{\beta\boldsymbol{\eta}} [g = \mathsf{I}, h = \mathsf{K}a]Y$.
Cheap terms are useful because for each $N\ (o{\to}(o{\to}o)){\to}(o{\to}o)$ in $\beta\boldsymbol{\eta}$ normal form there exists a unique cheap X such that

$$N =_{\beta\boldsymbol{\eta}} (\lambda b\lambda xy.b(\lambda uv.\lambda wz.x(uwz)(vwz))(\lambda wz.y)Iy)(\lambda fa.X).$$

We shall say that M is frugal if $M =_{\beta\boldsymbol{\eta}} \lambda fa.X$ where X is cheap. We shall try to prove our theorem for r.e. $\beta\boldsymbol{\eta}$ closed sets of frugal terms.

(iii) We define

$$M\ A^{\#} \text{ is frugal if } M =_{\beta\boldsymbol{\eta}} \lambda fgha.Y, \text{ where } Y \text{ is inexpensive.}$$

Similarly we say that $M : A^+$ is frugal if $M =_{\beta\boldsymbol{\eta}} \lambda fga.[h = g]Y$ where Y is inexpensive. This will not introduce any ambiguity because of the difference between A, A^+, and $A^{\#}$. For $M\ A^{\#}$ define $M^+\ A^+$ by $M^+ = \lambda fga.Mfgga$. We have M is frugal iff M^+ is frugal. Claim:

$$M : A^+ \text{ is frugal } \Longleftrightarrow$$
$$\lambda fga.f(g(Mfga))(g(Mfga)) =_{\beta\boldsymbol{\eta}} \lambda fga.Mfg(f(ga)(ga)).$$

Proof. W.l.o.g. we may assume that M is in long $\beta\boldsymbol{\eta}$ normal form. The proof of $\Leftarrow$ is by induction on the length of M. We let $M = \lambda fga.\lambda uv.Z$.
Case 1. $Z = aUV$. Let @ be the substitution $[a = f(ga)(ga)]$. In this case we have

$$\lambda uv.f(ga)(ga)(@U)(@V) =_{\beta\boldsymbol{\eta}} f(g(\lambda uv.aUV)(g(\lambda uv.aUV)).$$

This is only possible if $\lambda uv.Z$ eta reduces to a so M is frugal.
Case 2. $Z = fXYUV$. In this case we have

$$f(g(\lambda uv.fXYUV))(g(\lambda uv.fXYUV)) =_{\beta\boldsymbol{\eta}}$$
$$\lambda uv.f(@X)(@Y)(@U)(@V)$$

and as in case 1 this is not possible unless U eta reducse to u and V eta reduces to v. Thus $f(g(fXY))(g(fXY)) =_{\beta\boldsymbol{\eta}} f(@X)(@Y)$ Now by similar reasoning X eta reduces to gP and Y eta reduces to gQ. Thus $f(gP)(gQ) =_{\beta\boldsymbol{\eta}} @P$ and @Q;in particular, $@P =_{\beta\boldsymbol{\eta}} @Q$ so $P =_{\beta\boldsymbol{\eta}} Q$ and $f(gP)(gP) =_{\beta\boldsymbol{\eta}} @Pf(gQ)(gQ) =_{\beta\boldsymbol{\eta}} @Q$. Thus setting $T = \lambda fga.P$ and

$$L = \lambda fga.Q\lambda fga.f(g(Tfga))(g(Tfga)) =_{\beta\boldsymbol{\eta}} \lambda fga.Tfg(f(ga)(ga)),$$

and $\lambda fga.f(g(Lfga))(g(Lfga))=_{\beta\boldsymbol{\eta}}\lambda fga.Lfg(f(ga)(ga))$. Now the induction hypothesis applies to the long $\beta\boldsymbol{\eta}$-normal forms of T and L so these terms are frugal. Thus so is M.
Case 3. $Z = gXUV$. This case is impossible.
Case 4. $Z = uV$ or $Z = v$. These cases are impossible.
This completes the proof of the claim.

(iv) By Matijasevic's solution to Hilbert's 10th problem for every r.e. set S of natural numbers there exists F and G such that n belongs to S iff there exists N such that $FnN=_{\beta\boldsymbol{\eta}}GnN$ (here N can be taken to be a 12-tuple of Church numerals). Now there exists a bijective polynomial pairing function p(x,y) represented by a lambda term P. If $M\ A$ is frugal then we take as the Gödel number for M the number represented by the $\beta\boldsymbol{\eta}$-nf of MP_1. The set of Gödel numbers of members of an r.e. $\beta\boldsymbol{\eta}$ closed set of frugal members of A is itself r.e. and has such F and G. We can now put all of these facts together. Consider the system

$$\begin{aligned} Ha &= \lambda bxy.b(\lambda uvwz.x(uwz)(vwz))(\lambda wz.y)\mathsf{I}y) \\ b &= \lambda fa.cf\mathsf{II}a \\ d &= \lambda fga.cfgga \\ \lambda fga.dfg(f(ga)(ga)) &= \lambda fga.f(g(dfga))(g(dfga)) \\ F(bP_1)e &= G(bP_1)e \end{aligned}$$

is unifiable with $a = M$ iff M belongs to the original r.e. $\beta\boldsymbol{\eta}$-closed set of closed terms.

4.5.6. Consider $\Gamma_{n,m} = \{c_1\,o, \ldots, c_m\,o, f_1\,1, \ldots, f_n\,o\}$. Show that the unification problem with constants from Γ with several unknowns of type 1 can be reduced to the case where $m = 1$. This is equivalent to the following problem of Markov. Given a finite alphabet $\Sigma = \{a_1, \ldots, a_n\}$ consider equations between words over $\Sigma \cup \{X_1, \ldots, X_p\}$. The aim is to find for the unknowns $\vec{X}$ words $w_1, \ldots, w_p \in \Sigma^*$ such that the equations become syntactic identities. In Makanin [1977] it is proved that this problem is decidable (uniformly in n, p).

4.5.7. (Decidability of unification of second-order terms) Consider the unification problem $F\vec{x} = G\vec{x}$ of type A with $\mathrm{rk}(A) = 1$. Here we are interested in the case of pure unifiers of any types. Then $A = 1_m = o^m \to o$ for some natural number m. Consider for $i = 1, \ldots, m$ the systems

$$S_i = \{F\vec{x} = \lambda\vec{y}.y_i, G\vec{x} = \lambda\vec{y}.y_i\}.$$

(i) Observe that the original unification problem is solvable iff one of the systems S_i is solvable.

(ii) Show that systems whose equations have the form

$$F\vec{x} = \lambda\vec{y}.y_i$$

where $y_i : 0$ have the same solutions as single equations

$$H\vec{x} = \lambda xy.x$$

where $x, y : 0$

(iii) Show that provided there are closed terms of the types of the x_i the solutions to a matching equation

$$H\vec{x} = \lambda xy.x$$

are exactly the same as the lambda definable solutions to this equation in the minimal model.

(iv) Apply the method of Exercise 2.5.7 to the minimal model. Conclude that if there is a closed term of type A then the lambda definable elements of the minimal model of type A are precisely those invariant under the transposition of the elements of the ground domain. Conclude that unification of terms of type of rank 1 is decidable.

Chapter 5

Extensions

5.1. Lambda delta

[LambdaDelta]
In this section the simply typed lambda calculus will be extended by constants δ $(= \delta_{A,B})$, for every $A, B \in \mathbb{T}(o)$. Church [1940] used this extension to introduce a logical system called "the simple theory of types", based on classical logic. (The system is also refered to as "higher order logic", and denoted by HOL.) We will introduce a variant of this system denoted by Δ. The intuitive idea is that $\delta = \delta_{A,B}$ satisfies for all for all $a, a' : A, b, b' : B$

$$\begin{array}{rcll} \delta aa'bb' & = & b & \text{if } a = a'; \\ & = & b' & \text{if } a \neq a'. \end{array}$$

Therefore the type of the new constants is as follows

$$\delta_{A,B} : A{\rightarrow}A{\rightarrow}B{\rightarrow}B{\rightarrow}B.$$

The set of typed terms *á la Church* with as type atoms only o will be denoted by $\Lambda\delta$; its elements will be called $\lambda\delta$*-terms*. The classical variant of the theory in which each term and variable carries its unique type will be considered only, but will suppress types whenever there is little danger of confusion.

The theory Δ is a strong logical system, in fact stronger than each of the 1st, 2nd, 3rd, ... order logics. It turns out that because of the presence of δ's an arbitrary formula of Δ is equivalent to an equation. This fact will be an incarnation of the comprehension principle. It is because of the δ's that Δ is powerful, less so because of the presence of quantification over elements of arbitrary types. Moreover, the set of equational consequences of Δ can be axiomatized by a finite subset. These are the main results in this section. It is an open question whether there is a natural (decidable) notion of reduction that is confluent and has as convertibility relation exactly these equational consequences. Since the decision problem for (higher order) predicate logic is undecidable, this notion of reduction will be non-terminating.

Higher Order Logic

In the following logical system terms are elements of Λ_o^{CL}. Formulas are built up form equations between terms of the same type using implication ($\supset$) and typed quantification ($\forall x^A.\varphi$). Absurdity is defined by $\bot \equiv (\mathsf{K} = \mathsf{K}_*)$, where $\mathsf{K} \equiv \lambda x^o y^o.x, \mathsf{K}_* \equiv \lambda x^o y^o.y.$ and negation by $\neg\varphi \equiv \varphi \supset \bot$. By contrast to other sections in this book Γ stand for a set of formulas. Finally $\Gamma \vdash \varphi$ is defined by the following axioms and rules. Variables always have to be given types such that the terms involved are typable and have the same type if they occur in one equation. Below Γ is a set of formulas, and $\mathrm{FV}(\Gamma) = \{x \mid x \in \mathrm{FV}(\varphi),\ \varphi \in \Gamma\}$.

M, N, L, P, Q are terms.

Δ : Higher Order Logic	
$\Gamma \vdash (\lambda x.M)N = M[x := N]$	(beta)
$\Gamma \vdash \lambda x.Mx = M,\ x \notin \mathrm{FV}(M)$	(eta)
$\Gamma \vdash M = M$	(reflexivity)
$\dfrac{\Gamma \vdash M = N}{\Gamma \vdash N = M}$	(symmetry)
$\dfrac{\Gamma \vdash M = N,\ \Gamma \vdash N = L}{\Gamma \vdash M = L}$	(transivity)
$\dfrac{\Gamma \vdash M = N,\ \Gamma \vdash P = Q}{\Gamma \vdash MP = NQ}$	(cong-app)
$\dfrac{\Gamma \vdash M = N}{\Gamma \vdash \lambda x.M = \lambda x.N}\ x \notin \mathrm{FV}(\Gamma)$	(cong-abs)
$\dfrac{\varphi \in \Gamma}{\Gamma \vdash \varphi}$	(axiom)
$\dfrac{\Gamma \vdash \varphi \supset \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$	($\supset$ -elim)
$\dfrac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \supset \psi}$	($\supset$ -intr)
$\dfrac{\Gamma \vdash \forall x^A.\varphi}{\Gamma \vdash \varphi[x := M]}\ M \in \Lambda(A)$	($\forall$-elim)
$\dfrac{\Gamma \vdash \varphi}{\Gamma \vdash \forall x^A.\varphi}\ x^A \notin \mathrm{FV}(\Gamma)$	($\forall$-intr)
$\dfrac{\Gamma, M \neq N \vdash \bot}{\Gamma \vdash M = N}$	(classical)
$\Gamma \vdash M = N \supset \delta MNPQ = P$	(delta$_L$)
$\Gamma \vdash M \neq N \supset \delta MNPQ = Q$	(delta$_R$)

Provability in this system will be denoted by $\Gamma \vdash_\Delta \varphi$.

5.1.1. Definition. The other logical connectives are introduced in the usual classical manner.

$$\begin{aligned} \varphi \vee \psi &::= \neg\varphi \supset \psi; \\ \varphi\psi &::= \neg(\neg\varphi \vee \neg\psi); \\ \exists x^A.\varphi &::= \neg\forall x^A.\neg\varphi. \end{aligned}$$

5.1.2. Lemma. *For all formulas of* Δ *one has*

$$\bot \vdash \varphi.$$

Proof. By induction on the structure of φ. If $\varphi \equiv (M = N)$, then observe that by (eta)

$$\begin{array}{rcccl} M & = & \lambda\vec{x}.M\vec{x} & = & \lambda\vec{x}.\mathsf{K}(M\vec{x})(N\vec{x}), \\ N & = & \lambda\vec{x}.N\vec{x} & = & \lambda\vec{x}.\mathsf{K}_*(M\vec{x})(N\vec{x}), \end{array}$$

where the $\vec{x}$ are such that the type of $M\vec{x}$ is o. Hence $\bot \vdash M = N$, since $\bot \equiv (\mathsf{K} = \mathsf{K}_*)$. If $\varphi \equiv (\psi \supset \chi)$ or $\varphi \equiv \forall x^A.\psi$, then the result follows immediately from the induction hypothesis. ■

5.1.3. Proposition. $\delta_{A,B}$ *can be defined from* $\delta_{A,o}$.

Proof. Indeed, if we only have $\delta_{A,o}$ (with their properties) and define

$$\delta_{A,B} = \lambda mnpq\vec{x}\,.\,\delta_{A,o}mn(p\vec{x})(q\vec{x}),$$

then all $\delta_{A,B}$ satisfy the axioms. ■

The rule (classical) is equivalent to

$$\neg\neg(M = N) \supset M = N.$$

In this rule the terms can be restricted to type o and the same theory Δ will be obtained.

5.1.4. Proposition. *Suppose that in the formulation of* Δ *one requires*

$$\Gamma, \neg(M = N) \vdash_\Delta \bot \;\Rightarrow\; \Gamma \vdash_\delta M = N \tag{1}$$

only for terms x, y *of type* o. *Then (1) holds for terms of all types.*

Proof. By (1) we have $\neg\neg M = N \supset M = N$ for terms of type o. Assume $\neg\neg(M = N)$, with M, N of arbitrary type, in order to show $M = N$. We have

$$M = N \supset M\vec{x} = N\vec{x},$$

for all fresh $\vec{x}$ such that the type of $M\vec{x}$ is o. By taking the contrapositive twice we obtain

$$\neg\neg(M = N) \supset \neg\neg(M\vec{x} = N\vec{x}).$$

Therefore by assumption and (1) we get $M\vec{x} = N\vec{x}$. But then by (cong-abs) and (eta) it follows that $M = N$. ■

5.1.5. Proposition. *For all formulas* φ *one has*

$$\vdash_\Delta \neg\neg\varphi \supset \varphi.$$

PROOF. Induction on the structure of φ. If φ is an equation, then this is a rule of the system Δ. If $\varphi \equiv \psi \supset \chi$, then by the induction hypothesis one has $\vdash_\Delta \neg\neg\chi \supset \chi$ and we have the following derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\psi \supset \chi]^1 \quad [\psi]^3}{\chi} \quad [\neg\chi]^2}{\bot}}{\neg(\psi \supset \chi)}\,1 \quad [\neg\neg(\psi \supset \chi)]^4}{\bot}}{\neg\neg\chi}\,2 \qquad \begin{array}{c}\vdots\\ \neg\neg\chi \supset \chi\end{array}}{\psi \supset \chi}\,3$$
$$\dfrac{\psi \supset \chi}{\neg\neg(\psi \supset \chi) \supset \psi \supset \chi)}\,4$$

for $\neg\neg(\psi \supset \chi) \supset (\psi \supset \chi)$. If $\varphi \equiv \forall x.\psi$, then by the induction hypothesis $\vdash_\Delta \neg\neg\psi(x) \supset \psi(x)$. Now we have a similar derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[\forall x.\psi(x)]^1}{\psi(x)} \quad [\neg\psi(x)]^2}{\bot}}{\neg\forall x.\psi(x)}\,1 \quad [\neg\neg\forall x.\psi(x)]^3}{\bot}}{\neg\neg\psi(x)}\,2 \qquad \begin{array}{c}\vdots\\ \neg\neg\psi(x) \supset \psi(x)\end{array}}{\psi(x)}$$
$$\dfrac{\psi(x)}{\forall x.\psi(x)}$$
$$\dfrac{\forall x.\psi(x)}{\neg\neg\forall x.\psi(x) \supset \forall x.\psi(x)}\,3$$

for $\neg\neg\forall x.\psi(x) \supset \forall x.\psi(x)$. ■

Now we will derive some equations in Δ that happen to be strong enough to provide an equational axiomatization of the equational part of Δ.

5.1.6. PROPOSITION. *The following equations hold universally (for those terms such that the equations make sense).*

$$\begin{array}{rcll}
\delta MMPQ & = & P & \textit{(δ-identity);}\\
\delta MNPP & = & P & \textit{(δ-reflexivity);}\\
\delta MNMN & = & N & \textit{(δ-hypothesis);}\\
\delta MNPQ & = & \delta NMPQ & \textit{(δ-symmetry);}\\
F(\delta MNPQ) & = & \delta MN(FP)(FQ) & \textit{(δ-monotonicity);}\\
\delta MN(P(\delta MN))(Q(\delta MN)) & = & \delta MN(P\mathsf{K})(Q\mathsf{K}_*) & \textit{(δ-transitivity).}
\end{array}$$

PROOF. We only show δ-reflexivity, the proof of the other assertions being similar. By the δ axioms one has

$$\begin{aligned} M = N &\vdash \delta MNPP = P; \\ M \neq N &\vdash \delta MNPP = P. \end{aligned}$$

By the "contrapositive" of the first statement one has $\delta MNPP \neq P \vdash M \neq N$ and hence by the second statement $\delta MNPP \neq P \vdash \delta MNPP = P$. So in fact $\delta MNPP \neq P \vdash \bot$, but then $\vdash \delta MNPP = P$, by the classical rule. ■

5.1.7. DEFINITION. The equational theory δ consists of equations between $\lambda\delta$-terms of the same type, axiomatized as follows. (As usual the axioms and rules are assumed to hold universally, i.e. the free variables may be replaced by arbitrary terms. In the following $\mathcal{E}$ denotes a set of equations between $\lambda\delta$-terms of the same type.

δ : Equational version of Δ	
$\mathcal{E} \vdash (\lambda x.M)N = M[x := N]$	(β)
$\mathcal{E} \vdash \lambda x.Mx = M,\ x \notin \mathrm{FV}(M)$	(η)
$\mathcal{E} \vdash M = M$	(reflexivity)
$\dfrac{\mathcal{E} \vdash M = N}{\mathcal{E} \vdash N = M}$	(symmetry)
$\dfrac{\mathcal{E} \vdash M = N,\ \mathcal{E} \vdash N = L}{\mathcal{E} \vdash M = L}$	(transivity)
$\dfrac{\mathcal{E} \vdash M = N,\ \mathcal{E} \vdash P = Q}{\mathcal{E} \vdash MP = NQ}$	(cong-app)
$\dfrac{\mathcal{E} \vdash M = N}{\mathcal{E} \vdash \lambda x.M = \lambda x.N}\ x \notin \mathrm{FV}(\mathcal{E})$	(cong-abs)
$\mathcal{E} \vdash \delta MMPQ = P$	(δ-identity)
$\mathcal{E} \vdash \delta MNPP = P$	(δ-reflexivity)
$\mathcal{E} \vdash \delta MNMN = N$	(δ-hypothesis)
$\mathcal{E} \vdash \delta MNPQ = \delta NMPQ$	(δ-symmetry)
$\mathcal{E} \vdash F(\delta MNPQ) = \delta MN(FP)(FQ)$	(δ-monotonicity)
$\mathcal{E} \vdash \delta MN(P(\delta MN))(Q(\delta MN)) = \delta MN(P\mathsf{K})(Q\mathsf{K}_*)$	(δ-transitivity)

The system δ may be given more conventionally by leaving out all occurences of $\mathcal{E} \vdash_\delta$ and replacing in the rule (cong-abs) the proviso "$x \notin \mathrm{FV}(\mathcal{E})$" by "$x$ not occurring in any assumption on which $M = N$ depends".

There is a canonical map from formulas to equations, preserving provability in Δ.

5.1.8. Definition. (i) For an equation $E \equiv (M = N)$ in Δ, write $E.L ::= M$ and $E.R ::= N$.

(ii) Define for a formula φ the corresponding equation φ^+ as follows.

$$\begin{array}{rcl}(M=N)^+ & ::= & M = N;\\ (\psi \supset \chi)^+ & ::= & (\delta(\psi^+.L)(\psi^+.R)(\chi^+.L)(\chi^+.R) = \chi^+.R);\\ (\forall x.\psi)^+ & ::= & (\lambda x.\psi^+.L = \lambda x.\psi^+.R).\end{array}$$

So, if $\psi^+ \equiv (M = N)$ and $\chi^+ \equiv (P = Q)$, then

$$\begin{array}{rcl}(\psi \supset \chi)^+ & ::= & (\delta MNPQ = Q);\\ (\neg\psi)^+ & ::= & (\delta MN\mathsf{KK}_* = \mathsf{K}_*);\\ (\forall x.\psi)^+ & ::= & (\lambda x.M = \lambda x.N).\end{array}$$

(iii) If Γ is a set of formulas, then $\Gamma^+ = \{\varphi^+ \mid \varphi \in \Gamma\}$.

5.1.9. Theorem. **[eqeq]** *For every formula φ one has*

$$\vdash_\Delta \varphi \leftrightarrow \varphi^+.$$

Proof. By induction on the structure of φ. If φ is an equation, then this is trivial. If $\varphi \equiv \psi \supset \chi$, then the statement follows from

$$\vdash_\Delta (M = N \supset P = Q) \leftrightarrow (\delta MNPQ = Q).$$

If $\varphi \equiv \forall x.\psi$, then this follows from

$$\vdash_\Delta \forall x.(M = N) \leftrightarrow (\lambda x.M = \lambda x.N). \blacksquare$$

We will show now that Δ is conservative over δ. The proof occupies 5.1.10-5.1.17

5.1.10. Lemma. **[app.behind]**
(i) $\vdash_\delta \delta MNPQz = \delta MN(Pz)(Qz)$.
(ii) $\vdash_\delta \delta MNPQ = \lambda z.\delta MN(Pz)(Qz)$.
(iii) $\vdash_\delta \lambda z.\delta MNPQ = \delta MN(\lambda z.P)(\lambda z.Q)$.

Proof. (i) Use δ-monotonicity $F(\delta MNPQ) = \delta MN(FP)(FQ)$ for $F = \lambda x.xz$.
(ii) By (i) and (η).
(iii) By (ii) applied with $P := \lambda z.P$ and $Q := \lambda z.Q$. $\blacksquare$

5.1.11. Lemma. (i) $\delta MNPQ = Q \ \vdash_\delta \ \delta MNQP = P$.
(ii) $\delta MNPQ = Q,\ \delta MNQR = R \ \vdash_\delta \ \delta MNPR = R$.
(iii) $\delta MNPQ = Q,\ \delta MNUV = V \ \vdash_\delta \ \delta MN(PU) = (QV)$.

Proof. (i)
$$\begin{array}{rcll}P & = & \delta MNPP & \\ & = & \delta MN(\mathsf{K}PQ)(\mathsf{K}_*QP) & \\ & = & \delta MN(\delta MNPQ)(\delta MNQP), & \text{by } (\delta\text{-transitivity}),\\ & = & \delta MNQ(\delta MNQP), & \text{by assumption},\\ & = & \delta MNQ(\mathsf{K}_*QP), & \text{by } (\delta\text{-transitivity}),\\ & = & \delta MNQP. & \end{array}$$

(ii) $\begin{aligned} R &= \delta MNQR, && \text{by assumption,}\\ &= \delta MN(\delta MNPQ)(\delta MNQR), && \text{by assumption,}\\ &= \delta MN(\mathsf{K}PQ)(\mathsf{K}_*QR), && \text{by } (\delta\text{-transitivity}),\\ &= \delta MNPR. \end{aligned}$

(iii) Assuming $\delta MNPQ = Q$ and $\delta MNUV = V$ we obtain by (δ-transitivity) applied twice $\delta MN(PU)(QV) = (QV)$ and $\delta MN(PU)(QV) = (QV)$. Hence the result $\delta MN(PU)(QV) = QV$ follows by (ii). ■

5.1.12. Proposition (Deduction theorem I). *Let $\mathcal{E}$ be a set of equations. Then*

$$\mathcal{E}, M = N \vdash_\delta P = Q \;\Rightarrow\; \mathcal{E} \vdash_\delta \delta MNPQ = Q.$$

Proof. By induction on the derivation of $\mathcal{E}, M = N \vdash_\delta P = Q$. If $P = Q$ is an axiom of δ or in $\mathcal{E}$, then $\mathcal{E} \vdash_\delta P = Q$ and hence $\mathcal{E} \vdash_\delta \delta MNPQ = \delta MNQQ = Q$. If $(P = Q) \equiv (M = N)$, then $\mathcal{E} \vdash_\delta \delta MNPQ \equiv \delta MNMN = N \equiv N$. If $P = Q$ follows directly from $\mathcal{E}, N = M \vdash_\delta V = U$, then by the induction hypothesis one has $\mathcal{E} \vdash_\delta \delta MNQP = U$. But then by lemma 5.1.11(i) one has $\mathcal{E} \vdash_\delta \delta MNPQ = Q$. If $P = Q$ follows by (transitivity), (cong-app) or (cong-abs), then the result follows from the induction hypothesis, using lemma 5.1.11(ii), (iii) or lemma 5.1.10(iii) respectively. ■

5.1.13. Lemma. **[extra]**
(i) $\vdash_\delta \delta MN(\delta MNPQ)P = P$.
(ii) $\vdash_\delta \delta MNQ(\delta MNPQ) = Q$.

Proof. (i) By (δ-transitivity) one has

$$\delta MN(\delta MNPQ)P = \delta MN(\mathsf{K}PQ)P = \delta MNPP = P.$$

(ii) Similarly. ■

5.1.14. Lemma. (i) $\vdash_\delta \delta\mathsf{K}\mathsf{K}_* = \mathsf{K}_*$;
(ii) $\vdash_\delta \delta MN\mathsf{K}\mathsf{K}_* = \delta MN$;
(iii) $\vdash_\delta \delta(\delta MN)\mathsf{K}_*PQ = \delta MNQP$;
(iv) $\vdash_\delta \delta(\delta MN\mathsf{K}\mathsf{K}_*)\mathsf{K}_*(\delta MNPQ)Q = P$.

Proof. (i) $\begin{aligned} \mathsf{K}_* &= \delta\mathsf{K}\mathsf{K}_*\mathsf{K}\mathsf{K}_*, && \text{by } (\delta\text{-hypothesis}),\\ &= \lambda ab.\delta\mathsf{K}\mathsf{K}_*(\mathsf{K}ab)(\mathsf{K}_*ab), && \text{by } (\eta) \text{ and lemma 5.1.10(ii)},\\ &= \lambda ab.\delta\mathsf{K}\mathsf{K}_*ab\\ &= \delta\mathsf{K}\mathsf{K}_*, && \text{by } (\eta). \end{aligned}$

(ii) $\begin{aligned} \delta MN\mathsf{K}\mathsf{K}_* &= \delta MN(\delta MN)(\delta MN), && \text{by } (\delta\text{-transitivity}),\\ &= \delta MN, && \text{by } (\delta\text{-reflexivity}). \end{aligned}$

(iii) $\begin{aligned} \delta MNQP &= \delta MN(\delta\mathsf{K}\mathsf{K}_*PQ)(\delta\mathsf{K}_*\mathsf{K}_*PQ), && \text{by (i), } (\delta\text{-identity}),\\ &= \delta MN(\delta(\delta MN)\mathsf{K}_*PQ)(\delta(\delta MN)\mathsf{K}_*PQ), && \text{by } (\delta\text{-transitivity}),\\ &= \delta(\delta MN)\mathsf{K}_*PQ, && \text{by } (\delta\text{-reflexivity}). \end{aligned}$

(iv) By (ii) and (iii) we have

$$\delta(\delta MN\mathsf{K}\mathsf{K}_*)\mathsf{K}_*(\delta MNPQ)Q = \delta(\delta MN)\mathsf{K}_*(\delta MNPQ)Q = \delta MNQ(\delta MNPQ).$$

Therefore we are done by lemma 5.1.13(ii). ■

5.1.15. Lemma. [**eqded**] (i) $\delta MN = \mathsf{K} \vdash_\delta M = N$;
(ii) $\delta MN\mathsf{K}_*\mathsf{K} = \mathsf{K}_* \vdash_\delta M = N$.
(iii) $\delta(\delta MN\mathsf{K}\mathsf{K}_*)\mathsf{K}_*\mathsf{K}\mathsf{K}_* = \mathsf{K}_* \vdash_\delta M = N$.

Proof. (i) $M = \mathsf{K}MN = \delta MNMN = N$, by assumption and ($\delta$-hypothesis).
(ii) Suppose $\delta MN\mathsf{K}_*\mathsf{K} = \mathsf{K}_*$. Then by lemma 5.1.10(i) and (δ-hypothesis)

$$M = \mathsf{K}_*NM = \delta MN\mathsf{K}_*\mathsf{K}NM = \delta MN(\mathsf{K}_*NM)(\mathsf{K}NM) = \delta MNMN = N.$$

(iii) By lemma 5.1.14(ii) and (iii)

$$\delta(\delta MN\mathsf{K}\mathsf{K}_*)\mathsf{K}_*\mathsf{K}\mathsf{K}_* = \delta(\delta MN)\mathsf{K}_*\mathsf{K}\mathsf{K}_* = \delta MN\mathsf{K}_*\mathsf{K}.$$

Hence by (ii) we are done. ■

5.1.16. Exercise. Prove in δ the following equations.
(i) $\delta MN\mathsf{K}_*\mathsf{K} = \delta(\delta MN)\mathsf{K}_*$.
(ii) $\delta(\lambda z.\delta(Mz)(Nz))(\lambda z.\mathsf{K}) = \delta MN$.
[Hint. Start observing that $\delta(Mz)(Nz)(Mz)(Nz) = Nz$.]

Now we are able to prove the conservativity of Δ over δ.

5.1.17. Theorem. [**conservativity**] *For equations $\mathcal{E}, E$ and formulas Γ, φ of Δ one has the following.*
(i) $\Gamma \vdash_\Delta \varphi \iff \Gamma^+ \vdash_\delta \varphi^+$.
(ii) $\mathcal{E} \vdash_\Delta E \iff \mathcal{E} \vdash_\delta E$.

Proof. (i) ($\Rightarrow$) Suppose $\Gamma \vdash_\Delta \varphi$. By induction on this proof in Δ we show that $\Gamma^+ \vdash_\delta \varphi^+$. Case 1. φ is in Γ. Then $\varphi^+ \in \Gamma^+$ and we are done. Case 2. φ is an equational axiom. Then the result holds since δ has more equational axioms than Δ. Case 3. φ follows from an equality rule in Δ. Then the result follows from the induction hypothesis and the fact that δ has the same equational deduction rules. Case 4. φ follows from $\Gamma \vdash_\Delta \psi$ and $\Gamma \vdash_\Delta \psi \supset \varphi$. By the induction hypothesis $\Gamma^+ \vdash_\delta (\psi \supset \varphi)^+ \equiv (\delta MNPQ = Q)$ and $\Gamma^+ \vdash_\delta \psi^+ \equiv (M = N)$, where $\psi^+ \equiv (M = N)$ and $\varphi^+ \equiv (P = Q)$. Then $\Gamma^+ \vdash_\delta U = \delta MMPQ = Q$, i.e. $\Gamma^+ \vdash_\delta \varphi^+$. Case 5. $\varphi \equiv (\chi \supset \psi)$ and follows by an ($\supset$-intro) from $\Gamma, \chi \vdash_\Delta \psi$.By the induction hypothesis $\Gamma^+, \chi^+ \vdash_\delta \psi^+$ and we can apply the deduction theorem 5.1.12. In the cases that φ is introduced by a ($\forall$-elim) or ($\forall$-intro), the result follows easily from the induction hypothesis and axiom (β) or the rule (cong-abs). One needs that $\mathrm{FV}(\Gamma) = \mathrm{FV}(\Gamma^+)$. Case 8. $\varphi \cong (M = N)$ and follows from $\Gamma, M \neq N \vdash_\Delta \bot$ using the rule (classical). By the induction hypothesis $\Gamma^+, (M \neq N)^+ \vdash_\delta \mathsf{K} = \mathsf{K}_*$. By the deduction theorem it follows that $\Gamma^+ \vdash_\delta \delta(\delta MN\mathsf{K}\mathsf{K}_*)\mathsf{K}_*\mathsf{K}\mathsf{K}_* = \mathsf{K}_*$. Hence we are done by lemma 5.1.15(iii). Case 9. φ is the axiom $(M = N \supset \delta MNPQ = P)$. Then φ^+ is provable in δ by lemma 5.1.13(i). Case 10. φ is the axiom $(M \neq N \supset \delta MNPQ = Q)$. Then φ^+ is provable in δ by lemma 5.1.14(iv).
($\Leftarrow$) By the fact that δ is a subtheory of Δ and theorem 5.1.9.
(ii) By (i) and the fact that $E^+ \equiv E$. ■

n-th Order Logic

In this subsection some results will be sketched but not (completely) proved.

5.1.18. Definition. (i) The system Δ without the two delta rules is denoted by Δ^-.

(ii) $\Delta(n)$ is Δ^- extended by the two delta rules restricted to $\delta_{A,B}$'s with $\mathtt{rank}(A) \leq n$.

(iii) Similarly $\delta(n)$ is the theory δ in which only terms $\delta_{A,B}$ are used with $\mathtt{rank}(\mathtt{A}) \leq n$.

(iv) The rank of a formula φ is $\mathtt{rank}(\varphi) = \mathtt{max}\{\,\mathtt{rank}(\delta) \mid \delta \ \mathtt{occurs\ in}\ \varphi\}$.

In the applications section we will show that $\Delta(n)$ is essentially n-th order logic.

The relation between Δ and δ that we have seen also holds level by level. We will only state the relevant results, the proofs being similar, but using as extra ingredient the proof-theoretic normalization theorem for Δ. This is necessary, since a proof of a formula of rank n may use *a priori* formulas of arbitrarily high rank. By the normalization theorem this is not the case.

A natural deduction is called *normal* if there is no ($\forall$-intro) immediately followed by a ($\forall$-elim), nor a ($\supset$-intro) immediately followed by a ($\supset$-elim). If a reduction is not normal, then one can make Prawitz deductions reduction as follows.

$$\dfrac{\dfrac{\dfrac{\vdots\,\Sigma}{\varphi}}{\forall x.\varphi}}{\varphi[x := M]} \quad \Rightarrow \quad \dfrac{\vdots\,\Sigma[x := M]}{\varphi[x := M]}$$

$$\dfrac{\vdots\,\Sigma_2 \qquad \dfrac{\begin{array}{c}[\varphi]\\ \vdots\,\Sigma_1\\ \psi\end{array}}{\varphi \supset \psi}}{\varphi}\ \psi \quad \Rightarrow \quad \dfrac{\begin{array}{c}\vdots\,\Sigma_2\\ [\varphi]\\ \vdots\,\Sigma_1\\ \psi\end{array}}{\psi}$$

5.1.19. Theorem. *Every reduction sequence of Δ deductions terminates in a unique normal form.*

Proof. This has been proved essentially in Prawitz [1965]. The higher order quantifiers pose no problems. ■

Notation. (i) Let Γ_δ be the set of universal closures of

$$\begin{array}{rcl}
\delta mmpq & = & p,\\
\delta mnpp & = & p,\\
\delta mnmn & = & n,\\
\delta mnpq & = & \delta nmpq,\\
f(\delta mnpq) & = & \delta mn(fp)(fq),\\
\delta mm(p(\delta mn))(q(\delta mn)) & = & \delta mn(p\mathsf{K})(q\mathsf{K}_*).
\end{array}$$

(ii) Let $\Gamma_{\delta(n)} = \{\varphi \in \Gamma_\delta \mid \mathtt{rank}(\varphi) \leq n\}$.

5.1.20. PROPOSITION (Deduction theorem II). *Let $\mathcal{S}$ be a set of equations or negations of equations in Δ. Then for every n*

(i) $\mathcal{S}, \Gamma_{\delta(n)}, M = N \vdash_{\Delta(n)} P = Q \;\Rightarrow\; \mathcal{S}, \Gamma_{\delta(n)} \vdash_{\Delta(n)} \delta MNPQ = Q$.

(ii) $\mathcal{S}, \Gamma_{\delta(n)}, M \neq N \vdash_{\Delta(n)} P = Q \;\Rightarrow\; \mathcal{S}, \Gamma_{\delta(n)} \vdash_{\Delta(n)} \delta MNPQ = P$.

PROOF. In the same stayle as the proof of proposition 5.1.12, but now using the normalization theorem 5.1.19. ■

5.1.21. LEMMA. **[eqnoneq]** *Let $\mathcal{S}$ be a set of equations or negations of equations in Δ. Let $\mathcal{S}^*$ be $\mathcal{S}$ with each $M \neq N$ replaced by $\delta MN\mathsf{KK}_* = \mathsf{K}_*$. Then we have the following.*

(i) $\mathcal{S}, M = N \vdash_{\Delta(n)} P = Q \;\Rightarrow\; \mathcal{S}^* \vdash_{\delta(n)} \delta MNPQ = Q$.

(ii) $\mathcal{S}, M \neq N \vdash_{\Delta(n)} P = Q \;\Rightarrow\; \mathcal{S}^* \vdash_{\delta(n)} \delta MNPQ = P$.

PROOF. By induction on derivations. ■

5.1.22. THEOREM. $\mathcal{E} \vdash_{\Delta(n)} E \iff \mathcal{E} \vdash_{\delta(n)} E$.

PROOF. ($\Rightarrow$) By taking $\mathcal{S} = \mathcal{E}$ and $M \equiv N \equiv x$ in lemma 5.1.21(i) one obtains $\mathcal{E} \vdash_{\delta(n)} \delta xxPQ = Q$. Hence $\mathcal{E} \vdash_{\delta(n)} P = Q$, by ($\delta$-identity). ($\Leftarrow$) Trivial. ■

5.1.23. THEOREM. (i) *Let $\mathtt{rank}(\mathcal{E}, M = N) \leq 1$. Then*

$$\mathcal{E} \vdash_\Delta M = N \iff \mathcal{E} \vdash_{\delta(1)} M = N.$$

(ii) *Let Γ, A be a first-order sentences. Then*

$$\Gamma \vdash_\Delta A \iff \Gamma \vdash_{\delta(1)} A^+.$$

PROOF. See Statman [2000]. ■

In Statman [2000] it is also proved that $\Delta(0)$ is decidable. Since $\Delta(n)$ for $n \geq 1$ is at least first order predicate logic, these systems are undecidable. It is observed in Gödel [1931] that the consistency of $\Delta(n)$ can be proved in $\Delta(n+1)$.

5.2. Surjective pairing 21.2.2008:897

A *pairing* on a set X consists of three maps π, π_1, π_1 such that

$$\begin{aligned} \pi &: X \to X \to X \\ \pi_i &: X \to X \end{aligned}$$

and for all $x_1, x_2 \in X$ one has

$$\pi_i(\pi x_1 x_2) = x_i.$$

Using a pairing one can pack two or more elements of X into one element:

$$\begin{aligned} \pi xy &\in X, \\ \pi x(\pi yz) &\in X. \end{aligned}$$

A pairing on X is called *surjective* if one also has for all $x \in X$

$$\pi(\pi_1 x)(\pi_2 x) = x.$$

This is equivalent to saying that every element of X is a pair. Using a (surjective) pairing datastructures can be encoded.

The main results in this section are the following. 1. After adding a surjective pairing to $\lambda^o_\rightarrow$ the resulting system λ_{SP} becomes Hilbert-Post complete. This means that an equation between terms is either provable or inconsistent. 2. Every recursively enumerable set $\mathcal{X}$ of terms that is closed under provable equality is Diophantine, i.e. satisfies for some terms F, G

$$M \in \mathcal{X} \iff \exists N\, FMN = GMN.$$

Both results will be proved by introducing Cartesian monoids and studying freely generated ones.

The system λ_{SP}

We define λ_{SP} as an extension of the simply typed lambda calculus $\lambda^o_\rightarrow$.

5.2.1. Definition. (i) The set of *types* of λ_{SP}, notation $\mathbb{T} = \mathbb{T}(\lambda_{SP})$, is the same as $\mathbb{T}(\lambda^o_\rightarrow)$: $\mathbb{T} = o \mid \mathbb{T}\rightarrow\mathbb{T}$.

(ii) The *terms* of λ_{SP}, notation Λ_{SP} (or $\Lambda_{SP}(A)$ for terms of a certain type A or $\Lambda^\emptyset$, $\Lambda^\emptyset_{SP}(A)$ for closed terms), are obtained from $\lambda^o_\rightarrow$ by adding to the formation of terms the constants $\pi : 1_2 = o^2\rightarrow o$, $\pi_1 : 1$, $\vdash : 1$.

(iii) Equality for λ_{SP} is axiomatized by β, η and the following scheme. For all $M, M_1, M_2 : o$

$$\begin{aligned} \pi_i(\pi M_1 M_2) &= M_i; \\ \pi(\pi_1 M)(\pi_2 M) &= M. \end{aligned}$$

(iv) A notion of reduction SP is introduced on λ_{SP}-terms by the following contraction rules: for all $M, M_1, M_2 : o$

$$\begin{aligned} \pi_i(\pi M_1 M_2) &\rightarrow M_i; \\ \pi(\pi_1 M)(\pi_2 M) &\rightarrow M. \end{aligned}$$

Usually we will consider SP in combination of $\beta\boldsymbol{\eta}$, obtaining $\beta\boldsymbol{\eta}SP$.

5.2.2. Theorem. *The conversion relation $=_{\beta\boldsymbol{\eta}SP}$, generated by the notion of reduction $\beta\boldsymbol{\eta}SP$, coincides with that of the theory λ_{SP}.*

Proof. As usual. ■

For objects of higher type pairing can be defined in terms of $\pi, \pi_1, \vdash$ in the following way.

5.2.3. DEFINITION. For every type $A \in \mathbb{T}$ we define $\pi^A : A \to A \to A, \pi_i : A \to A$ as follows, cf. definition **??**.

$$\begin{array}{rcl} \pi^o & \equiv & \pi; \\ \pi_i^o & \equiv & \pi_i; \\ \pi^{A\to B} & \equiv & \lambda xy\,(A\to B)\lambda z\,A.\pi^B(xz)(yz); \\ \pi_i^{A\to B} & \equiv & \lambda x\,(A\to B)\lambda z\,A.\pi_i^B(xz). \end{array}$$

Sometimes we may suppress type annotations in $\pi^A, \pi_1^A, \vdash^A$, but the types can always and unambiguously be reconstructed from the context.

The defined constants for higher type pairing can easily be shown to be a surjective pairing also.

5.2.4. PROPOSITION. *Let $\pi = \pi^A, \pi_i = \pi_i^A$. Then for $M, M_1, M_2 \in \Lambda_{SP}(A)$*

$$\begin{array}{rcl} \pi(\pi_1 M)(\vdash M) & \twoheadrightarrow_{\beta\boldsymbol{\eta}SP} & M; \\ \pi_i(\pi M_1 M_2) & \twoheadrightarrow_{\beta\boldsymbol{\eta}SP} & M_i, \quad (i = 1, 2). \end{array}$$

PROOF. By induction on the type A. ■

Note that the above reductions may involve more than one step, typically additional $\beta\boldsymbol{\eta}$-steps.

Now we will show that the notion of reduction $\beta\boldsymbol{\eta}SP$ is confluent.

5.2.5. LEMMA. *The notion of reduction $\beta\boldsymbol{\eta}SP$ satisfies* WCR.

PROOF. By the critical pair lemma of Mayr and Nipkow [1998]. But a simpler argument is possible, since SP reductions only reduce to terms of type o that did already exist and hence cannot create any redexes. ■

5.2.6. LEMMA. (i) *The notion of reduction SP is SN.*
(ii) *If $M \twoheadrightarrow_{\beta\boldsymbol{\eta}SP} N$, then there exists P such that $M \twoheadrightarrow_{\beta\boldsymbol{\eta}} P \twoheadrightarrow_{SP} N$.*
(iii) *The notion of reduction $\beta\boldsymbol{\eta}SP$ is SN.*

PROOF. (i) Since SP-reductions are strictly decreasing.

(ii) Show $M \to_{SP} L \to_{\beta\boldsymbol{\eta}} N \;\Rightarrow\; \exists L'\, M \twoheadrightarrow_{\beta\boldsymbol{\eta}} L' \twoheadrightarrow_{\beta\boldsymbol{\eta}SP} N$. Then (ii) follows by a staircase diagram chase.

(iii) By (i), the fact that $\beta\boldsymbol{\eta}$ is SN and a staircase diagram chase, possible by (ii). ■

5.2.7. PROPOSITION. *$\beta\boldsymbol{\eta}SP$ is CR.*

PROOF. By lemma 5.2.6(iii) and Newman's Lemma 5.3.14. ■

5.2.8. DEFINITION. (i) An *SP-retraction pair from A to B* is a pair of terms $M\, A\to B$ and $N\, B\to A$ such that $N \circ M =_{\beta\boldsymbol{\eta}SP} \mathsf{I}_A$.

(ii) A is a *SP-retract* of B, notation $A \lhd_{SP} B$, if there is an SP-retraction pair between A and B.

The proof of the following result is left as an exercise to the reader.

5.2.9. PROPOSITION. *Define types N_n as follows. $N_0 \equiv o$ and $N_{n+1} \equiv N_n \to N_n$. Then for every type A, one has $A \lhd_{SP} N_{rank(A)}$.*

Cartesian monoids

We start with the definition of a Cartesian monoid, introduced in Scott [1980] and, independently, in Lambek [1980].

5.2.10. DEFINITION. (i) A *Cartesian monoid* is a structure

$$\mathcal{C} = \langle \mathcal{M}, *, I, L, R, \langle \cdot, \cdot \rangle \rangle$$

such that $(\mathcal{M}, *, I)$ is a monoid ($*$ is associative and I is a two sided unit), $L, R \in \mathcal{M}$ and $\langle \cdot, \cdot \rangle : \mathcal{M}^2 \to \mathcal{M}$ and satisfy

$$\begin{array}{rcl} L * \langle x, y \rangle & = & x \\ R * \langle x, y \rangle & = & y \\ \langle x, y \rangle * z & = & \langle x * z, y * z \rangle \\ \langle L, R \rangle & = & I \end{array}$$

(ii) $\mathcal{M}$ is called *trivial* if $L = R$.

Note that if $\mathcal{M}$ is trivial, then it consists of only one element: for all $x, y \in \mathcal{M}$

$$x = L\langle x, y \rangle = R\langle x, y \rangle = y.$$

5.2.11. LEMMA. *The last axiom of the Cartesian monoids can be replaced equivalently by the surjectivity of the pairing:*

$$\langle L * x, R * x \rangle = x.$$

PROOF. First suppose $\langle L, R \rangle = I$. Then $\langle L * x, R * x \rangle = \langle L, R \rangle * x = I * x = x$. Conversely suppose $\langle L * x, R * x \rangle = x$, for all x. Then $\langle L, R \rangle = \langle L * \mathsf{I}, R * \mathsf{I} \rangle = \mathsf{I}$. ■

5.2.12. LEMMA. *Let $\mathcal{M}$ be a Cartesian monoid. Then for all $x, y \in \mathcal{M}$*

$$L * x = L * y \ \& \ R * x = R * y \ \Rightarrow \ x = y.$$

PROOF. $x = \langle L * x, R * x \rangle = \langle L * y, R * y \rangle = y$. ■

A first example of a Cartesian monoid has as carier set the closed $\beta\boldsymbol{\eta}SP$-terms of type $1 = o \to o$.

5.2.13. Definition. Write for $M, N \in \Lambda^{\emptyset}_{SP}(1)$

$$\begin{array}{rcl} \langle M, N\rangle & \equiv & \pi^1 MN; \\ M \circ N & \equiv & \lambda x\, o.M(Nx); \\ \mathsf{I} & \equiv & \lambda x\, o.x; \\ \mathsf{L} & \equiv & \pi^o_1; \\ \mathsf{R} & \equiv & \vdash^o . \end{array}$$

Define

$$\mathcal{C}^0 = \langle \Lambda^{\emptyset}_{SP}(1)/ =_{\beta\boldsymbol{\eta}SP}, \circ, \mathsf{I}, \mathsf{L}, \mathsf{R}, \langle\cdot,\cdot\rangle\rangle.$$

The reason to call this structure $\mathcal{C}^0$ and not $\mathcal{C}^1$ is that we will generalize it to $\mathcal{C}^n$ being based on terms of the type $1^n \to 1$.

5.2.14. Proposition. *$\mathcal{C}^0$ is a non-trivial Cartesian monoid.*

Proof. For $x, y, z\, 1$ the following equations are valid in λ_{SP}.

$$\begin{array}{rcl} \mathsf{I} \circ x & = & x; \\ x \circ \mathsf{I} & = & x; \\ \mathsf{L} \circ \langle x, y\rangle & = & x; \\ \mathsf{R} \circ \langle x, y\rangle & = & y; \\ \langle x, y\rangle \circ z & = & \langle x \circ z, y \circ z\rangle; \\ \langle \mathsf{L}, \mathsf{R}\rangle & = & \mathsf{I}. \blacksquare \end{array}$$

The third equation is intuitively right, if we remember that the pairing on type 1 is lifted pointwise from a pairing on type o; that is, $\langle f, g\rangle = \lambda x.\pi(fx)(gx)$.

5.2.15. Example. Let $[\cdot,\cdot]$ be any surjective pairing of natural numbers, with left and right projections $l, r : \mathbb{N} \to \mathbb{N}$. For example, we can take Cantor's well-known bijection[1] from $\mathbb{N}^2$ to $\mathbb{N}$. We can lift the pairing function to the level of functions by putting $\langle f, g\rangle(x) = [f(x), g(x)]$ for all $x \in \mathbb{N}$. Let I be the identity function and let $\circ$ denote function composition. Then

$$\mathcal{N}^1 = \langle \mathbb{N} \to \mathbb{N}, I, \circ, l, r, \langle\cdot,\cdot\rangle\rangle.$$

is a non-trivial Cartesian monoid.

Now we will show that the equalities in the theory of Cartesian monoids are generated by a confluent rewriting system.

5.2.16. Definition. (i) Let T_{CM} be the terms in the signature of Cartesian monoids. Let T^o_{CM} be the closed such terms.

[1] A variant of this function is used in Section 5.3 as a non-surjective pairing function $[x, y] + 1$, such that, deliberaately, 0 does not encode a pair. This variant is specified in detail and explained in Figure 5.2.

(ii) Consider the rewrite system CM on T_{CM} defined as follows.

$$\begin{array}{rcl} L * \langle x, y\rangle & \to & x \\ R * \langle x, y\rangle & \to & y \\ \langle x, y\rangle * z & \to & \langle x * z, y * z\rangle \\ \langle L, R\rangle & \to & I \\ \langle L * x, R * x\rangle & \to & x \\ I * x & \to & x \\ x * I & \to & x \end{array}$$

modulo the associativity axioms (i.e. terms like $f * (g * h)$ and $(f * g) * h$ are considered as the same).

5.2.17. Proposition. (i) *CM is WCR.*
(ii) *CM is SN.*
(iii) *CM is CR.*

Proof. (i) Examine all critical pairs. Modulo associativity there are many such pairs, but they all converge. Consider, as an example, the following reductions:

$$x * z \leftarrow (L * \langle x, y\rangle) * z = L * (\langle x, y\rangle) * z) \to L * \langle x * z, y * z\rangle \to x * z$$

(ii) Interprete CM as integers by putting

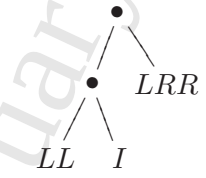
$$\begin{array}{rcll} [\![x]\!]_\rho & = & \rho(x); & \\ [\![e]\!]_\rho & = & 2, & \text{if } e \text{ is } L, R \text{ or } I; \\ [\![e_1 * e_2]\!]_\rho & = & [\![e_1]\!]_\rho . [\![e_2]\!]_\rho; & \\ [\![\langle e_1, e_2\rangle]\!]_\rho & = & [\![e_1]\!]_\rho + [\![e_2]\!]_\rho + 1. & \end{array}$$

Then $[\![\cdot]\!]_\rho$ preserves associativity and

$$e \to_{CM} e' \;\Rightarrow\; [\![e]\!]_\rho > [\![e']\!]_\rho.$$

Therefore CM is SN.
(iii) By (i), (ii) and Newman's lemma 5.3.14. ■

Closed terms in CM-nf can be represented as binary trees with strings of L, Rn (the empty string becomes I) at the leaves, for example



represents $\langle\langle L * L, I\rangle, L * R * R\rangle$. In such trees the subtree corresponding to $\langle L, R\rangle$ will not occur, since this term reduces to I.

The free Cartesian monoids $\mathcal{F}[x_1,\ldots,x_n]$

5.2.18. Definition. (i) The closed term model of the theory of Cartesian monoids consists of $T_{\rm CM}$ modulo $=_{\rm CM}$ and is denoted by $\mathcal{F}$. It is the *free Cartesian monoid* with no generators.

(ii) The *free Cartesian monoid* over the generators $\vec{x}$, notation $\mathcal{F}[\vec{x}]$, is generated from I, L, R and the indeterminates $\vec{x}$ using $*$ and $\langle -,-\rangle$. Usually $\vec{x} = x_1,\ldots,x_n$ is finite.

5.2.19. Proposition. (i) *For all $a, b \in \mathcal{F}$ one has*

$$a \neq b \;\Rightarrow\; \exists c,d\in\mathcal{F}\,[c*a*d = L \;\&\; c*b*d = R].$$

(ii) *$\mathcal{F}$ is simple: every homomorphism $g : \mathcal{F}\to\mathcal{M}$ to a non-trivial Cartesian monoid $\mathcal{M}$ is injective.*

Proof. (i) We can assume that a, b are in normal form. The binary tree part of the normal form is called Δ. The Δ's of a, b can be made to be congruent by expansions of the form $x \leftarrow \langle L*x, R*x\rangle$. The expanded trees are distinct in some leaf, which can be reached by a string of L's and R's joined by $*$. Thus there is such a string, say c, such that $c*a \neq c*b$ and both of these reduce to $\langle\,\rangle$-free strings of L's and R's joined by $*$. We can also assume that neither of these strings is a suffix of the other, since c could be replaced by $L*c$ or $R*c$ (depending on an R or an L just before the suffix). Thus there are $\langle\,\rangle$-free a', b' and integers k, l such that

$$\begin{aligned} c*a*\langle I,I\rangle^k * \langle R,L\rangle^l &= a'*L \quad\text{and}\\ c*b*\langle I,I\rangle^k * \langle R,L\rangle^l &= b'*R \end{aligned}$$

and there exist integers n and m, being the length of a' and of b', respectively, such that

$$\begin{aligned} a'*L*\langle\langle I,I\rangle^n * L, \langle I,I\rangle^m * R\rangle &= L \quad\text{and}\\ b'*R*\langle\langle I,I\rangle^n * L, \langle I,I\rangle^m * R\rangle &= R \end{aligned}$$

Therefore we can set $d = \langle I,I\rangle^k * \langle R,L\rangle^l * \langle\langle I,I\rangle^n * L, \langle I,I\rangle^m * R\rangle$.

(ii) By (i) and the fact that $\mathcal{M}$ is non-trivial. ■

Finite generation of $\mathcal{F}[x_1,\ldots,x_n]$

Now we will show that $\mathcal{F}[x_1,\ldots,x_n]$ is finitely generated as a monoid, i.e. from finitely many of its elements using the operation $*$ only.

5.2.20. Notation. In a monoid $\mathcal{M}$ we define list-like left-associative and right-associative iterated $\langle\,\rangle$-expressions of length > 0 as follows. Let the elements of $\vec{x}$ range over $\mathcal{M}$.

$$\begin{aligned} \langle\langle x\rangle &\equiv x;\\ \langle\langle x_1,\ldots,x_{n+1}\rangle &\equiv \langle\langle\langle x_1,\ldots,x_n\rangle, x_{n+1}\rangle, \qquad n>0;\\ \langle x\rangle\rangle &\equiv x;\\ \langle x_1,\ldots,x_{n+1}\rangle\rangle &\equiv \langle x_1, \langle x_2,\ldots,x_{n+1}\rangle\rangle\rangle, \qquad n>0. \end{aligned}$$

5.2.21. Definition. Define $\mathcal{G} \subseteq \mathcal{F}$ as follows.

$$\mathcal{G} = \{\langle X * L, Y * L * R, Z * R * R\rangle\!\rangle \mid X, Y, Z \in \{L, R, I\}\} \cup \{\langle I, I, I\rangle\!\rangle\}.$$

Let $[\mathcal{G}]$ be the set generated from $\mathcal{G}$ using $*$, i.e. the least subset of $\mathcal{F}$ containing $\mathcal{G}$ and closed under $*$. We will show that $[\mathcal{G}] = \mathcal{F}$.

5.2.22. Lemma. *Define a* string *to be an expression of the form $X_1 * \ldots * X_n$, with $X_i \in \{L, R, I\}$. Then for all strings s, s_1, s_2, s_3 one has the following.*

(i) $\langle s_1, s_2, s_3\rangle\!\rangle \in [\mathcal{G}]$.

(ii) $s \in [\mathcal{G}]$.

Proof. (i) Note that

$$\langle X * L, Y * L * R, Z * R * R\rangle\!\rangle * \langle s_1, s_2, s_3\rangle\!\rangle = \langle X * s_1, Y * s_2, Z * s_3\rangle\!\rangle.$$

Hence, starting from $\langle I, I, I\rangle\!\rangle \in \mathcal{G}$ every triple of strings can be generated because the X, Y, Z range over $\{L, R, I\}$.

(ii) Notice that

$$\begin{aligned} s &= \langle L, R\rangle * s \\ &= \langle L * s, R * s\rangle \\ &= \langle L * s, \langle L, R\rangle * R * s\rangle \\ &= \langle L * s, L * R * s, R * R * s\rangle\!\rangle, \end{aligned}$$

which is in $[\mathcal{G}]$ by (i). ■

5.2.23. Lemma. *Suppose $\langle\!\langle s_1, \ldots, s_n\rangle\!\rangle \in [\mathcal{G}]$. Then*

(i) *$s_i \in [\mathcal{G}]$, for $1 \leq i \leq n$.*

(ii) *$\langle\!\langle s_1, \ldots, s_n, \langle s_i, s_j\rangle\rangle\!\rangle \in [\mathcal{G}]$ for $0 \leq i, j \leq n$.*

(iii) *$\langle\!\langle s_1, \ldots, s_n, X * s_i\rangle\!\rangle \in [\mathcal{G}]$ for $X \in \{L, R, I\}$.*

Proof. (i) By lemma 5.2.22(ii) one has $F_1 \equiv L^{(n-1)} \in [\mathcal{G}]$ and $F_i \equiv R * L^{(n-i)} \in [\mathcal{G}]$. Hence

$$\begin{aligned} s_1 &= F_1 * \langle\!\langle s_1, \ldots, s_n\rangle\!\rangle \in [\mathcal{G}]; \\ s_i &= F_i * \langle\!\langle s_1, \ldots, s_n\rangle\!\rangle \in [\mathcal{G}], \qquad \text{for } i = 2, \ldots, n. \end{aligned}$$

(ii) By lemma 5.2.22(i) one has $\langle I, \langle F_i, F_j\rangle\rangle = \langle I, F_i, F_j\rangle\!\rangle \in [\mathcal{G}]$. Hence

$$\langle\!\langle s_1, \ldots, s_n, \langle s_i, s_j\rangle\rangle\!\rangle = \langle I, \langle F_i, F_j\rangle\rangle * \langle\!\langle s_1, \ldots, s_n\rangle\!\rangle \in [\mathcal{G}].$$

(iii) Similarly $\langle\!\langle s_1, \ldots, s_n, X * s_i\rangle\!\rangle = \langle I, X * F_i\rangle * \langle\!\langle s_1, \ldots, s_n\rangle\!\rangle \in [\mathcal{G}]$. ■

5.2.24. Theorem. *As a monoid, $\mathcal{F}$ is finitely generated. In fact $\mathcal{F} = [\mathcal{G}]$.*

Proof. Let $e \in \mathcal{F}$. Then there is a formation sequence $e_1 \equiv L, e_2 \equiv R, e_3 \equiv I, \ldots, e_n \equiv e$ such that for each $4 \leq k \leq n$ there are $i, j < k$ such that $e_k \equiv \langle e_i, e_j\rangle$ or $e_k \equiv X * e_i$, with $X \in \{L, R, I\}$.

We have $I = \langle L, R\rangle = \langle\!\langle L, R\rangle\!\rangle \in [\mathcal{G}]$. By lemma 5.2.23(ii), (iii) it follows that

$$\langle\!\langle e_1, e_2, e_3, \ldots, e_n\rangle\!\rangle \in [\mathcal{G}].$$

Therefore by (i) of that lemma $e \equiv e_n \in [\mathcal{G}]$.

The following corollary is similar to a result of Böhm, who showed that the monoid of untyped lambda terms has two generators, see Barendregt [1984].

5.2.25. COROLLARY. (i) *Let $\mathcal{M}$ be a finitely generated cartesian monoid. Then $\mathcal{M}$ is generated by two of its elements.*

(ii) *$\mathcal{F}[x_1,\ldots,x_n]$ is generated by two elements.*

PROOF. (i) Let $\mathcal{G} = \{g_1,\ldots,g_n\}$ be the set of generators of $\mathcal{M}$. Then $\mathcal{G}$ and hence $\mathcal{M}$ is generated by R and $\langle\!\langle g_1,\ldots,g_n,L\rangle$.

(ii) $\mathcal{F}[\vec{x}]$ is generated by $\mathcal{G}$ and the $\vec{x}$, hence by (i) by two elements. ■

Invertibility in $\mathcal{F}$

5.2.26. DEFINITION. (i) Let $\mathcal{L}$ ($\mathcal{R}$) be the submonoid of the right (left) invertible elements of $\mathcal{F}$

$$\begin{aligned}\mathcal{L} &= \{a\in\mathcal{F} \mid \exists b\in\mathcal{F}\ b*a = I\};\\ \mathcal{R} &= \{a\in\mathcal{F} \mid \exists b\in\mathcal{F}\ a*b = I\}.\end{aligned}$$

(ii) Let $\mathcal{I}$ be the subgroup of $\mathcal{F}$ consisting of invertible elements

$$\mathcal{I} = \{a\in\mathcal{F} \mid \exists b\in\mathcal{F}\ a*b = b*a = I\}.$$

It is easy to see that $\mathcal{I} = \mathcal{L}\cap\mathcal{R}$.

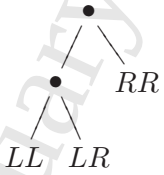
5.2.27. EXAMPLES. (i) $L, R\in\mathcal{L}$, since both have the right inverse $\langle I, I\rangle$.

(ii) $a = \langle\langle L,R\rangle, L\rangle$ having as 'tree'

has as left inverse $b = \langle R, RL\rangle$, where we do not write the $*$ in strings.

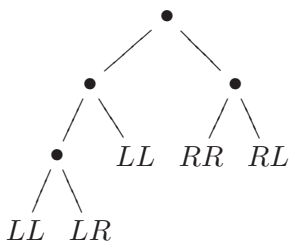
(iii)  has no left inverse, since "R cannot be obtained".

(iv) $a = \langle\langle RL, LL\rangle, RR\rangle$ having the following tree

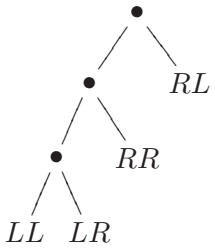


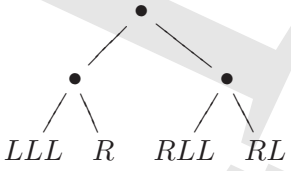
has the following right inverses $b = \langle\langle\langle RL, LL\rangle, \langle c, R\rangle\rangle, R\rangle$. Indeed

$$a*b = \langle\langle RLb, LLb\rangle, RRb\rangle = \langle\langle LL, RL\rangle, R\rangle = \langle L, R\rangle = I.$$

(v) 

has no right inverse, as "LL occurs twice".

(vi) 

has a two-sided inverse, as "all strings of two letters" occur exactly once, the inverse being 

.

For normal forms $f \in \mathcal{F}$ we have the following characterizations.

5.2.28. Proposition. (i) *f has a right inverse if and only if f can be expanded (by replacing x by $\langle Lx, Rx\rangle$) so that all of its strings at the leaves have the same length and none occurs more than once.*

(ii) *f has a left inverse if and only if f can be expanded so that all of its strings at the leaves have the same length, say n, and each of the possible 2^n strings of this length actually occurs.*

(iii) *f is doubly invertible if and only if f can be expanded so that all of its strings at the leaves have the same length, say n, and each of the possible 2^n strings of this length occurs exactly once.*

Proof. This is clear from the examples. ■

The following terms are instrumental to generate $\mathcal{I}$ and $\mathcal{R}$.

5.2.29. Definition.

$$\begin{aligned} B_n &= \langle LR^0, \ldots, LR^{n-1}, LLR^n, RLR^n, RR^n\rangle\!\rangle; \\ C_0 &= \langle R, L\rangle \\ C_{n+1} &= \langle LR^0, \ldots, LR^{n-1}, LRR^n, LR^n, RRR^n\rangle\!\rangle. \end{aligned}$$

5.2.30. Proposition. (i) $\mathcal{I} = [\{B_n \mid n \in \mathbb{N}\} \cup \{C_n \mid n \in \mathbb{N}\}]$.

(ii) $\mathcal{R} = L * \mathcal{I} = R * \mathcal{I}$.

5.2.31. Remark. The B_n alone generate the so-called Thompson-Freyd-Heller group, see exercise 5.6.25.

5.2.32. PROPOSITION. *If $f(\vec{x})$ and $g(\vec{x})$ are distinct members of $\mathcal{F}[\vec{x}]$, then there exists $\vec{h} \in \mathcal{F}$ such that $f(\vec{h}) \neq g(\vec{h})$. We say that $\mathcal{F}[\vec{x}]$ is* separable.

PROOF. Suppose that $f(\vec{x})$ and $g(\vec{x})$ are distinct normal members of $\mathcal{F}[\vec{x}]$. We shall find $\vec{h}$ such that $f(\vec{h}) \neq g(\vec{h})$. First remove subexpressions of the form $L * x_i * h$ and $R * x_j * h$ by substituting $\langle y, z\rangle$ for x_i, x_j and renormalizing. This process terminates, and is invertible by substituting $L * x_i$ for y and $R * x_j$ for z. Thus we can assume that $f(\vec{x})$ and $g(\vec{x})$ are distinct normal and without subexpressions of the two forms above. Indeed, expressions like this can be recursively generated as a string of x_i's followed by a string of L's and R's, or as a string of x_i's followed by a single $\langle\ \rangle$ of expressions of the same form. Let m be a large number relative to $f(\vec{x}), g(\vec{x})$ $(> \#f(\vec{x}), \#g(\vec{x})$, where $\#t$ is the number of symbols in t.) For each positive integer i, set

$$h_i = \langle\langle R^m, \ldots, R^m, I\rangle\rangle, R^m\rangle$$

where the right-associative $\langle\ \rangle$-expression contains i times R^m. We claim that both $f(\vec{x})$ and $g(\vec{x})$ can be reconstructed from the normal forms of $f(\vec{h})$ and $g(\vec{h})$, so that $f(\vec{h}) \neq g(\vec{h})$.

Define $d_r(t)$, for a normal $t \in \mathcal{F}$, as follows.

$$\begin{array}{rcll} d_r(\vec{w}) & = & 0 & \text{if } \vec{w} \text{ is a string of } L, R\text{'s;} \\ d_r(\langle t, s\rangle & = & d_r(s). & \end{array}$$

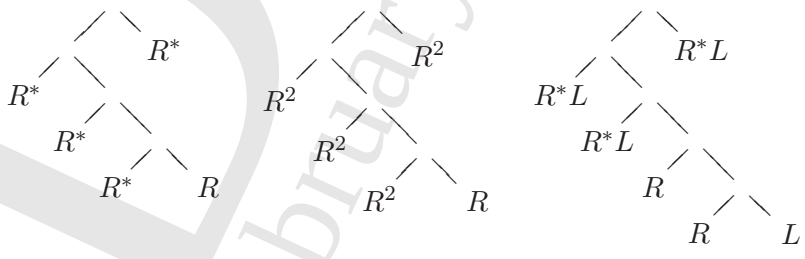
Note that if t is a normal member of $\mathcal{F}$ and $d_r(t) < m$, then

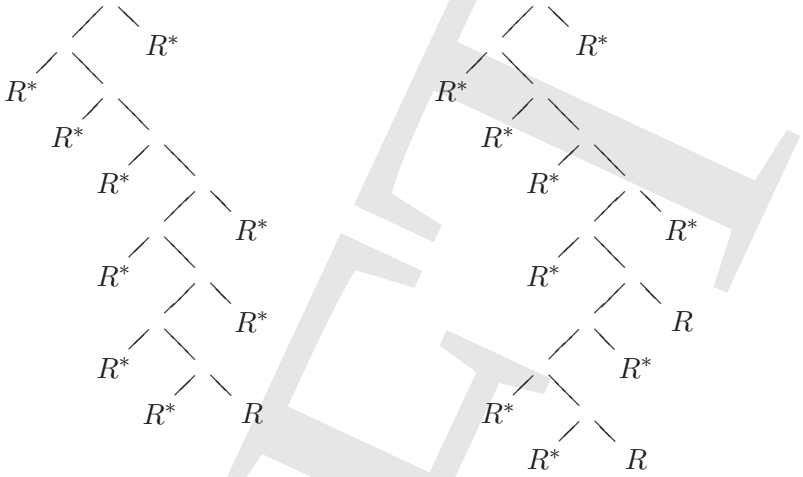
$$h_i * t =_{\mathrm{CM}} \langle\langle t', \ldots, t', t\rangle\rangle, t'\rangle,$$

where $t' \equiv R^m t$ is $\langle\ \rangle$-free. Also note that if s is the CM-nf of $h_i * t$, then $d_r(s) = 1$. The normal form of, say, $f(\vec{h})$ can be computed recursively bottom up as in the computation of the normal form of $h_i * t$ above. In order to compute back $f(\vec{x})$ we consider several examples.

$$\begin{array}{rcl} f_1(\vec{x}) & = & x_3 R; \\ f_2(\vec{x}) & = & \langle\langle R^2, R^2, R^2, R\rangle\rangle, R^2\rangle; \\ f_3(\vec{x}) & = & \langle R, R, L\rangle\rangle \\ f_4(\vec{x}) & = & x_3 x_1 x_2 R; \\ f_5(\vec{x}) & = & x_3 x_1 \langle x_2 R, R\rangle. \end{array}$$

Then $f_1(\vec{h}), \ldots, f_5(\vec{h})$ have as trees respectively





In these trees the R^* denote long sequences of R's of possibly different lengths. ■

Cartesian monoids inside λ_{SP}

Remember $\mathcal{C}^0 = \langle \Lambda^{\emptyset}_{SP}(1)/ =_{\beta\boldsymbol{\eta}SP}, \circ, \mathsf{I}, \mathsf{L}, \mathsf{R}, \langle \cdot, \cdot \rangle \rangle$.

5.2.33. PROPOSITION. *There is a surjective homomorphism $h : \mathcal{F} \to \mathcal{C}^0$.*

PROOF. Now if $M : 1$ is a closed term and in long $\beta\boldsymbol{\eta}SP$ normal form, then M has one of the following shapes: $\lambda a.a$, $\lambda a.\pi X_1 X_2$, $\lambda a.\pi_i X$ for $i = 1$ or $i = 2$. Then we have $M \equiv \mathsf{I}$, $M = \langle \lambda a.X_1, \lambda a.X_2 \rangle$, $M = \mathsf{L} \circ (\lambda a.X)$ or $M = \mathsf{R} \circ (\lambda a.X)$, respectively. Since the terms $\lambda a.X_i$ are smaller than M, this yields an inductive definition of the set of closed terms of λ_{SP} modulo $=$ in terms of the combinators $\mathsf{I}, \mathsf{L}, \mathsf{R}, \langle\ \rangle, \circ$. Thus the elements of $\mathcal{C}^0$ are generated from $\{\mathsf{I}, \circ, \mathsf{L}, \mathsf{R}, \langle \cdot, \cdot \rangle\}$ in an algebraic way. Now define

$$\begin{aligned} h(I) &= \mathsf{I}; \\ h(L) &= \mathsf{L}; \\ h(R) &= \mathsf{R}; \\ h(\langle a, b \rangle) &= \langle h(a), h(b) \rangle; \\ h(a * b) &= h(a) \circ h(b). \end{aligned}$$

Then h is a surjective homomorphism. ■

Now we will show in two different ways that this homomorphism is in fact injective and hence an isomorphism.

5.2.34. THEOREM. $\mathcal{F} \cong \mathcal{C}^0$.

PROOF 1. We will show that the homomorphism h in proposition 5.2.33 is injective. By a careful examination of CM-normal forms one can see the following. Each expression can be rewritten uniquely as a binary tree whose nodes

correspond to applications of $\langle\cdot,\cdot\rangle$ with strings of L's and R's joined by $*$ at its leaves (here I counts as the empty string) and no subexpressions of the form $\langle L*e, R*e\rangle$. Thus

$$a \neq b \;\Rightarrow\; a^{\mathtt{nf}} \not\equiv b^{\mathtt{nf}} \;\Rightarrow\; h(a^{\mathtt{nf}}) \neq h(b^{\mathtt{nf}}) \;\Rightarrow\; h(a) \neq h(b),$$

so h is injective. $\blacksquare_1$

PROOF 2. By proposition 5.2.19. $\blacksquare_2$

The structure $\mathcal{C}^0$ will be generalized as follows.

5.2.35. DEFINITION. Consider the type $1^n{\to}1 = (o{\to}o)^n{\to}o{\to}o$. Define

$$\mathcal{C}^n = \langle \Lambda^{\emptyset}_{SP}(1^n{\to}1)/ =_{\beta\boldsymbol{\eta}SP}, \mathsf{I}_n, \mathsf{L}_n, \mathsf{R}_n, \circ_n, \langle -,-\rangle_n\rangle,$$

where writing $\vec{x} = x_1, \ldots, x_n$ 1

$$\begin{array}{rcl}
\langle M, N\rangle_n &=& \lambda\vec{x}.\langle M\vec{x}, N\vec{x}\rangle;\\
M \circ_n N &=& \lambda\vec{x}.(M\vec{x}) \circ (N\vec{x});\\
\mathsf{I}_n &=& \lambda\vec{x}.\mathsf{I};\\
\mathsf{L}_n &=& \lambda\vec{x}.\mathsf{L};\\
\mathsf{R}_n &=& \lambda\vec{x}.\mathsf{R}.
\end{array}$$

5.2.36. PROPOSITION. *$\mathcal{C}^n$ is a non-trivial Cartesian monoid.*

PROOF. Easy. $\blacksquare$

5.2.37. PROPOSITION. $\mathcal{C}^n \cong \mathcal{F}[x_1, \ldots, x_n]$.

PROOF. As before, let $h_n : \mathcal{F}[\vec{x}]{\to}\mathcal{C}^n$ be induced by

$$\begin{array}{rcllcl}
h_n(x_i) &=& \lambda\vec{x}\lambda z\, o.x_i z &=& \lambda\vec{x}.x_i;\\
h_n(I) &=& \lambda\vec{x}\lambda z\, o.z &=& \mathsf{I}_n;\\
h_n(L) &=& \lambda\vec{x}\lambda z\, o.\pi_1 z &=& \mathsf{L}_n;\\
h_n(R) &=& \lambda\vec{x}\lambda z\, o.\pi_2 z &=& \mathsf{R}_n;\\
h_n(\langle s,t\rangle) &=& \lambda\vec{x}\lambda z\, o.\pi(s\vec{x}z)(t\vec{x}z) &=& \langle h_n(s), h_n(t)\rangle_n.
\end{array}$$

As before one can show that this is an isomoprphism. $\blacksquare$

In the sequel an important case is $n = 1$, i.e. $\mathcal{C}^{1\to 1} \cong \mathcal{F}[x]$.

Hilbert-Post completeness of $\lambda_{\to}SP$

The claim that an equation $M = N$ is either a $\beta\boldsymbol{\eta}SP$ convertibility or inconsistent is proved in two steps. First it is proved for the type $1{\to}1$ by the analysis of $\mathcal{F}[x]$; then it follows for arbitrary types by reducibility of types in λ_{SP}.

Remember that $M\#_T N$ means that $T \cup \{M = N\}$ is inconsistent.

5.2.38. PROPOSITION. (i) *Let $M, N \in \Lambda^{\emptyset}_{SP}(1)$. Then*

$$M \neq_{\beta\boldsymbol{\eta}SP} \Rightarrow\; M\#_{\beta\boldsymbol{\eta}SP}N.$$

(ii) *The same holds for* $M, N \in \Lambda^{\emptyset}_{SP}(1{\to}1)$.

PROOF. (i) Since $\mathcal{F} \cong \mathcal{C}^0 = \Lambda^{\emptyset}_{SP}(1)$, by theorem 5.2.34, this follows from proposition 5.2.19(i).

(ii) If $M, N \in \Lambda^{\emptyset}_{SP}(1{\to}1)$, then

$$\begin{array}{lcll} M \neq N & \Rightarrow & M = \lambda f\ 1.M_1[f] \neq \lambda f\ 1.N_1[f] = N & \text{with } M_1[f], N_1[f] \\ & \Rightarrow & M_1[f] \neq N_1[f] & \\ & \Rightarrow & M_1[F] \neq N_1[F] & \text{for some } F \in \Lambda^{\emptyset}_{SP}(1), \\ & & & \text{by 5.2.32,} \\ & \Rightarrow & M_1[F] \# N_1[F] & \\ & \Rightarrow & M \# N. \ \blacksquare & \end{array}$$

We now want to generalize this last result for all types by using type reducibility in the context of λ_{SP}.

5.2.39. DEFINITION. Let $A, B \mathbb{T}(\lambda_{\to})$. We say that A is $\beta\boldsymbol{\eta}SP$-reducible to B, notation $A \leq_{\beta\boldsymbol{\eta}SP} B$, iff there exists $\Phi : A{\to}B$ such that for any closed $N_1, N_2 : A$

$$N_1 = N_2 \iff \Phi N_1 = \Phi N_2.$$

5.2.40. PROPOSITION. *For each type* A *one has* $A \leq_{\beta\boldsymbol{\eta}SP} 1{\to}1$.

PROOF. We can copy the proof 3.4.7 to obtain $A \leq_{\beta\boldsymbol{\eta}SP} 1_2{\to}o{\to}o$. Moreover, by

$$\lambda uxa.u(\lambda z_1 z_2.x(\pi(xz_1)(xz_2)))a$$

one has $1_2{\to}o{\to}o \leq_{\beta\boldsymbol{\eta}SP} 1{\to}1$. ■

5.2.41. COROLLARY. *Let* $A \in \mathbb{T}(\lambda_{\to})$ *and* $M, N \in \Lambda^{\emptyset}_{SP}$. *Then*

$$M \neq_{\beta\boldsymbol{\eta}SP} N \ \Rightarrow\ M \#_{\beta\boldsymbol{\eta}SP} N.$$

PROOF. Let $A \leq_{\beta\boldsymbol{\eta}SP} 1{\to}1$ using Φ. Then

$$\begin{array}{lcl} M \neq N & \Rightarrow & \Phi M \neq \Phi N \\ & \Rightarrow & \Phi M \# \Phi N, \text{ by corollary 5.2.38(ii),} \\ & \Rightarrow & M \# N. \ \blacksquare \end{array}$$

We obtain the following Hilbert-Post completeness theorem.

5.2.42. THEOREM. *Let* $\mathcal{M}$ *be a model of* λ_{SP}. *For any type* A *and closed terms* $M, N \in \Lambda^{\emptyset}(A)$ *the following are equivalent.*

(i) $M =_{\beta\boldsymbol{\eta}SP} N$;

(ii) $\mathcal{M} \models M = N$;

(iii) $\lambda_{SP} \cup \{M = N\}$ *is consistent.*

PROOF. ((i)⇒(ii)) By soundness. ((ii)⇒(iii)) Since truth implies consistency. ((iii)⇒(i)) By corollary 5.2.41. ■

The result also holds for equations between open terms (consider their closures). The moral is that every equation is either provable or inconsistent. Or that every model of λ_{SP} has the same (equational) theory.

Diophantine relations

5.2.43. DEFINITION. Let $R \subseteq \Lambda^{\emptyset}(A_1) \times \ldots \times \Lambda^{\emptyset}(A_n)$ be an n-ary relation.

(i) R is called *equational* iff

$\exists B \in \mathbb{T}(o) \exists M, N \in \Lambda^{\emptyset}(A_1 \to \ldots \to A_n \to B)\, \forall \vec{F}$

$$R(F_1, \ldots, F_n) \iff MF_1 \ldots F_n = NF_1 \ldots F_n. \tag{1}$$

Here $=$ is taken in the sense of the theory of λ_{SP}.

(ii) R is called the *projection* of the $n+m$-ary relation S if

$$R(\vec{F}) \iff \exists \vec{G}\ S(\vec{F}, \vec{G})$$

(iii) R is called *Diophantine* if it is the projection of an equational relation.

Note that syntactic relations are closed coordinate wise under $=$ and are recursive (since λ_{SP} is CR and SN). A Diophantine relation is clearly closed under $=$ (coordinate wise) and recursivly enumerable. Our main result will be the converse.

5.2.44. THEOREM. *A relation R on closed Λ_{SP} terms is Diophantine if and only if R is closed coordinate wise under $=$ and recursively enumerable.*

The rest of this section is devoted to the proof of this claim.

5.2.45. PROPOSITION. (i) *Equational relations are closed under substitution of lambda definable functions. This means that if R is equational and R' is defined by*

$$R'(\vec{F}) \iff R(H_1\vec{F}, \ldots, H_n\vec{F}),$$

then R' is equational.

(ii) *Equational relations are closed under conjunction.*

(iii) *Equational relations are Diophantine.*

(iv) *Diophantine relations are closed under substitution of lambda definable functions, conjunction and projection.*

PROOF. (i) Easy.

(ii) Use (simple) pairing.

(iii) By dummy projections.

(iv) By some easy logical manipulations. ■

5.2.46. LEMMA. *Let* $R \subseteq \Pi_{i=1}^n \Lambda^{\emptyset}(A_i)$. *Suppose that* $\Phi_i : A_i \leq_{\beta\boldsymbol{\eta}SP} 1{\to}1$. *Define* $R^{\Phi} \subseteq \Lambda^{\emptyset}(1{\to}1)^n$ *by*

$$R^{\Phi}(G_1, \ldots, G_n) \iff \exists F_1 \ldots F_n \, [\Phi_1 F_1 = G_1 \& \ldots \Phi_n F_n = G_n \;\&\; R(F_1, \ldots, F_n)].$$

Then

(i) *R is Diophantine iff* R^{Φ} *is Diophantine.*

(ii) *R is* $=$*-closed and re iff* R^{Φ} *is* $=$*-closed and re.*

PROOF. (i) ($\Rightarrow$) By proposition 5.2.45.

($\Leftarrow$) By noting that $R(F_1, \ldots, F_n) \iff R^{\Phi}(\Phi F_1, \ldots, \Phi F_n)$.

(ii) Similarly. ■

By pairing we can assume without loss of generality that $n = 1$.

5.2.47. LEMMA. *Let* $R \subseteq \Lambda^{\emptyset}(1{\to}1)^n$. *Define* $R^{\wedge} \subseteq \Lambda^{\emptyset}(1{\to}1)$ *by*

$$R^{\wedge}(F) \iff R(\pi_1^{1\to1}(F), \ldots, \pi_n^{1\to1}(F)).$$

Then

(i) *R is Diophantine iff* $R^{\wedge}$ *is Diophantine.*

(ii) *R is* $=$*-closed and re iff* $R^{\wedge}$ *is* $=$*-closed and re.*

PROOF. By proposition 5.2.45(i) and the pairing functions. ■

5.2.48. COROLLARY. *In order to prove that every RE relation* $R \subseteq \Pi_{i=1}^n \Lambda^{\emptyset}(A_i)$ *that is closed under* $=_{\beta\boldsymbol{\eta}SP}$ *is Diophantine, it suffices to do this just for such* $R \subseteq \Lambda^{\emptyset}(1{\to}1)$.

PROOF. By the previous two lemmas. ■

So now we are interested in recursively enumerable subsets of $\Lambda^{\emptyset}(1{\to}1)$ closed under $=_{\beta\boldsymbol{\eta}SP}$. Since

$$\mathtt{exp}(\mathtt{CM} \cup \{\mathtt{x}\})/=_{\mathtt{CM}} \; = \mathcal{F}[x] \cong \mathcal{C}^1 = \Lambda^{\emptyset}(1{\to}1)/=_{\beta\boldsymbol{\eta}SP}$$

one can shift atention to relations on $\mathtt{exp}(\mathtt{CM} \cup \{\mathtt{x}\})$ closed under $=_{\mathrm{CM}}$. We say loosly that such relations are on $\mathcal{F}[x]$. The definition of Such relations to be equational (Diophantine) is slightly different (but completely in accordance with the isomorphism $\mathcal{C}^1 \cong \mathcal{F}[x]$).

5.2.49. DEFINITION. A k-ary relation R on $\mathcal{F}[\vec{x}]$ is called *Diophantine* if there exist $s(u_1, \ldots, u_k, \vec{v}), t(u_1, \ldots, u_k, \vec{v}) \in \mathcal{F}[\vec{u}, \vec{v}]$ such that

$$R(f_1[\vec{x}], \ldots, f_k[\vec{x}]) \iff \exists \vec{v} \in \mathcal{F}[\vec{x}] . s(f_1[\vec{x}], \ldots, f_k[\vec{x}], \vec{v}) = t(f_1[\vec{x}], \ldots, f_k[\vec{x}], \vec{v}).$$

Diophantine relations on $\mathcal{F}$ are closed under conjunction as before.

5.2.50. PROPOSITION (Transfer lemma). (i) *Let* $X \subseteq (\mathcal{F}[x_1, \ldots, x_n])^k$ *be equational (Diophantine). Then* $h_n(X) \subseteq (\mathcal{C}^n)^k$ *is equational (Diophantine), respectively.*

(ii) *Let $X \subseteq (\mathcal{C}^n)^k$ be RE and closed under $=_{\beta\boldsymbol{\eta}SP}$. Then $h_n^{-1}(X) \subseteq (\mathcal{F}[x_1,\ldots,x_n])^k$ is RE and closed under $=_{\mathrm{CM}}$.*

5.2.51. Corollary. *In order to prove that every RE relation on $\mathcal{C}^1$ closed under $=_{\beta\boldsymbol{\eta}SP}$ is Diophantine it suffices to show that every RE relation on $\mathcal{F}[x]$ closed under $=_{\mathrm{CM}}$ is Diophantine.*

Before proving that every recursively enumerable relation on $\mathcal{F}[x]$ is Diophantine, for the sake of clarity we shall give the proof first for $\mathcal{F}$. It consists of two steps: first we encode Matijasevic's solution to Hilbert's 10^{th} problem into this setting; then we give a Diophantine coding of $\mathcal{F}$ in $\mathcal{F}$, and finish the proof for $\mathcal{F}$. Since the coding of $\mathcal{F}$ can easily be extended to $\mathcal{F}[x]$ the result then holds also for this structure and we are done.

5.2.52. Definition. Write $s_n = R^n \in \mathcal{F}$. The set of *numerals* in $\mathcal{F}$ is defined by $\mathcal{N} = \{s_n \mid n \in \mathbb{N}\}$.

We have the following.

5.2.53. Proposition. $f \in \mathcal{N} \iff f * R = R * f$.

Proof. This is because if f is normal and $f * R = R * f$, then f the binary tree part of f must be trivial, that is, f must be a string of L's and R's, thus only R's. ■

5.2.54. Definition. A sequence of k-ary relations on $R_n \subseteq \mathcal{F}$ is called *Diophantine uniformly in n* iff there is a $k+1$-ary Diophantine relation $P \subseteq \mathcal{F}^{k+1}$ such that

$$R_n(\vec{u}) \iff P(s_n, \vec{u}).$$

Now we build up a toolkit of Diophantine relations on $\mathcal{F}$.

1. $\mathcal{N}$ is equational (hence Diophantine).

   Proof. In 5.2.53 is was proved that

   $$f \in \mathcal{N} \iff f * R = R * f. \blacksquare$$

2. The sets $\mathcal{F} * L$, $\mathcal{F} * R \subseteq \mathcal{F}$ and $\{L, R\}$ are equational. In fact one has

   $$\begin{aligned} f \in \mathcal{F} * L &\iff f * \langle L, L\rangle = f. \\ f \in \mathcal{F} * R &\iff f * \langle R, R\rangle = f. \\ f \in \{L, R\} &\iff f * \langle I, I\rangle = f. \end{aligned}$$

   Proof. To see that the first equivalence holds, notice that if $f \in \mathcal{F} * L$, then $f = g * L$, for some $g \in \mathcal{F}$ hence $f * \langle L, L\rangle = f$. Conversely, if $f = f * \langle L, L\rangle$, then $f = f * \langle \mathsf{I}, \mathsf{I}\rangle * L \in \mathcal{F} * L$.

   The second equivalence is proved similarly.

   In the third equivalence ($\Leftarrow$) follows by induction on the nf of f. ■

3. Notation
$$\begin{aligned}[ ] &= R;\\ [f_0,\dots,f_{n-1}] &= \langle f_0 * L,\dots,f_{n-1}*L,R\rangle, \quad \text{if } n>0.\end{aligned}$$
One easily sees that $[f_0,\dots,f_{n-1}]*[I,f_n]=[f_0,\dots,f_n]$. Write
$$\mathrm{Aux}_n(f)=[f,f*R,\dots,f*R^{n-1}].$$
Then the relation $h=\mathrm{Aux}_n(f)$ is Diophantine uniformly in n.
Proof. Indeed,
$$h=\mathrm{Aux}_n(f) \iff R^n*h=R \,\&\, h=R*h*\langle\langle L,L\rangle,\langle f*R^{n-1}*L,R\rangle\rangle.$$
To see ($\Rightarrow$), assume $h=[f,f*R,\dots,f*R^{n-1}]$, then
$h=\langle f*L,f*R*L,\dots,f*R^{n-1}*L,R\rangle$, so $R^n*h=R$ and
$$\begin{aligned} R*h &= [f*R,\dots,f*R^{n-1}]\\ R*h*\langle\langle L,L\rangle,\langle f*R^{n-1}*L,R\rangle\rangle &= [f,f*R,\dots,f*R^{n-1}]\\ &= h.\end{aligned}$$
To see ($\Leftarrow$), note that we always can write $h=\langle h_0,\dots,h_n\rangle$. By the assumptions $h=R*h*\langle\langle L,L\rangle,\langle f*R^{n-1}*L,R\rangle\rangle = R*h*—$, say. So by reading the following equality signs in the correct order (first the left ='s top to bottom; then the right ='s bottom to top) it follows that
$$\begin{aligned} h_0 &= h_1 * — &&= f*L\\ h_1 &= h_2 * — &&= f*R*L\\ &\dots\\ h_{n-2} &= h_{n-1} * — &&= f*R^{n-2}*L\\ h_{n-1} &= f*R^{n-1}*L\\ h_n &= R.\end{aligned}$$
Therefore $h=\mathrm{Aux}_n(f)$ ■.

4. Write $\mathtt{Seq}_n(f) \iff f=[f_0,\dots,f_{n-1}]$, for some $f_0,\dots,f_{n-1}$. Then $\mathtt{Seq}_n$ is Diophamtine uniformly in n.
Proof. One has $\mathtt{Seq}_n(f)$ iff
$$R^n*f=R \,\&\, \mathrm{Aux}_n(L)*\langle I,L\rangle*f=\mathrm{Aux}_n(L)*\langle I,L\rangle*f*\langle L,L\rangle,$$
as can be proved similarly (use 2(i)). ■

5. Define
$$\mathrm{Cp}_n(f)=[f,\dots,f],\ (n \text{ times } f).$$
(By default $\mathrm{Cp}_0(f)=[\,]=R$.) Then Cp_n is Diophantine uniformly in n. Then Cp_n is Diophantine uniformly in n.
Proof. $\mathrm{Cp}_n(f)=g$ iff
$$\mathtt{Seq}_n(g) \,\&\, g=R*g*\langle L,f*L,R\rangle\rangle.$$ ■

6. Let $\mathrm{Pow}_n(f) = f^n$. Then Pow_n is Diophantine uniformly in n.
PROOF. One has $\mathrm{Pow}_n(f) = g$ iff

$$\exists h[\mathtt{Seq}_n(h) \;\&\; h = R * h * \langle f * L, f * L, R\rangle \;\&\; L * h = g].$$

This can be proved in a similar way (it helps to realize that h has to be of the form $h = [f^n, \ldots, f^1]$). ■

Now we can show that the operations $+$ and $\times$ on $\mathcal{N}$ are 'Diophantine'.

7. There are Diophantine ternary relations P_+, $P_\times$ such that for all n, m, k
   (1) $P_+(s_n, s_m, s_k) \iff n + m = k$.
   (2) $P_\times(s_n, s_m, s_k) \iff n.m = k$.

PROOF. (i) Define $P_+(x, y, z) \iff x{*}y = z$. This relation is Diophantine and works: $R^n * R^m = R^k \iff R^{n+m} = R^k \iff n + m = k$.

(ii) Let $\mathrm{Pow}_n(f) = g \iff P(s_n, f, g)$, with P Diophantine. Then $P_\times(x, y, z) \iff P(x, y, z)$. ■

8. Let $X \subseteq \mathbb{N}$ be a recursively enumerable set of narural numbers. Then $\{s_n \mid n \in X\}$ is Diophantine.
PROOF. By 7 and the famous theorem of Matijasevič [1971]. ■

9. Define $\mathtt{Seq}_n^{\mathcal{N}} = \{[s_{m_0}, \ldots, s_{m_{n-1}}] \mid m_0, \ldots, m_{n-1} \in \mathbb{N}\}$. Then the relation $f \in \mathtt{Seq}_n^{\mathcal{N}}$ is Diophantine uniformly in n.
PROOF. Indeed, $f \in \mathtt{Seq}_n^{\mathcal{N}}$ iff

$$\mathtt{Seq}_n(f) \;\&\; f * \langle R * L, R\rangle = \mathrm{Aux}_n(R * L) * \langle I, R^n\rangle * f.$$ ■

10. Let $f = [f_0, \ldots, f_{n-1}]$ and $g = [g_0, \ldots, g_{n-1}]$. We write

$$f\#g = [f_0 * g_0, \ldots, f_{n-1} * g_{n-1}].$$

Then there exists a Diophantine relation P such that for arbitrary n and $f, g \in \mathtt{Seq}_n$ one has

$$P(f, g, h) \iff h = f\#g.$$

PROOF. Let

$$\mathsf{Cmp}_n(f) = [L * f, L * R * f * R, \ldots, L * R^{n-1} * f * R^{n-1}].$$

Then $g = \mathsf{Cmp}_n(f)$ is Diophantine uniformly in n.
This requires some work. One has by the by now familiar technique

$$\begin{array}{rcl}
\mathsf{Cmp}_n(f) = g & \iff & \\
\exists h_1, h_2, h_3 & [& \\
\mathtt{Seq}_n(h_1) & \& & f = h_1 * \langle I, R^n * f\rangle \\
\mathtt{Seq}_{n^2}(h_2) & \& & h_2 = R^n * h_2 * \langle\langle L, L\rangle, h_1 * \langle R^{n-1} * L, R\rangle\rangle \\
\mathtt{Seq}_n^{\mathcal{N}}(h_3) & \& & h_3 = R * h_3 * \langle\langle I, I\rangle^{n+1} * L, \langle R^{n^2-1} * L, R\rangle\rangle \\
 & \& & g = \mathrm{Aux}_n(L^2) * \langle h_3, R^n\rangle * \langle h_2, R\rangle \\
 &] & .
\end{array}$$

For understanding it helps to identify the h_1, h_2, h_3. Suppose $f = \langle f_0, \ldots, f_{n-1}, f_n \rangle\!\rangle$. Then

$$\begin{array}{rcl} h_1 & = & [f_0, f_1, \ldots, f_{n-1}]; \\ h_2 & = & [f_0, f_1, \ldots, f_{n-1}, \\ & & f_0 * R, f_1 * R, \ldots, f_{n-1} * R, \\ & & \ldots, \\ & & f_0 * R^{n-1}, f_1 * R^{n-1}, \ldots, f_{n-1} * R^{n-1}]; \\ h_3 & = & [I, R^{n+1}, R^{2(n+1)}, \ldots, R^{(n-1)(n+1)}]. \end{array}$$

Now define

$$P(f, g, h) \iff \exists n[\mathtt{Seq}_n(f) \;\&\; \mathtt{Seq}_n(g) \;\&\; \mathtt{Cmp}_n(f * L) * \langle I, R^n \rangle * g = h].$$

Then P is Diophantine and for arbitrary n and $f, g \in \mathtt{Seq}_n$ one has

$$h = f \# g \iff P(f, g, h). \blacksquare$$

11. For $f = [f_0, \ldots, f_{n-1}]$ define $\Pi(f) = f_0 * \ldots * f_{n-1}$. Then there exists a Diophantine relation P such that for all $n \in \mathbb{N}$ and all $f \in \mathtt{Seq}_n$ one has

$$P(f, g) \iff \Pi(f) = g.$$

PROOF. Define $P(f, g)$ iff

$$\begin{array}{rcl} \exists n, h & [& \\ \mathtt{Seq}_n(f) & \& & \\ \mathtt{Seq}_{n+1}(h) & \& & h = ((f * \langle I, R \rangle) \# (R * h)) * \langle L, I * L, R \rangle\!\rangle \\ & \& & g = L * h * \langle I, R \rangle \\ &] & . \end{array}$$

Then P works as can be seen realizing h has to be

$$[f_0 * \ldots * f_{n-1}, f_1 * \ldots * f_{n-1}, \ldots, f_{n-2} * f_{n-1}, f_{n-1}, I]. \blacksquare$$

12. Define $\mathrm{Byte}_n(f) \iff f = [b_0, \ldots, b_{n-1}]$, for some $b_i \in \{L, R\}$. Then Byte_n is Diophantine uniformly in n.
PROOF. Using 2 one has $\mathrm{Byte}_n(f)$ iff

$$\mathtt{Seq}_n(f) \;\&\; f * \langle \langle I, I \rangle, R \rangle = \mathrm{Cp}_n(I). \blacksquare$$

13. Let $m \in \mathbb{N}$ and let $[m]_2$ be its binary notation of length n. Let $[m]_{\mathrm{Byte}} \in \mathtt{Seq}_n^{\mathcal{N}}$ be the corresponding element, where L corresponds to a 1 and R to a 0 and the most significant bit is written last. For example $[6]_2 = 110$, hence $[6]_{\mathrm{Byte}} = [R, L, L]$. Then there exists a Diophantine relation Bin such that for all $m \in \mathbb{N}$

$$\mathrm{Bin}(s_m, f) \iff f = [m]_{\mathrm{Byte}}.$$

PROOF. We need two auxiliary maps.

$$\begin{aligned} \mathrm{Pow2}(n) &= [R^{2^{n-1}}, \ldots, R^{2^0}]; \\ \mathrm{Pow2}I(n) &= [\langle R^{2^{n-1}}, I\rangle, \ldots, \langle R^{2^0}, I\rangle]. \end{aligned}$$

For these the relations $\mathrm{Pow2}(n) = g$ and $\mathrm{Pow2}I(n) = g$ are Diophantine uniformly in n. Indeed, $\mathrm{Pow2}(n) = g$ iff

$$\mathtt{Seq}_n(g) \;\&\; g = ((R * g)\#(R * g)) * [I, R];$$

and $\mathrm{Pow2}I(n) = g$ iff

$$\begin{aligned} \mathtt{Seq}_n(g) \quad &\& \quad \mathrm{Cp}_n(L)\#g = \mathrm{Pow2}(n); \\ &\& \quad \mathrm{Cp}_n(R)\#g = \mathrm{Cp}_n(I). \end{aligned}$$

It follows that Bin is Diophantine since $\mathrm{Bin}(m, f)$ iff

$$m \in \mathcal{N} \;\&\; \exists n[\mathrm{Byte}_n(f) \;\&\; \Pi(f\#\mathrm{Pow2}I(n)) = m]. \blacksquare$$

14. We now define a surjection $\varphi : \mathbb{N} \to \mathcal{F}$. Remember that $\mathcal{F}$ is generated by two elements $\{e_0, e_1\}$ using only $*$. One has $e_1 = L$. Define

$$\varphi(n) = e_{i_0} * \ldots * e_{i_{m-1}},$$

where $[n]_2 = i_{m-1} \ldots i_0$. We say that n is a *code* of $\varphi(n)$. Since every $f \in \mathcal{F}$ can be written as $L * \langle I, I\rangle * f$ the map φ is surjective indeed.

15. $\mathrm{Code}(n, f)$ defined by $\varphi(n) = f$ is Diophantine uniformly in n.
PROOF. Indeed, $\mathrm{Code}(n, f)$ iff

$$\exists g\,[\mathrm{Bin}(n, g) \;\&\; \Pi(g * \langle\langle e_0, e_1\rangle, R\rangle) = f. \blacksquare$$

16. Every RE subset $\mathcal{X} \subseteq \mathcal{F}$ is Diophantine.
PROOF. Since the word problem for $\mathcal{F}$ is decidable, $\#\mathcal{X} = \{m \mid \exists f \in \mathcal{X}\ \varphi(m) = f\}$ is also RE. By (8), $\#\mathcal{X} \subseteq \mathbb{N}$ is Diophantine. Hence by (15) $\mathcal{X}$ is Diophantine via

$$g \in \mathcal{X} \iff \exists f\ f \in \#\mathcal{X} \;\&\; \mathrm{Code}(f, g). \blacksquare$$

17. Every RE subset $\mathcal{X} \subseteq \mathcal{F}[\vec{x}]$ is Diophantine.
PROOF. Similarly, since also $\mathcal{F}[\vec{x}]$ is generated by two of its elements. We need to know that all the Diophantine relations $\subseteq \mathcal{F}$ are also Diophantine $\subseteq \mathcal{F}[x]$. This follows from exercise 5.6.23 and the fact that such relations are closed under intersection. $\blacksquare$

PROOF OF 5.2.44. By 17 and corollaries 5.2.48 and 5.2.51. $\blacksquare$

5.3. Gödel's system $\mathcal{T}$: higher-order primitive recursion

The set of primitive recursive functions is the smallest set containing zero, successor and projection functions which is closed under composition and the following schema of first-order primitive recursion:

$$\begin{aligned} F(0,\vec{x}) &= G(\vec{x}) \\ F(n+1,\vec{x}) &= H(F(n,\vec{x}),n,\vec{x}) \end{aligned}$$

This schema defines F from G and H by stating that $F(0) = G$ and by expressing $F(n+1)$ in terms of $F(n)$, H and n. The parameters $\vec{x}$ range over the natural numbers.

Thus defined, the primitive recursive functions were conjectured by Skolem to cover all computable functions, early this century. This conjecture was shown to be false in Ackermann [1928], who gave a computable function that is not primitive recursive. A few years later the class of computable functions was shown to be much larger by Church and Turing. Nevertheless the primitive recursive functions include almost all functions that one encounters 'in practice', such as addition, multiplication, exponentiation, and many more.

Besides the existence of computable functions that are not primitive recursive, there is another reason to generalize the above schema, namely the existence of computable objects that are not number theoretic functions. For example, given a number theoretic function F and a number n, compute the maximum that F takes on arguments less than n. Other examples of computations where inputs and/or outputs are functions: compute the function that coincides with F on arguments less than n and zeroes otherwise, compute the n-th iterate of F, and so on. These computations define mappings that are commonly called functionals, to stress that they are more general than number theoretic functions.

Consider the full typestructure over the natural numbers, that is, sets $\mathbb{N}_{\mathsf{N}} = \mathbb{N}$ and $\mathbb{N}_{A\to B} = \mathbb{N}_A \to \mathbb{N}_B$, the set of all mappings from $\mathbb{N}_A$ to $\mathbb{N}_B$. Application of $F \in \mathbb{N}_{A\to B}$ to $G \in \mathbb{N}_A$ is denoted by FG or $F(G)$. We allow a liberal use of currying, so the following denotations are all identified:

$$FGH \equiv (FG)H \equiv F(G,H) \equiv F(G)H \equiv F(G)(H)$$

Application is left associative, so $F(GH)$ is notably different from the above denotations.

The abovementioned interest in higher-order computations leads to the following schema of higher-order primitive recursion proposed in Gödel [1958]:[2]

$$\begin{aligned} RMN0 &= M \\ RMN(n+1) &= N(RMNn)n \end{aligned}$$

Here M need not be a natural number, but can have any $A \in \mathsf{TT}^o$ as type (see Section 1.1). The corresponding type of N is $A\to\mathsf{N}\to A$, where N is the type of

[2] For the purpose of a translation of intuitionistic arithmetic into the quantifier free theory of primitive recursive functionals of finite type, yielding a consistency proof for arithmetic.

the natural numbers. We make some further observations with respect to this schema. First, the dependency of F from G and H in the first-order schema is made explicit by defining RMN, which is to be compared to F. Second, the parameters $\vec{x}$ from the first-order schema are left out above since they are no longer necessary: we can have higher-order objects as results of computations. Third, the type of R depends on the type of the result of the computation. In fact we have a family of *recursors* $R_A : A{\to}(A{\to}\mathsf{N}{\to}A){\to}\mathsf{N}{\to}A$ for every type A.

5.3.1. Exercise. Show that the first-order schema of primitive recursion is subsumed by the higher-order schema, by expressing F in terms of R, G and H.

The set of primitive recursive functionals is the smallest set of functionals containing 0, the successor function and functionals R of all appropriate types, which is closed under explicit $\lambda^o_{\to}$-definition. This definition implies that the primitive recursive functionals include projection functions and are closed under application, composition and the above schema of higher-order primitive recursion.

We shall now exhibit a number of examples of primitive recursive functionals. First, let K, K^* be defined explicitly by $K(x,y) = x$, $K^*(x,y) = y$ for all $x, y \in \mathbb{N}$, that is, the first and the second projection. Obviously, K and K^* are primitive recursive functionals, as they come from $\lambda^o_{\to}$-terms. Now consider $P \equiv R0K^*$. Then we have $P0 = 0$ and $P(n+1) = R0K^*(n+1) = K^*(R0K^*)n = n$ for all $n \in \mathbb{N}$, so that we call P the predecessor function. Now consider $x \dot{-} y \equiv Rx(P * K)y$. Here $P * K$ is the composition of P and K, that is, $(P * K)xy = P(K(x,y)) = P(x)$. We have $x \dot{-} 0 = x$ and $x \dot{-} (y+1) = Rx(P*K)(y+1) = (P*K)(Rx(P*K)y)y = P(Rx(P*K)y) = P(x \dot{-} y)$. Thus we have defined cut-off subtraction $\dot{-}$ as primitive recursive functional.

5.3.2. Exercise. Which function is computed if we replace P in $Rx(P * K)y$ by the successor function? Define multiplication, exponentiation and division with remainder as primitive recursive functionals.

In the previous paragraph, we have only used R_{N} in order to define some functions that are, in fact, already definable with first-order primitive recursion. In this paragraph we are going to use $R_{\mathsf{N}\to\mathsf{N}}$ as well. Given a functions F, F' and natural numbers x, y, define explicitly the functional G by $G(F, F', x, y) = F'(F(y))$ and abbreviate $G(F)$ by G_F. Now consider RIG_F, where R is actually $R_{\mathsf{N}\to\mathsf{N}}$ and I is the identity function on the natural numbers. We calculate $RIG_F0 = I$ and $RIG_F(n+1) = G_F(RIG_Fn)n$, which is a function assigning $G(F, RIG_Fn, n, m) = RIG_Fn(Fm)$ to every natural number m. In other words, RIG_Fn is a function which iterates F precisely n times, and we denote this function by F^n.

We finish this paragraph with an example of a computable function A that is not first-order primitive recursive, a result due to Ackermann. The essential difficulty of the function A is the nested recursion in the third clause below.

$$\begin{aligned} A(0,m) &= m+1 \\ A(n+1,0) &= A(n,1) \\ A(n+1,m+1) &= A(n,A(n+1,m)) \end{aligned}$$

In other words, $A(0)$ is the successor function, and by using the the last two equations we see $A(n+1,m) = A(n)^{m+1}(1)$. Thus we can obtain $A = RSH$, where S is the successor function and $H(F,x,y) = F^{y+1}1$. As examples we calculate $A(1,m) = H(A(0),1,m) = A(0)^{m+1}1 = m+2$ and $A(2,m) = H(A(1),1,m) = A(1)^{m+1}1 = 2m+3$.

5.3.3. Exercise. Calculate $A(3,m)$ and verify that $A(4,0) = 13$ and $A(4,1) = 65533$.

5.3.4. Exercise. With one occurrence hidden in H, RSH contains $R_{\mathsf{N}\to\mathsf{N}}$ twice. Define A using R_{N} and $R_{\mathsf{N}\to\mathsf{N}}$ only once. Is it possible to define A with R_{N} only, possibly with multiple occurrences?

course of value recursion
multiple recursion

5.3.5. Exercise (simultaneous primitive recursion). Assume G_i, H_i $(i = 1,2)$ have been given and define F_i $(i = 1,2)$ as follows.

$$\begin{aligned} F_i(0,\vec{x}) &= G_i(\vec{x}) \\ F_i(n+1,\vec{x}) &= H_i(F_1(n,\vec{x}),(F_2(n,\vec{x}),n,\vec{x}) \end{aligned}$$

Show that F_i $(i = 1,2)$ can be defined by first-order primitive recursion. Hint: use a pairing function such as in Figure 5.2.

5.3.6. Exercise (nested recursion, Péter [1967]). Define

$$\begin{aligned} \psi(n,m) &= 0 \qquad \text{if } m \cdot n = 0 \\ \psi(n+1,m+1) &= \beta(m,n,\psi(m,\gamma(m,n,\psi(m+1,n))),\psi(m+1,n)) \end{aligned}$$

Show that ψ can be defined from β,γ using higher-order primitive recursion.

5.3.7. Exercise (Dialectica translation). We closely follow Troelstra [1973], Section 3.5; the solution can be found there. Let HA^ω be the theory of higher-order primitive recursive functionals equipped with many-sorted intuitionistic predicate logic with equality for natural numbers and axioms for arithmetic, in particular the schema of aritmetical induction:

$$(\varphi(0) \wedge \forall x\ (\varphi(x)\ \Rightarrow\ \varphi(x+1)))\ \Rightarrow\ \forall x\ \varphi(x)$$

The *Dialectica interpretation* of Gödel [1958], D-interpretation for short, assigns to every formula φ in the language of HA^ω a formula $\varphi^D \equiv \exists\vec{x}\ \forall\vec{y}\ \varphi_D(\vec{x},\vec{y})$ in the same language. The types of $\vec{x},\vec{y}$ depend on the logical structure of φ only. We define φ_D and φ^D by induction on φ:

1. If φ is prime, that is, an equation of lowest type, then $\varphi^D \equiv \varphi_D \equiv \varphi$.

For the binary connectives, assume $\varphi^D \equiv \exists \vec{x}\, \forall \vec{y}\, \varphi_D(\vec{x}, \vec{y})$, $\psi^D \equiv \exists \vec{u}\, \forall \vec{v}\, \psi_D(\vec{u}, \vec{v})$.

2. $(\varphi \wedge \psi)^D \equiv \exists \vec{x}, \vec{u}\, \forall \vec{y}, \vec{v}\, (\varphi \wedge \psi)_D$, with $(\varphi \wedge \psi)_D \equiv (\varphi_D(\vec{x}, \vec{y}) \wedge \psi_D(\vec{u}, \vec{v}))$.

3. $(\varphi \vee \psi)^D \equiv \exists z, \vec{x}, \vec{u}\, \forall \vec{y}, \vec{v}\, (\varphi \vee \psi)_D$, with $(\varphi \vee \psi)_D \equiv ((z = 0 \Rightarrow \varphi_D(\vec{x}, \vec{y})) \wedge (z \neq 0 \Rightarrow \psi_D(\vec{u}, \vec{v})))$.

4. $(\varphi \Rightarrow \psi)^D \equiv \exists \vec{u'}, \vec{y'}\, \forall \vec{x}, \vec{v}\, (\varphi \Rightarrow \psi)_D$, with $(\varphi \Rightarrow \psi)_D \equiv (\varphi_D(\vec{x}, \vec{y'}\vec{x}\vec{v}) \Rightarrow \psi_D(\vec{u'}\vec{x}, \vec{v}))$.

Note that the clause for $\varphi \Rightarrow \psi$ introduces quantifications over higher types than those used for the formulas φ, ψ. This is also the case for formulas of the form $\forall z\ \varphi(z)$, see the sixth case below. For both quantifier clauses, assume $\varphi^D(z) \equiv \exists \vec{x}\, \forall \vec{y}\, \varphi_D(\vec{x}, \vec{y}, z)$.

5. $(\exists z\ \varphi(z))^D \equiv \exists z, \vec{x}\, \forall \vec{y}\, (\exists z\ \varphi(z))_D$, with $(\exists z\ \varphi(z))_D \equiv \varphi_D(\vec{x}, \vec{y}, z)$.

6. $(\forall z\ \varphi(z))^D \equiv \exists \vec{x'}\, \forall z, \vec{y}\, (\forall z\ \varphi(z))_D$, with $(\forall z\ \varphi(z))_D \equiv \varphi_D(\vec{x'}z, \vec{y}, z)$.

With φ, ψ as in the case of a binary connective, determine $(\varphi \Rightarrow (\varphi \vee \psi))^D$ and give a sequence $\vec{t}$ of higher-order primitive recursive functionals such that $\forall \vec{y}\, (\varphi \Rightarrow (\varphi \vee \psi))_D(\vec{t}, \vec{y})$. We say that thus the D-interpretation of $(\varphi \Rightarrow (\varphi \vee \psi))^D$ is validated by higher-order primitive recursive functionals. Validate the D-interpretation of $(\varphi \Rightarrow (\varphi \wedge \varphi))^D$. Validate the D-interpretation of induction. The result of Gödel [1958] can now be rendered as: the D-interpretation of every theorem of HA^ω can be validated by higher-order primitive recursive functionals. This yields a consistency proof for HA^ω, since $0 = 1$ cannot be validated. Note that the D-interpretation and the successive validation translates arbitrarily quantified formulas into universally quantified propositional combinations of equations.

Syntax of λ_T

In this section we formalize Gödel's $\mathcal{T}$ as an extension of the simply typed lambda calculus $\lambda^o_\to$, called λ_T.

5.3.8. Definition. The *types* of λ_T are the types of $\lambda^o_\to$ over a base type N. The *terms* of λ_T are obtained by adding to the term formation rules of $\lambda^o_\to$ the constants $\mathsf{0} : \mathsf{N}$, $\mathsf{S}^+ : \mathsf{N}\to\mathsf{N}$ and $\mathsf{R}_A : A\to(A\to\mathsf{N}\to A)\to\mathsf{N}\to A$ for all types A. We denote the set of (closed) terms of type A by $\Lambda_T(A)$ ($\Lambda^\emptyset_T(A)$) and put $\Lambda_T = \bigcup_A \Lambda_T(A)$ ($\Lambda^\emptyset_T = \bigcup_A \Lambda^\emptyset_T(A)$). Terms constructed from $\mathsf{0}$ and S^+ only are called *numerals*, with $\mathsf{1}$ abbreviating $\mathsf{S}^+(\mathsf{0})$, $\mathsf{2}$ abbreviating $\mathsf{S}^+(\mathsf{S}^+(\mathsf{0}))$, and so on. An arbitrary numeral will be denoted by n. We define inductively $\mathsf{n}^{A\to B} \equiv \lambda x^A.\mathsf{n}^B$, with $\mathsf{n}^\mathsf{N} \equiv \mathsf{n}$.

The *formulas* of λ_T are equations between terms (of the same type). The *theory* of λ_T is axiomatized by equality axioms and rules, β-conversion and the schema of higher-order primitive recursion from the previous section. The

reduction relation $\to_T$ of λ_T is the compatible closure of the β-rules and the following (schematic) rules for the constants R_A:

$$\begin{array}{rcl} \mathsf{R}_A MN\mathsf{0} & \to_T & M \\ \mathsf{R}_A MN(\mathsf{S}^+P) & \to_T & N(\mathsf{R}_A MNP)P \end{array}$$

5.3.9. THEOREM. *The conversion relation generated by* $\to_T$ *coincides with the theory* λ_T.

PROOF. By an easy extension of the proof of this result in untyped lambda calculus, see B[1984] Proposition 3.2.1. ■

5.3.10. LEMMA. *Every closed normal form of type* N *is a numeral.*

PROOF. Consider the leftmost symbol of a closed normal form of type N. This symbol cannot be a variable since the term is closed. The leftmost symbol cannot be a λ, since abstraction terms are not of type N and a redex is not a normal form. If the leftmost symbol is $\mathsf{0}$, then the term is the numeral $\mathsf{0}$. If the leftmost symbol is S^+, then the term must be of the form S^+P, with P a closed normal form of type N. If the leftmost term is R, then for typing reasons the term must be $\mathsf{R}MNP\vec{Q}$, with P a closed normal form of type N. In the latter two cases we can complete the argument by induction, since P is a smaller term. Hence P is a numeral, so also S^+P. The case $\mathsf{R}MNP$ with P a numeral can be excluded, as $\mathsf{R}MNP$ should be a normal form. ■

We now prove SN and CR for λ_T, two results that could be proved independently from each other. However, the proof of CR can be simplified by using SN, which we prove first by an extension of the proof of SN for $\lambda^o_\to$, Theorem 2.2.2.

5.3.11. THEOREM. *Every* $M \in \Lambda_T$ *is* SN *with respect to* $\to_T$.

PROOF. We recall the notion of computability from the proof of Theorem 2.2.2, generalised to terms of λ_T. We shall frequently use that computable terms are SN, see formula (2) in the proof of Theorem 2.2.2. In view of the definition of computability it suffices to prove that the constants $\mathsf{0}, \mathsf{S}^+, \mathsf{R}_A$ of λ_T are computable. The constant $\mathsf{0} : \mathsf{N}$ is computable since it is SN. Consider S^+P with computable $P : \mathsf{N}$, so P is SN and hence S^+P. It follows that S^+ is computable. In order to prove that R_A is computable, assume that M, N, P are computable and of appropriate type such that $\mathsf{R}_A MNP$ is of type A. Since $P : \mathsf{N}$ is computable, it is SN. Since $\to_T$ is finitely branching, P has only finitely many normal forms, which are numerals by Lemma 5.3.10. Let $\#P$ be the largest of those numerals. We shall prove by induction on $\#P$ that $\mathsf{R}_A MNP$ is computable. Let $\vec{Q}$ be computable such that $\mathsf{R}_A MNP\vec{Q}$ is of type N. We have to show that $\mathsf{R}_A MNP\vec{Q}$ is SN. If $\#P = \mathsf{0}$, then every reduct of $\mathsf{R}_A MNP\vec{Q}$ passes through a reduct of $M\vec{Q}$, and SN follows since $M\vec{Q}$ is computable. If $\#P = \mathsf{S}^+\mathsf{n}$, then every reduct of $\mathsf{R}_A MNP\vec{Q}$ passes through a reduct of $N(\mathsf{R}_A MNP')P'\vec{Q}$, where P' is such that S^+P' is a reduct of P. Then

we have $\#P' = \mathsf{n}$ and by induction it follows that $\mathsf{R}_A MNP'$ is computable. Now SN follows since all terms involved are computable. We have proved that $\mathsf{R}_A MNP$ is computable whenever M, N, P are, and hence R_A is computable. ■

5.3.12. THEOREM. *Every $M \in \Lambda_T$ is* WCR *with respect to $\to_T$.*

PROOF. Different redexes in the same term are either completely disjoint, or one redex is included in the other. In the first case the order of the reduction steps is irrelevant, and in the second case a common reduct can be obtained by reducing (possibly multiplied) included redexes. ■

5.3.13. THEOREM. *Every $M \in \Lambda_T$ is* CR *with respect to $\to_T$.*

PROOF. By Newman's Lemma 5.3.14, using Theorem 5.3.11. ■

5.3.14. LEMMA (Newman, localized). *Let S be a set and $\to$ a binary relation on S that is* WCR. *For every $a \in S$ we have: if $a \in$ SN, then $a \in$ CR.*

PROOF. Call an element *ambiguous* if it reduces to two (or more) distinct normal forms. Assume $a \in$ SN, then a reduces to at least one normal form and all reducts of a are SN. It suffices for $a \in$ CR to prove that a is not ambiguous, i.e. that a reduces to exactly one normal form. Assume by contradiction that a is ambiguous, reducing to different normal forms n_1, n_2, say $a \to b \to \cdots \to n_1$ and $a \to c \to \cdots \to n_2$. Applying WCR to the diverging reduction steps yields a common reduct d such that $b \twoheadrightarrow d$ and $c \twoheadrightarrow d$. Since $d \in$ SN reduces to a normal form, say n, distinct of at least one of n_1, n_2, it follows that at least one of b, c is ambiguous. See Figure 5.1. Hence a has a one-step reduct which is again ambiguous and SN. Iterating this argument yields an infinite reduction sequence contradicting $a \in$ SN, so a cannot be ambiguous. ■

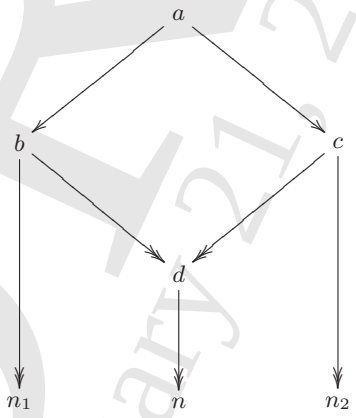


Figure 5.1: Ambiguous a has ambiguous reduct b or c.

If one considers λ_T also with $\boldsymbol{\eta}$-reduction, then the above results can also be obtained. For SN it simply suffices to strenghten the notion of computability for

the base case to SN with also $\boldsymbol{\eta}$-reductions included. WCR and hence CR are harder to obtain and require techniques like $\boldsymbol{\eta}$-postponement, see Barendregt [1984], Section 15.1.6.

Semantics of λ_T

In this section we give some interpetations of Gödel's $\mathcal{T}$, preceded by a general model definition of λ_T building on that of $\lambda^o_\to$.

5.3.15. Definition. A model of λ_T is a model of $\lambda^o_\to$ with interpretations of the constants $\mathsf{0}$, S^+ and R_A for all A, such that the schema of higher-order primitive recursion is valid.

5.3.16. Example. Recall the full typestructure over the natural numbers, that is, sets $\mathbb{N}_\mathsf{N} = \mathbb{N}$ and $\mathbb{N}_{A\to B} = \mathbb{N}_A \to \mathbb{N}_B$, with set-theoretic application. The full typestructure becomes the canonical model of λ_T by interpreting $\mathsf{0}$ as 0, S^+ as the successor function, and the constants R_A as primitive recursors of the right type. The proof that $[\![\mathsf{R}_A]\!]$ is well-defined goes by induction.

5.3.17. Exercise. Consider for any type B the set of closed terms of type B modulo convertibility. Prove that this yields a model for Gödel's $\mathcal{T}$. This model is called the *closed term model* of Gödel's $\mathcal{T}$.

Let $*$ be *Kleene application*, that is, $i * n$ stands for applying the i-th partial recursive function to the input n. If this yield a result, then we flag $i * n{\downarrow}$, otherwise $i * n{\uparrow}$. Equality between expressions with Kleene application is taken to be strict, that is, equality does only hold if left and right hand sides do yield a result and the results are equal. Similarly, $i * n \in S$ should be taken in the strict sense of $i * n$ actually yielding a result in S.

5.3.18. Exercise. By induction we define for every type B a set $HRO_B \subseteq \mathbb{N}$:

$$\begin{array}{rcl} HRO_\mathsf{N} & = & \mathbb{N} \\ HRO_{B\to B'} & = & \{x \in \mathbb{N} \mid x * y \in HRO_{B'} \text{ for all } y \in HRO_B\} \end{array}$$

Prove that HRO with Kleene application constitutes a model for Gödel's $\mathcal{T}$.

5.3.19. Exercise. By simultaneous induction we define for every type B a set $HEO_B \subseteq \mathbb{N}$ equipped with an equivalence relation $=_B$ by

$$\begin{array}{rcl} HEO_\mathsf{N} & = & \mathbb{N} \\ x =_\mathsf{N} y & \iff & x = y \\ HEO_{B\to B'} & = & \{x \in \mathbb{N} \mid x * y \in HEO_{B'} \text{ for all } y \in HEO_B \text{ and} \\ & & \quad x * y =_{B'} x * y' \text{ for all } y, y' \in HEO_B \text{ with } y =_B y'\} \\ x =_{B\to B'} x' & \iff & x, x' \in HEO_{B\to B'} \text{ and } x * y =_{B'} x' * y \text{ for all } y \in HEO_B \end{array}$$

Prove that HEO with Kleene application constitutes a model for Gödel's $\mathcal{T}$.

5.3.20. Exercise. Recall that extensionality essentially means that objects having the same applicative behaviour can be identified. Which of the above models of λ_T, the full type structure, the closed term model, HRO and HEO, is extensional?

Computational strength

Brief review of ordinal theory

Here are some ordinal numbers, simply called *ordinals*, in increasing order:

$$\underline{0}, \underline{1}, \underline{2}, \ldots \omega, \omega + \underline{1}, \omega + \underline{2}, \ldots \omega + \omega = \omega \cdot \underline{2}, \ldots \omega \cdot \omega = \omega^{\underline{2}}, \ldots \omega^{\omega}, \ldots \omega^{(\omega^{\omega})}, \ldots$$

Apart from ordinals, also some basic operations of ordinal arithmetic are visible, namely addition, multiplication and exponentiation, denoted in the same way as in highschool algebra. The dots ... stand for many more ordinals in between, produced by iterating the previous construction process.

The most important structural property of ordinals is that $<$ is a well-order, that is, an order such that every non-empty subset contains a smallest element. This property leads to the principle of (transfinite) induction for ordinals, stating that $P(\alpha)$ holds for all ordinals α whenever P is *inductive*, that is, $P(\alpha)$ follows from $\forall \beta < \alpha . P(\beta)$ for all α.

In fact the aritmetical operations are defined by means of two more primitive operations on ordinals, namely the *successor* operation $+\underline{1}$ and the *supremum* operation $\bigcup$. The supremum $\bigcup a$ of a set of ordinals a is the least upper bound of a, which is equal to the smallest ordinal greater than all ordinals in the set a. A typical example of the latter is the ordinal ω, the first infinite ordinal, which is the supremum of the sequence of the finite ordinals $\underline{n}$ produced by iterating the successor operation on $\underline{0}$.

These primitive operations divide the ordinals in three classes: the *successor* ordinals of the form $\alpha + \underline{1}$, the *limit* ordinals $\lambda = \bigcup\{\alpha \mid \alpha < \lambda\}$, i.e. ordinals which are the supremum of the set of smaller ordinals, and the *zero* ordinal $\underline{0}$. (In fact $\underline{0}$ is the supremum of the empty set, but is not considered to be a limit ordinal.) Thus we have zero, successor and limit ordinals.

Addition, multiplication and exponentiation are now defined according to Table 5.1. Ordinal arithmetic has many properties in common with ordinary arithmetic, but there are some notable exceptions. For example, addition and multiplication are associative but not commutative: $\underline{1} + \omega = \omega \neq \omega + \underline{1}$ and $\underline{2} \cdot \omega = \omega \neq \omega \cdot \underline{2}$. Furthermore, multiplication is left distributive over addition, but not rigth distributive: $(\underline{1} + \underline{1}) \cdot \omega = \omega \neq \underline{1} \cdot \omega + \underline{1} \cdot \omega$. The sum $\alpha + \beta$ is weakly increasing in α and strictly increasing in β. Similarly for the product $\alpha \cdot \beta$ with $\alpha > \underline{0}$. The only exponentiations we shall use, 2^{α} and ω^{α}, are strictly increasing in α.

Addition	Multiplication	Exponentiation ($\alpha > \underline{0}$)
$\alpha + \underline{0} = \alpha$	$\alpha \cdot \underline{0} = \underline{0}$	$\alpha^{\underline{0}} = \underline{1}$
$\alpha + (\beta + \underline{1}) = (\alpha + \beta) + \underline{1}$	$\alpha \cdot (\beta + \underline{1}) = \alpha \cdot \beta + \alpha$	$\alpha^{\beta + \underline{1}} = \alpha^{\beta} \cdot \alpha$
$\alpha + \lambda = \bigcup\{\alpha + \beta \mid \beta < \lambda\}$	$\alpha \cdot \lambda = \bigcup\{\alpha \cdot \beta \mid \beta < \lambda\}$	$\alpha^{\lambda} = \bigcup\{\alpha^{\beta} \mid \beta < \lambda\}$

Table 5.1: Ordinal arithmetic (with λ limit ordinal in the third row).

The operations of ordinal arithmetic as defined above provide examples of a more general phenomenon called transfinite iteration, to be defined below.

5.3.21. Definition. Let f be an ordinal function. Define by induction $f^{\underline{0}}(\alpha) = \alpha$, $f^{\beta+\underline{1}}(\alpha) = f(f^{\beta}(\alpha))$ and $f^{\lambda}(\alpha) = \bigcup\{f^{\beta}(\alpha) \mid \beta < \lambda\}$ for every limit ordinal λ. We call f^{β} the β-th *transfinite iteration* of f.

As examples we redefine the arithmetical operations above: $\alpha+\beta = f^{\beta}(\alpha)$ with f the successor function; $\alpha \cdot \beta = g_{\alpha}^{\beta}(\underline{0})$ with $g_{\alpha}(\gamma) = \gamma + \alpha$; $\alpha^{\beta} = h_{\alpha}^{\beta}(\underline{1})$ with $h_{\alpha}(\gamma) = \gamma \cdot \alpha$.

5.3.22. Exercise. Verify the redefinition of the ordinal arithmetic is correct.

We proceed with the canonical construction for finding the least fixpoint of a weakly increasing ordinal function if there exists one.

5.3.23. Lemma. *Let f be a weakly increasing ordinal function. Then:*
(i) *$f^{\alpha+\underline{1}}(\underline{0}) \geq f^{\alpha}(\underline{0})$ for all α;*
(ii) *$f^{\alpha}(\underline{0})$ is weakly increasing in α;*
(iii) *$f^{\alpha}(\underline{0})$ does not surpass any fixpoint of f;*
(iv) *$f^{\alpha}(\underline{0})$ is strictly increasing (and hence $f^{\alpha}(\underline{0}) \geq \alpha$), until a fixpoint of f is reached, after which $f^{\alpha}(\underline{0})$ becomes constant.*

5.3.24. Exercise. Prove the above lemma. More precisely:
(i) To be proved by induction on α;
(ii) Prove $\alpha \leq \beta \;\Rightarrow\; f^{\alpha}(\underline{0}) \leq f^{\beta}(\underline{0})$ by induction on β;
(iii) Assume $f(\beta) = \beta$ and prove $f^{\alpha}(\underline{0}) \leq \beta$ by induction on α;
(iv) Prove $\alpha < \beta \;\Rightarrow\; f^{\alpha}(\underline{0}) < f^{\beta}(\underline{0})$ for all α, β such that $f^{\alpha}(\underline{0})$ is below any fixpoint, by induction on β.

If a weakly increasing ordinal function f has a fixpoint, then it has a smallest fixpoint and Lemma 5.3.23 above guarantees that this so-called *least fixpoint* is of the form $f^{\alpha}(\underline{0})$, that is, can be obtained by transfinite iteration of f starting at $\underline{0}$. This justifies the following definition.

5.3.25. Definition. Let f be a weakly increasing ordinal function having a least fixpoint which we denote by $\mathrm{lfp}(f)$. The *closure ordinal* of f is the smallest ordinal α such that $f^{\alpha}(\underline{0}) = \mathrm{lfp}(f)$.

Closure ordinals can be arbitrarily large, or may not even exist. The following lemma gives a condition under which the closure ordinal exists and does not surpass ω.

5.3.26. Lemma. *If f is a weakly increasing ordinal function such that $f(\lambda) = \bigcup\{f(\alpha) \mid \alpha < \lambda\}$ for every limit ordinal λ, then the closure ordinal exists and is at most ω.*

Proof. Let conditions be as in the lemma. Consider the sequence of finite iterations of f: $\underline{0}, f(\underline{0}), f(f(\underline{0}))$ and so on. If this sequence becomes constant, then the closure ordinal is finite. If the sequence is strictly increasing, then the supremum must be a limit ordinal, say λ. Then we have $f(\lambda) = \bigcup\{f(\alpha) \mid \alpha < \lambda\} = f^{\omega}(\underline{0}) = \lambda$, so the closure ordinal is ω. ■

5.3.27. Exercise. Justify the equation $f(\lambda) = \lambda$ in the proof above.

For example, $f(\alpha) = \underline{1} + \alpha$ has $\mathrm{lfp}(f) = \omega$, and $f(\alpha) = (\omega + \underline{1}) \cdot \alpha$ has $\mathrm{lfp}(f) = \underline{0}$. In contrast, $f(\alpha) = \alpha + \underline{1}$ has no fixpoint (note that the latter f is weakly increasing, but the condition on limit ordinals is not satisfied). Finally, $f(\alpha) = \underline{2}^\alpha$ has $\mathrm{lfp}(f) = \omega$, and the least fixpoint of $f(\alpha) = \omega^\alpha$ is denoted by ϵ_0, being the supremum of the sequence:

$$\underline{0}, \omega^{\underline{0}} = \underline{1}, \omega^{\underline{1}} = \omega, \omega^\omega, \omega^{\omega^\omega}, \omega^{\omega^{\omega^\omega}}, \ldots$$

In the following proposition we formulate some facts about ordinals that we need in the sequel.

5.3.28. Proposition. (i) *Every ordinal $\alpha < \epsilon_0$ can be written uniquely as $\alpha = \omega^{\alpha_1} + \omega^{\alpha_2} + \cdots + \omega^{\alpha_n}$ with $n \geq 0$ and $\alpha_1, \alpha_2, \ldots, \alpha_n$ a weakly decreasing sequence of ordinals smaller than α.*

(ii) *For all α, β we have $\omega^\alpha + \omega^\beta = \omega^\beta$ if and only if $\alpha < \beta$. if α is a limit and $\beta < \gamma + \underline{2}^\alpha$, then there is an $\alpha' < \alpha$ such that $\beta < \gamma + \underline{2}^{\alpha'}$.*

Proof. (i) This is a special case of Cantor normal forms with base ω, the generalization of the position system for numbers to ordinals, where terms of the form $\omega^\alpha \cdot \underline{n}$ are written as $\omega^\alpha + \cdots + \omega^\alpha$ (n summands). The fact that the exponents in the Cantor normal form are *strictly* less than α comes from the assumption that $\alpha < \epsilon_0$.

(ii) The proof of this so-called absorption property goes by a induction on β. The case $\alpha \geq \beta$ can be dealt with by using Cantor normal forms. ■

From now on *ordinal* will mean *ordinal less than* ϵ_0, unless explicitly stated otherwise. This also applies to $\forall\alpha$, $\exists\alpha$, $f(\alpha)$ and so on.

Encoding ordinals in the natural numbers

Systematic enumeration of grid points in the plane, such as shown in Figure 5.2, yields an encoding of pairs $\langle x, y\rangle$ of natural numbers x, y as given in Definition 5.3.29.

Finite sequences $[x_1, \ldots, x_k]$ of natural numbers, also called *lists*, can now be encoded by iterating the pairing function. The number 0 does not encode a pair and can hence be used to encode the empty list $[\,]$. All functions and relations involved, including projection functions to decompose pairs and lists, are easily seen to be primitive recursive.

5.3.29. Definition. Recall that $1 + 2 + \ldots + n = \frac{1}{2}n(n+1)$ gives the number of grid points satisfying $x + y < n$. The function $-$ below is to be understood as cut-off subtraction, that is, $x - y = 0$ whenever $y \geq x$. Define the following functions:

$$\begin{aligned}
\langle x, y\rangle &= \tfrac{1}{2}(x+y)(x+y+1) + x + 1 \\
\mathrm{sum}(p) &= \min\{n \mid p \leq \tfrac{1}{2}n(n+1)\} - 1 \\
x(p) &= p - \langle 0, \mathrm{sum}(p)\rangle \\
y(p) &= \mathrm{sum}(p) - x(p)
\end{aligned}$$

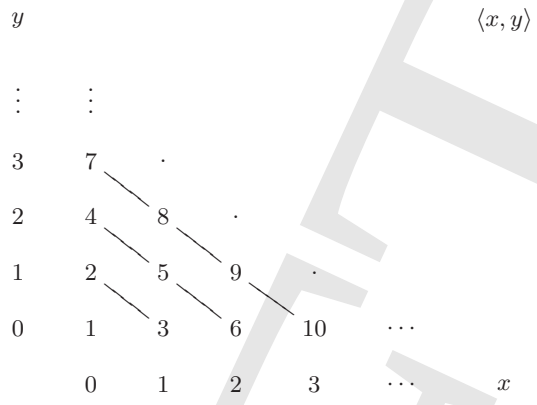


Figure 5.2: $\langle x, y\rangle$-values for $x + y \leq 3$

Now let $[\,] = 0$ and, for $k > 0$, $[x_1, \ldots, x_k] = \langle x_1, [x_2, \ldots, x_k]\rangle$ encode lists. Define $\mathtt{lth}(0) = 0$ and $\mathtt{lth}(p) = 1 + \mathtt{lth}(y(p))$ $(p > 0)$ to compute the length of a list.

The following lemma is a straithforward consequence of the above definition.

5.3.30. Lemma. *For all $p > 0$ we have $p = \langle x(p), y(p)\rangle$. Moreover, $\langle x, y\rangle > x$, $\langle x, y\rangle > y$, $\mathtt{lth}([x_1, \ldots, x_k]) = k$ and $\langle x, y\rangle$ is strictly increasing in both arguments. Every natural number encodes a unique list of smaller natural numbers. Every natural number encodes a unique list of lists of lists and so on, ending with the empty list.*

Based on the Cantor normal form and the above encoding of lists we can represent ordinals below ϵ_0 as natural numbers in the following way. We write $\overline{\alpha}$ for the natural number representing the ordinal α.

5.3.31. Definition. Let $\alpha < \epsilon_0$ have Cantor normal form $\omega^{\alpha_1} + \cdots + \omega^{\alpha_k}$. We encode α by putting $\overline{\alpha} = [\overline{\alpha_1}, \overline{\alpha_2}, \ldots, \overline{\alpha_n}]$. This representation is well-defined since every α_i $(1 \leq i \leq n)$ is strictly smaller than α. The zero ordinal $\underline{0}$, having the empty sum as Cantor normal form, is thus represented by the empty list $[\,]$, so by the natural number 0.

Examples are $\overline{\underline{0}} = [\,]$, $\overline{\underline{1}} = [[\,]]$, $\overline{\underline{2}} = [[\,], [\,]], \ldots$ and $\overline{\omega} = [[[\,]]]$, $\overline{\omega + \underline{1}} = [[[\,]], [\,]]$ and so on. Observe that $[[\,], [[\,]]]$ does not represent an ordinal as $\omega^{\underline{0}} + \omega^{\underline{1}}$ is not a Cantor normal form. The following lemmas allow one to identify which natural numbers represent ordinals and to compare them.

5.3.32. Lemma. *Let $\prec$ be the lexicographic ordering on lists. Then $\prec$ is primitive recursive and $\overline{\alpha} \prec \overline{\beta} \iff \alpha < \beta$ for all $\alpha, \beta < \epsilon_0$.*

Proof. Define $\langle x, y\rangle \prec \langle x', y'\rangle \iff (x \prec x') \vee (x = x' \wedge y \prec y')$ and $x \not\prec 0$, $0 \prec \langle x, y\rangle$. The primitive recursive relation $\prec$ is the lexicographic ordering on pairs, and hence also on lists. Now the lemma follows using Cantor normal forms. (Note that $\prec$ is not a well-order itself, as $[1] \prec [0, 1] \prec [0, 0, 1], \ldots$ has no smallest element.) ■

5.3.33. Lemma. *For all $x \in \mathbb{N}$, define:*
(i) *Ord(x) if and only if $x = \overline{\alpha}$ for some ordinal $\alpha < \epsilon_0$.*
(ii) *Succ(x) if and only if $x = \overline{\alpha}$ for some successor ordinal $< \epsilon_0$.*
(iii) *Lim(x) if and only if $x = \overline{\alpha}$ for some limit ordinal $< \epsilon_0$.*
(iv) *Fin(x) if and only if $x = \overline{\alpha}$ for some ordinal $\alpha < \omega$.*
Then Ord, Fin, Succ and Lim are primitive recursive predicates.

Proof.
(i) Put $\mathrm{Ord}(0)$ and $\mathrm{Ord}(\langle x, y\rangle) \iff (\mathrm{Ord}(x) \wedge \mathrm{Ord}(y) \wedge (y > 0 \Rightarrow x(y) \preceq x))$.
(ii) Put $\neg\mathrm{Succ}(0)$ and $\mathrm{Succ}(\langle x, y\rangle) \iff (\mathrm{Ord}(\langle x, y\rangle) \wedge (x > 0 \Rightarrow \mathrm{Succ}(y)))$.
(iii) Put $\mathrm{Lim}(x) \iff (\mathrm{Ord}(x) \wedge \neg\mathrm{Succ}(s) \wedge x \neq [\,])$.
(iv) Put $\mathrm{Fin}(x) \iff (x = [\,] \vee (x = \langle 0, y\rangle \wedge \mathrm{Fin}(y)))$. ■

5.3.34. Lemma. *There exist primitive recursive functions exp (base ω exponentiation), succ (successor), pred (predecessor), plus (addition), exp2 (base $\underline{2}$ exponentiation) such that for all α, β: exp$(\overline{\alpha}) = \overline{\omega^\alpha}$, succ$(\overline{\alpha}) = \overline{\alpha + \underline{1}}$, pred$(\overline{\underline{0}}) = \overline{\underline{0}}$, pred$(\overline{\alpha + \underline{1}}) = \overline{\alpha}$, plus$(\overline{\alpha}, \overline{\beta}) = \overline{\alpha + \beta}$, exp2$(\overline{\alpha}) = \overline{\underline{2}^\alpha}$.*

Proof. Put $\exp(x) = [x]$. Put $\mathrm{succ}(0) = \langle 0, 0\rangle$ and $\mathrm{succ}(\langle x, y\rangle) = \langle x, \mathrm{succ}(y)\rangle$, then $\mathrm{succ}([x_1, \ldots, x_k]) = [x_1, \ldots, x_k, 0]$. Put $\mathrm{pred}(0) = 0$, $\mathrm{pred}(\langle x, 0\rangle) = x$ and $\mathrm{pred}(\langle x, y\rangle) = \langle x, \mathrm{pred}(y)\rangle$ for $y > 0$. For plus, use the absorption property in adding the Cantor normal forms of α and β. For exp2 we use $\omega^\beta = \underline{2}^{\omega\cdot\beta}$. Let α have Cantor normal form $\omega^{\alpha_1} + \cdots + \omega^{\alpha_k}$. Then $\omega \cdot \alpha = \omega^{\underline{1}+\alpha_1} + \cdots + \omega^{\underline{1}+\alpha_k}$. By absorption, $\underline{1} + \alpha_i = \alpha_i$ whenever $\alpha_i \geq \omega$. It follows that we have $\alpha = \omega \cdot (\omega^{\alpha_1} + \cdots + \omega^{\alpha_i} + \omega^{\underline{n_1}} + \cdots + \omega^{\underline{n_p}}) + \underline{n}$ for suitable n_j, n with $\alpha_1 \geq \cdots \geq \alpha_i \geq \omega$, $\underline{n_j} + \underline{1} = \alpha_{i+j} < \omega$ for $1 \leq j \leq p$ and $n = k - i - p$ with $\alpha_{k'} = \underline{0}$ for all $i + p < k' \leq k$. Using $\omega^\beta = \underline{2}^{\omega\cdot\beta}$ we can calculate $\underline{2}^\alpha = \omega^\beta \cdot \underline{2^n}$ with $\beta = \omega^{\alpha_1} + \cdots + \omega^{\alpha_i} + \omega^{\underline{n_1}} + \cdots + \omega^{\underline{n_p}}$ and n as above. If $\overline{\alpha} = [x_1, \ldots, x_i, \ldots, x_j, \ldots, 0, \ldots, 0]$, then $\overline{\beta} = [x_1, \ldots, x_i, \ldots, \mathrm{pred}(x_j), \ldots]$ and we can obtain $\mathrm{exp2}(\overline{\alpha}) = \overline{\underline{2}^\alpha} = \overline{\omega^\beta \cdot \underline{2^n}}$ by doubling n times $\overline{\omega^\beta} = \exp(\overline{\beta})$ using plus. ■

5.3.35. Lemma. *There exist primitive recursive functions num, mun such that num$(n) = \overline{\underline{n}}$ and mun$(\overline{\underline{n}}) = n$ for all n. In particular we have mun(num$(n)) = n$ and num(mun$(\overline{\underline{n}})) = \overline{\underline{n}}$ for all n. In other words, num is the order isomorphism between $(\mathbb{N}, <)$ and $(\{\overline{\underline{n}} \mid n \in \mathbb{N}\}, \prec)$ and mun is the inverse order isomorphism.*

Proof. Put $\mathrm{num}(0) = 0 = [\,]$ and $\mathrm{num}(n+1) = \mathrm{succ}(\mathrm{num}(n))$ and $\mathrm{mun}(0) = 0$ and $\mathrm{mun}(\langle x, y\rangle) = \mathrm{mun}(y) + 1$. ■

5.3.36. Lemma. *There exists a primitive recursive function p such that $p(\overline{\alpha}, \overline{\beta}, \overline{\gamma}) = \overline{\alpha'}$ with $\alpha' < \alpha$ and $\beta < \gamma + \underline{2}^{\alpha'}$, provided that α is a limit and $\beta < \gamma + \underline{2}^\alpha$.*

Proof. Let conditions be as above. The existence of α' follows directly from the definition of the operations of ordinal arithmetic on limit ordinals. The interesting point, however, is that $\overline{\alpha'}$ can be computed from $\overline{\alpha}, \overline{\beta}, \overline{\gamma}$ in a primitive recursive way, as will become clear by the following argument. If $\beta \leq \gamma$, then

we can simply take $\alpha' = \underline{0}$. Otherwise, let $\beta = \omega^{\beta_1} + \cdots + \omega^{\beta_n}$ and $\gamma = \omega^{\gamma_1} + \cdots + \omega^{\gamma_m}$ be Cantor normal forms. Now $\gamma < \beta$ implies that $\gamma_i < \beta_i$ for some smallest index $i \leq m$, or no such index exists. In the latter case we have $m < n$ and $\gamma_j = \beta_j$ for all $1 \leq j \leq m$, and we put $i = m + 1$. Since α is a limit, we have $\alpha = \omega \cdot \xi$ (see ...) for suitable ξ, and hence $\underline{2}^\alpha = \omega^\xi$. Since $\beta < \gamma + \underline{2}^\alpha$ it follows by absorption that $\omega^{\beta_i} + \cdots + \omega^{\beta_n} < \omega^\xi$. Hence $\beta_i + \underline{1} \leq \xi$, so $\omega^{\beta_i} + \cdots + \omega^{\beta_n} \leq \omega^{\beta_i} \cdot \underline{n} < \omega^{\beta_i} \cdot \underline{2}^{\underline{n}} = \underline{2}^{\omega \cdot \beta_i + \underline{n}}$. Now take $\alpha' = \omega \cdot \beta_i + \underline{n} < \omega \cdot (\beta_i + \underline{1}) \leq \omega \cdot \xi = \alpha$ and observe $\beta < \gamma + \underline{2}^{\alpha'}$. ■

From now on we will freely use ordinals in the natural numbers instead of their codes. This includes uses like α *is finite* instead of $\mathrm{Fin}(\overline{\alpha})$, $\alpha \prec \beta$ instead of $\overline{\alpha} \prec \overline{\beta}$, and so on. Note that we avoid using $<$ for ordinals now, as it would be ambiguous. Phrases like $\forall \alpha\ P(\alpha)$ and $\exists \alpha\ P(\alpha)$ should be taken as relativized quantifications over natural numbers, that is, $\forall x\ (\mathrm{Ord}(x) \Rightarrow P(x))$, and $\exists x\ (\mathrm{Ord}(x) \wedge P(x))$, respectively. Finally, functions defined in terms of ordinals are assumed to take value 0 for arguments that do not encode any ordinal.

Transfinite induction and recursion

Transfinite induction (TI) is a principle of proof that generalizes the usual schema of structural induction from natural numbers to ordinals. Define:

$$\begin{aligned} \mathrm{Ind}(P) &\equiv \forall \alpha\ ((\forall \beta < \alpha\ P(\beta)) \Rightarrow P(\alpha)) \\ \mathrm{TI}_\alpha &\equiv \mathrm{Ind}(P) \Rightarrow \forall \beta < \alpha\ P(\beta) \end{aligned}$$

Here $\mathrm{Ind}(P)$ expresses that P is *inductive*, that is, $\forall \beta < \alpha\ P(\beta)$ induces $P(\alpha)$ for all ordinals α. For proving a property P to be inductive it suffices to prove $(\forall \beta < \alpha\ P(\beta)) \Rightarrow P(\alpha)$ for limit ordinals α only, in addition to $P(\underline{0})$ and $P(\alpha) \Rightarrow P(\alpha + \underline{1})$ for all α. If a property is inductive then TI_γ implies that every ordinal up to γ has this property. (For the latter conclusion, in fact inductivity up to γ suffices. Note that ordinals may exceed ϵ_0 in this paragraph.)

By Lemma 5.3.35, TI_ω is equivalent to structural induction on the natural numbers. Obviously, the strength of TI_α increases with α. Therefore TI_α can be used to measure the proof theoretic strength of theories. Given a theory T, for which α can we prove TI_α? We shall show that TI_α is provable in Peano Arithmetic for all ordinals $\alpha < \epsilon_0$ by a famous argument due to Gentzen.

The computational counterpart of transfinite induction is transfinite recursion TR, a principle of definition which can be used to measure computational strength. By a translation of Gentzen's argument we shall show that every function which can be defined by TR_α for some ordinal $\alpha < \epsilon_0$, is definable in Gödel's $\mathcal{T}$. Thus we have established a lower bound to the computational strength of Gödel's $\mathcal{T}$.

5.3.37. Lemma. *The schema TI_ω is provable in Peano Arithmetic.*

Proof. Observe that TI_ω is structural induction on an isomorphic copy of the natural numbers by Lemma 5.3.35. ■

5.3.38. Lemma. *The schema $TI_{\omega\cdot\underline{2}}$ is provable in Peano Arithmetic with the schema TI_ω.*

Proof. Assume TI_ω and $\mathrm{Ind}(P)$ for some P. In order to prove $\forall\alpha<\omega\cdot\underline{2}\ P(\alpha)$ define $P'(\alpha)\equiv\forall\beta<\omega+\alpha\ P(\beta)$. By TI_ω we have $P'(\underline{0})$. Also $P'(\alpha)\ \Rightarrow\ P'(\alpha+\underline{1})$, as $P'(\alpha)$ implies $P(\omega+\alpha)$ by $\mathrm{Ind}(P)$. If $\mathrm{Lim}(\alpha)$, then $\beta<\omega+\alpha$ implies $\beta<\omega+\alpha'$ for some $\alpha'<\alpha$, and hence $P'(\alpha')\ \Rightarrow\ P(\beta)$. It follows that P' is inductive, which can be combined with TI_ω to conclude $P'(\omega)$, so $\forall\beta<\omega+\omega\ P(\beta)$. This completes the proof of $\mathrm{TI}_{\omega\cdot\underline{2}}$. ■

5.3.39. Lemma. *The schema $TI_{\underline{2}^\alpha}$ is provable in Peano Arithmetic with the schema TI_α, for all $\alpha<\epsilon_0$.*

Proof. Assume TI_α and $\mathrm{Ind}(P)$ for some P. In order to prove $\forall\alpha'<\underline{2}^\alpha\ P(\alpha')$ define $P'(\alpha')\equiv\forall\beta(\forall\beta'<\beta\ P(\beta')\ \Rightarrow\ \forall\beta'<\beta+\underline{2}^{\alpha'}\ P(\beta'))$. The intuition behind $P'(\alpha')$ is: if P holds on an arbitrary initial segment, then we can prolong this segment with $\underline{2}^{\alpha'}$. The goal will be to prove $P'(\alpha)$, since we can then prolong the empty inititial segment on which P vacuously holds to one of length $\underline{2}^\alpha$. We prove $P'(\alpha)$ by proving first that P' is inductive and then combining this with TI_α, similar to the proof of the previous lemma. We have $P'(\underline{0})$ as P is inductive and $\underline{2}^{\underline{0}}=\underline{1}$. The argument for $P'(\alpha)\ \Rightarrow\ P'(\alpha+\underline{1})$ amounts to applying $P'(\alpha)$ twice, relying on $\underline{2}^{\alpha+\underline{1}}=\underline{2}^\alpha+\underline{2}^\alpha$. Assume $P'(\alpha)$ and $\forall\beta'<\beta\ P(\beta')$ for some β. By $P'(\alpha)$ we have $\forall\beta'<\beta+\underline{2}^\alpha\ P(\beta')$. Hence again by $P'(\alpha)$, but now with $\beta+\underline{2}^\alpha$ instead of β, we have $\forall\beta'<\beta+\underline{2}^\alpha+\underline{2}^\alpha\ P(\beta')$. We conclude $P'(\alpha+\underline{1})$. The limit case is equally simple as in the previous lemma. It follows that P' is inductive, and the proof can be completed as explained above. ■

The general idea of the above proofs is that the stronger axiom schema is proved by applying the weaker schema to more complicated formulas (P' as compared to P). This procedure can be iterated as long as the more complicated formulas remain well-formed. In the case of Peano arithmetic we can iterate this procedure finitely many times. This yields the following result.

5.3.40. Lemma (Gentzen). *TI_α is provable in Peano Arithmetic for every ordinal $\alpha<\epsilon_0$.*

Proof. Use $\omega^\beta=\underline{2}^{\omega\cdot\beta}$, so $\underline{2}^{\omega\cdot\underline{2}}=\omega^{\underline{2}}$ and $\underline{2}^{\omega^{\underline{2}}}=\omega^\omega$. From ω^ω on, iterating exponentiation with base $\underline{2}$ yields the same ordinals as with base ω. We start with Lemma 5.3.37 to obtain TI_ω, continue with Lemma 5.3.38 to obtain $\mathrm{TI}_{\omega\cdot\underline{2}}$, and surpass TI_α for every ordinal $\alpha<\epsilon_0$ by iterating Lemma 5.3.39 a sufficient number of times. ■

We now translate the Gentzen argument from transfinite induction to transfinite recursion, closely following the development of Terlouw [1982].

5.3.41. Definition. For any functional F of type $0\to A$ and ordinals α,β we define primitive recursively

$$[F]^\alpha_\beta(\beta')=\begin{cases}F(\beta') & \text{if } \beta'\prec\beta\preceq\alpha,\\ 0^A & \text{otherwise.}\end{cases}$$

By convention, 'otherwise' includes the cases in which α, β, β' are not ordinals, and the case in which $\alpha \prec \beta$. Furthermore, we define $[F]_\alpha = [F]^\alpha_\alpha$, that is, the functional F restricted to an initial segment of ordinals smaller than α.

5.3.42. Definition. The class of functionals *definable by* TR_α is the smallest class of functionals which contains all primitive recursive functionals and is closed under the definition schema TR_α, defining F from G (of appropriate types) in the following way:

$$F(\beta) = G([F]^\alpha_\beta, \beta)$$

Note that, by the above definition, $F(\beta) = G(0^{0\to A}, \beta)$ if $\alpha \prec \beta$ or if the argument of F does not encode an ordinal.

The following lemma is to be understood as the computational counterpart of Lemma 5.3.38, with the primitive recursive functionals taking over the role of Peano Arithmetic.

5.3.43. Lemma. *Every functional definable by the schema TR_ω is $\mathcal{T}$-definable.*

Proof. Let $F_0(\alpha) = G([F_0]^\omega_\alpha, \alpha)$ be defined by TR_ω. We have to show that F_0 is $\mathcal{T}$-definable. Define primitive recursively F_1 by $F_1(0) = 0^{0\to A}$ and

$$F_1(n+1, \alpha) \quad = \quad \begin{cases} F_1(n, \alpha) & \text{if } \alpha < \underline{n} \\ G([F_1(n)]^\omega_\alpha, \alpha) & \text{otherwise} \end{cases}$$

By induction one shows $[F_0]^\omega_{\underline{n}} = [F_1(n)]^\omega_{\underline{n}}$ for all n. Define primitive recursively F_2 by $F_2(\underline{n}) = F_1(n+1, \underline{n})$ and $F_2(\alpha) = 0^A$ if α is not a finite ordinal, then $F_2 = [F_0]^\omega_\omega$. Now it is easy to define F_0 explicitly in F_2

$$F_0(\alpha) \quad = \quad \begin{cases} F_2(\alpha) & \text{if } \alpha < \omega \\ G(F_2, \omega) & \text{if } \alpha = \omega \\ G(0^{0\to A}, \alpha) & \text{otherwise} \end{cases}$$

Note that we used both num and mun implicitly in the definition of F_2. ■

The general idea of the proofs below is that the stronger schema is obtained by applying the weaker schema to functionals of more complicated types.

5.3.44. Lemma. *Every functional definable by the schema $TR_{\omega\cdot\underline{2}}$ is definable by the schema TR_ω.*

Proof. Put $\omega \cdot \underline{2} = \alpha$ and let $F_0(\beta) = G([F_0]^\alpha_\beta, \beta)$ be defined by TR_α. We have to show that F_0 is definable by TR_ω (applied with functionals of more complicated types). First define $F_1(\beta) = G([F_1]^\omega_\beta, \beta)$ by TR_ω. Then we can prove $F_1(\beta) = F_0(\beta)$ for all $\beta < \omega$ by TI_ω. So we have $[F_1]_\omega = [F_0]_\omega$, which is to be compared to $P'(\underline{0})$ in the proof of Lemma 5.3.38. Now define H of type $0\to(0\to A)\to(0\to A)$ by TR_ω as follows. The more complicated type of H

as compared to the type $0\to A$ of F is the counterpart of the more complicated formula P' as compared to P in the proof of Lemma 5.3.38.

$$\begin{array}{rcl} H(\underline{0},F) & = & [F_1]_\omega \\ H(\beta+\underline{1},F,\beta') & = & \left\{\begin{array}{ll} H(\beta,F,\beta') & \text{if } \beta'<\omega+\beta \\ G(H(\beta,F),\beta') & \text{if } \beta'=\omega+\beta \\ 0^A & \text{otherwise}\end{array}\right. \end{array}$$

This definition can easily be casted in the form $H(\beta)=G'([H]^\omega_\beta,\beta)$ for suitable G', so that H is actually defined by TR_ω. We can prove $H(\beta,0^{0\to A})=[F_0]^\alpha_{\omega+\beta}$ for all $\beta<\omega$ by TI_ω. Finally we define

$$F_2(\beta') \;=\; \left\{\begin{array}{ll} F_1(\beta') & \text{if } \beta'<\omega \\ G(H(\beta,0^{0\to A}),\beta') & \text{if } \beta'=\omega+\beta<\alpha \\ G(0^{0\to A},\beta') & \text{otherwise}\end{array}\right.$$

Note that F_2 is explicitly defined in G and H and therefore defined by TR_ω only. One easily shows that $F_2=F_0$, which completes the proof of the lemma. ■

5.3.45. Lemma. *Every functional definable by the schema $TR_{\underline{2}^\alpha}$ is definable by the schema TR_α, for all $\alpha<\epsilon_0$.*

Proof. Let $F_0(\beta)=G([F_0]^{\underline{2}^\alpha}_\beta,\beta)$ be defined by $\mathrm{TR}_{\underline{2}^\alpha}$. We have to show that F_0 is definable by TR_α (applied with functionals of more complicated types). Like in the previous proof, we will define by TR_α an auxiliary functional H in which F_0 can be defined explicitly. The complicated type of H compensates for the weaker definition principle. The following property satisfied by H is to be understood in the same way as the property P' in the proof of Lemma 5.3.39, namely that we can prolong initial segments with $\underline{2}^\alpha$.

$$\mathit{propH}(\alpha')\equiv\forall\beta,F\;([F]^{\underline{2}^\alpha}_\beta=[F_0]^{\underline{2}^\alpha}_\beta\;\Rightarrow\;[H(\alpha',\beta,F)]^{\underline{2}^\alpha}_{\beta+\underline{2}^{\alpha'}}=[F_0]^{\underline{2}^\alpha}_{\beta+\underline{2}^{\alpha'}})$$

To make *propH* come true, define H of type $0\to0\to(0\to A)\to(0\to A)$ as follows.

$$H(\underline{0},\beta,F,\beta')=\left\{\begin{array}{ll} F(\beta') & \text{if } \beta'<\beta\le\underline{2}^\alpha \\ G([F]^{\underline{2}^\alpha}_\beta,\beta) & \text{if } \beta'=\beta\le\underline{2}^\alpha \\ 0^A & \text{otherwise}\end{array}\right.$$

$$H(\alpha'+\underline{1},\beta,F)=H(\alpha',\beta+\underline{2}^{\alpha'},H(\alpha',\beta,F))$$

If α' is a limit ordinal, then we use the function p from Lemma 5.3.36.

$$H(\alpha',\beta,F,\beta')=\left\{\begin{array}{ll} H(\mathrm{p}(\alpha',\beta',\beta),\beta,F,\beta') & \text{if } \beta'<\beta+\underline{2}^{\alpha'} \\ 0^A & \text{otherwise}\end{array}\right.$$

This definition can easily be casted in the form $H(\beta)=G'([H]^\alpha_\beta,\beta)$ for suitable G', so that H is in fact defined by TR_α. We shall prove that $\mathit{propH}(\alpha')$ is inductive, and conclude $\mathit{propH}(\alpha')$ forall $\alpha'\le\alpha$ by TI_α. This implies

$[H(\alpha', \underline{0}, 0^{0\to A})]^{\underline{2}^\alpha}_{\underline{2}^{\alpha'}} = [F_0]^{\underline{2}^\alpha}_{\underline{2}^{\alpha'}}$ for all $\alpha' \leq \alpha$, so that one could manufacture F_0 from H in the following way:

$$F_0(\beta) \quad = \quad \begin{cases} H(\alpha, \underline{0}, 0^{0\to A}, \beta) & \text{if } \beta < \underline{2}^\alpha \\ G(H(\alpha, \underline{0}, 0^{0\to A}), \beta) & \text{if } \beta = \underline{2}^\alpha \\ G(0^{0\to A}, \beta) & \text{otherwise} \end{cases}$$

It remains to show that $\mathit{propH}(\alpha')$ is inductive up to and including α. For the case $\alpha' = \underline{0}$ we observe that $H(\underline{0}, \beta, F)$ follows F up to β, applies G to the initial segment of $[F]^{\underline{2}^\alpha}_\beta$ in β, and zeroes after β. This entails $\mathit{propH}(\underline{0})$, as $\underline{2}^{\underline{0}} = \underline{1}$. Analogous to the successor case in the proof of Lemma 5.3.39, we prove $\mathit{propH}(\alpha + \underline{1})$ by applying $\mathit{propH}(\alpha)$ twice, once with β and once with $\beta + \underline{2}^\alpha$. Given β and F we infer:

$$[F]^{\underline{2}^\alpha}_\beta = [F_0]^{\underline{2}^\alpha}_\beta \;\Rightarrow\; [H(\alpha', \beta, F)]^{\underline{2}^\alpha}_{\beta+\underline{2}^{\alpha'}} = [F_0]^{\underline{2}^\alpha}_{\beta+\underline{2}^{\alpha'}} \;\Rightarrow$$
$$[H(\alpha', \beta + \underline{2}^{\alpha'}, H(\alpha', \beta, F))]^{\underline{2}^\alpha}_{\beta+\underline{2}^{\alpha'+\underline{1}}} = [F_0]^{\underline{2}^\alpha}_{\beta+\underline{2}^{\alpha'+\underline{1}}}$$

For the limit case, assume $\alpha' \leq \alpha$ is a limit ordinal such that propH holds for all smaller ordinals. Recall that, according to Lemma 5.3.36 and putting $\alpha'' = \mathrm{p}(\alpha', \beta', \beta)$, $\alpha'' < \alpha'$ and $\beta' < \beta + \underline{2}^{\alpha''}$ whenever $\beta' < \beta + \underline{2}^{\alpha'}$. Now assume $[F]^{\underline{2}^\alpha}_\beta = [F_0]^{\underline{2}^\alpha}_\beta$ and $\beta' < \beta + \underline{2}^{\alpha'}$, then $[H(\alpha'', \beta, F)]^{\underline{2}^\alpha}_{\beta+\underline{2}^{\alpha''}} = [F_0]^{\underline{2}^\alpha}_{\beta+\underline{2}^{\alpha''}}$ by $\mathit{propH}(\alpha'')$, so $H(\alpha'', \beta, F, \beta') = F_0(\beta')$. It follows that $[H(\alpha', \beta, F)]^{\underline{2}^\alpha}_{\beta+\underline{2}^{\alpha'}} = [F_0]^{\underline{2}^\alpha}_{\beta+\underline{2}^{\alpha'}}$. ■

5.3.46. Lemma. *Every functional definable by the schema TR_α for some ordinal $\alpha < \epsilon_0$ is $\mathcal{T}$-definable.*

Proof. Analogous to the proof of Lemma 5.3.40. ■

Lemma 5.3.46 establishes ϵ_0 as a lower bound for the computational strength of Gödel's $\mathcal{T}$. It can be shown that ϵ_0 is a sharp bound for $\mathcal{T}$, see Tait [1965], Howard [1970] and Schwichtenberg [1975]. In the next section we will introduce Spector's system $\mathcal{B}$. It is also known that $\mathcal{B}$ is much stronger than $\mathcal{T}$, lower bounds have been established for subsystems of $\mathcal{B}$, but the computational strength of $\mathcal{B}$ in terms of ordinals remains one of the great open problems in this field.

5.4. Spector's system $\mathcal{B}$: bar recursion

Spector [1962] extends Gödel's $\mathcal{T}$ with a definition schema called bar recursion.[3] Bar recursion is a principle of definition by recursion on a well-founded tree of finite sequences of functionals of the same type. For the formulation of bar recursion we need finite sequences of functionals of type A. These can

[3] For the purpose of characterizing the provably recursive functions of analysis, yielding a consistency proof of analysis.

conveniently be encoded by pairs consisting of a functional of type N and one of type $\mathsf{N}\to A$. The intuition is that x, C encode the sequence of the first x values of C, that is, $C(0), \ldots, C(x-1)$. We need auxiliary functionals to extend finite sequences of any type. A convenient choice is the primitive recursive functional $\mathrm{Ext}_A : (\mathsf{N}\to A)\to\mathsf{N}\to A\to\mathsf{N}\to A$ defined by:

$$\mathrm{Ext}_A(C, x, A, y) = \begin{cases} C(y) & \text{if } y < x, \\ A & \text{otherwise.} \end{cases}$$

We shall often omit the type subscript in Ext_A, and abbreviate $\mathrm{Ext}(C, x, A)$ by $C *_x A$ and $\mathrm{Ext}(C, x, \mathsf{0}^A)$ by $[C]_x$. We are now in a position to formulate the schema of bar recursion:[4]

$$\varphi(x, C) = \begin{cases} G(x, C) & \text{if } Y[C]_x < x, \\ H(\lambda a^A.\varphi(x+1, C *_x a), x, C) & \text{otherwise.} \end{cases}$$

The case distinction is governed by $Y[C]_x < x$, the so-called *bar condition*. The base case of bar recursion is the case in which the bar condition holds. In the other case φ is recursively called on all extensions of the (encoded) finite sequence.

A key feature of bar recursion is its proof theoretic strength as established by Spector[1962]. As a consequence, some properties of bar recursion are hard to prove, such as SN and the existence of a model. As an example of the latter phenomenon we shall show that the full set theoretic model of Gödel's $\mathcal{T}$ is not a model of bar recursion.

Consider functionals Y, G, H defined by $G(x, C) = 0$, $H(Z, x, C) = 1 + Z(1)$ and

$$Y(F) = \begin{cases} 0 & \text{if } F(m) = 1 \text{ for all } m, \\ n & \text{otherwise, where } n = \min\{m \mid F(m) \neq 1\}. \end{cases}$$

Let $\mathsf{1}^{\mathsf{N}\to\mathsf{N}}$ be the constant 1 function. The crux of Y is that $Y[\mathsf{1}^{\mathsf{N}\to\mathsf{N}}]_x = x$ for all x, so that the bar recursion is not well-founded. We calculate

$$\varphi(0, \mathsf{1}^{\mathsf{N}\to\mathsf{N}}) = 1 + \varphi(1, \mathsf{1}^{\mathsf{N}\to\mathsf{N}}) = \ldots = n + \varphi(n, \mathsf{1}^{\mathsf{N}\to\mathsf{N}}) = \ldots$$

which shows that φ is not well-defined.

Syntax of λ_B

In this section we formalize Spector's $\mathcal{B}$ as an extension of Gödel's $\mathcal{T}$ called λ_B.

5.4.1. Definition. The *types* of λ_B are the types of λ_T. We use A^{N} as shorthand for the type $\mathsf{N}\to A$. The *terms* of λ_B are obtained by adding constants

$$\begin{aligned} \mathsf{B}(A, B) &: (A^{\mathsf{N}}\to\mathsf{N})\to(\mathsf{N}\to A^{\mathsf{N}}\to B)\to((A\to B)\to\mathsf{N}\to A^{\mathsf{N}}\to B)\to\mathsf{N}\to A^{\mathsf{N}}\to B \\ \mathsf{B}^c_{A,B} &: (A^{\mathsf{N}}\to\mathsf{N})\to(\mathsf{N}\to A^{\mathsf{N}}\to B)\to((A\to B)\to\mathsf{N}\to A^{\mathsf{N}}\to B)\to\mathsf{N}\to A^{\mathsf{N}}\to\mathsf{N}\to B \end{aligned}$$

[4] Spector uses $[C]_x$ instead of C as last argument of G and H. Both formulations are easily seen to be equivalent since they are schematic in G, H (as well as in Y).

for all types A, B to the constants of λ_T. The set of (closed) terms of λ_B (of type A) is denoted with $\Lambda_B^{(0)}(A)$. The *formulas* of λ_B are equations between terms of λ_B (of the same type). The *theory* of λ_B extends the theory of λ_T with the above schema of bar recursion (with φ abbreviating $\mathsf{B}YGH$). The *reduction relation* $\to_B$ of λ_B extends $\to_T$ by adding the following (schematic) rules for the constants B, B^c (omitting type annotations A, B):

$$\begin{array}{rcl} \mathsf{B}YGHXC & \to_B & \mathsf{B}^cYGHXC(X \dot{-} [C]_X) \\ \mathsf{B}^cYGHXC(\mathsf{S}^+N) & \to_B & GXC \\ \mathsf{B}^cYGHXC\mathsf{0} & \to_B & H(\lambda a.\mathsf{B}YGH(\mathsf{S}^+X)(C *_X a))XC \end{array}$$

The reduction rules for B, B^c require some explanation. First observe that $x \dot{-} Y[C]_x = 0$ iff $Y[C]_x \geq x$, so that testing $x \dot{-} Y[C]_x = 0$ amounts to evaluating the (negation) of the bar condition. Consider a primitive recursive functional If_B satisfying $\mathrm{If}_B \mathsf{0} M_1 M_0 = M_0$ and $\mathrm{If}_B(\mathsf{S}^+P)M_1M_0 = M_1$. A straightforward translation of the definition schema of bar recursion into a reduction rule:

$$\mathsf{B}YGHXC \;\to\; \mathrm{If}\,(X \dot{-} [C]_X)(GXC)(H(\lambda x.\mathsf{B}YGH(\mathsf{S}^+X)(C *_X x))XC)$$

would lead to infinite reduction sequences (the innermost B can be reduced again and again). It turns out to be necessary to evaluate the boolean first. This has been achieved by the interplay between B and B^c.

Theorem 5.3.9, Lemma 5.3.10 and Theorem 5.3.12 carry over from λ_T to λ_B with proofs that are easy generalizations. We now prove SN for λ_B and then obtain CR for λ_B using Newman's Lemma 5.3.14. The proof of SN for λ_B is considerably more difficult than for λ_T, which reflects the metamathematical fact that λ_B corresponds to analysis (see Spector [1962]), whereas λ_T corresponds to arithmetic. We start with defining hereditary finiteness for *sets* of terms, an analytical notion which plays a similar role as the arithmetical notion of computability for *terms* in the case of λ_T. Both are logical relations in the sense of Section 3.3, although hereditary finiteness is defined on the power set. Both computability and hereditary finiteness strengthen the notion of strong normalization, both are shown to hold by induction on terms. For metamathematical reasons, notably the consistency of analysis, it should not come as a surprise that we need an analytical induction loading in the case of λ_B.

5.4.2. Definition. For every set $\mathcal{X} \subseteq \Lambda_B$, let $nf(\mathcal{X})$ denote the set of normal forms of terms from $\mathcal{X}$. For all $\mathcal{X} \subseteq \Lambda_B(A{\to}B)$ and $\mathcal{Y} \subseteq \Lambda_B(A)$, let $\mathcal{X}\mathcal{Y}$ denote the set of all applications of terms from $\mathcal{X}$ to terms from $\mathcal{Y}$. Furthermore, if $M(x_1, \ldots, x_k)$ is a term with free variables $x_1, \ldots, x_k$, and $\mathcal{X}_1, \ldots, \mathcal{X}_k$ are sets of terms such that every term from $\mathcal{X}_i$ has the same type as x_i $(1 \leq i \leq k)$, then we denote the set of all corresponding substitution instances by $M(\mathcal{X}_1, \ldots, \mathcal{X}_k)$.

By induction on the type A we define $\mathcal{X} \in \mathcal{HF}_A$, expressing that the set $\mathcal{X}$ of closed terms of type A is *hereditarily finite*.

$$\begin{array}{rcl} \mathcal{X} \in \mathcal{HF}_{\mathsf{N}} & \iff & \mathcal{X} \subseteq \Lambda_B^{\emptyset}(\mathsf{N}) \cap \mathrm{SN} \text{ and } nf(\mathcal{X}) \text{ is finite} \\ \mathcal{X} \in \mathcal{HF}_{A\to B} & \iff & \mathcal{X} \subseteq \Lambda_B^{\emptyset}(A{\to}B) \text{ and } \mathcal{X}\mathcal{Y} \in \mathcal{HF}_B \text{ whenever } \mathcal{Y} \in \mathcal{HF}_A \end{array}$$

A closed term M is hereditarily finite, denoted by $M \in \mathrm{HF}^0$, if $\{M\} \in \mathcal{HF}$. If $M(x_1, \ldots, x_k)$ is a term all whose free variables occur among $x_1, \ldots, x_k$, then $M(x_1, \ldots, x_k)$ is hereditarily finite, denoted by $M(x_1, \ldots, x_k) \in \mathrm{HF}$, if $M(\mathcal{X}_1, \ldots, \mathcal{X}_k)$ is hereditarily finite for all $\mathcal{X}_i \in \mathcal{HF}$ of appropriate types ($1 \leq i \leq k$).

Some basic properties of hereditary finiteness are summarized in the following lemmas. We use vector notation to abbreviate sequences of arguments of appropriate types both for terms and for sets of terms. For example, $M\vec{N}$ abbreviates $MN_1 \ldots N_k$ and $\mathcal{X}\vec{\mathcal{Y}}$ stands for $\mathcal{X}\mathcal{Y}_1 \ldots \mathcal{Y}_k$. The first two lemmas are instrumental for proving hereditary finiteness.

5.4.3. Lemma. *$\mathcal{X} \subseteq \Lambda^{\emptyset}_B(A_1 \to \cdots \to A_n \to \mathsf{N})$ is hereditarily finite if and only if $\mathcal{X}\vec{\mathcal{Y}} \in \mathcal{HF}_{\mathsf{N}}$ for all $\mathcal{Y}_1 \in \mathcal{HF}_{A_1}, \ldots, \mathcal{Y}_n \in \mathcal{HF}_{A_n}$.*

Proof. By induction on n, applying Definition 5.4.2. ■

5.4.4. Definition. Given two sets of terms $\mathcal{X}, \mathcal{X}' \subseteq \Lambda^{\emptyset}_B$, we say that $\mathcal{X}$ is *adfluent* with $\mathcal{X}'$ if every maximal reduction sequence starting in $\mathcal{X}$ passes through a reduct of a term in $\mathcal{X}'$. Let $A \equiv A_1 \to \cdots \to A_n \to \mathsf{N}$ with $n \geq 0$ and let $\mathcal{X}, \mathcal{X}' \subseteq \Lambda^{\emptyset}_B(A)$. We say that $\mathcal{X}$ is *hereditarily adfluent* with $\mathcal{X}'$ if $\mathcal{X}\vec{\mathcal{Y}}$ is adfluent with $\mathcal{X}'\vec{\mathcal{Y}}$, for all $\mathcal{Y}_1 \in \mathcal{HF}_{A_1}, \ldots, \mathcal{Y}_n \in \mathcal{HF}_{A_n}$.

5.4.5. Lemma. *Let $\mathcal{X}, \mathcal{X}' \subseteq \Lambda^{\emptyset}_B(A)$ be such that $\mathcal{X}$ is hereditarily adfluent with $\mathcal{X}'$. Then $\mathcal{X} \in \mathcal{HF}_A$ whenever $\mathcal{X}' \in \mathcal{HF}_A$.*

Proof. Let conditions be as in the lemma and assume $\mathcal{X}' \in \mathcal{HF}_A$ with $A \equiv A_1 \to \cdots \to A_n \to \mathsf{N}$. Let $\mathcal{Y}_1 \in \mathcal{HF}_{A_1}, \ldots, \mathcal{Y}_n \in \mathcal{HF}_{A_n}$, then $\mathcal{X}\vec{\mathcal{Y}}$ is adfluent with $\mathcal{X}'\vec{\mathcal{Y}}$. It follows that $\mathcal{X}\vec{\mathcal{Y}} \subseteq \mathrm{SN}$ since $\mathcal{X}'\vec{\mathcal{Y}} \subseteq \mathrm{SN}$ and $nf(\mathcal{X}\vec{\mathcal{Y}}) \subseteq nf(\mathcal{X}'\vec{\mathcal{Y}})$, so $nf(\mathcal{X}\vec{\mathcal{Y}})$ is finite since $nf(\mathcal{X}'\vec{\mathcal{Y}})$ is. Applying Lemma 5.4.3 we obtain $\mathcal{X} \in \mathcal{HF}_A$. ■

Note that the above lemma holds in particular if $n = 0$, that is, if $A \equiv \mathsf{N}$.

5.4.6. Lemma. *For every type A we have: (i) $\mathrm{HF}_A \subseteq \mathrm{SN}$ and (ii) $\mathsf{0}^A \in \mathrm{HF}_A$.*

Proof. We prove (ii) and (iii) $\mathrm{HF}^0_A \subseteq \mathrm{SN}$ by simultaneous induction on A. Then (i) follows immediately. Obviously, $\mathsf{0} \in \mathrm{HF}_{\mathsf{N}}$ and $\mathrm{HF}^0_{\mathsf{N}} \subseteq \mathrm{SN}$. For the induction step $A \to B$, assume (ii) and (iii) hold for all smaller types. If $M \in \mathrm{HF}^0_{A \to B}$, then by the induction hypothesis (ii) $\mathsf{0}^A \in \mathrm{HF}^0_A$, so $M\mathsf{0}^A \in \mathrm{HF}^0_B$, so $M\mathsf{0}^A$ is SN by the induction hypothesis (iii), and hence M is SN. Recall that $\mathsf{0}^{A \to B} \equiv \lambda x^A.\mathsf{0}^B$. We use Lemma 5.4.3 to prove the induction step for (ii). Let $\mathcal{X} \in \mathcal{HF}_A$, then $\mathcal{X} \subseteq \mathrm{SN}$ by the induction hypothesis. It follows that $\mathsf{0}^{A \to B}\mathcal{X}$ is hereditarily adfluent with $\mathsf{0}^B$. By the induction hypothesis we have $\mathsf{0}^B \in \mathrm{HF}_B$, so $\mathsf{0}^{A \to B}\mathcal{X} \in \mathcal{HF}_B$ by Lemma 5.4.5. It follows that $\mathsf{0}^{A \to B} \in \mathrm{HF}_{A \to B}$. ■

The proofs of the following three lemmas are left to the reader.

5.4.7. Lemma. *Every reduct of a hereditarily finite term is hereditarily finite.*

5.4.8. Lemma. *Subsets of hereditarily finite sets of terms are hereditarily finite.*

In particular elements of a hereditarily finite set are hereditarily finite.

5.4.9. Lemma. *Finite unions of hereditarily finite sets are hereditarily finite.*

In this connection of course only unions of the same type make sense.

5.4.10. Exercise. Prove the above three lemmas.

5.4.11. Lemma. *The hereditarily finite terms are closed under application.*

Proof. Immediate from Definition 5.4.2. ■

5.4.12. Lemma. *The hereditarily finite terms are closed under lambda abstraction.*

Proof. Let $M(x, x_1, \ldots, x_k) \in \mathrm{HF}$ be a term all whose free variables occur among $x, x_1, \ldots, x_k$. We have to prove $\lambda x.M(x, x_1, \ldots, x_k) \in \mathrm{HF}$, that is,

$$\lambda x.M(x, \mathcal{X}_1, \ldots, \mathcal{X}_k) \in \mathcal{HF}$$

for given $\vec{\mathcal{X}} = \mathcal{X}_1, \ldots, \mathcal{X}_k \in \mathcal{HF}$ of appropriate types. Let $\mathcal{X} \in \mathcal{HF}$ be of the same type as the variable x, so $\mathcal{X} \subseteq \mathrm{SN}$ by Lemma 5.4.6. We also have $M(x, \vec{\mathcal{X}}) \subseteq \mathrm{SN}$ by the assumption on M and Lemma 5.4.6. It follows that $(\lambda x.M(x, \vec{\mathcal{X}})\mathcal{X}$ is hereditarily adfluent with $M(\mathcal{X}, \vec{\mathcal{X}})$. Again by the assumption on M we have that $M(\mathcal{X}, \vec{\mathcal{X}}) \in \mathcal{HF}$, so that $(\lambda x.M(x, \vec{\mathcal{X}}))\mathcal{X} \in \mathcal{HF}$ by Lemma 5.4.5. We conclude that $\lambda x.M(x, \vec{\mathcal{X}}) \in \mathcal{HF}$, so $\lambda x.M(x, x_1, \ldots, x_k) \in \mathrm{HF}$. ■

5.4.13. Theorem. *Every term of λ_T is hereditarily finite.*

Proof. By Lemma 5.4.11 and Lemma 5.4.12, the hereditarily finite terms are closed under application and lambda abstraction, so it suffices to show that the constants and the variables are hereditarily finite. Variables and the constant $\mathsf{0}$ are obviously hereditarily finite. Regarding S^+, let $\mathcal{X} \in \mathcal{HF}_{\mathsf{N}}$, then $\mathsf{S}^+\mathcal{X} \subseteq \Lambda^{\emptyset}_B(\mathsf{N}) \cap \mathrm{SN}$ and $nf(\mathsf{S}^+\mathcal{X})$ is finite since $nf(\mathcal{X})$ is finite. Hence $\mathsf{S}^+\mathcal{X} \in \mathcal{HF}_{\mathsf{N}}$, so S^+ is hereditarily finite. It remains to prove that the constants R_A are hereditarily finite. Let $\mathcal{M}, \mathcal{N}, \mathcal{X} \in \mathcal{HF}$ be of appropriate types and consider $\mathsf{R}_A\mathcal{MNX}$. We have in particular $\mathcal{X} \in \mathcal{HF}_{\mathsf{N}}$, so $nf(\mathcal{X})$ is finite, and the proof of $\mathsf{R}_A\mathcal{MNX} \in \mathcal{HF}$ goes by induction on the largest numeral in $nf(\mathcal{X})$. If $nf(\mathcal{X}) = \{\mathsf{0}\}$, then $\mathsf{R}_A\mathcal{MNX}$ is hereditarily adfluent with $\mathcal{M}$. Since $\mathcal{M} \in \mathcal{HF}$ we can apply Lemma 5.4.5 to obtain $\mathsf{R}_A\mathcal{MNX} \in \mathcal{HF}$. For the induction step, assume $\mathsf{R}_A\mathcal{MNX}' \in \mathcal{HF}$ for all $\mathcal{X}' \in \mathcal{HF}$ such that the largest numeral in $nf(\mathcal{X}')$ is n. Let, for some $\mathcal{X} \in \mathcal{HF}$, the largest numeral in $nf(\mathcal{X})$ be $\mathsf{S}^+\mathsf{n}$. Define

$$\mathcal{X}' = \{X \mid \mathsf{S}^+X \text{ is a reduct of a term in } \mathcal{X}\}$$

Then $\mathcal{X}' \in \mathcal{HF}$ since $\mathcal{X} \in \mathcal{HF}$, and the largest numeral in $nf(\mathcal{X}')$ is n. It follows by the induction hypothesis that $\mathsf{R}_A\mathcal{MNX}' \in \mathcal{HF}$, so $\mathcal{N}(\mathsf{R}_A\mathcal{MNX}')\mathcal{X}' \in \mathcal{HF}$ and hence $\mathcal{N}(\mathsf{R}_A\mathcal{MNX}')\mathcal{X}' \cup \mathcal{M} \in \mathcal{HF}$ by Lemma 5.4.11,5.4.9. We have that $\mathsf{R}_A\mathcal{MNX}$, is hereditarily adfluent with $\mathcal{N}(\mathsf{R}_A\mathcal{MNX}')\mathcal{X}' \cup \mathcal{M}$ so $\mathsf{R}_A\mathcal{MNX} \in \mathcal{HF}$ by Lemma 5.4.5. This completes the induction step. ■

Before we can prove that B is hereditarily finite we need the following lemma.

5.4.14. Lemma. *Let $\mathcal{Y}, \mathcal{G}, \mathcal{H}, \mathcal{X}, \mathcal{C} \in \mathcal{HF}$ be of appropriate type. Then*

$$\mathsf{B}\mathcal{Y}\mathcal{G}\mathcal{H}\mathcal{X}\mathcal{C} \in \mathcal{HF},$$

whenever $\mathsf{B}\mathcal{Y}\mathcal{G}\mathcal{H}(\mathsf{S}^+\mathcal{X})(\mathcal{C} *_{\mathcal{X}} \mathcal{A}) \in \mathcal{HF}$ for all $\mathcal{A} \in \mathcal{HF}$ of appropriate type.

Proof. Let conditions be as above. Abbreviate $\mathsf{B}\mathcal{Y}\mathcal{G}\mathcal{H}$ by $\mathcal{B}$ and $\mathsf{B}^c\mathcal{Y}\mathcal{G}\mathcal{H}$ by $\mathcal{B}^c$. Assume $\mathcal{B}(\mathsf{S}^+\mathcal{X})(\mathcal{C} *_{\mathcal{X}} \mathcal{A}) \in \mathcal{HF}$ for all $\mathcal{A} \in \mathcal{HF}$. Below we will frequently and implicitly use that $\dot{-}, *, [\,]$ are primitive recursive and hence hereditarily finite, and that hereditary finiteness is closed under application. Since hereditarily finite terms are strongly normalizable, we have that $\mathcal{B}\mathcal{X}\mathcal{C}$ is hereditarily adfluent with $\mathcal{B}^c\mathcal{X}\mathcal{C}(\mathcal{X} \dot{-} \mathcal{Y}[\mathcal{C}]_{\mathcal{X}})$, and hence with $\mathcal{G}\mathcal{C}\mathcal{X} \cup \mathcal{H}(\lambda a.\mathcal{B}(\mathsf{S}^+\mathcal{X})(\mathcal{C} *_{\mathcal{X}} a))\mathcal{C}\mathcal{X}$. It suffices to show that the latter set is in $\mathcal{HF}$. We have $\mathcal{G}\mathcal{C}\mathcal{X} \in \mathcal{HF}$, so by Lemma 5.4.9 the union is hereditarily finite if $\mathcal{H}(\lambda a.\mathcal{B}(\mathsf{S}^+\mathcal{X})(\mathcal{C} *_{\mathcal{X}} a))\mathcal{C}\mathcal{X}$ is. It suffices that $\lambda a.\mathcal{B}(\mathsf{S}^+\mathcal{X})(\mathcal{C} *_{\mathcal{X}} a) \in \mathcal{HF}$, and this will follow by the assumption above. We first observe that $\{\mathsf{0}^A\} \in \mathcal{HF}$ so $\mathcal{B}(\mathsf{S}^+\mathcal{X})(\mathcal{C} *_{\mathcal{X}} \{\mathsf{0}^A\}) \in \mathcal{HF}$ and hence $\mathcal{B}(\mathsf{S}^+\mathcal{X})(\mathcal{C} *_{\mathcal{X}} a) \subseteq \mathrm{SN}$ by Lemma 5.4.6. Let $\mathcal{A} \in \mathcal{HF}$. Since $\mathcal{B}(\mathsf{S}^+\mathcal{X})(\mathcal{C} *_{\mathcal{X}} a), \mathcal{A} \subseteq \mathrm{SN}$ we have that $(\lambda a.\mathcal{B}(\mathsf{S}^+\mathcal{X})(\mathcal{C} *_{\mathcal{X}} a))\mathcal{A}$ is adfluent with $\mathcal{B}(\mathsf{S}^+\mathcal{X})(\mathcal{C} *_{\mathcal{X}} \mathcal{A}) \in \mathcal{HF}$ and hence hereditarily finite itself by Lemma 5.4.5. ■

We now have arrived at the crucial step, where not only the language of analysis will be used, but also the axiom of dependent choice in combination with classical logic. We will reason by contradiction. Suppose B is not hereditarily finite. Then there are hereditarily finite $\mathcal{Y}, \mathcal{G}, \mathcal{H}, \mathcal{X}$ and $\mathcal{C}$ such that $\mathsf{B}\mathcal{Y}\mathcal{G}\mathcal{H}\mathcal{X}\mathcal{C}$ is not hereditarily finite. We introduce the following abbreviations: $\mathcal{B}$ for $B\mathcal{Y}\mathcal{G}\mathcal{H}$ and $\mathcal{X}{+}\mathsf{n}$ for $\mathsf{S}^+(\ldots(\mathsf{S}^+\mathcal{X})\ldots)$ (n times S^+). By Lemma 5.4.14, there exists $\mathcal{U} \in \mathcal{HF}$ such that $\mathcal{B}(\mathcal{X}{+}1)(\mathcal{C} *_{\mathcal{X}} \mathcal{U})$ is not hereditarily finite. Hence again by Lemma 5.4.14, there exists $\mathcal{V} \in \mathcal{HF}$ such that $\mathcal{B}(\mathcal{X}{+}2)((\mathcal{C} *_{\mathcal{X}} \mathcal{U}) *_{\mathcal{X}+1} \mathcal{V})$ is not hereditarily finite. Using dependent choice, let

$$\mathcal{D} = \mathcal{C} \cup (\mathcal{C} *_{\mathcal{X}} \mathcal{U}) \cup ((\mathcal{C} *_{\mathcal{X}} \mathcal{U}) *_{\mathcal{X}+1} \mathcal{V}) \cup \ \ldots$$

be the infinite union of the sets obtained by iterating the argument above. Note that all sets in the infinite union are hereditarily finite of type A^{N}. Since the union is infinite, it does not follow from Lemma 5.4.9 that $\mathcal{D}$ itself is hereditarily finite. However, since $\mathcal{D}$ has been built up from terms of type A^{N} having longer and longer initial segments in common we will nevertheless be able to prove that $\mathcal{D} \in \mathcal{HF}$. Then we will arrive at a contradiction, since $\mathcal{Y}\mathcal{D} \in \mathcal{HF}$ implies that $\mathcal{Y}$ is bounded on $\mathcal{D}$, so that the bar condition is satisfied after finitely many steps, which conflicts with the construction process.

5.4.15. Lemma. *The set $\mathcal{D}$ constructed above is hereditarily finite.*

Proof. Let $\mathcal{N}, \vec{\mathcal{Z}} \in \mathcal{HF}$ be of appropriate type, that is, $\mathcal{N}$ of type N and $\vec{\mathcal{Z}}$ such that $\mathcal{D}\mathcal{N}\vec{\mathcal{Z}}$ is of type N. We have to show $\mathcal{D}\mathcal{N}\vec{\mathcal{Z}} \in \mathcal{HF}$. Since all elements

of $\mathcal{D}$ are hereditarily finite we have $\mathcal{D}\mathcal{N}\vec{\mathcal{Z}} \subseteq \mathrm{SN}$. By an easy generalization of Theorem 5.3.12 we have WCR for λ_B, so by Newman's Lemma 5.3.14 we have $\mathcal{D}\mathcal{N}\vec{\mathcal{Z}} \subseteq \mathrm{CR}$. Since $\mathcal{N} \in \mathcal{HF}$ it follows that $nf(\mathcal{N})$ is finite, say $nf(\mathcal{N}) \subseteq \{\mathsf{0}, \ldots, \mathsf{n}\}$ for n large enough. It remains to show that $nf(\mathcal{D}\mathcal{N}\vec{\mathcal{Z}})$ is finite. Since all terms in $\mathcal{D}\mathcal{N}\vec{\mathcal{Z}}$ are CR, their normal forms are unique. As a consequence we may apply a leftmost innermost reduction strategy to any term $DN\vec{Z} \in \mathcal{D}\mathcal{N}\vec{\mathcal{Z}}$. At this point it might be helpful to remind the reader of the intended meaning of $*$: $C *_x A$ represents the finite sequence $C0, \ldots, C(x-1), A$. More formally,

$$(C *_x A)y = \begin{cases} C(y) & \text{if } y < x, \\ A & \text{otherwise.} \end{cases}$$

With this in mind it is easily seen that $nf(\mathcal{D}\mathcal{N}\vec{\mathcal{Z}})$ is a subset of $nf(\mathcal{D}_{\mathsf{n}}\mathcal{N}\vec{\mathcal{Z}})$, with

$$\mathcal{D}_{\mathsf{n}} = \mathcal{C} \cup (\mathcal{C} *_{\mathcal{X}} \mathcal{U}) \cup ((\mathcal{C} *_{\mathcal{X}} \mathcal{U}) *_{\mathcal{X}+1} \mathcal{V}) \cup \ldots \cup (\cdots (\mathcal{C} *_{\mathcal{X}} \mathcal{U}) * \cdots *_{\mathcal{X}+\mathsf{n}} \mathcal{W})$$

a finite initial part of the infinite union $\mathcal{D}$. The set $nf(\mathcal{D}_{\mathsf{n}}\mathcal{N}\vec{\mathcal{Z}})$ is finite since the union is finite and all sets involved are in $\mathcal{HF}$. Hence $\mathcal{D}$ is hereditarily finite by Lemma 5.4.3. ■

Since $\mathcal{D}$ is hereditarily finite, it follows that $nf(\mathcal{Y}\mathcal{D})$ is finite. Let k be larger than any numeral in $nf(\mathcal{Y}\mathcal{D})$. Consider

$$\mathcal{B}_{\mathsf{k}} = \mathcal{B}(\mathcal{X}{+}\mathsf{k})(\cdots(\mathcal{C} *_{\mathcal{X}} \mathcal{U}) * \cdots *_{\mathcal{X}+\mathsf{k}} \mathcal{W}')$$

as obtained in the construction above, iterating Lemma 5.4.14, hence not hereditarily finite. Since k is a strict upper bound of $nf(\mathcal{Y}\mathcal{D})$ it follows that the set $nf((\mathcal{X}{+}\mathsf{k}) \dot{-} \mathcal{Y}\mathcal{D})$ consists of numerals greater than $\mathsf{0}$, so that $\mathcal{B}_{\mathsf{k}}$ is hereditarily adfluent with $\mathcal{G}(\mathcal{X}{+}\mathsf{k})\mathcal{D}$. The latter set is hereditarily finite since it is an application of hereditarily finite sets (use Lemma 5.4.15). Hence $\mathcal{B}_{\mathsf{k}}$ is hereditarily finite by Lemma 5.4.5, which yields a plain contradiction.

By this contradiction, B must be hereditarily finite, and so is B^c, which follows by inspection of the reduction rules. As a consequence we obtain the main theorem of this section.

5.4.16. Theorem. *Every bar recursive term is hereditarily finite.*

5.4.17. Corollary. *Every bar recursive term is strongly normalizable.*

5.4.18. Remark. The first normalization result for bar recursion is due to Tait [1971], who proves WN for λ_B. Vogel [1976] strengthens Tait's result to SN, essentially by introducing B^c and by enforcing every B-redex to reduce via B^c. Both Tait and Vogel use infinite terms. The proof above is based on Bezem [1985] and avoids infinite terms by using the notion of hereditary finiteness, which is a syntactic version of Howard's compactness of functionals of finite type, see Troelstra [1973], Section 2.8.6.

If one considers λ_B also with $\boldsymbol{\eta}$-reduction, then the above results can also be obtained in a similar way as for λ_T with $\boldsymbol{\eta}$-reduction.

Semantics of λ_B

In this section we give some interpretations of bar recursion.

5.4.19. Definition. A model of λ_B is a model of λ_T with interpretations of the constants $\mathsf{B}_{A,B}$ and $\mathsf{B}^c_{A,B}$ for all A, B, such that the rules for these constants can be interpreted as valid equations. In particular we have then that the schema of bar recursion is valid, with $[\![\varphi]\!] = [\![\mathsf{B}YGH]\!]$.

We have seen at the beginning of this section that the full set theoretic model of Gödel's $\mathcal{T}$ is not a model of bar recursion, due to the existence of functionals (such as Y unbounded on binary functions) for which the bar recursion is not well-founded. Designing a model of λ_B amounts to ruling out such functionals, while maintaining the necessary closure properties. There are various solutions to this problem. The simplest solution is to take the closed terms modulo convertibility, which form a model by CR and SN. However, interpreting terms (almost) by themselves does not explain very much. In Exercise 5.4.24 the reader is asked to prove that the closed term model is extensional. An interesting model is obtained by using continuity in the form of the Kleene [1959a] and Kreisel [1959] continuous functionals. Continuity is on one hand a structural property of bar recursive terms, since they can use only finitely many information about their arguments, and on the other hand ensures that bar recursion is well-founded, since a continuous Y becomes eventually constant on increasing initial segments $[C]_x$. In Exercise 5.4.23 the reader is asked to elaborate this model in detail. Refinements can be obtained by considering notions of computability on the continuous functionals, such as Kleene's [1959b] S1-S9 recursive functionals. Computability alone, without uniform continuity on all binary functions, does not yield a model of bar recursion, see Exercise 5.4.22. The model of bar recursion we will elaborate in the next paragraphs is based on the same idea as the proof of strong normalization in the previous section. Here we consider the notion of hereditary finiteness semantically instead of syntactically. The intuition is that the set of increasing initial segments is hereditarily finite, so that any hereditarily finite functional Y is bounded on that set, and hence the bar recursion is well-founded.

5.4.20. Definition (hereditarily finite functionals). Recall the full type structure over the natural numbers: $\mathbb{N}_{\mathsf{N}} = \mathbb{N}$ and $\mathbb{N}_{A\to B} = \mathbb{N}_A \to \mathbb{N}_B$. A set $\mathcal{X} \subseteq \mathbb{N}_{\mathsf{N}}$ is hereditarily finite if $\mathcal{X}$ is finite. A set $\mathcal{X} \subseteq \mathbb{N}_{A\to B}$ is hereditarily finite if $\mathcal{X}\mathcal{Y} \subseteq \mathbb{N}_B$ is hereditarily finite for every hereditarily finite $\mathcal{Y} \subseteq \mathbb{N}_A$. Here and below, $\mathcal{X}\mathcal{Y}$ denotes the set of all results that can be obtained by applying functionals from $\mathcal{X}$ to functionals from $\mathcal{Y}$. A functional F is hereditarily finite if the singleton set $\{F\}$ is hereditarily finite. Let $\mathcal{HF}$ be the substructure of the full type structure consisting of all hereditarily finite functionals.

The proof that $\mathcal{HF}$ is a model of λ_B has much in common with the proof that λ_B is SN from the previous paragraph. The essential step is that the interpretation of the bar recursor is hereditarily finite. This requires the following semantic version of Lemma 5.4.14:

5.4.21. LEMMA. *Let $\mathcal{Y}, \mathcal{G}, \mathcal{H}, \mathcal{X}, \mathcal{C}$ be hereditarily finite sets of appropriate type. Then $[\![\mathsf{B}]\!]\mathcal{Y}\mathcal{G}\mathcal{H}\mathcal{X}\mathcal{C}$ is well defined and hereditarily finite whenever $[\![\mathsf{B}]\!]\mathcal{Y}\mathcal{G}\mathcal{H}(\mathcal{X}+1)(\mathcal{C}*_{\mathcal{X}}\mathcal{A})$ is so for all hereditarily finite $\mathcal{A}$ of appropriate type.*

The proof proceeds by iterating this lemma in the same way as how the SN proof proceeds after Lemma 5.4.14. The set of longer and longer initial sequences with elements taken from hereditarily finite sets (cf. the set $\mathcal{D}$ in Lemma 5.4.15) is hereditarily finite itself. As a consequence, the bar recursion must be well-founded when the set $\mathcal{Y}$ is also hereditarily finite. It follows that the interpretation of the bar recursor is well-defined and hereditarily finite.

5.4.22. EXERCISE. This exercise shows that *HEO* is *not* a model for bar recursion. Recall that $*$ stands for partial recursive function application. Consider functionals Y, G, H defined by $G(x, C) = 0$, $H(Z, x, C) = 1 + Z(0) + Z(1)$ and $Y(F)$ is the smallest number n such that $i * i$ converges in less than n steps for some $i < n$ and, moreover, $i * i = 0$ if and only if $F(i) = 0$ does *not* hold. The crux of the definition of Y is that no *total* recursive function F can distinguish between $i * i = 0$ and $i * i > 0$ for *all* i with $i * i\!\downarrow$. But for any finite number of such i's we do have a total recursive function making the correct distinctions. This implies that Y, although continuous and well-defined on all total recursive functions, is not uniformly continuous and not bounded on total recursive binary functions. Show that all functionals involved can be represented in *HEO* and that the latter model of λ_T is not a model of λ_B.

5.4.23. EXERCISE. This exercise introduces the continuous functionals, Kleene [1959a]. Define for $f, g \in \mathbb{N}\to\mathbb{N}$ the (partial) application of f to g by $f(g) = f(\overline{g}\,n) - 1$, where n is the smallest number such that $f(\overline{g}\,n) > 0$, provided there is such n. If there is no such n, then $f * g$ is undefined. The idea is that f uses only finitely many information about g for determining the value of $f * g$ (if any). Define inductively for every type A a set $\mathcal{C}_A$ together with an association relation between elements of of $\mathbb{N}\to\mathbb{N}$ and elements of $\mathcal{C}_A$. For the base type we put $\mathcal{C}_{\mathsf{N}} = \mathbb{N}$ and let the constant functions be the associates of the corresponding natural numbers. For higher types we define that $f \in \mathbb{N}\to\mathbb{N}$ is an associate of $F \in \mathcal{C}_A\to\mathcal{C}_B$ if for any associate g of $G \in \mathcal{C}_A$ the function h defined by $h(n) = f(n{:}g)$ is an associate of $F(G) \in \mathcal{C}_B$. Here $n{:}g$ is shorthand for the function taking value n at 0 and value $g(k-1)$ for all $k > 0$. (Note that we have implicitly required that h is total.) Now $\mathcal{C}_{A\to B}$ is defined as the subset of those $F \in \mathcal{C}_A\to\mathcal{C}_B$ that have an associate. Show that $\mathcal{C}$ is a model for bar recursion.

Following Troelstra [1973], Section 2.4.5 and 2.7.2, we define the following notion of hereditary extensional equality. We put $\approx_{\mathsf{N}}$ to be $=$, convertibility of closed terms in $\Lambda^{\emptyset}_B(\mathsf{N})$. For the type $A \equiv B\to B'$ we define $M \approx_A M'$ if and only if $M, M' \in \Lambda^{\emptyset}_B(A)$ and $MN \approx_{B'} M'N'$ for all N, N' such that $N \approx_B N'$. By (simultaneous) induction on A one shows easily that $\approx_A$ is symmetric, transitive and partially reflexive, that is, $M \approx_A M$ holds whenever $M \approx_A N$ for some N.

The corresponding axiom of hereditary extensionality is simply stating that $\approx_A$ is (totally) reflexive: $M \approx_A M$, schematic in $M \in \Lambda^{\emptyset}_B(A)$ and A. It follows from the next exercise that the closed term model is extensional.

5.4.24. Exercise. Show $M \approx M$ for any closed term $M \in \Lambda_B^{\emptyset}$. Hint: define a predicate $\mathrm{Ext}(M(x_1, \ldots, x_n))$ for any open term M with free variables among $x_1, \ldots, x_n$ by

$$M(X_1, \ldots, X_n) \approx M(X'_1, \ldots, X'_n)$$

for all $X_1, \ldots, X_n, X'_1, \ldots, X'_n \in \Lambda_B^{\emptyset}$ with $X_1 \approx X'_1, \ldots, X_n \approx X'_n$. Then prove by induction on terms that Ext holds for any open term, so in particular for closed terms. For B, prove first the lemma below.

5.4.25. Lemma. *For all $Y \approx Y', G \approx G', H \approx H', X \approx X', C \approx C'$, if*

$$\mathsf{B}YGH(\mathsf{S}^+X)(C *_X A) \approx \mathsf{B}Y'G'H'(\mathsf{S}^+X')(C' *_{X'} A')$$

for all $A \approx A'$, then $\mathsf{B}YGHXC \approx \mathsf{B}Y'G'H'X'C'$.

5.5. Platek's system $\mathcal{Y}$: fixed point recursion

Platek [1966] introduces a simply typed lambda calculus extended with fixed point combinators. Here we study Platek's system as an extension of Gödel's $\mathcal{T}$. An almost identical system is called PCF in Plotkin [1977].

A *fixed point combinator* is a functional Y of type $(A \to A) \to A$ such that YF is a fixed point of F, that is, $YF = F(YF)$, for every F of type $A \to A$. Fixed point combinators can be used to compute solutions to recursion equations. The only difference with the type-free lambda calculus is that here all terms are typed, including the fixed point combinators themselves.

As an example we consider the recursion equations of the schema of higher order primitive recursion in Gödel's system $\mathcal{T}$, Section 5.3. We can rephrase these equations as

$$RMNn = \mathrm{If}\ n\ (N(RMN(n-1))(n-1))M,$$

where $\mathrm{If}\ nM_1M_0 = M_0$ if $n = 0$ and M_1 if $n > 0$. Hence we can write

$$\begin{aligned} RMN &= \lambda n.\ \mathrm{If}\ n\ (N(RMN(n-1))(n-1))M \\ &= (\lambda fn.\ \mathrm{If}\ n\ (N(f(n-1))(n-1))M)(RMN) \end{aligned}$$

This equation is of the form $YF = F(YF)$ with

$$F = \lambda fn.\ \mathrm{If}\ n\ (N(f(n-1))(n-1))M$$

and $YF = RMN$. It is easy to see that YF satisfies the recursion equation for RMN *uniformly* in M, N. This show that, given functionals If and a predecessor function (to compute $n-1$ in case $n > 0$), higher-order primitive recursion is definable by fixed point recursion. However, for computing purposes it is convenient to have primitive recursors at hand. By a similar argument, one can show bar recursion to be definable by fixed point recursion.

In addition to the above argument we show that every partial recursive function can be defined by fixed point recursion, by giving a fixed point recursion for minimization. Let F be a given function. Define by fixed point recursion

$G_F = \lambda n.$If $F(n)$ $G_F(n+1)$ n. Then we have $G_F(0) = 0$ if $F(0) = 0$, and $G_F(0) = G_F(1)$ otherwise. We have $G_F(1) = 1$ if $F(1) = 0$, and $G_F(1) = G_F(2)$ otherwise. By continuing this argument we see that

$$G_F(0) = \min\{n \mid F(n) = 0\},$$

that is, $G_F(0)$ computes the smallest n such that $F(n) = 0$, provided that such n exists. If there exists no n such that $F(n) = 0$, then $G_F(0)$ as well as $G_F(1), G_F(2), \ldots$ are undefined. Given a function F of *two* arguments, minimization with respect to the second argument can now be obtained by the partial function $\lambda x.G_{F(x)}(0)$.

In the paragraph above we saw already that fixed point recursions may be indefinite: if F does not zero, then $G_F(0) = G_F(1) = G_F(2) = \ldots$ does not lead to a definite value, although one could consistently assume G_F to be a constant function in this case. However, the situation is in general even worse: there is no natural number n that can consistently be assumed to be the fixed point of the successor function, that is, $n = Y(\lambda x.x + 1)$, since we cannot have $n = (\lambda x.x + 1)n = n + 1$. This is the price to be paid for a formalism that allows one to compute all partial recursive functions.

Syntax of λ_Y

In this section we formalize Platek's $\mathcal{Y}$ as an extension of Gödel's $\mathcal{T}$ called λ_Y.

5.5.1. Definition. The *types* of λ_Y are the types of λ_T. The *terms* of λ_Y are obtained by adding constants

$$\mathsf{Y}_A \quad : \quad (A{\to}A){\to}A$$

for all types A to the constants of λ_T. The set of (closed) terms of λ_Y (of type A) is denoted with $\Lambda^{\emptyset}_Y(A)$. The *formulas* of λ_Y are equations between terms of λ_Y (of the same type). The *theory* of λ_Y extends the theory of λ_T with the schema $\mathsf{Y}F = F(\mathsf{Y}F)$ for all appropriate types. The *reduction relation* $\to_Y$ of λ_Y extends $\to_T$ by adding the following rule for the constants Y (omitting type annotations A):

$$\mathsf{Y} \quad \to_Y \quad \lambda f.f(\mathsf{Y}f)$$

The reduction rule for Y requires some explanation, as the rule $\mathsf{Y}F \to F(\mathsf{Y}F)$ seems simpler. However, with the latter rule we would have diverging reductions $\lambda f.\mathsf{Y}f \to_{\boldsymbol{\eta}} \mathsf{Y}$ and $\lambda f.\mathsf{Y}f \to_Y \lambda f.f(\mathsf{Y}f)$ that cannot be made to converge, so that we would lose CR of $\to_Y$ in combination with $\boldsymbol{\eta}$-reduction.

The Church-Rosser property for λ_Y with β-reduction and with $\beta\boldsymbol{\eta}$-reduction can be proved by standard techniques from higher-order rewriting theory, for example, by using weak orthogonality, see Raamsdonk [1996].

5.5.2. Exercise. It is possible to define λ_Y as extension of $\lambda^o_{\to}$ using Church numerals $c_n \equiv \lambda x^{\mathsf{N}}\, f^{\mathsf{N}\to\mathsf{N}}.f^n x$. Show that every partial recursive function is also definable in this version of λ_Y.

Although λ_Y has universal computational strength in the sense that all partial recursive functions can be computed, not every computational phenomenon can be represented. In the next section we shall see that λ_Y does not have enumerators. Moreover, λ_Y is inherently sequential. For example, λ_Y doesn't have a term P such that $PMN = 0$ if and only if $M = 0$ or $N = 0$. The problem is that M and N cannot be evaluated in parallel, and if the argument that is evaluated first happens to be undefined, then the outcome is undefined even if the other argument equals 0. For a detailed account of the so-called sequentiality of λ_Y, see Plotkin [1977].

Semantics of λ_Y

In this section we explore the semantics of λ_Y and give one model.

5.5.3. Definition. A model of λ_Y is a model of λ_T with interpretations of the constants Y_A for all A, such that the rules for these constants can be interpreted as valid equations.

Models of λ_Y differ from those of λ_T, λ_B in that they have to deal with partiality. As we saw in the introduction of this section, no natural number n can consistently be assumed to be the fixed point of the successor function. Nevertheless, we want to interpret terms like $\mathsf{Y}\mathsf{S}^+$. The canonical way to do so is to add an element $\perp$ to the natural numbers, representing undefined objects like the fixed point of the successor function. Let $\mathbb{N}^\perp$ denote the set of natural numbers extended with $\perp$. Now higher types are interpreted as function spaces over $\mathbb{N}^\perp$. The basic intuition is that $\perp$ contains less information than any natural number, and that functions and functionals give more informative output when the input becomes more informative. One way of formalizing these intuitions is by using partial orderings. We equip $\mathbb{N}^\perp$ with the partial ordering $\sqsubseteq$ such that $\perp \sqsubset n$ for all $n \in \mathbb{N}$. In order to be able to interpret Y, every function must have a fixed point. This requires some extra structure on the partial orderings, which can be formalized by the notion of complete partial ordering (cpo). The next lines bear some similarity to the introductory treatment of ordinals in Section 5.3. We call a set *directed* if it contains an upper bound for every two elements of it. Completeness of a partial ordering means that every directed set has a supremum. A function on cpo's is called *continuous* if it preserves suprema of directed sets. Every continuous function f of cpo's is monotone and has a least fixed point $\mathrm{lfp}(f)$, being the supremum of the directed set enumerated by iterating f starting at $\perp$. The function lfp is itself continuous and serves as the interpretation of Y. We are now ready for the following definition.

5.5.4. Definition. Define by induction $\mathbb{N}^\perp_{\mathsf{N}} = \mathbb{N}^\perp$ and $\mathbb{N}^\perp_{A\to B}$ is the set of all continuous mappings $\mathbb{N}^\perp_A \to \mathbb{N}^\perp_B$.

Given the fact that cpo's with continuous mappings form a cartesian closed category and that the successor, predecessor and conditional can be defined in a continuous way, the only essential step in the proof of the following lemma is to put $[\![\mathsf{Y}]\!] = \mathrm{lfp}$ for all appropriate types.

5.5.5. LEMMA. *The type structure of cpo's $\mathbb{N}_A^\perp$ is a model for λ_Y.*

In fact, as the essential requirement is the existence of fixed points, we could have taken monotone instead of continuous mappings on cpo's. This option is elaborated in detail in van Draanen [1995].

5.6. Exercises

5.6.1. Prove proposition 5.2.9: for all types A one has $A \lhd_{SP} N_{rank(A)}$.

5.6.2. Let λ_{P} be $\lambda^o_\to$ extended with a simple (not surjective) pairing. Show that theorem 5.2.42 does not hold for this theory. [Hint show that in this theory the equation $\lambda x\, o.\langle \pi_1 x, \pi_2 x\rangle = \lambda x\, o.x$ does not hold by constructing a counter model, but is nevertheless consistent.]

5.6.3. Does every model of λ_{SP} have the same first order theory?

5.6.4. (i) Show that if a pairing function $\langle\ ,\ \rangle : o{\to}(o{\to}o)$ and projections $L, R : o{\to}o$ satisfying $L\langle x,y\rangle = x$ and $R\langle x,y\rangle = y$ are added to $\lambda^o_\to$, then for a non-trivial model $\mathcal{M}$ one has (see 4.2)

$$\forall A \in \mathbb{TT} \forall M, N \in \Lambda^\emptyset(A)[\mathcal{M} \models M = N \ \Rightarrow\ M{=}_{\beta\boldsymbol{\eta}} N].$$

(ii) (Schwichtenberg and Berger [1991]) Show that for $\mathcal{M}$ a model of λ_T one has (see 4.3)

$$\forall A \in \mathbb{TT} \forall M, N \in \Lambda^\emptyset(A)[\mathcal{M} \models M = N \ \Rightarrow\ M{=}_{\beta\boldsymbol{\eta}} N].$$

5.6.5. Show that $\mathcal{F}[x_1, \ldots, x_n]$ for $n \geq 0$ does not have one generator. [Hint. Otherwise this monoid would be commutative, which is not the case.]

5.6.6. Show that $R \subseteq \Lambda^\emptyset(A) \times \Lambda^\emptyset(B)$ is equational iff

$$\exists M, N \in \Lambda^\emptyset(A{\to}B{\to}1{\to}1)\, \forall F[R(F) \iff MF = NF].$$

5.6.7. Show that there is a Diophantine equation $\mathtt{lt} \subseteq \mathcal{F}^2$ such that for all $n, m \in \mathbb{N}$

$$\mathtt{lt}(R^n, R^m) \iff n < m.$$

5.6.8. Define $\mathtt{Seq}_n^{\mathcal{N}_k}(h)$ iff $h = [R^{m_0}, \ldots, R^{m_{n-1}}]$, for some $m_0, \ldots, m_{n-1} < k$. Show that $\mathtt{Seq}_n^{\mathcal{N}_k}$ is Diophantine uniformly in n.

5.6.9. Let $\mathcal{B}$ be some finite subset of $\mathcal{F}$. Define $\mathtt{Seq}_n^{\mathcal{B}}(h)$ iff $h = [g_0, \ldots, g_{n-1}]$, with each $g_i \in \mathcal{B}$. Show that $\mathtt{Seq}_n^{\mathcal{B}}$ is Diophantine uniformly in n.

5.6.10. For $\mathcal{B} \subseteq \mathcal{F}$ define $\mathcal{B}^+$ to be the submonoid generated by $\mathcal{B}$. Show that if $\mathcal{B}$ is finite, then $\mathcal{B}^+$ is Diophantine.

5.6.11. Show that $\mathcal{F} \subseteq \mathcal{F}[x]$ is Diophantine.

5.6.12. Construct two concrete terms $t(a,b), s(a,b) \in \mathcal{F}[a,b]$ such that for all $f \in \mathcal{F}$ one has

$$f \in \{R^n \mid n \in \mathbb{N}\} \cup \{L\} \iff \exists g \in \mathcal{F}\, [t(f,g) = s(f,g)].$$

[Remark. It is not sufficient to notice that Diophantine sets are closed under union. But the solution is not hard and the terms are short.]

5.6.13. Let $2 = \{0, 1\}$ be the discrete topological space with two elements. Let Cantor space be $\mathbf{C} = 2^{\mathbb{N}}$ endowed with the product topology. Define $Z, O : \mathbf{C} \to \mathbf{C}$ 'shift operators' on Cantor space as follows.

$$\begin{array}{rcl} Z(f)(0) & = & 0; \\ Z(f)(n+1) & = & f(n); \\ O(f)(0) & = & 1; \\ O(f)(n+1) & = & f(n). \end{array}$$

Write $0f = Z(f)$ and $1f = O(f)$. If $\mathcal{X} \subseteq \mathbf{C} \to \mathbf{C}$ is a set of maps, let $\mathcal{X}^+$ be the closure of $\mathcal{X}$ under the rule

$$A_0, A_1 \in \mathcal{X} \;\Rightarrow\; A \in \mathcal{X},$$

where A is defined by

$$\begin{array}{l} A(0f) = A_0(f); \\ A(1f) = A_1(f). \end{array}$$

(i) Show that if $\mathcal{X}$ consists of continuous maps, then so does $\mathcal{X}^+$.

(ii) Show that $A \in \{Z, O\}^+$ iff

$$A(f) = g \;\Rightarrow\; \exists r, s \in \mathbb{N} \forall t > s . g(t) = f(t - s + r).$$

(iii) Define on $\{Z, O\}^+$ the folllowing.

$$\begin{array}{rcll} I & = & \lambda x \in \{Z, O\}^+ . z; & \\ L & = & Z; & \\ R & = & O; & \\ x * y & = & y \circ x; & \\ \langle x, y \rangle & = & x(f), & \text{if } f(0) = 0; \\ & = & y(f), & \text{if } f(0) = 1. \end{array}$$

Then $\langle \{Z, O\}^+, *, I, L, R, \langle -, - \rangle \rangle$ is a Cartesian monoid isomorphic to $\mathcal{F}$, via $\varphi : \mathcal{F} \to \{Z, O\}^+$.

(iv) The Thompson-Freyd-Heller group can be defined by

$$\{f \in \mathcal{I} \mid \varphi(f) \text{ preserves the lexicographical ordering on } \mathbf{C}\}.$$

Show that the B_n defined in definition 5.2.29 generate this group.

5.6.14. Prove proposition 5.2.9: for all types A one has $A \lhd_{SP} N_{rank(A)}$.

5.6.15. Let λ_{P} be $\lambda^o_{\to}$ extended with a simple (not surjective) pairing. Show that theorem 5.2.42 does not hold for this theory. [Hint show that in this theory the equation $\lambda x\, o . \langle \pi_1 x, \pi_2 x \rangle = \lambda x\, o . x$ does not hold by constructing a counter model, but is nevertheless consistent.]

5.6.16. Does every model of λ_{SP} have the same first order theory?

5.6.17. Show that $\mathcal{F}[x_1,\ldots,x_n]$ for $n \geq 0$ does not have one generator. [Hint. Otherwise this monoid would be commutative, which is not the case.]

5.6.18. Show that $R \subseteq \Lambda^{\emptyset}(A) \times \Lambda^{\emptyset}(B)$ is equational iff

$$\exists M, N \in \Lambda^{\emptyset}(A{\to}B{\to}1{\to}1)\, \forall F[R(F) \iff MF = NF].$$

5.6.19. Show that there is a Diophantine equation $\mathtt{lt} \subseteq \mathcal{F}^2$ such that for all $n, m \in \mathbb{N}$

$$\mathtt{lt}(R^n, R^m) \iff n < m.$$

5.6.20. Define $\mathtt{Seq}_n^{\mathcal{N}_k}(h)$ iff $h = [R^{m_0}, \ldots, R^{m_{n-1}}]$, for some $m_0, \ldots, m_{n-1} < k$. Show that $\mathtt{Seq}_n^{\mathcal{N}_k}$ is Diophantine uniformly in n.

5.6.21. Let $\mathcal{B}$ be some finite subset of $\mathcal{F}$. Define $\mathtt{Seq}_n^{\mathcal{B}}(h)$ iff $h = [g_0, \ldots, g_{n-1}]$, with each $g_i \in \mathcal{B}$. Show that $\mathtt{Seq}_n^{\mathcal{B}}$ is Diophantine uniformly in n.

5.6.22. For $\mathcal{B} \subseteq \mathcal{F}$ define $\mathcal{B}^+$ to be the submonoid generated by $\mathcal{B}$. Show that if $\mathcal{B}$ is finite, then $\mathcal{B}^+$ is Diophantine.

5.6.23. Show that $\mathcal{F} \subseteq \mathcal{F}[x]$ is Diophantine.

5.6.24. Construct two concrete terms $t(a,b), s(a,b) \in \mathcal{F}[a,b]$ such that for all $f \in \mathcal{F}$ one has

$$f \in \{R^n \mid n \in \mathbb{N}\} \cup \{L\} \iff \exists g \in \mathcal{F}\, [t(f,g) = s(f,g)].$$

[Remark. It is not sufficient to notice that Diophantine sets are closed under union. But the solution is not hard and the terms are short.]

5.6.25. Let $2 = \{0,1\}$ be the discrete topological space with two elements. Let Cantor space be $\mathbf{C} = 2^{\mathbb{N}}$ endowed with the product topology. Define $Z, O : \mathbf{C}{\to}\mathbf{C}$ 'shift operators' on Cantor space as follows.

$$\begin{aligned} Z(f)(0) &= 0; \\ Z(f)(n+1) &= f(n); \\ O(f)(0) &= 1; \\ O(f)(n+1) &= f(n). \end{aligned}$$

Write $0f = Z(f)$ and $1f = O(f)$. If $\mathcal{X} \subseteq \mathbf{C}{\to}\mathbf{C}$ is a set of maps, let $\mathcal{X}^+$ be the closure of $\mathcal{X}$ under the rule

$$A_0, A_1 \in \mathcal{X} \Rightarrow A \in \mathcal{X},$$

where A is defined by

$$\begin{aligned} A(0f) &= A_0(f); \\ A(1f) &= A_1(f). \end{aligned}$$

(i) Show that if $\mathcal{X}$ consists of continuous maps, then so does $\mathcal{X}^+$.
(ii) Show that $A \in \{Z, O\}^+$ iff

$$A(f) = g \Rightarrow \exists r, s \in \mathbb{N} \forall t > s.g(t) = f(t - s + r).$$

(iii) Define on $\{Z,O\}^+$ the folllowing.

$$\begin{array}{rcll} I & = & \lambda x \in \{Z,O\}^+.z; & \\ L & = & Z; & \\ R & = & O; & \\ x * y & = & y \circ x; & \\ \langle x,y\rangle & = & x(f), & \text{if } f(0)=0; \\ & = & y(f), & \text{if } f(0)=1. \end{array}$$

Then $\langle \{Z,O\}^+, *, I, L, R, \langle -,-\rangle\rangle$ is a Cartesian monoid isomorphic to $\mathcal{F}$, via $\varphi : \mathcal{F} \rightarrow \{Z,O\}^+$.

(iv) The Thompson-Freyd-Heller group can be defined by

$$\{f \in \mathcal{I} \mid \varphi(f) \text{ preserves the lexicographical ordering on } \mathbf{C}\}.$$

Show that the B_n defined in definition 5.2.29 generate this group.

Chapter 6

Applications

6.1. Functional programming

Lambda calculi are prototype programming languages. As is the case with imperative programming languages, where several examples are untyped (machine code, assembler, Basic) and several are typed (Algol-68, Pascal), systems of lambda calculi exist in untyped and typed versions. There are also other differences in the various lambda calculi. The lambda calculus introduced in Church [1936] is the untyped $\lambda\mathsf{I}$-calculus in which an abstraction $\lambda x.M$ is only allowed if x occurs among the free variables of M. Nowadays, "lambda calculus" refers to the $\lambda\mathsf{K}$-calculus developed under the influence of Curry, in which $\lambda x.M$ is allowed even if x does not occur in M. There are also typed versions of the lambda calculus. Of these, the most elementary are two versions of the simply typed lambda calculus $\lambda_{\rightarrow}$. One version is due to Curry [1934] and has implicit types. Simply typed lambda calculus with explicit types is introduced in Church [1940] (this system is inspired by the theory of types of Russell and Whitehead [1910–13] as simplified by Ramsey [1925]). In order to make a distinction between the two versions of simply typed lambda calculus, the version with explicit types is sometimes called the *Church* version and the one with implicit types the *Curry* version. The difference is that in the Church version one explicitly types a variable when it is bound after a lambda, whereas in the Curry version one does not. So for example in Church's version one has $\mathsf{I}_A = (\lambda x\ A.x) : A{\rightarrow}A$ and similarly $\mathsf{I}_{A\rightarrow B} : (A{\rightarrow}B){\rightarrow}(A{\rightarrow}B)$, while in Curry's system one has $\mathsf{I} = (\lambda x.x) : A{\rightarrow}A$ but also $\mathsf{I} : (A{\rightarrow}B){\rightarrow}(A{\rightarrow}B)$ for the same term I. See B[92] for more information about these and other typed lambda calculi. Particularly interesting are the second and higher order calculi $\lambda 2$ and $\lambda\omega$ introduced by Girard [1972] (under the names 'system F' and 'system $F\omega$') for applications to proof theory and the calculi with dependent types introduced by de Bruijn [1970] for proof verification.

Computing on data types

In this subsection we explain how it is possible to represent data types in a very direct manner in the various lambda calculi.

Lambda definability was introduced for functions on the set of natural num-

bers $\mathbb{N}$. In the resulting mathematical theory of computation (recursion theory) other domains of input or output have been treated as second class citizens by coding them as natural numbers. In more practical computer science, algorithms are also directly defined on other data types like trees or lists.

Instead of coding such data types as numbers one can treat them as first class citizens by coding them directly as lambda terms *while preserving their structure*. Indeed, lambda calculus is strong enough to do this, as was emphasized in Böhm [1963] and Böhm and Gross [1966]. As a result, a much more efficient representation of algorithms on these data types can be given, than when these types were represented via numbers. This methodology was perfected in two different ways in Böhm and Berarducci [1985] and Böhm et al. [1994] or Berarducci and Bohm [1993]. The first paper does the representation in a way that can be typed; the other papers in an essentially stronger way, but one that cannot be typed. We present the methods of these papers by treating labeled trees as an example.

Let the (inductive) data-type of labeled trees be defined by the following abstract syntax.

$$\begin{array}{rcl} \mathtt{tree} & = & \bullet \,|\, \mathtt{leaf\ nat} \,|\, \mathtt{tree} + \mathtt{tree} \\ \mathtt{nat} & = & 0 \,|\, \mathtt{succ\ nat} \end{array}$$

We see that a label can be either a bud ($\bullet$) or a leaf with a number written on it. A typical such tree is $(\mathtt{leaf}\ 3) + ((\mathtt{leaf}\ 5) + \bullet)$. This tree together with its mirror image look as follows.

6.2. Logic and proof-checking

The Curry-de Bruijn-Howard correspondence

One of the main applications of type theory is its connection with logic. For several logical systems L there is a type theorie $\lambda-$ and a map translating formulas A of L into types $[A]$ of $\lambda-$ such that

$$\vdash_L A \iff \Gamma_A \vdash_{\lambda-} M : [A], \text{ for some } M,$$

where Γ_A is some context 'explaining' A. The term M can be constructed canonically from a natural deduction proof D of A. So in fact one has

$$\vdash_L A, \text{ with proof } D \iff \Gamma_A \vdash_{\lambda-} [D] : [A], \tag{6.1}$$

where the map $[\ ]$ is extended to cover also derivations. For deductions from a set of assumptions one has

$$\Delta \vdash_L A, \text{ with proof } D \iff \Gamma_A, [\Delta] \vdash_{\lambda-} [D] : [A].$$

Curry did not observe the correspondence in this precise form. He noted that inhabited types in $\lambda_\rightarrow$, like $A \rightarrow A$ or $A \rightarrow B \rightarrow A$, all had the form of a tautology of (the implication fragment of) propositional logic.

Howard [1980] (the work was done in 1968 and written down in the unpublished but widely circulated Howard [1969]), inspired by the observation of Curry and by Tait [1963], gave the more precise interpretation (6.1). He coined the term *propositions-as-types* and *proofs-as-terms*.

On the other hand, de Bruijn independently of Curry and Howard developed type systems satisfying (6.1). The work was started also in 1968 and the first publication was de Bruijn [1970]; see also de Bruijn [1980]. The motivation of de Bruijn was his visionary view that machine proof checking one day will be feasible and important. The collection of systems he designed was called the AUTOMATH family, derived from AUTOmatic MATHematics verification. The type systems were such that the right hand side of (6.1) was efficiently verifyable by machine, so that one had machine verification of provability. Also de Bruijn and his students were engaged in developing, using and implementing these systems.

Initially the AUTOMATH project received little attention from mathematicians. They did not understand the technique and worse they did not see the need for machine verification of provability. Also the verification process was rather painful. After five 'monk' years of work, van Benthem Jutting [1977] came up with a machine verification of Landau [1900] fully rewritten in the terse 'machine code' of one of the AUTOMATH languages. Since then there have been developed second generation versions in the AUTOMATH family, like NUPRL [1979], COQ ([1989]) and LEGO ([1991]), in which considerable help from the computer environment is obtained for the formalization of proofs. With these systems a task of verifying Landau [1900] took something like five months. An important contribution to these second generation systems came from Scott and Martin-Löf, by adding inductive data-types to the systems in order to make formalizations more natural.[1] In Kahn [1995] methods are developed in order to translate proof objects automatically into natural language. It is hoped that in the near future new proofcheckers will emerge in which formalizing is not much more difficult than, say, writing an article in TeX.

Computer Mathematics

Modern systems for computer algebra (CA) are able to represent mathematical notions on a machine and compute with them. These objects can be integers, real or complex numbers, polynomials, integrals and the like. The computations are usually symbolic, but can also be numerical to a virtually arbitrary degree of precision. It is fair to say—as is sometimes done—that "a system for CA can represent $\sqrt{2}$ exactly". In spite of the fact that this number has an infinite decimal expansion, this is not a miracle. The number $\sqrt{2}$ is represented in a computer just as a symbol (as we do on paper or in our mind), and the

[1] For example, proving Gödel's incompleteness theorem is difficult for the following reason. The main step in the proof essentially consists of constructing a compiler from a universal programming language into arithmetic. For this one needs to describe strings over an alphabet in the structure of numbers with plus and times. This is difficult and Gödel used the Chinese remainder theorem to do this. Having available the datatype of strings, together with the corresponding operators, makes the translation much more natural.

machine knows how to manipulate it. The common feature of these kind of notions represented in systems for CA is that in some sense or another they are all computable. Systems for CA have reached a high level of sophistication and efficiency and are commercially available. Scientists and both pure and applied mathematicians have made good use of them for their research.

There is now emerging a new technology, namely that of systems for Computer Mathematics (CM). In these systems virtually all mathematical notions can be represented exactly, including those that do not have a computational nature. How is this possible? Suppose, for example, that we want to represent a non-computable object like the co-Diophantine set

$$X = \{n \in \mathbb{N} \mid \neg\exists \vec{x}\ D(\vec{x}, n) = 0\}.$$

Then we can do as before and represent it by a special symbol. But now the computer in general cannot operate on it because the object may be of a non-computational nature.

Before answering the question in the previous paragraph, let us first analyze where non-computability comes from. It is always the case that this comes from the quantifiers $\forall$ (for all) and $\exists$ (exists). Indeed, these quantifiers usually range over an infinite set and therefore one loses decidability.

Nevertheless, for ages mathematicians have been able to obtain interesting information about these non-computable objects. This is because there is a notion of *proof*. Using proofs one can state with confidence that e.g.

$$3 \in X, \text{ i.e., } \neg\exists \vec{x}\ D(\vec{x}, 3) = 0.$$

Aristotle had already remarked that it is often hard to find proofs, but the verification of a putative one can be done in a relatively easy way. Another contribution of Aristotle was his quest for the formalization of logic. After about 2300 years, when Frege had found the right formulation of predicate logic and Gödel had proved that it is complete, this quest was fulfilled. Mathematical proofs can now be completely formalized and verified by computers. This is the underlying basis for the systems for CM.

Present day prototypes of systems for CM are able to help a user to develop from primitive notions and axioms many theories, consisting of defined concepts, theorems and proofs.[2] All the systems of CM have been inspired by the AUTOMATH project of de Bruijn (see de Bruijn [1970] and[1990] and Nederpelt et al. [1994]) for the automated verification of mathematical proofs.

Representing proofs as lambda terms

Now that mathematical proofs can be fully formalized, the question arises how this can be done best (for efficiency reasons concerning the machine and pragmatic reasons concerning the human user). Hilbert represented a proof of statement A from a set of axioms Γ as a finite sequence $A_0, A_1 \ldots, A_n$ such that

[2]This way of doing mathematics, the axiomatic method, was also described by Aristotle. It was Euclid [n.d.] who first used this method very successfully in his Elements.

$A = A_n$ and each A_i, for $0 \leq i \leq n$, is either in Γ or follows from previous statements using the rules of logic.

A more efficient way to represent proofs employs typed lambda terms and is called the *propositions-as-types* interpretation discovered by Curry, Howard and de Bruijn. This interpretation maps propositions into types and proofs into the corresponding inhabitants. The method is as follows. A statement A is transformed into the type (i.e., collection)

$$[A] = \text{ the set of proofs of } A.$$

So A is provable if and only if $[A]$ is 'inhabited' by a proof p. Now a proof of $A \Rightarrow B$ consists (according to the Brouwer-Heyting interpretation of implication) of a function having as argument a proof of A and as value a proof of B. In symbols

$$[A \Rightarrow B] = [A] \rightarrow [B].$$

Similarly

$$[\forall x \in X.Px] = \Pi x\ X.[Px],$$

where $\Pi x\ A.[Px]$ is the Cartesian product of the $[Px]$, because a proof of $\forall x \in A.Px$ consists of a function that assigns to each element $x \in A$ a proof of Px. In this way proof-objects become isomorphic with the intuitionistic natural deduction proofs of Gentzen [1969]. Using this interpretation, a proof of $\forall y \in A.Py \Rightarrow Py$ is $\lambda y\ A \lambda x\ Py.x$. Here $\lambda x\ A.B(x)$ denotes the function that assigns to input $x \in A$ the output $B(x)$. A proof of

$$(A \Rightarrow A \Rightarrow B) \Rightarrow A \Rightarrow B$$

is

$$\lambda p\ (A \Rightarrow A \Rightarrow B) \lambda q\ A.pqq.$$

A description of the typed lambda calculi in which these types and inhabitants can be formulated is given in B[92], which also gives an example of a large proof object. Verifying whether p is a proof of A boils down to verifying whether, in the given context, the type of p is equal (convertible) to $[A]$. The method can be extended by also representing connectives like and $\neg$ in the right type system. Translating propositions as types has as default intuitionistic logic. Classical logic can be dealt with by adding the excluded middle as an axiom.

If a complicated computer system claims that a certain mathematical statement is correct, then one may wonder whether this is indeed the case. For example, there may be software errors in the system. A satisfactory methodological answer has been given by de Bruijn. Proof-objects should be public and written in such a formalism that a reasonably simple proof-checker can verify them. One should be able to verify the program for this proof-checker 'by hand'. We call this the *de Bruijn criterion*. The proof-development systems Lego (see Luo and Pollack [1992]) and Coq (see Coquand and Huet [1988]) satisfy this criterion.

A way to keep proof-objects from growing too large is to employ the so-called Poincaré principle. Poincaré [1902], p. 12, stated that an argument

showing that $2 + 2 = 4$ "is not a proof in the strict sense, it is a verification" (actually he claimed that an arbitrary mathematician will make this remark). In the AUTOMATH project of de Bruijn the following interpretation of the Poincaré principle was given. If p is a proof of $A(t)$ and $t =_R t'$, then the same p is also a proof of $A(t')$. Here R is a notion of reduction consisting of ordinary β reduction and δ-reduction in order to deal with the unfolding of definitions. Since $\beta\delta$-reduction is not too complicated to be programmed, the type systems enjoying this interpretation of the Poincaré principle still satisfy the de Bruijn criterion[3].

In spite of the compact representation in typed lambda calculi and the use of the Poincaré principle, proof-objects become large, something like 10 to 30 times the length of a complete informal proof. Large proof-objects are tiresome to generate by hand. With the necessary persistence Jutting [1977] has written lambda after lambda to obtain the proof-objects showing that all proofs (but one) in Landau [1960] are correct. Using a modern system for CM one can do better. The user introduces the context consisting of the primitive notions and axioms. Then necessary definitions are given to formulate a theorem to be proved (the goal). The proof is developed in an interactive session with the machine. Thereby the user only needs to give certain 'tactics' to the machine. (The interpretation of these tactics by the machine does nothing mathematically sophisticated, only the necessary bookkeeping. The sophistication comes from giving the right tactics.) The final goal of this research is that the necessary effort to interactively generate formal proofs is not more complicated than producing a text in, say, LaTeX. This goal has not been reached yet. See Barendregt [1996] for references, including those about other approaches to computer mathematics. (These include the systems NuPrl, HOL, Otter, Mizar and the Boyer-Moore theorem prover. These systems do not satisfy the de Bruijn criterion, but some of them probably can be modified easily so that they do.)

Computations in proofs

The following is taken from Barendregt and Barendsen [1997]. There are several computations that are needed in proofs. This happens, for example, if we want to prove formal versions of the following intuitive statements.

(1) $[\sqrt{45}] = 6$, where $[r]$ is the integer part of a real;
(2) $\texttt{Prime}(61)$;
(3) $(x+1)(x+1) = x^2 + 2x + 1$.

A way to handle (1) is to use the Poincaré principle extended to the reduction relation $\twoheadrightarrow_\iota$ for primitive recursion on the natural numbers. Operations like

[3]The reductions may sometimes cause the proof-checking to be of an unacceptable time complexity. We have that p is a proof of A iff $\texttt{type}(p) =_{\beta\delta} A$. Because the proof is coming from a human, the necessary conversion path is feasible, but to find it automatically may be hard. The problem probably can be avoided by enhancing proof-objects with hints for a reduction strategy.

$f(n) = [\sqrt{n}\,]$ are primitive recursive and hence are lambda definable (using $\twoheadrightarrow_{\beta\iota}$) by a term, say F, in the lambda calculus extended by an operation for primitive recursion R satisfying

$$\begin{aligned} R\,A\,B\,\mathtt{zero} &\to_\iota A \\ R\,A\,B\,(\mathtt{succ}\,x) &\to_\iota B\,x\,(R\,A\,B\,x). \end{aligned}$$

Then, writing $\ulcorner 0 \urcorner = \mathtt{zero}, \ulcorner 1 \urcorner = \mathtt{succ}\,\mathtt{zero}, \ldots$, as

$$\ulcorner 6 \urcorner = \ulcorner 6 \urcorner$$

is formally derivable, it follows from the Poincaré principle that the same is true for

$$F \ulcorner 45 \urcorner = \ulcorner 6 \urcorner$$

(with the same proof-object), since $F \ulcorner 45 \urcorner \twoheadrightarrow_{\beta\iota} \ulcorner 6 \urcorner$. Usually, a proof obligation arises that F is adequately constructed. For example, in this case it could be

$$\forall n\ (F\,n)^2 \ \leq\ n\ < ((F\,n)+1)^2.$$

Such a proof obligation needs to be formally proved, but only once; after that reductions like

$$F \ulcorner n \urcorner \twoheadrightarrow_{\beta\iota} \ulcorner f(n) \urcorner$$

can be used freely many times.

In a similar way, a statement like (2) can be formulated and proved by constructing a lambda defining term $K_{\mathtt{Prime}}$ for the characteristic function of the predicate `Prime`. This term should satisfy the following statement

$$\begin{aligned} \forall n \quad & [(\mathtt{Prime}\,n \ \leftrightarrow\ K_{\mathtt{Prime}}\,n = \ulcorner 1 \urcorner) \\ & (K_{\mathtt{Prime}}\,n = \ulcorner 0 \urcorner \ \vee\ K_{\mathtt{Prime}}\,n = \ulcorner 1 \urcorner)]. \end{aligned}$$

which is the proof obligation.

Statement (3) corresponds to a symbolic computation. This computation takes place on the syntactic level of formal terms. There is a function g acting on syntactic expressions satisfying

$$g((x+1)(x+1)\,) = x^2 + 2x + 1,$$

that we want to lambda define. While $x + 1 : \mathsf{Nat}$ (in context $x\ \mathsf{Nat}$), the expression on a syntactic level represented internally satisfies 'x+1' : $\mathtt{term}(\mathsf{Nat})$, for the suitably defined inductive type $\mathtt{term}(\mathsf{Nat})$. After introducing a reduction relation $\twoheadrightarrow_\iota$ for primitive recursion over this data type, one can use techniques similar to those of §3 to lambda define g, say by G, so that

$$G\,\text{`}(x+1)(x+1)\text{'} \twoheadrightarrow_{\beta\iota} \text{`}x^2 + 2x + 1\text{'}.$$

Now in order to finish the proof of (3), one needs to construct a self-interpreter E, such that for all expressions $p : \mathsf{Nat}$ one has

$$\mathsf{E}\ \text{`}p\text{'}\ \twoheadrightarrow_{\beta\iota}\ p$$

and prove the proof obligation for G which is

$$\forall t\, \mathtt{term}(\mathsf{Nat})\ \mathsf{E}(G\, t) \ = \ \mathsf{E}\, t.$$

It follows that

$$\mathsf{E}(G\, \text{`}(x+1)(x+1)\text{'}) \ = \ \mathsf{E}\, \text{`}(x+1)(x+1)\text{'};$$

now since

$$\begin{array}{rll} \mathsf{E}(G\, \text{`}(x+1)(x+1)\text{'}) & \twoheadrightarrow_{\beta\iota} & \mathsf{E}\, \text{`}x^2+2x+1\text{'} \\ & \twoheadrightarrow_{\beta\iota} & x^2+2x+1 \\ \mathsf{E}\, \text{`}(x+1)(x+1)\text{'} & \twoheadrightarrow_{\beta\iota} & (x+1)(x+1), \end{array}$$

we have by the Poincaré principle

$$(x+1)(x+1) \ = \ x^2+2x+1.$$

The use of inductive types like Nat and $\mathtt{term}(\mathsf{Nat})$ and the corresponding reduction relations for primitive reduction was suggested by Scott [1970] and the extension of the Poincaré principle for the corresponding reduction relations of primitive recursion by Martin-Löf [1984]. Since such reductions are not too hard to program, the resulting proof checking still satisfies the de Bruijn criterion.

In Oostdijk [1996] a program is presented that, for every primitive recursive predicate P, constructs the lambda term K_P defining its characteristic function and the proof of the adequacy of K_P. The resulting computations for $P =$ `Prime` are not efficient, because a straightforward (non-optimized) translation of primitive recursion is given and the numerals (represented numbers) used are in a unary (rather than n-ary) representation; but the method is promising. In Elbers [1996], a more efficient ad hoc lambda definition of the characteristic function of `Prime` is given, using Fermat's small theorem about primality. Also the required proof obligation has been given.

Choice of formal systems

There are several possibilities for the choice of a formal system to be used for the representation of theories in systems of computer mathematics. Since, in constructing proof-objects, cooperation between researchers is desirable, this choice has to be made with some care in order to reach an international standard. As a first step towards this, one may restrict attention to systems of typed lambda calculi, since they provide a compact representation and meet de Bruijn's criterion of having a simple proof-checker. In their simplest form, these systems can be described in a uniform way as pure type systems (PTS's) of different strength, see B[92]. The PTS's should be extended by a definition mechanism to become DPTS's (PTS's with definitions), see Severi and Poll [1994]. The DPTS's are good for describing several variants of logic: many sorted predicate logic in its first, second or higher order versions. As stated before, the default logic is intuitionistic, but can be made classical by assuming the excluded middle.

The next step consists of adding inductive types (IT's) and the corresponding reduction relations in order to capture primitive recursion. We suggest that the right formal systems to be used for computer mathematics are the type systems (TS), consisting of DPTS's extended by IT's, as described e.g. in Paulin-Mohring [1994]. TS's come with two parameters. The first is the specification $\mathcal{A}$ of the underlying PTS specifying its logical strength, see B[92]. The second is $\mathcal{B}$ the collection of inductive types and their respective notions of reduction $\twoheadrightarrow_\iota$ specifying its mathematical and computational strength. In my opinion, a system for proof-checking should be able to verify proof-objects written in all the systems TS$(\mathcal{A},\mathcal{B})$ (for a 'reasonable' choice spectrum of the parameters). If someone wants to use it for only a subclass of the choice of parameters—dictated by that person's foundational views—then the proof-checker will do its work anyway. I believe that this generality will not be too expensive in terms of the complexity of the checking.[4]

Illative lambda calculus

Curry and his students continued to look for a way to represent functions and logic into one adequate formal system. Some of the proposed systems turned out to be inconsistent, other ones turned out to be incomplete. Research in TS's for the representation of logic has resulted in an unexpected side effect. By making a modification inspired by the TS's, it became possible, after all, to give an extension of the untyped lambda calculus, called *Illative Lambda Calculi* (ILC; 'illative' from the Latin word *inferre* which means to infer), such that first order logic can be faithfully and completely embedded into it. The method can be extended for an arbitrary PTS[5], so that higher order logic can be represented too.

The resulting ILC's are in fact simpler than the TS's. But doing computer mathematics via ILC is probably not very practical, as it is not clear how to do proof-checking for these systems.

One nice thing about the ILC is that the old dream of Church and Curry came true, namely, there is one system based on untyped lambda calculus (or combinators) on which logic, hence mathematics, can be based. More importantly there is a 'combinatory transformation' between the ordinary interpretation of logic and its propositions-as-types interpretation. Basically, the situation

[4]It may be argued that the following list of features is so important that they deserve to be present in TS's as primitives and be implemented: quotient types (see Hofmann [1977]), subtypes (see Aspinall and Compagnoni [1996]) and type inclusion (see Luo and Pollack [1992]). This is an interesting question and experiments should be done to determine whether this is the case or whether these can be translated into the more basic TS's in a sufficiently efficient way (possibly using some macros in the system for CM).

[5]For first order logic, the embedding is natural, but e.g. for second order logic this is less so. It is an open question whether there exists a natural representation of second and higher order logic in ILC.

is as follows. The interpretation of predicate logic in ILC is such that

$$\begin{aligned} \vdash_{\text{logic}} A \text{ with proof } p &\iff \forall r \vdash_{\text{ILC}} [A]_r[p] \\ &\iff \vdash_{\text{ILC}} [A]_{\mathsf{I}}[p] \\ &\iff \vdash_{\text{ILC}} [A]_{\mathsf{K}}[p] = \mathsf{K}[A]'_{\mathsf{I}}[p] = [A]'_{\mathsf{I}}, \end{aligned}$$

where r ranges over untyped lambda terms. Now if $r = \mathsf{I}$, then this translation is the propositions-as-types interpretation; if, on the other hand, one has $r = \mathsf{K}$, then the interpretation becomes an isomorphic version of first order logic denoted by $[A]'_{\mathsf{I}}$. See Barendregt et al. [1993] and Dekkers et al. [1997] for these results. A short introduction to ILC (in its combinatory version) can be found in B[84], Appendix B.

6.3. Proof theory

Lambda terms for natural deduction, sequent calculus and cut elimination

It is well-known that there is a good correspondence between natural deduction derivations and typed lambda terms. Moreover normalizing these terms is equivalent to eliminating cuts in the corresponding sequent calculus derivations. Several papers have been written on this topic. The correspondence between sequent calculus derivations and natural deduction derivations is, however, not a one-to-one map. This causes some syntactic technicalities. The correspondence is best explained by two extensionally equivalent type assignment systems for untyped lambda terms, one corresponding to natural deduction (λN) and the other to sequent calculus (λL). These two systems constitute different grammars for generating the same (type assignment relation for untyped) lambda terms. The second grammar is ambiguous, but the first one is not. This fact explains the many-one correspondence mentioned above. Moreover, the second type assignment system has a 'cut–free' fragment (λL^{cf}). This fragment generates exactly the typeable lambda terms in normal form. The cut elimination theorem becomes a simple consequence of the fact that typed lambda terms posses a normal form.

Introduction

It is well-known that there is a good correspondence between natural deduction derivations and typed lambda terms. The relation between lambda terms and derivations in sequent calculus, between normal lambda terms and cut–free derivations in sequent calculus and finally between normalization of terms and cut elimination of derivations has been observed by several authors (Prawitz [1965], Zucker [1974] and Pottinger [1977]). This relation is less perfect because several cut–free sequent derivations correspond to one lambda term. In Herbelin [1995] a lambda calculus with explicit substitution operators is used in order to establish a perfect match between terms of that calculus and sequent derivations. We will not avoid the mismatch, but get a satisfactory view of

it, by seeing the sequent calculus as a more intensional way to do the same as natural deduction: assigning lambda terms to provable formulas.

Next to the well-known system $\lambda_{\rightarrow}$ of Curry type assignment to type free terms, which here will be denoted by λN, there are two other systems of type assignment: λL and its cut-free fragment λL^{cf}. The three systems λN, λL and λL^{cf} correspond exactly to the natural deduction calculus NJ, the sequent calculus LJ and the cut–free fragment of LJ, here denoted by N, L and L^{cf} respectively. Moreover, λN and λL generate the same type assignment relation. The system λL^{cf} generates the same type assignment relation as λN restricted to normal terms and cut elimination corresponds exactly to normalization. The mismatch between the logical systems that was observed above, is due to the fact that λN is a syntax directed system, whereas both λL and λL^{cf} are not. (A syntax directed version of λL is possible if rules with arbitrarily many assumptions are allowed, see Capretta and Valentini [1998].)

The type assignment system of this paper is a subsystem of one in Barbanera, Dezani-Ciancaglini and de' Liguoro [1995] and also implicitly present in Mints [1996]. So our contribution is mainly expository.

For simplicity the results are presented only for the essential kernel of intuitionistic logic, i.e. for the minimal implicational fragment. The method probably can be extended to the full logical system, using the terms as in Mints [1996].

The logical systems N, L and $L^{\mathbf{cf}}$

6.3.1. Definition. The set `form` of formulas (of minimal implicational propositional logic) is defined by the following abstract syntax.

`form`	=	`atom` | `form`→`form`
`atom`	=	`p` | `atom`′

We write $p, q, r, \ldots$ for arbitrary atoms and $A, B, C, \ldots$ for arbitrary formulas. Sets of formulas are denoted by $\Gamma, \Delta, \ldots$. The set Γ, A stands for $\Gamma \cup \{A\}$.

6.3.2. Definition. (i) A statement A is *derivable* in the system N from the set Γ, notation $\Gamma \vdash_N A$, if $\Gamma \vdash A$ can be generated by the following axiom and rules.

N

$$\frac{A \in \Gamma}{\Gamma \vdash A}\ \text{axiom}$$

$$\frac{\Gamma \vdash A{\rightarrow}B \qquad \Gamma \vdash A}{\Gamma \vdash B}\ {\rightarrow}\ \text{elim}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A{\rightarrow}B}\ {\rightarrow}\ \text{intr}$$

(ii) A statement A is *derivable* from assumptions Γ in the system L, notation $\Gamma \vdash_L A$, if $\Gamma \vdash A$ can be generated by the following axiom and rules.

L	
$\dfrac{A \in \Gamma}{\Gamma \vdash A}$	axiom
$\dfrac{\Gamma \vdash A \qquad \Gamma, B \vdash C}{\Gamma, A{\to}B \vdash C}$	$\to$ left
$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A{\to}B}$	$\to$ right
$\dfrac{\Gamma \vdash A \qquad \Gamma, A \vdash B}{\Gamma \vdash B}$	cut

(iii) The system L^{cf} is obtained from the system L by omitting the rule (cut).

L^{cf}	
$\dfrac{A \in \Gamma}{\Gamma \vdash A}$	axiom
$\dfrac{\Gamma \vdash A \qquad \Gamma, B \vdash C}{\Gamma, A{\to}B \vdash C}$	$\to$ left
$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A{\to}B}$	$\to$ right

6.3.3. Lemma. *Suppose* $\Gamma \subseteq \Gamma'$. *Then*

$$\Gamma \vdash A \;\Rightarrow\; \Gamma' \vdash A$$

in all systems.

Proof. By a trivial induction on derivations. ■

6.3.4. Proposition. *For all* Γ *and* A *we have*

$$\Gamma \vdash_N A \iff \Gamma \vdash_L A.$$

Proof. ($\Rightarrow$) By induction on derivations in N. For the rule ($\to$ elim) we need the rule (cut).

$$\dfrac{\Gamma \vdash_L A{\to}B \qquad \dfrac{\Gamma \vdash_L A \qquad \dfrac{}{\Gamma, B \vdash_L B}\,(\text{axiom})}{\Gamma, A{\to}B \vdash_L B}\,(\to \text{ left})}{\Gamma \vdash_L B}\,(\text{cut})$$

($\Leftarrow$) By induction on derivations in L. The rule ($\to$ left) is treated as follows.

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash_N A}{\Gamma, A{\to}B \vdash_N A}\,(6.3.3) \quad \dfrac{}{\Gamma, A{\to}B \vdash_N A{\to}B}\,(\text{axiom})}{\Gamma, A{\to}B \vdash_N B}\,(\to \text{elim}) \quad \dfrac{\Gamma, B \vdash_N C}{\Gamma \vdash_N B{\to}C}\,(\to \text{intr})}{\Gamma, A{\to}B \vdash_N C}\,(\to \text{elim})$$

The rule (cut) is treated as follows.

$$\dfrac{\Gamma \vdash_N A \quad \dfrac{\Gamma, A \vdash_N B}{\Gamma \vdash_N A{\to}B}\,(\to \text{intr})}{\Gamma \vdash_N B}\,(\to \text{elim}).\ \blacksquare$$

6.3.5. Definition. Consider the following rule as alternative to the rule (cut).

$$\dfrac{\Gamma, A{\to}A \vdash B}{\Gamma \vdash B}\,(\text{cut'})$$

The system L' is defined by replacing the rule (cut) by (cut').

6.3.6. Proposition. *For all* Γ *and* A

$$\Gamma \vdash_L A \iff \Gamma \vdash_{L'} A.$$

Proof. ($\Rightarrow$) The rule (cut) is treated as follows.

$$\dfrac{\dfrac{\Gamma \vdash_{L'} A \qquad \Gamma, A \vdash_{L'} B}{\Gamma, A{\to}A \vdash_{L'} B}\,(\to \text{left})}{\Gamma \vdash_{L'} B}\,(\text{cut'})$$

($\Leftarrow$) The rule (cut') is treated as follows.

$$\dfrac{\Gamma, A{\to}A \vdash_L B \quad \dfrac{\dfrac{}{\Gamma, A \vdash_L A}\,(\text{axiom})}{\Gamma \vdash_L A{\to}A}\,(\to \text{right})}{\Gamma \vdash_L B}\,(\text{cut}).\ \blacksquare$$

Note that we have not yet investigated the role of L^{cf}.

The type assignment systems λN, λL and λL^{cf}

6.3.7. Definition. The set `term` of type–free lambda terms is defined as follows.

`term`	=	`var` $\mid$ `term term` $\mid$ λ`var.term`
`var`	=	`x` $\mid$ `var`$'$

We write $x, y, z, \ldots$ for arbitrary variables in terms and $P, Q, R, \ldots$ for arbitrary terms. Equality of terms (up to renaming of bound variables) is denoted by $\equiv$. The identity is $\mathsf{I} \equiv \lambda x.x$. A term P is called a β normal form (P *is in* β*-nf*) if P has no *redex* as part, i.e. no subterm of the form $(\lambda x.R)S$. Such a redex is said to reduce as follows

$$(\lambda x.R)S \to_\beta R[x{:=}S],$$

where $R[x{:=}S]$ denotes the result of substituting S for the free occurrences of x. The transitive reflexive closure of $\to_\beta$ is denoted by $\twoheadrightarrow_\beta$. If $P \twoheadrightarrow_\beta Q$ and Q is in β-nf, then Q is called the β-nf of P (one can show it is unique). A collection $\mathcal{A}$ of terms is said to be *strongly normalizing* if for no $P \in \mathcal{A}$ there is an infinite reduction path

$$P \to_\beta P_1 \to_\beta P_2 \ldots .$$

6.3.8. Definition. (i) A *type assignment* is an expression of the form

$$P : A,$$

where P is a lambda term and A is a formula.

(ii) A *declaration* is a type assignment of the form

$$x : A.$$

(iii) A *context* Γ is a set of declarations such that for every variable x there is at most one declaration $x\ A$ in Γ.

6.3.9. Definition. (i) A type assignment $P : A$ is *derivable* from the context Γ in the system λN (also known as $\lambda_\to$), notation

$$\Gamma \vdash_{\lambda N} P : A,$$

if $\Gamma \vdash P : A$ can be generated by the following axiom and rules.

λN

$$\frac{(x\ A) \in \Gamma}{\Gamma \vdash x : A} \quad \text{axiom}$$

$$\frac{\Gamma \vdash P : (A \to B) \qquad \Gamma \vdash Q : A}{\Gamma \vdash (PQ) : B} \quad \to \text{elim}$$

$$\frac{\Gamma, x\ A \vdash P : B}{\Gamma \vdash (\lambda x.P) : (A \to B)} \quad \to \text{intr}$$

(ii) A type assignment $P : A$ is *derivable* form the context Γ in the system λL, notation

$$\Gamma \vdash_{\lambda L} P : A,$$

if $\Gamma \vdash P : A$ can be generated by the following axiom and rules.

λL	
$\dfrac{(x\ A) \in \Gamma}{\Gamma \vdash x\ A}$	axiom
$\dfrac{\Gamma \vdash Q : A \qquad \Gamma, x\ B \vdash P : C}{\Gamma, y : A{\to}B \vdash P[x{:=}yQ] : C}$	$\to$ left
$\dfrac{\Gamma, x\ A \vdash P : B}{\Gamma \vdash (\lambda x.P) : (A{\to}B)}$	$\to$ right
$\dfrac{\Gamma \vdash Q : A \qquad \Gamma, x\ A \vdash P : B}{\Gamma \vdash P[x{:=}Q] : B}$	cut

In the rule ($\to$ left) it is required that $\Gamma, y\ A{\to}B$ is a context. This is the case if y is fresh or if $\Gamma = \Gamma, y\ A{\to}B$, i.e. $y\ A{\to}B$ already occurs in Γ.

(iii) The system λL^{cf} is obtained from the system λL by omitting the rule (cut).

λL^{cf}	
$\dfrac{(x\ A) \in \Gamma}{\Gamma \vdash x\ A}$	axiom
$\dfrac{\Gamma \vdash Q : A \qquad \Gamma, x\ B \vdash P : C}{\Gamma, y : A{\to}B \vdash P[x{:=}yQ] : C}$	$\to$ left
$\dfrac{\Gamma, x\ A \vdash P : B}{\Gamma \vdash (\lambda x.P) : (A{\to}B)}$	$\to$ right

6.3.10. Remark. The alternative rule (cut') could also have been used to define the variant $\lambda L'$. The right version for the rule (cut') with term assignment is as follows.

Rule cut' for $\lambda L'$	
$\dfrac{\Gamma,\ x\ A{\to}A \vdash P : B}{\Gamma \vdash P[x{:=}\mathsf{I}] : B}$	cut'

Notation. Let $\Gamma = \{A_1, \ldots, A_n\}$ and $\vec{x} = \{x_1, \ldots, x_n\}$. Write

$$\Gamma_{\vec{x}} = \{x_1\ A_1, \ldots, x_n\ A_n\}$$

and

$$\Lambda^\circ(\vec{x}) = \{P \in \mathtt{term} \mid FV(P) \subseteq \vec{x}\},$$

where $FV(P)$ is the set of free variables of P.

The following result has been observed for N and λN by Curry, Howard and de Bruijn. (See Troelstra and Schwichtenberg [1996] 2.1.5. and Hindley [1997] 6B3, for some fine points about the correspondence between deductions in N and corresponding terms in λN.)

6.3.11. Proposition (Propositions—as—types interpretation). *Let S be one of the logical systems N, L or L^{cf} and let λS be the corresponding type assignment system. Then*

$$\Gamma \vdash_S A \Leftrightarrow \exists \vec{x}\, \exists P \in \Lambda^\circ(\vec{x})\ \Gamma_{\vec{x}} \vdash_{\lambda S} P : A.$$

Proof. ($\Rightarrow$) By an easy induction on derivations, just observing that the right lambda term can be constructed. ($\Leftarrow$) By omitting the terms. ■

Since λN is exactly $\lambda_\rightarrow$, the simply typed lambda calculus, we know the following results whose proofs are not hard, but are omitted here. From corollary 6.3.15 it follows that the results also hold for λL.

6.3.12. Proposition. (i) *(Normalization theorem for λN)*

$$\Gamma \vdash_{\lambda N} P : A \ \Rightarrow\ P \text{ has a } \beta\text{-nf } P^{nf}.$$

(ii) *(Subject reduction theorem for λN)*

$$\Gamma \vdash_{\lambda N} P : AP \twoheadrightarrow_\beta P' \ \Rightarrow\ \Gamma \vdash_{\lambda N} P' : A.$$

(iii) *(Generation lemma for λN) Type assignment for terms of a certain syntactic form is caused in the obvious way.*

(1)	$\Gamma \vdash_{\lambda N}$	x	:	A	$\Rightarrow$	$(x\ A) \in \Gamma$.
(2)	$\Gamma \vdash_{\lambda N}$	PQ	:	B	$\Rightarrow$	$\Gamma \vdash_{\lambda N} P : (A \rightarrow B) \Gamma \vdash_{\lambda N} Q : A$, *for some type A.*
(3)	$\Gamma \vdash_{\lambda N}$	$\lambda x.P$	:	C	$\Rightarrow$	$\Gamma, x\ A \vdash_{\lambda N} P : BC \equiv A \rightarrow B$, *for some types A, B.*

Proof. See e.g. Gandy [1980a] for (i) and Barendregt [1992] for (ii) and (iii). ■

Actually, even strong normalization holds for terms typeable in λN (see e.g. de Vrijer [1987] or Barendregt [1992]).

Relating λN, λL and λL^{cf}

Now the proof of the equivalence between systems N and L will be 'lifted' to that of λN and λL.

6.3.13. PROPOSITION. $\Gamma \vdash_{\lambda N} P : A \;\Rightarrow\; \Gamma \vdash_{\lambda L} P : A$.

PROOF. By inductions on derivations in λN. Modus ponens ($\to$ elim) is treated as follows.

$$\dfrac{\Gamma \vdash_{\lambda L} P : A{\to}B \qquad \dfrac{\Gamma \vdash_{\lambda L} Q : A \quad \Gamma, x\ B \vdash_{\lambda L} x\ B}{\Gamma, y\ A{\to}B \vdash_{\lambda L} yQ : B}\ (\to \text{ left})}{\Gamma \vdash_{\lambda L} PQ : B}\ (\text{cut}).\ \blacksquare$$

6.3.14. PROPOSITION. (i) $\Gamma \vdash_{\lambda L} P : A \;\Rightarrow\; \Gamma \vdash_{\lambda N} P' : A$, *for some* $P' \twoheadrightarrow_\beta P$.

(ii) $\Gamma \vdash_{\lambda L} P : A \;\Rightarrow\; \Gamma \vdash_{\lambda N} P : A$.

PROOF. (i) By induction on derivations in λL. The rule ($\to$ left) is treated as follows (the justifications are left out, but they are as in the proof of 6.3.4).

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash_{\lambda N} Q : A}{\Gamma, y\ A{\to}B \vdash_{\lambda N} Q : A} \qquad \dfrac{}{\Gamma, y\ A{\to}B \vdash_{\lambda N} y\ A{\to}B}}{\Gamma, y\ A{\to}B \vdash_{\lambda N} yQ : B} \qquad \dfrac{\Gamma, x\ B \vdash_{\lambda N} P : C}{\Gamma \vdash_{\lambda N} (\lambda x.P) : B{\to}C}}{\Gamma, y\ A{\to}B \vdash_{\lambda N} (\lambda x.P)(yQ) : C}$$

Now $(\lambda x.P)(yQ) \to_\beta P[x{:=}yQ]$ as required. The rule (cut) is treated as follows.

$$\dfrac{\Gamma \vdash_{\lambda N} Q : A \quad \dfrac{\Gamma, x\ A \vdash_{\lambda N} P : B}{\Gamma \vdash_{\lambda N} (\lambda x.P) : A{\to}B}\ (\to \text{ intr})}{\Gamma \vdash_{\lambda N} (\lambda x.P)Q : B}\ (\to \text{ elim})$$

Now $(\lambda x.P)Q \to_\beta P[x{:=}Q]$ as required.

(ii) By (i) and the subject reduction theorem for λN (6.3.12(ii)). $\blacksquare$

6.3.15. COROLLARY. $\Gamma \vdash_{\lambda L} P : A \iff \Gamma \vdash_{\lambda N} P : A$.

PROOF. By propositions 6.3.13 and 6.3.14(ii). $\blacksquare$

Now we will investigate the role of the cut–free system.

6.3.16. PROPOSITION.

$$\Gamma \vdash_{\lambda L^{\mathrm{cf}}} P : A \;\Rightarrow\; P \textit{ is in } \beta\textit{-nf}.$$

PROOF. By an easy induction on derivations. $\blacksquare$

6.3.17. Lemma. *Suppose*

$$\Gamma \vdash_{\lambda L^{\mathrm{cf}}} P_1 : A_1,\ \ldots\ ,\ \Gamma \vdash_{\lambda L^{\mathrm{cf}}} P_n : A_n.$$

Then

$$\Gamma, x\ A_1{\to}\ldots{\to}A_n{\to}B \vdash_{\lambda L^{\mathrm{cf}}} xP_1 \ldots P_n : B$$

for those variables x such that $\Gamma, x\ A_1{\to}\ldots{\to}A_n{\to}B$ is a context.

Proof. We treat the case $n = 2$, which is perfectly general. We abbreviate $\vdash_{\lambda L^{\mathrm{cf}}}$ as $\vdash$.

$$\dfrac{\Gamma \vdash P_1 : A_1 \qquad \dfrac{\Gamma \vdash P_2 : A_2 \qquad \dfrac{}{\Gamma, z\ B \vdash z : B}\,(\text{axiom})}{\Gamma, y\ A_2{\to}B \vdash yP_2 \equiv z[z{:=}yP_2] : B}\,(\to \text{left})}{\Gamma, x\ A_1{\to}A_2{\to}B \vdash xP_1P_2 \equiv (yP_2)[y{:=}xP_1] : B}\,(\to \text{left})$$

Note that x may occur in some of the P_i. ■

6.3.18. Proposition. *Suppose that P is a β-nf. Then*

$$\Gamma{\vdash_{\lambda N}} P : A \ \Rightarrow\ \Gamma \vdash_{\lambda L^{\mathrm{cf}}} P : A.$$

Proof. By induction on the following generation of normal forms.

$$\mathrm{nf} = \mathtt{var} \mid \mathtt{var}\ \mathrm{nf}^{+} \mid \lambda\mathtt{var}.\mathrm{nf}$$

The cases $P \equiv x$ and $P \equiv \lambda x.P_1$ are easy. The case $P \equiv xP_1 \ldots P_n$ follows from the previous lemma, using the generation lemma for λN (6.3.12(iii)). ■

Now we get as bonus the Hauptsatz of Gentzen [1936] for minimal implicational sequent calculus.

6.3.19. Theorem (Cut elimination).

$$\Gamma \vdash_L A \ \Rightarrow\ \Gamma \vdash_{L^{\mathrm{cf}}} A.$$

Proof.
$\Gamma \vdash_L A \ \Rightarrow\ \Gamma_{\vec{x}} \vdash_{\lambda L} P : A$, for some $P \in \Lambda^{\circ}(\vec{x})$, by 6.3.11,
$\Rightarrow\ \Gamma_{\vec{x}} \vdash_{\lambda N} P : A$, by 6.3.14(ii),
$\Rightarrow\ \Gamma_{\vec{x}} \vdash_{\lambda N} P^{\mathrm{nf}} : A$, by 6.3.12(i),(ii),
$\Rightarrow\ \Gamma_{\vec{x}} \vdash_{\lambda L^{\mathrm{cf}}} P^{\mathrm{nf}} : A$, by 6.3.18,
$\Rightarrow\ \Gamma \vdash_{L^{\mathrm{cf}}} A$, by 6.3.11. ■

As it is clear that the proof implies that cut-elimination can be used to normalize terms typable in $\lambda N = \lambda{\to}$, Statman [1979] implies that the expense of cut-elimination is beyond elementary time (Grzegorczyk class 4). Moreover, as the cut-free deduction is of the same order of complexity as the corresponding normal lambda term, the size of the cut-free version of a derivation is non elementary in the size of the original derivation.

Discussion

The main technical tool is the type assignment system λL corresponding exactly to sequent calculus (for minimal propositional logic). The type assignment system λL is a subsystem of a system studied in Barbanera, Dezani-Ciancaglini and de' Liguoro [1995]. The terms involved in λL are also in Mints [1996]. The difference between the present approach and the one by Mints is that in that paper derivations in L are first order citizens, whereas in λL the provable formulas and the lambda terms are.

In λN typeable terms are built up as usual (following the grammar of lambda terms). In λL^{cf} only normal terms are typeable. They are built up from variables by transitions like

$$P \longmapsto \lambda x.P$$

and

$$P \longmapsto P[x{:=}yQ]$$

This is an ambiguous way of building terms, in the sense that one term can be built up in several ways. For example, one can assign to the term $\lambda x.yz$ the type $C{\to}B$ (in the context $z\ A, y\ A{\to}B$) via two different cut–free derivations:

$$\dfrac{\dfrac{x\ C, z\ A \vdash z : A \qquad x\ C, z\ A, u\ B \vdash u : B}{x\ C, z\ A, y\ A{\to}B \vdash yz : B}\ (\to \text{left})}{z\ A, y\ A{\to}B \vdash \lambda x.yz : C{\to}B}\ (\to \text{right})$$

and

$$\dfrac{z\ A \vdash z\ A \qquad \dfrac{x\ C, z\ A, u\ B \vdash u : B}{z\ A, u\ B \vdash \lambda x.u : C{\to}B}\ (\to \text{right})}{z\ A, y\ A{\to}B \vdash \lambda x.yz : C{\to}B}\ (\to \text{left})$$

These correspond, respectively, to the following two formations of terms

$$\begin{array}{lllll} u & \longmapsto & yz & \longmapsto & \lambda x.yz, \\ u & \longmapsto & \lambda x.u & \longmapsto & \lambda x.yz. \end{array}$$

Therefore there are more sequent calculus derivations giving rise to the same lambda term. This is the cause of the mismatch between sequent calculus and natural deduction as described in Zucker [1974], Pottinger [1977] and Mints [1996]. See also Dyckhoff and Pinto [1997], Schwichtenberg [1997] and Troelstra [1998].

In Herbelin [1995] the mismatch between L-derivations and lambda terms is repaired by translating these into terms with explicit substitution:

$$\begin{array}{c} \lambda x.(u < u{:=}yz >), \\ (\lambda x.u) < u{:=}yz > . \end{array}$$

In our paper lambda terms are considered as first class citizens also for sequent calculus. This gives an insight into the mentioned mismatch by understanding it as an intensional aspect how the sequent calculus generates these terms.

It is interesting to note, how in the full system λL the rule (cut) generates terms not in β–normal form. The extra transition now is

$$P \longmapsto P[x{:=}F].$$

This will introduce a redex, if x occurs actively (in a context xQ) and F is an abstraction ($F \equiv \lambda x.R$), the other applications of the rule (cut) being superfluous. Also, the alternative rule (cut') can be understood better. Using this rule the extra transition becomes

$$P \longmapsto P[x{:=}\mathsf{I}].$$

This will have the same effect (modulo one β–reduction) as the previous transition, if x occurs in a context xFQ. So with the original rule (cut) the argument Q (in the context xQ) is waiting for a function F to act on it. With the alternative rule (cut') the function F comes close (in context xFQ), but the 'couple' FQ has to wait for the 'green light' provided by I.

Also, it can be observed that if one wants to manipulate derivations in order to obtain a cut–free proof, then the term involved gets reduced. By the strong normalization theorem for λN ($= \lambda_{\rightarrow}$) it follows that eventually a cut–free proof will be reached.

We have not studied in detail whether cut elimination can be done along the lines of this paper for the full system of intuitionistic predicate logic, but there seems to be no problem. More interesting is the question, whether there are similar results for classical and linear logic.

6.4. Grammars, terms and types

Typed lambda calculus is widely used in the study of natural language semantics, in combination with a variety of rule-based syntactic engines. In this section, we focus on categorial type logics. The type discipline, in these systems, is responsible both for the construction of grammatical form (syntax) and for meaning assembly. We address two central questions. First, what are the *invariants* of grammatical composition, and how do they capture the uniformities of the form/meaning correspondence across languages? Secondly, how can we reconcile grammatical invariants with structural diversity, i.e. variation in the realization of the form/meaning correspondence in the 6000 or so languages of the world?

The grammatical architecture to be unfolded below has two components. Invariants are characterized in terms of a minimal *base system*: the pure logic of residuation for composition and structural incompleteness. Viewing the types of the base system as formulas, we model the syntax-semantics interface along the lines of the Curry-Howard interpretation of derivations. Variation arises from the combination of the base logic with a *structural module*. This component characterizes the structural deformations under which the basic form-meaning associations are preserved. Its rules allow reordering and/or restructuring of grammatical material. These rules are not globally available, but keyed to unary type-forming operations, and thus anchored in the lexical type declarations.

It will be clear from this description that the type-logical appraoch has its roots in the type calculi developed by Jim Lambek in the late Fifties of the last century. The technique of controlled structural options is a more recent development, inspired by the modalities of linear logic.

Grammatical invariants: the base logic

Compared to the systems used elsewhere in this chapter, the type system of categorial type logics can be seen as a specialization designed to take linear order and phrase structure information into account.

$$\mathcal{F} ::= \mathcal{A} \mid \mathcal{F}/\mathcal{F} \mid \mathcal{F} \bullet \mathcal{F} \mid \mathcal{F}\backslash\mathcal{F}$$

The set of type atoms $\mathcal{A}$ represents the basic ontology of phrases that one can think of as grammatically 'complete'. Examples, for English, could be np for noun phrases, s for sentences, n for common nouns. There is no claim of universality here: languages can differ as to which ontological choices they make. Formulas A/B, $B\backslash A$ are directional versions of the implicational type $B \to A$. They express incompleteness in the sense that expressions with slash types produce a phrase of type A in composition with a phrase of type B to the right or to the left. Product types $A \bullet B$ explicitly express this composition.

Frame semantics provides the tools to make the informal description of the interpretation of the type language in the structural dimension precise. Frames $F = (W, R_\bullet)$, in this setting, consist of a set W of linguistic resources (expressions, 'signs'), structured in terms of a ternary relation $R_\bullet$, the relation of grammatical composition or 'Merge' as it is known in the generative tradition. A valuation $V : \mathcal{S} \mapsto \mathcal{P}(W)$ interprets types as sets of expressions. For complex types, the valuation respects the clauses below, i.e. expressions x with type $A \bullet B$ can be disassembled into an A part y and a B part z. The interpretation for the directional implications is dual with respect to the y and z arguments of the Merge relation, thus expressing incompleteness with respect to composition.

$$x \in V(A \bullet B) \text{ iff } \exists yz.R_\bullet xyz \text{ and } y \in V(A) \text{ and } z \in V(B)$$

$$y \in V(C/B) \text{ iff } \forall xz.(R_\bullet xyz \text{ and } z \in V(B)) \text{ implies } x \in V(C)$$

$$z \in V(A\backslash C) \text{ iff } \forall xy.(R_\bullet xyz \text{ and } y \in V(A)) \text{ implies } x \in V(C)$$

Algebraically, this interpretation turns the product and the left and right implications into a residuated triple in the sense of the following biconditionals:

$$A \longrightarrow C/B \iff A \bullet B \longrightarrow C \iff B \longrightarrow A\backslash C \qquad \text{(Res)}$$

In fact, we have the *pure* logic of residuation here: (Res), together with Reflexivity ($A \longrightarrow A$) and Transitivity (from $A \longrightarrow B$ and $B \longrightarrow C$, conclude $A \longrightarrow C$), fully characterizes the derivability relation, as the following completeness result shows.

COMPLETENESS $A \longrightarrow B$ is provable in the grammatical base logic iff for every valuation V on every frame F we have $V(A) \subseteq V(B)$ (Došen 1992, Kurtonina 1995).

Notice that we do not impose any restrictions on the interpretation of the Merge relation. In this sense, the laws of the base logic capture grammatical *invariants*: properties of type combination that hold no matter what the structural particularities of individual languages may be. And indeed, at the level of the base logic important grammatical notions, rather than being postulated, can be seen to emerge from the type structure.

- Valency. Selectional requirements distinguishing verbs that are intransitive $np\backslash s$, transitive $(np\backslash s)/np$, ditransitive $((np\backslash s)/np)/np$, etcetera are expressed in terms of the directional implications. In a context-free grammar, these would require the postulation of new non-terminals.
- Case. The distinction between phrases that can fulfill any noun phrase selectional requirement versus phrases that insist on playing the subject $s/(np\backslash s)$, direct object $((np\backslash s)/np)\backslash(np\backslash s)$, prepositional object $(pp/np)\backslash pp$, etc role, is expressed through higher-order type assignment.
- Complements versus modifiers. Compare exocentric types (A/B with $A \neq B$) versus endocentric types A/A. The latter express modification; optionality of A/A type phrases follows.
- Filler-gap dependencies. Nested implications $A/(C/B)$, $A/(B\backslash C)$, etc, signal the withdrawal of a gap hypothesis of type B in a domain of type C.

Parsing-as-deduction For automated proof search, one turns the algebraic presentation in terms of (Res) into a sequent presentation enjoying cut elimination. Sequents for the grammatical base logic are statements $\Gamma \Rightarrow A$ with Γ a structure, A a type formula. Structures are binary branching trees with formulas at the leaves: $\mathcal{S} ::= \mathcal{F} \mid (\mathcal{S}, \mathcal{S})$. In the rules, we write $\Gamma[\Delta]$ for a structure Γ containing a substructure Δ. Lambek (1958, 1961) proves that Cut is an admissible rule in this presentation. Top-down backward-chaining proof search in the cut-free system respects the subformula property and yields a decision procedure.

$$\frac{}{A \Rightarrow A}\ \mathrm{Ax} \qquad \frac{\Delta \Rightarrow A \quad \Gamma[A] \Rightarrow B}{\Gamma[\Delta] \Rightarrow B}\ \mathrm{Cut}$$

$$\frac{\Gamma \Rightarrow A \quad \Delta \Rightarrow B}{(\Gamma, \Delta) \Rightarrow A \bullet B}\ (\bullet R) \qquad \frac{\Gamma[(A, B)] \Rightarrow C}{\Gamma[A \bullet B] \Rightarrow C}\ (\bullet L)$$

$$\frac{\Delta \Rightarrow B \quad \Gamma[A] \Rightarrow C}{\Gamma[(\Delta, B\backslash A)] \Rightarrow C}\ (\backslash L) \qquad \frac{(B, \Gamma) \Rightarrow A}{\Gamma \Rightarrow B\backslash A}\ (\backslash R)$$

$$\frac{\Delta \Rightarrow B \quad \Gamma[A] \Rightarrow C}{\Gamma[(A/B, \Delta)] \Rightarrow C}\ (/L) \qquad \frac{(\Gamma, B) \Rightarrow A}{\Gamma \Rightarrow A/B}\ (/R)$$

To specify a grammar for a particular language it is enough now to give its lexicon. $Lex \subseteq \Sigma \times \mathcal{F}$ is a relation associating each word with a finite number of types. A string belongs to the language for lexicon Lex and goal type B, $w_1 \cdots w_n \in L(Lex, B)$ iff for $1 \leq i \leq n$, $(w_i, A_i) \in Lex$, and $\Gamma \Rightarrow B$ where Γ is

a tree with yield $A_1 \cdots A_n$. Buszkowski and Penn (1990) model the acquisition of lexical type assignments as a proces of solving type equations. Their unification-based algorithms take function-argument structures as input (binary trees with a distinguised daughter); one obtains variations depending on whether the solution should assign a unique type to every vocabulary item, or whether one accepts multiple assignments. Kanazawa (1998) studies learnable classes of grammars from this perspective, in the sense of Gold's notion of identifiability 'in the limit'; the formal theory of learnability for type-logical grammars has recently developed into a quite active field of research.

Meaning assembly Lambek's original work looked at categorial grammar from a purely syntactic point of view, which probably explains why this work was not taken into account by Richard Montague when he developed his theory of modeltheoretic semantics for natural languages. In the 1980-ies, Van Benthem played a key role in bringing the two traditions together, by introducing the Curry-Howard perspective, with its dynamic, derivational view on meaning assembly rather than the static, structure-based view of rule-based approaches.

For semantic interpretation, we want to associate every type A with a semantic domain D_A, the domain where expressions of type A find their denotations. It is convenient to set up semantic domains via a mapping from the directional syntactic types used so far to the undirected type system of the typed lambda calculus. This indirect approach is attractive for a number of reasons. On the level of atomic types, one may want to make different basic distinctions depending on whether one uses syntactic or semantic criteria. For complex types, a map from syntactic to semantic types makes it possible to forget information that is relevant only for the way expressions are to be configured in the form dimension. For simplicity, we focus on implicational types here — accommodation of product types is straightforward.

For a simple extensional interpretation, the set of atomic semantic types could consist of types e and t, with D_e the domain of discourse (a non-empty set of entities, objects), and $D_t = \{0, 1\}$, the set of truth values. $D_{A \to B}$, the semantic domain for a functional type $A \to B$, is the set of functions from D_A to D_B. The mapping from syntactic to semantic types $(\cdot)'$ could now stipulate for basic syntactic types that $np' = e$, $s' = t$, and $n' = e \to t$. Sentences, in this way, denote truth values; (proper) noun phrases individuals; common nouns functions from individuals to truth values. For complex syntactic types, we set $(A/B)' = (B\backslash A)' = B' \to A'$. On the level of semantic types, the directionality of the slash connective is no longer taken into account. Of course, the distinction between numerator and denominator — domain and range of the interpreting functions — is kept. Below some common parts of speech with their corresponding syntactic and semantic types.

determiner	$(s/(np\backslash s))/n$	$(e \to t) \to (e \to t) \to t$
intransitive verb	$np\backslash s$	$e \to t$
transitive verb	$(np\backslash s)/np$	$e \to e \to t$
reflexive pronoun	$((np\backslash s)/np)\backslash(np\backslash s)$	$(e \to e \to t) \to e \to t$
relative pronoun	$(n\backslash n)/(np\backslash s)$	$(e \to t) \to (e \to t) \to e \to t$

Formulas-as-types, proofs as programs Curry's basic insight was that one can see the functional types of type theory as logical implications, giving rise to a one-to-one correspondence between typed lambda terms and natural deduction proofs in positive intuitionistic logic. Translating Curry's 'formulas-as-types' idea to the categorial type logics we are discussing, we have to take the differences between intuitionistic logic and the grammatical resource logic into account. Below we give the slash rules of the base logic in natural deduction format, now taking term-decorated formulas as basic declarative units. Judgements take the form of sequents $\Gamma \vdash M : A$. The antecedent Γ is a structure with leaves $x_1 : A_1, \ldots, x_n : A_n$. The x_i are unique variables of type A'_i. The succedent is a term M of type A' with exactly the free variables $x_1, \ldots, x_n$, representing a program which given inputs $k_1 \in D_{A'_1} \ldots, k_n \in D_{A'_n}$ produces a value of type A' under the assignment that maps the variables x_i to the objects k_i. The x_i in other words are the parameters of the meaning assembly procedure; for these parameters we will substitute the actual lexical meaning recipes when we rewrite the leaves of the antecedent tree to terminal symbols (words). A derivation starts from axioms $x : A \vdash x : A$. The Elimination and Introduction rules have a version for the right and the left implication. On the meaning assembly level, this syntactic difference is ironed out, as we already saw that $(A/B)' = (B\backslash A)'$. As a consequence, we don't have the *isomorphic* (one-to-one) correspondence between terms and proofs of Curry's original program. But we do read off meaning assembly from the categorial derivation.

$$\frac{(\Gamma, x : B) \vdash M : A}{\Gamma \vdash \lambda x.M : A/B}\ I/ \qquad \frac{(x : B, \Gamma) \vdash M : A}{\Gamma \vdash \lambda x.M : B\backslash A}\ I\backslash$$

$$\frac{\Gamma \vdash M : A/B \quad \Delta \vdash N : B}{(\Gamma, \Delta) \vdash MN : A}\ E/ \qquad \frac{\Gamma \vdash N : B \quad \Delta \vdash M : A}{(\Gamma, \Delta) \vdash MN : A}\ E\backslash$$

A second difference between the programs/computations that can be obtained in intuitionistic implicational logic, and the recipes for meaning assembly associated with categorial derivations has to do with the resource management of assumptions in a derivation. In Curry's original program, the number of occurrences of assumptions (the 'multiplicity' of the logical resources) is not critical. One can make this style of resource management explicit in the form of structural rules of Contraction and Weakening, allowing for the duplication and waste of resources.

$$\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\, C \qquad \frac{\Gamma \vdash B}{\Gamma, A \vdash B}\, W$$

In contrast, the categorial type logics are resource sensitive systems where each assumption has to be used exactly once. We have the following correspondence between resource constraints and restrictions on the lambda terms coding derivations:

1. no empty antecedents: each subterm contains a free variable;
2. no Weakening: each λ operator binds a variable free in its scope;
3. no Contraction: each λ operator binds at most one occurrence of a variable in its scope.

Taking into account also word order and phrase structure (in the absense of Associativity and Commutativity), the slash introduction rules responsible for the λ operator can only reach the immediate daughters of a structural domain.

These constraints imposed by resource-sensitivity put severe limitations on the expressivity of the derivational semantics. There is an interesting division of labour here in natural language grammars between derivational and lexical semantics. The proof term associated with a derivation is a *uniform* instruction for meaning assembly that fully abstracts from the contribution of the particular lexical items on which it is built. At the level of the lexical meaning recipes, we do not impose linearity constraints. Below some examples of non-linearity; syntactic type assignment for these words was given above. The lexical term for the reflexive pronoun is a pure combinator: it identifies the first and second coordinate of a binary relation. The terms for relative pronouns or determiners have a double bind λ to compute the intersection of their two $(e \to t)$ arguments (noun and verb phrase), and to test the intersection for non-emptiness in the case of 'some'.

a, some (determiner)	$(e \to t) \to (e \to t) \to t$	$\lambda P \lambda Q.(\exists\ \lambda x.((P\ x) \wedge (Q\ x)))$
himself (reflexive pronoun)	$(e \to e \to t) \to e \to t$	$\lambda R \lambda x.((R\ x)\ x)$
that (relative pronoun)	$(e \to t) \to (e \to t) \to e \to t$	$\lambda P \lambda Q \lambda x.((P\ x) \wedge (Q\ x)))$

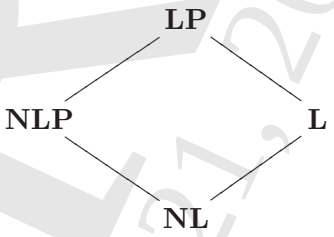
The interplay between lexical and derivational aspects of meaning assembly is illustrated with the natural deduction below. Using variables $x_1, \ldots, x_n$ for the leaves in left to right order, the proof term for this derivation is $((x_1\ x_2)\ (x_4\ x_3))$. Substituting the above lexical recipes for 'a' and 'himself' and non-logical constants $\mathbf{boy}^{e \to t}$ and $\mathbf{hurt}^{e \to e \to t}$, we obtain, after β conversion, $(\exists\ \lambda y.((\mathbf{boy}\ y) \wedge ((\mathbf{hurt}\ y)\ y)))$. Notice that the proof term reflects the derivational history (modulo directionality); after lexical substitution this transparency is lost. The full encapsulation of lexical semantics is one of the strong attractions of the categorial approach.

$$\dfrac{\dfrac{\dfrac{\text{a}}{(s/(np\backslash s))/n}\quad\dfrac{\text{boy}}{n}}{(\text{a, boy}) \vdash s/(np\backslash s)}\,(/E)\qquad\dfrac{\dfrac{\text{hurt}}{(np\backslash s)/np}\quad\dfrac{\text{himself}}{((np\backslash s)/np)\backslash(np\backslash s)}}{(\text{hurt, himself}) \vdash np\backslash s}\,(\backslash E)}{((\text{a, boy}), (\text{hurt, himself})) \vdash s}\,(/E)$$

(NOTE insert an example of derivational ambiguity: one structure, multiple proofs = multiple readings)

Structural variation

A second source of expressive limitations of the grammatical base logic is of a more structural nature. Consider situations where a word or phrase makes a uniform semantic contribution, but appears in contexts which the base logic cannot relate derivationally. In generative grammar, such situations are studied under the heading of 'displacement', a suggestive metaphor from our type-logical perspective. Displacement can be *overt* (as in the case of question words, relative pronouns and the like: elements that enter into a dependency with a 'gap' following at a potentially unbounded distance, cf. 'Who do you think that Mary likes (gap)?'), or *covert* (as in the case of quantifying expressions with the ability for non-local scope construal, cf. 'Alice thinks someone is cheating', which can be construed as 'there is a particular x such that Alice thinks x is cheating'). We have seen already that such expressions have higher-order types of the form $(A \to B) \to C$. The Curry-Howard interpretation then effectively dictates the uniformity of their contribution to the meaning assembly process as expressed by a term of the form $(M^{(A\to B)\to C}\ \lambda x^A.N^B)^C$, where the 'gap' is the λ bound hypothesis. What remains to be done, is to provide the fine-structure for this abstraction process, specifying which subterms of N^B are in fact 'visible' for the λ binder.



To work out this notion of visibility or structural accessibility, we introduce structural rules, in addition to the logical rules of the base logic studied so far. From the pure residuation logic, one obtains a hierarchy of categorial calculi by adding the structural rules of Associativity, Commutativity or both. For reasons of historical precedence, the system of Lambek (1958), with an associative composition operation, is known as **L**; the more fundamental system of Lambek (1961) as **NL**, i.e. the non-associative version of **L**. Addition of commutativity turns these into **LP** and **NLP**, respectively. For linguistic application, it is clear that *global* options of associativity and/or commutativity are too crude: they would entail that arbitrary changes in constituent structure and/or word order

cannot affect wellformedness of an expression. What is needed, is a *controlled* form of structural reasoning, anchored in lexical type assignment.

Control operators The strategy is familiar from linear logic: the type language is extended with a pair of unary operators ('modalities'). They are constants in their own right, with logical rules of use and of proof. In addition, they can provide controlled access to structural rules.

$$\mathcal{F} ::= \mathcal{A} \mid \Diamond\mathcal{F} \mid \Box\mathcal{F} \mid \mathcal{F}\backslash\mathcal{F} \mid \mathcal{F} \bullet \mathcal{F} \mid \mathcal{F}/\mathcal{F}$$

Consider the logical properties first. The truth conditions below characterize the control operators $\Diamond$ and $\Box$ as inverse duals with respect to a binary accessibility relation $R_\Diamond$. This interpretation turns them into a residuated pair, just like composition and the left and right slash operations, i.e. we have $\Diamond A \longrightarrow B$ iff $A \longrightarrow \Box B$ (Res).

$x \in V(\Diamond A)$ iff $\exists y.R_\Diamond xy$ and $y \in V(A)$ $\quad$ $x \in V(\Box A)$ iff $\forall y.R_\Diamond yx$ implies $y \in V(A)$

We saw that for composition and its residuals, completeness with respect to the frame semantics doesn't impose restrictions on the interpretation of the merge relation $R_\bullet$. Similarly, for $R_\Diamond$ in the pure residuation logic of $\Diamond, \Box$. This means that consequences of (Res) characterize grammatical invariants, in the sense indicated above. From (Res) one easily derives the fact that the control operators are monotonic ($A \longrightarrow B$ implies $\Diamond A \longrightarrow \Diamond B$ and $\Box A \longrightarrow \Box B$), and that their compositions satisfy $\Diamond\Box A \longrightarrow A \longrightarrow \Box\Diamond A$. These properties can be put to good use in refining lexical type assignment so that selectional dependencies are taken into account. Compare the effect of an assignment A/B versus $A/\Diamond\Box B$. The former will produce an expression of type A in composition both with expressions of type B and $\Diamond\Box B$, the latter only with the more specific of these two, $\Diamond\Box B$. An expression typed as $\Box\Diamond B$ will *resist* composition with either A/B or $A/\Diamond\Box B$.

For sequent presentation, the antecedent tree structures now have unary in addition to binary branching: $\mathcal{S} ::= \mathcal{F} \mid (\mathcal{S}) \mid (\mathcal{S}, \mathcal{S})$. The residuation pattern then gives rise to the following rules of use and proof. Cut elimination carries over straightforwardly to the extended system, and with it decidability and the subformula property.

$$\frac{\Gamma[(A)] \Rightarrow B}{\Gamma[\Diamond A] \Rightarrow B}\ \Diamond L \qquad \frac{\Gamma \Rightarrow A}{(\Gamma) \Rightarrow \Diamond A}\ \Diamond R$$

$$\frac{\Gamma[A] \Rightarrow B}{\Gamma[(\Box A)] \Rightarrow B}\ \Box L \qquad \frac{(\Gamma) \Rightarrow A}{\Gamma \Rightarrow \Box A}\ \Box R$$

Controlled structural rules Let us turn then to use of $\Diamond, \Box$ as control devices, providing restricted access to structural options that would be destructive in a global sense. Consider the role of the relative pronoun 'that' in the phrases below. The (a) example, where the gap hypothesis is in subject position, is derivable in the structurally-free base logic with the type-assignment given.

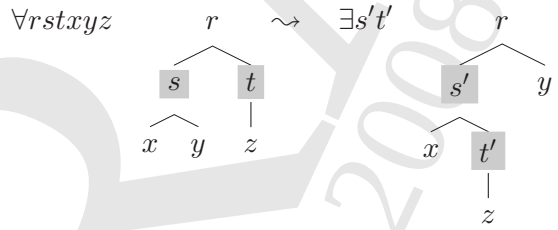
The (b) example might suggest that the gap in object position is accessible via rebracketing of $(np, ((np\backslash s)/np, np))$ under associativity. The (c) example shows that apart from rebracketing also reordering would be required to access a non-peripheral gap.

(a)	the paper that appeared today	$(n\backslash n)/(np\backslash s)$
(b)	the paper that John wrote	$(n\backslash n)/(s/np) + \text{Ass}$
(c)	the paper that John wrote today	$(n\backslash n)/(s/np) + \text{Ass,Com??}$

The controlled structural rules below allow the required restructuring and reordering only for $\Diamond$ marked resources. In combination with a type assignment $(n\backslash n)/(s/\Diamond\Box np)$ to the relative pronoun, they make the right branches of structural configurations accessible for gap introduction. As long as the gap subformula $\Diamond\Box np$ carries the licensing $\Diamond$, the structural rules are applicable; as soon as it has found the appropriate structural position where it is selected by the transitive verb, it can be used as a regular np, given $\Diamond\Box np \longrightarrow np$.

$$(P1)\quad (A \bullet B) \bullet \Diamond C \longrightarrow A \bullet (B \bullet \Diamond C) \qquad (P2)\quad (A \bullet B) \bullet \Diamond C \longrightarrow (A \bullet \Diamond C) \bullet B$$

Frame constraints, term assignment Whereas the structural interpretation of the pure residuation logic does not impose restrictions on the $R_\Diamond$ and $R_\bullet$ relations, completeness for structurally extended versions requires a frame constraint for each structural postulate. In the case of $(P2)$ above, the constraint guarantees that whenever we can connect root r to leaves x, y, z via internal nodes s, t, one can rewire root and leaves via internal nodes s', t'.



As for term assignment and meaning assembly, we have two options. The first is to treat $\Diamond, \Box$ purely as syntactic control devices. One then sets $(\Diamond A)' = (\Box A)' = A'$, and the inference rules affecting the modalities leave no trace in the term associated with a derivation. The second is to actually provide denotation domains $D_{\Diamond A}$, $D_{\Box A}$ for the new types, and to extend the term language accordingly. This is done in Wansing (2002), who develops a set-theoretic interpretation of minimal temporal intuitionistic logic. The temporal modalities of future possibility and past necessity are indistinguishable from the control operators $\Diamond, \Box$, prooftheoretically and as far as their relational interpretation is concerned, which in principle would make Wansing's approach a candidate for linguistic application.

Embedding translations A general theory of substructural communication in terms of $\Diamond, \Box$ is worked out in Kurtonina and Moortgat (1996). Let $\mathcal{L}$ and

$\mathcal{L}'$ be neighbours in the landscape of Figure NN. We have translations $\cdot^\natural$ from $\mathcal{F}(/, \bullet, \backslash)$ of $\mathcal{L}$ to $\mathcal{F}(\diamondsuit, \Box, /, \bullet, \backslash)$ of $\mathcal{L}'$ such that

$$\mathcal{L} \vdash A \longrightarrow B \quad \text{iff} \quad \mathcal{L}' \vdash A^\natural \longrightarrow B^\natural$$

The $\cdot^\natural$ translation decorates formulas of the source logic $\mathcal{L}$ with the control operators $\diamondsuit, \Box$. The modal decoration has two functions. In the case where the target logic $\mathcal{L}'$ is more discriminating than $\mathcal{L}$, it provides access to controlled versions of structural rules that are globally available in the source logic. This form of communication is familiar from the embedding theorems of linear logic, showing that no expressivity is lost by removing free duplication and deletion (Contraction/Weakening). The other direction of communication obtains when the target logic $\mathcal{L}'$ is less discriminating than $\mathcal{L}$. The modal decoration in this case blocks the applicability of structural rules that by default are freely available in the more liberal $\mathcal{L}$.

As an example, consider the grammatical base logic **NL** and its associative neighbour **L**. For $\mathcal{L} = \mathbf{NL}$ and $\mathcal{L}' = \mathbf{L}$, the $\cdot^\natural$ translation below affectively removes the conditions for applicability of the associativity postulate $A \bullet (B \bullet C) \longleftrightarrow (A \bullet B) \bullet C$ (Ass), restricting the set of theorems to those of **NL**. For $\mathcal{L} = \mathbf{L}$ and $\mathcal{L}' = \mathbf{NL}$, the $\cdot^\natural$ translation provides access to a controlled form of associativity ($\text{Ass}_\diamond$) $\diamondsuit(A \bullet \diamondsuit(B \bullet C)) \longleftrightarrow \diamondsuit(\diamondsuit(A \bullet B) \bullet C)$, the image of (Ass) under $\cdot^\natural$.

$$\begin{array}{rcl} p^\natural & = & p \quad (p \in \mathcal{A}) \\ (A \bullet B)^\natural & = & \diamondsuit(A^\natural \bullet B^\natural) \\ (A/B)^\natural & = & \Box A^\natural / B^\natural \\ (B\backslash A)^\natural & = & B^\natural \backslash \Box A^\natural \end{array}$$

Generative capacity, computational complexity The embedding results discussed above allow one to determine the Cartesian coordinates of a language in the logical space for diversity. Which regions of that space are actually populated by natural language grammars? In terms of the Chomsky hierarchy, recent work in a variety of frameworks has converged on the so-called mildly context-sensitive grammars: formalisms more expressive than context free, but strictly weaker than context-sensitive, and allowing polynomial parsing algorithms. The minimal system in the categorial hierarchy **NL** is strictly context-free and has a polynomial recognition problem, but, as we have seen, needs structural extensions. Such extensions are not innocent, as shown in Pentus (1993, 2003): whereas **L** remains strictly context-free, the addition of global associativity makes the derivability problem NP complete. Also for **LP**, coinciding with the multiplicative fragment of linear logic, we have NP completeness. Moreover, Van Benthem (1995) shows that **LP** recognizes the full permutation closure of context-free languages, a lack of structural discrimination making this system unsuited for actual grammar development. The situation with $\diamondsuit$ controlled structural rules is studied in Moot (2002), who establishes a PSPACE complexity ceiling for linear (for $\bullet$), non-expanding (for $\diamondsuit$) structural rules via simulation of lexicalized context-sensitive grammars.

The identification of tighter restrictions on allowable structure rules, leading to mildly context-sensitive expressivity, is an open problem.

For a grammatical framework assigning equal importance to syntax and semantics, strong generative capacity is more interesting that weak capacity. Tiede (1999, 2002) studies the natural deduction proof trees that form the skeleton for meaning assembly from a tree-automata perspective, arriving at a strong generative capacity hierarchy. The base logic **NL**, though strictly context-free at the string level, can assign *non-local* derivation trees, making it more expressive than context-free grammars in this respect. Normal form **NL** proof trees remain regular; the proof trees of the associative neighbour **L** can be non-regular, but do not extend beyond the expressivity of indexed grammars, generally considered to be an upper bound for the complexity of natural language grammars.

Variants, further reading

The material discussed in this section is covered in greater depth in Moortgat and Buszkowski's chapters in the Handbook of Logic and Language. Van Benthem (1995) is an indispensible monograph for the relations between categorial derivations, type theory and lambda calculus. and for discussion of the place of type-logical grammars within the general landscape of resource-sensitive logics. Morrill (1994) provides a detailed type-logical analysis of syntax and semantics for a rich fragment of English grammar, and situates the type-logical approach within Richard Montague's Universal Grammar framework. A versatile computational tool for categorial exploration is the the grammar development environment GRAIL of Moot (2002). The kernel is a general type-logical theorem prover based on proof nets and structural graph rewriting. The system is publicly available, or can be accessed online at `http://grail.let.uu.nl`. Bernardi (2002) and Vermaat (2006) are recent PhD theses studying syntactic and semantic aspects of cross-linguistic variation for a wide variety of languages.

This section has concentrated on the Lambek-style approach to type-logical deduction. The framework of Combinatory Categorial Grammar, studied by Steedman and his co-workers, takes its inspiration more from the Curry-Feys tradition of combinatory logic. The particular combinators used in CCG are not so much selected for completeness with respect to some structural model for the type-forming operations (such as the frame semantics introduced above) but for their computational efficiency, which places CCG among the mildly context-sensitive formalisms. Steedman (2000) is a good introduction to this line of work, whereas Baldrigde (2002) shows how one can fruitfully import the technique of lexically anchored modal control into the CCG framework.

Another variation elaborating on Curry's distinction between an abstract level of *tectogrammatical* organization and its concrete *phenogrammatical* realizations is the framework of Abstract Categorial Grammar (ACG, De Groote, Muskens). An abstract categorial grammar is a structure $(\Sigma_1, \Sigma_2, \mathcal{L}, s)$, where the Σ_i are higher-order linear signatures, the abstract vocabulary Σ_1 versus the object vocabulary Σ_2, $\mathcal{L}$ a mapping from the abstract to the object vocabulary, and s the distinguished type of the grammar. In this setting, one

can model the syntax-semantics interface in terms of the abstract versus object vocabulary distinction. But one can also study the composition of natural language *syntax* from the perspective of non-directional linear implicational types, using the canonical λ term encodings of strings and trees and operations on them discussed elsewhere in this book. Expressive power for this framework can be measured in terms of the maximal order of the constants in the abstract vocabulary and of the object types interpreting the atomic abstract types. A survey of results for the ensuing complexity hierarchy can be found in De Groote (2006). Ironically, Lambek categorial grammars themselves have eluded characterization in terms of ACGs so far. Whether one approaches natural language grammars from the top (non-directional linear implications at the **LP** level) or from the bottom (the structurally-free base logic NL) of the categorial hierarchy is to a certain extent a matter of taste, reflecting the choice, for the structural regime, between allowing everything except what is explicitly forbidden, or forbidding everything except what is explicitly allowed. The Kurtonina-Moortgat theory of structural control shows that the two viewpoints are feasible.

Chapter 7

Further Reading

Baxter, L. D. [1976]. *The Complexity of Unification*, Dissertation, University of Waterloo.

Bezem, M. A. [1986]. *Bar Recursion and Functionals of Finite Type*, Dissertation, University of Utrecht.

Bezem, M. A. [1988]. Equivalence of bar recursors in the theory of functionals of finite type, *Arch. Math. Logic* **27**(2), pp. 149–160.

Bruce, K. B., R. Di Cosmo and G. Longo [1992]. Provable isomorphisms of types, *Math. Structures Comput. Sci.* **2**(2), pp. 231–247. International Conference on Symbolic Computation (Zurich, 1990).

Damm, W. [1982]. The IO- and OI-hierarchies, *Theoret. Comput. Sci.* **20**(2), pp. 95–207.

Gandy, R. [1980]. An early proof of normalization by A. M. Turing, *in:* J. P. Seldin and J. R. Hindley (eds.), *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press, New York, pp. 453–457.

Goldfarb, W. D. [1981]. The undecidability of second-order unification, *Theoretical Computer Science* **13**, pp. 225–230.

Harrington, L. A., M. D. Morley, A. Ščedrov and S. G. Simpson (eds.) [1985]. *Harvey Friedman's research on the foundations of mathematics*, Studies in Logic and the Foundations of Mathematics 117, North-Holland Publishing Co., Amsterdam.

Howard, W. A. [1973]. Appendix: Hereditarily majorizable function of finite type, *Metamathematical investigation of intuitionistic arithmetic and analysis*, Springer, Berlin, pp. 454–461. Lecture Notes in Math., Vol. 344.

Howard, W.A. [1980]. The formulas-as-types notion of construction, *in:* J.P. Seldin and J.R. Hindley (eds.), *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press, New York, pp. 479–490.

Huet, G. P. [1975]. A unification algorithm for typed lambda-calculus, *Theoretical Computer Science* **1**, pp. 27–57.

Joly, Th. [1997]. Un plongement de la logique classique du 2nd ordre dans AF2, *C.R. de l'Academie des Sciences de Paris* **325**(1), pp. 1–4.

Joly, Th. [2000a]. *Codages, separabilité et représentation de fonctions en lambda-calcul simplement typé et dans d'autres systèmes de typés*, Dissertation, Paris VII.

Joly, Th. [2000b]. Non finitely generated types and lambda-terms combinatoric representation cost, *C.R. de l'Academie des Sciences de Paris* **331**(8), pp. 581–586.

Joly, Th. [2001a]. Constant time parallel computations in lambda-calculus, *TCS* **266**(1), pp. 975–985.

Joly, Th. [2001b]. The finitely generated types of the λ-calculus, *Proceedings of TLCA'01*, LNCS 2044, Springer, pp. 240–252.

Joly, Th. [2002]. About λ-definability and combinatoric generation of types, Thierry.Joly@pps.jussieu.fr.

Joly, Th. [2003]. Encoding of the halting problem into the monster type & applications, *Proceedings of TLCA'03*, LNCS 2701, Springer, pp. 153–166.

Joly, Th. [2005]. On λ-Definability I: the Fixed Model Problem and Generalizations of the Matching Problem., *Fundamenta Informaticae* **66**, pp. 1–17.

Jung, A. and J. Tiuryn [1993]. A new characterization of lambda definability, *Typed lambda calculi and applications (Utrecht, 1993)*, Lecture Notes in Comput. Sci. 664, Springer, Berlin, pp. 245–257.

Kreisel, G. [1959]. Interpretation of analysis by means of constructive functionals of finite types, *in:* A. Heyting (ed.), *Constructivity in Mathematics*, North-Holland, Amsterdam, pp. 101–128. Studies in Logic and the Foundations of Mathematics.

Läuchli, H. [1970]. An abstract notion of realizability for which intuitionistic predicate calculus is complete, *Intuitionism and Proof Theory (Proc. Conf., Buffalo, N.Y., 1968)*, North-Holland, Amsterdam, pp. 227–234.

Loader, R. [2001]. The undecidability of lambda definability, *Church memorial volume*, Reidel.

Loader, R. [2003]. Higher order β matching is undecidable, *Log. J. IGPL* **11**(1), pp. 51–68.

Padovani [1995]. On equivalence classes of interpolation equations, *in:* M. Dezani-Ciancaglini and G. Plotkin (eds.), *Proc.Int'l Conf. Typed Lambda Calculi and Applications (TLCA'95)*, LNCS 902, pp. 335–349.

Padovani, V. [1994]. Atomic matching is decidable, Unpublished manuscript.

Padovani, V. [1996a]. Decidability of all minimal models, *in:* S. Berardi and M. Coppo (eds.), *Proc. 3rd Int'l Workshop Types for Proofs and Programs (TYPES'95)*, LNCS 1158, pp. 201–215.

Padovani, V. [1996b]. *Filtrage d'ordre supérieur*, Dissertation, Université de Paris VII. Thèse de Doctorat.

Padovani, V. [2000]. Decidability of fourth order matching, *Mathematical Structures in Computer Science* **3**(10), pp. 361 – 372.

Padovani, V. [2001]. Retracts in simple types, *in:* S. Abramsky (ed.), *Proc.Int'l Conf. Typed Lambda Calculi and Applications (TLCA'01)*, LNCS 2044, Springer, pp. 376–384.

Plotkin, G. D. [1980]. Lambda-definability in the full type hierarchy, *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, Academic Press, London, pp. 363–373.

Pollack, R. [1994]. Closure under alpha-conversion, *in:* H. P. Barendregt and T. Nipkov (eds.), *Types for proofs and programs (Nijmegen, 1993)*, Lecture Notes in Comput. Sci. 806, Springer, Berlin, pp. 313–332.

Statman, R. [1985]. Equality between functionals revisited, in *Harrington et al. [1985]*, North-Holland, Amsterdam, pp. 331–338.

Statman, R. [1997]. On Cartesian monoids, *in:* D. van Dalen and M. Bezem (eds.), *Computer science logic (Utrecht, 1996)*, Lecture Notes in Comput. Sci. 1258, Springer, Berlin, pp. 446–459.

Statman, R. [2000]. Church's lambda delta calculus, *Logic for programming and automated reasoning (Reunion Island, 2000)*, Lecture Notes in Comput. Sci. 1955, Springer, Berlin, pp. 293–307.

Statman, R. [2004]. On the λY calculus, *Ann. Pure Appl. Logic* **130**(1-3), pp. 325–337.

Tait, W. W. [1967]. Intensional interpretations of functionals of finite type. I, *J. Symbolic Logic* **32**, pp. 198–212.

de Vrijer, R. [1987]. Exactly estimating functionals and strong normalization, *Indagationes Mathematicae* **49**, pp. 479–493.

Zaionc, Marek [1987]. Word operation definable in the typed λ-calculus, *Theoret. Comput. Sci.* **52**(1-2), pp. 1–14.

Zaionc, Marek [1989]. On the λ definable higher order Boolean operations, *Fund. Inform.* **12**(2), pp. 181–189.

Zaionc, Marek [1990]. A characterization of lambda definable tree operations, *Inform. and Comput.* **89**(1), pp. 35–46.

Zaionc, Marek [1991]. λ-definability on free algebras, *Ann. Pure Appl. Logic* **51**(3), pp. 279–300.

@ T.Joly The ksi-rule of the untyped lambda-calculus is not finitely axiomatizable, preprint (submitted to JFP)

@ T.Joly Normalization by reducibility and Bohm theorems for the typed lambda-calculus, preprint

@ T.Joly About lambda-definability and combinatoric generation of types preprint

Part II

Recursive Types $\lambda^{\mathcal{A}}_{=}$ 21.2.2008:897

Survey Part II

In part I we dealt with the freely generated types $\mathbb{T}^{\mathbb{A}}$.

8. Basic Concepts

8.1 type algebras $\mathcal{A}$ satisfying extra equations $\mathcal{E}$.
$\mathcal{A}/\mathcal{E} \models \mathcal{E}$.

8.2 type assignment a la Curry and Church is defined.
8.3 $\mathbb{T}$ gives $\mathbb{T}/\mathcal{E}$
$\mathbb{T}(\vec{X})/\mathcal{R} = \mathbb{T}[\mathcal{R}]$.
$\mathbb{T}_\mu$ (algebraic closure:all possible simultaneous recursion sytems already have a solution there).

8.4 Algebras and coalgebras.
8.5 Infinite trees as type structure.
8.6 $\mathbb{T}_\infty$ We define $=^*$ if two types have the same tree.
$\mathbb{T}^*_\mu$ and $\mathbb{T}[\mathcal{R}]^*$.

8.7 Exercises

9. Properties of Recursive Types

In 9.1 every simultaneous recursion can be solved in $\mathcal{A}_\mu$. [We need the other direction: For every type $A \in \mathbb{T}_\mu$ there is an $\mathcal{R}$ and an $A' \in \mathbb{T}[\mathcal{R}]$ having exactly the same terms. There should be a better formulation.] In 9.2 we show decidability of $=$ on mu types. We show $\mathcal{A}_\mu$ is invertible. We also have the axiomatization and decidability of $=^*$. In 9.3 we show the decidabilty of $=$ and invertibility in $\mathbb{T}[\mathcal{R}]$. We show that solvability of $\mathcal{E}$ in $\mathbb{T}[\mathcal{R}]$ is decidable.

10. Properties of Terms with Types

In 10.1 we show that for $\lambda^{\mathcal{A}}_{=}$ subject reduction holds iff $\mathcal{A}$ is invertible. In 10.2 the PTS theorem is proved for $\mathcal{A}$ being $\mathbb{T}[\mathcal{R}]$ and $\mathcal{A}_m u$. In 10.3 we do Mendler's theorem: A is 'positive' iff for all M in A one has M is SN.

11. Models

11.1 Scott semantics for $\lambda_{\rightarrow}$-type as subsets of λ-models extended to types of $\mathbb{T}_{\mu}$ and $\mathbb{T}[\mathcal{R}]$ models with suitable approximation.

Prop. For all A, B $[\![A]\!] = [\![B]\!] \iff A =^{*} B$.

Prop. [Soundness] $\Gamma \vdash M : A \Rightarrow \Gamma \models M : A$.

Prop. [Completeness] $\Gamma \models M : A \iff \Gamma \vdash_{\text{Approx}} M : A$.

Prop. $\vdash M : A$, A special $\Rightarrow$ M unsolvable. $\mu t.t$ belongs to the special types.

11.2 Semantics for Church version of $\lambda_{=}^{\mathcal{A}}$, notably $\mathbb{T}_{\mu}$.

For types A one has $[\![A]\!] \in \mathbf{CPO}$.
If $\vdash M : A$, then $[\![M]\!] \in [\![A]\!]$.
Prop. Genericity for terms in special types.
Prop. $\{M = N \mid M, N : A \text{ special}\}$ is consistent.
[Closure Semantics for fold and unfold.(?)]

12. Applications

12.1 PL with recursive types. (Move to further reading.)

12.2 $\leq$: trans, refl, $\eta^{=}$.
$A \leq B \iff [\![A]\!] \subseteq [\![B]\!]$.
From $\lambda_{\rightarrow}$ to $\lambda_{\leq\mu}$ via $\lambda_{\leq}$ and via λ_{μ}.
$\leq$ is decidable.
$\lambda_{\rightarrow}$—$\lambda_{=}$—$\lambda_{\leq}$—$\lambda_{\cap}$.

13. Further Reading

21.2.2008:897

The simple types $\lambda_{\rightarrow}$ of Part I were *freely* generated from the type atoms $\mathbb{A}$. This means that there are no identifications like $\alpha = \alpha{\rightarrow}\beta$ or $o{\rightarrow}o = (o{\rightarrow}o){\rightarrow}o$.

With the recursive types of this part the situation changes. Now, one allows extra identifications between types; for this purpose one considers types modulo a congruence determined by some set $\mathcal{E}$ of equations between types. Following a suggestion of Scott [1975a], this can be described by considering *type algebras*, consisting of a set $\mathcal{A}$ on which a binary operation $\rightarrow$ is defined (one then can have in such structures e.g. $a = a{\rightarrow}b$). Another way of obtaining a type algebra is to add the fixed-point operator μ as a syntactic type constructor, together with a canonical congruence $\sim$ on the resulting terms. Given a type $A[\alpha]$ in which α may occur, the type $\mu\alpha.A[\alpha]$ has as intended meaning a solution X of the equation $X = A[X]$. For example for $A \equiv \mu\alpha.\alpha{\rightarrow}B$ one has $A \sim A{\rightarrow}B$, which will become an equality in the type algebra.

We mainly study systems with only $\rightarrow$ as type constructor, since this restriction focuses on the most interesting phenomena. For applications sometimes other constructors, like $+$ and $\times$ are needed; these can be added easily. Examples of recursive specifications of types using such extra type constructors are used in programming languages. One can, for example, define the type of lists of objects of type A by the equation

$$\texttt{list} = 1 + (A \times \texttt{list}).$$

This needs the presence of a type constant 1 for the one element type (intended to contain `nil`), and type constructors $+$ for disjoint union of types and $\times$ for cartesian product.

Using type algebras one can define a notion of type assignment to lambda terms, that is stronger than the one using simple types. In a type algebra in which one has a type $C = C \rightarrow A$ one can give the term $\lambda x.xx$ the type C. Indeed, the following is a derivation of this.

$$\dfrac{\dfrac{\dfrac{x\ C \vdash x : C}{x\ C \vdash x\ C{\rightarrow}A}\ {\scriptstyle C=C\rightarrow A} \qquad x\ C \vdash x : C}{x\ C \vdash xx : A}}{\dfrac{\vdash \lambda x.xx : C \rightarrow A}{\vdash \lambda x.xx : C}\ {\scriptstyle C\rightarrow A=C}}$$

Another example is the fixed-point operator $\mathsf{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ that now will have as type $(A \rightarrow A) \rightarrow A$ for all types A such that there exists a type C satisfying $C = C \rightarrow A$.

Several properties of the simple type systems are valid for the recursive type systems. For example Subject Reduction and the decidability of type assignment. Some other properties are lost, for example Strong Normalization of typable terms and the canonical connection with logic in the form of the formulas-as-type interpretation. By making some natural assumption on the type algebras the Strong Normalization property is preserved.

Chapter 8

Basic Concepts 21.2.2008:897

basic.rec.types

In the present Part II of this book we will again consider the set of types $\mathbb{T} = \mathbb{T}^{\mathbb{A}}$ freely generated from atomic types $\mathbb{A}$ and the type constructor $\rightarrow$. (Sometimes other type constructors, including constants, will be allowed.) But now the freely generated types will be 'bent together' by making identifications like $A = A \rightarrow B$. This is done by considering types modulo a congruence relation $\approx$ (an equivalence relation preserved by $\rightarrow$). Then one can define the operation $\rightarrow$ on the equivalence classes. As suggested by Scott [1975a] this can be described by considering type-algebras consisting of a set with a binary operation $\rightsquigarrow$ on it. In such structures one can have for example $a = a \rightsquigarrow b$.

8.1. Type-algebras and type assignment

sectionone

8.1.1. Definition. **Type-Algebras**

(i) A *type-algebra* is a structure

$$\mathcal{A} = \langle |\mathcal{A}|, \rightsquigarrow_{\mathcal{A}} \rangle,$$

where $\rightsquigarrow_{\mathcal{A}}$ is any binary operation on $|\mathcal{A}|$.

(ii) If $\mathcal{A}$ is a type-algebra we write $a \in \mathcal{A}$ for $a \in |\mathcal{A}|$. In the same style, if there is little danger of confusion we often write $\mathcal{A}$ for $|\mathcal{A}|$.

(iii) The type-algebra $\mathbb{T}^{\mathbb{A}} = \langle \mathbb{T}^{\mathbb{A}}, \rightarrow \rangle$ is called the *free type-algebra over* $\mathbb{A}$. This terminology will be justified in 8.2.1 below.

(iv) We will use $\alpha,\ \beta,\ \ldots$ to denote arbitrary elements of $\mathbb{A}$ and $A, B, C, \ldots$ to range over $\mathbb{T}^{\mathbb{A}}$. On the other hand $a, b, c, \ldots$ range over a type algebra $\mathcal{A}$.

Quotients and syntactic type-algebras and morphisms

8.1.2. Definition. **Quotients**

(i) A *congruence* on a type algebra $\mathcal{A} = \langle |\mathcal{A}|, \rightsquigarrow \rangle$ is an equivalence relation $\approx$ on $|\mathcal{A}|$ such that for all $a, b, a', b' \in \mathcal{A}$ one has

$$a \approx a' \ \& \ b \approx b' \ \Rightarrow \ (a \rightsquigarrow b) \approx (a' \rightsquigarrow b').$$

(ii) In this situation define for $a \in \mathcal{A}$ its *equivalence class*, notation $[a]_\approx$, by

$$[a]_\approx = \{b \in \mathcal{A} \mid a \approx b\}.$$

(iii) The *quotient type algebra* of $\mathcal{A}$ under $\approx$, notation $\mathcal{A}/\approx$, is defined by

$$\langle \mathcal{A}/\approx, \rightsquigarrow_\approx \rangle,$$

where

$$\begin{array}{rcl} \mathcal{A}/\approx & = & \{[a]_\approx \mid a \in \mathcal{A}\} \\ [a]_\approx \rightsquigarrow_\approx [b]_\approx & = & [a \rightsquigarrow b]_\approx. \end{array}$$

Since $\approx$ is a congruence, the operation $\rightsquigarrow_\approx$ is well-defined.

A special place among type-algebras is taken by quotients of the free type-algebras modulo some congruence. In fact, in Proposition 8.1.10 we shall see that every type algebra has this form, up to isomorphism.

8.1.3. Definition. **typeA** Let $\mathbb{T} = \mathbb{T}^{\mathbb{A}}$. A *syntactic* type-algebra over $\mathbb{A}$ is of the form

$$\mathcal{A} = \langle \mathbb{T}^{\mathbb{A}}/\approx, \rightarrow_\approx \rangle,$$

where $\approx$ is a congruence on $\langle \mathbb{T}^{\mathbb{A}}, \rightarrow \rangle$.

8.1.4. Remark. **Identify1**
(i) A syntactic type-algebra $\mathcal{A} = \langle \mathbb{T}/\approx, \rightarrow_\approx \rangle$ will simply be denoted by $\mathcal{A} = \mathbb{T}/\approx$ (no confusion can arise as $\rightarrow_\approx$ is fully determined by $\approx$).

(ii) We often simply write A for $[A]_\approx$ (for example in "$A \in \mathbb{T}/\approx$"), thereby identifying $\mathbb{T}/\approx$ with $\mathbb{T}$ and $\rightarrow_\approx$ with $\rightarrow$.

(iii) The free type-algebra over $\mathbb{A}$ is also syntactic, in fact it is the same as $\mathbb{T}^{\mathbb{A}}/=$, where $=$ is the ordinary equality relation on $\mathbb{T}^{\mathbb{A}}$. This algebra will henceforth be denoted simply by $\mathbb{T}^{\mathbb{A}}$.

8.1.5. Definition. Let $\mathcal{A}$ and $\mathcal{B}$ be type-algebras.

(i) A map $h : |\mathcal{A}| \rightarrow |\mathcal{B}|$ is called a *morphism* between $\mathcal{A}$ and $\mathcal{B}$, notation[1] $h : \mathcal{A} \rightarrow \mathcal{B}$, iff for all $a, b \in \mathcal{A}$ one has

$$h(a \rightsquigarrow_{\mathcal{A}} b) = h(a) \rightsquigarrow_{\mathcal{B}} h(b).$$

(ii) An *isomorphism* is a morphism $h : \mathcal{A} \rightarrow \mathcal{B}$ that is injective and surjective. Note that in this case the inverse map h^{-1} is also a morphism. $\mathcal{A}$ and $\mathcal{B}$ are called *isomorphic*, notation $\mathcal{A} \cong \mathcal{B}$, if there is an isomorphism $h : \mathcal{A} \rightarrow \mathcal{B}$.

(iii) We say that $\mathcal{A}$ is *embeddable* in $\mathcal{B}$, notation $\mathcal{A} \hookrightarrow \mathcal{B}$, if there is an injective morphism $i : \mathcal{A} \rightarrow \mathcal{B}$. In this case we also write $i : \mathcal{A} \hookrightarrow \mathcal{B}$.

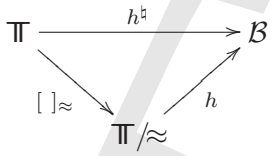
8.1.6. Remark. **syntactic-hom-rem**

[1] This is an overloading of the symbol "$\rightarrow$" with little danger of confusion.

(i) Let $\mathcal{A} = \mathbb{T}/\approx$ be a syntactic type algebra and let $\langle \mathcal{B}, \rightsquigarrow \rangle$ be any type algebra. Morphisms $h : \mathbb{T}/\approx \to \mathcal{B}$ correspond exactly to morphisms $h^\natural : \mathbb{T} \to \mathcal{B}$ such that for all $A, B \in \mathbb{T}$

$$\begin{array}{rcl} A \approx B & \Rightarrow & h^\natural(A) = h^\natural(B); \\ h^\natural(A) & = & h([A]). \end{array}$$

In picture



We call such a map $h^\natural$ a *syntactic* morphism and often identify h and $h^\natural$.

(ii) If $\mathbb{T} = \mathbb{T}^{\mathbb{A}}$ for some set $\mathbb{A}$ of atomic types then $h^\natural$ is uniquely determined by its restriction $h^\natural \restriction \mathbb{A}$.

(iii) If moreover $\mathcal{B} = \mathbb{T}'/\approx'$ then $h^\natural(A) = [B]_{\approx'}$ for some $B \in \mathbb{T}'$. Identifying B with its equivalence class in $\approx'$, we can write simply $h^\natural(A) = B$. The first condition in (i) then becomes $A \approx B \;\Rightarrow\; h^\natural(A) \approx' h^\natural(B)$.

Constructing type-algebras by equating elements

The following construction makes extra identifications in a given type algebra. It will serve in the next subsection as a tool to build a type-algebra satisfying a given set of equations. What we do here is just bending together elements (like considering numbers modulo p). In the next subsection we also extend type algebras in order to get new elements that will be cast with a special role (like extending the real numbers with an element X, obtaining the ring $\mathbb{R}[X]$ and then bending $X^2 = -1$ to create the imaginary number i).

8.1.7. Definition. **WeakCeq** Let $\mathcal{A}$ be a type-algebra.

(i) An *equation over* $\mathcal{A}$ is of the form $a = b$, with $a, b \in \mathcal{A}$. Let $\mathcal{E}$ denote a (possibly infinite) set of equations over $\mathcal{A}$.

(ii) The least congruence relation on $\mathcal{A}$ extending $\mathcal{E}$, notation $=_{\mathcal{E}}$, is defined by the following axioms and rules, where a, a', b, b', c range over $\mathcal{A}$.

(axiom)	$\vdash a = b$ if $(a = b) \in \mathcal{E}$
(refl)	$\vdash a = a$
(symm)	$\dfrac{\vdash a = b}{\vdash b = a}$
(trans)	$\dfrac{\vdash a = b \quad \vdash b = c}{\vdash a = c}$
($\to$-cong)	$\dfrac{\vdash a = a' \quad \vdash b = b'}{\vdash a \rightsquigarrow b = a' \rightsquigarrow b'}$

We write $\vdash_{\mathcal{E}} a = b$ or $a =_{\mathcal{E}} b$ if there is a proof of $\vdash a = b$ by the above axioms and rules. If $\mathcal{E}'$ is another set of equations over $\mathcal{A}$ we write

$$\vdash_{\mathcal{E}} \mathcal{E}'$$

if $\vdash_{\mathcal{E}} a = b$ for all $a = b \in \mathcal{E}'$.

(iii) The *quotient type-algebra* $\mathcal{A}$ *modulo* $\mathcal{E}$, notation $\mathcal{A}/\mathcal{E}$ is defined as

$$\mathcal{A}/\mathcal{E} := \mathcal{A}/=_{\mathcal{E}}.$$

8.1.8. Remark. (i) $\mathcal{E}$ is a congruence relation on $\mathcal{A}$ iff $=_{\mathcal{E}}$ coincides with $\mathcal{E}$.

(ii) The definition of a quotient type-algebra $\mathcal{A}/\approx$ is a particular case of the construction 8.1.7(ii) since by (i) one has $\approx = (=_{\approx})$. In most cases a syntactic type-algebra is given by $\mathbb{T}/\mathcal{E}$ where $\mathcal{E}$ is a set of equations between elements of the free type-algebra $\mathbb{T}$.

The following easy result motivates the definition of quotient type-algebras.

8.1.9. Proposition. **ExtTa-prop** *Let $\mathcal{A}$ be a type-algebra and $\mathcal{E}$ a set of equations over $\mathcal{A}$. Then $\mathcal{A}/\mathcal{E} \models \mathcal{E}$, i.e. all the equations in $\mathcal{E}$ are valid in the type algebra $\mathcal{A}/\mathcal{E}$.*

We now show that every type-algebra can be considered as syntactic one.

8.1.10. Proposition. **ta-synt** *Every type-algebra is isomorphic to a syntactic one.*

Proof. Given $\mathcal{A} = \langle |\mathcal{A}|, \rightsquigarrow \rangle$, take $\mathbb{A} = \{\overline{a} \mid a \in |\mathcal{A}|\}$ and

$$\mathcal{E} = \{\overline{a} \to \overline{b} = \overline{a \rightsquigarrow b} \mid a, b \in \mathbb{A}\}.$$

Then $\mathcal{A}$ is isomorphic to $\mathbb{T}^{\mathbb{A}}/\mathcal{E}$ via the isomorphism $a \mapsto [\overline{a}]_{\mathcal{E}}$. ■

We are now ready to set up formal systems for assigning recursive types to λ-terms. We will focus our presentation mainly on type inference systems à la Curry, but for any of them a corresponding typed calculus à la Church can be defined.

The systems à la Curry

CurrySystems

The formal rules to assign types to λ-terms are defined like in Chapter 1.2, but here the types are elements in an arbitrary type algebra $\mathcal{A}$. This means that the judgments of the systems are of the following shape.

$$\Gamma \vdash_{\lambda\mathcal{A}} M : a,$$

where one has $a \in \mathcal{A}$ and Γ (a *basis*) is a set of statements of the shape $x\,a$, where x is a term variable and $a \in \mathcal{A}$. As before, the subjects in $\Gamma = \{x_1\,a_1, \ldots, x_n\,a_n\}$ should be distinct, i.e. $x_i = x_j \;\Rightarrow\; i = j$.

8.1.11. DEFINITION. **CurrysystemR CurrySystemTA**

Let $\mathcal{A}$ be a Type Algebra and $a, b \in \mathcal{A}$. Then the system $(\lambda\mathcal{A})$ is defined by the following rules.

$$\begin{array}{ll} \text{(axiom)} & \Gamma \vdash x : a \qquad \text{if } (x\ a) \in \Gamma \\ (\to\text{E}) & \dfrac{\Gamma \vdash M : a \to b \qquad \Gamma \vdash N : a}{\Gamma \vdash (MN) : b} \\ (\to\text{I}) & \dfrac{\Gamma, x\ a \vdash M : b}{\Gamma \vdash (\lambda x.M) : (a \to b)} \end{array}$$

We write $\Gamma \vdash_{\lambda\mathcal{A}} M : a$ if $\Gamma \vdash M : a$ can be derived from the above rules.

Although the axioms and rules are the same as for $\lambda_{\to}$, the fact that we work with type algebras makes a dramatic difference. There are examples of type assignment in $\lambda\mathcal{A}$ to terms which have no type in the simple type assignment system $\lambda_{\to}$.

8.1.12. PROPOSITION. **basictypings** *Let $\mathcal{A}$ be a type algebra and let $a, b \in \mathcal{A}$ be such that $b = (b \to a)$. Then*

(i) $\vdash_{\lambda\mathcal{A}} (\lambda x.xx) : b$.
(ii) $\vdash_{\lambda\mathcal{A}} \mathbf{\Omega} : a$, *where* $\mathbf{\Omega} = (\lambda x.xx)(\lambda x.xx)$.
(iii) $\vdash_{\lambda\mathcal{A}} \mathsf{Y} : (a \to a) \to a$,

where $\mathsf{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ *is the fixed-point operator.*

PROOF. (i) The following is a deduction of $\vdash_{\lambda\mathcal{A}} (\lambda x.xx) : b$.

$$\dfrac{\dfrac{\dfrac{x\ b \vdash x : b \quad \dfrac{x\ b \vdash x : b}{x\ b \vdash x : b \to a}\ (\to\text{E})}{x\ b \vdash xx : a}\ (\to\text{I})}{\vdash (\lambda x.xx) : (b \to a)}\ \text{, since } b = (b \to a),}{\vdash (\lambda x.xx) : b}\ \text{, since } (b \to a) = b,$$

(ii) As $\vdash (\lambda x.xx) : b$, we also have $\vdash (\lambda x.xx) : (b \to a)$, since $b = b \to a$. Therefore $\vdash (\lambda x.xx)(\lambda x.xx) : a$.

(iii) We can prove $\vdash \mathsf{Y} : (\alpha \to \alpha) \to \alpha$ in $(\lambda\mathcal{A})$ in the following way. First modify the deduction constructed in (i) to obtain $f\ a \to a \vdash \lambda x.f(xx) : b$. Since $b = b \to a$ we have as in (ii) by rule $(\to\text{E})$

$$f : a \to a \vdash (\lambda x.f(xx))(\lambda x.f(xx)) : a$$

from which we get

$$\vdash_{\lambda\mathcal{A}} \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)) : (a \to a) \to a. \blacksquare$$

A type b satisfying $b = (b \to a)$ can be easily obtained by working modulo $=_{\mathcal{E}}$, with $\mathcal{E} = \{b = b \to a\}$.

8.1.13. DEFINITION. If $\mathcal{A} = \mathbb{T}/{\approx}$ is a syntactic type algebra, then we write

$$x_1\, A_1, \ldots, x_n\, A_n \vdash_{\approx} M : A$$

for

$$x_1\, [A_1], \ldots, x_n\, [A_n] \vdash_{\lambda(\mathbb{T}/\approx)} M : [A].$$

We will present systems often in the following form.

8.1.14. PROPOSITION. *The assignment $\vdash_{\approx}$ can be axiomatized by the following axioms and rules.*

$$\begin{array}{ll} \text{(axiom)} & \Gamma \vdash x : A \qquad \textit{if } (x\, A) \in \Gamma \\[2ex] (\to\text{E}) & \dfrac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash (MN) : B} \\[2ex] (\to\text{I}) & \dfrac{\Gamma, x\, A \vdash M : B}{\Gamma \vdash (\lambda x.M) : (A \to B)} \\[2ex] \text{(equal)} & \dfrac{\Gamma \vdash M : A \qquad A \approx B}{\Gamma \vdash M : B} \end{array}$$

where now A, B range over $\mathbb{T}$ and Γ is of the form $\{x_1\, A_1, \ldots, x_n\, A_n\}$, $\vec{A} \in \mathbb{T}$.

PROOF. Easy. ■

Systems of type assignment can be related via the notion of type algebra morphism. The following property can easily be proved by induction on derivations.

8.1.15. LEMMA. **SystemHom** *Let $h : \mathcal{A} \to \mathcal{B}$ be a type algebra morphism. Then for $\Gamma = \{x_1\, A_1, \ldots, x_n\, A_n\}$*

$$\Gamma \vdash_{\lambda\mathcal{A}} M : A \;\Rightarrow\; h(\Gamma) \vdash_{\lambda\mathcal{B}} M : h(A),$$

where $h(\Gamma) = \{x_1\, h(A_1), \ldots, x_n\, h(A_n)\}$.

In Chapter 10 we will prove the following properties of type assignment.

1. A type assignment system $\lambda\mathcal{A}$ has the subject reduction property for β-reduction iff $\mathcal{A}$ is invertible: $a \to b = a' \to b' \;\Rightarrow\; a = a'$ & $b = b'$, for all $a, a', b, b' \in \mathcal{A}$.

2. For the type assignment introduced in this Section there is a notion of 'principal type scheme' with properties similar to that of the basic system $\lambda_{\to}$. As a consequence of this, most questions about typing λ-terms in given type algebras are decidable.

3. There is a simple characterization of the collection of type algebras for which a strong normalization theorem holds. It is decidable whether a given λ-term can be typed in them.

Explicitly typed systems (à la Church)

ChurchFormulation

Explicitly typed versions (*à la* Church) of λ-calculus with recursive types can also be defined as for the simpy typed lambda calculus in Part I, where now, as in the previous section, the types are from a (syntactic) type algebra.

In the explicitly typed systems each term is defined as a member of a specific type, which is uniquely determined by the term itself. In particular, as in Section 1.4, we assume now that each variable is coupled with a unique type which is part of it. We also assume without loss of generality that all terms are well named according to Definition 1.2.4.

8.1.16. DEFINITION. **ChurchSystems** Let $\mathcal{A} = \mathbb{T}/\approx$ be a syntactic type algebra and $A, B \in \mathcal{A}$. The well typed terms of the system ($\lambda\mathcal{A}$-Ch), notation $\Lambda^{\mathrm{Ch}}_{\mathcal{A}}(A)$ for each type A, are defined by the following term formation rules:

$$\begin{array}{rcl} & & x^A \in \Lambda^{\mathrm{Ch}}_{\mathcal{A}}(A); \\ M \in \Lambda^{\mathrm{Ch}}_{\mathcal{A}}(A{\to}B), N \in \Lambda^{\mathrm{Ch}}_{\mathcal{A}}(A) & \Rightarrow & (MN) \in \Lambda^{\mathrm{Ch}}_{\mathcal{A}}(B); \\ M \in \Lambda^{\mathrm{Ch}}_{\mathcal{A}}(B) & \Rightarrow & (\lambda x^A.M) \in \Lambda^{\mathrm{Ch}}_{\mathcal{A}}(A{\to}B); \\ M \in \Lambda^{\mathrm{Ch}}_{\mathcal{A}}(A) \text{ and } A \approx B & \Rightarrow & M \in \Lambda^{\mathrm{Ch}}_{\mathcal{A}}(B). \end{array}$$

A formulation of the system in the "de Bruijn" style could be possible as well. The "de Bruijn" formulation is indeed the most widely used to denote explicitly typed systems in the literature, especially in the field of Computer Science. The "Church" style, on the other hand, emphasizes the distinction between explicitly and implicitly typed systems, and is more suitable for the study of models in section 11. Given a syntactic type algebra $\mathcal{A} = \mathbb{T}/\approx$ the formulation of the system ($\lambda\mathcal{A}$-dB) in the de Bruijn style is given by the following rules.

$$\begin{array}{ll} \text{(axiom)} & \Gamma \vdash x : A \qquad \text{if } (x\ A) \in \Gamma \\ \text{(}{\to}\text{E)} & \dfrac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B} \\ \text{(}{\to}\text{I)} & \dfrac{\Gamma, x\ A \vdash M : B}{\Gamma \vdash (\lambda x\ A.M) : A \to B} \\ \text{(equiv)} & \dfrac{\Gamma \vdash M : A \qquad A \approx B}{\Gamma \vdash M : B} \end{array}$$

the Church and de Bruijn formulations, holds also for recursive types. The system $\lambda\mathcal{A}$-Ch, for instance, is the counterpart à la de Church of the system $\lambda\mathcal{A}$, in this sense: for each deduction of a typing $\Gamma \vdash_{\lambda\mathcal{A}} M : A$ there is a corresponding explicitly typed term M' such that $\Gamma \vdash_{\lambda\mathcal{A}\text{-Ch}} M' : A$. This M' is obtained from M by adding types in M for the bound variables. In particular there is a well typed term of $\lambda\mathcal{A}$-Ch for each of the type deductions shown

in Proposition 8.1.12 (see Exercise 8.7.14). Conversely given a deduction of $\Gamma \vdash_{\lambda\mathcal{A}\text{-dB}} M : A$ we can build a corresponding deduction in $(\lambda\mathcal{A})$ by erasing all type informations in rule $(\to$-cong).

In an explicitly typed calculus we expect that a term completely codes the deduction of its type. Now any type algebra introduced in the previous sections is defined via a notion of equivalence on types which is used, in general, to prove that a term is well typed. But in the systems $(\lambda\mathcal{A}$-Ch) the way in which type equivalences are proved is not coded in the term. To do this we must introduce new terms representing equivalence proofs.

Take for instance $\mathcal{A} = \mathbb{T}/=_{\mathcal{E}}$ with $\mathcal{E} = \{\beta = \beta \to \alpha\}$. We skip the rule

$$M \in \Lambda^{\text{ch}}_{\mathcal{A}}(A) \;\&\; A \approx B \;\Rightarrow\; M \in \Lambda^{\text{ch}}_{\mathcal{A}}(B).$$

Instead we introduce two constants, usually called *coercions*. The first is fold_β of type $(\beta \to \alpha) \to \beta$ and the second unfold_β of type $\beta \to (\beta \to \alpha)$, which represent the two ways in which the equation $\beta = \beta \to \alpha$ can be applied. The isomorphism between β and $\beta \to \alpha$ can be expressed at the computational level by introducing the following contraction rules

$$\begin{array}{lrcl} (R^{\text{uf}}_{\mathcal{E}}) & \mathit{unfold}_\beta(\mathit{fold}_\beta\ M^{\beta\to\alpha}) & \to & M^{\beta\to\alpha}; \\ (R^{\text{fu}}_{\mathcal{E}}) & \mathit{fold}_\beta(\mathit{unfold}_\beta\ M^{\beta}) & \to & M^{\beta}. \end{array}$$

We denote such a system by $(\lambda(\mathbb{T}/\mathcal{E})\text{-Ch}_0)$.

8.1.17. Example. The following term is the version of $\lambda x.xx$ in the system $(\lambda(\mathbb{T}/\mathcal{E})\text{-Ch}_0)$ above.

$$\mathit{fold}_\beta(\lambda x^\beta.((\mathit{unfold}_\beta x^{\beta)}\ x^\beta)) \in \Lambda^{\text{Ch}_0}_{\mathbb{T}/=_{\mathcal{E}}}(\beta)$$

Note that a system $(\lambda\mathcal{A}\text{-Ch}_0)$ in which all type equivalences are expressed via coercions is weaker than the system $(\lambda\mathcal{A}$-Ch), since it cannot exploit the structural rules of Def. 8.1.7 defining $=_{\mathcal{E}}$. For instance variable $x^{\gamma\to\beta}$ cannot be reduced via the coercions $\mathit{unfold}_\beta, \mathit{fold}_\beta$ to a term of type $\gamma \to \alpha \to \beta$. To make $(\lambda\mathcal{A}\text{-Ch}_0)$ equivalent to $(\lambda\mu$-Ch) we should add a rule to define typing modulo η equality (see Exercise 8.7.17). However most of the interesting typing that can be proved in the other systems can be proved also in $(\lambda\mathcal{A}\text{-Ch}_0)$.

8.2. More on type algebras

Free algebras

8.2.1. Definition. **Freeness** Let $\mathbb{A}$ be a set of atoms, and let $\mathcal{A}$ be a type algebra such that $\mathbb{A} \subseteq |\mathcal{A}|$. We say that $\mathcal{A}$ is *the free type algebra over* $\mathbb{A}$ if, for any type algebra $\mathcal{B}$ and any function $f : \mathbb{A} \to |\mathcal{B}|$, there is a unique morphism $f^\sharp : \mathcal{A} \to \mathcal{B}$ such that, for any $\alpha \in \mathbb{A}$,

$$f^\sharp(\alpha) = f(\alpha). \tag{8.1}$$

The following is a classical result, see, e.g., **?**, characterizing the free type algebra over a set of atoms $\mathbb{A}$:

8.2.2. PROPOSITION. $\langle \mathbb{T}^{\mathbb{A}}, \rightarrow \rangle$ *is the free type algebra over* $\mathbb{A}$.

PROOF. Given a mapping $f : \mathbb{A} \to |\mathcal{B}|$, define $f^\sharp : \mathbb{T}^{\mathbb{A}} \to \mathcal{B}$ by the following recursive clauses.

$$\begin{array}{rcl} f^\sharp(\alpha) & = & f(\alpha) \\ f^\sharp(A \rightarrow B) & = & f^\sharp(A) \rightsquigarrow_{\mathcal{B}} f^\sharp(B). \end{array}$$

It is straightforward to check that this is a morphism and that it is the unique morphism that satisfies equation (8.1). ■

Subalgebras, quotients and morphisms

8.2.3. DEFINITION. Let $\mathcal{A} = \langle |\mathcal{A}|, \rightsquigarrow_{\mathcal{A}} \rangle$, $\langle \mathcal{B} = |\mathcal{B}|, \rightsquigarrow_{\mathcal{B}} \rangle$ be two type algebras. Then $\mathcal{A}$ is a sub type algebra of $\mathcal{B}$, notation $\mathcal{A} \subseteq \mathcal{B}$, if $|\mathcal{A}| \subseteq |\mathcal{B}|$ and

$$\rightsquigarrow_{\mathcal{A}} = \rightsquigarrow_{\mathcal{B}} \restriction |\mathcal{A}|,$$

i.e. for all $a_1, a_2 \in |\mathcal{A}|$ one has $a_1 \rightsquigarrow_{\mathcal{A}} a_2 = a_1 \rightsquigarrow_{\mathcal{B}} a_2$.

Clearly any subset of $\mathcal{B}$ closed under $\rightsquigarrow_{\mathcal{B}}$ induces a sub type algebra of $\mathcal{B}$.

8.2.4. PROPOSITION. **CanMorph** *Let $\mathcal{A}$ be a type algebra and let $\approx$ be a congruence on $\mathcal{A}$. Then the canonical map*

$$[\ ]_\approx : \mathcal{A} \to \mathcal{A}/\!\approx$$

is a surjective morphism.

PROOF. The map is surjective by the definition of $\mathcal{A}/\!\approx$; it is a morphism by the definition of $\rightsquigarrow_\approx$. ■

8.2.5. DEFINITION. **RangeAlgebra** Let $\langle \mathcal{A}, \rightsquigarrow_{\mathcal{A}} \rangle$, $\langle \mathcal{B}, \rightsquigarrow_{\mathcal{B}} \rangle$ be two type algebras. Suppose $f\ \mathcal{A} \to \mathcal{B}$ is a morphism. Define

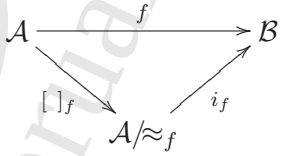
(i) $f(\mathcal{A}) = \{b \mid \exists a \in \mathcal{A}. f(a) = b\} \subseteq \mathcal{B}$ and

(ii) $a \approx_f a' \iff f(a) = f(a')$ for $a, a' \in \mathcal{A}$

8.2.6. PROPOSITION. **isom-prop** *Let $\langle \mathcal{A}, \rightsquigarrow_{\mathcal{A}} \rangle$, $\langle \mathcal{B}, \rightsquigarrow_{\mathcal{B}} \rangle$ be two type algebras and $f\ \mathcal{A} \to \mathcal{B}$ a morphism. Then*

(i) *$f(\mathcal{A})$ is a sub-type algebra of $\mathcal{B}$.*

(ii) *There are canonical morphisms $[\ ]_f : \mathcal{A} \to (\mathcal{A}/\!\approx_f)$ and $i_f : (\mathcal{A}/\!\approx_f) \to \mathcal{B}$ making the ('epi-mono') factorization of f:*



with $f = i_f \circ [\ ]_f$, where $[\ ]_f$ is surjective and i_f is injective.

(iii) $(\mathcal{A}/\approx_f) \cong f(\mathcal{A}) \subseteq \langle \mathcal{B}, \rightsquigarrow_{\mathcal{B}} \rangle$.

PROOF. (i) $f(\mathcal{A})$ is closed under $\rightsquigarrow_{\mathcal{B}}$. Indeed, $f(a) \rightsquigarrow_{\mathcal{B}} f(a') = f(a \rightsquigarrow_{\mathcal{A}} a')$.

(ii) Define $i_f : \mathcal{A}/\approx_f \to \mathcal{B}$ by $i_f([a]) = f(a)$ and $[\ ]_f = [\ ]_{\approx_f}$. Then i_f is a well-defined, injective morphism and has range $f(\mathcal{A})$. By Proposition 8.2.4 the map $[\ ]_f$ is a surjective morphism.

(iii) Easy. ■

Invertible type algebras and prime elements

8.2.7. DEFINITION. (i) A relation $\sim$ on a type algebra $\langle \mathcal{A}, \rightsquigarrow \rangle$ is called *invertible* if for all $a, b, a', b' \in \mathcal{A}$

$$(a \rightsquigarrow b) \sim (a' \rightsquigarrow b') \ \Rightarrow\ a \sim a' b \sim b'.$$

(ii) A type algebra $\mathcal{A}$ is *invertible* if the equality relation $=$ on $\mathcal{A}$ is invertible.

Invertibility has a simple characterization for syntactic type algebras.

8.2.8. REMARK. A syntactic type algebra $\mathbb{T}/\approx$ is invertible if one has

$$(A \to B) \ \approx\ (A' \to B') \ \Rightarrow\ A \ \approx\ A'B \ \approx\ B',$$

i.e. if the congruence $\approx$ on the free type algebra $\mathbb{T}$ is *invertible*.

The free syntactic type algebra $\mathbb{T}$ is invertible. See example 8.2.15(ii) for an example of a non-invertible type algebra. Another useful notion concerning type algebras is that of prime element.

8.2.9. DEFINITION. **prime-def** Let $\mathcal{A}$ be a type algebra.

(i) An element $a \in \mathcal{A}$ is *prime* if $a \neq (b \rightsquigarrow c)$ for all $b, c \in \mathcal{A}$.

(ii) We write $||\mathcal{A}|| = \{a \in \mathcal{A} \mid a \text{ is a prime element}\}$.

8.2.10. REMARK. **sint-prime** If $\mathcal{A} = \mathbb{T}/\approx$ is a syntactic type algebra, then an element $A \in \mathbb{T}$ is prime if $A \not\approx (B \to C)$ for all $B, C \in \mathbb{T}$. In this case we also say that A is prime with respect to $\approx$.

In Exercise 8.7.18(i) it is shown that a type algebra is not always generated by its prime elements. Moreover in item (iii) of that Exercise it is shown that a morphism $h\, \mathcal{A} \to \mathcal{B}$ is not uniquely determined by $h \restriction ||\mathcal{A}||$.

Well-founded type algebras

8.2.11. DEFINITION. A type algebra $\mathcal{A}$ is *well-founded* iff $\mathcal{A}$ is generated by $||\mathcal{A}||$. That is, if $\mathcal{A}$ is the least subset of $\mathcal{A}$ containing $||\mathcal{A}||$ and closed under $\rightsquigarrow$.

The free type algebra $\mathbb{T}^{\mathbb{A}}$ is well-founded while, for instance the algebra $\mathbb{T}^{\{\alpha,\beta\}}/\alpha = \alpha \to \beta$ is not. Indeed a well-founded type algebra is isomorphic to a set of simple types.

8.2.12. PROPOSITION. **PrimeEmbed** *Let $\mathcal{A}$ be an invertible type algebra.*

(i) $\mathbb{T}^{||\mathcal{A}||} \hookrightarrow \mathcal{A}$.

(ii) *If moreover $\mathcal{A}$ is well-founded, then* $\mathbb{T}^{||\mathcal{A}||} \cong \mathcal{A}$.

PROOF. (i) Let i be the morphism determined by $i(a) = a$ for $a \in ||\mathcal{A}||$. Then $i : \mathbb{T}^{||\mathcal{A}||} \hookrightarrow \mathcal{A}$. Indeed, note that the tye algebra $\mathbb{T}^{||\mathcal{A}||}$ is free and prove the injectivity of i by induction on the structure of the types, using the invertibility of $\mathcal{A}$.

(ii) By (i) and well-foundedness. ■

In Exercise 8.7.18(ii) it will be shown that this embedding is not necessarily surjective: some elements may not be generated by prime elements.

8.2.13. PROPOSITION. *Let $\mathcal{A}, \mathcal{B}$ be type algebras such that $\mathcal{A}$ is well-founded and invertible. Let $\sim, \approx$ be congruence relations on $\mathcal{A}, \mathcal{B}$, respectively. Let $h : \mathcal{A}/\!\sim \to \mathcal{B}/\!\approx$ be a map. Then h is a morphism iff there exists a morphism $h_0 : \mathcal{A} \to \mathcal{B}$ such that*

$$h([a]_\sim) = [h_0(a)]_\approx$$

$$\begin{array}{ccc} \mathcal{A} & \xrightarrow[h_0]{} & \mathcal{B} \\ {\scriptstyle [\,]_\sim}\downarrow & & \downarrow{\scriptstyle [\,]_\approx} \\ \mathcal{A}/\!\sim & \xrightarrow[h]{} & \mathcal{B}/\!\approx \end{array}$$

PROOF. ($\Rightarrow$) Define

$$\begin{array}{rcll} h_0(a) & = & b, & \text{if } a \in ||\mathcal{A}||, \text{ for some chosen } b \in h([a]_\approx); \\ h_0(a \rightsquigarrow_{\mathcal{A}} b) & = & h_0(a) \rightsquigarrow_{\mathcal{B}} h_0(b). & \end{array}$$

Then by well-founded induction one has that $h_0(a)$ is defined for all $a \in \mathcal{A}$ and $h([a]_\sim) = [h_0(a)]_\approx$, using also that $\mathcal{A}$ is invertible. The map h_0 is by definition a morphism.

($\Leftarrow$) From $h([a]) = [h_0(a)]$, with h_0 a morphism it follows that h is a morphism. ■

Satisfaction

Let $\mathcal{A}$ be a type algebra and $\mathcal{L}_\mathcal{A}$ a language of terms corresponding to $\mathcal{A}$, containing names for the elements of $\mathcal{A}$, variables and the operator $\to$. An *equation over* $\mathcal{A}$ is a formula of the form $t = s$, with t and s terms in $\mathcal{L}_\mathcal{A}$. Such an equation is called *closed* iff t, s do not contain free variables.

In the following we indicate with $\vec{\varphi}$ both a sequence of elements $\varphi_1, ..., \varphi_n$ $(n > 0)$ and the corresponding set of elements. So we will write, for instance, $\varphi_i \in \vec{\varphi}$ $(1 \leq i \leq n)$. [We do not find this necessary to formulate.]

8.2.14. DEFINITION. **EqSolution** Let $\mathcal{A}$ be a type algebra and $\mathcal{E} = \mathcal{E}(\vec{a}, \vec{X})$ be a set of equations over $\mathcal{A}$ containing $\vec{a} \in \mathcal{A}$ and free variables from $\vec{X}$.

(i) We say that $\mathcal{E}$ is valid at $\vec{b}$ in $\mathcal{A}$, notation

$$\mathcal{A} \models \mathcal{E}(\vec{a}, \vec{b}),$$

if the equations in $\mathcal{E}$ are true in $\mathcal{A}$ with the free variables $\mathrm{FV}(\mathcal{E}) = \{\vec{X}\}$ interpreted as $\vec{b}$ respectively. In this case we say that $\vec{b} \in \mathcal{A}$ is a *solution* for $\mathcal{E}(\vec{a}, \vec{X})$ in $\mathcal{A}$.

(ii) We say that $\exists \vec{X}.\mathcal{E}$ is valid in $\mathcal{A}$, notation

$$\mathcal{A} \models \exists \vec{X}.\mathcal{E}(\vec{a}, \vec{X})$$

if for some $\vec{b} \in \mathcal{A}$ one has $\mathcal{A} \models \mathcal{E}(\vec{a}, \vec{b})$.

8.2.15. EXAMPLE. **SomeQuotients**

(i) Let $\mathbb{T}^o = \mathbb{T}^{\{o\}}$, $\mathcal{E}_1 = \{o = o \to o\}$. Then all elements of $\mathbb{T}^o$ are equated in $\mathbb{T}^o/\mathcal{E}_1$. As a type algebra, $\mathbb{T}^o/\mathcal{E}_1$ contains therefore only one element $[o]_{\mathcal{E}_1}$ (that will be identified with o itself by Remark 8.1.4(i)). For instance we have

$$\mathbb{T}^o/\mathcal{E}_1 \quad \models \quad o = o \to o \to o.$$

Moreover we have that o is a solution for $X = X \to o$ in $\mathbb{T}^o/\mathcal{E}_1$.

At the semantic level an equation like $o = o \to o$ is satisfied by many models of the type free λ-calculus. Indeed using such a type it is possible to assign type X to all pure type free terms (see Exercise 8.7.12).

(ii) **inf-ext** Let $\mathbb{T}^\infty = \mathbb{T}^{\mathbb{A}_\infty}$ be a set of types with $\infty \in \mathbb{A}_\infty$. Define $\mathcal{E}_\infty$ as the set of equations

$$\infty = T \to \infty, \;\; \infty = \infty \to T,$$

where T ranges over $\mathbb{T}^\infty$. Then in $\mathbb{T}^\infty/\mathcal{E}_\infty$ the element ∞ is a solution of all equations of the form $X = A(X)$ over $\mathbb{T}^\infty$, where $A(X)$ is any type expression over $\mathbb{T}^\infty$ with at least one free occurrence of X.

A standard property of morphisms is the following.

8.2.16. LEMMA. **hom-exist** *Let $\mathcal{A}$, $\mathcal{B}$ be type algebras and suppose $h : \mathcal{A} \to \mathcal{B}$ is a morphism.*

(i) **hom-exist-1** *Let $\mathcal{E}(\vec{a})$ be a set of closed equations over $\mathcal{A}$. Then*

$$\mathcal{A} \models \mathcal{E}(a_1, \ldots, a_n) \;\Rightarrow\; \mathcal{B} \models \mathcal{E}(h(a_1), \ldots, h(a_n)).$$

(We write this as $\mathcal{A} \models \mathcal{E}(\vec{a}) \;\Rightarrow\; \mathcal{B} \models \mathcal{E}(h(\vec{a}))$.)

(ii) *Now let $\mathcal{E}(\vec{a}, \vec{X})$ be a set of equations over $\mathcal{A}$ with free variables among $\vec{X}$. Then*

$$\mathcal{A} \models \exists \vec{X}.\mathcal{E}(\vec{a}, \vec{X}) \;\Rightarrow\; \mathcal{B} \models \exists \vec{X}.\mathcal{E}(h(\vec{a}), \vec{X}).$$

PROOF. (i) Standard. (ii) By (i). ■

Enriched type algebras

The notions can be generalized in a straightforward way to type algebras having more constructors, including constants (0-ary constructors). This will happen only in exercises and applications.

8.2.17. DEFINITION. **enrich-def**

(i) A type algebra $\mathcal{A}$ is called *enriched* if there are next to $\rightsquigarrow$ also other type constructors (of arity ≥ 0) present in the signature of $\mathcal{A}$, that denote operations over $\mathcal{A}$.

(ii) An *enriched* set of types over the atoms $\mathbb{A}$, notation $\mathbb{T} = \mathbb{T}^{\mathbb{A}}_{C_1,\ldots,C_k}$ is the collection of types freely generated from $\mathbb{A}$ by $\rightarrow$ and some other constructors $C_1, \ldots, C_k$.

For enriched type algebras (of the same signature), the definitions of morphisms and congruences are extended by taking into account also the new constructors. A *congruence* over an enriched set of types $\mathbb{T}$ is an equivalence relation $\approx$ that is preserved by all constructors. For example, if C is a constructor of arity 2, we must have $a \approx b, a' \approx b' \;\Rightarrow\; C(a,b) \approx C(a',b')$.

In particular, an enriched set of types $\mathbb{T}$ together with a congruence $\approx$ yields in a natural way an *enriched syntactic type algebra* $\mathbb{T}/\approx$. For example, if $+, \times$ are two new binary type constructors and 1 is a (0-ary) type constant, we have an enriched type algebra $\langle \mathbb{T}^{\mathbb{A}}_{1,+,\times}, \rightarrow, +, \times, 1\rangle$ which is useful for applications (think of it as the set of types for a small meta-language for denotational semantics).

Sets of equations over type algebras

In general $\mathbb{T}/\mathcal{E}$ is not invertible. Take e.g. in Example 8.2.15(ii) $\mathbb{A}_\infty = \{\alpha, \infty\}$. Then in $\mathbb{T}^{\mathbb{A}_\infty}/\mathcal{E}_\infty$ one has $a \rightarrow \infty = \infty \rightarrow \infty$, but $a \neq \infty$.

Note also that in a system of equations $\mathcal{E}$ the same type can be the left-hand side of more than one equation of $\mathcal{E}$. For instance, this is the case for ∞ in Example 8.2.15 (ii).

8.2.18. PROPOSITION. **weak-decidable** *If $\mathcal{E}$ is a finite set of closed equations over $\mathbb{T}^{\mathbb{A}}$, then $=_{\mathcal{E}}$ is decidable.*

PROOF (Ackermann [1928]). Write $A =_n B$ if there is a derivation of $A =_{\mathcal{E}} B$ using a derivation of length at most n. It can be shown by a routine induction on the length of derivations that

$$\begin{aligned} A =_n B \quad \Rightarrow \quad & A \equiv B \;\vee \\ & [A \equiv A_1 \rightarrow A_2 \;\&\; B \equiv B_1 \rightarrow B_2 \;\& \\ & A_1 =_{m_1} B_1 \;\&\; A_2 =_{m_2} B_2, \text{ with } m_1, m_2 < n] \vee \\ & [A =_{m_1} A' \;\&\; B =_{m_2} B' \;\& \\ & ((A' = B') \in \mathcal{E} \vee (B' = A') \in \mathcal{E}) \text{ with } m_1, m_2 < n] \end{aligned}$$

(the most difficult case is when $A =_{\mathcal{E}} B$ has been obtained using rule (trans)).

This implies that if $A =_{\mathcal{E}} B$, then every type occurring in a derivation is a subtype of a type in $\mathcal{E}$ or of A or of B. From this we can conclude that for finite $\mathcal{E}$ the relation $=_{\mathcal{E}}$ is decidable: trying to decide that $A = B$ leads to a list of finitely many such equations with types in a finite set; eventually one should hit an equation that is immediately provable. For the details see Exercise 8.7.19 ■

In the following Lemma (i) states that working modulo some systems of equations is compositional and (ii) states that a quotient of a syntactic type algebra $\mathcal{A} = \mathbb{T}/\approx$ is just the syntactic type algebra $\mathbb{T}/\mathcal{E}$ with $\approx \subseteq \mathcal{E}$. Point (i) implies that type equations can the be solved incrementally.

8.2.19. LEMMA. **ExtTA**

(i) *Let $\mathcal{E}_1$, $\mathcal{E}_2$ be sets of equations over $\mathcal{A}$. Then*

$$\mathcal{A}/(\mathcal{E}_1 \cup \mathcal{E}_2) \cong (\mathcal{A}/\mathcal{E}_1)/\mathcal{E}_{12},$$

where $\mathcal{E}_{12}$ is defined by

$$([A]_{\mathcal{E}_1} = [B]_{\mathcal{E}_1}) \in \mathcal{E}_{12} \iff (A = B) \in \mathcal{E}_2.$$

(ii) *Let $\mathcal{A} = \mathbb{T}/\approx$ and let $\mathcal{E}$ be a set of equations over $\mathcal{A}$. Then*

$$\mathcal{A}/\mathcal{E} \cong \mathbb{T}/\mathcal{E}',$$

where

$$\mathcal{E}' = \{A = B \mid A \approx B\} \cup \{A = B \mid ([A]_{\approx} = [B]_{\approx}) \in \mathcal{E}\}.$$

PROOF. (i) By induction on derivations it follows that for $A, B \in \mathcal{A}$ one has

$$\vdash_{\mathcal{E}_1 \cup \mathcal{E}_2} A = B \iff \vdash_{\mathcal{E}_{12}} [A]_{\mathcal{E}_1} = [B]_{\mathcal{E}_1}.$$

It follows that the map $h\, \mathbb{T}/(\mathcal{E}_1 \cup \mathcal{E}_2) \to (\mathbb{T}/\mathcal{E}_1)/\mathcal{E}_{12}$, given by

$$h([A]_{\mathcal{E}_1 \cup \mathcal{E}_2}) = [[A]_{\mathcal{E}_1}]_{\mathcal{E}_{12}},$$

is well-defined and an isomorphism.

(ii) Define

$$\begin{aligned} \mathcal{E}_1 &= \{A = B \mid A \approx B\}, \\ \mathcal{E}_2 &= \{A = B \mid ([A]_{\approx} = [B]_{\approx}) \in \mathcal{E}\}. \end{aligned}$$

Then $\mathcal{E}_{12}$ in the notation of (i) is $\mathcal{E}$. Now we can apply (i):

$$\begin{aligned} \mathcal{A}/\mathcal{E} &= (\mathbb{T}/\approx)/\mathcal{E} \\ &= (\mathbb{T}/\mathcal{E}_1)/\mathcal{E}_{12} \\ &= \mathbb{T}/(\mathcal{E}_1 \cup \mathcal{E}_2). \blacksquare \end{aligned}$$

NOTATION. In general to make notations easier we often identify the level of types with that of equivalence classes of types. We do this whenever the exact nature of the denoted objects can be recovered unambiguously from the context. For example, if $\mathcal{A} = \mathbb{T}/\approx$ is a syntactic type algebra and A denotes as usual an element of $\mathbb{T}$, then in the formula $A \in \mathcal{A}$ the A stands for $[A]_{\approx}$. Similarly if $A(\vec{X}) = B(\vec{X})$ is declared as an equation over $\mathcal{A}$ then all types occurring in A, B must be considered modulo $\approx$. If we consider $\mathcal{A}$ modulo $\mathcal{E}$, then $A =_{\mathcal{E}} B$ is equivalent to $A =_{\mathcal{E}'} B$, with $\mathcal{E}'$ as in Lemma 8.2.19(ii).

8.3. Definitions of recursive types

In this section we construct type algebras containing elements satisfying recursive equations, like $a = a \rightsquigarrow b$ or $c = d \rightsquigarrow c$. There are essentially two ways to do this: defining the recursive types as the solutions of a given system of recursive type equations or via a general fixpoint operator μ in the type syntax. Recursive type equations allow to define explicitly only a finite number of recursive types, while the introduction of a fixpoint operator in the syntax makes all recursive types expressible without an explicit separate definition.

For both ways one considers types modulo a congruence relation. Some of these congruence relations will be defined proof-theoretically (inductively), as in the previous section, Definition 8.1.7. Other congruence relations will be defined semantically, using possibly infinite trees (co-inductively), as is done in Section 8.5.

Systems of Type Equations and Simultaneous recursions

In algebra one constructs, for a given ring R and set of indeterminates $\vec{X}$, a new object $R[\vec{X}]$, the ring of polynomials over $\vec{X}$ with coefficients in R. A similar construction will be made for type algebras. Intuitively $\mathcal{A}[\vec{X}]$ is the type algebra obtained by "adding" to $\mathcal{A}$ one new object for each indeterminate in $\vec{X}$ and taking the closure under $\rightsquigarrow$. Since this definition of $\mathcal{A}[\vec{X}]$ is somewhat syntactic we assume, using Prop. 8.1.10, that $\mathcal{A}$ is a syntactic type algebra. For the same reason we will always use the formal operator $\to$ instead of $\rightsquigarrow$, identifying elements of a type algebra with the corresponding terms.

Often we will take for $\mathcal{A}$ the free syntactic type algebra $\mathbb{T}^{\mathbb{A}}$ over an arbitrary nonmepty set of atomic types $\mathbb{A}$.

8.3.1. Definition. **ExtDef** Let $\mathcal{A} = \mathbb{T}^{\mathbb{A}}/\approx$ be a syntactic type algebra. Let $\vec{X} = X_1, \ldots, X_n$ $(n \geq 0)$ be a set of *indeterminates*, i.e. a set of type symbols such that $\vec{X} \cap \mathbb{A} = \emptyset$. The extension of $\mathcal{A}$ with $\vec{X}$ is defined as

$$\mathcal{A}[\vec{X}] = \mathbb{T}^{\mathbb{A} \cup \{\vec{X}\}}/\approx.$$

Note that $\mathbb{T}/\approx$ is a notation for $\mathbb{T}/=_{\approx}$. So in $\mathcal{A}[\vec{X}] = \mathbb{T}^{\mathbb{A} \cup \{\vec{X}\}}/\approx$ the relation $\approx$ is extended with the identity on the $\vec{X}$. Note also that in $\mathcal{A}[\vec{X}]$ the indeterminates are not related to any other element, since $\approx$ is not defined for elements of $\vec{X}$. By Proposition 8.1.10 this construction can be applied to arbitrary type algebras as well.

8.3.2. Proposition. $\mathcal{A} \hookrightarrow \mathcal{A}[\vec{X}]$.

We consider extensions of a type algebra $\mathcal{A}$ with indeterminates in order to build solutions to $\mathcal{E}(\vec{a}, \vec{X})$, where $\mathcal{E}(\vec{a}, \vec{X})$ (or simply $\mathcal{E}(\vec{X})$ giving $\vec{a}$ for understood) is a set of equations over $\mathcal{A}$ in the indeterminates $\vec{X}$. This solution may not exist in $\mathcal{A}$, but via the indeterminates we can build an extension $\mathcal{A}'$ of $\mathcal{A}$ containing elements $\vec{c}$ solving $\mathcal{E}(\vec{X})$.

For simplicity consider the free type algebra $\mathbb{T} = \mathbb{T}^{\mathbb{A}}$. A first way of extending $\mathbb{T}$ with elements satisfying a given set of equations $\mathcal{E}(\vec{X})$ is to consider the

type algebra $\mathbb{T}[\vec{X}]/\mathcal{E}$ whose elements are the equivalence classes of $\mathbb{T}[\vec{X}]$ under $=_{\mathcal{E}}$. By Lemma 8.1.10 and the subsequent notational remark this construction can be applied to arbitrary type algebras. The proof of the following Lemma is immediate by Lemma 8.1.9.

8.3.3. DEFINITION. Let $\mathcal{A}$ be a type algebra and $\mathcal{E} = \mathcal{E}(\vec{X})$ be a set of equations over $\mathcal{A}$ in the indeterminates $\vec{X}$. Write $\mathcal{A}[\mathcal{E}] = \mathcal{A}[\vec{X}]/\mathcal{E}$

8.3.4. LEMMA. **eq-solves** *For $\mathcal{A}$ be a type algebra and $\mathcal{E} = \mathcal{E}(\vec{X})$ be a set of equations over $\mathcal{A}$ we have*

$$\mathcal{A}[\mathcal{E}] \models \mathcal{E}(\mathbf{X}_1, \ldots, \mathbf{X}_n).$$

where $\mathbf{X}_i = [X_i]_{=_{\mathcal{E}}}$.

8.3.5. EXAMPLE. There exists a type algebra $\mathcal{A}$ such that

$$\mathcal{A} \models \exists X.(X \to X) = (X \to X \to X). \tag{1}$$

Indeed, take as type algebra $\mathcal{A} = \mathbb{T}[X]/\{X \to X = X \to X \to X\}$ and as solution $[X]_{\{X \to X = X \to X \to X\}}$. Note that in the formula (1) the X is seen as a variable, while in $\mathbb{T}[X]$ as an indeterminate.

In general $\mathcal{A}[\mathcal{E}]$ does not need to be invertible. The following notion will specialize to particular $\mathcal{E}$, such that $\mathcal{A}[\mathcal{E}]$ is invertible. A *simultaneous recursion* ('sr' also for the plural) is represented by a set $\mathcal{R}(\vec{X})$ of type equations of a particular shape over $\mathcal{A}$, in which the indeterminates $\vec{X}$ represent the recursive types to be added to $\mathcal{A}$. Such types occur in programming languages, for the first time in Algol-68.

8.3.6. DEFINITION. **sr-general** Let $\mathcal{A}$ be a type algebra. A system of recursive equations (sr) over $\mathcal{A}$ in the indeterminates $\vec{X} = \{X_1, \ldots, X_n\}$ is a finite set $\mathcal{R}(\vec{X})$ of n equations over $\mathcal{A}$ of the form

$$\begin{aligned} X_1 &= A_1(\vec{X}); \\ &\ldots \\ X_n &= A_n(\vec{X}). \end{aligned}$$

where all indeterminates $X_1, \ldots, X_n$ are different. We also say that $\vec{X}$ is the *domain* of $\mathcal{R}$ (notation $\mathrm{Dom}(\mathcal{R})$).

8.3.7. EXAMPLE. For example

$$\begin{aligned} X_1 &= X_2 \to X_1 \\ X_2 &= \alpha \to X_1 \end{aligned}$$

is an sr in the indeterminates $\{X_1, X_2\}$ over $\mathbb{T}^{\mathbb{A}}$ with $\alpha \in \mathbb{A}$.

An important property of sr is that they do not identify elements of the $\mathcal{A}$.

8.3.8. LEMMA. **NoConfusion** *Let $\mathcal{R}(\vec{X})$ be a sr over a type algebra $\mathcal{A}$. Then for all $a, b \in \mathcal{A}$ $a \neq b$ imply $a \neq_{\mathcal{R}} b$.*

PROOF. Easy. ■

8.3.9. DEFINITION. **TAmodR** Let $\mathcal{R}(\vec{X})$ be a set of equations in $\vec{X}$ over a type algebra $\mathcal{A}$ (i.e. a set of equations between elements of $\mathcal{A}[\vec{X}]$). Write

$$\mathcal{A}[\mathcal{R}] \quad = \quad \mathcal{A}[\vec{X}]/\mathcal{R}$$

Lemma 8.3.8 is no longer true, in general, if we start work with a set of equations $\mathcal{E}(\vec{X})$ instead of an sr $\mathcal{R}(\vec{X})$. Take e.g. $\mathcal{E}(X) = \{X = a, X = b\}$. In this case $a =_{\mathcal{E}} b$. In the following we will use indeterminates only in the definition of sr. Generic equations will be considered only between closed terms (i.e. without indeterminates).

8.3.10. THEOREM. **adjunction**

(i) $\mathcal{A}[\mathcal{R}] \models \mathcal{R}(\vec{\mathbf{X}})$, *where* $\mathbf{X}_i = [X_i]_{=_{\mathcal{R}}}$;

(ii) $\mathcal{A} \hookrightarrow \mathcal{A}[\mathcal{R}]$ *and* $\mathcal{A}[\mathcal{R}]$ *is generated from (the image of)* $\mathcal{A}$ *and the equivalence classes with respect to* $=_{\mathcal{R}}$ *of the indeterminates* $\vec{X}$.

PROOF. (i) By Lemma 8.3.4.

(ii) The map $A \mapsto [A]_{=_{\mathcal{R}}}$ is injective by Lemma 8.3.8. Clearly $\mathcal{A}[\mathcal{R}]$ is generated by the image of $\mathcal{A}$ and the $\vec{\mathbf{X}}$. ■

Note that he image of $\mathcal{A}$ and the $\vec{\mathbf{X}}$ are not necessarily disjoint. For instance if $\mathcal{R}$ contains an equation $X = a$ where $X \in \vec{X}$ and $a \in \mathcal{A}$ we have $[X] = [a]$.

We say that $\mathcal{A}[\mathcal{R}]$ is obtained by *adjunction* of the elements $\vec{\mathbf{X}}$ to $\mathcal{A}$. The method of adjunction then allows us to define recursive types incrementally, according to Lemma 8.2.19 (i).

8.3.11. REMARK. (i) In Chapter 9 it will be proved that $=_{\mathcal{R}}$ is invertible, and hence $\mathbb{T}[\mathcal{R}]$ is an invertible type algebra. Note that if $\mathcal{E}$ is an arbitrary set of equations $=_{\mathcal{E}}$, in general, is not invertible.

(ii) Let the indeterminates or $\mathcal{R}_1$ and $\mathcal{R}_2$ be disjoint, then $\mathcal{R}_1 \cup \mathcal{R}_2$ is an sr again. By Lemma 8.2.19 (ii) $\mathcal{A}[\mathcal{R}_1 \cup \mathcal{R}_2] = \mathcal{A}[\mathcal{R}_1][\mathcal{R}_2]$. Recursive types can therefore be defined incrementally.

Simultaneous recursions are a natural tool to specify types satisfying given equations. We call *unfolding (modulo $\mathcal{R}$)* the operation of replacing an occurrence of X_i by $A_i(\vec{X})$, for any equation $X_i = A_i(\vec{X}) \in \mathcal{R}$; *folding* is the reverse operation. If $a, b \in \mathcal{A}[\vec{X}]$ then $a =_{\mathcal{R}} b$ if they can be transformed one into the other by a finite number of applications of the operations folding and unfolding. In this case one can say that a is *weakly* equivalent to b with respect to $\mathcal{R}$.

8.3.12. EXAMPLE. **constraintexampleone**

(i) The sr $\mathcal{R}_0 = \{X_0 = A \to X_0\}$, where $A \in \mathbb{T}$ is a type, specifies a type X_0 which is such that

$$X_0 =_{\mathcal{R}_0} A \to X_0 =_{\mathcal{R}_0} A \to A \to X_0 \ldots$$

i.e. $X_0 =_{\mathcal{R}_0} A^n \to X_0$ for any n. This represents the behavior of a function which can take an arbitrary number of arguments of type A.

(ii) The sr $\mathcal{R}_1 = \{X_1 = A \to A \to X_1\}$ is similar to $\mathcal{R}_0$ but not all equations modulo $\mathcal{R}_0$ hold modulo $\mathcal{R}_1$. For instance $X_1 \neq_{\mathcal{R}_1} A \to X_1$ (i.e. we cannot derive $X_1 = A \to X_1$ from the equations of Definition 8.1.7(ii)).

8.3.13. Remark. Note that $=_{\mathcal{R}}$ is the minimal congruence with respect to $\to$ satisfying $\mathcal{R}$. Two types can be different w.r.t. it even if they seem to represent the same behavior, like X_0 and X_1 in the above example. As another example take $\mathcal{R} = \{X = A \to X,\ Y = A \to Y\}$. Then we have $X \neq_{\mathcal{R}} Y$ since we cannot prove $X = Y$ using only the equations of definition 8.1.7(ii). These types will instead be identified in the tree equivalence introduced in Section 8.5.

It is useful to consider restricted forms of simultaneous recursion.

8.3.14. Definition (Simultaneous recursion). **SimRec**
(i) A sr $\mathcal{R}(\vec{X})$ is *proper* if all equations of the form $X_i = X_j \in \mathcal{R}$ are such that $i < j$.
(ii) A sr $\mathcal{R}(\vec{X})$ is *simplified* if no equation $X_i = X_j$ occurs in $\mathcal{R}$.

Note that a simplified sr is proper. In general a sr $\mathcal{R}$ is proper if it is possible to define a total order $\prec$ between its indeterminates in such a way that if $X = Y \in \mathcal{R}$ then $X \prec Y$. The definition of proper is intended to rule out circular definitions like $X = X$ or $X = Y, Y = X$. Non-proper sr are somewhat pathological and uninteresting. We always can make an sr leaving out an equation $X_j = X_i$, with $j > i$, and replacing in the other equations X_j by X_i. In the rest of this part we will consider only proper simultaneous recursions.

It is sometimes useful to transform an sr into an equivalent one. To this aim we introduce the notion of equivalence for simultaneous recursion.

8.3.15. Definition. **sr-equivalence** Let $\mathcal{R} = \mathcal{R}(\vec{X})$ and $\mathcal{R}' = \mathcal{R}'(\vec{X}')$ be sr over $\mathcal{A}$.
(i) $\mathcal{R}$ and $\mathcal{R}'$ are *equivalent* if $\mathcal{A}[\mathcal{R}] \cong \mathcal{A}[\mathcal{R}']$.
(ii) Let $\vec{X} = \vec{X}'$ be the same set of indeterminates. Then $\mathcal{R}(\vec{X})$ and $\mathcal{R}'(\vec{X})$ are *logically equivalent* if

$$\forall a, b \in \mathcal{A}[\vec{X}].a =_{\mathcal{R}} b \iff a =_{\mathcal{R}'} b.$$

8.3.16. Remark. (i) It is easy to see that $\mathcal{R}$ and $\mathcal{R}'$ over the same $\vec{X}$ are logically equivalent iff

$$\vdash_{\mathcal{R}} \mathcal{R}' \ \& \ \vdash_{\mathcal{R}'} \mathcal{R}.$$

(ii) Two logically equivalent sr are also equivalent.

(iii) There are equivalent $\mathcal{R}, \mathcal{R}'$ that are not logically equivalent, e.g.

$$\mathcal{R} = \{X = \alpha\} \text{ and } \mathcal{R}' = \{X = \beta\}.$$

Note that $\mathcal{R}$ and $\mathcal{R}'$ are on the same set of indeterminates.

Simplified and flat sr are equivalent to proper ones. This is formalized in the following proposition.

8.3.17. PROPOSITION. **simplify**

(i) *Every proper sr $\mathcal{R}(\vec{X})$ over $\mathcal{A}$ is such that, for all $X_i \in \vec{X}$, either $X_i =_{\mathcal{R}} a$ where $a \in \mathcal{A}$ or $X_i =_{\mathcal{R}} (b \to c)$ for some $b, c \in \mathcal{A}[\vec{X}]$.*

(ii) *Every proper sr $\mathcal{R}(\vec{X})$ over $\mathcal{A}$ is equivalent to a simplified $\mathcal{R}'(\vec{X}')$, where $\vec{X}'$ is a subset of $\vec{X}$.*

PROOF. (i) Left as an easy exercise.

(ii) If $\mathcal{R}$ is not simplified, then $\mathcal{R} = \mathcal{R}_1 \cup \{X_i = X_j\}$, with $i<j$. Now define

$$\mathcal{R}^-(X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n),$$

by $\mathcal{R}^- = \mathcal{R}_1[X_i = X_j]$. Note that $\mathcal{R}^-$ is still proper (since an equation $X_k = X_i$ in $\mathcal{R}$ becomes $X_k = X_j$ in $\mathcal{R}^-$ and $k<i<j$), equivalent to $\mathcal{R}$, and has one equation less. So after finitely many such steps the simplified $\mathcal{R}'$ is obtained. ■

8.3.18. REMARK. **trandormation** It is easy to prove that also an arbitrary sr can be transformed in a proper one possibly adding some new prime elements corresponding to circular definitions(see Exercise 8.7.5).

The prime elements of the type algebras $\mathbb{T}[\mathcal{R}]$, where $\mathcal{R}$ is proper and $\mathbb{T} = \mathbb{T}^{\mathbb{A}}$, can easily be characterized.

8.3.19. LEMMA. **prime-lemma** *Let $\mathcal{R}(\vec{X})$ be a proper sr over $\mathbb{T}^{\mathbb{A}}$. Then*

$$||\mathbb{T}[\mathcal{R}]|| = \{[\alpha] \mid \alpha \in \mathbb{A}\}.$$

PROOF. The elements of $\mathbb{T}[\mathcal{R}]$ are generated from the $\mathbb{A}$ and the $\vec{X}$. Now note that by Lemma 8.3.17 (i) an undeterminate X either is such that $X =_{\mathcal{R}} A \to B$ for some $A, B \in \mathbb{T}^{\mathbb{A} \cup \vec{X}}$ (and then $[X]$ is not prime) or $X =_{\mathcal{R}} \alpha$ for some atomic type α. Lemma 8.3.8 and a simple induction on the number of equations in $\mathcal{R}$ shows that, in this case, no other atomic types or arrow types can belong to $[\alpha]$. Therefore, the only prime elements in $\mathbb{T}[\mathcal{R}]$ are the equivalence classes of the $\alpha \in \mathbb{A}$. ■

For a proper sr $\mathcal{R}$ we can write, for instance $||\mathbb{T}[\mathcal{R}]|| = \mathbb{A}$ choosing α as the representative of $[\alpha]$.

Representing sets of equations in a type algebra

A useful notion is the following. Let $\mathcal{E}$ be a system of equations over a set $\mathbb{T}$ of types. A type algebra $\mathcal{A}$ represents $\mathcal{E}$ if there is a map $h : \mathbb{T} \to \mathcal{A}$ such that $\mathcal{A} \models h(\mathcal{E})$. This amounts to saying that h is a type algebra homomorphism from $\mathbb{T}/\mathcal{E}$ to $\mathcal{A}$. The intuition is that a type constant in $\mathbb{T}$ is interpreted as an element in $\mathcal{A}$ in such a way that the equations in $\mathcal{E}$ become valid. This notion can be generalized by taking instead of $\mathbb{T}$ an arbitrary type algebra $\mathcal{B}$.

8.3.20. DEFINITION. **Solvessr** Let $\mathcal{A}$ and $\mathcal{B}$ be type algebras and let $\mathcal{E}$ be a system of type equations over $\mathcal{B}$ (i.e. between expressions for elements of $\mathcal{B}$).

(i) $\mathcal{A}$ *represents* $\mathcal{E}$ over $\mathcal{B}$ if there is a type algebra morphism $h : \mathcal{B}/\mathcal{E} \to \mathcal{A}$.

(ii) If moreover $\mathcal{R}$ is an sr over $\mathcal{A}$, then we say that $\mathcal{R}$ *represents* $\mathcal{E}$ if $\mathcal{A}[\mathcal{R}]$ represents $\mathcal{E}$.

If there is little danger of confusion we may leave out 'over $\mathcal{B}$'.

8.3.21. EXAMPLE. Let $\mathcal{E} = \{\alpha \to \beta = \alpha \to \alpha \to \beta\}$. Then $\mathcal{R} = \{X = \alpha \to X\}$ represents $\mathcal{E}$ over $\mathbb{T}^{\{\alpha,\beta\}}$ as we have the homomorphism

$$h : \mathbb{T}^{\{\alpha,\beta\}}/\mathcal{E} \to \mathbb{T}^{\{\alpha\}}[\mathcal{R}]$$

determined by $h([\alpha]_{\mathcal{E}}) = [\alpha]_{\mathcal{R}}$, $h([\beta]_{\mathcal{E}}) = [X]_{\mathcal{R}}$, or, with our notational conventions, $h(\alpha) = \alpha$, $h(\beta) = X$ (where h is indeed a syntactic homomorphism). The reason that h is a morphism is that

$$\vdash_{\mathcal{R}} h(\mathcal{E}).$$

Note that by Remark 8.1.6(iii) we can say that an sr $\mathcal{R}[\vec{X}']$ over $\mathbb{T}'$ represents $\mathcal{E}$ over $\mathbb{T}$ if

$$\exists h\, \mathbb{T} \to \mathbb{T}'[\vec{X}'].\, \forall A,\, B \in \mathbb{T}.\, [(A = B) \in \mathcal{E} \;\Rightarrow\; h(A) =_{\mathcal{R}} h(B)].$$

Solving an sr

8.3.22. LEMMA. **SolutionAsHom** *Let $\mathcal{A}$ be a type algebra and $\mathcal{R}(\vec{X})$ be a sr over $\mathcal{A}$. Then*

$$\mathcal{A} \models \exists \vec{X}.\mathcal{R}(\vec{X}) \iff \exists h\, \mathcal{A}[\mathcal{R}] \to \mathcal{A}\, \forall a \in \mathcal{A}.h(a) = a.$$

The choice of $\vec{a} \in \mathcal{A}$ for the $\vec{X}$ uniquely determines such an h and vice-versa.

PROOF. ($\Rightarrow$) Let $\vec{A} \in \mathcal{A}$ be such that $\mathcal{A} \models \mathcal{R}(\vec{a})$. Define h as the homomorphism such that $h(X_i) = a_i$ for all $X_i \in \vec{X}$ and $h(a) = a$ for all $a \in \mathcal{A}$. Note that h is well defined since $\vec{a}$ is a solution of $\mathcal{R}$.

($\Leftarrow$) Conversely note that by 8.1.9 and 8.2.16 the $h(X_1), \ldots, h(X_n)$ form a solution of $\mathcal{R}$ in $\mathcal{A}$. ■

A last general notion concerning sr is the following.

8.3.23. DEFINITION. **closed** Let $\mathcal{A}$ be a type algebra.

(i) $\mathcal{A}$ is *closed* iff every system of sr $\mathcal{R}$ over $\mathcal{A}$ can be solved in $\mathcal{A}$.
(ii) $\mathcal{A}$ is *uniquely closed*, if any proper $\mathcal{R}$ over $\mathcal{A}$ has a unique solution in $\mathcal{A}$.

For instance the type algebra $\mathbb{T}_\infty/\mathcal{E}_\infty$ of Example 8.2.15 (ii) is closed but not uniquely closed. For instance the equation $X = \alpha \to X$, with $\alpha \in \mathbb{A}_\infty$, can be solved by taking $h_1(X) = \infty$, $h_1(\alpha) = \alpha$ or by taking $h_2(X) = \infty$, $h_2(\alpha) = \infty$. A uniquely closed type algebra that is more simple to construct will be given in section 8.5.

From Proposition 8.2.18 we know that $=_\mathcal{R}$ is decidable for any (finite) $\mathcal{R}$. In Chapter 9 we will prove some other properties of $\mathbb{T}[\mathcal{R}]$, in particular the following.
(i) $\mathbb{T}[\mathcal{R}]$ is an invertible type algebra.
(ii) It is decidable whether an sr $\mathcal{R}$ represents a set $\mathcal{E}$ of equations.
We will use (i) already in Lemma 8.6.1 in this Chapter.

μ-types

mutypes

Another way of representing recursive types is that of enriching the syntax of types with a new operator μ to explicitly denote solutions of recursive type equations. The resulting (syntactic) type algebra "solves" arbitrary type equations, i.e. is closed in the sense of definition 8.3.23.

8.3.24. DEFINITION (μ-types). **types** Let $\mathbb{A}$ be a set of atomic types containing a set $\mathbb{V}$ of type variables. The set $\mathbb{T}_\mu^\mathbb{A}$ ($\mathbb{T}_\mu$ for short when $\mathbb{A}$ is understood) is defined by the following abstract syntax:

$$\boxed{\mathbb{T}_\mu^\mathbb{A} \quad = \quad \mathbb{A} \mid \mathbb{T}_\mu^\mathbb{A} \to \mathbb{T}_\mu^\mathbb{A} \mid \mu\alpha.\mathbb{T}_\mu^{\mathbb{A}\cup\{\alpha\}}}$$

We assume that $\to$ takes precedence over μ, so that, for example, the type $\mu\alpha.A \to B$ should be parsed as $\mu\alpha.(A \to B)$. The subset of $\mathbb{T}_\mu^\mathbb{A}$ containing only types without occurrences of the μ operator equations coincides with the set $\mathbb{T}^\mathbb{A}$ of simple types.

In $\mu\beta.A$ the operator μ binds the variable β, which can therefore be renamed by α-conversion: $\mu\beta.A \equiv_\alpha \mu\gamma.A[\beta = \gamma]$, provided that γ does not occur in A. We will always assume in the sequel that the names of bound and free variables in types are distinct: this can be easily obtained by a renaming of bound variables.

According to the intuitive semantics of recursive types, a type expression of the form $\mu\alpha.A$ should be regarded as the solution for α in the equation $\alpha = A$, and is then equivalent to the type expression $A[\alpha = \mu\alpha.A]$. The notion of type equivalence $=_\mu$ is defined by a set of formal rules in which this equality is extended to a congruence on $\mathbb{T}_\mu$ by adding structural rules and transitivity.

8.3.25. DEFINITION. **muweakeq**

(i) The equational theory (μ) is defined by the following axioms and rules.

$$
\begin{array}{ll}
(\mu\text{-eq}) & \vdash \mu\alpha.A = A[\alpha := \mu\alpha.A] \\
(\text{ident}) & \vdash A = A \\
(\text{symm}) & \dfrac{\vdash A = B}{\vdash B = A} \\
(\text{trans}) & \dfrac{\vdash A = B \quad \vdash B = C}{\vdash A = C} \\
(\to\text{-cong}) & \dfrac{\vdash A = A' \quad \vdash B = B'}{\vdash A \to B = A' \to B'} \\
(\mu\text{-cong}) & \dfrac{\vdash A = A'}{\vdash \mu\alpha.A = \mu\alpha.A'}
\end{array}
$$

We say that $A, B \in \mathbb{T}_\mu$ are *(weakly) equivalent*, notation $A =_\mu B$, if $\vdash_\mu A = B$, i.e. $\vdash A = B$ is provable in (μ).

(ii) We will often identify $\mathbb{T}_\mu$ with the (syntactic) type algebra $\mathbb{T}_\mu / =_\mu$, if there is little danger of confusion.

8.3.26. Remark. (i) In the theory (μ) we call *unfolding* the operation consisting in replacing $\mu\alpha.A$ by $A[\alpha := \mu\alpha.A]$ and *folding* its inverse.

(ii) Two types in $\mathbb{T}_\mu$ are weakly equivalent if they can be transformed one into the other by a finite number of applications of folding and unfolding.

We will use $\equiv$ to denote syntactic equality of types modulo α-conversion. For example $\mu\alpha.\alpha \to \alpha \equiv \mu\beta.\beta \to \beta$.

8.3.27. Example. **weakexample** Let $B \equiv \mu\beta.A \to \beta$. Then we have

$$B =_\mu A \to B =_\mu A \to A \to B =_\mu \ldots$$

Now let $B' \equiv \mu\beta.(A \to A \to \beta)$. Then

$$B' =_\mu A \to A \to B' =_\mu \ldots$$

It is easy to see that $B \neq_\mu B'$ (see also Section 9.2). Indeed B and B' are the μ-types solving, respectively, the type equations in $\mathcal{R}_0$ and $\mathcal{R}_1$ of Example 8.3.12.

8.3.28. Lemma. **WeakInvSubst** *If $A =_\mu B$ then $A[\alpha := C] =_\mu B[\alpha := C]$.*

Proof. By induction on the derivation of $A =_\mu B$. ■

Each μ-type A can be unfolded to a form in which its main constructor (a type variable or $\to$) is brought at top level, unless A belongs to the following pathological subset of $\mathbb{T}_\mu$.

8.3.29. DEFINITION. **dumb-mu-free** Let $A \in \mathbb{T}_\mu$.
(i) A is called *circular* iff $A = \mu\alpha_1 \ldots \alpha_n.\alpha_i$ for some $1 \leq i \leq n$.
(ii) The most typical circular type is $\mathbf{\Omega} \equiv \mu\alpha.\alpha$.
(iii) A is *contractive* if it is not circular.

Circular types correspond to non-proper sr.

8.3.30. LEMMA. *Let A be a circular type. Then $A =_\mu \mathbf{\Omega}$. Therefore the circular types form a unique equivalence class in $\mathbb{T}_\mu$.*

PROOF. Let $A = \mu\alpha_1 \ldots \alpha_n.\alpha_i$. Note that $\mu\alpha.B =_\mu B$ if $\alpha \notin \mathrm{FV}(B)$. Hence $\mu\alpha_i \ldots \alpha_n.\alpha_i = \mu\alpha_i.\alpha_i = \mathbf{\Omega}$, and therefore $A =_\mu \mathbf{\Omega}$. ■

8.3.31. COROLLARY. *Let $A \in \mathbb{T}_\mu$ then we have exactly one of the following cases*

$$\begin{array}{lll} A & =_\mu & \alpha; \\ A & =_\mu & B \to C; \\ A & =_\mu & \mathbf{\Omega}. \end{array}$$

PROOF. By induction on the complexity of A. In case $A \equiv \mu\beta.B$, then by the induction hypothesis B is of one of the three forms. Then A is of the same form by unfolding $\mu\beta.B$, unless $B =_\mu \beta$ and then $A = \mathbf{\Omega}$. ■

The three forms in this Corollary are called the *reduced forms* of a type. If A is contractive then it has either α or $B \to C$ as reduced form. In Example 8.3.27 one has $A \to B$ for B as reduced form.

8.3.32. LEMMA. (i) *The prime elements of $\mathbb{T}_\mu$ are given by*

$$||\mathbb{T}_\mu|| = \{[\alpha] \mid \alpha \in \mathbb{A}\} \cup \{[\mathbf{\Omega}]\}$$

(ii) $\mathbb{T}^{\mathbb{A}} \hookrightarrow \mathbb{T}_\mu^{\mathbb{A}}$.

PROOF. (i) See Exercise 8.7.9.
(ii) One has

$$\begin{array}{lll} \mathbb{T}^{\mathbb{A}} & \hookrightarrow & \mathbb{T}^{||\mathbb{T}_\mu||}, \quad \text{by (i)}, \\ & \hookrightarrow & \mathbb{T}_\mu^{\mathbb{A}}, \quad \text{by Lemma 8.2.12}, \end{array}$$

the necessary condition that $\mathbb{T}_\mu$ is invertible will be proved in Chapter 9. ■

8.3.33. REMARK. In Section 9 we will show the following properties of $\mathbb{T}_\mu$.
(i) $\mathbb{T}_\mu$ is closed, i.e. $\mathbb{T}_\mu \models \exists \vec{X}.\mathcal{R}$ for all sr $\mathcal{R}$ over $\mathbb{T}_\mu$.
(ii) $=_\mu$ is invertible, and hence $\mathbb{T}_\mu$ is an invertible type algebra.
(iii) $=_\mu$ is decidable.

As a shorthand we denote the type assignment systems corresponding to $\mathbb{T}_\mu$, $\mathbb{T}/{\approx}$ and $\mathbb{T}[\mathcal{R}]$ by $(\lambda\mu)$, $(\lambda{\approx})$ and $(\lambda\mathcal{R})$ respectively. The only difference in the formal presentations of these systems is in the set of types and in the definition of the corresponding equivalence. We will use a similar conventions also for the type algebras introduced in the following sections.

8.3.34. EXAMPLE. **MuTypeEx** Consider the deductions in Proposition 8.1.12. We can obtain them in $(\lambda\mu)$ by taking a as an arbitrary type $A \in \mathbb{T}_\mu$ and b as $\mu t.t \to A$. Note in fact that $\mu t.t \to A =_\mu (\mu t.t \to A) \to A$

8.3.35. REMARK. **mu-subst** It is easy to see that any substitution $s : \mathbb{A} \to \mathbb{T}_\mu$ determines a type algebra homomorphism of $s : \mathbb{T}_\mu \to \mathbb{T}_\mu$. So, by Lemma 8.1.15, all the the typings of lemma 8.1.12 holds in $(\lambda\mu)$ if we replace α by any type $A \in \mathbb{T}_\mu$ (taking the substitution $[\alpha := A]$). Note in particular that $(\lambda x.xx)(\lambda x.xx)$ has all types and that the fixpoint operator has all the types of the form $(A \to A) \to A$. Obviously the same property holds in all other systems that will be considered.

8.4. Introduction to algebras and coalgebras 21.2.2008:897

This subsection has been entirely remodeled, often without explicit notice. The algebras with parameters have disappeared, and the categorical treatment of algebras has been used from the start. Throughout this section only, the parts marked \fc{} (displayed in green) are mostly to be regarded as erased, when appropriate. This section is intended as a light introduction to algebras, coalgebras and among these the (co-)inductive ones: to the best of my knowledge this is not the standard terminology with their corresponding (co-)induction and (co-)recursion principles. All these notions have natural categorical descriptions, that generalize their standard set-theoretic presentation, and simplify the proofs of many of their properties.

Algebras

A typical example of algebra in the general sense in which we shall use this word throughout this introduction is the structure $\langle \mathbb{N}, 0, \mathtt{suc} \rangle$, namely the algebra of the natural numbers with a constant $0 \in \mathbb{N}$ and a unary operator $\mathtt{suc} : \mathbb{N} \to \mathbb{N}$, the successor function. Another example is the algebra of lists $\langle \mathtt{List}_\mathbb{A}, \mathtt{nil}, \mathtt{cons} \rangle$ over a set $\mathbb{A}$ of atoms, where $\mathtt{nil} \in \mathtt{List}_\mathbb{A}$ denotes the empty list with no elements, and the list constructor $\mathtt{cons} : \mathbb{A} \times \mathtt{List}_\mathbb{A} \to \mathtt{List}_\mathbb{A}$ builds from arguments $\langle \alpha, \ell \rangle$ the list having the atom α as its head and the list ℓ as its tail.

Observe now that in the category of sets and functions, two functions $f : A \to C$ and $g : B \to C$ can be 'packed' into one function, in the following way:

8.4.1. DEFINITION. (i) Let A, B be sets. Define the disjoint union

$$A + B = A \times \{0\} \cup B \times \{1\}.$$

(ii) There are canonical maps

$$\begin{aligned} \mathtt{inl} &: A \to A + B \\ \mathtt{inr} &: B \to A + B \end{aligned}$$

defined by

$$\begin{aligned} \mathtt{inl}(a) &= \langle a, 0 \rangle, \\ \mathtt{inr}(b) &= \langle b, 1 \rangle. \end{aligned}$$

(iii) Two functions $f : A \to C$ and $g : B \to C$ can be packed together as $[f, g] : A + B \to C$ by setting

$$\begin{aligned} [f, g](\mathtt{inl}(a)) &= f(a), \\ [f, g](\mathtt{inr}(b)) &= g(b). \end{aligned}$$

This allows to pack together the operations of an algebra as one map from a disjoint union into the carrier set of the algebra. For example, by applying the above definition to the data defining the algebra of natural numbers $\langle \mathbb{N}, 0 \in \mathbb{N}, \mathtt{suc} : \mathbb{N} \to \mathbb{N} \rangle$ and regarding 0 as a map $0 : 1 \to \mathbb{N}$, where 1 is any (fixed) one-element set, we obtain

$$[0, \mathtt{suc}] : 1 + \mathbb{N} \to \mathbb{N}.$$

Similarly, for lists over $\mathbb{A}$, we have a mapping

$$[\mathtt{nil}, \mathtt{cons}] : 1 + (\mathbb{A} \times \mathtt{List}_{\mathbb{A}}) \to \mathtt{List}_{\mathbb{A}}.$$

These algebras are special cases of functions of the form $1 + X \to X$ and $1 + (\mathbb{A} \times X) \to X$, respectively, where X is any set. Categorically these notions can be formulated conveniently as follows.

8.4.2. Definition. Let $T : \mathsf{Set} \to \mathsf{Set}$ be a functor. We call it the *signature functor*.

(i) A T-*algebra* consists of a pair $\langle X, \xi \rangle$ with $X \in \mathsf{Set}$ the *carrier* and ξ a morphism

$$\xi : T(X) \to X.$$

(ii) If $\langle X, \xi : T(X) \longrightarrow X \rangle$ and $\langle Y, \zeta : T(Y) \longrightarrow Y \rangle$ are two T-algebras, then a *morphism* is a mapping $f : X \longrightarrow Y$ such that the following diagram commutes.

$$\begin{array}{ccc} T(X) & \xrightarrow{\xi} & X \\ {\scriptstyle T(f)}\downarrow & & \downarrow{\scriptstyle f} \\ T(Y) & \xrightarrow{\zeta} & Y \end{array}$$

It is easy to check that the T-algebras with these morphisms form a category.

(iii) A T-algebra $\langle A, \mathtt{in} \rangle$ is called *initial* iff it is an initial object in the category of T-algebras, i.e. for every T-algebra $\langle X, \xi \rangle$ there is a unique morphism $\varphi \langle A, \mathtt{in} \rangle \to \langle X, \xi \rangle$.

For the familiar (initial) algebras the functors ars defined as follows.

8.4.3. Examples. **exs.functor**

(i) The natural numbers $\langle \mathbb{N}, 0, \mathtt{suc} \rangle$ form the initial algebra of the signature functor

$$T_{\mathbb{N}}(X) = 1 + X.$$

(ii) The simple types $\mathbb{T}^{\mathbb{A}}$ form the initial algebra of the signature functor

$$T_{\mathbb{T}^{\mathbb{A}}}(X) = \mathbb{A} + (X \times X).$$

(iii) The lists $\mathtt{list}_{\mathbb{A}}$ form the initial algebra of the signature functor

$$T_{\mathbb{L}^{\mathbb{A}}}(X) = 1 + (\mathbb{A} \times X).$$

In general we can observe that all functors discussed so far has the shape of a sum (= disjoint union) of powers (= iterated cartesian products) of one "indeterminate" X. This motivates the following definition:

8.4.4. Definition. A *polynomial* functor $T\ \mathsf{Set} \to \mathsf{Set}$ has the form

$$T(X) = a_0 \times X^0 + a_1 \times X^1 + \ldots + a_n \times X^n,$$

where the $a_k \in \mathsf{Set}$ are arbitrary objects and $X^n =_{\text{def}} \underbrace{X \times \cdots \times X}_{n \text{ times}}$.

8.4.5. Proposition. *For every polynomial functor $T\ \mathsf{Set} \to \mathsf{Set}$ there is an initial algebra $\langle I, \mathtt{in}\rangle$ that is unique up to isomorphism in the category of T-algebras. Moreover, $\mathtt{in} : T(I) \to I$ is an isomorphism, hence $T(I) \cong I$.*

Proof. (Sketch) Take I as the direct limit (union with identifications via the $T^k(f)$)

$$0 \xrightarrow{f} T(0) \xrightarrow{T(f)} T^2(0) \xrightarrow{T^2(f)} \cdots,$$

where $0 = \emptyset$ is the initial element of Set and $f\ 0 \to T(0)$ the canonical map. The inverse of the morphism $\mathtt{in}$ is defined by mapping an element of I, say in $T^k(0)$, via $T^k(f)$ to the next level in I. This is an isomorphism. ■

Observe that the unique morphism $I \to X$ from the initial T-algebra $\mathtt{in} : T(I) \to I$ to an arbitrary T-algebra $T(X) \to X$ can be regarded as the function defined by iteration from $\mathtt{in} : T(I) \to I$, precisely in the same sense as the unique morphism from the initial $T_{\mathbb{N}}$-algebra $[0, \mathtt{suc}] : 1 + \mathbb{N} \to \mathbb{N}$ to any other $T_{\mathbb{N}}$-algebra $T_{\mathbb{N}}(X) \to X$ is said to be defined by iteration. A general principle of definition by primitive recursion follows directly (in the case of $\mathbb{N}$, see **?**; these ideas stem of course from **?**).

Coalgebras

Coalgebras appear in nature as infinitely proceeding processes. Coalgebras come with operations that do not construct the elements like in the case of algebras (having the algebra as their codomain), but that *deconstruct* them (having the coalgebra as their domain): in fact they are categorically dual to algebras, whence their name. A simple coalgebra is that of streams, i.e., the infinite lists of elements of some set $\mathbb{A}$: this can be characterized as the final coalgebra of the signature functor $T_{\mathbb{S}^{\mathbb{A}}}(X) = \mathbb{A} \times X$. Observe that the initial algebra has no elements. Another coalgebra is that of *lazy lists*, consisting of both the finite lists and the streams. Of importance for this Part II on recursive types is the coalgebra of the finite and infinite trees, treated in the next section.

8.4.6. DEFINITION. (i) The *streams over* $\mathbb{A}$, notation $\mathbb{S}^{\mathbb{A}}$, consists of all infinite sequences of elements of $\mathbb{A}$, i.e.

$$\mathbb{S}^{\mathbb{A}} = \mathbb{A}^{\mathbb{N}}.$$

An element $s \in \mathbb{S}^{\mathbb{A}}$ is often written as $s = \langle s_0, s_1, s_2, \ldots \rangle$, with $s_i = s(i)$.

(ii) There are two basic operations ('head' and 'tail') on $\mathbb{S}^{\mathbb{A}}$:

$$\begin{array}{rcl} \mathtt{hd} & : & \mathbb{S}^{\mathbb{A}} \to \mathbb{A}; \\ \mathtt{tl} & : & \mathbb{S}^{\mathbb{A}} \to \mathbb{S}^{\mathbb{A}}. \end{array}$$

defined by

$$\begin{array}{rcl} \mathtt{hd}(\langle s_0, s_1, s_2, \ldots \rangle) & = & s_0; \\ \mathtt{tl}(\langle s_0, s_1, s_2, \ldots \rangle) & = & \langle s_1, s_2, s_3, \ldots \rangle. \end{array}$$

(iii) The *derivative* of s is $s' = \mathtt{tl}(s)$.

(iv) By definition two streams $s, t \in \mathbb{S}$ are equal iff $\forall n \in \mathbb{N}. s(n) = t(n)$.

8.4.7. DEFINITION (Coalgebra). Let $T : \mathsf{Set} \to \mathsf{Set}$ be a signature functor.

(i) A *T-coalgebra* is a pair $\langle X, \xi : X \longrightarrow TX \rangle$.

(ii) If $\langle X, \xi : X \longrightarrow TX \rangle$ and $\langle Y, \zeta : Y \longrightarrow TY \rangle$ are T-coalgebras, then a *T-coalgebra morphism* is a mapping $f : X \longrightarrow Y$ such that the following diagram commutes.

$$\begin{array}{ccc} T(X) & \xleftarrow{\xi} & X \\ {\scriptstyle T(f)}\downarrow & & \downarrow{\scriptstyle f} \\ T(Y) & \xleftarrow{\zeta} & Y \end{array}$$

(iii) A *final* T-coalgebra $\langle C, \mathtt{out} : C \longrightarrow T(C) \rangle$ is a final object in the category Set_T of T-coalgebras and T-coalgebra morphisms, i.e., for any T-coalgebra $\langle X, \xi : X \longrightarrow TX \rangle$ there is a unique T-coalgebra morphism $\varphi : X \longrightarrow C$.

We can define functions whose codomain is the final coalgebra by schemata which are dual to the familiar iteration and (primitive) recursion schemata:

8.4.8. PROPOSITION (Coiteration and primitive corecursion). **prop:prim-corec** *Let T be a functor in a category $\mathbb{C}$ and let $\langle C, \mathtt{out} \rangle$ be the final T-coalgebra. Then*

(i) *For all $\alpha{:}X \to T(X)$ there exists a unique $f{:}X \to C$ such that the following diagram commutes*

$$\begin{array}{ccc} T(C) & \xleftarrow{\mathtt{out}} & C \\ {\scriptstyle T(f)}\uparrow & & \uparrow{\scriptstyle f} \\ T(X) & \xleftarrow{\alpha} & X \end{array}$$

Figure 8.1: Coiteration

(ii) *For all $\alpha{:}X \to T(C + X)$ there exists a unique $f{:}X \to C$ such that the following diagram commutes*

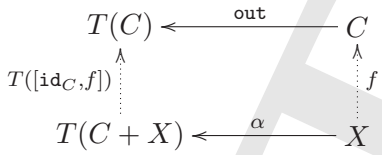


Figure 8.2: Primitive corecursion

PROOF. Do Exercise 8.7.20. ■

8.4.9. PROPOSITION. *For every polynomial functor T $\mathsf{Set} \to \mathsf{Set}$ there is a final coalgebra $\langle C, \mathtt{out}\rangle$ that is unique up to isomorphism in the category of T-coalgebras. Moreover, $\mathtt{out} : C \to T(C)$ is an isomorphism, hence $C \cong T(C)$.*

PROOF. (Sketch) Take C the projective limit of

$$1 \xleftarrow{f} T(1) \xleftarrow{T(f)} T^2(1) \xleftarrow{T^2(f)} \cdots$$

where $1 = \{\emptyset\}$ is the final element of Set and $f\ T(1) \to 1$ the canonical map. It consists of all sequences $\langle x_0, x_1, \ldots\rangle$ such that $\forall k \in \mathbb{N}. f^{k+1}(x_{k+1}) = x_k$, where $f^k = T^k(f)$. The morphism $\mathtt{out} : C \to T(C)$ is defined in Exercise 8.7.28. ■

Coinduction

For finite or infinite trees there is a principle of coinduction that enables a smooth way of proving equality, as we shall see in Lemma **??**. As a didactic warm-up we present here the principle for streams.

8.4.10. DEFINITION. A relation $R \subseteq \mathbb{S}^{\mathbb{A}} \times \mathbb{S}^{\mathbb{A}}$ is called a *bisimulation* iff for all $s, t \in \mathbb{S}^{\mathbb{A}}$

$$sRt \;\Rightarrow\; s_0 = t_0 \;\&\; s'Rt'. \tag{ci}$$

8.4.11. PROPOSITION (Principle of Coinduction). *Let R be a bisimulation over $\mathbb{S}$. Then for all $s, t \in \mathbb{S}$ one has*

$$sRt \;\Rightarrow\; s = t.$$

PROOF. Let R be a bisimulation and suppose sRt. Then by (ci) one has $s(0) = t(0)$ and $s'Rt'$. Again by (ci) one has $s(1) = s'(0) = t'(0) = s(1)$ and $s''Rt''$. Hence by ordinary induction $s(n) = t(n)$ for all $n \in \mathbb{N}$. Therefore $s = t$. ■

Although this Principle of Coinduction is obvious, one can use it to prove less obvious results. This is similar to the use of ordinary induction.

8.4.12. DEFINITION. Define maps $\mathtt{even}, \mathtt{odd} : \mathbb{S} \to \mathbb{S}$ and $\mathtt{zip} : \mathbb{S} \times \mathbb{S} \to \mathbb{S}$ as follows, *by corecursion.*

$$\begin{aligned}
(\mathtt{even}(s))(0) &= s(0)\\
(\mathtt{even}(s))' &= \mathtt{even}(s'')\\
(\mathtt{odd}(s))(0) &= s(1)\\
(\mathtt{odd}(s))' &= \mathtt{odd}(s'')\\
(\mathtt{zip}(s,t))(0) &= s(0)\\
(\mathtt{zip}(s,t))' &= \mathtt{zip}(t, s').
\end{aligned}$$

This has the effect that

$$\begin{aligned} \mathtt{even}(\langle a_0, a_1, \ldots\rangle) &= (\langle a_0, a_2, \ldots\rangle) \\ \mathtt{odd}(\langle a_0, a_1, \ldots\rangle) &= (\langle a_1, a_3, \ldots\rangle) \\ \mathtt{zip}(\langle a_0, a_1, \ldots\rangle, \langle b_0, b_1, \ldots\rangle) &= (\langle a_0, b_0, a_1, b_1, , \ldots\rangle). \end{aligned}$$

8.4.13. Application. For all $s, t \in \mathbb{S}$ one has

$$\mathtt{even}(\mathtt{zip}(s,t)) = s.$$

Proof. Define $R = \{\langle \mathtt{even}(\mathtt{zip}(s,t), s\rangle \mid s,t \in \mathbb{S}\}$. Then R is coinductive:

$$\begin{aligned} \mathtt{even}(\mathtt{zip}(s,t))(0) &= \mathtt{zip}(s,t)(0) \\ &= s(0); \\ \mathtt{even}(\mathtt{zip}(s,t))' &= \mathtt{even}(\mathtt{zip}(s,t)'') \\ &= \mathtt{even}(\mathtt{zip}(t,s')') \\ &= \mathtt{even}(\mathtt{zip}(s',t')) \\ R\ & s'. \end{aligned}$$

By definition one has $\mathtt{even}(\mathtt{zip}(s,t))\ R\ s$. Hence by the principle of coinduction it follows that $\mathtt{even}(\mathtt{zip}(s,t)) = s$. ■

The idea is that if we want to show that $s = t$, then we search for a coinductive R such that sRt. This is found by suitably enlarging the initial set $R_0 = \{\langle s, t\rangle\}$ so that it becomes a bisimulation.

8.5. Tree type-algebras

trees

We now introduce the type-algebra $\mathsf{Tr}_\infty^{\mathbb{A}}$ consisting of the finite and infinite trees over a set of atoms $\mathbb{A}$. These are type-algebras that solve arbitrary simultaneous recursions $\mathcal{R}$ over $\mathbb{TT}^{\mathbb{A}}$ in a unique way and induce a stronger notion of equivalence between types. For example, if $A = A \to b$ and $B = (B \to b) \to b$, then in Tr_∞ one has $A = B$.

We first need to recall the basic terminology needed to present (infinite) labelled trees by means of addresses and suitable labelings for them. If Σ is a set, then Σ^* denotes the set of all finite sequences of elements of Σ. The elements of Σ^* are usually called the *words* over the *alphabet* Σ. As usual concatenation is represented simply by juxtaposition; ϵ denotes the empty word.

8.5.1. Definition (Trees). **treesaspf** A *type-tree* (or just *tree*, for short) over $\mathbb{A}$ is a partial function

$$t : \{0,1\}^* \rightharpoonup \mathbb{A} \cup \{\to\} \cup \{\bullet\}$$

satisfying the conditions.

1. $\epsilon \in \mathrm{dom}(t)$ (i.e. a tree has at least the root node);

2. if $uv \in \mathrm{dom}(t)$, then also $u \in \mathrm{dom}(t)$
   (i.e. if a node is in a tree, then all smaller nodes are in that tree);

3. if $t(u) = \rightarrow$ then $u0, u1 \in \mathrm{dom}(t)$ and $uk \notin \mathrm{dom}(t)$, for $k \geq 2$
   (i.e. if $\rightarrow$ is given at a node, then it has exactly two successors);

4. if $t(u) \in \mathbb{A} \cup \{\bullet\}$ then $uv \notin \mathrm{dom}(t)$ for all $v \neq \epsilon$
   (i.e. labels other than '$\rightarrow$' occur only at endpoints).

The set of trees over $\mathbb{A}$ will be denoted by $\mathsf{Tr}_\infty^{\mathbb{A}}$. We define $\mathsf{Tr}_{\mathrm{F}}^{\mathbb{A}} \subseteq \mathsf{Tr}_\infty^{\mathbb{A}}$ to be the set of *finite* trees, i.e. those with finite domain. We will often write simply Tr_∞, Tr_{F} when $\mathbb{A}$ is understood or not relevant. Following Ariola and Klop the element $\bullet$ will be called the 'black-hole'.

8.5.2. Example. The function t defined on the (finite) domain $\{\epsilon, 0, 1\}$ as follows.

$t(\epsilon) = \rightarrow$.

$t(0) = \alpha$.

$t(1) = \bullet$.

represents the following tree

We say that the symbol $c \in \mathbb{A} \cup \{\rightarrow, \bullet\}$ occurs in tree t at node $u \in \Sigma^*$ if $t(u) = c$.

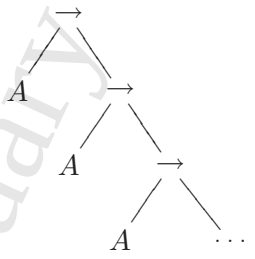
Since $\mathsf{Tr}_{\mathrm{F}}(\mathbb{A})$ is clearly isomorphic to $\mathsf{T}\!\!\mathsf{T}(\mathbb{A})$, we will often identify them, considering (and representing) simple types as finite trees and vice versa. The operation of substitution of types, for elements of $\mathbb{A}$, can be extended to trees. Among infinite trees, we single out trees having a certain periodic structure, which represent solutions of (systems of) equations of the form

$$\xi = A[\alpha{:=}\xi],$$

where $A \in \mathsf{Tr}_{\mathrm{F}}$. These will be called the regular trees.

8.5.3. Example. **infinite-tree** Take the equation $\xi = A \rightarrow \xi$ where A is any type. The solution of this equation is given by $\xi = t_0$ where t_0 is the following infinite tree.

$t_0 =$



Note that t_0 has only a finite number of distinct subtrees, namely A and t_0 itself. Such a tree is called *regular*.

8.5.4. DEFINITION (Regular trees). (i) Let $t \in \mathsf{Tr}_\infty$ be a tree and $w \in \mathrm{dom}(t)$ be a word. The *subtree of* t *at* w, notation $t|w$, is the tree defined by the following conditions.

(1) $\mathrm{dom}(t|w) = \{u \in \{0,1\}^* \mid wu \in \mathrm{dom}(t)\}$;

(2) $(t|w)(u) = t(wu)$, for all $u \in \mathrm{dom}(t|w)$.

A *subtree* of t is $t|w$ for some $w \in \mathrm{dom}(t)$.

(ii) A tree is *regular* if the set of its subtrees is finite. The set of regular trees is denoted by Tr_{R}.

Note that $\mathsf{Tr}_{\mathrm{F}} \subseteq \mathsf{Tr}_{\mathrm{R}} \subseteq \mathsf{Tr}_\infty$. Moreover, we will see that regular trees are closed under substitutions, i.e. if t, s are regular trees then so is $t[\alpha = s]$.

We consider $\mathsf{Tr}_\infty, \mathsf{Tr}_{\mathrm{R}}$ and Tr_{F} as type-algebras. Since $\to$ is injective, all of them are invertible.

8.5.5. LEMMA. **AllInvetible** $\mathsf{Tr}_\infty, \mathsf{Tr}_{\mathrm{R}}$ *and* Tr_{F} *are invertible type-algebras.*

8.5.6. DEFINITION. **TruncationDef** Given $t \in \mathsf{Tr}_\infty(\mathbb{A})$, define for every $k \in \omega$ the tree $t|_k \in \mathsf{Tr}_{\mathrm{F}}$, its *truncation* at level k, as follows.

1. $t|_0 = \bullet$;
2. $\alpha|_{k+1} = \alpha$, for $\alpha \in \mathbb{A}$;
3. $(t \to u)|_{k+1} = t|_k \to u|_k$.

8.5.7. EXAMPLE. Let $t = \alpha \to \beta$. Then

$$\begin{array}{lcll} (t)_0 & = & \bullet; & \\ (t)_1 & = & \bullet \to \bullet; & \\ (t)_n & = & t & \text{for } n \geq 2. \end{array}$$

8.5.8. DEFINITION. For $t, u \in \mathsf{Tr}_\infty$ define $t =_k u$ by $(t)_k = (u)_k$.

The following obvious fact will be useful.

8.5.9. LEMMA. **obvious-k** *Let* $t, u, t', u' \in \mathsf{Tr}_\infty$.

$$\begin{array}{llllll} \text{(i)} & t & = & u & \iff & \forall k \geq 0 \,.\, t =_k u. \\ \text{(ii)} & t & =_0 & u. & & \\ \text{(iii)} & t \to u & =_{k+1} & t' \to u' & \iff & t =_k u \;\&\; t' =_k u'. \end{array}$$

8.5.10. DEFINITION. Let $t, s \in \mathsf{Tr}_\infty$. The *substitution* of s for the occurrences of α in t, notation $t[\alpha = s]$, is defined as follows. (Note that α may occur infinitely many times in t.)

$$\begin{array}{lcll} t[\alpha = s](u) & = & s(w) & \text{if } u = vw \;\&\; t(v) = \alpha, \\ & & & \text{for some } v, w \in \Sigma^*; \\ & = & t(u) & \text{else.} \end{array}$$

Every invertible type-algebra can be mapped into a tree type-algebra, by a morphism determined by the "tree-unfolding" of the types in the algebra. Invertibility is needed in order to do the unfolding in a unique fashion. The construction has a flavour similar to the construction of Böhm-trees for untyped lambda terms, see Barendregt [1984], in that both have a co-inductive nature.

8.5.11. DEFINITION. **tree-translation** Let $\mathcal{A} = \langle |\mathcal{A}|, \rightsquigarrow \rangle$ be an invertible type algebra. Write $\mathsf{Tr}^{\mathcal{A}}_{\infty} = \mathsf{Tr}^{||\mathcal{A}||}_{\infty}$. The *tree unfolding* of a type $a \in \mathcal{A}$, notation $(a)^{\mathsf{tr}} = (a)^{\mathsf{tr}}_{\mathcal{A}}$, is defined as follows.

$$\begin{array}{rcll} (a)^{\mathsf{tr}}(\epsilon) & = & a, & \text{if } a \in ||\mathcal{A}||; \\ (a)^{\mathsf{tr}}(\epsilon) & = & \rightarrow, & \text{if } a = b_0 \rightsquigarrow b_1; \\ (a)^{\mathsf{tr}}(iw) & = & \uparrow, & \text{if } a \in ||\mathcal{A}||; \\ (a)^{\mathsf{tr}}(iw) & = & (b_i)^{\mathsf{tr}}(w) \quad (i = 0, 1), & \text{if } a = b_0 \rightsquigarrow b_1. \end{array}$$

8.5.12. REMARK. **intui-tree-translation** Intuitively the map $(-)^{\mathsf{tr}}_{\mathcal{A}}$ can be defined by

$$\begin{array}{rcll} (a)^{\mathsf{tr}} & = & a & \text{if } a \in ||\mathcal{A}||; \\ (a)^{\mathsf{tr}} & = & \rightarrow & \text{if } a = b_1 \rightsquigarrow b_2;. \end{array}$$

$$\begin{array}{ccc} & \rightarrow & \\ (b_1)^{\mathsf{tr}} & & (b_2)^{\mathsf{tr}} \end{array}$$

Note that this is not an inductive definition: a may be as complex as $a \rightsquigarrow b$.

At the end of this section we shall see that in fact $(-)^{\mathsf{tr}}$ is uniquely characterized as a function towards a final coalgebra, therefore its definition has a coinductive nature.

8.5.13. REMARK. **star-remark**

(i) Note that $(-)^{\mathsf{tr}}_{\mathcal{A}}$ is well defined since $\mathcal{A}$ is invertible.

(ii) The map $(-)^{\mathsf{tr}} = (-)^{\mathsf{tr}}_{\mathcal{A}} : \mathcal{A} \to \mathsf{Tr}^{\mathcal{A}}_{\infty}$ is a morphism.

(iii) In spite of the technicality, the construction of definition 8.5.11 is quite intuitive. The tree $(a)^{\mathsf{tr}}_{\mathcal{A}}$, which has the (names of the) prime elements of $\mathcal{A}$ as leaves, corresponds to the (possibly) infinite "unfolding" of a.

8.5.14. DEFINITION. **can-bisim** Let $\mathcal{A}$ be an invertible type-algebra.

(i) Define the relation $=^{\mathsf{tr}}_{\mathcal{A}} \subseteq \mathcal{A} \times \mathcal{A}$ by

$$a =^{\mathsf{tr}}_{\mathcal{A}} b \iff (a)^{\mathsf{tr}} = (b)^{\mathsf{tr}}.$$

If there is little danger of confusion we write $=^{\mathsf{tr}}$ for $=^{\mathsf{tr}}_{\mathcal{A}}$.

(ii) Define $\mathcal{A}^{\mathsf{tr}} = \mathcal{A}/=^{\mathsf{tr}}$.

(iii) For $a \in \mathcal{A}$ write $a^{\mathsf{tr}} = [a]_{=^{\mathsf{tr}}}$.

It is immediate from the definition that $=^{\mathsf{tr}}_{\mathcal{A}}$ is a congruence with respect to $\rightarrow$ and hence $\mathcal{A}^{\mathsf{tr}}$ is well defined.

8.5.15. LEMMA. **star-max** *Let $\mathcal{A}$ be an invertible type-algebra.*

(i) $=^{\mathsf{tr}}_{\mathcal{A}}$ *is the greatest invertible congruence over $\mathcal{A}$ such that for all $a \in ||\mathcal{A}||$*

$$[a]_{=^{\mathsf{tr}}} = \{a\}.$$

(ii) $\mathcal{A}^{\mathsf{tr}}$ *is an invertible type-algebra and* $||\mathcal{A}^{\mathsf{tr}}|| = \{[a]_{=^{\mathsf{tr}}} \mid a \in ||\mathcal{A}||\}$.

PROOF. (i) To prove invertibility note that $a_1 \to a_2 =^{\mathsf{tr}} b_1 \to b_2$ implies, by definition, $(a_i)^{\mathsf{tr}} = (b_i)^{\mathsf{tr}}$ $(i = 1, 2)$ and then $a_i =^{\mathsf{tr}} b_i$. Note also that if $a \in ||\mathcal{A}||$, then $a =^{\mathsf{tr}} b$ iff $a = b$. Hence $[a]_{=^{\mathsf{tr}}} = \{a\}$ and indeed $[a]_{=^{\mathsf{tr}}}$ is prime.

On the other hand, let now $\approx$ be any invertible congruence over $\mathcal{A}$ such that $\approx \not\subseteq =^{\mathsf{tr}}$. Then there are two elements a and b such that $a \approx b$ but $a \neq^{\mathsf{tr}} b$. In this case there must be some finite path w such that $(a)^{\mathsf{tr}}(w) \neq (b)^{\mathsf{tr}}(w)$. It is easy to show by induction on w and using invertibility that this implies that for some prime element $a' \in \mathcal{A}$ either $a' \approx b' \rightsquigarrow c'$ or $a' \approx d'$ for some other prime element d' of $\mathcal{A}$. In both cases the prime element a' is not preserved by $\approx$.

(ii) By (i). ■

8.5.16. PROPOSITION. **unique-hom** *Let $\mathcal{A}$ be an invertible type-algebra.*

(i) *There are canonical morphisms*

$$\begin{array}{rcccl} i_{\mathcal{A}} & : & \mathbb{T}^{||\mathcal{A}||} & \hookrightarrow & \mathcal{A} \\ f & : & \mathcal{A} & \to & \mathcal{A}/=^{\mathsf{tr}} \\ i_{\mathsf{tr}} & : & \mathcal{A}^{\mathsf{tr}} & \hookrightarrow & \mathsf{Tr}^{\mathcal{A}}_{\infty} \end{array}$$

with f surjective, $i_{\mathcal{A}}, i_{\mathsf{tr}}$ injective and $(\ )^{\mathsf{tr}} = i_{\mathsf{tr}} \circ f$; in diagram

$$\begin{array}{ccccc} \mathbb{T}^{||\mathcal{A}||} & \xrightarrow{i_{\mathcal{A}}} & \mathcal{A} & \xrightarrow{(\ )^{\mathsf{tr}}} & \mathsf{Tr}^{\mathcal{A}}_{\infty}, \\ & & {\scriptstyle f}\searrow & \nearrow{\scriptstyle i_{\mathsf{tr}}} & \\ & & \mathcal{A}/=^{\mathsf{tr}} & & \end{array}$$

(ii) *The maps in (i) are uniquely determined by postulating for all $a \in ||\mathcal{A}||$*

$$\begin{array}{rcl} i_{\mathcal{A}}(a) & = & a; \\ f(a) & = & [a]_{=^{\mathsf{tr}}}; \\ i_{\mathsf{tr}}([a]_{=^{\mathsf{tr}}}) & = & (a)^{\mathsf{tr}}. \end{array}$$

PROOF. (i) By Propositions 8.2.12 and 8.2.6(ii).

(ii) By the (easy) proof of these Propositions. ■

Summarizing, we have the following canonical morphisms.

Let $\mathcal{A}$ be an invertible type-algebra. Then the canonical morphisms i, h, j, g and $(-)^{\mathsf{tr}}_{\mathcal{A}}$ in the following commutative diagram

$$\begin{array}{ccccc} \mathbb{T}(||\mathcal{A}||) & \xrightarrow{i} & \mathcal{A} & \xrightarrow{(-)^{\mathsf{tr}}_{\mathcal{A}}} & \mathsf{Tr}^{\mathcal{A}}_{\infty} \\ & & \downarrow{\scriptstyle h} & & \downarrow{\scriptstyle j\ (\cong)} \\ & & \mathcal{A}^{\mathsf{tr}} & \xrightarrow[g = (-)^{\mathsf{tr}}_{\mathcal{A}^{\mathsf{tr}}}]{} & \mathsf{Tr}^{\mathcal{A}^{\mathsf{tr}}}_{\infty} \end{array}$$

are uniquely determined by

$$\begin{aligned} i(a) &= a, \\ (a)^{\mathsf{tr}}_{\mathcal{A}} &= a, \\ h(a) &= [a]_{=^{\mathsf{tr}}}, \\ g([a]_{=^{\mathsf{tr}}}) &= a, \\ j(a) &= [a]_{=^{\mathsf{tr}}}, \end{aligned}$$

where $a \in ||\mathcal{A}||$. The arrow j is an isomorphism because for $a, b \in ||\mathcal{A}||$ one has $a =^{\mathsf{tr}} b \;\Rightarrow\; a = b$.

In most cases we will work with $\mathcal{A} = \mathbb{T}/\approx$, a syntactic type algebra. In this case $\mathcal{A}^{\mathsf{tr}}$ can easily be characterized by an extension $\approx^*$ of $\approx$.

8.5.17. DEFINITION. **synt-star-rem** Let $\mathcal{A} = \mathbb{T}/\approx$ be a syntactic type algebra. Define $A \approx^* B$ if $([A])^{\mathsf{tr}}_{\mathcal{A}} = ([B])^{\mathsf{tr}}_{\mathcal{A}}$.

We have then (see lemma 8.2.19) $\mathcal{A}^{\mathsf{tr}} \cong \mathbb{T}/\approx^*$

It is easy to see that in this case Lemma 8.5.15 means that $\approx^*$ is the greatest invertible congruence extending $\approx$ and preserving all prime elements of $\mathcal{A}$. (see exercise 8.7.7).

[Henk: I left out trees as final coalgebras, since we do not like coalgebras any longer.]

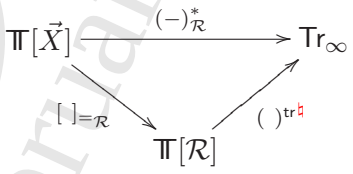
8.6. Tree equivalence for recursive types

The most interesting applications of the tree translations, Definition 8.5.11, are for syntactic type-algebras of the form $\mathcal{A} = \mathbb{T}/\approx$. In this case the syntactic morphism $(-)^{\mathsf{tr}\natural}_{\mathcal{A}}$ determined by $(-)^{\mathsf{tr}}_{\mathcal{A}}$ can be considered as an *interpretation* of the types in $\mathbb{T}$ as infinite trees.

We will discuss here in more details the cases in which the tree interpretation is applied to the type-algebras $\mathbb{T}[\mathcal{R}]$ and $\mathbb{T}_\mu$. We will prove in section Chapter 9 that both these type algebras are invertible; for $\mathbb{T}[\mathcal{R}]$ this is stated below without proof.

Tree equivalence for sr

Let $\mathcal{R}(\vec{X})$ be a proper sr over $\mathbb{T}$. Then $\mathbb{T}[\mathcal{R}] = (\mathbb{T}[\vec{X}]/=_{\mathcal{R}})$ is invertible by Theorem 9.3.7 and $(\ )^{\mathsf{tr}} = (\ )^{\mathsf{tr}}_{\mathbb{T}[\mathcal{R}]} : \mathbb{T}[\mathcal{R}] \to \mathsf{Tr}_\infty$. Write $(-)^*_{\mathcal{R}}$ for the map $(\ )^{\mathsf{tr}\natural}_{\mathbb{T}[\mathcal{R}]} = (\ )^{\mathsf{tr}\natural} \circ [\ ]_{=_{\mathcal{R}}}$:



There is a natural characterization of this map $(\ )^*_{\mathcal{R}}$.

8.6.1. LEMMA. **st-to-trees** *Let $\mathcal{R}(\vec{X})$ be a proper sr over $\mathbb{T}$. For $A \in \mathbb{T}[\vec{X}]$ the tree $(A)^*_{\mathcal{R}}$ can be characterized in the following way.*

(i) *If $A \equiv [\alpha]_{=_{\mathcal{R}}}$ for some atomic type α, then $(A)^*_{\mathcal{R}}$ is a leaf.*

(ii) *If $A \equiv B \to C$ for some $B, C \in \mathbb{T}[\vec{X}]$ then $(A)^*_{\mathcal{R}}$ is*

(iii) *If $A \equiv X$ for some $X \in \vec{X}$ and $X = B \in \mathcal{R}$ then $(A)^*_{\mathcal{R}} = (B)^*_{\mathcal{R}}$.*

PROOF. (i) Note that if $\alpha \in \mathbb{A}$, then $[\alpha]$ is a prime element of $\mathbb{T}[\mathcal{R}]$.

(ii) and (iii) follow by invertibility and the fact that $(-)^*_{\mathcal{R}}$ is a (syntactic) morphism. ■

Points (i), (ii) and (iii) above can be seen as a procedure to construct the (possibly infinite) tree $(A)^*_{\mathcal{R}}$.

8.6.2. NOTATION. (i) Write $=^*_{\mathcal{R}}$ for $=_{(-)^*_{\mathcal{R}}}$, the canonical congruence relation on $\mathbb{T}[\vec{X}]$ induced by $(-)^*_{\mathcal{R}}$.

(ii) Let $\mathcal{R} = \mathcal{R}[\vec{X}]$. Write $\mathbb{T}[\mathcal{R}]^* = \mathbb{T}[\vec{X}]/=^*_{\mathcal{R}}$.

(iii) In general we will use 'tr' to denote operations over generic type algebras and '*' to denote the corresponding operations at the syntactic level over types.

8.6.3. LEMMA. **IsoTreeStar** $\mathbb{T}[\mathcal{R}]^{\mathsf{tr}} \cong \mathbb{T}[\mathcal{R}]^*$.

PROOF. The morphisms $(-)^*_{\mathcal{R}}$ and $(-)^{\mathsf{tr}}_{\mathbb{T}[\mathcal{R}]}$ have the same range, say, Tr. Hence by Lemma 8.2.6(iii) one has

$$\mathbb{T}[\mathcal{R}]^* \cong \mathsf{Tr} \cong \mathbb{T}[\mathcal{R}]^{\mathsf{tr}}. \blacksquare$$

The isomorphism $h^{\mathsf{tr}} : \mathbb{T}[\mathcal{R}]^* \to \mathbb{T}[\mathcal{R}]^{\mathsf{tr}}$ is defined by

$$h^{\mathsf{tr}}([A]_{=^*_{\mathcal{R}}}) = ([A]_{=_{\mathcal{R}}})^{\mathsf{tr}}.$$

8.6.4. REMARK. If $\mathcal{R}$ is not proper we can have a equations of the shape $X = X$, or $X = Y, Y = X$. In this case the above procedure becomes circular due to point 3. Indeed, in these cases $[X]$, $[X, Y]$ are prime element with respect to $=_{\mathcal{R}}$, and should be considered as atomic types.

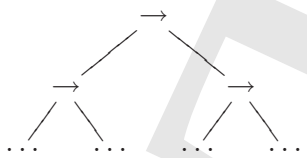
8.6.5. EXAMPLE. **strong-eq-ex**

(i) In both the sr $\mathcal{R}_0$ and $\mathcal{R}_1$ defined in Example 8.3.12 we have $X_i =^{\mathsf{tr}}_{\mathcal{R}_i} A \to X_i$ where $i = 0, 1$. In particular $(X_0)^*_{\mathcal{R}_0} = (X_1)^*_{\mathcal{R}_1} = t_0$ where t_0 is the tree defined in Example 8.5.3.

(ii) let $\mathcal{R}$ be the sr defined by the following equations.

$$\begin{aligned} X_1 &= X_2 \to X_1 \\ X_2 &= X_1 \to X_2 \end{aligned}$$

It is easy to see that both $(X_1)^*_{\mathcal{R}} = (X_2)^*_{\mathcal{R}} = t_1$ where t_1 is the following infinite tree

$t_1 =$ 

Therefore, $X_1 =^*_{\mathcal{R}} X_2$. Note that $X_1 \neq_{\mathcal{R}} X_2$. Also $(X)^*_{\mathcal{R}_2} = t_1$, where $\mathcal{R}_2 = \{X = X \to X\}$.

(iii) take the sr $\mathcal{R} = \{X = A \to X,\ Y = A \to Y\}$. We have that $X =^*_{\mathcal{R}} Y$. In fact both $(X)^*_{\mathcal{R}}$ and $(Y)^*_{\mathcal{R}}$ are equal to the tree t_0 defined in Example 8.5.3. Note that $X \neq_{\mathcal{R}} Y$

The mapping $(-)^*_{\mathcal{R}}$ is commutes with substitution. The proofs of the following lemmas are left as an exercise.

8.6.6. DEFINITION. Let $[A]$ denote the equivalence class of $A \in \mathcal{A}$ with respect to some equivalence relation over $\mathcal{A}$. Define

$$[A][[\alpha] := [B]] = \{A'[\alpha := B'] \mid A' \in [A],\ B' \in [B]\}.$$

This is indeed an equivalence class.

8.6.7. LEMMA. **SRCommWithSubS** *Let $\mathcal{R} = \mathcal{R}(\vec{X})$ and $A, B \in \mathbb{T}[\vec{X}]$. Then*

$$(A[\alpha\ = B])^*_{\mathcal{R}} = (A)^*_{\mathcal{R}}[\alpha\ = B_{\mathcal{R}^*}].$$

PROOF. ???. ■

[Henk: "This should be moved in sec. 8.3" where??]

8.6.8. LEMMA. **SRCommWithSubW** *Let $\mathcal{R} = \mathcal{R}(\vec{X})$, $A, B \in \mathbb{T}[\vec{X}]$, and $\alpha \notin \vec{X}$. Then $[A[\alpha\ = B]]_{=_{\mathcal{R}}} = [A]_{=_{\mathcal{R}}}[[\alpha]\ = [B]_{=_{\mathcal{R}}}]$.*

The type-algebra $\mathbb{T}[\mathcal{R}]^{\mathsf{tr}}$ solves $\mathcal{R}$.

8.6.9. THEOREM. **ta.extension*** *Let $\mathcal{R}(\vec{X})$ be an sr over $\mathbb{T}$, where $\vec{X} = X_1, \ldots, X_n$. Then we have*

(i) *$\mathbb{T}[\mathcal{R}]^* \models \mathcal{R}(\vec{\boldsymbol{X}})$, where $\vec{\boldsymbol{X}} = [X_1]_{=^*_{\mathcal{R}}}, \ldots, [X_n]_{=^*_{\mathcal{R}}}$.*

(ii) *Moreover, $\vec{\boldsymbol{X}}$ is the unique solution of $\mathcal{R}$ in $\mathbb{T}[\mathcal{R}]^*$.*

(iii) *$\mathbb{T}[\mathcal{R}]^*$ is generated by (the image of) $\mathbb{T}$ and the $\vec{\boldsymbol{X}}$.*

PROOF. (i) It is easy to see that $\vec{\mathbf{X}}$ solves $\mathcal{R}$ (apply 8.3.10(i) and 8.2.16(ii)).

(ii) By Lemma 8.3.22(i) for each solution of $\mathcal{R}$ in $\mathbb{T}[\mathcal{R}]^*$ there exists a morphism $h : \mathbb{T}[\mathcal{R}] \to \mathbb{T}[\mathcal{R}]^*$ which determines it. Then apply Lemmas 8.6.3 and 8.5.16.

(iii) This follows by Theorem 8.3.10(ii) and Lemma 8.5.16 (ii). ■

All trees in the codomain of $(-)^*_{\mathcal{R}}$ are regular.

8.6.10. LEMMA. **regular-solution** *Let $\mathcal{R}$ be an sr over $\mathbb{T}$. Then for all $A \in \mathbb{T}[\vec{X}]$ $(A)^*_{\mathcal{R}}$ is a regular tree, i.e. $(-)^*_{\mathcal{R}} : \mathbb{T}[\mathcal{R}] \to \mathsf{Tr}_{\mathrm{R}}$.*

PROOF. Let $\vec{X} = X_1, \ldots, X_n$ $(n \geq 0)$ and $\mathcal{R} = \{X_i = B_i \mid 1 \leq i \leq n\}$. If $A \in \mathbb{T}[\vec{X}]$ let $\mathcal{S}(A)$ denote the set of all subtypes of A. Obviously $\mathcal{S}(A)$ is finite for all A. Now it easy to prove by induction on w that for all $w \in \{0,1\}^*$ for which $(A)^*_{\mathcal{R}}(w)$ is defined we have

$$(A)^*_{\mathcal{R}}|w = (C)^*_{\mathcal{R}} \quad \text{for some} \quad C \in \mathcal{S}(A) \cup \bigcup_{1 \leq i \leq n} \{X_i\} \cup \bigcup_{1 \leq i \leq n} \mathcal{S}(B_i).$$

Since $\mathcal{S}(A)$ and all $\mathcal{S}(B_i)$ for $1 \leq i \leq n$ are finite, $(A)^*_{\mathcal{R}}$ can have only a finite number of subtypes and hence is regular. ■

The following theorem is an immediate consequence of this.

8.6.11. THEOREM. **sol-in-trees**
(i) *Let $\mathcal{R}(\vec{X})$, where $\vec{X} = X_1, \ldots, X_n$ $(n \geq 0)$, be a proper sr over $\mathbb{T}$. Then $\mathcal{R}$ has a unique solution in Tr_{R} given by $\vec{t} = (X_1)^*_{\mathcal{R}}, \ldots, (X_n)^*_{\mathcal{R}}$.*
(ii) *Tr_{R} (and hence Tr_∞) is a uniquely closed type algebra.*

PROOF. (i) By theorem 8.6.9 and 8.6.10.
(ii) By (i).

8.6.12. REMARK. **regular-solutions**
(i) On the other hand each regular tree can be obtained as a component of a solution of some sr $\mathcal{R}$ (exercise 8.7.10). See also **?**
(ii) It is easy to see that lemma 8.6.10 and theorem 8.6.11 holds also if assume more generally that $\mathcal{R}$ is an sr with coefficients in Tr_{R} (see also **?**).

8.6.13. REMARK. **UniqueStar** The mapping $(-)^*_{\mathcal{R}}$ can be seen as a *unifier* of $\mathcal{R}$ in Tr_{R}. It is well known (see **?**) that $(-)^*_{\mathcal{R}}$ is indeed the *most general* unifier of $\mathcal{R}$ in Tr_{R}. This means that each syntactic morphism $h : \mathbb{T}[\mathcal{R}] \to \mathsf{Tr}_{\mathrm{R}}$ can be written as $h = s \circ (-)^*_{\mathcal{R}}$ where s is a substitution in Tr_{R}. Note that any such substitution can be seen as a morphism $s : \mathsf{Tr}_{\mathrm{R}} \to \mathsf{Tr}_{\mathrm{R}}$.

The relation $=^*_{\mathcal{R}}$ has a more semantic nature than $=_{\mathcal{R}}$ and turns out to be the type equivalence induced by the interpretation of types in continuous models (see Chapter 11). Indeed, $=^*_{\mathcal{R}}$ can be also characterized as the equational theory of a suitable sr $\mathcal{R}^*$. In particular in Section 9 we will prove (in a constructive way) the following property.

8.6.14. THEOREM. **star-char** *Let $\mathcal{R}(\vec{X})$ be an sr over $\mathbb{T}$. Then there is an sr $\mathcal{R}^*(\vec{X})$ such that*
(i) *For all $A, B \in \mathbb{T}[\vec{X}]$ $A =^*_{\mathcal{R}} B$ iff $A =_{\mathcal{R}^*} B$.*
(ii) *$\mathbb{T}[\mathcal{R}]^* \cong \mathbb{T}[\mathcal{R}^*]$.*

In Section 9 we will give an alternative axiomatization for the relation $=^{\mathsf{tr}}_{\mathcal{R}}$ using a coinduction principle. Actually, simultaneous recursions of a special kind can directly be seen as coalgebras.

Tree equivalence for μ-types

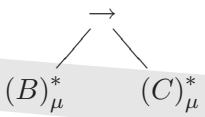
The construction of 8.5.11 can be applied to $\mathbb{T}_\mu = \mathbb{T}_\mu/=_\mu$, which is an invertible type-algebra by Theorem 9.2.4.

Similarly as before we denote the syntactic morphism $(-)^{\mathsf{tr}}_{\mathbb{T}_\mu}{}^{\natural}$ by $(-)^*_\mu$. By the definition of $(-)^*_\mu$ the equivalence class containing all circular types is a prime element of $\mathbb{T}_\mu$ and therefore it must be mapped to $[\mu\alpha.\alpha]$, a prime element of Tr_∞. We write $[\mu\alpha.\alpha] = \bullet \in \mathsf{Tr}_\infty$.Following remark 8.5.17 we will denote with $=^*$ the induced congruence on μ-types. Let's denote $\mathbb{T}^*_\mu$ the type-algebra $\mathbb{T}_\mu/=^*$.

Also $(-)^*_\mu$ has a natural characterization.

8.6.15. LEMMA. **mu-to-trees** *Let $A \in \mathbb{T}_\mu{}^{\mathbb{A}}$. The tree $(A)^*_\mu$ can be characterized in the following way.*

1. *If $A \equiv \alpha \in \mathbb{A}$, then $(A)^*_\mu$ is a leaf α;*
2. *If A is circular then $(A)^*_\mu$ is a leaf $\bullet$;*
3. *If $A \equiv B \to C$ for some $B, C \in \mathbb{T}_\mu$ then $(A)^*_\mu$ is*



4. *If $A \equiv \mu\alpha.B$ is contractive then $(A)^*_\mu = (A')^*_\mu$, where A' is the head μ-free form of A.*

PROOF. Similar to that of 8.6.1.

Also in this case points 1., 2., 3. and 4. above can be seen as a procedure to construct the (possibly infinite) tree $(A)^*_\mu$. The basic properties of $\mathbb{T}^*_\mu$ follow immediately from Lemmas 8.5.15 and 8.5.16. In particular $=^*$ is an invertible congruence extending $=_\mu$ and there is a unique morphism $h^* : \mathbb{T}_\mu/=_\mu \to \mathbb{T}^*_\mu/=_\mu$ defined by $h^*([A]_{=_\mu}) = [A]_{=^*}$.

8.6.16. EXAMPLE. **strongexample** If we consider again the types B and B' defined in example 8.3.27 we have that $B =^* B'$. In fact both $(B)^*$ and $(B')^*$ are equal to the infinite tree t of example 8.5.3.

Indeed, μ-types are just notations for regular trees, see **?**, Theorem 4.5.7.

8.6.17. LEMMA. **mu-iso-reg**

(i) *If $A \in \mathbb{T}_\mu$ then $(A)^*_\mu \in \mathsf{Tr}_{\mathrm{R}}$.*

(ii) *Each regular tree t can be written as $t = (A)^*_\mu$, for some type $A \in \mathbb{T}_\mu$.*

(iii) $\mathbb{T}^*_\mu \cong \mathsf{Tr}_{\mathrm{R}}$.

PROOF. (i) This follows from remark 8.6.12(i).

(ii) By the fact that $\mathbb{T}_\mu$ is a closed type-algebra (a formal proof of this will be given in Section 9).

(iii) Immediate from (i). ■

8.6.18. LEMMA. **CommWithSub** $(A[\alpha := B])^*_\mu = (A)^*_\mu[(\alpha)^*_\mu := (B)^*_\mu]$.

PROOF. Show by a routine induction on w that for all $w \in \{0,1\}^*$ one has $(A[\alpha := B])^*_\mu(w) = (A)^*_\mu[(\alpha)^*_\mu := (B)^*_\mu](w)$. ■

8.6.19. REMARK. Note that by lemma 8.6.18

$$(\mu\alpha.A)^*_\mu = (A[\alpha := \mu\alpha.A])^*_\mu = (A)^*_\mu[\alpha := (\mu\alpha.A)^*_\mu].$$

So $(\mu\alpha.A)^*_\mu$ is a solution of the equation $X = (A)^*_\mu[\alpha := X]$ in Tr_{R}. It is well known that if $A \neq \alpha$ this solution is unique (see **?**). This follows also from remark 8.6.12(ii)

In Chapter 9 we will give a complete axiomatization of $=^*$. As an application we obtain a constructive proof that $=^*$ is decidable.

8.7. Exercises

8.7.1. **kernel** Let $h : \mathcal{A} \to \mathcal{S}$ be a homomorphism. Define the *kernel* of h as $\mathtt{ker}(h) = \{a = b \mid h(a) = h(b)\}$.

(i) Show that $\mathtt{ker}(h)$ is always a congruence relation and hence

$$(a = b) \in \mathtt{ker}(h) \iff a =_{\mathtt{ker}(h)} b.$$

(ii) Let $h : \mathcal{A}[\mathcal{E}] \to \mathcal{S}$ be a homomorphism. Show

$$\mathcal{E} \subseteq \mathtt{ker}(h) \iff (=_{\mathcal{E}}) \subseteq \mathtt{ker}(h).$$

8.7.2. Let $\mathcal{E}$ be a binary relation on a type algebra $\mathcal{A}$. The following plays in the category of type algebras with homomorphisms.

(i) Show that a homomorphism $h : \mathcal{A}/\mathcal{E} \to \mathcal{S}$ is uniquely determined by a homomorphism $h^\natural : \mathcal{A} \to \mathcal{S}$ such that $\mathcal{E} \subseteq \mathtt{ker}(h)$ by $h([a]_{\mathcal{E}}) = h^\natural(a)$.

(ii) The canonical map $[-]_{\mathcal{E}} : \mathcal{A} \to \mathcal{A}/\mathcal{E}$ is an epimorphism having as kernel $=_{\mathcal{E}}$. Conversely, if $h : \mathcal{A} \to \mathcal{S}$ is an epimorphism and $\mathcal{E}$ is its kernel, then $\mathcal{S} \cong \mathcal{A}/\mathcal{E}$.

(iii) $\mathcal{A}/\mathcal{E}$ is up to isomorphism determined as the initial object such that $[-]_{\mathcal{E}} : \mathcal{A} \to \mathcal{A}/\mathcal{E}$ with kernel $\mathcal{E}$, i.e. if $\mathcal{S}$ is such that there is a $h : \mathcal{A} \to \mathcal{S}$ with kernel $\mathcal{E}$, then there is a unique arrow $k : \mathcal{A}/\mathcal{E} \to \mathcal{S}$ such that $k \circ [-]_{\mathcal{E}} = h$.

8.7.3. Suppose there is a homomorphism $h : \mathcal{A} \to \mathcal{S}$. Show that if $\mathcal{R}[\vec{X}]$ is sovable in $\mathcal{A}$, than also in $\mathcal{S}$.

8.7.4. Given a type algebra $\mathcal{A} = \mathbb{T}/\approx$ and a set $\mathcal{E}$ of closed equations over $\mathcal{A}$. Let $\mathcal{S} = \mathbb{T}/\approx'$ be also a type algebra. Then one has

$$\mathcal{S} \models \mathcal{E} \cup \approx \iff \exists h\, \mathcal{A} \to \mathcal{S} \text{ such that } \mathcal{E} \subseteq \mathtt{ker}(h).$$

8.7.5. **proper-proof** Let $\mathcal{R}$ be an arbitrary s.r. over $\mathbb{T} = \mathbb{T}(\mathbb{A})$. In which sense $\mathcal{R}$ can be transformed in a proper s.r.? Modify Theorem 8.3.17(**??**) and prove it.

8.7.6. **TypesTreesIso** Show that for every regular tree $T \in \mathsf{Tr}_{\mathrm{R}}$ the exists a s.r. $\mathcal{R}(\vec{X})$ over $\mathbb{TT}$ such that $t = (X_1)^*_{\mathcal{R}}$. [Hint. Let $t_1, \ldots, t_n$ be the distinct subtrees occurring in t and take $\vec{X} = X_1, \ldots, X_n$.]

8.7.7. **max-ex** Show that the relation $\approx^*$ introduced in Remark 8.5.17 is the greatest invertible congruence extending $\approx$ and preserving all prime elements.

8.7.8. **no-two-var-equal** Show that if $\mathcal{R}$ is a proper s.r. over $\mathbb{TT}(\mathbb{A})$ then $\alpha \neq_{\mathcal{R}} \beta$ for all $\alpha, \beta \in \mathbb{A}$.

8.7.9. **PrimeMu** Show that $||\mathbb{TT}_\mu|| = \{[\alpha] \mid \alpha \in \mathbb{A}\} \cup \{[\mu t.t]\}$.

8.7.10. **reg-as-solution** Show that for each regular tree t there is a proper s.r. $\mathcal{R}(\vec{X})$ over $\mathbb{TT}$ such that $t = t_1$ where $t_1 \ldots, t_n$ is the solution of $\mathcal{R}$ in Tr_{R}. [Hint. Take n indeterminates $X_1, \ldots, X_n$ corresponding to the n distinct subtrees of t.]

8.7.11. Let $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$ where $\mathcal{R}_0$ and $\mathcal{R}_1$ are defined as in example 8.3.12. Give a formal proof that $X_1 =^*_{\mathcal{R}} X'_1$.

8.7.12. **AllTermsTyped** Let U denote the type $\mu t.(t \to t)$. Show that, for all pure λ-terms M, $\Gamma_0 \vdash_{\lambda\mu} M : U$, where Γ_0 is the environment which assigns type U to all free variables of M.

8.7.13. **UrzyEx** [Urzyczyn] Let $\mathsf{K} = \lambda x.\lambda y.x$, $\mathbf{two} = \lambda f.\lambda x.f(f\ x)$ and $T_0 = \mu t.(v \to t)$. Show a deduction of $\vdash_{\lambda\mu} (\mathbf{two}\ \mathbf{two}\ \mathsf{K}) : T_0$. [Hint. Prove first that $\vdash_{\lambda\mu} \mathsf{K} : T_0 \to T_0$.] It can be proved (?) that $(\mathbf{two}\ \mathbf{two}\ \mathsf{K})$ is strongly normalizable but it has no type in the second-order typed lambda calculus (system F).

8.7.14. **dBtyping** Show the typed terms of corresponding to the deductions of Proposition 8.1.12.

8.7.15. **mu-intro-adm** Show that the rule

$$\frac{\Gamma \vdash M : A \quad \alpha \text{ does not occur in } \Gamma}{\Gamma \vdash M : \mu\alpha.A}$$

is admissible in $(\lambda\mu)$.

8.7.16. **ChurchTypingY** Inserting the *fold*_ and *unfold*_ constants at the proper types find a version Y_μ of the fixed-point operator Y which is well typed in $(\lambda\mu\text{-Ch}_0)$. Verify that assuming the reduction rule $(\mu\ 1)$ it can be proved that this has the same reduction properties of Y.

8.7.17. **mueqeta** Let $M = \lambda x^{\gamma\to\mu\alpha.\beta\to\alpha}.x^{\gamma\to\mu\alpha.\beta\to\alpha}$ be a term of type $(\gamma \to \mu\alpha.\alpha \to \beta) \to \gamma \to \mu\alpha.\alpha \to \beta$ in $(\lambda\mu\text{-Ch}_0)$. Find a Term M', obtained by adding occurrences of *fold* or *unfold* to some η-expansion of M such that M' has type $(\gamma \to \mu\alpha.\alpha \to \beta) \to \gamma \to \beta \to \mu\alpha.\alpha \to \beta$ in $(\lambda\mu\text{-Ch}_0)$.
In general Show that if M is a term of type A in $(\lambda\mu\text{-Ch})$ then there is a term M' of type A in $(\lambda\mu\text{-Ch}_0)$ such that $\widehat{M'} =_\eta M$ where $\widehat{M'}$ is the term obtained from M' by erasing all occurrences of *fold* and *unfold*.

8.7.18. **nofactorization**

(i) A type algebra $\mathcal{A}$ is not always generated by its prime elements. [Hint. If $\mathcal{A} = \{A\}$ with $A = A \to A$, then $||\mathcal{A}|| = \emptyset$.]
(ii) The embedding 8.2.12 is not always surjective. [Hint. Use (i).]
(iii) A morphism $h\, \mathcal{A} \to \mathcal{B}$ is not uniquely determined by $h \restriction ||\mathcal{A}||$.

8.7.19. **ackermann** This elaborates some points in the proof of Proposition 8.2.18.
(i) Show that the following is a 'cut-free' (no application of transitivity) axiomatization of $\vdash_{\mathcal{E}}$.

$$\text{(refl)} \quad \frac{}{A = A}\,;$$

$$(\to) \quad \frac{A_1 = B_1 \quad A_2 = B_2}{(A_1 \to A_2) = (B_1 \to B_2)}\,;$$

$$(\mathcal{E}[P,Q]) \quad \frac{A = P \quad B = Q}{A = B}\,, \qquad \text{if } (P = Q) \in \mathcal{E} \text{ or } (Q = P) \in \mathcal{E}.$$

(ii) Given a statement $A = B$, construct a tree (see figure 8.3) that describes all possible attempts of backwards derivations of $A = B$, following the cut-free rules. The nodes are labelled with equations $P = Q$ (expressing an equation to be proved) or a symbol '$\hookrightarrow$' or a pair $\mathcal{E}[P,Q]$ (expressing proof-steps in the cut-free system), with always P, Q being subtypes of a type occurring in $\mathcal{E} \cup \{A = B\}$, such that the following holds.
1. There are exactly two targets of the $\to$ or $\mathcal{E}[P,Q]$ nodes.
2. A node with label $P = Q$ is provable if both targets of at least one of its targets are provable (provability in the cut-free version of $\vdash_{\mathcal{E}}$). [The $P = Q$ nodes are called 'or-nodes' (having dotted lines) and the $\to$ and $\mathcal{E}[P,Q]$ are called 'and-nodes' (having solid lines).]
3. The tree path terminates at an or-node if the formula is $A = A$ (indicating local success) or if the formula is '$P = Q$' which is already below along the same path from the root (indicating local failure).

This tree is finite and contains all possible proofs of $A = B$ (if any exists). Check among all possible subtrees of this tree in which at an or-node only one successor, at an and-node both successors are chosen, whether at a terminating node one always has success. If there is at least one such subtree, then $A = B$ is provable in the cut-free system and hence in $\mathcal{E}$, otherwise it is not provable.

8.7.20. Prove Proposition 8.4.8

8.7.21. Show that the maps $\texttt{even}, \texttt{odd} : \mathbb{S}^{\mathbb{A}} \to \mathbb{S}^{\mathbb{A}}$ and $\texttt{zip} : \mathbb{S}^2 \to \mathbb{S}$ can be defined by coiteration.

8.7.22. Using (a slightly modified form of) coinduction, show that

$$\forall s \in \mathbb{S}.\mathtt{zip}(\mathtt{even}(s), \mathtt{odd}(s)) = s.$$

8.7.23. Let $F, G : \mathbb{S} \to \mathbb{S}$. Suppose

$$\forall s \in \mathbb{S}.(F\,s)(0) = (G\,s)(0)\ \&\ (F\,s)' = F(s')\ \&\ (G\,s)' = G(s').$$

Show by coinduction that

$$\forall s \in \mathbb{S}.F\,s = G\,s.$$

8.7.24. Define $\mathtt{map} : (\mathbb{A} \to \mathbb{A}) \to \mathbb{S} \to \mathbb{S}$ by

$$\begin{array}{rcl}(\mathtt{map}\ f\ s)(0) & = & f(s(0)),\\ (\mathtt{map}\ f\ s)' & = & \mathtt{map}\ f\ s'\end{array}$$

Using Exercise 8.7.23 show that

$$\forall f, g \in \mathbb{A} \to \mathbb{A} \forall s \in \mathbb{S}.\mathtt{map}\,g\,(\mathtt{map}\,f\,s) = \mathtt{map}\,(g \circ f)\,s.$$

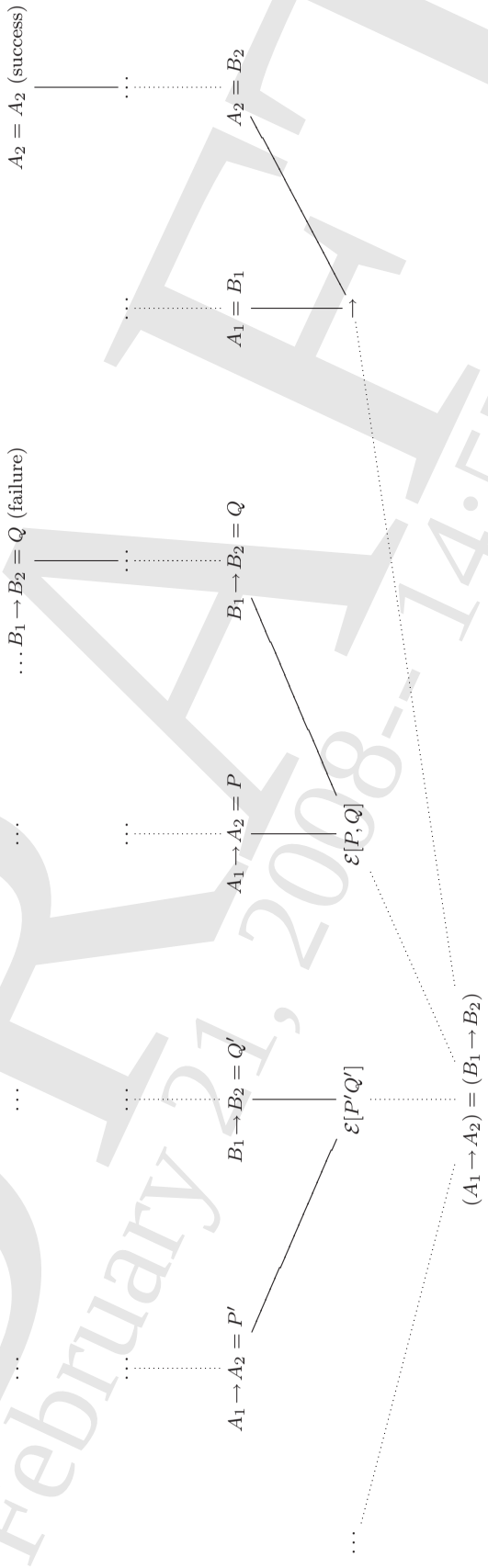


Figure 8.3: Exhaustive search for derivation in the cut-free version of $\vdash_{\mathcal{E}}$, in this case for the formula $A = B$ of the form $(A_1 \to A_2) = (B_1 \to B_2)$.

8.7.25. Define the coalgebra *lazy lists* consisting of finite and infinite lists. Is there a polynomial funtor for which this is the final coalgebra?

8.7.26. Construct by primitive corecursion a map $g\,\mathbb{S}^{\mathbb{A}} \to \mathbb{S}^{\mathbb{A}}$ such that $g(s) = \langle 0, s_0, s_1, s_2, \ldots\rangle$.

8.7.27. (**?**) For $s, t \in \mathbb{S}$ define the sum and convolution-product corecursively as follows.

$$\begin{array}{rcl}(s+t)(0) &=& s(0)+t(0),\\ (s+t)' &=& s'+t';\\ (s\times t)(0) &=& s(0)*t(0),\\ (s\times t)' &=& (s'\times t)+(s\times t').\end{array}$$

(i) Show the following for all $s, t, u \in \mathbb{S}$.

$$\begin{array}{rrcl}\text{(i)} & s+t &=& t+s;\\ \text{(ii)} & (s+t)+u &=& s+(t+u);\\ \text{(iii)} & (s+t)\times u &=& (s\times u)+(t\times u);\\ \text{(iv)} & s\times t &=& t\times s;\\ \text{(v)} & (s\times t)\times u &=& s\times(t\times u).\end{array}$$

[Hint. Use coinduction.]

(ii) Show that

$$\langle a_0, a_1, a_2, \ldots\rangle \times \langle b_0, b_1, b_2, \ldots\rangle = \langle c_0, c_1, c_2, \ldots\rangle,$$

with $c_n = \Sigma_{k=0}^{n} \binom{n}{k} a_k b_{n-k}$.

(iii) Show that

$$\langle 1, r, r^2, \ldots\rangle \times \langle 1, s, s^2, \ldots\rangle = \langle 1, r+s, (r+s)^2, \ldots\rangle.$$

[Hint. Do this coinductively, after first giving the right corecursive definition of $GS(r) = \langle 1, r, r^2, \ldots\rangle$.]

8.7.28. Let T be a polynomial functor in Set and let $C = \lim_{\leftarrow k}(T^k(1), f_k)$ with $f_k = T^k(f) : T^{k+1}(1) \to T^k(1)$ for the unique $f\,T(1) \to 1$. Define a map $\mathtt{out} : C \to T(C)$ that makes C into a cofinal coalgebra. [Hint. As T is polynomial, it is of the form $T(X) = \Sigma_n(A_n \times X^n)$, with n ranging over the natural numbers. We have that

$$C = \{(x_0, x_1, \ldots) | x_k : T^k(1)\ \&\ f_k(x_{k+1}) = x_k \text{ for every } k\}.$$

As $x_1 : T_1 = \sum_n A_n \times 1^n$, we have, say, $x_i = \langle n, a_n, \langle y_{11}, \ldots, y_{1n}\rangle\rangle$ for some $n \in \mathbb{N}$ and $a_n : A_n$, with all $y_{1i} = *$, the only element of 1. Since $f_1(x_2) = x_1$ with $e_1 = T(e_0)$, we must have $x_2 = \langle n, a_n, \langle y_{21}, \ldots, y_{2n}\rangle\rangle$, with $y_{2i} : T_1$ such that $e_0(y_{2i}) = y_{1i}$. Continuing we get

$$x_{k+1} = \langle n, a_n, \langle y_{(k+1)1}, \ldots, y_{(k+1)n}\rangle\rangle$$

with $y_{ki} : T^k(1)$, such that $f_k(y_{(k+1)i} = y_{ki}$, for every k. Now we can define $\mathtt{out}(x) \in T(C) = \Sigma_n(A_n \times C^n)$ by $x = \langle n, an, \langle y_1, \ldots, y_n\rangle\rangle$, with $y_i = \langle y_{i1}, y_{i2}, \ldots\rangle \in C$.]

Chapter 9

Properties of recursive types

21.2.2008:897

PropertiesOfTypes

In this Chapter we study the properties of recursive types independently of their use in typing λ-terms. Most of these properties will however be useful in the study of the properties of typed terms in the next Chapter.

We will discuss properties of μ-types and of systems of type equations in two separate sections, even if most of them are indeed similar. In Chapter 8 we have built a solution of an sr using μ-types, showing the substantial equivalence of the two notations for recursive types, although there are aspects which are treated more naturally in one approach rather than in the other. For instance, In Chapter 10, the notion of principal type scheme is formulated and studied in a more natural way using symultaneous recursion.

9.1. Simultaneous recursions vs μ-types

MuCTranslation The μ-types and the recursive types defined via an sr turn out to be essentially equivalent.

From sr to μ-types

First we show that types in any sr can be simulated (in a precise sense) by μ-types.

Take any sr $\mathcal{R} = \mathcal{R}(\vec{X})$ over $\mathbb{T}$. We show that $\mathbb{T}_\mu \models \exists \vec{X}.\mathcal{R}(\vec{X})$. We do this in a constructive way by building an n-tuple of types $S_1, \ldots, S_n \in \mathbb{T}_\mu$ such that $\mathbb{T}_\mu \models \mathcal{R}(\vec{S})$. This means that

$$\begin{array}{lll} S_1 & =_\mu & C_1[\vec{X} := \vec{S}] \\ & \cdots & \\ S_n & =_\mu & C_n[\vec{X} := \vec{S}] \end{array}$$

if $\mathcal{R}$ is of the form

$$\begin{array}{lll} X_1 & = & C_1(\vec{X}) \\ & \cdots & \\ X_n & = & C_n(\vec{X}). \end{array} \tag{1}$$

The following construction is taken from **?** and is known as 'folklore' of the subject.

9.1.1. DEFINITION. **systemsolution** Let $\mathcal{R}$ be an sr over $\mathbb{T}_\mu$ as (1) above. First define the types $\vec{P} = P_1, \ldots, P_n \in \mathbb{T}_\mu$ for $(1 \leq i \leq n\ )$ as follows.

$$\begin{array}{lcl} P_1 & = & \mu X_1.C_1 \\ P_2 & = & \mu X_2.(C_2[X_1 := P_1]) \\ & \vdots & \\ P_n & = & \mu X_n.(C_n[X_1 := P_1] \ldots [X_{n-1} := P_{n-1}]). \end{array}$$

Then define the types $\vec{S} = S_1, \ldots, S_n \in \mathbb{T}_\mu$ by

$$\begin{array}{lcl} S_n & = & P_n \\ S_{n-1} & = & P_{n-1}[X_n := S_n] \\ & \vdots & \\ S_1 & = & P_1[X_2 := S_2, \ldots, X_n := S_n]. \end{array}$$

Write $Sol_\mu(\mathcal{R}) = \vec{S} = S_1, \ldots, S_n$ for the intended solution of the $\mathcal{R}$ in $\mathbb{T}_\mu$.

9.1.2. EXAMPLE. **constraintexampletwo** Take the following sr $\mathcal{R}_1$:

$$\begin{array}{lcl} X_1 & = & X_2 \to X_1 \\ X_2 & = & X_1 \to X_2 \end{array}$$

Applying Definition 9.1.1 we have:

$$\begin{array}{lcl} P_1 & = & \mu X_1.(X_2 \to X_1) \\ P_2 & = & \mu X_2.((\mu X_1.(X_2 \to X_1)) \to X_2) \end{array}$$

and then

$$\begin{array}{lcl} S_2 & = & \mu X_2.((\mu X_1.(X_2 \to X_1)) \to X_2) \\ S_1 & = & \mu X_1.(\mu X_2.((\mu X_1'.(X_2 \to X_1')) \to X_2) \to X_1) \end{array}$$

It is easy to check that $S_1 =_\mu S_2 \to S_1$ and $S_2 =_\mu S_1 \to S_2$, but note that $S_1 \neq_\mu S_2$ as can be proved with the technique shown in Section 9.2.

9.1.3. REMARK. $\langle S_1, \ldots, S_n \rangle$ is not unique when the considered equivalence is $=_\mu$. In Example 9.1.2 $\langle S_2, S_1 \rangle$ also solves $\mathcal{R}_1$, as well as $\langle S, S \rangle$ where $S = \mu X.X \to X$. Note, however, that $S_1^* = S_2^* = S^*$.

9.1.4. THEOREM. **MuSimulation** *$Sol_\mu(\mathcal{R})$ is a solution of $\mathcal{R}$ in $\mathbb{T}_\mu$, i.e. for each $1 \leq i \leq n$ we have*

$$S_i =_\mu C_i[\vec{X} := \vec{S}].$$

PROOF. For notational simplicity we write down the proof for $n = 2$. The proof in this case is similar to the first proof of the Double Fixed-Point Theorem 6.5.1 in Barendregt [1984]. Let $\mathcal{R}(X_1, X_2)$ be

$$\begin{aligned} X_1 &= C_1[X_1, X_2], \\ X_2 &= C_2[X_1, X_2]. \end{aligned}$$

Write the solution of Definition 9.1.1 as follows.

$$\begin{aligned} P_1^{X_2} &= \mu X_1.P(X_1, X_2), \\ P_2 &= \mu X_2.C_2[P_1^{X_2}, X_2]; \\ S_2 &= P_2, \\ S_1 &= P_1^{S_2}. \end{aligned}$$

Then

$$\begin{aligned} S_2 &= P_2 \\ &= C_2[P_1^{P_2}, P_2], \\ &= C_2[S_1, S_2]; \\ S_1 &= P_1^{S_2}, \\ &= C_1[S_1, S_2]. \blacksquare \end{aligned}$$

We refer to the notations introduced in Def. 9.1.1. The proof is by induction on n. If $n = 1$ the proof is straightforward. In fact $P_1 = S_1 = \mu X_1.S_1 =_\mu S_1[X_1 := \mu X_1.S_1] = S_1[X_1 := S_1]$.
As for the induction step let $\mathcal{R}'$ be the sr obtained from $\mathcal{R}$ by removing the last equation and considering X_n as a variable. Then $\mathcal{R}'$, by induction hypothesis, has a solution $\{S'_1, \ldots, S'_{n-1}\}$ obtained via $\{P'_1, \ldots, P'_{n-1}\}$. Note that $P'_i = P_i$ for $1 \leq i \leq n-1$. Moreover it is easy to show that $S_i = S'_i[X_n := S_n]$ for $1 \leq i \leq n-1$. Now distinguish two cases.

- If $i < n$ then $S'_i =_\mu S_i[X_1 := S'_1, \ldots, X_{n-1} := S'_{n-1}]$ (by induction hypothesis) and this implies immediately, by Remark 8.3.28, that

$$\begin{aligned} S_i &= S'_i[X_n := S_n] \\ &=_\mu S_i[X_1 := S'_1, \ldots, X_{n-1} := S'_{n-1}][X_n := S_n] \\ &= S_i[X_1 := S_1, \ldots, X_n := S_n]. \end{aligned}$$

- If $i = n$, then note that

$$\begin{aligned} S_n &= \mu X_n.S_n[X_1 := P_1] \ldots [X_{n-1} := P_{n-1}] \\ &=_\mu S_n[X_1 := P_1] \ldots [X_{n-1} := P_{n-1}][X_n := S_n] \\ &= S_n[X_1 := P_1] \ldots [X_{n-1} := P_{n-1}[X_n := S_n], X_n := S_n] \\ &= S_n[X_1 := P_1] \ldots [X_{n-1} := S_{n-1}, X_n := S_n] \\ &\vdots \\ &= S_n[X_1 := S_1, \ldots, X_n := S_n]. \blacksquare \end{aligned}$$

Note that in the proof of Theorem 9.1.4 all the needed type equivalences can be obtained by axiom (μ-eq). No other rules of Definition 8.3.25 (for instance ($\to$-cong)) have been used. So the solution is one in a rather strong sense.

By Lemma 8.3.22(i) we have that for all $A, B \in \mathbb{T}[\vec{X}]$

$$A =_{\mathcal{R}} B \;\Rightarrow\; A[\vec{X} := \vec{S}] =_{\mu} B[\vec{X} := \vec{S}].$$

Note that the reverse implication is not true, in general. For instance, take $\mathcal{R}_0 = \{X_1 = X_1 \to X_1,\ X_2 = X_2 \to X_2\}$ Applying the construction of definition 9.1.1 we obtain $S_1 = S_2 = \mu X.(X \to X)$ but $X_1 \neq_{\mathcal{R}} X_2$.

9.1.5. Remark. **TypingTranslation** An immediate consequence of Lemmas 8.3.22 (**??**) and 8.1.15 is that all typing judgments proved in ($\lambda\mathcal{R}$) can be proved also in ($\lambda\mu$).

Note also that, by Lemma 8.2.16 (i), since there is a type algebra homomorphism $id : \mathbb{T}_{\mu} \to \mathbb{T}_{\mu}{}^{*}$ (see the remark after ??? **??**) any solution of an $\mathcal{R}$ with respect to $\mathbb{T}_{\mu}$ is also a solution with respect to $\mathbb{T}_{\mu}{}^{*}$.
The construction defined in Definition 9.1.1 gives also a solution of $\mathcal{R}$ w.r.t. the strong equivalence $=^{*}$. In particular it is easy to prove the following lemma.

9.1.6. Lemma. **EqualInInftree** *$Sol_{\mu}(\mathcal{R})$ solves $\mathcal{R}$ in $\mathbb{T}_{\mu}{}^{*}$, i.e. for all $1 \leq i \leq n$ $(S_i) =^{*} S_i[X_1 := S_1, \ldots, X_n := S_n]$.*

In the case of solutions modulo strong equivalence there are other constructions in the literature which give simpler solutions (see e.g. **?**).

From μ-type to sr

On the other side any μ-type can be represented in an equivalent way via an sr. The result is [problematic to state using equational equalities], but it can be nicely expressed in the case of strong equalities.

We define for each μ-type A an sr which generates the same infinite tree as A. In the following definition we assume, as usual, that all bound variables in A have different names, which are also different from the names of all free variables in A.

9.1.7. Definition. **SRSimulationDef** We associate to a $A \in \mathbb{T}_{\mu}$ a type $\mathcal{T}(A) \in \mathbb{T}$ and a sr $\mathcal{SR}(A)$ over $\mathbb{T}$ defined in the following way.

$$\begin{array}{ll} \mathcal{T}(\alpha) = \alpha & \mathcal{SR}(\alpha) = \emptyset \\ \mathcal{T}(A \to B) = \mathcal{T}(A) \to \mathcal{T}(B) & \mathcal{SR}(A \to B) = \mathcal{SR}(A) \cup \mathcal{SR}(B) \\ \mathcal{T}(\mu\alpha A) = \alpha & \mathcal{SR}(\mu\alpha A) = \mathcal{SR}(A) \cup \{\alpha = \mathcal{T}(A)\}. \end{array}$$

9.1.8. Theorem. **SRSimulationTh** $A^{\mathsf{tr}} = \mathcal{T}(A)^{\mathsf{tr}}_{\mathcal{SR}(A)}$

Proof. See Exercise 9.4.1. ■

9.2. Properties of μ-types

PropertiesOfMuTypes

In this section we investigate properties of μ-types both under the weak and the strong equivalence. In particular we show that weak equivalence is invertible and decidable. Also we give a complete formal system for proving strong equivalence and show that this relation is decidable.

Decidability and invertibility of weak equivalence

Both invertibility and decidability will be proved by defining a combinatory reduction system (CRS) which generates $=_\mu$ as its convertibility relation. For an introduction to term rewriting systems (TRS) and CRS see e.g. **?**, **?** and Terese [2003]

9.2.1. Definition. **ReductioOfMuTypes** Let $\Rightarrow_\mu \in \mathbb{T}_\mu \times \mathbb{T}_\mu$ be the reduction relation defined by:

1. $\mu\alpha.A \Rightarrow_\mu A[\alpha := \mu\alpha.A]$.
2. If $A \Rightarrow_\mu A'$ then
   - $A \to B \Rightarrow_\mu A' \to B$
   - $B \to A \Rightarrow_\mu B \to A'$
   - $\mu\alpha.A \Rightarrow_\mu \mu\alpha.A'$.

As usual ${\Rightarrow_\mu}^n$ denotes reduction in n steps and ${\Rightarrow_\mu}^*$ is the transitive and reflexive closure of $\Rightarrow_\mu$. A subexpression of a type B of the shape $\mu\alpha.A$ is called a *redex*. The *contraction* of a redex in B is obtained by replacing the redex by the r.h.s. of the corresponding rule. The proof of the following lemma is immediate (see exercise 9.4.3).

9.2.2. Lemma. **MuConv** *The relation $=_\mu$ is the convertibility relation generated by $\Rightarrow_\mu$.*

It is well known that $\langle \mathbb{T}_\mu, \Rightarrow_\mu \rangle$ is an orthogonal combinatory Term Rewriting System (see **?**) and hence Church-Rosser (CR).

9.2.3. Lemma. **CRmu** *the reduction relation $\Rightarrow_\mu$ is CR, i.e. if $A =_\mu B$ then there is a type $C \in \mathbb{T}_\mu$ such that $A \Rightarrow_\mu^* C$ and $B \Rightarrow_\mu^* C$.*

Note however that $\Rightarrow_\mu$ is not normalizing: types can be infinitely unfolded. The CR property however is enough to prove invertibility.

9.2.4. Theorem. **MuInvertibility** *$=_\mu$ is invertible, i.e. $A_1 \to B_1 =_\mu A_2 \to B_2$ implies $A_1 =_\mu A_2$ and $B_1 =_\mu B_2$.*

Proof. By Lemma 9.2.3 there is a type C such that $A_1 \to B_1 \Rightarrow_\mu^* C$ and $A_2 \to B_2 \Rightarrow_\mu^* C$. But then we must have $C \equiv C_1 \to C_2$ and so $A_i \Rightarrow_\mu^* C_1$ and $B_i \Rightarrow_\mu^* C_2$ for $i = 1, 2$. By 9.2.2 this implies $A_i =_\mu B_i$ for $i = 1, 2$.

To prove decidability we need some more properties of reduction systems. An important notion about them is that of standard reduction. We define this notion in the case of $\Rightarrow_\mu$.

9.2.5. DEFINITION. Let

$$R : A_1 \Rightarrow_\mu A_2 \Rightarrow_\mu \ldots \Rightarrow_\mu A_n$$

Assume that at each step $A_i \Rightarrow_\mu A_{i+1}$ we mark with $*$ all μ occurring in A_i to the left of the contracted μ and assume that all starred μ remain such through the remaining steps of R. Then R is *standard* if only non starred μ are reduced in it.

In other words a reduction is standard if reductions are performed from left to right (and from the outside in).

9.2.6. EXAMPLE. The following reduction is standard.

$$\begin{aligned}
\underline{\mu\alpha.\alpha\to\alpha} &\Rightarrow_\mu (\mu\alpha.\alpha\to\alpha)\to(\underline{\mu\alpha.\alpha\to\alpha})\\
&\Rightarrow_\mu (\mu^*\alpha.\alpha\to\alpha)\to(\underline{\mu\alpha.\alpha\to\alpha})\to\mu\alpha.\alpha\to\alpha\\
&\Rightarrow_\mu (\mu^*\alpha.\alpha\to\alpha)\to((\mu^*\alpha.\alpha\to\alpha)\to\underline{\mu\alpha.\alpha\to\alpha})\to\mu\alpha.\alpha\to\alpha\\
&\Rightarrow_\mu \ldots
\end{aligned}$$

where we have underlined the subterms corresponding to the last expanded redex.

The following standardization theorem for $\Rightarrow_\mu$ can be obtained as a particular case of the standardization theorem for CRS. See **??** for a proof.

9.2.7. LEMMA. **Standardization** *If $A \Rightarrow_\mu^* B$ then there is standard reduction from A to B.*

Let now $A \in \mathbb{T}_\mu$. We define the

9.2.8. DEFINITION. **SubtermClosureWeak** If $A \in \mathbb{T}_\mu$ its *subterm closure* $\mathcal{SC}(A) \subseteq \mathbb{T}_\mu$ is defined by cases in the following way.

1. $\mathcal{SC}(c) = \{c\}$ if c is a variable or a constant.
2. $\mathcal{SC}(A_1 \to A_2) = \{A_1 \to A_2\} \cup \mathcal{SC}(A_1) \cup \mathcal{SC}(A_2)$
3. $\mathcal{SC}(\mu\alpha.A) = \{\mu\alpha.A\} \cup \mathcal{SC}(A)[a := \mu\alpha.A]$

The following Lemma is proved in **?**.

9.2.9. LEMMA. **SCfinite** *For all types $A \in \mathbb{T}_\mu$ $\mathcal{SC}(A)$ is finite.*

Write $\vdash A = B$ if $A = B$ can be proved using the rules of the system (μ) in Definition 8.3.25(i). In the following definition we introduce a slightly different (but equivalent) system.

9.2.10. DEFINITION. **muweakeqminus** The system (μ^-) is defined as the system (μ) of Definition 8.3.25(i) by replacing (μ-eq), (symm) and (trans) by the following rules

$$(\mu\text{-tr-left}) \quad \frac{A[\alpha := \mu\alpha.A] = B}{\mu\alpha.A = B}$$

$$(\mu\text{-tr-right}) \quad \frac{A = B[\alpha := \mu\alpha.B]}{A = \mu\alpha.B}$$

We write $\vdash^- A = B$ if $A = B$ is provable in (μ^-).

9.2.11. LEMMA. **SystemEq**

$$\vdash A = B \ \ (\textit{i.e. } A \sim B) \quad \Leftrightarrow \quad \vdash^- A = B.$$

PROOF. The ($\Leftarrow$) direction is trivial since rules (μ-tr-left), (μ-tr-right) can be derived using (μ-eq) and (trans). As for ($\Rightarrow$), by Lemmas 9.2.2 we have A and B are convertible under $\Rightarrow_\mu$ and then, by the CR property, there exists a type C such that $A \Rightarrow_\mu{}^n C$ and $B \Rightarrow_\mu{}^m C$ for some $n, m \geq 0$. Moreover by the standardization theorem 9.2.7 we can assume that both reductions are standard. The proof is now by induction on $n + m$.
If $m = n = 0$ then $A \equiv B$ and we are done.
The induction step is by structural induction on A and B. The base step is represented by case 1. below, the induction step by cases 2. and 3.. Note that if one of the two types is atomic we are in case 1.

Case 1. Let $A \equiv \mu\alpha.A'$ and let $A \Rightarrow_\mu A'[\alpha := \mu\alpha.A'] \Rightarrow_\mu{}^{n-1} C$ (i.e. $n > 0$ and the first step in the reduction of A is an unfolding of the top level μ). By induction hypothesis we have $\vdash^- A'[\alpha := \mu\alpha.A'] = B$ and we can get $\vdash^- \mu\alpha.A' = B$ by rule (μ-tr-left). The symmetric case on B is handled similarly using (μ-tr-right).

Case 2. Let $A \equiv \mu\alpha.A'$, $B \equiv \mu\alpha.B'$ and the first step in the reduction of both A and B is not by unfolding the top level μ. In this case, since the reduction is standard we must have $C \equiv \mu\alpha.C'$ and both $A' \Rightarrow_\mu{}^n C'$ and $B' \Rightarrow_\mu{}^n C'$. By induction hypothesis on the structure of types we have $\vdash^- A' = B'$ and we can get $\vdash^- A = B$ by rule (μ-cong).

Case 3. Let $A \equiv A_1 \to A_2$. In this case B cannot be atomic. If $B \equiv \mu\beta.B'$ the first step in the reduction of B must be the reduction of a top level μ and we are in case 1. above. So let $B \equiv B_1 \to B_2$. Then we must have $C \equiv C_1 \to C_2$ and both $A_1 \Rightarrow_\mu{}^{n_1} C_1$, $A_2 \Rightarrow_\mu{}^{n_2} C_2$, $B_1 \Rightarrow_\mu{}^{m_1} C_1$ and $B_2 \Rightarrow_\mu{}^{m_2} C_2$ where $n = n_1 + n_2$ and $m = m_1 + m_2$ and then $n_1, n_2 \leq n$, and $m_1, m_2 \leq m$. By induction hypothesis then we have $\vdash^- A_1 = B_1$ and $\vdash^- A_2 = B_2$. So we can get $\vdash^- A = B$ by rule ($\to$-cong). ■

Deductions in $\vdash^-$ are interesting for the following reason.

9.2.12. LEMMA. **DeductionFinite**
(i) *Let D be a deduction of $\vdash^- A = B$ then all statements $A' = B'$ occurring in it are such that $A' \in \mathcal{SC}(A)$ and $B' \in \mathcal{SC}(B)$.*

(ii) *It is decidable whether* $\vdash^- A = B$.

PROOF. (i) By a straightforward induction on deduction.
(ii) Let $N = |\mathcal{SC}(A)|$ and $M = |\mathcal{SC}(B)|$. Let D be any deduction in $\vdash^- A = B$. By point (i) there are at most $M \cdot N$ possible distinct statements that can occur in D. So we can assume that D has no branch of length greater then $M \cdot N$, since otherwise we can reduce the size of D by replacing the innermost occurrence of a statement $A' = B'$ in a branch of length greater then $M \cdot N$ by the outermost occurrence of the same statement. So there are only a finite number of possible deductions for $\vdash^- A = B$.

By Lemma 9.2.11 then we have the decidability of $\sim$. Upperbound??

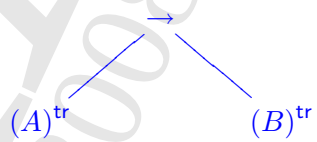
9.2.13. THEOREM. *It is decidable whether* $A =_\mu B$.

It is easy to devise an algorithm to check whether two types are weakly equivalent (see Exercise 9.4.2).

Decidability of strong equivalence

We will show that it is decidable wether given two types $A, B \in \mathbb{T}_\mu$ their unfolding as infinite trees are equal, i.e. $(A)^{\mathsf{tr}} = (B)^{\mathsf{tr}}$, also written as $A =^* B$. The decidability is due to **?**, with an exponential algorithm. The method was improved by Koster [??] and later **?**, bringing the complexity down to $O(n^2)$. We only present the exponential algorithm.

9.2.14. PROPOSITION. *The type algebra* $\mathbb{T}_\mu$ *is invertible.*

PROOF. By the fact that $(A \to B)^{\mathsf{tr}} =$  .

Hence $(A \to B)^{\mathsf{tr}} = (A' \to B')^{\mathsf{tr}}$ implies $(A)^{\mathsf{tr}} = (A')^{\mathsf{tr}}$ and $(B)^{\mathsf{tr}} = (B')^{\mathsf{tr}}$. ■

Clearly $A =^* B$ if at all nodes u the trees $(A)^{\mathsf{tr}}$ and $(B)^{\mathsf{tr}}$ have the same label.

9.2.15. DEFINITION. **SubtermClosureWeak** Let $A \in \mathbb{T}_\mu$. The *subterm closure* of A, notation $\mathcal{SC}(A) \subseteq \mathbb{T}_\mu$, is defined in the following way.

$$\begin{aligned} \mathcal{SC}(c) &= \{c\}, \text{ if } c \text{ is a variable or a constant;} \\ \mathcal{SC}(A_1 \to A_2) &= \{A_1 \to A_2\} \cup \mathcal{SC}(A_1) \cup \mathcal{SC}(A_2); \\ \mathcal{SC}(\mu\alpha.A) &= \{\mu\alpha.A\} \cup \mathcal{SC}(A)[\alpha := \mu\alpha.A]. \end{aligned}$$

9.2.16. LEMMA. **SCfinite** *For all types* $A \in \mathbb{T}_\mu$ $\mathcal{SC}(A)$ *is finite.*

PROOF. By induction on the generation of A. ■

The following proof is essentially from Grabmayer [2005], [2007].

9.2.17. THEOREM. *The relation* $=^* \subseteq \mathbb{T}_\mu^2$ *is decidable.*

PROOF. Let n_A, n_B be respectively the number of generated sybterms of A, B, which exists by Lemma 9.2.16. We claim it is enough to compare $(A)^{\mathsf{tr}}$ and $(B)^{\mathsf{tr}}$ until level $n = n_A.n_B + 1$. Indeed, if at some node before this level the trees have dissimilar subtrees, then clearly $A \neq^* B$. If on the other hand the trees $(A)^{\mathsf{tr}} =_n (B)^{\mathsf{tr}}$, but would have different labels at some node of a bigger depth k, then take such k minimal. The subtrees along the path from the root to this node (at depth k) must contain a repetition and therefore $(A)^{\mathsf{tr}}$ and $(B)^{\mathsf{tr}}$ differ also at a node of depth less than k, contradicting k's minimality. ■

Axiomatization of strong equivalence

AxiomatizationOfStrongEquivalence

While the notion of weak equivalence was introduced by means of a formal inference system, that of strong equivalence was introduced in Chapter 8.1 in a semantic way (via the interpretation of recursive types as trees). We show in this section that also strong equivalence can be represented by a (rather simple) finite set of formal rules, exploiting implicitly the proof principle of coinduction. The formal system BH and the proof that

$$\vdash_{BH} A = B \iff A =^* B$$

are taken from **?**. Other complete formalizations of strong equivalence have been given by **?**, and will be discussed below.

The formal presentation can be given for both μ-types and simultaneous recursions: the approach and the proof techniques are essentially the same. Below, we will carry out the proofs for μ-types. As a by-product of the proof of the completeness of this formalization we obtain a second proof of the decidability of strong equivalence, for both μ-types and simultaneous recursions.

In the system BH we have judgments of the form

$$\mathcal{H} \vdash A = B$$

in which $\mathcal{H}$ is a set of equations between types of the shape $A_1 \to A_2 = B_1 \to B_2$, where $A_1, B_1, A_2, B_2 \in \mathbb{T}_\mu$. The meaning of this judgment is that we can derive $A = B$ using the equations in $\mathcal{H}$. We will show that provability in BH from $\mathcal{H} = \emptyset$ corresponds exactly to strong equivalence.

9.2.18. DEFINITION. **CompleteStrongMuSystem** Let $\mathcal{H}$ denote a set of statements of the form $A_1 \to A_2 = B_1 \to B_2$, where $A_1, B_1, A_2, B_2 \in \mathbb{T}_\mu$. The system BH is defined by the following rules in Figure 9.1. We write $\mathcal{H} \vdash_{BH} A = B$ if $\mathcal{H} \vdash A = B$ can be derived by the rules of BH . Note that the rule ($\to$cong) can be derived from the rule (coind).

The set of rules for BH includes the rules of Definition 8.3.25. Rule (μ-cong) is needed only to prove equality of circular types, see Exercise 9.4.6. Rule (coind) is obviously the crucial one. It says that if we are able to prove $A = A'$

$$
\begin{array}{ll}
(\mu\text{-eq}) & \mathcal{H} \vdash \mu\alpha.A = A[\alpha := \mu\alpha.A] \\
(\text{ident}) & \mathcal{H} \vdash A = A \\
(\text{symm}) & \dfrac{\mathcal{H} \vdash A = B}{\mathcal{H} \vdash B = A} \\
(\mu\text{-cong}) & \dfrac{\mathcal{H} \vdash A = A'}{\mathcal{H} \vdash \mu\alpha.A = \mu\alpha.A'}\ , \ \alpha \text{ not free in } \mathcal{H}, \\
(\to\text{cong}) & \dfrac{\mathcal{H} \vdash A = A' \ \&\ \mathcal{H} \vdash B = B'}{\mathcal{H} \vdash (A \to B) = (A' \to B')} \\
(\text{trans}) & \dfrac{\mathcal{H} \vdash A = B \quad \mathcal{H} \vdash B = C}{\mathcal{H} \vdash A = C} \\
(\text{hyp}) & \mathcal{H} \vdash A = B \quad \text{if} \quad A = B \in \mathcal{H} \\
(\text{coind}) & \dfrac{\mathcal{H} \cup \{A \to B = A' \to B'\} \vdash A = A' \quad \mathcal{H} \cup \{A \to B = A' \to B'\} \vdash B = B'}{\mathcal{H} \vdash A \to B = A' \to B'}
\end{array}
$$

Figure 9.1: The system BH.

and $B = B'$ *assuming* $A \to B = A' \to B'$ then we can conclude that $A \to B = A' \to B'$. Note that rule ($\to$-cong) goes in the opposite direction.

It is through the rule (coind) that system BH exploits the coinductive characterization of equality of infinite trees. Given a formal derivation in system BH of a judgment of the form $\mathcal{H} \vdash A = B$, we can regard each application of rule (coind) as a step in the construction of a bisimulation that relates the infinite trees A^* and B^*.

As an example, let $S \equiv \mu\beta.\mu\alpha.\alpha \to \beta$, $T \equiv \mu\alpha.\alpha \to S$, and $B \equiv \mu\alpha.\alpha \to \alpha$. Observe that $S =_\mu T =_\mu T \to S$, and $B =_\mu B \to B$ but $S \neq_\mu B$. Furthermore, let $\mathcal{D}$ stand for the following derivation of $\mathcal{H} \vdash T = B$ with $\mathcal{H} = \{T \to S = B \to B\}$ (which amounts to assume $T = B$):

$$
\dfrac{\dfrac{\dfrac{}{\mathcal{H} \vdash T = T \to S}(\mu\text{-eq}) \quad \dfrac{}{\mathcal{H} \vdash T \to S = B \to B}(\text{hyp})}{\mathcal{H} \vdash T = B \to B}(\text{trans}) \quad \dfrac{\dfrac{}{\mathcal{H} \vdash B = B \to B}(\mu\text{-eq})}{\mathcal{H} \vdash B \to B = B}(\text{symm})}{\mathcal{H} \vdash T = B}(\text{trans})
$$

Consider now the following derivation in BH.

$$\dfrac{\dfrac{(\mu\text{-eq})}{S=T}\qquad \dfrac{\dfrac{(\mu\text{-eq})}{T=T\to S}\qquad \dfrac{\dfrac{\mathcal{D}}{\mathcal{H}\vdash T=B}\qquad \dfrac{\dfrac{(\mu\text{-eq})}{\mathcal{H}\vdash S=T}\qquad \dfrac{\mathcal{D}}{\mathcal{H}\vdash T=B}}{\mathcal{H}\vdash S=B}\,(\text{trans})}{T\to S=B\to B}\,(\text{coind})\qquad \dfrac{\dfrac{(\mu\text{-eq})}{B=B\to B}}{B\to B=B}\,(\text{symm})}{T\to S=B}\,(\text{trans})}{T=B}\,(\text{trans})}{S=B}\,(\text{trans})$$

Now we will prove the soundness and completeness

$$A =^* B \iff \vdash_{BH} A = B.$$

The proof occupies 9.2.19-9.2.27.

9.2.19. DEFINITION. **eqsSemantics**
(i) Let $A, B \in \mathbb{T}_\mu$ and $k \geq 0$. Define

$$A =^*_k B \text{ if } (A)^*|_k = (B)^*|_k .$$

(ii) A set $\mathcal{H}$ of formal equations of the form $A = B$ is said *k-valid* if $A =^*_k B$ for all $A = B \in \mathcal{H}$.
(iii) $\mathcal{H}$ is *valid* if $A =^* B$ for all $A = B \in \mathcal{H}$.

Note that $A =^* B \iff \forall k \geq 0 . A =^*_k B$, by Lemma 8.5.9(i).

9.2.20. LEMMA. **SoundnessUno** *Let $\mathcal{H}$ be valid. Then*

$$\mathcal{H} \vdash_{\text{BH}} A = B \;\Rightarrow\; A =^* B.$$

PROOF. It suffices to show for all $k \geq 0$ that if $\mathcal{H}$ is k-valid then

$$\mathcal{H} \vdash_{\text{BH}} A = B \;\Rightarrow\; A =^*_k B.$$

We use induction on k. If $k = 0$ this is trivial, by Lemma 8.5.9(ii). If $k > 0$ the proof is by induction on derivations. The most interesting case is when the last applied rule is (coind), i.e.

$$(\text{coind})\quad \frac{\mathcal{H} \cup \{A \to B = A' \to B'\} \vdash A = A' \quad \mathcal{H} \cup \{A \to B = A' \to B'\} \vdash B = B'}{\mathcal{H} \vdash A \to B = A' \to B'}$$

Since $\mathcal{H}$ is k-valid it is also $(k-1)$-valid. By the induction hypothesis on k we have $A \to B =^*_{k-1} A' \to B'$. But then $\mathcal{H} \cup \{A \to B = A' \to B'\}$ is also $k-1$ valid and hence, again by the induction hypothesis on k, we have $A =^*_{k-1} A'$ and $B =^*_{k-1} B'$. By Lemma 8.5.9(iii) we conclude $A \to B =^*_k A' \to B'$. ■

9.2.21. COROLLARY (Soundness). **SoundnessForSMu** $\vdash_{\text{BH}} A = B \;\Rightarrow\; A =^* B$.

PROOF. Take $\mathcal{H}$ empty. ■

The proof of completeness of BH is given in a constructive way. Below in Definition 9.2.22 we define a recursive predicate $\mathcal{S}(\mathcal{H}, A, B)$, where $\mathcal{H}$ is a set of equations and $A, B \in \mathbb{T}_\mu$. The relation $\mathcal{S}$ will satisfy

$$\mathcal{S}(\mathcal{H}, A, B) \;\Rightarrow\; \mathcal{H} \vdash_{\mathrm{BH}} A = B.$$

If A is a non-circular type its *head-μ-free form* is the type A' obtained by unfolding A at the top level until its main constructor is not the μ operator. Note that each non-circular type has a head-μ-free form $A' \in \mathcal{SC}(A)$ (see Definition 9.2.15). Moreover, if A' is the head-μ-free form of A we have for all $\mathcal{H}$ that $\mathcal{H} \vdash_{\mathrm{BH}} A = A'$ by (μ-eq) and (trans). Note that it is trivially decidable if a type A is non-circular or not. If A and B are circular we can easily prove $\vdash_{\mathrm{BH}} A = B$ (see exercise 9.4.6).

9.2.22. DEFINITION. **MuAlgorithm** Let A, B be two types and let $\mathcal{H}$ be a set of equations of the form $A_1 \to A_2 = B_1 \to B_2$. The predicate $\mathcal{S}(\mathcal{H}, A, B)$ is defined by the following clauses.

1. If both A and B are circular then $\mathcal{S}(\mathcal{H}, A, B) = \mathsf{true}$.
2. If A is circular and B is not, or vice versa, then $\mathcal{S}(\mathcal{H}, A, B) = \mathsf{false}$.
3. Otherwise let A' and B' denote, respectively, the μ-free form of A and B. We distinguish three cases:
   (a) If $A' \equiv B'$ or $A' = B' \in \mathcal{H}$ then $\mathcal{S}(\mathcal{H}, A, B) = \mathsf{true}$.
   (b) If A' and B' have different main type constructors, i.e. either are different atomic types, or one is atomic and the other is an arrow type then $\mathcal{S}(\mathcal{H}, A, B) = \mathsf{false}$.
   (c) Otherwise let $A' \equiv A_1 \to A_2$ and $B' \equiv B_1 \to B_2$. Then
   $$\begin{aligned}\mathcal{S}(\mathcal{H}, A, B) \;=\; & \mathcal{S}(\mathcal{H} \cup \{A_1 \to A_2 = B_1 \to B_2\}, A_1, B_1)\,\&\\ & \mathcal{S}(\mathcal{H} \cup \{A_1 \to A_2 = B_1 \to B_2\}, A_2, B_2).\end{aligned}$$

9.2.23. NOTATION. (i) Write

$$\begin{aligned}\mathcal{S}(\mathcal{H}, A, B) &\succ \mathcal{S}(\mathcal{H}', A_1, B_1)\\ \mathcal{S}(\mathcal{H}, A, B) &\succ \mathcal{S}(\mathcal{H}', A_2, B_2),\end{aligned}$$

where $\mathcal{H}' = \mathcal{H} \cup (A_1 \to A_2) = (B_1 \to B_2)$, if the value of $\mathcal{S}(\mathcal{H}, A, B)$ depends directly on that of $\mathcal{S}(\mathcal{H}', A_i, B_i)$ according to case 3c of the previous definition. Note that, owing to this case, $\succ$ is in general non-deterministic.

(ii) The notations $\succ^n$ and $\succ^*$ have the usual meaning: n-step rewriting and the reflexive transitive closure of $\succ$.

9.2.24. LEMMA. **TerminationOfSeqmu**

(i) *If* $\mathcal{S}(\mathcal{H}, A, B) \succ^* \mathcal{S}(\mathcal{H}', A', B')$, *then* $A' \in \mathcal{SC}(A)$, $B' \in \mathcal{SC}(B)$ *and all equations in* $\mathcal{H}'$ *not in* $\mathcal{H}$ *are of the form* $A'' = B''$ *where* $A'' \in \mathcal{SC}(A)$, $B'' \in \mathcal{SC}(B)$
(ii) *The relation* $\succ$ *is well-founded.*
(iii) *The predicate* $\mathcal{S}$ *decidable, i.e. seen as map* $\mathcal{S}$ *is a total and computable.*

PROOF. (i) If $\mathcal{S}(\mathcal{H}, A, B) \succ^* \mathcal{S}(\mathcal{H}', A', B')$, then

$$\mathcal{S}(\mathcal{H}, A, B) \succ^n \mathcal{S}(\mathcal{H}', A', B'),$$

for some n. The statement follows by induction on n.
(ii) By (i) there cannot exist an infinite sequence

$$\mathcal{S}(\mathcal{H}, A, B) \succ \mathcal{S}(\mathcal{H}_1, A_1, B_1) \succ \mathcal{S}(\mathcal{H}_2, A_2, B_2) \succ \dots .$$

Indeed, if $\mathcal{S}(\mathcal{H}_k, A_k, B_k) \succ \mathcal{S}(\mathcal{H}_{k+1}, A_{k+1}, B_{k+1})$, then $(A_{k+1} = B_{k+1}) \in \mathcal{H}_{k+1}/\mathcal{H}_k$, for otherwise by clause 3a the sequence stops, with the last item being true. By Lemma 9.2.16 all the possible $A'_k = B'_k$ form a finite set, so eventually the sequence must terminate.
(iii) By (ii) and Köning's lemma the evaluation of $\mathcal{S}(\mathcal{H}, A, B)$ must terminate in a finite conjunction of Booleans. From this the value is uniquely determined. ■

Each step of the definition of $\mathcal{S}(\mathcal{H}, A, B)$ which does not determine a $false$ value corresponds to the application of one or more deduction rules in BH. For instance point 1 corresponds to the proof, by (μ-eq) (μ-cong) and (trans) that two non circular types are equal and step 3c to the application of rule (coind).

9.2.25. LEMMA. **NotFail** *If* $\mathcal{S}(\mathcal{H}, A, B) = \mathsf{true}$, *then* $\mathcal{H} \vdash_{\mathrm{BH}} A = B$.

PROOF. By the remark just made. ■

9.2.26. LEMMA. **OnlyCorrectPairs** *If* $A =^* B$, *then* $\mathcal{S}(\mathcal{H}, A, B) = \mathsf{true}$.

PROOF. Let n be the maximum number such that $\mathcal{S}(\mathcal{H}, A, B) \succ^n \mathcal{S}(\mathcal{H}', A', B')$. By Lemma 9.2.24(ii) we know that such an n must certainly exist. The proof is by induction on n. If $n = 0$, then $A =^* B$ implies that $\mathcal{S}(\mathcal{H}, A, B) = \mathsf{true}$. If $n > 0$, then we are in case 3c of Definition 9.2.22. Let m_i be the maximum number of steps such that $\mathcal{S}(\mathcal{H}, A_i, B_i) \succ^{m_i} \mathcal{S}(\mathcal{H}', A'_i, B'_i)$, for $i = 1, 2$. We have that $m_i < n$. The proof follows immediately by induction hypothesis and the fact that $A =^* B$ implies $A_i =^* B_i$. ■

Now we can harvest.

9.2.27. THEOREM (Completeness). **CompletenessForSeqmu** *Let* $A, B \in \mathbb{T}_\mu$. *Then the following are equivalent.*
(i) $A =^* B$.
(ii) $\mathcal{S}(\emptyset, A, B) = \mathsf{true}$.
(iii) $\vdash_{\mathrm{BH}} A = B$.

PROOF. (i) ⇒ (ii). By Lemma 9.2.26.
(ii) ⇒ (iii). By Lemmas 9.2.25.
(iii) ⇒ (i). By Corollary 9.2.21. ■

Now we obtain for the second time the decidability of $=^*$, but the proof is essentially the same as that of Theorem 9.2.17.

9.2.28. COROLLARY. *Given $A, B \in \mathbb{T}_\mu$, it is decidable whether $A =^* B$.*

PROOF. By Theorem 9.2.27 and Lemma 9.2.24. ■

The predicate $\mathcal{S}$ defined in 9.2.22 is a computable procedure to test equality of $\mu-$types. In the present form $\mathcal{S}$ is $O(2^{n\times m})$ where n, m are, respectively, the number of arrow types in $\mathcal{SC}(A)$ and $\mathcal{SC}(B)$. However, more efficient algorithms are known (see e.g. **?**), which test equality of recursive types in quadratic time in the size of A and B.

Note that in the proof of $\vdash_{\mathrm{BH}} A = B$ implicit in the construction given in Lemma 9.2.25 and Definition 9.2.22 rule (→-cong) is not used. As mentioned before rule (μ-cong) is used only to prove equality of two circular types. Hence it could be replaced in the the system BH by a more specific rule asserting the equality of all circular types.

Other systems

Other systems have been proposed in the literature to give a complete axiomatization of $=^*$. In **?** a formal system is mentioned to prove strong equivalence (that we will denote by (μ^*AK)) defined by adding to the rules of system (μ) (see Definition 8.3.25) the following rule. The system is also used by **?**.

$$(\mathrm{AK})\quad \frac{\vdash A = B[\alpha := A]}{A = \mu\alpha.B} \quad \text{provided } B \text{ is non-circular in } \alpha$$

where we say that B is non-circular in α if it is not of the shape $\mu\alpha_1 \ldots \mu\alpha_k.\alpha$ with $k \geq 0$ and $\alpha \in \{\alpha_1, \ldots, \alpha_k\}$. The soundness of this system can be proved by a standard induction on derivations using the uniqueness of fixed-points in Tr_∞ (Theorem **??** [the theorem is disappeared. Should we make some remarks about fixpoints in Tr_∞? The notion is not used elsewhere]) when the last applied rule is (AK). In fact we have that both $(\mu\alpha.B)^*$ (by definition) and $(A)^*$ (by induction hypothesis) are fixed-points of $\lambda\zeta \in \mathsf{Tr}_\infty.(B)^*[\alpha := \zeta]$. In **?** is proved that also (μ^*AK) is complete with respect to the tree semantics. So (μ^*AK) is equivalent to BH.

Rule (AK) has indeed a great expressive power; it sometimes allows more synthetic proofs than rule (coind). The system presented in this section, however, uses a more basic proof principle (coinduction) and suggests a natural algorithm (obtained by going backward in the deduction tree) to test type equality.

9.2.29. EXAMPLE. In (μ^*AK) we have $\vdash \mu\alpha.(\beta\to\alpha)=\mu\alpha.(\beta\to\beta\to\alpha)$. Indeed, by two applications of (μ-eq)we have $\vdash \mu\alpha.(\beta\to\alpha)=\beta\to\beta\to\mu\alpha.(\beta\to\alpha)$. Then we can apply rule (AK) with $B[\alpha]=\beta\to\beta\to\alpha$.Compare his proof with that of Exercise 9.4.8.

Some general properties of strong equality can be easily proved by rule (AK). As an example of this, in the following proposition we show that two consecutive applications of the μ-operator can be contracted into a single one.

9.2.30. PROPOSITION. **MuContraction** *The following are directly provable by the rule (AK).*

$$\begin{array}{lrcll} \text{(i)} & \mu\alpha.A & = & A, & \text{if } \alpha \text{ does not occur in } A;\\ \text{(ii)} & \mu\alpha\beta.A & = & \mu\beta\alpha.A; & \\ \text{(iii)} & \mu\alpha\beta.A(\alpha,\beta) & = & \mu\alpha.A(\alpha,\alpha). & \end{array}$$

PROOF. Do exercise 9.4.16. ■

9.3. Properties of types defined by an sr

In this section we will study some fundamental properties of type algebras defined via type equations and simultaneous recursions over a set $\mathbb{T}$ of types. All results can however be easily generalized to arbitrary type algebras. Some properties of types defined in this way are essentially the same as those of μ-types, but their proofs require sometimes slightly different techniques.

Decidability and invertibility of weak equivalence for an sr

Decidability and invertibility for recursive types defined by a sr $\mathcal{R}$ can be proved via the construction of a Church-Rosser and strongly normalizing term rewriting system which generates $=_{\mathcal{R}}$. Indeed decidability of $=_{\mathcal{R}}$ follows also as a particular case of Proposition 8.2.18. The proof given here is taken from Statman **?**. A more algebraic but less direct approach is given in Marz **?**.

The proof of invertibility is based on the idea of associating to a sr a suitable Term Rewriting System (see **?**). A first way of doing this is that of orienting the equations of $\mathcal{R}$.

9.3.1. DEFINITION. Let $\mathcal{R}=\{X_i=A_i \mid 1\le i\le n\}$ be a proper sr over $\mathbb{T}$. The rewriting system Trs($\mathcal{R}$) consists of all rewriting rules $A_i \Rightarrow X_i$ for all $X_i=A_i\in\mathcal{R}$.

Note that $=_{\mathcal{R}}$ is the convertibility relation over $\mathbb{T}[\vec{X}]\times\mathbb{T}[\vec{X}]$ generated by Trs($\mathcal{R}$).

It is easy to see that Trs($\mathcal{R}$) is strongly normalizing, because each contraction decreases the size of the type to which it is applied and, if $X_j\Rightarrow X_i$ we have $j>i$ (by Definition 8.3.14). However Trs($\mathcal{R}$) it is not, in general, Church-Rosser, as we show in Example 9.3.2 below. We can however transform the given sr into an equivalent one which is Church-Rosser, via a construction which amounts to Knuth-Bendix completion (see **?**).

9.3.2. EXAMPLE. Let $\mathcal{R}$ be the sr

$$\begin{aligned} X_0 &= X_0 \to X_2 \\ X_1 &= (X_0 \to X_2) \to X_2 \\ X_2 &= X_0 \to X_1 \end{aligned}$$

Then $\mathrm{Trs}(\mathcal{R})$ consists of the rules

$$\begin{aligned} X_0 \to X_2 &\Rightarrow X_0 \\ (X_0 \to X_2) \to X_2 &\Rightarrow X_1 \\ X_0 \to X_1 &\Rightarrow X_2. \end{aligned}$$

Observe that the l.h.s. of the first equation is a subterm of the l.h.s. of the second one. In particular $(X_0 \to X_2) \to X_2$ can be reduced both to X_1 and to $X_0 \to X_2$ which further reduces to X_0: it has then two distinct normal forms X_1 and X_0. Therefore $\mathrm{Trs}(\mathcal{R})$ is not confluent (i.e., does not have the Church-Rosser property).

Expressions like $X_0 \to X_2$ and $(X_0 \to X_2) \to X_2$ in the example above are called *critical pairs* in the literature on term rewriting systems. In $\mathrm{Trs}(\mathcal{R})$ there is a critical pair whenever there are i, j such that $i \neq j$ and A_i is a subexpression of A_j. By a well known result of Knuth and Bendix (see **?**) a strongly normalizing term rewriting system without critical pairs is Church-Rosser.

We give now an algorithm, which amounts to the Knuth-Bendix completion algorithm, for transforming any proper sr into a logically equivalent one (see Definition 8.3.15 (ii)) without critical pairs.

Given a proper sr $\mathcal{R}$ we define two sequences of sets of equations $\mathcal{D}_n$, $\mathcal{I}_n$, where $\mathcal{D}_n$ is a proper sr and $\mathcal{D}_n \cup \mathcal{I}_n$ is a set of equations of the form $X_i = X_j$ with $i < j$. The sets $\mathcal{D}_n$ and $\mathcal{I}_n$ are defined in such a way that, for all n, $\mathcal{D}_n \cup \mathcal{I}_n$ is logically equivalent to $\mathcal{R}$.

9.3.3. DEFINITION. (i) Let $\mathcal{R}$ be a proper sr. We define by induction on n a sequence of sets of equations $\mathcal{D}_n$, $\mathcal{I}_n$ $(n \geq 0)$.
Let $\mathcal{D}_0 = \mathcal{R}$ and $\mathcal{I}_0 = \emptyset$.
Define $\mathcal{D}_{n+1}$, $\mathcal{I}_{n+1}$ from $\mathcal{D}_n$, $\mathcal{I}_n$ $(n \geq 0)$ in the following way:

(1) if there exists a pair of equations $X_i = A_i, X_j = A_j \in \mathcal{D}_n$ such that A_j is a proper subexpression of A_i take

$$\begin{aligned} \mathcal{D}_{n+1} &= (\mathcal{D}_n - \{X_i = A_i\}) \cup \{X_i = A_i^*\} \\ \mathcal{I}_{n+1} &= \mathcal{I}_n. \end{aligned}$$

where A_i^* is the result of replacing all occurrences of A_j in A_i by X_j.

(2) If there exist two equations $X_i = A, X_j = A \in \mathcal{D}_n$ then, assuming $i < j$, take

$$\begin{aligned} \mathcal{D}_{n+1} &= \mathcal{D}_n[X_i := X_j] \\ \mathcal{I}_{n+1} &= \mathcal{I}_n \cup \{X_i = X_j\}. \end{aligned}$$

(3) otherwise take $\mathcal{D}_{n+1} = \mathcal{D}_n$ and $\mathcal{I}_{n+1} = \mathcal{I}_n$

(ii) Let N be the least n such that $\mathcal{D}_{n+1} = \mathcal{D}_n$. Define
-$\mathcal{R}^\diamond = \mathcal{D}_N \cup \mathcal{I}_N$
-$\mathrm{Trs}^\diamond(\mathcal{R}) = \mathrm{Trs}(\mathcal{R}^\diamond)$

Note that N is well defined since, in both steps 1 and 2 of Definition 9.3.3, the total number of symbols in $\mathcal{D}_n$ strictly decreases. So we must eventually reach a value of n for which neither 1 nor 2 apply. In the above definition note that in $\mathcal{D}_n$ there can be several pairs of equations satisfying 1 or 2, so the $\mathcal{D}_n$'s and $\mathcal{I}_n$'s are not uniquely determined.

9.3.4. Lemma. *Let $\mathcal{R}$ be a proper sr.*
(i) *$\mathcal{R}^\diamond$ is a proper sr logically equivalent to $\mathcal{R}$;*
(ii) *$Trs^\diamond(\mathcal{R})$ is strongly normalizing and Church-Rosser.*

Proof. (i)We prove by a straightforward induction on n that for all $n \geq 0$ $\mathcal{D}_n \cup \mathcal{I}_n$ is a proper sr logically equivalent to $\mathcal{R}$.
(ii) First note that $\mathrm{Trs}^\diamond(\mathcal{R})$ is strongly normalizing since each reduction of a rule belonging to $\mathrm{Trs}(\mathcal{D}_N)$ reduces the size of the expression to which it is applied and each rule of the shape $X_j \Rightarrow X_i$ is such that $X_i < X_j$ and so no infinite sequence of such reductions is possible.

Now note that $\mathrm{Trs}^\diamond(\mathcal{R})$ has no critical pairs. In fact $\mathrm{Trs}(\mathcal{D}_N)$ has no such pairs, otherwise we could apply step 1 or 2 of Definition 9.3.3 to $\mathcal{D}_N$. Moreover if $X_j = X_i \in \mathcal{I}_N$ then X_j does not occur in $\mathcal{D}_N$ and there is no other equation of the form $X_j = X_{i'}$ in $\mathcal{I}_N$. In fact, if $X_j = X_i$ has been put in $\mathcal{I}_k$ at step 2 of Definition 9.3.3, for some $(0 < k \leq N)$, then X_j does not occur in $\mathcal{D}_k$, hence not in $\mathcal{D}_n$ for all $n \geq k$. Consequently no other equation containing X_j can be put in any $\mathcal{I}_n$ for $n > k$.
By the Knuth-Bendix theorem then $\Rightarrow$ is Church-Rosser. ■

9.3.5. Example. Applying the above algorithm to the sr $\mathcal{R}$ defined in Example 9.3.2 we have

$$\mathcal{D}_1 = \{X_0 = X_0 \to X_2,\ X_1 = X_0 \to X_2,\ X_2 = X_0 \to X_1\}$$
$$\mathcal{I}_1 = \emptyset$$

and

$$\mathcal{D}_2 = \{X_0 = X_0 \to X_2,\ X_2 = X_0 \to X_0\}$$
$$\mathcal{I}_2 = \{X_1 = X_0\}$$

no more transformations are possible, so we have $N = 2$. Note that $\mathcal{D}_2 \cup \mathcal{I}_2$ is logically equivalent to $\mathcal{R}$ and has no critical pairs.

Since $\mathcal{R}^\diamond$ is logically equivalent to $\mathcal{R}(\vec{X})$ we have that $=_\mathcal{R}$ is also the convertibility relation over $\mathbb{T}[\vec{X}] \times \mathbb{T}[\vec{X}]$ generated by $\mathrm{Trs}^\diamond(\mathcal{R})$. Now, given a pair of types $A, B \in \mathbb{T}[\vec{X}]$, we can easily decide whether $A =_\mathcal{R} B$ simply by checking that their (unique) normal forms with respect to $\mathrm{Trs}^\diamond(\mathcal{R})$ are identical.

9.3.6. COROLLARY. *Let $\mathcal{R}$ be a proper sr. Then $=_{\mathcal{R}}$ is decidable.* ■

Another application of the properties of $\mathrm{Trs}^{\diamond}(\mathcal{R})$ is also the proof of invertibility of an sr.

9.3.7. THEOREM. *Let $\mathcal{R}$ be a proper sr. Then $=_{\mathcal{R}}$ is invertible.*

PROOF. Let $A \to B =_{\mathcal{R}} A' \to B'$. Let T_N denote the normal form with respect to $\mathrm{Trs}^{\diamond}(\mathcal{R})$ of type T where T is one of $A,\ B,\ A',\ B'$. Then $T =_{\mathcal{R}} T_N$ for all T and $A_N \to B_N =_{\mathcal{R}} A'_N \to B'_N$. Now if $A_N \to B_N$ is itself in normal form then, by uniqueness of normal forms, we must have $A_N \to B_N \equiv A'_N \to B'_N$ and, then $A_N \equiv A'_N$ and $B_N \equiv B'_N$. By transitivity we have therefore $A =_{\mathcal{R}} A'$ and $B =_{\mathcal{R}} B'$.
Otherwise, if $A_N \to B_N$ is not in normal form, it must itself be a redex with respect to $\mathrm{Trs}^{\diamond}(\mathcal{R})$. Then $A_N \to B_N \Rightarrow X_i$ for some atomic type X_i in a single step. Applying the same argument to $A'_N \to B'_N$ and by uniqueness of normal form we have also $A'_N \to B'_N \Rightarrow X_i$ and, since $\mathrm{Trs}^{\diamond}(\mathcal{R})$ has no critical pairs, this implies $A_N \to B_N \equiv A'_N \to B'_N$. We conclude the proof as in the previous case. ■

Strong equivalence as an equational theory

In this section we show that for every sr $\mathcal{R}$ we can constructively find another sr $\mathcal{R}^*$ such that $=^*_{\mathcal{R}}$ coincides with the equational theory of $\mathcal{R}^*$. This allows to shift to tree type algebras generated by a sr all the results and the techniques valid for the equational theories. To this aim we need some preliminary definition

9.3.8. DEFINITION. A sr $\mathcal{R}[\vec{X}]$ is *flat* form if all equations of $\mathcal{R}$ are of the form $X = \alpha$ where α is an atomic type or $X = Y \to Z$, where $X,\ Y,\ Z \in \vec{X}$

Note that a sr in flat form is simplified. Any proper sr can be transformed in an equivalent one in flat form by adding or removing, if needed, new indeterminates.

9.3.9. LEMMA. *Each sr can be transformed into an equivalent one in flat form.*

PROOF. Let $\mathcal{R}(\vec{X})$ be a proper sr over $\mathbb{T}$. First transform $\mathcal{R}(\vec{X})$ into an equivalent sr $\mathcal{R}'(\vec{X'})$ applying the construction in the proof of Lemma 8.3.17 (ii). Now take any equation $X = A_1 \to A_2 \in \mathcal{R}'$ such that either A_1 or A_2 are not indeterminates. Assume that A_1 is not an indeterminate. Then replace in $\mathcal{R}'$ the equation $X = A_1 \to A_2$ by the two equations $X = Y \to X_2$, $Y = A_1$, where Y is a fresh new indeterminate. Do similarly if A_2 is not an indeterminate. Repeat this steps until the resulting sr is in flat form; it is trivial to prove that this process terminates. It is also trivial to prove that at each step $\mathcal{R}'$ is transformed into an equivalent sr.

9.3.10. EXAMPLE. Let $\mathcal{R}_1$ be the sr defined in example 8.3.12 (ii). Its flat form is the sr $\mathcal{R}'_1 = \{X_1 = X_2 \to X_3,\ X_2 = \alpha,\ X_3 = X_4 \to X_1,\ X_4 = \alpha\}$.

In the following construction we define, give a sr $\mathcal{R}$, a sequence of equivalence relations on $\mathrm{Dom}(\mathcal{R})$. The aim is to group in the same equivalence class the indeterminates $X \in \mathrm{Dom}(\mathcal{R})$ which generates the same tree.

9.3.11. Definition. Let $\mathcal{R} = \mathcal{R}[\vec{X}]$ be a sr in flat form. Define, inductively on n, the equivalence relation $R_n \subseteq \vec{X} \times \vec{X}$ $(n \geq 0)$ as the minimal equivalence relations such that:

1. XR_nX' for all $n \geq 0$ if $X = \alpha$, $X' = \alpha \in \mathcal{R}$ for the same atomic type α;
2. XR_0X' if $X = Y \to Z$, $X' = Y' \to Z' \in \mathcal{R}$ for some Y, Y', Z, $Z' \in \vec{X}$;
3. $XR_{n+1}Y$ $(n \geq 0)$ if $X = Y \to Z$, $X' = Y' \to Z' \in \mathcal{R}$ and XR_nY' and ZR_nZ'.

Let N be the first N such that $R_N = R_{N+1}$. For each equivalence class $[X_i]$ with $X_i \in \vec{X}$ chose a representative X'_i. Let $\mathcal{R}^* = \mathcal{R}^*[\vec{X'}]$ the sr in flat form defined by:

a. $X' = \alpha \in \mathcal{R}^*$ if $X' = \alpha \in \mathcal{R}$;

b. $X' = Y' \to Z' \in \mathcal{R}^*$ if there is an equation $X' = Y \to Z$ in $\mathcal{R}$ and $Y \in [Y']$, $Z \in [Z']$.

Note that N is well defined since in the above construction either R_{n+1} has more equivalence classes of R_n or $R_{n+1} = R_n$. But the number of equivalence classes in R_n cannot be grater then the cardinality of $\vec{X}$.

9.3.12. Example. Let $\mathcal{R}'_1$ be defined as in Example 9.3.10. Then we have $X_1R_0X_3$, $X_2R_0X_4$ and $N = 0$. Choosing X_1 as representative of the equivalence class $\{X_1, X_3\}$ and X_2 as representative of the equivalence class $\{X_2, X_4\}$ we get $\mathcal{R}'^*_1 = \{X_1 = X_2 \to X_1,\ X_2 = \alpha\}$. Note that $\mathcal{R}'^*_1$ is equivalent to the sr $\mathcal{R}_0$ of Example 8.3.12 (i).

We have that weak (equational) equality w.r.t. $\mathcal{R}$ correspond to strong equality w.r.t. $\mathcal{R}^*$. To prove this we need a preliminary Lemma.

9.3.13. Lemma. *Let $\mathcal{R}$ be a sr in flat form and $\mathcal{R}^* = \mathcal{R}^*[\vec{X'}]$ be defined as in 9.3.9 and let X', $Y' \in \vec{X'}$. Then $(X')^{\mathsf{tr}}_{\mathcal{R}} = (Y')^{\mathsf{tr}}_{\mathcal{R}}$ imply $X' \equiv Y'$.*

Proof. Assume $X' \not\equiv Y'$, i.e. X' and Y' belong to two different equivalence classes in R_N (recall that we choose a unique representative for each equivalence class). Let k be the first value such that X' is not in relation R_k with Y'. We show by induction on k that $(X')^{\mathsf{tr}}_{\mathcal{R}} \neq (Y')^{\mathsf{tr}}_{\mathcal{R}}$.

If $k = 0$ the proof is trivial. Otherwise let $k = k' + 1$ with $k' \geq 0$. Then we must have $X' = X'_1 \to X'_2$ and $Y' = Y'_1 \to Y'_2$ in $\mathcal{R}$ where for some $i \in \{1, 2\}$ X'_i is not in relation $R_{k'}$ with Y'_i. By induction hypothesis we have $(X'_i)^{\mathsf{tr}}_{\mathcal{R}} \neq (Y'_i)^{\mathsf{tr}}_{\mathcal{R}}$ which implies

$$(X')^{\mathsf{tr}}_{\mathcal{R}} = (X'_1)^{\mathsf{tr}}_{\mathcal{R}} \to (X'_2)^{\mathsf{tr}}_{\mathcal{R}} \neq (Y'_1)^{\mathsf{tr}}_{\mathcal{R}} \to (Y'_2)^{\mathsf{tr}}_{\mathcal{R}} = (Y')^{\mathsf{tr}}_{\mathcal{R}}$$

If A is type expression its *length* (denoted $|A|$) is defined as the number of symbols occurring in it.

9.3.14. THEOREM. *Let $\mathcal{R}(\vec{X})$ be a sr and $A, B \in \mathbb{T}[\mathcal{R}]$. There there is a sr $\mathcal{R}^*$ such that*

(i) $A =_{\mathcal{R}^*} B$ *iff* $A =^*_{\mathcal{R}} B$.

(ii) $\mathbb{T}[\mathcal{R}]^*$ *is isomorphic to* $\mathbb{T}[\mathcal{R}^*]$.

PROOF. (i)By Lemma 9.3.9 we can assume that $\mathcal{R}$ is in flat form. Then take $\mathcal{R}^*$ as defined in 9.3.11
Only if part. Assume $A =_{\mathcal{R}^*} B$. An easy induction on k shows that for all $k \geq 0$ $(A)^{\mathsf{tr}}_{\mathcal{R}}\big|_k = (B)^{\mathsf{tr}}_{\mathcal{R}}\big|_k$, and then by Lemma **??** $(A)^{\mathsf{tr}}_{\mathcal{R}} = (B)^{\mathsf{tr}}_{\mathcal{R}}$.
If part. Assume $(A)^{\mathsf{tr}}_{\mathcal{R}} = (B)^{\mathsf{tr}}_{\mathcal{R}}$. We prove by induction on $|A| + |B|$ that $A =_{\mathcal{R}^*} B$.
case 1. If A is an atomic type α then B must be either the same atomic type or an undeterminate X' such that $X' = \alpha \in \mathcal{R}^*$. In both cases the proof is trivial.
case 2. If $A \equiv A_1 \to A_2$ then either $B \equiv B_1 \to B_2$ or $B \equiv Y$ for some $Y \in \vec{X}$ such that $Y = B_1 \to B_2 \in \mathcal{R}$ (in this case $B_1, B_2 \in \vec{X}$). By Lemma 8.5.5 we have $A_1 =^*_{\mathcal{R}} B_1$, $A_2 =^*_{\mathcal{R}} B_2$ and, by induction hypothesis, $A_1 =_{\mathcal{R}^*} B_1$, $A_2 =_{\mathcal{R}^*} B_2$. We then get $A =_{\mathcal{R}} B$ by an application of rule ($\to$-cong).
Case 3. If $A \equiv Y$ for some $Y \in \vec{X}$ then either $Y = \alpha \in \mathcal{R}$ and we argue as in case 1. or $Y = Y_1 \to Y_2 \in \vec{X}$ and we argue as in case 2.
(ii) Immediate by (i). ■

Solvability of type equations in sr

In the study of recursive type inference it will useful to know if a given sr solves a given set of equations, according to definition **??**. We prove here that this is a decidable property. The original proof is due to Statman (**?**).

We first need to extend the axiomatization of type equality to force a type algebra generated by a set of equations $\mathcal{E}$ to have the invertibility property.

9.3.15. DEFINITION. Let $\mathcal{E}$ be a system of type equations over a set $\mathbb{T}$ of types. The relation $=^{\text{inv}}_{\mathcal{E}}$ is defined as the least relation satisfying the axioms and rules for equality given in Definition 8.1.7 plus the following rules for invertibility

$$(\text{inv}) \quad \frac{\vdash A_1 \to A_2 = B_1 \to B_2}{\vdash A_i = B_i} \quad (i = 1, 2)$$

We write $\vdash^{\text{inv}}_{\mathcal{E}} A = B$ (or $A =^{\text{inv}}_{\mathcal{E}} B$) to mean that $A = B$ is provable in $(\mathcal{E}^{\text{inv}})$.

Then $=^{\text{inv}}_{\mathcal{E}}$ is the least invertible congruence containing $=_{\mathcal{E}}$. Given a system of type equations we define now a sr which generates $=^{\text{inv}}_{\mathcal{E}}$.

9.3.16. DEFINITION. Let $\mathcal{E}$ be a system of type equations over $\mathbb{T}(\mathbb{A})$.

(i) Let $\alpha_0, \alpha_1, \ldots$ denote the elements of $\mathbb{A}$. As in of Definition 9.3.3 we define by induction on n a sequence of sets of equations $\mathcal{D}_n$, $\mathcal{I}_n$ $(n \geq 0)$. Let $\mathcal{D}_0 = \mathcal{E}$ and $\mathcal{I}_0 = \emptyset$. Define $\mathcal{D}_{n+1}$, $\mathcal{I}_{n+1}$ from $\mathcal{D}_n$, $\mathcal{I}_n$ $(n \geq 0)$ in the following way:

(1) If there is an equation $A \to B = C \to D \in \mathcal{D}_n$ then take

$$\mathcal{D}_{n+1} = \mathcal{D}_n - \{A \to B = A' \to B'\} \cup \{A = A', B = B'\}$$

and $\mathcal{I}_{n+1} = \mathcal{I}_n$;

(2) If there are two equations $\alpha_i = A \to B, \alpha_i = A' \to B' \in \mathcal{D}_n$, assuming $|A \to B| \leq |A' \to B'|$, then take

$$\mathcal{D}_{n+1} = \mathcal{D}_n - \{\alpha_i = A' \to B'\} \cup \{A = A', B = B'\}$$

and $\mathcal{I}_{n+1} = \mathcal{I}_n$;

(3) If there is an equation $\alpha_i = \alpha_i \in \mathcal{D}_n$ then take $\mathcal{D}_{n+1} = \mathcal{D}_n - \{\alpha_i = \alpha_i\}$ and $\mathcal{I}_{n+1} = \mathcal{I}_n$;

(4) If there is an equation $\alpha_i = \alpha_j \in \mathcal{D}_n$ (where α_i and α_j are both atomic and $i \neq j$) then take

$$\begin{array}{rcl} \mathcal{D}_{n+1} & = & \mathcal{D}_n[\alpha_h := \alpha_k] - \{\alpha_i = \alpha_j\} \\ \mathcal{I}_{n+1} & = & \mathcal{I}_n \cup \{a_h = a_k\} \end{array}$$

where $h = \min(i, j)$ and $k = \max(i, j)$;

(5) Otherwise take $\mathcal{D}_{n+1} = \mathcal{D}_n$ and $\mathcal{I}_{n+1} = \mathcal{I}_n$.

(ii) Let N be the least n such that $\mathcal{D}_{n+1} = \mathcal{D}_n$ and $\mathcal{I}_{n+1} = \mathcal{I}_n$. This number certainly exists since the total number of symbols in $\mathcal{D}_n \cup \mathcal{I}_n$ decreases at each step. Define $\mathcal{E}^{\text{inv}}$ as $\mathcal{D}_N \cup \mathcal{I}_N$.

In the sr $\mathcal{E}^{\text{inv}}$ the atomic types α_i which occur in l.h.s. of the equations play the role of unknowns.

9.3.17. Lemma. *Let $\mathcal{E}$ a system of type equations. Then $\mathcal{E}^{\text{inv}}$ is a proper sr such that $\vdash^{\text{inv}}_{\mathcal{E}} \mathcal{E}^{\text{inv}}$ and $\vdash_{\mathcal{E}^{\text{inv}}} \mathcal{E}$ (i.e. $\mathcal{E}^{\text{inv}}$ is equivalent to $\mathcal{E}$ plus invertibility).*

Proof. It easy to prove, by induction of n, that for all $n \geq 0$

$$\vdash_{\mathcal{D}_n \cup \mathcal{I}_n} \mathcal{E} \quad \text{and} \quad \vdash^{\text{inv}}_{\mathcal{E}} \mathcal{D}_n \cup \mathcal{I}_n$$

To show that $\mathcal{D}_N \cup \mathcal{I}_N$ is a proper sr note that, by step i(2) and i(4) of Definition 9.3.16 $\mathcal{D}_N$ is a proper sr. Now note that when an equation $\alpha_i = \alpha_j$ is put in $\mathcal{I}_n$ we must have $i < j$ and moreover α_i does not occur in $\mathcal{D}_m$ for all $m \geq n$. So no other equation of the form $\alpha_i = \alpha_h$ can be put in some $\mathcal{I}_m$ for $m \geq n$. ■

We see now an application of this result.

9.3.18. Theorem. *Let $\mathcal{E}$ be a set of equations over $\mathbb{T}$. It is decidable whether a sr $\mathcal{R}(\vec{X})$ over $\mathbb{T}' = \mathbb{T}(\mathbb{A}')$ solves $\mathcal{E}$.*

PROOF. By Definition **??** there is a map $h : \mathbb{T} \to \mathbb{T}'[\vec{X}]$ such that $A =_{\mathcal{E}} B$ implies $h(A) =_{\mathcal{R}} h(B)$. Since, by Lemma 9.3.7, $=_{\mathcal{R}}$ is invertible, we have that $h(A_1 \to B_1) =_{\mathcal{R}} h(A_2 \to B_2)$ implies $h(A_i) =_{\mathcal{R}} h(B_i)$ for $i = 1, 2$. Then $\mathcal{R}$ solves also the proper sr $\mathcal{E}^{\text{inv}}$ as defined in 9.3.16.

Let now $\mathcal{E}^{\text{inv}} = \mathcal{D}_{\mathcal{E}} \cup \mathcal{I}_{\mathcal{E}}$ where $\mathcal{D}_{\mathcal{E}}$ contains only equations of the form $\alpha = A \to B$ and $\mathcal{I}_{\mathcal{E}}$ only equations of the form $\alpha = \beta$ where where $\alpha \in \text{Dom}(\mathcal{E}^{\text{inv}})$ and β is an atomic type. Now note that it is enough to solve $\mathcal{D}_{\mathcal{E}}$ in $\mathcal{R}$; in fact we can easily extend a solution h of $\mathcal{D}_{\mathcal{E}}$ in $\mathcal{R}$ to a solution h' of $\mathcal{D}_{\mathcal{E}} \cup \mathcal{I}_{\mathcal{E}}$ in $\mathcal{R}$ in the following way. For each atomic type $\alpha \in \text{Dom}(\mathcal{I}_{\mathcal{E}})$ let $[\alpha]$ denote the equivalence class of α w.r.t. $\mathcal{E}^{\text{inv}}$. By lemma 8.3.19 $[\alpha]$ contains only atomic types. Then define:

- $h'(\alpha) = h'(\beta)$ if $[\alpha]$ contains an atomic type $\beta \notin \text{Dom}(\mathcal{E}^{\text{inv}})$ occurring in $\mathcal{D}_{\mathcal{E}}$
- $h'(\alpha) = A$ where $A \in \mathbb{T}'$ is arbitrary, otherwise.

We can further simplify the crucial set of equations to be solved by observing that we can successively eliminate from $\mathcal{D}_{\mathcal{E}}$ all equations $\alpha = A$ in which α does not occur in A by replacing α with A in the remaining set $\mathcal{D}'_{\mathcal{E}}$ of equations. Since α does not occur in $\mathcal{D}'_{\mathcal{E}}$ we can extend a solution h' of $\mathcal{D}'_{\mathcal{E}}$ in $\mathcal{R}$ to a solution h of $\mathcal{D}_{\mathcal{E}}$ in $\mathcal{R}$ by defining $h(\alpha) = h'(A)$.

Finally let $\mathcal{S}_{\mathcal{E}}$ denote the resulting system of equations. We are left with the problem of solving the sr $\mathcal{S}_{\mathcal{E}}$ in $\mathcal{R}$.

By remark 8.1.6 (ii) in order to find a solution h of $\mathcal{S}_{\mathcal{E}}$ in $\mathcal{R}$ it is enough to find, for each atomic type α occurring in $\mathcal{S}_{\mathcal{E}}$ a type $h(\alpha)$ such that if $\alpha = A \in \mathcal{S}_{\mathcal{E}}$ we have $h(\alpha) =_{\mathcal{R}} h(A)$. The other atomic types of $\mathbb{A}$ can in fact be mapped into arbitrary types of $\mathbb{T}'$.

By Lemmas 9.3.4 (i) and (ii) we can always assume that for all α which have an occurrence in $\mathcal{S}_{\mathcal{E}}$, $h(\alpha)$ is in normal form with respect to $\text{Trs}^{\diamond}(\mathcal{R})$. Let now $\alpha = A \in \mathcal{S}_{\mathcal{E}}$, then $h(A) \Rightarrow^* h(\alpha)$. Since A is non-atomic and contain α itself, this reduction must be in at least one step. Moreover since $h(\alpha)$ is in normal form, no subexpression of $h(\alpha)$ can be a $\text{Trs}^{\diamond}(\mathcal{R})$- redex. But since $h(\alpha)$ is itself a subexpression of $h(A)$, $h(\alpha)$ must be a proper subexpression of some $\text{Trs}^{\diamond}(\mathcal{R})$-redex occurring in the reduction of $h(A) \Rightarrow^+ h(\alpha)$ (in other words all occurrences of $h(\alpha)$ in $h(A)$ can have no residual in $h(\alpha)$). As for the other atomic types α' occurring in A note that also $h(\alpha')$ must be a proper subexpression of a $\text{Trs}^{\diamond}(\mathcal{R})$-redex in the reduction $h(A) \Rightarrow^+ h(\alpha)$.
So we have that for all atomic α occurring in $\mathcal{S}_{\mathcal{E}}$, $h(\alpha)$ must be a subexpression of a redex in $\text{Trs}^{\diamond}(\mathcal{R})$. This bound the possible choices for h proving decidability.

9.3.19. EXAMPLE. Let $\mathcal{R}_0 = \{X_1 = \alpha \to X_1, X_2 = \beta \to \beta \to X_2\}$ a sr over $\mathbb{T}(\mathbb{A})$ where $\alpha, \beta \in \mathbb{A}$. We want to search for a solution of $\mathcal{R}_0$ in the sr $\mathcal{R}_1 = \{X'_1 = X'_1 \to X'_1, X'_2 = \gamma \to X'_2\}$ over $\mathbb{T}(\mathbb{A}')$ where $\gamma \in \mathbb{A}'$. Following the construction presented in the proof of the Lemma both $h(X_1)$ and $h(X_2)$ must be subterms of some redex in $\text{Trs}^{\diamond}(\mathcal{R}_1)$. The possible redexes in $\text{Trs}^{\diamond}(\mathcal{R}_1)$ are $X'_1 \to X'_1$ and $\gamma \to X'_2$. Two possible choices for h are:

1. $h_1(X_1) = X'_1$, $h_1(X_2) = X'_2$, $h_1(\alpha) = X'_1$, $h_1(\beta) = \gamma$.
2. $h_1(X_1) = X'_2$, $h_1(X_2) = X'_2$, $h_1(\alpha) = \gamma$, $h_1(\beta) = \gamma$.

The values of h_1, h_2 on the other elements of $\mathbb{A}$ are arbitrary.

9.3.20. Remark. Note that using the construction given in the proof of the previous lemma we can indeed build a finite number $h_1, \ldots, h_k$, with $0 \leq k \leq n$, of distinct solutions of $\mathcal{E}^{\text{inv}}$ in $\mathcal{R}$. Each of them defines indeed a class of solutions which differ only for the values of $h_i(\alpha)$ on the atoms α for which $h_i(\alpha)$ is arbitrary. Assuming that in $Atoms'$ there are enough atomic types we can define a canonical representative $\overline{h_i}$ for each class $0 \leq i \leq k$) by setting $\overline{h_i}(\alpha) = \alpha'$ where α' is a new fresh type variable. It then is possible to prove that any type algebra homomorphism $h' : \mathbb{T}/\mathcal{E} \to \mathbb{T}'[\mathcal{R}]$ can be written as $h' = h'' \circ \overline{h_i}$ for some $1 \leq i \leq k$ where h'' is an homomorphism of $\mathbb{T}'[\mathcal{R}]$ in itself.

By Theorem this result implies decidability of a system of type equations in type algebras of the form $\mathbb{T}'[\mathcal{R}]^*$.

9.3.21. Theorem. *Let $\mathcal{E}$ be a set of equations over $\mathbb{T}$ and $\mathcal{R}(\vec{X})$ a sr over $\mathbb{T}' = \mathbb{T}(\mathbb{A}')$. It is decidable if $\mathbb{T}'[\mathcal{R}]^*$ solves $\mathcal{E}$.*

Proof. By Theorem 9.3.14 (ii)$\mathbb{T}'[\mathcal{R}]^*$ solves $\mathcal{E}$ iff $\mathcal{R}^*$ solves $\mathcal{E}$

Statman **?** has shown that the solvability of a sr in another sr is indeed a NP-complete problem. [to Felice. There is a P-time subproblem. Could you look at it?]

Axiomatization of strong equivalence for sr

Given a sr $\mathcal{R}$ Theorem 9.3.14 (ii) shows that $=^*_{\mathcal{R}}$ can be axiomatized by the rules of Definition 8.1.7 simply by taking $\mathcal{R}^*$ instead of $\mathcal{R}$. However, in a way similar to what we have done in section 9.2 for $=^*$, we can also define a coinductive system $(\mathcal{R}^*)$ to directly axiomatize $=^*_{\mathcal{R}}$. Also in this systems we have judgments of the form $\mathcal{H} \vdash A = B$ in which $\mathcal{H}$ is a set of equations of the shape $A \to A' = B \to B'$. As in system the crucial point is the introduction of a rule (coind) which is based on the same coniduction principle of the corresponding rule of the system BH (which has indeed the same form). Rule (coind), for instance, has exactly the same shape as in Definition 9.2.18 and can be motivated similarly.

9.3.22. Definition. Let $\mathcal{R}$ be a proper sr. The system $(\mathcal{R}^*)$ is defined by the

following axioms and rules:

$$
\begin{array}{ll}
(\mathcal{R}\text{-eq}) & \mathcal{H} \vdash X = A \quad \text{if } X = A \in \mathcal{R} \\
(\text{ident}) & \mathcal{H} \vdash A = A \\
(\text{symm}) & \dfrac{\mathcal{H} \vdash A = B}{\mathcal{H} \vdash B = A} \\
(\text{trans}) & \dfrac{\mathcal{H} \vdash A = B \quad \mathcal{H} \vdash B = C}{\mathcal{H} \vdash A = C} \\
(\text{hyp}) & \mathcal{H} \vdash A_1 \to A_2 = B_1 \to B_2 \quad \text{if} \quad A_1 \to A_2 = B_1 \to B_2 \in \mathcal{H} \\
(\text{coind}) & \dfrac{\mathcal{H} \cup \{A \to B = A' \to B'\} \vdash A = A' \quad \mathcal{H} \cup \{A \to B = A' \to B'\} \vdash B = B'}{\mathcal{H} \vdash A \to B = A' \to B'}
\end{array}
$$

We write $\mathcal{H} \vdash^*_{\mathcal{R}} A = B$ to mean that $\mathcal{H} \vdash A = B$ can be derived by the above rules.

Also in this system rule ($\to$-cong) is missing but it is easy to prove that is derivable. Note that in this system there are no types having properties analogous to these of the circular types in $\mathbb{T}_\mu$, $\mathbb{T}^*_\mu$.

9.3.23. Example. Let $\mathcal{R}_1 = \{X = A \to A \to X\}$ where A is a any type. We have the following proof of $\vdash^*_{\mathcal{R}_1} X = A \to X$. Let C denote $A \to A \to X$.

$$
\dfrac{\dfrac{(\text{hyp})}{\vdash X = C} \qquad \dfrac{\dfrac{(\text{ident})}{\{C = A \to X\} \vdash A = A} \qquad \dfrac{\dfrac{(\mathcal{R}\text{-eq})}{\{C = A \to X\} \vdash X = C} \qquad \dfrac{(\text{hyp})}{\{C = A \to X\} \vdash C = A \to X}\,(\text{trans})}{\{C = A \to X\} \vdash X = A \to X}\,(\text{coind})}{\vdash C = A \to X}\,(\text{trans})}{\vdash X = A \to X}
$$

With the same technique as used in Section 9.2 we can prove the soundness and completeness of $(\mathcal{R}^*)$ with respect to strong equivalence. The completeness proof, in particular, is based on a variant of the algorithm introduced in Definition 9.2.22 (up to some minor adjustments due to the different set of types) which, given two types A and B builds a proof of $A = B$ iff $A =^*_{\mathcal{C}} B$. Therefore we have the following properties:

9.3.24. Theorem. *(i)* $\vdash^*_{\mathcal{R}} A = B$ *iff* $A =^*_{\mathcal{R}} B$.

(ii) Let $\mathcal{R}(\vec{X})$ *be an sr over* $\mathbb{T}$. *Given* $A, B \in \mathbb{T}[\vec{X}]$ *it is decidable whether* $A =^*_{\mathcal{R}} B$.

9.4. Exercises

9.4.1. Prove Theorem 9.1.8 [*give some hint of the solution?*].

9.4.2. Design an algorithm to decide if two μ-types are weakly equivalent *[Hint: try to match types by unfolding them following standard reduction, keeping track of the subtypes considered for matching to avoid looping].*

9.4.3. Prove Lemma 9.2.2.

9.4.4. Prove that rule (μ-cong) as referred in section 9.2 is admissible in BH. Conversely prove that assuming rule (μ-cong) rule (triv) becomes admissible.m

9.4.5. Prove that the subterm closure $\mathcal{SC}(A)$ of a type A (see Definition **??**) is finite.

9.4.6. Let A, B be circular μ-types. Show that $\vdash_{\mathrm{BH}} A = B$.

9.4.7. Show that $\vdash_{\mathrm{BH}} \mu\alpha.(\beta \to \beta \to \alpha) = \mu\alpha.(\beta \to \alpha)$, where β is any type variable (see Example 8.3.27).

9.4.8. Prove $\mu\alpha.((\alpha \to \alpha) \to \alpha) =^* \mu\alpha.(\alpha \to \alpha \to \alpha)$ in BH. Compare the proof with that in (μ^*AK).

9.4.9. Prove that if A, $B \in \mathbb{T}_\mu$ and $A\, R_\mu\, B$, where R_μ is a bisimulation over types defined in Remark **??** then $(A)^* = (B)*$.

9.4.10. Prove Remark 9.3.20.

9.4.11. Prove that $\mu\alpha.((\alpha \to \alpha) \to \alpha) =^* \mu\alpha.(\alpha \to \alpha \to \alpha)$.

9.4.12. Amadio and Cardelli **?** introduce he following rule, equivalent to (AK).

$$\text{(AC)}\quad \frac{\vdash B = A[\alpha := B] \qquad \vdash C = A[\alpha := C]}{B = C} \quad \text{provided } A \text{ is contractive in } \alpha$$

Show that (AC) is equivalent to (AK).

9.4.13. Prove soundness, completeness and decidability of ($\mathcal{R}^*$).

9.4.14. Give a formulation of rule (AK) for systems of type equations. Use it to prove $\vdash^*_{\{c=\beta\to\beta\to c\}} c = \beta \to c$.

9.4.15. Prove Theorem 9.3.24.

9.4.16. Prove Proposition 9.2.30.

Chapter 10

Properties of terms with types

21.2.2008:897

In this Chapter we establish some basic properties of terms which have types in the systems introduced in the previous Chapters. We only consider type inference systems à la Curry. Some of the results shown in this Chapter are meaningless for typed systems à la Church, while some others can be easily adapted to them (like, for instance, the Subject Reduction Theorem). In Section 10.1 we study the subject reduction property for recursive type systems, in Section 10.2 the problem of finding types for untyped terms and in Section 10.3 we study specific recursive types such that strong normalization holds for terms that can be typed using these types.

10.1. Subject reduction

We will show that for invertible $\mathcal{A}$ the recursive type systems $\lambda\mathcal{A}$ have essentially the subject reduction property, i.e.

$$\left.\begin{array}{l}\Gamma \vdash_{\lambda\mathcal{A}} M : a \\ M \twoheadrightarrow_{\beta(\eta)} M'\end{array}\right\} \Rightarrow \Gamma \vdash_{\lambda\mathcal{A}} M' : a,$$

where $\vdash$ is defined in Section 8.1.

This property is important for type systems since it means that typings are stable with respect to β- or $\beta\boldsymbol{\eta}$-reduction, which is the fundamental evaluation process for λ-terms.

Types are not preserved, in general, by the reverse operation of expansion.

We will study the subject reduction property for a type assignment system induced by an arbitrary type algebra $\mathcal{A}$. It satisfies the subject reduction property iff it is invertible.

We start with some basic lemmas.

10.1.1. Lemma. $\Gamma \vdash_{\lambda\mathcal{A}} M : a,\ \Gamma \vdash_{\lambda\mathcal{A}} N : b \ \Rightarrow\ \Gamma \vdash_{\lambda\mathcal{A}} M[x{:=}N] : a.$

Proof. By induction on the derivation of $\Gamma, x\ b \vdash_{\lambda\mathcal{A}} M : a$. ■

10.1.2. LEMMA. *Suppose* $x \notin \mathrm{FV}(M)$, *then*

$$\Gamma, x\, b \vdash_{\lambda\mathcal{A}} M : a \;\Rightarrow\; \Gamma \vdash M : a.$$

PROOF. By induction on the derivation of $\Gamma, x\, b \vdash M : a$. ■

10.1.3. LEMMA (Inversion Lemma). *Let* $\mathcal{A}$ *be a type algebra.*

(i) $\quad \Gamma \vdash_{\lambda\mathcal{A}} x : a \iff (x\, a) \in \Gamma.$

(ii) $\quad \Gamma \vdash_{\lambda\mathcal{A}} (MN) : a \iff \exists b \in \mathcal{A}.[\Gamma \vdash_{\lambda\mathcal{A}} M : (b \to a)\ \&\ \Gamma \vdash_{\lambda\mathcal{A}} N : b].$

(iii) $\quad \Gamma \vdash_{\lambda\mathcal{A}} (\lambda x.M) : a \iff \exists b, c \in \mathcal{A}.[a = (b \to c)\ \&\ \Gamma, x\, b \vdash_{\lambda\mathcal{A}} M : c].$

PROOF. ($\Leftarrow$) immediate. ($\Rightarrow$) The proof is in all cases by induction on the derivation of $\Gamma \vdash P : d$.

(i) To prove $\Gamma \vdash x : A$ we can use only rule (axiom).

(ii) Similar to (i). Now we can use only rule ($\to E$).

(iii) Now we can only use rule ($\to I$). ■

For syntactic type algebras this boils down to the following.

10.1.4. LEMMA (Inversion Lemma). *Let* $\mathcal{A} = \mathbb{T}/{\simeq}$ *be a syntactic type algebra.*

(i) $\quad \Gamma \vdash_{\simeq} x : A \iff \exists A' \in \mathbb{T}.[A' \simeq A\ \&\ x\, A' \in \Gamma].$

(ii) $\quad \Gamma \vdash_{\simeq} (MN) : A \iff \exists B \in \mathbb{T}.[\Gamma \vdash_{\simeq} M : B \to A\ \&\ \Gamma \vdash_{\simeq} N : B].$

(iii) $\quad \Gamma \vdash_{\simeq} \lambda x.M : A \iff \exists B, C \in \mathbb{T}.[A \simeq (B \to C)\ \&\ \Gamma, x\, B \vdash_{\simeq} M : C].$

We prove the subject reduction property for one step $\beta\boldsymbol{\eta}$-reduction.

10.1.5. LEMMA. (i) *If* $\Gamma \vdash_{\lambda\mathcal{A}} \lambda x.(Mx) : a$ *and* $x \notin \mathrm{FV}(M)$, *then* $\Gamma \vdash_{\lambda\mathcal{A}} M : a$.

(ii) *If moreover* $\mathcal{A}$ *is invertible, then*

$$\Gamma \vdash_{\lambda\mathcal{A}} (\lambda x.M)N : a \;\Rightarrow\; \Gamma \vdash_{\lambda\mathcal{A}} (M[x := N]) : a.$$

PROOF. (i) By the Inversion Lemma 10.1.3(iii) we have that $\Gamma, x\, b \vdash (Mx) : c$ for some $b, c \in \mathcal{A}$ such that $a = (b \to c)$. Moreover, by Lemma 10.1.3 (ii) and (i) we have that $\Gamma, x\, b \vdash M : d \to c$ and $\Gamma, x\, B \vdash_{\lambda\mathcal{A}} x : d$ for some type d such that $d = b$. Then $(d \to c) = (b \to c) = a$. Then $\Gamma \vdash M : a$, by Lemma 10.1.2.

(ii) Suppose $\Gamma \vdash (\lambda x.M)N : a$. By the Inversion Lemma 10.1.3(iii)

$$\Gamma \vdash (\lambda x.M) : b \to a\ \&\ \Gamma \vdash_{\lambda\mathcal{A}} N : b,$$

for some type b. Moreover, by Lemma 10.1.3(iii), we have that for some types b' and a' such that $b' \to a' = b \to a$

$$\Gamma, x\, b' \vdash M : a'$$

Then $b' = b$ and $a' = a$, by invertibility. Hence $\Gamma \vdash N : b'$ and by Lemma 10.1.1

$$\Gamma \vdash (M[x := N]) : a'.\ ■$$

10.1.6. COROLLARY (Subject Reduction). *Let $\mathcal{A}$ be an invertible type algebra. Then $(\lambda\mathcal{A})$ satisfies the subject reduction property for $\beta\boldsymbol{\eta}$.*

PROOF. By Lemma 10.1.5. ■

Invertibility of $\mathcal{A}$ is a characterization of all the type algebras $\mathcal{A}$ such that $\vdash_{\lambda\mathcal{A}}$ has the subject reduction property. The proof is from **?**.

10.1.7. THEOREM. *Let $\mathcal{A}$ be a type algebra. Then*

$$\mathcal{A} \text{ is invertible} \iff \lambda\mathcal{A} \text{ satisfies the subject reduction property for } \beta.$$

PROOF. ($\Rightarrow$) By Lemma 10.1.6.

($\Leftarrow$) Let $a, a', b, b' \in \mathcal{A}$ and assume $a \to b = a' \to b'$. We first prove that $b = b'$. Writing $\Gamma_1 = \{x\, b,\ y\, a'\}$ we have $\Gamma_1 \vdash ((\lambda z.x)y) : b'$. Then by subject reduction $\Gamma_1 \vdash x : b'$. Hence by the Inversion Lemma 10.1.3(i) $b = b'$.

Now take $\Gamma_2 = \{x\ (a \to b),\ y\ (a \to a),\ z\ a'\}$. Using the assumption we can derive $\Gamma_2 \vdash (\lambda u.x(yu))z : b'$. Again by subject reduction we obtain $\Gamma_2 \vdash (x(yz)) : b'$. Now by the Inversion Lemma 10.1.3(ii) there is a type $c \in \mathbb{T}$ such that $\Gamma_2 \vdash x : c \to b'$ and $\Gamma_2 \vdash (yz) : c$. Again by the Inversion Lemma 10.1.3 (iii), (i) the second statement implies $a \to a = a' \to c$. By the first part of the proof we then have $a = c$ and so $a \to a = a' \to a$. Now use this to prove $x\, a' \vdash ((\lambda y.y)x) : a$. A final application of subject reduction and the Inversion Lemma yields $a = a'$. ■

10.2. Finding types

In this section we consider the problem of finding the types (if any) of a given untyped term in systems $(\lambda\mathcal{A})$ à la Curry. As in the case of simple types this kind of problems is much simpler for explicitly typed terms. The well formed terms of the typed calculi à la Church or à la de Bruijn belong to a unique type and have all type information written in them. To check that they are well formed with respect to the formation rules the only non trivial step is that of checking type equivalence in rule (equiv). But we have seen in Chapter 9 that all interesting type equivalences are decidable. Type checking is even simpler in the system ($\lambda\mu$-Ch$_0$) where also type equivalence is explicit.

The questions that can be asked about typing terms in recursive type system are similar to those introduced in Section 2.2 but we have here one more parameter, indeed the type algebra in which we want to work. Among the questions that can be asked about checking and finding types of a λ-term, we only consider the more interesting ones.

Assume, for simplicity, that M is a closed term. We will investigate the following type reconstruction problems about M.

1. Find all type algebras $\mathcal{A}$ and all types A such that $\vdash_{\lambda\mathcal{A}} M : A$.
2. Given a type algebra $\mathcal{A}$ find all types A such that $\vdash_{\lambda\mathcal{A}} M : A$.

Typeability problems can be seen as a particular case of both 1. and 2. in which we simply ask weather the set of all possible types of a given term is empty or not. In particular, given a λ-term M the typeability problems corresponding to 1. and 2. are:

1. Do there exist a type algebra $\mathcal{A}$ and a type A such that $\vdash_{\lambda\mathcal{A}} M : A$?

2. Given a type algebra $\mathcal{A}$, does there exist a type A such that $\vdash_{\lambda\mathcal{A}} M : A$?

Note that, from Lemma 10.1.3(ii),

$$x_1\, A_1, ..., x_n\, A_n \vdash_{\lambda\mathcal{A}} M : A \;\Leftrightarrow\; \vdash_{\lambda\mathcal{A}} \lambda x_1 \ldots x_n.M : A_1 \to ... \to A_n \to A.$$

So it is not restrictive to formulate these questions for closed terms only. We will not consider here the possibility of having a fixed type environment Γ, but the results of this section can easily be generalized to consider it.

The problem of inhabitation is sometimes trivial. As pointed out in Remark **??**, for instance, in the systems $(\lambda\mu)$, $(\lambda\mu^*)$ al types are inhabited.

Lastly we will present here the basic results concerning type reconstruction considering only pure $\lambda-$terms. The generalization to terms containing constants is rather straightforward and will be discussed in Remark 10.2.8.

Finding types in simultaneous recursions

The most natural way to describe a type algebra is via a s.r. over a set $\mathbb{T}$ of types. So we will first investigate the above questions assuming that $\mathcal{A}$ is of the form $\mathbb{T}[\mathcal{R}]$. In this case the above problems can be written in this way:

1. Characterize all s.r. $\mathcal{R}$ and all types A such that $\vdash_{\lambda\mathcal{R}} M : A$.

2. Given an s.r. $\mathcal{R}$ characterize all types A such that $\vdash_{\lambda\mathcal{R}} M : A$.

There is one more interesting question about typeability in $\vdash_{\lambda\mathcal{R}}$ which does not follow immediately from the previous ones:

3. Given an s.r. $\mathcal{R}$ and a type A does $\vdash_{\lambda\mathcal{R}} M : A$ hold?

Note that by Theorem 9.3.14 the analogous problems about $\vdash_{\lambda\mathcal{R}^\sharp}$ can be seen as a particular case. We will show that the above questions in this case are all decidable. The answer to similar questions for μ-types follows easily from this case.

Via the notion of type algebra homomorphism (see section 9) we can define, in a quite natural way, a notion of a principal type scheme for type assignments with respect to arbitrary invertible type algebras. It turns out to be a quite natural generalization of principal type scheme result holding for the simple assignment system à la Curry.
In the following definition we assume, without loss of generality, that all free and bound variables in a term have distinct names.

10.2.1. Definition. Let M be a pure λ-term and $\mathbb{T} = \mathbb{T}(\mathbb{A})$. The system of equations $\mathcal{E}_M$, the basis Γ_M and type T_M are defined as follows.
Let $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$ where $\mathcal{S}_1$ is the set of all bound and free variables of M and $\mathcal{S}_2$ is the set of all occurrences of subterms of M which are not variables. Associate to each element P of $\mathcal{S}$ a fresh atomic type $\pi(P)$. Let $\mathbf{I}_M$ the set of all such atomic types. Now define a system of type equations $\mathcal{E}_M$ over $\mathbb{T}_M = \mathbb{T}(\mathbf{I}_M)$ in the following way.

(a) For every $\lambda x.P \in \mathcal{S}_2$ put $\pi(\lambda x.P) = \pi(x) \to \pi(P)$ in $\mathcal{E}_M$.

(b) For every $P = (P_1 P_2) \in \mathcal{S}_2$ put $\pi(P_1) = \pi(P_2) \to \pi(P)$ in $\mathcal{E}_M$.

Now let $\Gamma_M = \{x : \pi(x) \mid x \text{ is free in } M\}$ and $T_M = \pi(M)$. Moreover define $\mathcal{A}_M = \mathbb{T}_M/\mathcal{E}_M$ and $\mathcal{R}_M = {\mathcal{E}_M}^{\text{inv}}$.

Note that $\mathcal{E}_M$ is a system of type equations but not, in general, a s.r..

10.2.2. Example. Consider the term $\lambda x.xx$ and take $\pi(x) = \alpha_1$, $\pi(xx) = \alpha_2$, $\pi(\lambda x.xx) = \alpha_3$. Then we have $\mathcal{E}_{\lambda x.xx} = \mathcal{R}_{\lambda x.xx} = \{\alpha_1 = \alpha_1 \to \alpha_2,\ \alpha_3 = \alpha_1 \to \alpha_2\}$, $\Gamma_{\lambda x.xx} = \emptyset$, $T_{\lambda x.xx} = \alpha_3$.

It is immediate to prove, by induction on M, the following property.

10.2.3. Lemma. *Let M be a pure λ-term. Then*
(i) $\Gamma_M \vdash_{\lambda\mathcal{A}_M} M : T_M$.
(ii) $\Gamma_M \vdash_{\lambda\mathcal{R}_M} M : T_M$

Indeed to type any pure λ-term it would be enough to take a s.r. with only one equation $\mathcal{R}_0 = X = X \to X$. So the problem of typeability is trivial for pure $\lambda-$terms. The typings obtained assuming $\mathcal{R}_0$ are trivial and often not interesting. On the other side $\mathcal{E}_M$ gives the weakest type constraints necessary to type M, and then characterizes all type algebras in which M can be typed (Theorem 10.2.4 below). There are obviously terms which can be typed only with respect to $\mathcal{R}_0$, like $(\lambda x.xx)(\lambda y.y)$ (we leave to the reader to show this).
By Theorem 10.1.7 we are mainly interested in type systems built on invertible type algebras. In the next theorem we will therefore consider $\mathcal{R}_M$ rather then $\mathcal{E}_M$.

10.2.4. Theorem. *Let $M \in \Lambda$ and $\mathcal{A}$ an invertible type algebra. Then*

$$\Gamma \vdash_{\lambda\mathcal{A}} M : a \iff a = h(T_M) \text{ and } h(\Gamma_M) \text{ is a subset of } \Gamma$$
$$\text{for some type algebra morphism } h : \mathbb{T}_M[\mathcal{R}_M] \to \mathcal{A}.$$

Proof. ($\Leftarrow$) By Lemmas 10.2.3 (ii) and 8.1.15. ($\Rightarrow$) Note that, by the same argument as used at the beginning of the proof of Theorem 9.3.18, being $\mathcal{A}$ invertible, a homorphism $h : \mathcal{A}_M \to \mathcal{A}$ is also an homomorphism from $\mathbb{T}_M[\mathcal{R}_M]$ to $\mathcal{A}$. So it is enough to show that there via a homomorphism $h : \mathcal{A}_M \to \mathcal{A}$ satisfying the statement of the Lemma. Now take a deduction D of $\Gamma \vdash_{\lambda\mathcal{A}} M : A$. For each $P \in \mathcal{S}$ (see definition 10.2.1) let $\tau(P)$ be the type assigned to P in D

(in particular for variables x take the type assigned to x in the assumptions). Now let $h : \mathbb{T}_M \to \mathbb{T}$ be the type homomorphism defined by $h(\alpha) = B$ if $\alpha = \pi(P)$ for some $P \in \mathcal{S}$ and $\tau(P) = B$. Note that h is well defined since π is a bijection between $\mathcal{S}$ and $\mathbf{I}_M$. Obviously $\Gamma \supseteq h(\Gamma_M)$ (Γ_M contains only variables which occur in M) and $A = h(\pi(M))$. We now prove that h is indeed a type algebra isomorphism between $\mathcal{A}_M$ and $\mathcal{A}$. This is equivalent to prove that for all equations $(\alpha_1 = \alpha_2 \to \alpha_3) \in \mathcal{E}_M$ $h(\alpha_1) \simeq h(\alpha_2) \to h(\alpha_3)$. We distinguish two cases according to the equation that has been put in $\mathcal{E}_M$ by case (a) or (b) of Definition 10.2.1.
In case (a) let $A_1 = \tau(\lambda x.P_1)$, $A_2 = \tau(x)$ and $A_3 = \tau(P_1)$. We have by definition $A_k = h(\alpha_k)$ $(k = 1, 2, 3)$. By Lemma 10.1.3(ii) we must have $A_1 \simeq A_2 \to A_3$.
Case (b) is handled similarly, using Lemma 10.1.3(iii). ■

If the type algebra $\mathcal{A}$ is defined via an s.r. $\mathcal{R}$ over a set $\mathcal{A}$ of types the problem of deciding whether M has a type in $(\lambda\mathcal{R})$ amounts to finding a type algebra homomorphism from $h : \mathbb{T}_M[\mathcal{R}_M] \to \mathbb{T}[\mathcal{R}]$, that is to find a solution of $\mathcal{R}_M$ in $\mathcal{R}$. By Theorems 9.3.18 we then have the following property, that by theorem 9.3.14 applies also to $(\lambda\mathcal{R}^*)$:

10.2.5. THEOREM. *Given a pure λ-term M and a proper s.r. $\mathcal{R}$ it is decidable whether*
(i) *M has a type in $(\lambda\mathcal{R})$;*
(ii) *M has a type in $(\lambda\mathcal{R}^*)$.*

10.2.6. REMARK. (i) $\mathcal{E}_M$, Γ_M and T_M are the generalization of the notion of principal type and basis scheme for simple types (see **?**, **?**). In typing a term in the simple type assignment system we require that $\mathcal{R}_M$ can be solved in an initial type algebra $\mathbb{T}$. In this case $\mathcal{R}_M$ simply defines the substitution which applied to Γ_M and T_M gives the principal typing of M in the usual sense. Theorem 10.2.4 is then a generalization of the Principal Type Theorem (Theorem 2.2.14) for Simple Curry's assignment system.
(ii) By Remark 9.3.20 each homomorphism $h : \mathcal{A}_M \to \mathbb{T}[\mathcal{R}]$ can be written as $h' \circ \overline{h_k}$ where h' is an homomorphism of $\mathbb{T}[\mathcal{R}]$ in itself and the $\overline{h_k}$ are a finite set of homomorphisms determined by the (possibly many) choices to solve $\mathcal{R}_M$ in $\mathcal{R}$. Obviously we have $\overline{h_k}(\Gamma_M) \vdash_{\lambda\mathcal{R}} M : \overline{h_k}(T_M)$. Then the $(\overline{h_k}(\Gamma_M), \overline{h_k}(T_M))$ can be seen as a finite set of principal pairs for M in $\mathcal{R}$, in the sense of Theorem 2.2.14, from which we can generate al possible other typings of M in $\mathcal{R}$ via homomorphisms.

A related decidability result is the following

10.2.7. THEOREM. *Given a pure λ-term M , a proper s.r. $\mathcal{R}$ over $\mathbb{T}$, a type context Γ and a type A. It is decidable whether*
(i) *$\Gamma \vdash_{\lambda\mathcal{R}} M : A$ and*
(ii) *$\Gamma \vdash_{\lambda\mathcal{R}^\sharp} M : A$*

PROOF (Sketch). We prove (i), (ii) follows immediately by Theorem 9.3.14. Assume $\Gamma \vdash_{\lambda\mathcal{R}} M : A$.

Γ contains exactly the variables free in M. By Theorem 10.2.4 $\Gamma \vdash_{\lambda\mathcal{R}} M : A$ iff there is a type algebra homomorphism $h : \mathcal{A}_M \to \mathbb{T}[\mathcal{R}]$ such that $h(T_M) =_{\mathcal{R}} A$ and $h(\Gamma_M) =_{\mathcal{R}} \Gamma$. Then there must be at least one solution of $\mathcal{E}_M$ in $\mathcal{R}$. By Remark 9.3.20 we must have $h = h'' \circ \overline{h_i}$ where $\overline{h_i}$ is one of the solutions built in the proof of 9.3.18 and h'' is a type algebra homomorphism of $\mathbb{T}[\mathcal{R}]$ in itself. Moreover h'' must be such that:

- $h''(\overline{h_i}(T_M)) = A$
- $h''(\overline{h_i}(\alpha_i)) = A_i$ where x is a variable which occurs free in$M$, $x : A_i \in \Gamma$ and $\pi(x) = \alpha_i$ is as in Definition 10.2.1.

We now need the following property.
Property. Let $\mathcal{R}(\vec{X})$ be an s.r. over $\mathbb{T} = \mathbb{T}(\mathbb{A})$ and $\mathcal{S} = \{\langle A_i, B_i\rangle \mid 1 \leq i \leq n\}$ be a set of pairs of types belonging to $\mathbb{T}[\vec{X}]$. Then it is decidable whether there is a syntactic type algebra homomorphism $h : \mathbb{T}[\mathcal{R}] \to \mathbb{T}[\mathcal{R}]$ such that $h(A_i) =_{\mathcal{R}} B_i$ for all $1 \leq i \leq n$ (in this case we say that h solves $\mathcal{S}$).
Proof. Let's define a sequence $\mathcal{S}_m$ $(m \geq 0)$ where $\mathcal{S}_m$ is either a set of pair of types or FAIL. Let $\mathcal{S}_0 = \mathcal{S}$ and define $\mathcal{S}_{m+1}$ from $\mathcal{S}_m$ in the following way:

(a) If there is a pair $\langle A_1 \to A_2, B\rangle \in \mathcal{S}_m$ then:
- If $B \neq_{\mathcal{R}} B_1 \to B_2$ for all types B_1 and B_2 then $\mathcal{S}_{m+1} =$ FAIL.
- Otherwise let $\mathcal{S}_{m+1} = \mathcal{S}_m - \{\langle A_1 \to A_2, B\rangle\} \cup \{\langle A_i, B_i\rangle \mid i = 1, 2\}$.

(b) If there is a pair $\langle X, B\rangle \in \mathcal{S}_m$ where $X = A \in \mathcal{R}$ then $\mathcal{S}_{m+1} = \mathcal{S}_M \cup \{\langle A, B\rangle\}$;

(c) Otherwise take $\mathcal{S}_{m+1} = \mathcal{S}_m$

It is immediate to prove that, for each $m \geq 0$ h solves $\mathcal{S}$ iff h solves $\mathcal{S}_m$. In case (b), in particular, note that since h must be a homorphism we must have $h(X) =_{\mathcal{R}} h(A)$ but since we want also $h(X) =_{\mathcal{R}} B$ this is equivalent, by transitivity, to force $h(X) =_{\mathcal{R}} B$ and $h(A) =_{\mathcal{R}} B$.
Write $\mathcal{S}_m^{fun}$ for the subset of $\mathcal{S}_m$ containing only pairs of the shape $\langle a, B\rangle$ where a is atomic. It is then easy to show the following facts:
- either $\mathcal{S}_m =$ FAIL for some $m > 0$ or there is an $N > 0$ such that for all $m > N$ we have $\mathcal{S}_m^{fun} = \mathcal{S}_N^{fun}$
- h solves $\mathcal{S}_m$ iff h solves $\mathcal{S}_N^{fun}$.

Now for each $a \in \mathbb{A} \cup \vec{X}$ let $\mathcal{B}_a = \{B \mid a = B \in \mathcal{S}_N\}$. Then for all $a \in \mathbb{A} \cup \vec{X}$ such that $\mathcal{B}_a \neq \emptyset$ and for all $B', B'' \in \mathcal{B}_a$ check that $B' =_{\mathcal{R}} B''$. If this is not true then report failure. Otherwise set $h(a) = B$ for any $B \in \mathcal{B}_{a_j}$.
Moreover set $h(a) = a$ for all atoms a such that $\mathcal{B}_a = \emptyset$. It is straightforward to prove that the above process terminates without failure iff h solves $\mathcal{S}$.

Example Let $\mathcal{R} = \{X = Y \to X, Y = X \to Y\}$ and let $\mathcal{S} = \{\langle X, Y\rangle\}$. Then we have
- $\mathcal{S}_1 = \{\langle X, Y\rangle, \langle Y \to X, Y\rangle\}$
- $\mathcal{S}_2 = \{\langle X, Y\rangle, \langle Y, X\rangle\}$

It is easy to verify that $\mathcal{S}_m^{fun} = \mathcal{S}_2^{fun}$ for all $m > 2$ and then $N = 2$. So let h be defined as $h(X) = Y$ and $h(Y) = X$. Note that setting only $h(X) = Y$ would be wrong since h defined in this way is not a homomorphism. This shows the necessity of step b in the definition of h.

From the above property, and from the fact that the possible solutions $\overline{h_i}$ of $\mathcal{E}_M$ are finite we get the result. ■

10.2.8. Remark (About Term Constants). If M can contain occurrences of term constants c then we must assume a type $\tau(c)$ for each of them. In this case it is standard to assume that a set K of constant types (for instance a type **int** for the set of integers). For some constants c then $\tau(c)$ can contain also constant types (for instance $\tau(n) = \mathbf{int}$ for all numerals n, $\tau(+) = \mathbf{int} \to \mathbf{int} \to \mathbf{int}$, etc.). In this case usually one does not want that constant types are equated to anything else but themselves, and we consider a type error to assume, for instance, that **int** is equivalent to an arrow type. We say that $\mathcal{R}_M$ is *consistent* if it does not imply any equation $\kappa = A$ where k is a constant type and A is a either a different constant type or a non atomic type expression. Then as a consequence of Theorem 10.2.4 a term M can be typed (w.r.t. some type algebra) iff $\mathcal{R}_M$ is consistent.

We can easily take into account constants in the construction of $\mathcal{R}_M$ in Definition 10.2.1 by taking $\pi(c) = \tau(c)$ for each constant c occurring in M. We can easily prove that $\mathcal{R}_M$ is consistent iff for each atomic constant type κ $\mathcal{R}_M$ does not contain equations $\kappa = a_1, a_1 = a_2, \ldots, a_n = C$ where $N \geq 0$ and C is either a constant type different from κ or a non atomic type expression. It is then easily decidable whether $\mathcal{R}_M$ is consistent.

All the results given in this section still hold if we consider terms with constants (in the previous sense). In some cases this makes these results more interesting. For instance Problem 1. (to decide whether a given term has a type w.r.t. some s.r.) is not more trivial since there are terms, like (3 3), which have no type at all w.r.t. any s.r.. By the subject reduction theorem, moreover, we still have that a term to which it a type can be given in any system with recursive types can not produce bad applications during its evaluation. This is one of the interesting applications of these systems in Computer Science.

Finding types in systems with μ types (sketch)

In $\mathbb{T}_\mu$ all possible recursive types are present and so in considering problems about assigning types to λ-terms there is no the parameter $\mathcal{R}$ describing the recursive types available in the system. The kind of problems we are interested in are so similar to those for simple types.

The typeability problem is indeed trivial since all pure terms have type $\mu\alpha.\alpha \to \alpha$ (see Exercise 8.7.12). This type is however meaningless and, as in the case of the previous section, we are interested in more informative types and, in particular, to the possibility of characterizing all types of a given term.

The notions of principal type and principal pair (Definition 2.2.13) can be generalized in a straightforward way to μ-types considered modulo strong equivalence.

10.2.9. Definition. Let $\vec{S} = \langle S_1, ..., S_n \rangle = Sol_\mu(\mathcal{R}_M)$. Then define

(i) $T^*_M = T_M[\vec{\xi} := \vec{S}]$

(ii) $\Gamma^*_M = \Gamma_M[\vec{\xi} := \vec{S}]$

As pointed out in Remark **??** the solutions $Sol_\mu(\mathcal{R}_M)$ of $\mathcal{R}_M$ in $\mathbb{T}$ are not unique but they are all equivalent modulo $=^*$. We have the following version of the principal type Theorem 2.2.14 for $\mathbb{T}^*_\mu$.

10.2.10. THEOREM. (i) $\Gamma^*_M \vdash_{\lambda\mu^*} M : T^*_M$;
(ii) *If* $\Gamma \vdash_{\lambda\mu^*} M : A$ *then there is a substitution* $s : V \to \mathbb{T}_\mu$ *such that* $A =^* s(T^*_M)$ *and* $\Gamma \supseteq s(\Gamma^*_M)$ *(modulo* $=^*$*).*

PROOF. By Theorem 10.2.4, Lemma 8.6.9 (i) and Remark 8.6.13. ■

Indeed as remarked in section 8.5 μ-types are just notations for regular trees. The proof of the following Theorem is left as an exercise (see Exercise 10.4.4)

10.2.11. THEOREM. *Given a term* M*, a basis* Γ *and a type* A *it decidable wether* $\Gamma \vdash_{\lambda\mu^*} M : A$.

As for weak equivalence we have, by Theorem 10.2.4 that the set of possible typings of a term M in $(\lambda\mu)$ (i.e. the set of all Γ, A such that $\Gamma' \vdash_{\lambda\mu} M : A$ for all $\Gamma' \supseteq \Gamma$) is recursively enumerable. Moreover, by Lemma 8.3.22 (**??**) and Theorem 9.1.4, given a typing deduction in $(\lambda\mathcal{R})$ (for any s.r. $\mathcal{R}$) we can easily build one in $(\lambda\mu)$ (via the construction of Definition 9.1.1). The notion of principal type however cannot be directly expressed in $(\lambda\mu)$ owing to the many incomparable solution that the same s.r. can have in $\mathbb{T}_\mu$.

10.3. Strong normalization

As shown by the examples in Section 8.1 type systems with recursive types do not have, in general, the strong normalization property. This is easy: in a type $D = D \to D$ all untyped lambda terms can be typed. By restricting the use of the μ operator in type formation, however, it is possible to define systems in which the strong normalization property hold. The strong normalization theorem in this section comes from **?** and **?**. We will prove in detail the strong normalization theorem for an assignment system with the μ-types. Since we have shown in 9.1 that any sr can be solved in $\mathbb{T}_\mu$, this result can be extended immediately to assignment systems defined via an sr.

Strong normalization for μ-types

Using recursive types, in general, we can assign types to non-normalizing terms, like the fixed-point operator $\mathsf{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$, or even to unsolvable ones, like $\mathbf{\Omega} \equiv (\lambda x.xx)(\lambda x.xx)$. However, there is a subset $\mathbb{T}^+_\mu \subseteq \mathbb{T}_\mu$ such that terms that can be typed using only types in $\mathbb{T}^+_\mu$ are strongly normalizable. The set $\mathbb{T}^+_\mu$ completely characterizes the types assuring normalization in the following sense. For any type $T \in \mathbb{T}_\mu$, such that $T \notin \mathbb{T}^+_\mu$, it is possible to define a non normalizing term which can be typed using only the type T and its subtypes.

10.3.1. DEFINITION. (i) The notion of *positive* and *negative* occurrence of a subtype in a type of $\mathbb{T}_\mu$ is defined (inductively) by the following clauses.

(1) A is *positive* in A.
(2) If A is *positive* (*negative*) in B then A is *positive* (*negative*) in $C \to B$ and *negative* (*positive*) in $B \to C$.
(3) If A is *positive* (*negative*) in B then A is *positive* (*negative*) in $\mu t.B$.
(ii) We say that a type variable t which occurs free in A is *positive (negative)* in A if all occurrences of t in A are *positive* (*negative*).

Note that A is positive (negative) in B if A occurs in B on the left hand side of an even (odd) number of '$\to$'. Let this number be the *level* of the occurrence of A in B. For instance t is positive in $((\mu v.v \to t) \to u) \to w$, since the occurrence of t is at level 2, but not in $((\mu v.t \to v) \to u) \to w$, since now it is at level 3.

10.3.2. Definition. (i) A type $A \in \mathbb{T}_\mu$ is called *positive* if for every $\mu t.B \subseteq_\mu A$ all occurrences of t in B are positive.
(ii) $\mathbb{T}^+_\mu = \{A \in \mathbb{T}_\mu \mid A \text{ is positive}\}$.
(iii) Let $(\lambda\mu+)$ be the type assignment system defined as $(\lambda\mu)$, but restricting the set of types to $\mathbb{T}^+_\mu$.

10.3.3. Examples. (i) $\mu t.a \to t \in \mathbb{T}^+_\mu$.
(ii) $\mu t.t \to a \notin \mathbb{T}^+_\mu$.
(iii) $\mu t.t \to t \notin \mathbb{T}^+_\mu$.
(iv) $((\mu t.a \to t) \to b) \in \mathbb{T}^+_\mu$.
(v) $(\mu t.(a \to t) \to b) \notin \mathbb{T}^+_\mu$.

10.3.4. Definition. Let $A, B \in \mathbb{T}_\mu$.
(i) B is a subterm modulo μ, notation $B \subseteq_\mu A$ if B is a subtype of A.
(ii) $\mathrm{Sub}_\mu(A) = \{B \mid B \subseteq_\mu A\}$.

From the deninition of $\mathbb{T}^+_\mu$ the following follows immediately.

10.3.5. Lemma. *If $A \in \mathbb{T}^+_\mu$ and $B \subseteq_\mu A$, then $B \in \mathbb{T}^+_\mu$.*

Moreover, types in $\mathbb{T}^+_\mu$ are preserved by $=_\mu$.

10.3.6. Lemma. *If $A \in \mathbb{T}^+_\mu$ and $A =_\mu B$ then $B \in \mathbb{T}^+_\mu$.*

Proof. By induction on the proof that $A =_\mu B$. ■

Obviously any deduction in $(\lambda\mu+)$ is also a deduction in $(\lambda\mu)$. So any typing judgement provable in $(\lambda\mu+)$ is also a typing judgement in $(\lambda\mu)$.

Positive recursive types are often referred to as *inductive types* in the literature, since it is possible to define for them an induction principle, see **?**. In that paper the result is proved in fact for a stronger system, defined as an extension of second order typed lambda calculus with positive recursive types and induction and coinduction operators for each type.

The strong normalization proof is based on the notion of saturated set introduced in **?**, which is in turn based on Tait's method presented in Section 2 provided with an impredicative twist. We will prove it for the system $(\lambda\mu)$ à la Curry, but the proof extends to the systems with explicit typing, see section **??**. Let SN be the set of strongly normalizing type free λ-terms.

10.3.7. Definition. (i) A subset $\mathcal{X} \subseteq \mathtt{SN}$ is *saturated* if the following conditions hold.

(1) $\mathcal{X}$ is closed under reduction, i.e. for all $M, N \in \Lambda$

$$M \in \mathcal{X} \;\&\; M \twoheadrightarrow_\beta N \;\Rightarrow\; N \in \mathcal{X}.$$

(2) For all variables x one has

$$\forall n \geq 0 \forall R_1, \ldots, R_n \in \mathtt{SN}. xR_1 \ldots R_n \in \mathcal{X}.$$

(3) For all $Q \in \mathtt{SN}$ one has

$$P[x := Q]R_1 \ldots R_n \in \mathcal{X} \implies (\lambda x.P)QR_1 \ldots R_n \in \mathcal{X}$$

(ii) Let $\mathtt{SAT}$ be the set of all saturated subsets of $\mathtt{SN}$.

10.3.8. Proposition. $\mathtt{SAT}$ *is a complete lattice under the operations of set inclusion, with (l.u.b.) defined by set union and (g.l.b.) defined by set intersection.*

The top element of $\mathtt{SAT}$ is $\mathtt{SN}$ while its bottom element $\bot_{\mathtt{SAT}} = \bigcap_{\mathcal{X} \in \mathtt{SAT}} \mathcal{X}$ is the least set containing all terms of the shape $xR_1 \ldots R_n$ ($R_i \in \mathtt{SN}$) and that satisfies (3) of Definition 10.3.7(i). We recall that a function $f : \mathtt{SAT} \to \mathtt{SAT}$ is *monotonic* if $X \subseteq Y$ (where $X, Y \in \mathtt{SAT}$) implies $f(X) \subseteq f(Y)$ and *anti-monotonic* if $X \subseteq Y$ implies $f(Y) \subseteq f(X)$. If $f : \mathtt{SAT} \to \mathtt{SAT}$ is a monotonic function, then

$$fix(f) \;=\; \bigcup \{x \mid f(x) \supseteq x\}$$

is the least fixed point of f.

10.3.9. Definition. The operation $\Rightarrow$: $\mathtt{SAT} \times \mathtt{SAT} \to \mathtt{SAT}$ is defined by:

$$(\mathcal{X} \Rightarrow \mathcal{Y}) = \{M \mid \forall N \in \mathcal{X}.(MN) \in \mathcal{Y}\}$$

The proof of the following proposition is standard.

10.3.10. Proposition. $\Rightarrow$ *is well defined, i.e. for all* $\mathcal{X}, \mathcal{Y} \in \mathtt{SAT}$ $\mathcal{X} \Rightarrow \mathcal{Y} \in \mathtt{SAT}$. *Moreover* $\Rightarrow$ *is anti-monotonic in its first argument and monotonic in its second argument, i.e. if* $\mathcal{X}' \subseteq \mathcal{X}$ *and* $\mathcal{Y} \subseteq \mathcal{Y}'$, *then* $(\mathcal{X} \Rightarrow \mathcal{Y}) \;\subseteq\; (\mathcal{X}' \Rightarrow \mathcal{Y}')$.

We now define an interpretation of types in $\mathbb{T}^+_\mu$ as elements of $\mathtt{SAT}$ and of terms as elements of Λ. This is inspired by standard semantical notions, see Chapter 11. Let a *type environment* be a function $\tau : \mathbb{A} \to \mathtt{SAT}$.

10.3.11. Definition. The interpretation of a type $A \in \mathbb{T}^+_\mu$ under a type environment τ, notation $[\![A]\!]^\tau$, is defined by

(i) $[\![t]\!]^\tau = \tau(t)$.

(ii) $[\![A \to B]\!]^\tau = [\![A]\!]^\tau \Rightarrow [\![B]\!]^\tau$.

(iii) $[\![\mu t.A]\!]^\tau = \begin{cases} \bot_{\mathtt{SAT}}, & \text{if } \mu t.A \text{ is circular;} \\ fix(\lambda \mathcal{X}\, \mathtt{SAT}.[\![A]\!]^{\tau[t:=\mathcal{X}]}), & \text{otherwise.} \end{cases}$

10.3.12. LEMMA. (i) *If t is positive (negative) in $A \in \mathbb{T}^+_\mu$ then*

$$\lambda\mathcal{X} : \mathtt{SAT}.[\![A]\!]^{\tau[t:=\mathcal{X}]}$$

is a monotonic (anti-monotonic) function over the complete lattice SAT.

(ii) $[\![-]\!]^\tau$ *is well defined, i.e.* $[\![A]\!]^\tau \in \mathtt{SAT}$ *for all types* $A \in \mathbb{T}^+_\mu$ *and all type environments* τ.

PROOF. Points (i) and (ii) by simultaneous induction on A. ■

10.3.13. LEMMA. (i) $[\![A[t := B]]\!]^\tau = [\![A]\!]^{\tau[t:=[\![B]\!]^\tau]}$

(ii) $[\![\mu t.A]\!]^\tau = [\![A[t := \mu t.A]]\!]^\tau$. ■

PROOF. (i) by structural induction on A.
(ii) If A is circular the proof is trivial. Otherwise, since $[\![\mu t.A]\!]^\tau$ is a fixed-point of $\lambda\mathcal{X} : \mathtt{SAT}.[\![A]\!]^\tau[t := \mathcal{X}]$, we have $[\![\mu t.A]\!]^\tau = [\![A]\!]^{\tau[t:=[\![\mu t.A]\!]^\tau]}$ and the proof follows by point (i). ■

Using Lemma 10.3.13, we can immediately prove that type interpretation is preserved by weak equivalence

10.3.14. LEMMA. *let$A, B \in \mathbb{T}^+_\mu$. If $A =_\mu B$ then $[\![A]\!]^\tau = [\![B]\!]^\tau$.*

We define now an interpretation of terms in Λ. Let a *term environment* be a function from the set of term variables V to elements of Λ.

10.3.15. DEFINITION. Let $\rho : \mathsf{V} \to \Lambda$ be a term environment. The *evaluation* of a term M under ρ is defined by:

$$[\![M]\!]_\rho \;=\; M[x_1 := \rho(x_1), \ldots, x_n := \rho(x_n)]$$

where $x_1, \ldots, x_n$ $(n \geq 0)$ are the variables free in M.

We define now in a standard way the notion of satisfiability of a statement $\Gamma \vdash_{\lambda\mu+} M : A$ with respect to the previous interpretations of types and terms.

10.3.16. DEFINITION. Let τ be a type environment and ρ be a term environment. Define:
- $\tau, \rho \models M : A$ if $[\![M]\!]_\rho \in [\![A]\!]^\tau$.
  - $\tau, \rho \models \Gamma$ if for all $x : A \in \Gamma \quad \tau, \rho \models x : A$.
  - $\Gamma \models M : A$ if, for all τ and ρ, $\tau, \rho \models \Gamma \implies \tau, \rho \models M : A$

10.3.17. LEMMA. $\Gamma \vdash_{\lambda\mu+} M : A \implies \Gamma \models M : A$.

PROOF. By induction on the derivation of $M : A$ from Γ, using Lemma 10.3.14 in case (equiv). ■

10.3.18. THEOREM. $\Gamma \vdash_{\lambda\mu+} M : A \;\Rightarrow\; M$ *is SN.*

PROOF. By Lemma 10.3 $\Gamma \models M : A$. Let ρ_0 be the term environment defined by $\rho_0(x) = x$ for all $x \in \mathsf{V}$. Trivially, for any type environment τ one has $\tau, \rho_0 \models \Gamma$, and so $\tau, \rho_0 \models M : A$. Therefore $[\![M]\!]_{\rho_0} = M \in [\![A]\!]^\tau \subseteq \mathtt{SN}$. ■

We can reformulate this as follows. Define the set of terms typable from a set of types as follows.

10.3.19. DEFINITION. Let $\mathcal{X} \subseteq \mathbb{T}_\mu$. Then

$$\mathrm{Typable}(\mathcal{X}) = \{M \in \Lambda \mid M \text{ can be typed using only types in } \mathcal{X}\}.$$

Here 'using only types in $\mathcal{X}$' means that all types in the derivation (including the context) should be elements of $\mathcal{X}$.

10.3.20. COROLLARY. $B \in \mathbb{T}^+_\mu \;\Rightarrow\; \mathrm{Typable}(\mathrm{Sub}_\mu(B)) \subseteq \mathtt{SN}$.

PROOF. Immediate from Theorem 10.3.18 and Lemma 10.3.5. ■

Note that this Corollary also implies Theorem 10.3.18. Indeed, suppose that $G \vdash_{\lambda\mu+} M : A$. Let $\mathcal{X} = \{A_1, \ldots, A_n\} \subseteq \mathbb{T}^+_\mu$ be the set of types used in this deduction. Then we can apply the Corollary to $B = A_1 \to A_2 \to \ldots \to A_n \in \mathbb{T}^+_\mu$.

It is easy to extend this result and its proof to systems with other type constructors like cartesian product and disjoint union since all these operators are monotonic in both arguments.

Conversely, all μ-types which do not belong to $\mathbb{T}^+_\mu$ allow to type a non normalizable term. We see this in the next theorem. The construction is due to **?**.

Let $\vec{A}$ denote a sequence of types $A_1, \ldots, A_n$ $(n \geq 0)$ and $\vec{A} \to B$ denote the type $A_1 \to \ldots \to A_n \to B$. Moreover if $\vec{M}$ is a sequence of terms $M_1, \ldots, M_n$ then $\vec{M} : \vec{A}$ denotes the set of statements $M_1 : A_1, \ldots, M_n : A_n$.

10.3.21. THEOREM (**?**). *Let $B \in \mathbb{T}_\mu$ be a type such that there is a negative occurrence of a variable t in it. Then there is a term N without normal form that can be typed using only subtypes of $\mu t.B$ modulo $=_\mu$. In formula*

$$B \notin \mathbb{T}^+_\mu \;\Rightarrow\; \mathrm{Typable}(\mathrm{Sub}_\mu(B)) \not\subseteq_\mu \mathsf{N}.$$

PROOF. Assume that t has a negative occurrence in B. Then there is an integer $n \geq 0$ and types Q_i, $\vec{A}_i$, B_j with $0 \leq i \leq 2n+1$, $1 \leq j \leq 2n+1$ such that

- $B =_\mu Q_{2n+1}$;
- $Q_i = \vec{A}_i \to Q_{i-1} \to B_i$, with $1 \leq i \leq 2n+1$;
- $Q_0 = \vec{A}_0 \to t$,

where $Q_0, \ldots, Q_{2n+1}$ are in head μ-free form (see 9.2). This can be obtained by successively unfolding B.

We assume $n > 0$, the case $n = 0$ being left as an exercise.

Define now the terms N_i with $0 \leq i \leq 2n+1$ of type Q_i.

- $N_0 = \lambda \vec{x}_0 . N_{2n+1}$;
- $N_1 = \lambda \vec{x}_1 . \lambda y_0 . f^{2n+1,1}(y_0 \vec{z}_0 \vec{z}_{2n+1} y_{2n})$;

- $N_i = \lambda\vec{x}_i.\lambda y_{i-1}.f^{i-1,i}(y_{i-1}\vec{z}_{i-1}N_{i-2})$, with $2 \leq i \leq 2n+1$,

where $\vec{x}_i : \vec{A}_i$, $\vec{z}_i : \vec{A}_i$, $y_i : Q_i$, with $0 \leq i \leq 2n+1$ and $f^{k,h} : B_k \to B_h$, with $1 \leq k, h \leq 2n+1$. Note that these terms are well defined, since the terms N_j for odd values of j are defined from N_1 which is the starting point of the whole definition and the terms N_j for even values of j are defined from N_0 which is defined from N_{2n+1}. Moreover, note that y_i do not occur free in N_{i+1}, for $0 \leq i < 2n$, and y_{2n} occurs free in N_{2i+1}, for $1 \leq i < n$, but not in N_{2n+1}, where it is bound.

Now consider the term $N = (N_{2n+1}\vec{z}_{2n+1}N_{2n})$. It is straightforward to verify that N can be typed in context

$$\Gamma_0 = \{f^{k,h} : B_k \to B_h \mid 1 \leq k, h \leq 2n+1\} \cup \{\vec{z}_i : \vec{A}_i \mid 0 \leq i \leq 2n+1\},$$

using only subtypes of $\mu t.B$ modulo $=_\mu$. The term N has the following (unique) reduction which is infinite.

$$\begin{array}{rcl}
N & = & N_{2n+1}\vec{z}_{2n+1}N_{2n} \\
& \to_\beta & f^{2n,2n+1}(N_{2n}\vec{z}_{2n}N_{2n-1}[y_{2n} := N_{2n}]) \\
& \to_\beta & f^{2n,2n+1}(f^{2n-1,2n}(N_{2n-1}[y_{2n} := N_{2n}]\vec{z}_{2n-1}N_{2n-2})) \\
& \twoheadrightarrow_\beta & f^{2n,2n+1}(\ldots(f^{1,2}(N_1[y_{2n} := N_{2n}]\vec{z}_1N_0))\ldots) \\
& \to_\beta & f^{2n,2n+1}(\ldots(f^{1,2}(f^{2n+1,1}(N_0\vec{z}_0\vec{z}_{2n+1}N_{2n})))\ldots) \\
& \to_\beta & f^{2n,2n+1}(\ldots(f^{1,2}(f^{2n+1,1}(N_{2n+1}\vec{z}_{2n+1}N_{2n})))\ldots) \\
& = & f^{2n,2n+1}(\ldots(f^{1,2}(f^{2n+1,1}N))\ldots). \blacksquare
\end{array}$$

Theorem 10.3.21 can be immediately extended to ($\lambda\mu$-Ch) and ($\lambda\mu$-Ch$_0$).

10.3.22. Example. Let $T = ((t \to \alpha) \to \alpha) \to \alpha$. The we have $n = 1$, $B_i = \alpha$ for $i = 1, 2, 3$ and $Q_0 = t$, $Q_1 = t \to \alpha$, $Q_2 = (t \to \alpha) \to \alpha$, $Q_3 = ((t \to \alpha) \to \alpha) \to \alpha$ and $Q = \mu t.Q_3$. Moreover, we have $\Gamma_0 = \{f^{1,2} : \alpha \to \alpha,\ f^{2,3} : \alpha \to \alpha,\ f^{3,1} : \alpha \to \alpha\}$ and

$$\begin{array}{rcl}
N_0 = N_3 & = & \lambda y_2.f^{2,3}(y_2\, N_1) \quad = \quad \lambda y_2.f^{2,3}(y_2\, \lambda y_0.f^{1,2}(y_0\, y_2)) \\
N_1 & = & \lambda y_0.f^{1,2}(y_0\, y_2) \\
N_2 & = & \lambda y_1.f^{1,2}(y_1\, N_0).
\end{array}$$

Then $\Gamma_0 \vdash_{\lambda\mu} (N_0\, N_2) : \alpha$ and we need the types $Q \to \alpha$ and α to establish this type assignment. We have the infinite reduction

$$(N_0\, N_2) \to_\beta f^{2,3}(N_2\, N_1) \twoheadrightarrow_\beta f^{2,3}(f^{1,2}(f^{3,1}(N_0\, N_2))) \to \ldots\ .$$

The example can be simplified by realizing we do not need the $f^{i,j}$ and then we can take $N_0 \equiv \lambda x.\langle x\rangle$, $N_1 \equiv \langle N_0\rangle$, where $\langle P\rangle \equiv \lambda z.zP$. Then $\langle P\rangle Q \to_\beta QP$ and hence

$$N_0\langle N_0\rangle \to_\beta \langle N_0\rangle\langle\langle N_0\rangle\rangle \to_\beta \langle\langle N_0\rangle\rangle N_0 \to_\beta N_0\langle N_0\rangle \to_\beta \ldots\ .$$

Theorems 10.3.18 and 10.3.21 show that $\mathbb{T}_\mu^+$ is the largest subset $\mathcal{X} \subseteq \mathbb{T}_\mu$ such that if a term can be typed using the types in $\mathcal{X}$, then M is strongly normalizing.

10.3.23. Corollary. *Let $B \in \mathbb{T}_\mu$. Then*
(i) $\mathrm{Typeable}(\mathrm{Sub}_\mu(B)) \subseteq \mathsf{N} \Rightarrow B \in \mathbb{T}^+_\mu$.
(ii) $B \in \mathbb{T}^+_\mu \iff \mathrm{Typeable}(\mathrm{Sub}_\mu(B)) \subseteq \mathsf{SN}$.
$\iff \mathrm{Typeable}(\mathrm{Sub}_\mu(B)) \subseteq \mathsf{N}$.

Proof. (i) By Theorem 10.3.21.
(ii) $B \in \mathbb{T}^+_\mu \Rightarrow \mathrm{Typeable}(\mathrm{Sub}_\mu(B)) \subseteq \mathsf{SN}$, by Corollary 10.3.20,
$\Rightarrow \mathrm{Typeable}(\mathrm{Sub}_\mu(B)) \subseteq \mathsf{N}$
$\Rightarrow B \in \mathbb{T}^+_\mu$, by (i). ■

10.3.24. Corollary. *Let $B \in \mathbb{T}_\mu$. Then*

$$\mathrm{Typeable}(\mathrm{Sub}_\mu(B)) \subseteq \mathsf{N} \iff \mathrm{Typeable}(\mathrm{Sub}_\mu(B)) \subseteq \mathsf{SN}.$$

This reminds us of the old question whether $M \in \mathtt{SN} \Rightarrow M \in \mathsf{N}$. This is not the case. $\mathsf{KI}\boldsymbol{\Omega}$ has a nf, but is not $\mathtt{SN}$. The reason is that there is the subterm $\boldsymbol{\Omega}$ that has no nf. There is even a term $M_0 = (\lambda z.(\lambda xy.y)(zz))(\lambda z.(\lambda xy.y)(zz))$, such that every subterm has a nf, but the term itself is not SN. Note that M_0, the fixed-point of **false** $:= \lambda xy.y$. Nevertheless this term is not in contradiction with Corollary 10.3.24.

The strong normalization theorem holds also for the typed system ($\lambda\mu$-Ch) and ($\lambda\mu$-Ch$_0$) (which is indeed weaker than ($\lambda\mu$-Ch)). In the case of ($\lambda\mu$-Ch$_0$) it is easy to prove that the reduction rules (μ 1) and (μ 2) can not cause infinite reduction sequences.

On the other hand it is less natural to define positive types for $\vdash_{\lambda\mu^*}$.

Strong normalization for simultaneous recursion

We now define a notion similar to that of positive element also for recursive types defined via an sr.

10.3.25. Definition. We say that a sr $\mathcal{R}$ is *inductive* if for no undeterminate X we have $X =_{\mathcal{R}} C$ for some non atomic type expression C in which X has a negative occurrence.

10.3.26. Example. (i) Let $\mathcal{R}_0 = \{X_0 = X_1 \to X_0, X_1 = X_0 \to X_1\}$. Then $\mathcal{R}_0$ is inductive . Note that by unfolding we can get $X_0 =_{\mathcal{R}_0} (X_0 \to X_1) \to X_0)$ (and so on) but both occurrences of X_0 are positive.
(ii) Let $\mathcal{R}_1 = \{X_0 = X_1 \to X_1, X_1 = X_0 \to X_1\}$. Then $\mathcal{R}_1$ is not inductive. In fact $X_1 =_{\mathcal{R}_1} (X_1 \to X_1) \to X_1$.

By a routine check we get the following useful property.

10.3.27. Proposition. *A $\mathcal{R}$ is inductive iff $Sol_\mu(\mathcal{R}) \in (\mathbb{T}^+_\mu)^*$.*

By this Proposition it is easily decidable if a given sr is inductive. As consequence of Theorems 10.3.18 and 10.3.21 we can characterize those sr which can type only terms that are strongly normalizing.

10.3.28. THEOREM. *Let $\mathcal{R}$ be an inductive sr. Then*

$$\Gamma \vdash_{\lambda\mathcal{R}} M : A \;\Rightarrow\; M \text{ is SN.}$$

PROOF. Let $\mathcal{R} = \mathcal{R}(X_1, \ldots, X_n)$ and $h : \mathbb{T}[\mathcal{R}] \to \mathbb{T}^+_\mu$ be the type algebra morphism defined by $h(X_i) = S_i$, with $1 \leq i \leq n$, where $S_1, \ldots, S_n = Sol_\mu(\mathcal{R})$ and $h(\alpha) = \alpha$, for all other atomic types α. By Proposition 10.3.27 we have $S_i \in \mathbb{T}^+_\mu$, for $1 \leq i \leq n$. Then for all $B \in \mathbb{T}[\vec{X}]$ we have $h(B) \in \mathbb{T}^+_\mu$ and by Lemma 8.1.15 we get $h(\Gamma) \vdash_{\lambda\mu+} M : h(A)$. Now Theorem 10.3.28 applies. ■

10.3.29. THEOREM. *Let $\mathcal{R} = \{X_i = A_i \mid 1 \leq i \leq n\}$ be a non inductive sr. Then there is a term N without normal form such that for some basis Γ we have $\Gamma \vdash_{\lambda\mathcal{R}} N : X_i$.*

PROOF. The proof is the same of Theorem 10.3.21 where X_i replaces t and $\mu t.T$ and $=_{\mathcal{R}}$ replaces $=_\mu$. Note also that no other recursive types except those defined from $\mathcal{R}$ are needed to build the non normalizing term N.

Type Inference

The typability of a term M is decidable also with respect to inductive sr (or in $(\lambda\mu+)$). This property follows easily from the following Lemma.

10.3.30. LEMMA. (i) *Let $\mathcal{A} = \mathbb{T}/{\approx}$ and $\mathcal{A} = \mathbb{T}'/{\approx'}$ be syntactic type algebra and $h : \mathcal{A} \to \mathcal{A}'$ a morphism. If $A \in \mathbb{T}$ has a positive (negative) occurrence in $B \in \mathbb{T}$ then $h(A)$ has a positive (negative) occurrence in $h(B)$.*

(ii) *Let $\mathcal{R}(\vec{X})$ be a non inductive sr over $\mathbb{T}$. Assume that there is a morphism $h : \mathbb{T}[\mathcal{R}] \to \mathbb{T}[\mathcal{R}']$. The $\mathcal{R}'$ is non inductive.*

PROOF. (i) By induction on B.

(ii) Take $X \in \vec{X}$ such that $X =_{\mathcal{R}} C$ for some C which contains a negative occurrence of X. Since h is a morphism we have that $h(X) =_{\mathcal{R}'} h(C)$. Then $h(X)$ has a negative occurrence in $h(C)$, by (i). Using Lemma 9.3.7), it follows by a simple induction that $h(X)$ must contain a $Y \in \mathrm{dom}(\mathcal{R}')$ such that $Y =_{\mathcal{R}'} C'$ for some C' with negative occurrence Y in it. ■

10.3.31. DEFINITION. Let $M \in \Lambda$. We say that M can be *inductively* typed if it has a type in $(\lambda\mathcal{R})$, for some inductive sr $\mathcal{R}$.

10.3.32. THEOREM. *Let $M \in \Lambda$. Then*

$$M \text{ can be inductively typed} \iff \mathcal{R}_M \text{ is inductive.}$$

PROOF. (i) ($\Leftarrow$) Immediate by Lemma 10.2.3. ($\Rightarrow$) Assume $\Gamma \vdash_{\lambda\mathcal{R}} M : A$, for some non inductive sr $\mathcal{R}$. Then, by Lemma 10.3.30 and Theorem 10.2.4, $\mathcal{R}_M$ cannot be inductive. ■

10.3.33. EXAMPLE. Take the term $\lambda x.xx$. We have seen in Example 10.2.2 that $\mathcal{E}_{\lambda x.xx}$ contains an equation $i_1 = i_1 \to i_2$ and then is not inductive. Then there is no inductive sr which can give a type to $\lambda x.xx$.

Since inductive sr can be solved in $\mathbb{T}^+_\mu$ this implies also that if a term is typable from an inductive sr then it is also typable in $(\lambda\mu+)$. The proof of the converse is left as an exercise.

10.3.34. THEOREM. *Given a term M it is decidable whether it can be given a type in $(\lambda\mu+)$, i.e. if $\Gamma \vdash_{\lambda\mu+} M : A$ for some type A and basis Γ over $\mathbb{T}^+_\mu$.*

10.4. Exercises

10.4.1. Prove that the system ($\lambda\mu$-Ch$_0$) defined in **??** has the subject-reduction per Church. Note that this does not follow from Lemma 10.1.6 since type equivalence is not defined by a type algebra.

10.4.2. Show that $\{x : A \to \mu t.B\} \vdash_{\lambda\mu?} \lambda y.xy : A \to B[t := \mu t.B]$ but $\{x : A \to \mu t.B\} \not\vdash_{\lambda\mu?} x : A \to B[t := \mu t.B]$

10.4.3. Show that if a term has a type in $(\lambda\mu)$ then it has a type also in $(\lambda\mathcal{R})$ from some s.r. $\mathcal{R}$.

10.4.4. Prove Theorem 10.2.11. [Hint. Use Theorems **??**, 9.3.14, ?????].

10.4.5. Design an algorithm to decide directly (i.e. without passing through the translation in $\mathbb{T}^+_\mu$) weather a given s.r. is inductive.

10.4.6. Show that the typings of Example 8.1.12 are all principal.

Chapter 11

Models 21.2.2008:897

Our purpose in the present Chapter is to build concrete type algebras for the interpretation of recursive types. In Section 11.1 we focus on systems à la Curry, where infinitely many types can in general be inferred for each (type-free) λ-term. Accordingly, it is natural to regard the interpretation of a type as a collection of elements of a model of the untyped λ-calculus.

We shall also describe, in Sectionx 11.2, how to build models for explicitly typed systems with recursive types. Classical categories of domains yield straightforward models for these formulations. Beside these, we shall also consider models based on different constructions (like continuous closures or partial equivalence relations) that are of interest in their own.

11.1. Type interpretations in systems à la Curry

Before constructing a concrete interpretation of type inference systems of the form $(\lambda\mathcal{A})$ introduced in Definition 8.1.11, we need to define in general what data are needed to specify such an interpretation. In the sequel, we shall focus on pure λ-terms, i.e., we shall in general deal only with terms in Λ. There is no loss of generality in this choice, as the extension of our results to λ-terms with constants is mostly straightforward.

Anticipating on a notion that will be described in detail in Chapter 17 below, a λ-model $\mathcal{D}$ is defined as a structure $\langle D, F, G\rangle$, where $F : D \to (D \to D)$ and $G : (D \to D) \to D$ satisfy $F \circ G = 1_{(D\to D)}$, and the set $(D \to D)$ contains all λ-definable functions over $\mathcal{D}$. As usual, $\mathcal{D}$ can be turned into an applicative structure by defining $d \cdot e$ as $F(d)(e)$. (In the sequel, we shall identify $\mathcal{D}$ with its underlying set D, when no confusion is likely.)

Given a (term) environment $\rho : \mathsf{V} \to D$, the interpretation of λ-terms in D is defined by structural induction, as follows:

(i) $[\![x]\!]_\rho = \rho(x)$, for $x \in \mathsf{V}$;

(ii) $[\![MN]\!]_\rho = [\![M]\!]_\rho \cdot [\![N]\!]_\rho$;

(iii) $[\![\lambda x.M]\!]_\rho = F(f)$, where $f(d) = [\![M]\!]_{\rho[x\mapsto d]}$.

The interpretation of the set of types of $(\lambda\mathcal{A})$ will make use of a type algebra $\mathcal{S}(D)$ based on D, whose elements are subsets of D.

11.1.1. Definition (Type interpretations). A *type interpretation* of the types in a syntactic type algebra $\mathbb{T}/\approx$ is a type algebra homomorphism $h : \mathbb{T}/\approx \to \mathcal{S}(D)$.

Soundness of $(\lambda\mathcal{A})$ with respect to $\mathcal{S}(D)$ can now be defined formally:

11.1.2. Definition (Satisfaction and soundness). (i) Given a type interpretation h and a term environment ρ, the triple $\langle \mathcal{S}(D), h, \rho\rangle$ *satisfies* a typing statement $M : A$, written $\langle \mathcal{S}(D), h, \rho\rangle \models M : A$, if $[\![M]\!]_\rho \in h(A)$.

(ii) Given a basis Γ, $\langle \mathcal{S}(D), h, \rho\rangle \models \Gamma$ if $\langle \mathcal{S}(D), h, \rho\rangle \models x : A$ for all typing statements $x : A \in \Gamma$.

(iii) We write $\langle \mathcal{S}(D), h, \rho\rangle : \Gamma \models M : A$ when $\langle \mathcal{S}(D), h, \rho\rangle \models \Gamma$ implies $\langle \mathcal{S}(D), h, \rho\rangle \models M : A$.

(iv) We write $\mathcal{S}(D) : \Gamma \models M : A$ when $\langle \mathcal{S}(D), h, \rho\rangle : \Gamma \models M : A$ for all h and ρ.

(v) $(\lambda\mathcal{A})$ is *sound* with respect to $\mathcal{S}(D)$ if $\mathcal{S}(D) : \Gamma \models M : A$ whenever $\Gamma \vdash_{(\lambda\mathcal{A})} M : A$.

Approximating λ-models

We proceed now to show that interpretations of systems $(\lambda\mathcal{A})$ do exist. In order to do this, we build concrete type algebras of the form $\mathcal{S}(D)$, where D is a λ-model of a special – yet quite general – form. In particular, we shall assume throughout that $D \in \mathbf{ALG}$: this is mainly assumed for uniformity with the investigation of intersection types in Part III. Constructions of type interpretations related to that described below, with similar properties, can be carried out for complete partial orders too (see, e.g., **?**). Also, we need to assume that D has approximation properties for its elements that make it look similar to a solution of a recursive domain equation obtained as an inverse limit of a chain of embedding-projection pairs. This amounts to assume that D can be endowed with what we shall call a *notion of approximation*: this will be defined now in the general case where $\langle D, F, G\rangle$ is a λ-structure (see Definition 17.2.10).

11.1.3. Definition (Notion of approximation). Given a λ-structure $\langle D, F, G\rangle$, where $D = \langle D, \sqsubseteq\rangle$ is an algebraic lattice, a family of continuous functions $\{(\cdot)_n : D \to D\}_{n\in\omega}$ is a *notion of approximation* for D provided it satisfies the following conditions:

(i) For all $d \in D$, and all $n, m \in \omega$:

$$\bot_0 = \bot \tag{11.1}$$

$$n \leq m \Rightarrow d_n \sqsubseteq d_m \tag{11.2}$$

$$(d_n)_m = d_{\min(n,m)} = (d_m)_n \tag{11.3}$$

$$d = \bigsqcup_{n \in \omega} d_n. \tag{11.4}$$

(ii) For all $d, e \in D$ and $n \in \omega$:

$$d_0 \cdot e = d_0 = (d \cdot \bot)_0 \tag{11.5}$$

$$d_{n+1} \cdot e_n = d_{n+1} \cdot e = (d \cdot e_n)_n \tag{11.6}$$

$$d \cdot e = \bigsqcup_{n \in \omega} (d_{n+1} \cdot e_n). \tag{11.7}$$

The conditions of Definition 11.1.3 are satisfied by the D_∞ λ-models built by the classical construction of Scott **?**, although some of these may fail for the same contruction as modified by Park (see **?**). Furthermore, they also apply to models of the λ-calculus not explicitly obtained by means of an inverse limit construction, like Engeler's models D_A **?** or the filter λ-model $\mathcal{F}$ introduced in **?**.

Domains

We shall use a very general notion of domain, namely a partially ordered set with a least element and which enjoys some minimal completeness properties. The constructions on such structures can then be specialized to several (full) subcategories, in particular the category **ALG** of algebraic lattices.

11.1.4. Definition. A partial order $\langle D, \sqsubseteq \rangle$ with a least element $\bot_D$ is an *ω-complete partial order* (ω-cpo) if every increasing ω-chain $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \ldots \sqsubseteq d_n \sqsubseteq d_{n+1} \ldots$ of elements of D has a least upper bound $\bigsqcup_n d_n$.

A suitable notion of computable function between ω-cpo's is the following:

11.1.5. Definition (Continuous and strict functions). (i) If D, E are ω-cpo's, a *continuous function* $f : D \longrightarrow E$ is a monotonic function such that for every ω-chain $\langle d_n \rangle_{n \in \omega}$ in D

$$f(\bigsqcup_n d_n) = \bigsqcup_{n \in \omega} f(d_n).$$

The ω-cpo of continuous functions from D to E will be denoted by $[D \to E]$. (ii) A continuous function $f : D \longrightarrow E$ is *strict* if $f(\bot_D) = \bot_E$.

The category whose objects are the ω-cpo's and whose morphisms are the continuous functions between them will be denoted by ω-**CPO**.

The following is probably the most well-known (and useful) property of continuous endofunctions of an ω-cpo:

11.1.6. THEOREM. *If $\langle D, \sqsubseteq\rangle$ is an ω-cpo and $f : D \to D$ is a continuous function, then there is a continuous functional*

$$\mathbf{fix} \in [[D \to D] \to D]$$

such that, for every continuous $f : D \to D$,

$$f(\mathbf{fix}(f)) = \mathbf{fix}(f). \tag{11.8}$$

PROOF. Take **fix** to be the function which assigns to $f : D \to D$ the element

$$\bigsqcup_{n \in \omega} f^{(n)}(\bot_D).$$

The verification that equation 11.8 holds is left as an exercise. ■

The models D_∞

We consider first algebraic lattices D that solve domain equations of the form

$$D \cong [D \to D], \tag{11.9}$$

by means of the classical construction of Scott (see, e.g., **?**).

11.1.7. DEFINITION (Embedding-projection pairs). A pair of continuous functions

$$\langle e : D \longrightarrow E, p : E \longrightarrow D\rangle$$

such that $p \circ e = 1_D$ and $e \circ p \sqsubseteq 1_E$ is called an *embedding-projection pair*, p being the projection (of E onto D) and e the embedding (of D into E).

Both embeddings and projections are strict functions (Exercise 11.3.2).

The solution of equation (11.9) is obtained as the inverse limit of a sequence:

$$D_0 \xleftarrow{j_0} D_1 \xleftarrow{j_1} D_2 \xleftarrow{j_2} \cdots \xleftarrow{j_{n-1}} D_n \xleftarrow{j_n} D_{n+1} \xleftarrow{j_{n+2}} \cdots \tag{11.10}$$

where D_0 is an arbitrary algebraic lattice, $\langle i_0, j_0\rangle$ is an embedding-projection pair, and where:

$$\begin{aligned} D_{n+1} &= [D_n \to D_n]. \\ i_{n+1}(d) &= i_n \circ d \circ j_n \\ j_{n+1}(e) &= j_n \circ e \circ i_n. \end{aligned}$$

The inverse limit of (11.10) is defined explicitly as

$$D_\infty = \{\langle d^{(n)}\rangle_{n \in \omega} \mid \forall n \in \omega . d^{(n)} \in D_n \text{ and } j_{n+1}(d^{(n+1)}) = d^{(n)}\}. \tag{11.11}$$

and comes equipped with embeddings $i_{n\infty} : D_n \to D_\infty$ and projections $j_{\infty n} : D_\infty \to D_n$, for all $n \in \omega$. This turns out to be also the direct limit of the sequence of embeddings $\langle D_n, i_n\rangle$ (see **?**). This is enough to provide a notion of approximation for D_∞:

11.1.8. Proposition. *The family of mappings*

$$i_{n\infty} \circ j_{\infty n} : D_\infty \to D_\infty$$

for $n \in \omega$, defines a notion of approximation for D_∞.

Proof. This is left as Exercise 11.3.3. ■

In Exercises 11.3.4 and 11.3.5 we sketch proofs that the standard filter model of **?**, discussed in detail in Chapter 14, and the models built in **?** can be given a notion of approximation satisfying all the conditions of Definition 11.1.3.

Interpreting μ types

For the formulations à la Curry of the systems $(\lambda\mu)$ and $(\lambda\mu^*)$ we have to find a suitable class of subsets of a λ-model D closed under a construction $X \Rightarrow Y$ for any pair X, Y of such subsets, with the property that $\mathbb{T}_\mu / =_\mu$ and $\mathbb{T}_\mu / =^*$ can be mapped homomorphically to the type algebra so obtained.

One typical candidate for such a class of subsets is the collection of *ideals* of D, i.e., the non empty closed subsets of D with respect to the Scott topology (see **?**, **?**, **?**). Equivalently, these can be described as the non-empty, downward closed subsets X of D such that $\bigsqcup \Delta \in X$ whenever $\Delta \subseteq X$ is directed. Here we shall use a slightly more general semantical notion of type, by relaxing the requirement of downward closure and assuming only that types be *uniform*: if d belongs to a type, then d_n belongs to that type for all $n \in \omega$. (The names of these properties come from **?**.)

11.1.9. Remark. The closure properties that we require for our semantical notion of type are motivated by the fact that we are working essentially in *continuous* λ-models. For example, in all such models (see **?**)

$$\bot_D = [\![(\lambda x.xx)(\lambda x.xx)]\!],$$

hence $\bot_D$ belongs to all types, which accounts for non-emptiness. On the other hand, the interpretation of the fixed point combinator $\mathsf{Y} = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ defines the mapping $\mathbf{fix} \in [[D \to D] \to D]$, and

$$\mathbf{fix}(f) = \bigsqcup_{n \in \omega} f^{(n)}(\bot). \tag{11.12}$$

Now, it was proved in Proposition 8.1.12 that Y has type $(A \to A) \to A$, and this motivates completeness. Concerning uniformity, observe that if X is an ideal of D, the set $X_n =_{\text{def}} \{d_n \mid d \in X\}$ is not, in general, an ideal of D. For example, in D_∞ as described in 11.11, D_0 is not downward closed as a subset of D.

The construction described below can be performed, more generally, using n-ary complete and uniform relations over D as interpretations of types. This applies in particular to the interpretation of types as (complete and uniform) *partial equivalence relations (PER's)* over D, that can be exploited in the construction of an extensional model for the versions à la Church of both $(\lambda\mu)$ and $(\lambda\mu^*)$ (this is the content of Exercise 11.3.13).

11.1.10. Definition (Complete subsets). Given a reflexive cpo with a notion of approximation D, a subset $X \subseteq D$ is *complete* iff:

- $\bot_D \in X$,
- if $\{d^{(j)} \mid j \in \omega\}$ is an increasing chain in X, then

$$\bigsqcup_{j \in \omega} d^{(j)} \in X$$

The subsets that interpret types are required to respect the notion of approximation, in the following sense:

11.1.11. Definition (Uniformity). Given a cpo with a notion of approximation D, let ∇ be a complete subset of D_0. A subset $X \subseteq D$ is *uniform* if the following properties hold where, for $n \in \omega$, $X_n =_{\text{def}} \{d_n \mid d \in X\}$:

- $X_0 = \nabla$,
- $X_n \subseteq X$ for all $n \in \omega$.

Let $\mathcal{CU}(D)$ denote the set of *complete and uniform subsets* over D with *base* ∇. The following Proposition summarizes the main properties of $\mathcal{CU}(D)$.

11.1.12. Proposition (Basic properties of complete and uniform subsets). *For any $X \in \mathcal{CU}(D)$ and any $n \in \omega$:*

1. *$X_n \in \mathcal{CU}(D)$;*
2. *$X = Y$ if and only if, for all $n \in \omega$, $X_n = Y_n$;*
3. *$X_n = (X_{n+1})_n$;*
4. *Let $X^{(i)}$, for $i \in \omega$, be a sequence of complete and uniform subsets such that, for all $n \in \omega$:*

$$X^{(n)} = (X^{(n+1)})_n. \tag{11.13}$$

Then:

- *For all $k \geq n$ $X^{(n)} \subseteq X^{(k)}$;*
- *For all $k \leq n$ $(X^{(n)})_k = X^{(k)}$;*
- *Define $X^{(\infty)}$ as $\{d \mid d_n \in X^{(n)}\}$. Then $X^{(\infty)}$ is the unique complete and uniform subset such that, for all $n \in \omega$, $(X^{(\infty)})_n = X^{(n)}$.*

Proof. (1) X_n is uniform by Definition 11.1.3(11.3), and complete by continuity of the approximation mappings.
(2) Easy, using (2) and Definition 11.1.3(11.4).
(3) If $d_n \in X_n$ then, by 1 above, $d_n \in X$ and then $d_n \in X_{(n+1)}$, again from Definition 11.1.3(11.3). Conversely, $d \in (X_{(n+1)})_n$ implies $d \in X$ and $d_n = d$, so $d \in X_n$.
(4) The first point is proved by induction on $k - n$: if $k = n$ there is nothing to show; otherwise $k > n$ and we have $X^{(n)} \subseteq X^{(n)} \subseteq X^{(k)}$ by the assumption

(11.13) and the induction hypothesis. The proof of the second point is similar, by induction on $n-k$: the basis is obvious; for the induction step, calculate

$$\begin{aligned}(X^{(n)})_k &= \left((X^{(n)})_{k+1}\right)_k \\ &= \left(X^{(k+1)}\right)_k && \text{(by induction hypothesis)} \\ &= X^{(k)}\end{aligned}$$

For the last point, we have to show that $X^{(\infty)}$ is complete and uniform: first of all, $(X^{(\infty)})_0 = \nabla$, because if $d \in \nabla$ then $d \in X^{(i)}$ for all $i \in \omega$ as each $X^{(i)}$ is uniform; but then $d_i = d \in X^{(i)}$ and therefore $\nabla \subseteq (X^{(\infty)})_0$. Conversely, observe that $(X^{(\infty)})_0 \subseteq X^{(0)} = \nabla$, by assumption (11.13).

For uniformity of $X^{(\infty)}$, let $d \in X^{(\infty)}$, then $d_n \in (X^{(n)})_n$ so for all $k \in \omega$ we have $(d_n)_k \in (X^{(n)})_k$ using the two preceding points, so $d_n \in X^{(\infty)}$.

For completeness, let $d^{(j)} \in X^{(\infty)}$ be a chain of elements: then, for any $n \in \omega$, $(d^{(j)})_n \in X^{(n)}$ and therefore also $\bigsqcup_j (d^{(j)})_n = \left(\bigsqcup_j d^{(j)}\right)_n \in X^{(n)}$ by continuity of the approximation mappings, which entails that $\bigsqcup_j d^{(j)} \in X^{(\infty)}$. It is easy to see that $X^{(n)} \subseteq (X^{(\infty)})_n$ for every n, so $X^{(n)} = (X^{(\infty)})_n$ because the reverse inclusion holds by definition of $X^{(\infty)}$. Finally, $X^{(\infty)}$ is unique, because if Y is another complete and uniform subset with the property that $Y_n = X^{(n)}$, then $Y_n = (X^{(\infty)})_n$ for all $n \in \omega$, and this yields $Y = X^{(\infty)}$. ■

The operation on the class of complete and uniform subsets that provides the interpretations of function types can now be defined.

11.1.13. Definition. For $X, Y \in \mathcal{CU}(D)$, define

$$(X \Rightarrow Y) =_{\text{def}} \{d \in D \mid \forall a \in D\,(a \in X \implies d \cdot a \in Y)\}$$

11.1.14. Proposition. *If $X, Y \in \mathcal{CU}(D)$ then $X \Rightarrow Y \in \mathcal{CU}(D)$.*

Proof. We first show that $\nabla = (X \Rightarrow Y)_0$. Assume $d \in \nabla$, and let $a \in X$. Then

$$\begin{aligned}d \cdot a &= d_0 \cdot a && (\text{as } \nabla \subseteq D_0) \\ &= d && \text{(by Definition 11.1.3(11.5))}\end{aligned}$$

therefore $d \cdot a \in \nabla = Y_0 \subseteq Y$ by uniformity. So $\nabla \subseteq X \Rightarrow Y$, and $\nabla \subseteq (X \Rightarrow Y)_0$ as $\nabla = \nabla_0$. Conversely, let $d_0 \in (X \Rightarrow Y)_0$. Then

$$d_0 = (d \cdot \bot_D)_0 \qquad \text{(by Definition 11.1.3(11.5))}$$

but $\bot_D \in \nabla = X_0 \subseteq X$, hence $d \cdot \bot_D \in Y$, because $d \in X \Rightarrow Y$, so $(d \cdot \bot_D)_0 \in Y_0 = \nabla$ and finally $d_0 \in \nabla$.

For uniformity of $X \Rightarrow Y$, assume that $d \in X \Rightarrow Y$, and consider d_n for $n > 0$ (we already know that $(X \Rightarrow Y)_0 = \nabla \subseteq X \Rightarrow Y$). For any $a \in X$

$$\begin{aligned}d_n \cdot a &= d_n \cdot a_{n-1} \\ &= (d \cdot a_{n-1})_{n-1} && \text{(by Definition 11.1.3(11.6))}\end{aligned}$$

but $a_{n-1} \in X$ by uniformity of X and therefore $d \cdot a_{n-1} \in Y$, so also $(d \cdot a_{n-1})_{n-1}$ by uniformity of Y, and finally $d_n \cdot a \in X \Rightarrow Y$.

For completeness, assume that we have an increasing chain of elements $\{d^{(j)} \mid j \in \omega\}$ in $X \Rightarrow Y$ and $a \in X$. Then

$$\left(\bigsqcup_{j \in \omega} d^{(j)} \right) \cdot a = \bigsqcup_{j \in \omega} (d^{(j)} \cdot a)$$

by continuity of application, and therefore

$$\bigsqcup_{j \in \omega} (d^{(j)} \cdot a),$$

by completeness of Y, hence $\bigsqcup_{j \in \omega} d^{(j)} \in X \Rightarrow Y$.

The following property of the interpretation of function types is the key to the whole construction of the interpretation of recursive types:

11.1.15. PROPOSITION. *For any $X, Y \in \mathcal{CU}(D)$ and $n \in \omega$,*

$$(X \Rightarrow Y)_{n+1} = (X_n \Rightarrow Y_n)_{n+1}.$$

PROOF. Assume that $d_{n+1} \in (X \Rightarrow Y)_{n+1} \subseteq X \Rightarrow Y$ and that $a_n \in X_n \subseteq X$. Then $d_{n+1} \cdot a_n \in Y$. But

$$d_{n+1} \cdot a_n = (d_{n+1} \cdot a_n)_n \in Y_n$$

by Definition 11.1.3(11.6), so $d_{n+1} \in X_n \Rightarrow Y_n$ and this entails $d_{n+1} \in (X_n \Rightarrow Y_n)_{n+1}$. Conversely, let $d_{n+1} \in (X_n \Rightarrow Y_n)_{n+1}$, and assume that $a \in X$. Then $a_n \in X_n$. Observe now that

$$d_{n+1} \cdot a = d_{n+1} \cdot a_n \in Y_n \subseteq Y$$

using again Definition 11.1.3(11.6) and uniformity of Y, so $d_{n+1} \in X \Rightarrow Y$ and finally $d_{n+1} \in (X \Rightarrow Y)_{n+1}$. ■

Finally, we can define the type algebra used throughout this section:

11.1.16. DEFINITION. $\mathcal{S}(D) =_{\text{def}} \langle \mathcal{CU}(D), \Rightarrow \rangle$

As an example, let us see how the theory developed so far allows to interpret a type $T = T \to T$ as a complete and uniform subset $\Xi = \Xi \Rightarrow \Xi$. The latter is built in denumerably many steps

$$\Xi^{(0)}, \Xi^{(1)}, \Xi^{(2)}, \ldots$$

The 0-th stage is $\Xi^{(0)} = \nabla$. Then, whatever Ξ will be eventually, $\Xi_0 = \Xi^{(0)}$. Later stages are defined by the recurrence

$$\Xi^{(n+1)} = (\Xi^{(n)} \Rightarrow \Xi^{(n)})_{n+1}.$$

If we can show that for all $n \in \omega$

$$\left(\Xi^{(n+1)}\right)_n = \Xi^{(n)}, \tag{11.14}$$

then we can exploit Proposition 11.1.12(4) and take $\Xi = \Xi^{(\infty)}$, so that $\Xi_n = \Xi^{(n)}$ for all $n \in \omega$. The proof of (11.14) is therefore the core of the technique, and appeals in an essential way to Proposition 11.1.15. The fact that $\Xi = \Xi \Rightarrow \Xi$ is then a direct consequence of Proposition 11.1.12(2). Of course, this process must be carried out, in parallel, for *all* type expressions, by defining a whole family of approximate interpretations of types. We shall now show how to do this in the case of μ-types.

Approximate interpretations of μ-types

In order to state the definition of the approximate interpretation of types it is convenient to introduce an auxiliary notation:

NOTATION. For $X, Y \in \mathcal{CU}(D)$ and $n \in \omega$, let $X \Rightarrow^{n+1} Y$ denote $(X \Rightarrow Y)_{n+1}$.

It follows immediately from Proposition 11.1.15 that $X \Rightarrow^{n+1} Y = X_n \Rightarrow^{n+1} Y_n$. An easy argument, using Definition 11.1.3(11.6), shows that in addition $X \Rightarrow^{n+1} Y = \{d \in D_{n+1} \mid \forall a \in D\,(a \in X \implies d \cdot a \in Y)\}$.

11.1.17. DEFINITION (Approximate Interpretations of Types). For all $n \in \omega$, any type $A \in \mathbb{T}_\mu$ and any type environment $\eta : V \to \mathcal{CU}(D)$, define $\mathcal{I}^n[\![A]\!]_\eta$ (the *n-th approximation of the interpretation of A in the environment η*) as follows, by induction on n and the complexity of the type A:

- $\mathcal{I}^0[\![A]\!]_\eta = \nabla$;
- $\mathcal{I}^{n+1}[\![\alpha]\!]_\eta = (\eta(\alpha))_{n+1}$;
- $\mathcal{I}^{n+1}[\![A_1 \to A_2]\!]_\eta = \mathcal{I}^n[\![A_1]\!]_\eta \Rightarrow^{n+1} \mathcal{I}^n[\![A_2]\!]_\eta$;
- $\mathcal{I}^{n+1}[\![\mu\alpha.A_1]\!]_\eta = \mathcal{I}^{n+1}[\![A_1]\!]_{\eta[\alpha \mapsto \mathcal{I}^n[\![\mu\alpha.A_1]\!]_\eta]}$.

By a simple inductive argument (on n and then on the structure of the type) one can see that each $\mathcal{I}^n[\![A]\!]_\eta$ is a complete and uniform subset of D. We shall make frequent use below of the following properties, whose easy inductive proof is left as an exercise:

11.1.18. LEMMA. *For any $A \in \mathbb{T}_\mu$, $\alpha \in V$, any environment η:*

(i) $\mathcal{I}^n[\![A]\!]_\eta = \mathcal{I}^n[\![A]\!]_{(\eta \restriction n)}$, *where* $(\eta \restriction n)(\alpha) =_{\text{def}} (\eta(\alpha))_n$.

(ii) If A is contractive in α then

$$\mathcal{I}^{n+1}[\![A]\!]_\eta = \mathcal{I}^{n+1}[\![A]\!]_{\eta[\alpha \mapsto (\eta(\alpha))_n]}$$

for all $n \in \omega$. ■

For circular types, recursion has a trivial interpretation:

11.1.19. LEMMA. *If A is not contractive in α, then*

$$\mathcal{I}^n[\![\mu\alpha.A]\!]_\eta = \nabla$$

for any $n \in \omega$ and any environment η. ■

11.1.20. LEMMA. *For any $n \in \omega$, all types A and any type environment η:*

$$\mathcal{I}^n[\![A]\!]_\eta = \left(\mathcal{I}^{n+1}[\![A]\!]_\eta\right)_n .$$

PROOF. For all $n \in \omega$, A and η, the statement is clearly equivalento to the conjunction of the following two:

(a) $\mathcal{I}^n[\![A]\!]_\eta = \mathcal{I}^{n+1}[\![A]\!]_\eta$;

(b) $d \in \mathcal{I}^{n+1}[\![A]\!]_\eta$ implies that $d_n \in \mathcal{I}^n[\![A]\!]_\eta$,

so we show (a) and (b) by induction on n. The basis is obvious, and the induction step is proved by induction on the complexity of the type A.

- $A \equiv \alpha$, a type variable:
$\mathcal{I}^n[\![\alpha]\!]_\eta = (\eta(\alpha))_n$, and the result follows from Proposition 11.1.12(3).

- $A \equiv A_1 \to A_2$:
We have to show first that

$$\mathcal{I}^{n-1}[\![A_1]\!]_\eta \Rightarrow^n \mathcal{I}^{n-1}[\![A_2]\!]_\eta \subseteq \mathcal{I}^n[\![A_1]\!]_\eta \Rightarrow^{n+1} \mathcal{I}^n[\![A_2]\!]_\eta$$

so assume that $d \in \mathcal{I}^{n-1}[\![A_1]\!]_\eta \Rightarrow^n \mathcal{I}^{n-1}[\![A_2]\!]_\eta$ and that $a \in \mathcal{I}^n[\![A_1]\!]_\eta$. By induction hypothesis (b) on n, $a_{n-1} \in \mathcal{I}^{n-1}[\![A_1]\!]_\eta$, hence

$$d \cdot a_{n-1} = d \cdot a \in \mathcal{I}^{n-1}[\![A_2]\!]_\eta \subseteq \mathcal{I}^n[\![A_2]\!]_\eta$$

using the induction hypothesis (a), and equation (11.6) of Definition 11.1.3. If $d \in \mathcal{I}^n[\![A_1]\!]_\eta \Rightarrow^{n+1} \mathcal{I}^n[\![A_2]\!]_\eta$ and $a \in \mathcal{I}^{n-1}[\![A_1]\!]_\eta$, $a \in \mathcal{I}^n[\![A_1]\!]_\eta$ by induction hypothesis (a), so $d \cdot a \in \mathcal{I}^n[\![A_2]\!]_\eta$ and by induction hypothesis (b) we get $(d \cdot a)_{n-1} \in \mathcal{I}^{n-1}[\![A_2]\!]_\eta$. The result follows by observing that, using again equation (11.6) of Definition 11.1.3, $(d \cdot a)_{n-1} = d_n \cdot a$, therefore

$$d_n \in \mathcal{I}^{n-1}[\![A_1]\!]_\eta \Rightarrow^n \mathcal{I}^{n-1}[\![A_2]\!]_\eta .$$

- $A \equiv \mu\alpha.A_1$:
If A_1 is not contractive in α, then the property is trivially true by Lemma 11.1.19. So, assume that A_1 is contractive in α:

$$\begin{aligned}
\mathcal{I}^n[\![\mu\alpha.A_1]\!]_\eta &= \mathcal{I}^n[\![A_1]\!]_{\eta[\alpha \mapsto \mathcal{I}^{n-1}[\![\mu\alpha.A_1]\!]_\eta]} \\
&= \mathcal{I}^n[\![A_1]\!]_{\eta[\alpha \mapsto (\mathcal{I}^n[\![\mu\alpha.A_1]\!]_\eta)_{n-1}]} && \text{by induction hypothesis on } n \\
&= \mathcal{I}^n[\![A_1]\!]_{\eta[\alpha \mapsto \mathcal{I}^n[\![\mu\alpha.A_1]\!]_\eta]} && \text{by Lemma 11.1.18(ii)} \\
&\subseteq \mathcal{I}^{n+1}[\![A_1]\!]_{\eta[\alpha \mapsto \mathcal{I}^n[\![\mu\alpha.A_1]\!]_\eta]} \\
& && \text{by induction hypothesis on } A \\
&= \mathcal{I}^{n+1}[\![\mu\alpha.A_1]\!]_\eta .
\end{aligned}$$

Now, let $d \in \mathcal{I}^{n+1}[\![\mu\alpha.A_1]\!]_\eta = \mathcal{I}^{n+1}[\![A_1]\!]_{\eta[\alpha \mapsto \mathcal{I}^n[\![\mu\alpha.A_1]\!]_\eta]}$. This implies that $d_n \in \mathcal{I}^n[\![A_1]\!]_{\eta[\alpha \mapsto \mathcal{I}^n[\![\mu\alpha.A_1]\!]_\eta]}$ by induction hypothesis on the complexity of the type, and therefore $d_n \in \mathcal{I}^n[\![A_1]\!]_{\eta[\alpha \mapsto (\mathcal{I}^n[\![\mu\alpha.A_1]\!]_\eta)_{n-1}]}$

by Lemma 11.1.18, as we assumed that A_1 is contractive in α. But $(\mathcal{I}^n[\![\mu\alpha.A_1]\!]_\eta)_{n-1} = \mathcal{I}^{n-1}[\![\mu\alpha.A_1]\!]_\eta$ by induction hypothesis on n and we can conclude that

$$d_n \in \mathcal{I}^n[\![A_1]\!]_{\eta[\alpha\mapsto\mathcal{I}^{n-1}[\![\mu\alpha.A_1]\!]_\eta]} = \mathcal{I}^n[\![\mu\alpha.A_1]\!]_\eta.$$

■

The interpretation $\mathcal{I}[\![A]\!]_\eta$ of a type A can now be defined by glueing its approximate interpretations $\mathcal{I}^n[\![A]\!]_\eta$, as described in Proposition 11.1.12(4):

11.1.21. Definition (Type Interpretations). For every type A and type environment η,

$$\mathcal{I}[\![A]\!]_\eta =_{\text{def}} \mathcal{I}^\infty[\![A]\!]_\eta$$

Observe that the interpretation of types depends on the choice of the base ∇. For example, when $D = D_\infty$ is defined as in equation (11.11), as the inverse limit of a sequence (11.10) of algebraic lattices, where D_0 consists of the the two points $\bot, \top$, if $\nabla = \{\bot\}$ then $\top \notin \mathcal{I}[\![\mu t.t\to t]\!]_\eta$ whereas $\mathcal{I}[\![\mu t.t\to t]\!]_\eta = D$ when $\nabla = \{\bot, \top\}$.

11.1.22. Proposition. *For any type A and environment η,*

$$\Big(\mathcal{I}[\![A]\!]_\eta\Big)_n = \mathcal{I}^n[\![A]\!]_\eta,$$

for all $n \in \omega$.

Proof. From Proposition 11.1.12(4). ■

In order to study the properties of the interpretation of types, it is useful to know more about the relations between the interpretation of recursive types and the fixed points of a natural class of mappings over $\mathcal{CU}(D)$.

11.1.23. Definition (Ideal functions). A p-ary function over $\mathcal{CU}(D)$

$$f : \mathcal{CU}(D) \times \ldots \times \mathcal{CU}(D) \to \mathcal{CU}(D)$$

is *ideal* if, for all $X^{(1)}, \ldots, X^{(p)}$, and $n \in \omega$:

$$\Big(f(X^{(1)}, \ldots, X^{(p)})\Big)_{n+1} = \Big(f((X^{(1)})_n, \ldots, (X^{(p)})_n)\Big)_{n+1} \tag{11.15}$$

11.1.24. Lemma (Parameterized fixed-points of ideal functions). *Assume that the $p+1$-ary function f over $\mathcal{CU}(D)$ is ideal. Then, for all $X, \vec{X} = X^{(1)}, \ldots, X^{(p)}$, there is a unique element $f^\dagger(\vec{X})$ of $\mathcal{CU}(D)$ such that:*

$$f^\dagger(\vec{X}) = f(f^\dagger(\vec{X}), \vec{X}).$$

PROOF. Define inductively the sequence:

$$\begin{cases} \Phi^{(0)} & = \nabla \\ \Phi^{(n+1)} & = \left(f(\Phi^{(n)}, \vec{X})\right)_{n+1} \end{cases}$$

Then, for all $n \in \omega$, $\Phi^{(n)} = \left(\Phi^{(n+1)}\right)_n$. In fact, $\Phi^{(0)} = \nabla = \left(\Phi^{(1)}\right)_0$. Assuming that $\Phi^{(n-1)} = \left(\Phi^{(n)}\right)_{n-1}$ for $n > 0$, we have

$$\begin{aligned} \left(\Phi^{(n+1)}\right)_n &= \left((f(\Phi^{(n)}, \vec{X}))_{n+1}\right)_n && \text{(by definition)} \\ &= \left(f(\Phi^{(n)}, \vec{X})\right)_n && \text{(by uniformity)} \\ &= \left(f\left((\Phi^{(n)})_{n-1}, \vec{X}\right)\right)_n && (f \text{ is ideal}) \\ &= \left(f(\Phi^{(n-1)}, \vec{X})\right)_n && \text{(induction hypothesis)} \\ &= \Phi^{(n)} && \text{(by definition)} \end{aligned}$$

Using Proposition 11.1.12(4), define $f^\dagger(\vec{X})$ as $\Phi^{(\infty)}$, and observe that

$$\begin{aligned} \left(f(\Phi^{(\infty)}, \vec{X})\right)_n &= \left(f((\Phi^{(\infty)})_{n-1}, \vec{X})\right)_n && \text{(as } f \text{ is ideal)} \\ &= \left(f(\Phi^{(n-1)}, \vec{X})\right)_n \\ &= \Phi^{(n)} && \text{(by definition)} \end{aligned}$$

hence $f(f^\dagger(\vec{X}), \vec{X}) = f(\Phi^{(\infty)}, \vec{X}) = \Phi^{(\infty)} = f^\dagger(\vec{X})$. Uniqueness of $f^\dagger$ follows by a straightforward induction, using Proposition 11.1.12(1). ■

11.1.25. LEMMA. *For any type A and environment η:*

$$\mathcal{I}[\![\mu\alpha.A]\!]_\eta = \mathcal{I}[\![A]\!]_{\eta[\alpha \mapsto \mathcal{I}[\![\mu\alpha.A]\!]_\eta]}$$

PROOF. If A is not contractive in α, then the property is trivially true by Lemma 11.1.19. So, assume that A is contractive in α and observe that the mapping $X \longmapsto \mathcal{I}[\![A]\!]_{\eta[\alpha \mapsto X]} : \mathcal{CU}(D) \to \mathcal{CU}(D)$ is ideal by Lemma 11.1.18, then apply Lemma 11.1.24 (where $p = 0$). ■

11.1.26. LEMMA. *For any pair of types A, B, any type variable α and any environment η,*

$$\mathcal{I}[\![A[\alpha := B]]\!]_\eta = \mathcal{I}[\![A]\!]_{\eta[\alpha \mapsto \mathcal{I}[\![B]\!]_\eta]}.$$

11.1.27. THEOREM (Properties of Type Interpretations). *The following conditions are satisfied, for any type environment η and all types A, B:*

1. $\mathcal{I}[\![\alpha]\!]_\eta = \eta(\alpha)$
2. $\mathcal{I}[\![A \to B]\!]_\eta = \mathcal{I}[\![A]\!]_\eta \Rightarrow \mathcal{I}[\![B]\!]_\eta$
3. $\mathcal{I}[\![\mu\alpha.A]\!]_\eta = \mathcal{I}[\![A[\alpha := \mu\alpha.A]]\!]_\eta$

PROOF. (1) $d \in \mathcal{I}[\![\alpha]\!]_\eta$ iff, for all $n \in \omega$, $d_n \in \mathcal{I}^n[\![\alpha]\!]_\eta = (\eta(\alpha))_n$ iff $d \in \eta(\alpha)$.
(2) Observe that for all $n \in \omega$:

$$\begin{aligned}
(\mathcal{I}[\![A \to B]\!]_\eta)_{n+1} &= \mathcal{I}^{n+1}[\![A \to B]\!]_\eta && \text{by Proposition 11.1.22}\\
&= \mathcal{I}^n[\![A]\!]_\eta \Rightarrow^{n+1} \mathcal{I}^n[\![B]\!]_\eta\\
&= (\mathcal{I}[\![A]\!]_\eta)_n \Rightarrow^{n+1} (\mathcal{I}[\![B]\!]_\eta)_n\\
&= (\mathcal{I}[\![A]\!]_\eta \Rightarrow \mathcal{I}[\![B]\!]_\eta)_{n+1} && \text{by Proposition 11.1.15}
\end{aligned}$$

and the result follows by induction from Proposition 11.1.12(2), observing that $(\mathcal{I}[\![A \to B]\!]_\eta)_0 = \nabla = (\mathcal{I}[\![A]\!]_\eta \Rightarrow \mathcal{I}[\![B]\!]_\eta)_0$.
(3) From Lemmas 11.1.25 and 11.1.26. ■

Soundness

Theorem 11.1.27 implies that $\mathcal{I}[\![-]\!]_\eta$, for any type environment η, is a type algebra homomorphism from $\mathbb{T}_\mu/=_\mu$ to $\mathcal{S}(D)$ as defined in 11.1.16. We have immediately the following corollary:

11.1.28. PROPOSITION (Soundness of $\vdash_{\lambda\mu}$). *Let* $\Gamma \vdash_{\lambda\mu} M : A$. *Then* $\mathcal{S}(D) : \Gamma \models M : A$.

PROOF. By induction on the length of the derivation of the typing judgment $\Gamma \vdash_{\lambda\mu} M : A$, using Lemma 11.1.25 and Theorem 11.1.27. For example, if the last rule applied is

$$\frac{\Gamma, x : A \vdash_{\lambda\mu} M : B}{\Gamma \vdash_{\lambda\mu} \lambda x.M : A \to B}$$

assume that $\mathcal{S}(D), \mathcal{I}[\![-]\!]_\eta, \rho \models \Gamma$ and that $d \in \mathcal{I}[\![A]\!]_\eta$. By induction hypothesis we have

$$\mathcal{S}(D), \mathcal{I}[\![-]\!]_\eta, \rho[x \mapsto d] \models M : B$$

and, as $[\![M]\!]_{\rho[x \mapsto d]} = [\![\lambda x.M]\!]_\rho$ we conclude that $[\![\lambda x.M]\!]_\rho \in \mathcal{I}[\![A]\!]_\eta \to \mathcal{I}[\![B]\!]_\eta = \mathcal{I}[\![A \to B]\!]_\eta$. ■

We now proceed to show that the type interpretation defined in Theorem 11.1.27 even induces a type algebra homomorphism from $\mathbb{T}_\mu/=^*$ to $\mathcal{S}(D)$. To this end, we introduce a notion of approximate interpretation for the regular trees which result from unfolding infinitely often types in $\mathbb{T}_\mu$. This interpretation is of interest in its own, and is indeed the notion of interpretation which is taken as basic in ?.

11.1.29. DEFINITION (Approximate interpretations of regular trees). For all $n \in \omega$, any type $\alpha \in \mathsf{Tr}_{\mathrm{R}}$ and any type environment $\eta : V \to \mathcal{CU}(D)$, define $\mathcal{T}^n[\![\alpha]\!]_\eta$ by induction on n:

- $\mathcal{T}^0[\![\alpha]\!]_\eta = D_0$;
- $\mathcal{T}^{n+1}[\![\alpha]\!]_\eta = (\eta(\alpha))_{n+1}$;

- $\mathcal{T}^{n+1}[\![\bullet]\!]_\eta = D_0$;
- $\mathcal{T}^{n+1}[\![\alpha_1 \to \alpha_2]\!]_\eta = \mathcal{T}^n[\![\alpha_1]\!]_\eta \Rightarrow^{n+1} \mathcal{T}^n[\![\alpha_2]\!]_\eta$.

11.1.30. LEMMA. *For all types $A \in \mathbb{T}_\mu$, all type environments η and all $n \in \omega$:*

$$\mathcal{I}^n[\![A]\!]_\eta = \mathcal{T}^n[\![A^*]\!]_\eta,$$

where $(\cdot)^* : \mathbb{T}_\mu \to \mathbb{T}_{\mathrm{R}}$.

PROOF. By induction on n. The basis is obvious, while the induction step is proved by induction on $A \in \mathbb{T}_\mu$. Also in this case the basis is clear. For the induction step,

- let $A \equiv A_1 \to A_2$: then

$$\begin{aligned}
\mathcal{T}^{n+1}[\![(A_1 \to A_2)^*]\!]_\eta &= \mathcal{T}^{n+1}[\![(A_1)^* \Rightarrow (A_2)^*]\!]_\eta \\
&= \mathcal{T}^n[\![(A_1)^*]\!]_\eta \Rightarrow^{n+1} \mathcal{T}^n[\![(A_2)^*]\!]_\eta \\
&= \mathcal{I}^n[\![A_1]\!]_\eta \Rightarrow^{n+1} \mathcal{I}^n[\![A_2]\!]_\eta \\
&= \mathcal{I}^{n+1}[\![A_1 \to A_2]\!]_\eta;
\end{aligned}$$

- let $A \equiv \mu\alpha.A_1$: then

$$\mathcal{T}^{n+1}[\![(\mu\alpha.A_1)^*]\!]_\eta = \mathcal{T}^{n+1}[\![(A_1)^*[\alpha := (\mu\alpha.A_1)^*]]\!]_\eta$$

and we can prove by cases on the possible forms of $(A_1)^*$ that

$$\mathcal{T}^{n+1}[\![(\mu\alpha.A_1)^*]\!]_\eta = \mathcal{T}^{n+1}[\![(A_1)^*]\!]_{\eta[\alpha \mapsto \mathcal{T}^n[\![(\mu\alpha.A_1)^*]\!]_\eta}.$$

Then we have

$$\begin{aligned}
\mathcal{T}^{n+1}[\![(\mu\alpha.A_1)^*]\!]_\eta &= \mathcal{T}^{n+1}[\![(A_1)^*]\!]_{\eta[\alpha \mapsto \mathcal{T}^n[\![(\mu\alpha.A_1)^*]\!]_\eta} \\
&= \mathcal{I}^{n+1}[\![A_1]\!]_{\eta[\alpha \mapsto \mathcal{I}^n[\![\mu\alpha.A_1]\!]_\eta} \\
&= \mathcal{I}^{n+1}[\![\mu\alpha.A_1]\!]_\eta
\end{aligned}$$

using the induction hypotheses on n and A_1. ■

11.1.31. PROPOSITION. *If $A =^* B$ for $A, B \in \mathbb{T}_\mu$, then $\mathcal{I}[\![A]\!]_\eta = \mathcal{I}[\![B]\!]_\eta$ for all type environments η.*

PROOF. For all $A \in \mathbb{T}_\mu$, $n \in \omega$ and all type environments η we have:

$$\begin{aligned}
\mathcal{I}^n[\![A]\!]_\eta &= \mathcal{T}^n[\![A^*]\!]_\eta \\
&= \mathcal{T}^n[\![B^*]\!]_\eta \\
&= \mathcal{I}^n[\![B]\!]_\eta
\end{aligned}$$

by Lemma 11.1.30 and the fact that $A^* = B^*$. Then the statement follows from the fact that for $X, Y \in \mathcal{CU}(D)$, if $X_n = Y_n$ for all $n \in \omega$, then $X = Y$. ■

As an immediate consequence we have the soundness of rule (equiv) in Definition 8.1.11 and therefore, by a straightforward inductive argument, also the soundness of the typing rules of the system à la Curry with strong equality of types:

11.1.32. COROLLARY. *[Soundness of $\vdash_{\lambda\mu^*}$] Let $\Gamma \vdash_{\lambda\mu^*} M : A$. Then $\mathcal{S}(D) : \Gamma \models M : A$.*

As an application of Corollary 11.1.32, we show that types of a special form are inhabited only by unsolvable terms, in the sense of **?**.

We say that a type is *tail μ-free* if it is equivalent to a type of the shape

$$A_1 \to \cdots \to A_n \to \alpha \quad (n \geq 0) \tag{11.16}$$

For instance $\mu\alpha.t \to \alpha$ is tail μ-free while $\mu\alpha.\alpha \to t$ is not. Since type interpretation is preserved by equivalence we will assume that all tail μ-free types are of the form (11.16). We also say that a basis is tail μ-free if all types in its range are tail μ-free.

11.1.33. FACT. Let D be a λ-model with a notion of approximation and $A = A_1 \to \cdots \to A_n \to \alpha$ be a tail μ-free type. Then $\mathcal{I}[\![A]\!]_{\eta[\alpha \mapsto D]} = D$.

PROOF. By induction on n. The basis is immediate. If $d \in D$, then for any $a \in \mathcal{I}[\![A_1]\!]_{\eta[\alpha \mapsto D]}$ we have $d \cdot a \in D \subseteq \mathcal{I}[\![A_2 \to \cdots \to A_n \to \alpha]\!]_{\eta[\alpha \mapsto D]}$ by induction hypothesis, so $D \subseteq \mathcal{I}[\![A_1 \to \cdots \to A_n \to \alpha]\!]_{\eta[\alpha \mapsto D]}$, which entails the conclusion. ■

In the sequel, we take D to be Scott's λ-model D_∞ where D_0 is the two-points lattice, and $\nabla = \{\bot\}$.

11.1.34. THEOREM. *Let Γ be a tail μ-free basis. If*

$$\Gamma \vdash_{\lambda\mu^*} M : A_1 \to ... \to A_n \to \mu\alpha.B_1 \to ... \to B_m \to \alpha \quad (n, m \geq 0)$$

where $A_1, ..., A_n, B_1, ..., B_m$ are tail μ-free, then M is unsolvable.

PROOF. Notice that we can assume that M is closed and Γ is empty. Otherwise, let $\Gamma = \{x_1 : A'_1, ..., x_k : A'_k\}$ $(k \geq 0)$ where the free variables of M are among $x_1, ..., x_k$. Then $\vdash_{\lambda\mu^*} \lambda x_1 ... \lambda x_k.M : A'_1 \to ... \to A'_k \to A_1 \to ... \to A_n \to \mu\alpha.B_1 \to ... \to B_m \to \alpha$ (observe that M is unsolvable iff $\lambda x_1 ... \lambda x_k.M$ is unsolvable). Now, let η_D be the type environment such that $\eta_D(\beta) = D$ for all type variables β. Then, by Fact 11.1.33, $\mathcal{I}[\![A_i]\!]_{\eta_D} = D = \mathcal{I}[\![B_j]\!]_{\eta_D}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. Let now $C = \mu\alpha.B_1 \to ... \to B_m \to \alpha$. By a straightforward induction on n it is easy to see that $\mathcal{I}^n[\![C]\!]_{\eta_D} = \{\bot\}$ and then $\mathcal{I}[\![C]\!]_{\eta_D} = \{\bot\}$ which implies $\mathcal{I}[\![A_1 \to ... \to A_n \to \mu\alpha.B_1 \to ... \to B_m \to \alpha]\!]_{\eta_D} = \{\bot\}$. By soundness (Corollary 11.1.32) we get the result. ■

For example we have $\vdash_{\lambda\mu^*} (\mathsf{Y}\,\mathsf{K}) : \mu\alpha.\beta\to\alpha$ (see Exercise **??**) where α is an atomic type and is therefore tail μ-free, and recall that $(\mathsf{Y}\,\mathsf{K})$ is a typical unsolvable term. On the other hand the condition that $A_1, ..., A_n, B_1, ...B_m$ are tail μ-free cannot be avoided, in general. For instance take $H = \lambda y.y((\lambda x.x\,x)(\lambda x.x\,x))$ which is solvable. Then $\vdash_{\lambda\mu^*} H : \mu\alpha.T\to\alpha$ where $T = \mu\beta.\alpha\to\mu\alpha.\beta\to\alpha$ (see exercise 11.3.10). Observe that T is not tail μ-free.

Another application of the type interpretations described so far is the following, easy proof of a standard result on definability of elements of D_∞ by means of *untyped* terms (see **?**).

11.1.35. Proposition. *The only definable element of $D_0 \subseteq D_\infty$ is $\bot$.*

Proof. Let $\xi =_{\rm def} \mu t.t\to t$, and recall that every pure λ-term has type ξ. Now, if $d\in D_0$ is definable, $d = [\![M]\!]$ for some pure closed term M such that $\vdash_{\lambda\mu^*} M : \xi$. Therefore $d\in\mathcal{I}[\![\xi]\!]_\eta$ (for any type environment η) by Corollary 11.1.32, and then $d = d_0 \in (\mathcal{I}[\![\xi]\!]_\eta)_0 = \nabla$. Hence $d = \bot$. ■

Completeness

Partial converses to the above soundness results have been proved in **?** (see also **?**) for a type interpretation which makes use of ideals (instead of complete and uniform sets) over a D of the shape $\mathbf{A} + [D\to D]$. On the one hand, the interpretation in D of every unsolvable λ-term is $\bot$ (see **?** for the notion of unsolvable terms and their interpretation in topological λ-models). Now, $\bot$ is an element of every ideal (or of any complete and uniform subset, for that matter), therefore, if M is such a term, it is true that $[\![M]\!]_\rho \in \mathcal{I}[\![A]\!]_\eta$ for any term environment ρ, any type A and any type environment η. The incompleteness of all type inference systems presented above, in particular of $\vdash_{\lambda\mu^*}$, becomes apparent just considering the λ-term $\Delta_3\Delta_3$, where $\Delta_3 =_{\rm def} \lambda x.xxx$, which is unsolvable of order 0, yet has principal type scheme $\mu\alpha.\alpha\to\alpha$. Clearly this cannot be remedied by adding to $\vdash_{\lambda\mu^*}$ a rule giving the same types to $(\beta\eta)$-convertible terms:

$$(Eq)\quad \frac{\Gamma\vdash_{\lambda\mu^*} M : A \quad M =_{\beta\eta} N}{\Gamma\vdash_{\lambda\mu^*} N : A}$$

The system introduced in **?** exploits the notion of *approximant* of a λ-term in the formulation of an infinitary rule that assigns a type to a term when all its approximants can be assigned that type:

11.1.36. Definition (Approximants of λ-terms). Let $\Lambda_\bot$ be the set of λ-terms with a new constant $\bot$. (i) For a λ-term M, define the notion of *direct approximant* $\omega(M)\in\Lambda_\bot$ of M as follows, by induction on M:

1. $\omega(x) = x$ if x is a variable,
2. $\omega(\lambda x.M) = \lambda x.\omega(M)$,
3. $\omega(xM_1\dots M_k) = x\omega(M_1)\dots\omega(M_k)$, for $k > 0$,

4. $\omega(cM_1 \dots M_k) = c\omega(M_1) \dots \omega(M_k)$, for $k > 0$,

5. $\omega((\lambda x.M)M_1 \dots M_k) = \bot$, for $k > 0$.

(ii) The set of *approximants*, $\mathcal{A}(M)$, of the λ-term M, is defined by

$$\mathcal{A}(M) =_{\text{def}} \{A \in \Lambda_\bot \mid M \to^*_\beta N \text{ and } A = \omega(N)\},$$

where $\to^*_\beta$ is the transitive closure of β-reduction.

Of course, the constant $\bot$ is interpreted as the least element of D. A classical result is the following Approximation Theorem (**?**):

11.1.37. THEOREM. *For any any term M and term environment ρ:*

$$[\![M]\!]_\rho = \bigsqcup\{[\![A]\!]_\rho \mid A \in \mathcal{A}(M)\}.$$

We can now introduce the new type inference system:

11.1.38. DEFINITION. The type inference system $(\lambda\mu^*\infty)$ is defined by adding to the system $\vdash_{\lambda\mu^*}$ the rules:

$$(\bot) \quad \Gamma \vdash_{\lambda\mu^*\infty} \bot : A$$

$$(C) \quad \frac{\Gamma \vdash_{\lambda\mu^*\infty} P : A \ \text{ for all } P \in \mathcal{A}(M)}{\Gamma \vdash_{\lambda\mu^*\infty} M : A}$$

For the system $(\lambda\mu^*\infty)$ it is possible to prove a form of completeness:

11.1.39. THEOREM. *$\Gamma \vdash_{\lambda\mu^*\infty} M : A$ if, and only if, $\Gamma \models M : A$.* ■

A proof of this result and a thorough discussion of the system $(\lambda\mu^*\infty)$ is contained in **?**.

More directly relevant to applications is another completness result, which states that $=^*$ completely describes the equivalence which identifies two recursive types whenever their interpretations are identical in all type environments: this provides a strong semantical evidence for the use of $=^*$ as the preferred notion of equality of recursive types. This result can be proved by following the same idea as in **?** and **?**, by using a domain $D \cong \mathbf{A} + [D \to D]$:

11.1.40. THEOREM. *Let $A, B \in \mathbb{T}_\mu$. Then $A =^* B$ if and only if, for all type environments η, $\mathcal{I}[\![A]\!]_\eta = \mathcal{I}[\![B]\!]_\eta$.* ■

Interpreting simultaneous recursions

The same ideas used in the definition of the interpretation of μ-types can be applied to recursive types defined by simultaneous recursions. This allows us to extend the present treatment to type algebras of the form $(\lambda\mathcal{R})$, for a simultaneous recursion

$$\mathcal{R}(\vec{X}) = \{X_i = A_i \mid 1 \leq i \leq n\}$$

where $A_i \in \mathbb{T}(\mathbb{A} \cup \vec{X})$, for each $i = 1, \ldots, n$. Let $\vec{\xi} = \xi_1, \ldots, \xi_n$ be a solution of $\mathcal{R}(\vec{X})$, then $\xi_i \in \mathsf{Tr}_{\mathrm{R}}$ and, by Lemma 8.6.17(i) there is $A_{\xi_i} \in \mathbb{T}_\mu(\mathbb{A})$ such that $(A_{\xi_i})^* = \xi_i$. Let η be the type environment defined by the mapping $X_i \mapsto \mathcal{I}[\![A_{\xi_i}]\!]$, which is independent from the choice of A_{ξ_i} by Proposition 11.1.31. Then the clauses:

$$\begin{array}{rcl} \mathcal{J}[\![X_i]\!]\eta & = & \eta(X_i) \\ \mathcal{J}[\![A' \to A'']\!]\eta & = & \mathcal{J}[\![A']\!]\eta \to \mathcal{J}[\![A'']\!]\eta \end{array}$$

define inductively the extension of η to type homomorphisms:

$$\mathbb{T}(\mathbb{A} \cup \vec{X})/=_{\mathcal{R}(\vec{X})} \longrightarrow \mathcal{S}(D)$$

and

$$\mathbb{T}(\mathbb{A} \cup \vec{X})/=^*_{\mathcal{R}(\vec{X})} \longrightarrow \mathcal{S}(D).$$

11.2. Type interpretations in systems with explicit typing

One straightforward way of interpreting the explicitly typed calculi with recursive types consists in restricting the full type structure introduced in Part I, Chapter 3.1[1], by replacing sets with domains and functions with (Scott-)continuous functions. Indeed, there is a sense in which domain theory and the solution of recursive domain equations can be regarded as a purely semantical theory of recursive types. The leading idea is to define $\mathcal{M}(A \to B)$ as the domain of continuous functions from $\mathcal{M}(A)$ to $\mathcal{M}(B)$; a recursive type $\mu\alpha.A$ is interpreted as the solution of the domain equation $D \cong FD$, where F interprets the mapping over domains defined by (the interpretation of) A: such a solution can be built, for example, as the limit of an inverse sequence of approximations, following **?**.

In order to overcome some pathologies that arise in the resulting model, for example the fact that for many natural choices of category of domains the interpretation of type $\mu\alpha.\alpha \to \alpha$ collapses to one point, alternative interpretations are available.

On the one hand, we can interpret types as closures over the λ-model $\mathcal{P}\omega$ following **?**. Such a construction yields far more than a model for simple recursive types: indeed, it has been used in **?** and **?** to interpret the polymorphic λ-calculus and in **?** to give a model for calculi with dependent types and a type of all types.

[1] Some labels in Part I at the cited point are useful here and below. Could you (or we) add them?

On the other hand, types can be interpreted as partial equivalence relations (more precisely, as complete and uniform per's) over a continuous λ-model with a notion of approximation, along the lines of Section 11.1. We shall sketch these model constructions in the exercises (see Exercises **??**).

In this section we focus in particular on the system ($\lambda\mu$-Ch$_0$) in which all possible recursive types are present. The advantage of this system is that the terms $fold_{\mu\alpha.A}$ and $fold_{\mu\alpha.A}$ afford an explicit notation for the isomorphisms provided by the solution of recursive domain equations and formalize the intuitive idea that a recursive type is interpreted as such a solution.

We assume below that $\mathbb{A}$ is a collection of basic types containing at least one element with a non empty interpretation.

Domain models

The category ω-**CPO** defined in §11.1 has a terminal object $\mathbf{1} =_{\text{def}} \{\bot\}$. If D, E are ω-cpo's, their *product* $D \times E$ is their cartesian product ordered coordinatewise, with projections $\mathtt{fst} : (D \times E) \to D$ and $\mathtt{snd} : (D \times E) \to E$. Given continuous $f : C \to D, g : C \to E$ there is a unique continuous function $\langle f, g\rangle : C \to (D \times E)$ such that $\mathtt{fst} \circ \langle f, g\rangle = f$ and $\mathtt{snd} \circ \langle f, g\rangle = g$. Hence ω-**CPO** is cartesian. As usual, product is made functorial by defining its action on morphisms as

$$f \times g =_{\text{def}} \langle f \circ \mathtt{fst}, g \circ \mathtt{snd}\rangle : (D \times E) \to (D' \times E')$$

for $f : D \to D'$ and $g : E \to E'$. If D, E are ω-cpo's, their *exponential* $[D \to E]$ is the ω-cpo of all continuous functions $D \longrightarrow E$ ordered pointwise, by setting

$$f \sqsubseteq g \quad \text{if and only if} \quad \forall d \in D. f(d) \sqsubseteq g(d)$$

for $f, g \in [D \to E]$. For continuous $f : D' \longrightarrow D, g : E \longrightarrow E'$, define $[f \to g] : [D \to E] \to [D' \to E']$ as $\boldsymbol{\lambda} h \in [D \to E]. g \circ h \circ f$. These constructions make ω-**CPO** a cartesian closed category.

Solving recursive domain equations

A recursive domain equation has the general form $D \cong T(D, D)$, where we consider only the case of a binary functor over ω-**CPO**, contravariant in its first argument and covariant in the second one:

$$T : \omega\text{-}\mathbf{CPO}^{op} \times \omega\text{-}\mathbf{CPO} \longrightarrow \omega\text{-}\mathbf{CPO} \tag{11.17}$$

with nice continuity properties (the sketch below summarizes the results that we shall need for the model construction; complete details can be found in **?** or **?**). We associate to the category ω-**CPO** a category ω-$\mathbf{CPO}^{ep}$ whose morphisms from D to E are the embedding-projection pairs $\langle e : D \longrightarrow E, p : E \longrightarrow D\rangle$ as in Definition 11.1.7. There are two properties of a functor (of mixed variance) that ensure that it is well-behaved.

11.2.1. Definition. 1. T is *locally monotonic* if $T(f, g) \sqsubseteq T(f', g')$ whenever $f \sqsubseteq f' : D' \longrightarrow D$ and $g \sqsubseteq g' : E \longrightarrow E'$.

2. T is *locally continuous* if

$$T(\bigsqcup_{n\in\omega} f_n, \bigsqcup_{n\in\omega} g_n) = \bigsqcup_{n\in\omega} T(f_n, g_n)$$

whenever $\langle f_n\rangle_{n\in\omega}$ and $\langle g_n\rangle_{n\in\omega}$ are increasing ω-chains.

11.2.2. LEMMA. *1. A locally monotonic functor preserves embedding-projection pairs.*

2. If T as in (11.17) is locally monotonic, then it induces a functor

$$T^{ep} : \omega\text{-}\mathbf{CPO}^{ep} \times \omega\text{-}\mathbf{CPO}^{ep} \longrightarrow \omega\text{-}\mathbf{CPO}^{ep}$$

by setting $T^{ep}(D,E) =_{\text{def}} T(D,E)$ and $T^{ep}(\langle e,p\rangle,\langle e',p'\rangle) =_{\text{def}} T(p,e')$. If T is also locally continuous, then T^{ep} preserves ω-colimits in ω-$\mathbf{CPO}^{ep}$.

11.2.3. EXAMPLE. It is easily seen that

$$[\cdot\to\cdot] : \omega\text{-}\mathbf{CPO}^{op} \times \omega\text{-}\mathbf{CPO} \longrightarrow \omega\text{-}\mathbf{CPO}$$

is ω-monotonic and ω-continuous: this follows from continuity of composition. Observe that, if $\langle e_1,p_1\rangle : D_1 \longrightarrow E_1$ and $\langle e_2,p_2\rangle : D_2 \longrightarrow E_2$ are embedding-projection pairs, then we have an embedding-projection pair $\langle [p_1\to e_2],[e_1\to p_2]\rangle : [D_1\to D_2] \longrightarrow [E_1\to E_2]$.

In the sequel, let $F(D) =_{\text{def}} T^{ep}(D,D)$. The solution of an equation $D \cong F(D)$ will be obtained as the inverse limit of a sequence of projections:

$$D_0 \xleftarrow{j_0} D_1 \xleftarrow{j_1} D_2 \xleftarrow{j_2} \cdots \xleftarrow{j_{n-1}} D_n \xleftarrow{j_n} D_{n+1} \xleftarrow{j_{n+2}} \cdots \tag{11.18}$$

or, equivalently, as the direct limit of the sequence of embeddings:

$$D_0 \xrightarrow{i_0} D_1 \xrightarrow{i_1} D_2 \xrightarrow{i_2} \cdots \qquad D_n \xrightarrow{i_n} D_{n+1} \xrightarrow{i_{n+1}} \cdots \tag{11.19}$$

where:

$$\begin{aligned} D_0 &= \mathbf{1} \\ D_{n+1} &= F(D_n) \end{aligned}$$

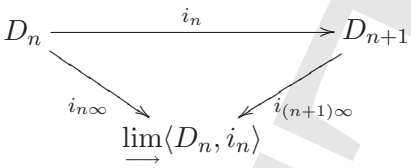
and the pairs $\langle i_n, j_n\rangle$, for $n > 0$, are embedding-projection pairs from D_n to D_{n+1} and are defined inductively by

$$\begin{aligned} i_n &= F(i_{n-1}) : F(D_{n-1}) \longrightarrow F(D_n) \\ j_n &= F(j_n) : F(D_n)F(D_{n-1}) \longrightarrow F(D_n) \end{aligned}$$

The direct limit of the sequence (11.19) can be described explicitly as

$$D_\infty =_{\text{def}} \varinjlim\langle D_n, i_n\rangle = \{\langle d_n\rangle_{n\in\omega} \mid \forall n\in\omega.d_n \in D_n \text{ and } j_{n+1}(d_{n+1}) = d_n\}.$$

with embeddings $i_{n\infty} : D_n \to D_\infty$ and projections $j_{\infty n} : D_\infty \to D_n$, for all $n \in \omega$, making all the diagrams



commute. From the continuity of F and universality we obtain:

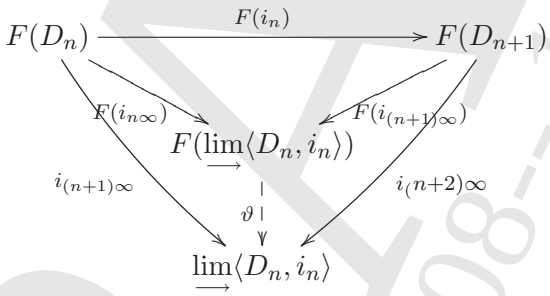
11.2.4. Proposition. *There is a unique isomorphism $\vartheta : F(D_\infty) \longrightarrow D_\infty$, given by*

$$\bigsqcup_{n \in \omega} \left(i_{(n+1)\infty} \circ F(j_{\infty n})\right)$$

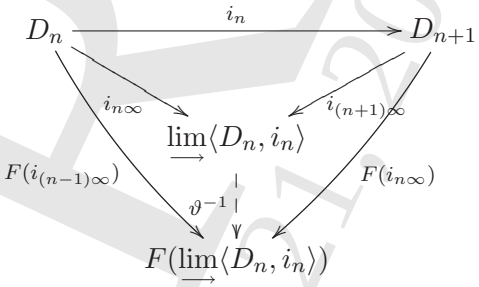
whose inverse ϑ^{-1} is

$$\bigsqcup_{n \in \omega} \left(F(i_{n\infty}) \circ j_{\infty(n+1)})\right)$$

For a proof, we refer the reader to **?**. The construction is based on the observation that all the following diagrams (in ω-$\mathbf{CPO}^{ep}$) commute:



(11.20)



(11.21)

Interpreting μ-types as domains

With the machinery set up for the solution of recursive domain equations we can now easily define a $\mathbb{T}_\mu$-applicative type structure by interpreting each $A \in \mathbb{T}_\mu = \mathbb{T}_\mu(\mathbb{A})$ as an $\omega - \mathbf{CPO}$ $\mathcal{M}^{\omega\text{-CPO}}(A)$ (an interpretation in ω-$\mathbf{CPO}$), by induction on A. If η maps $\mathbb{A}$ to the objects of ω-$\mathbf{CPO}$, then it can be extended to an interpretation of $\mathbb{T}_\mu$ as follows:

(a) $\mathcal{M}_\eta^{\omega\text{-CPO}}(\alpha) = \eta(\alpha)$;

(b) $\mathcal{M}_\eta^{\omega\text{-CPO}}(A \to B) = [\mathcal{M}_\eta^{\omega\text{-CPO}}(A) \to \mathcal{M}_\eta^{\omega\text{-CPO}}(B)]$;

(c) $\mathcal{M}_\eta^{\omega\text{-CPO}}(\mu\alpha.A) = \varinjlim \langle F^n(\mathbf{1}), i_n \rangle$,

where $F(D) =_{\text{def}} \mathcal{M}^{\omega\text{-CPO}}_{\eta[\alpha \mapsto D]}(A)$ and $i_{n+1} =_{\text{def}} F(i_n)$, with i_0 and j_0 uniquely determined by strictness of embeddings. Observe that F preserves ω-colimits of embeddings because it arises from the composition of functors that are locally monotonic and locally continuous: identity, constant functors, exponentiation and product. Summarizing we have the following result:

11.2.5. THEOREM. *$\mathcal{M}^{\omega\text{-}CPO}$ is a Π_μ-applicative type structure.*

by We can extend in a straightforward way Definition [3.1.7] and define an interpretation of every term in $M \in \Lambda^{\text{Ch}}_{\Pi_\mu}(A)$. In particular, define

$$\begin{aligned} fold_{\mu\alpha.A} &=_{\text{def}} \theta : F(\mathcal{M}_\eta^{\omega\text{-CPO}}(\mu\alpha.A)) \longrightarrow \mathcal{M}_\eta^{\omega\text{-CPO}}(\mu\alpha.A) \\ unfold_{\mu\alpha.A} &=_{\text{def}} \theta^{-1} : \mathcal{M}_\eta^{\omega\text{-CPO}}(\mu\alpha.A) \longrightarrow F(\mathcal{M}_\eta^{\omega\text{-CPO}}(\mu\alpha.A)) \end{aligned}$$

where $F(D)$ is defined, as above, as $F(D) =_{\text{def}} \mathcal{M}^{\omega\text{-CPO}}_{\eta[\alpha \mapsto D]}(A)$, and θ and θ^{-1} are as in diagrams (11.20) and (11.21), respectively. It is immediate to verify that this interpretation is well-defined, and not trivial.

11.2.6. THEOREM. *$\mathcal{M}^{\omega\text{-}CPO}$ is a λ_{Π_μ}-model.*

[Should we prove somewhere that $\mathcal{M}^{\omega\text{-CPO}}$ is not trivial?]. I think that $\mathcal{M}^{\omega\text{-CPO}}$ is also a $\lambda_{\Pi^*_\mu}$ applicative structure and model. It should be pointed out somewhere]

The following result is a sample application of the interpretation of terms of ($\lambda\mu$-Ch$_0$) outlined above. It relates the interpretation of the typed fixed-point combinator Y_μ of Exercise 8.7.16 to the iterative construction of fixed-points of continuous endofunctions of an ω-cpo D (see Theorem 11.1.6). It is similar to a well-known result of **?** on the interpretation of the **Y** combinator in Scott's D_∞ models, and was stated for the typed case in **?**:

11.2.7. THEOREM. *The interpretation of Y_μ as a term of type $(A \to A) \to A$ coincides, for any environment η, with the functional* **fix** $\in [\mathcal{M}_\eta^{\omega\text{-}CPO}(A) \to \mathcal{M}_\eta^{\omega\text{-}CPO}(A)] \to \mathcal{M}_\eta^{\omega\text{-}CPO}(A)$.

PROOF. A straightforward analysis of the interpretation of recursive types in the category ω-**CPO**, whose details are left as an exercise. ■

[I'm still working out the following results: some statements need to be refined.] Observe that types like,e.g., $\mu\alpha.\alpha \to \alpha$, have a trivial interpretation in $\mathcal{M}^{\omega\text{-CPO}}$. While this cannot be fixed generally, we can nevertheless draw some useful information from this. Let's first characterize a class of types having trivial interpretations in $\mathcal{M}^{\omega\text{-CPO}}$.

11.2.8. DEFINITION. A type B is *trivial* if either $B =_\mu \mu\alpha_1 ... \mu\alpha_k.A_1 \to ... \to A_n \to \alpha_i$ $(k, n \geq 0, 1 \leq i \leq k)$ or $B =_\mu B' \to B''$ where B'' is trivial.

The proof of the following property is a simple exercise (see Exercise 11.3.11):

11.2.9. LEMMA. *let A be a trivial type. Then $\mathcal{M}^{\omega\text{-}CPO}(A) \cong \mathbf{1}$.*

Let's define the set of *contexts* obtained by enriching the syntax of terms by adding a new term, the *hole*, $[-]^A$ to each type A. Let $\mathcal{C}^{\mathrm{Ch}}_{\Pi_\mu}(A)$ denote the resulting set of terms for each type A. A context, denoted $C[-]^A$, represents a term with possible occurrences of the hole of type A that can be filled by any term of type A. If $C[-]^A \in \mathcal{C}^{\mathrm{Ch}}_{\Pi_\mu}(B)$ and $M \in \Lambda^{\mathrm{Ch}}_{\Pi_\mu}(A)$, then $\mathcal{C}[M]^A \in \Lambda^{\mathrm{Ch}}_{\Pi_\mu}(B)$ is the well formed term obtained by replacing $[-]^A$ with M. Note that contexts allows the capture of free variables in M by λ abstractions occurring in $\mathcal{C}[-]^A$. For instance if $\mathcal{C}[-]^A = \lambda x^A\, \lambda y^{A\to B}.y^{A\to B}[-]^A \in \mathcal{C}^{\mathrm{Ch}}_{\Pi_\mu}(A \to (A \to B) \to B)$ then $\mathcal{C}[x^A]^A$ is the well formed term $\lambda x^A\, \lambda y^{A\to B}.y^{A\to B}x^A \in \Lambda^{\mathrm{Ch}}_{\Pi_\mu}(A \to (A \to B) \to B)$ in which the variable x^A is bound.

We have the following Genericity Lemma for ($\lambda\mu$-Ch) which states that all terms having trivial type are observationally equivalent in the following sense.

11.2.10. LEMMA. *[Genericity for ($\lambda\mu$-Ch)] Let $C[-]^A \in \mathcal{C}^{\mathrm{Ch}}_{\Pi_\mu}(\alpha)$ where $\alpha \in \mathbb{A}$ is a basic type. Then given any two terms $M, N \in \Lambda^{\mathrm{Ch}}_{\Pi_\mu}(A)$ we have $C[M]^A =_{\beta\eta} C[N]^A$.*

PROOF (easy).

We can now state an easy result that shows that it is consistent to identify terms with the same trivial type.

11.2.11. THEOREM. (i) *Let $\mathcal{T} = \{M = N \mid M, N \in \Lambda^{\mathrm{Ch}}_{\Pi_\mu}(A)$ and A is trivial$\}$. Then $\mathcal{T} \subseteq \mathrm{Th}(\mathcal{M}^{\omega\text{-}CPO})$.*

(ii) *$\mathrm{Th}(\mathcal{M}^{\omega\text{-}CPO})$ is consistent.*

PROOF. ...

[Question (for Felice). Is it true that $\mathrm{Th}(\mathcal{M}^{\omega\text{-CPO}}) = \mathcal{T}$.]

11.3. Exercises

11.3.1. Fill in the details of Theorem 11.1.6.

11.3.2. Given an embedding-projection pair

$$e : D \to E \quad p : E \to D$$

as in Definition 11.1.7, show that:

- e is injective and strict;
- p is an onto and strict.

11.3.3. Prove that the equations of Definition 11.1.3 hold when D is defined as the inverse limit of the sequence of projections 11.10.
[*Hint.* See **?**.]

11.3.4. Let $\mathcal{F}$ be the model described in Chapter 14 [Need precise references to definitions contained in that chapter.] To every intersection type A associate a *height* $|A|$ as follows:

(i) $|\Omega| = 0$,

(ii) $|\varphi_i| = 1$, for all $\varphi_i \in \mathbb{C}$ such that $\varphi_i \neq \Omega$,

(iii) $|A \cap B| = \max\{|A|, |B|\}$,

(iv) $|A \to B| = 1 + \max\{|A|, |B|\}$.

Given a filter $d \in \mathcal{F}$, define

$$d[n] =_{\text{def}} \{A \in d \mid |A| \leq n\}$$

and set $d_n =_{\text{def}} \uparrow d[n]$, where $\uparrow X$ is the filter generated by the set X (this is the intersection of all filters containing X). Prove that the mappings $d \mapsto d_n : \mathcal{F} \to \mathcal{F}$ define a notion of approximation over $\mathcal{F}$.
[*Hint.*

1. Show that, for a filter d, the set $d[n]$ is closed under finite intersections. As a consequence, $B \in \uparrow d[n]$ if and only if $B \geq A$ for some $A \in d[n]$.
2. For equation (11.1), prove that $d_0 = \uparrow\{\Omega\}$.
3. In order to prove some of the equations for a notion of approximation, you may need the following properties of the preorder on intersection types:
   - If $B \nsim \Omega$ and $A' \to B' \leq A \to B$, then $A \leq A'$ and $B' \leq B$.
   - If $C \leq A \to B$ and $|C| \leq n+1$, then there exist A', B' such that $|A'|, |B'| \leq n$ and $C \leq A' \to B' \leq A \to B$.

   For the proof of the latter, define the relation $\lessdot$ on types by the same axioms and rules as $\leq$, with the exception of reflexivity and transitivity, then show that $A \leq B$ if and only if $A(\lessdot)^* B$.

]

11.3.5. Define Engeler's model D_A as $\mathcal{P}(D^*)$, where $D^* = \bigcup_{n \in \omega} D_n$ and

$$\begin{aligned} D_0 &= \emptyset \\ D_1 &= A \\ D_{n+2} &= \{(\beta, m) \mid m \in D_{n+1}, \beta \subseteq_{\text{fin}} D_{n+1}\} \end{aligned}$$

D_A can be turned into an applicative structure by defining, for $d, e \in D_A$:

$$d \cdot e = \{m \in D_A \mid \exists \beta \subseteq_{\text{fin}} e.(\beta, m) \in d\}.$$

Show that $d_n =_{\text{def}} d \cap D_n$ defines a notion of approximation for D_A.

11.3.6. Prove Lemma 11.1.18.

11.3.7. Prove Lemma 11.1.26.

[*Hint.* Prove that

$$\forall n \in \omega . \mathcal{I}^n [\![A[\alpha := B]]\!]_\eta = \mathcal{I}^n [\![A]\!]_{\eta[\alpha \mapsto \mathcal{I}[\![B]\!]_\eta]}.$$

by induction.]

11.3.8. (Bekic's Theorem for recursive types) Prove that the equation:

$$\mu\beta.(A[\alpha := \mu\alpha.B]) = (\mu\beta.A)[\alpha := \mu\alpha.(B[\beta := \mu\beta.A])],$$

is valid in the interpretation described in Section 11.1.
[*Hint.* Let $\hat{A} =_{\text{def}} A[\alpha := \mu\alpha.B]$ and $\hat{B} =_{\text{def}} B[\alpha := \mu\beta.A]$. Then prove simultaneously by induction on $n \in \omega$ that the following two equations hold:

(i) $\mathcal{I}^n [\![\mu\beta.\hat{A}]\!]_\eta = \mathcal{I}^n [\![\mu\beta.A[\alpha := \mu\alpha.\hat{B}]]\!]_\eta$,

(ii) $\mathcal{I}^n [\![\mu\alpha.\hat{B}]\!]_\eta = \mathcal{I}^n [\![\mu\alpha.B[\beta := \mu\beta.\hat{A}]]\!]_\eta$.

]

11.3.9. (Metric interpretation of contractive recursive types) For $X, Y \in \mathcal{CU}$, let

$$\delta(X,Y) = \begin{cases} \max\{n \mid X_n = Y_n\} & \text{if } X \neq Y \\ \infty & \text{otherwise} \end{cases}$$

Define a function $d : \mathcal{CU} \times \mathcal{CU} \to \mathbf{R}^+$ by setting

$$d(X,Y) = \begin{cases} S^{\delta(X,Y)} & \text{if } X \neq Y \\ 0 & \text{otherwise} \end{cases}$$

Show that

(1) $\langle \mathcal{CU}, d \rangle$ is a complete ultrametric space.

Given a metric space $\langle M, d \rangle$, a mapping $f : M \to M$ is *contractive* if there is a positive real number $c < 1$ such that $d(f(x), f(y) \leq d(x,y)$ for all $x, y \in M$. Banach's Theorem states that every such mapping has a unique fixed point $\mathbf{fix}(f)$, defined as $\lim_{n \to \infty} f^{(n)}(x)$, where $x \in M$ is an arbitrary point.

(2) The functions $\boldsymbol{\lambda} X \in \mathcal{CU}.X \to Y$ for a fixed $Y \in \mathcal{CU}$, $\boldsymbol{\lambda} Y \in \mathcal{CU}.X \to Y$ for a fixed $X \in \mathcal{CU}$ and $\boldsymbol{\lambda} X \in \mathcal{CU}.X \to X$ are contractive, where the function $\to : \mathcal{CU} \times \mathcal{CU} \to \mathcal{CU}$ is defined in Definition 11.1.13.

(3) If $\mu\alpha.A \in \mathbb{T}^{\mu c}$ define

$$[\![\mu\alpha.A]\!]^\eta = \mathbf{fix}(\boldsymbol{\lambda} X \in \mathcal{CU}.[\![A]\!]^\eta[\alpha \mapsto X]).$$

Use these facts to give an interpretation of all $A \in \mathbb{T}^{\mu c}$.
[*Hint.* See (**?**, **?**).]

11.3.10. Show that $\vdash_{\lambda\mu^*} \lambda y.y((\lambda x.x\,x)(\lambda x.x\,x)) : \mu\alpha.T \to \alpha$ where $T = \mu\beta.\alpha \to \mu\alpha.\beta \to \alpha$. [Recall that $(\lambda x.x\,x)(\lambda x.x\,x)$ has all types].

11.3.11. Prove Lemma 11.2.9.

11.3.12. A *partial equivalence relation (per)* over D is any symmetric and transitive relation $R \subseteq D \times D$, i. e.: if $\langle d, d' \rangle \in R$ then $\langle d', d \rangle \in R$ and if $\langle d, d' \rangle, \langle d', d'' \rangle \in R$ then $\langle d, d'' \rangle \in R$.

(i) Assuming that D is a λ-model with a notion of approximation, give a definition of complete and uniform per over D.

(ii) Generalize Definition 11.1.17 to the case where types are interpreted as complete and uniform partial equivalence relations over a λ-model D with a notion of approximation.

11.3.13. Use the interpretation of types as complete and uniform partial equivalence relations of Exercise 11.3.12 to build a model of ($\lambda\mu$-Ch$_0$).
[*Hint.* The main steps are as follows:

(i) Prove the following Fundamental Lemma: If $\Gamma \vdash_{\lambda\mu\mathrm{Ch}} M : A$, η is a type environment and ρ_1, ρ_2 are term environments such that $\langle \rho_1(x_i), \rho_2(x_i) \rangle \in \mathcal{I}[\![A_i]\!]_\eta$ whenever $x_i : A_i \in \Gamma$, then

$$\langle [\![M]\!]_{\rho_1}, [\![M]\!]_{\rho_2} \rangle \in \mathcal{I}[\![A]\!]_\eta.$$

(ii) Define the model $\mathcal{M} = \langle T, \{\mathcal{M}(R)\}_{R \in T}, \{\Phi_{RS}\}_{R,S \in T}, \mathrm{unf}_A \rangle$ setting T as the set of all complete and uniform per's over D, $\mathcal{M}(R) = \{[d]_R \mid \langle d, d \rangle \in R\}$ where, for a per R, $[d]_R =_{\mathrm{def}} \{e \in D \mid \langle d, e \rangle \in R\}$. Let $[\mathcal{M}(R) \to \mathcal{M}(S)]$ be the set of continuous functions $F(d) : D \to D$ for all $d \in D$ such that $\langle d, d \rangle \in R \to S$. Define the mappings Φ_{RS} by the assignment $\Phi_{RS}([d])([e]) =_{\mathrm{def}} [d \cdot e]_S$, where $[d]_{R \to S} = \in \mathcal{M}(R \to S)$ and $[e]_R \in \mathcal{M}(R)$.

(iii) Show that the above construction is well-defined

]

11.3.14. In this exercise, we sketch how to build a model $\mathcal{M}$ for the system ($\lambda\mu$-Ch$_0$) where types are interpreted as *closures* over the λ-model $\mathcal{P}\omega$. The construction can be adapted easily to systems of the form ($\lambda\mathcal{A}$-Ch), for a type algebra $\mathcal{A}$, provided we assume that elements of $\mathcal{A}$ can be mapped homomorphically to closures. We presuppose a basic knowledge of the λ-model $\mathcal{P}\omega$, see for example **?** and **?**. For a related construction see **?**. Define a *closure* (over $\mathcal{P}\omega$) as a continuous function $a : \mathcal{P}\omega \to \mathcal{P}\omega$ such that $\mathbf{I} \subseteq a = a \circ a$ where $\mathbf{I} \in \mathcal{P}\omega$ is the interpretation of the identity function. Let $\mathcal{V}$ denote the collection of all closures over $\mathcal{P}\omega$. Given $a \in \mathcal{V}$, its range $\mathrm{im}(a)$ coincides with the set $\{x \in \mathcal{P}\omega \mid a(x) = x\}$. It can easily be proved that $\mathrm{im}(a)$ is an ω-algebraic lattice. Moreover, **?** proves that, if D is an algebraic lattice with a countable basis, then there is a closure a_D over $\mathcal{P}\omega$ such that $D \cong \mathrm{im}(a_D)$. This representation of countably based algebraic lattices as closures over $\mathcal{P}\omega$ lifts nicely to continuous function spaces, as for all closures a, b there is a continuous bijection

$$\Phi_{ab} : \mathrm{im}(a \Rightarrow b) \xrightarrow{\sim} [\mathrm{im}(a) \to \mathrm{im}(b)].$$

11.3.15. Define the mapping $\mathcal{I}[\![\cdot]\!]_\eta : \mathbb{T}_\mu \to \mathcal{V}$, where η is a type environment assigning to each atom a closure:

(a) $\mathcal{I}[\![\alpha]\!]_\eta = \eta(\alpha)$;

(b) $\mathcal{I}[\![A \to B]\!]_\eta = \mathcal{I}[\![A]\!]_\eta \Rightarrow \mathcal{I}[\![B]\!]_\eta$;

(c) $\mathcal{I}[\![\mu\alpha.A]\!]_\eta = \mathbf{fix}(\lambda\!\!\lambda a \in \mathcal{V}.\mathcal{I}[\![A]\!]_{\eta[\alpha \mapsto a]})$;

Show that this mapping is well-defined.

11.3.16. Prove that whenever $A, B \in \mathbb{T}_\mu$ are such that $A =_\mu B$ in the formal system of Definition 8.3.25(i), then $\mathcal{I}[\![A]\!]_\eta = \mathcal{I}[\![B]\!]_\eta$, for any type environment η.

11.3.17. Define a partial order on $\mathsf{Tr}_\infty(\mathbb{A})$, exploiting the "undefined" tree $\bullet$, as follows: for $s, t \in \mathsf{Tr}_\infty(\mathbb{A})$

$$\begin{aligned} s \preccurlyeq t \Leftrightarrow \mathrm{dom}(s) \subseteq \mathrm{dom}(t), \quad & \text{and} \\ \forall w \in \mathrm{dom}(s).s(w) \neq \bullet \quad & \text{implies} \quad s(w) = t(w) \end{aligned}$$

- Prove that the partially ordered set $\langle \mathsf{Tr}_\infty(\mathbb{A}), \preccurlyeq \rangle$ is ω-complete and has $\bullet$ as least element.
- Prove that $\mathsf{Tr}_\infty(\mathbb{A})$ is the initial ω-complete Σ-algebra, where $\Sigma = \mathbb{A} \cup \{\to\} \cup \{\bullet\}$.
- Conclude that for every $\eta : \mathbb{A} \to \mathcal{V}$, there is a unique continuous Σ-homomorphic extension of η:

$$\mathcal{T}[\![\cdot]\!]\eta : \mathsf{Tr}_\infty(\mathbb{A}) \to \mathcal{V}$$

which satisfies the equations:

(i) $\mathcal{T}[\![\alpha]\!]\eta = \eta(\alpha)$ for any $\alpha \in \mathbb{A}$;

(ii) $\mathcal{T}[\![\bullet]\!]\eta = \mathbf{I}$;

(iii) $\mathcal{T}[\![t_1 \to t_2]\!]\eta = \mathcal{T}[\![t_1]\!]\eta \Rightarrow \mathcal{T}[\![t_2]\!]\eta$

for all $t_1, t_2 \in \mathsf{Tr}_\infty(\mathbb{A})$.

[*Hint.* You may like to consult **?** or **?**]

- Prove that for all types $A, B \in \mathbb{T}_\mu(\mathbb{A})$ and all type environments η:

(i) $\mathcal{T}[\![A^*]\!]\eta = \mathcal{I}[\![A]\!]_\eta$.

(ii) $\mathcal{I}[\![A]\!]_\eta = \mathcal{I}[\![B]\!]_\eta$ whenever $A =^* B$.

[*Hint.* By structural induction on $A \in \mathbb{T}_\mu(\mathbb{A})$. Show first that for any pair of infinite trees $s, t \in \mathsf{Tr}_\infty(\mathbb{A})$, we have

$$\mathcal{T}[\![s[\alpha := t]]\!]\eta = \mathcal{T}[\![s]\!]\eta[\alpha \mapsto \mathcal{T}[\![t]\!]\eta]$$

and use the fact that $\mathcal{I}[\![\mu\alpha.A_1]\!]_\eta = \bigsqcup_{n \in \omega} \Theta^{(n)}(\bot)$, where $\Theta^{(0)}(\bot) = \bot$ and $\Theta^{(n+1)}(\bot) = \mathcal{I}[\![A_1]\!]_{\eta[\alpha \mapsto \Theta^{(n)}(\bot)]}$. See **?**, where it is also proved the converse of this soundness result: for $A, B \in \mathbb{T}_\mu(\mathbb{A})$, $A =^* B$ if and only if, for all $\eta : \mathbb{A} \to \mathcal{V}$, $\mathcal{I}[\![A]\!]_\eta = \mathcal{I}[\![B]\!]_\eta$.]

11.3.18. For any $a \in \mathcal{V}$, define $\mathcal{M}(a)$ as $\mathrm{im}(a)$: we have isomorphisms

$$\Phi_{ab} : \mathcal{M}(a \Rightarrow b) \longrightarrow [\mathcal{M}(a) \to \mathcal{M}(b)]$$

for any pair $a, b \in \mathcal{V}$ and (trivial) isomorphisms

$$\Psi_{ab} : \mathcal{M}(a) \to \mathcal{M}(b)$$

whenever $a = b$. Define the model $\mathcal{M}$ of as a structure:

$$\mathcal{M} = \langle \mathcal{V}, \{\mathcal{M}(a)\}_{a \in \mathcal{V}}, \{\Phi_{ab}\}_{a,b \in \mathcal{V}}, \{\Psi_{ab}\}_{a,b \in \mathcal{V}} \rangle,$$

11.3.19. Given a basis Γ, a Γ, η-*environment* is a function ρ such that $\rho(x) \in \mathcal{M}(\mathcal{I}[\![A]\!]_\eta)$ whenever the statement $x : A$ belongs to the basis Γ. Define the interpretation of a term (w.r.t. a Γ, η-environment) by induction on its construction tree in the system ($\lambda\mu$-Ch$_0$):

$$\begin{aligned}
&[\![\Gamma, x : A \vdash_{\lambda\mu\mathrm{Ch}} x : A]\!]_{\eta\rho} = \\
&\quad \rho(x), \\
&[\![\Gamma \vdash_{\lambda\mu\mathrm{Ch}} MN : B]\!]_{\eta\rho} = \\
&\quad \Phi_{ab}\left([\![\Gamma \vdash_{\lambda\mu\mathrm{Ch}} M : A \to B]\!]_{\eta\rho}\right)[\![\Gamma \vdash_{\lambda\mu\mathrm{Ch}} N : A]\!]_{\eta\rho}, \\
&[\![\Gamma \vdash_{\lambda\mu\mathrm{Ch}} (\lambda x : A.M) : A \to B]\!]_{\eta\rho} = \\
&\quad \Phi_{ab}^{-1}(f),
\end{aligned}$$

where f is the function defined by the assignment:

$$f(d) = [\![\Gamma, x : A \vdash_{\lambda\mu\mathrm{Ch}} M : B]\!]_{\eta\rho[x \mapsto d]}$$

for any $d \in \mathcal{M}(\mathcal{I}[\![A]\!]_\eta)$. Observe that f does indeed belong to the set $[\mathcal{M}(a) \to \mathcal{M}(b)]$, for $a = \mathcal{I}[\![A]\!]_\eta$ and $b = \mathcal{I}[\![B]\!]_\eta$.
Let $a = \mathcal{I}[\![\mu\alpha.A]\!]_\eta$ and $b = \mathcal{I}[\![A[\alpha := \mu\alpha.A]]\!]_\eta$, then:

$$\begin{aligned}
&[\![\Gamma \vdash_{\lambda\mu\mathrm{Ch}} \mathit{fold}_{\mu\alpha.A}(M) : \mu\alpha.A]\!]_{\eta\rho} = \\
&\quad \Psi_{ab}^{-1}\left([\![\Gamma \vdash_{\lambda\mu\mathrm{Ch}} M : A[\alpha := \mu\alpha.A]]\!]_{\eta\rho}\right) \\
&[\![\Gamma \vdash_{\lambda\mu\mathrm{Ch}} \mathit{unfold}_{\mu\alpha.A}(M) : A[\alpha := \mu\alpha.A]]\!]_{\eta\rho} = \\
&\quad \Psi_{ab}\left([\![\Gamma \vdash_{\lambda\mu\mathrm{Ch}} M : \mu\alpha.A]\!]_{\eta\rho}\right)
\end{aligned}$$

Chapter 12

Applications 21.2.2008:897

12.1. Recursive types in programming languages

1. A short discussion of the recursive modes of ALGOL-68, that seems to have been the first programming language including recursive definitions of types;

2. A discussion of recursive types (especially algebraic types) in modern functional languages;

3. Mention of the extension of the standard type-inference algorithm for ML-like languages to deal with recursive (= infinite) types in the section on type-checking in Aho, Sethi and Ullman.

12.2. Subtyping

The model of recursive types discussed in the previous Chapter justifies thinking of types as subsets of a model for the untyped λ-calculus (possibly extended with constants). This interpretation suggests a straightforward notion of *subtyping*: recursive types A, B are in the subtype relation, written $A \leqslant B$, if $\mathcal{I}[\![A]\!]_\eta \subseteq \mathcal{I}[\![B]\!]_\eta$ for all type environments η interpreting the free type variables occurring in them as complete and uniform subsets of D. We shall see later that a natural formal system for deriving subtyping judgements for recursive types is sound and complete for this interpretation. We start by discussing some motivations for introducing a subtype relation along with recursive types, mainly drawn from the theory of object-oriented programming. Then we introduce some basic systems of subtyping for simple types, showing by way of examples how assuming that some types are included in others can achieve the same effects as having recursive types. Finally, we discuss in some more detail the formal system of Brandt & Henglein for axiomatizing the subtyping relation on recursive types introduced by Amadio & Cardelli, which is by now standard.

Foundations of object-oriented programming

Recursive types have been widely used in the theoretical study of object-oriented programming, especially in the many proposals that have been made

to encode objects as terms of λ-calculi extended with record structures. There is a huge literature on this subject, originating with the work of Cardelli; we just discuss a few motivating examples.

Record types and objects

12.2.1. DEFINITION. The set $\mathbb{T}_\mu = \mathbb{T}_\mu(\mathbb{A})$ is extended with *record types* $\{\ell_1 : A_1, \ldots, \ell_n : A_n\}$, and is defined by the following abstract syntax:

$$\boxed{\mathbb{T}_\mu \quad = \quad \mathbb{A} \mid \mathbb{T}_\mu \to \mathbb{T}_\mu \mid \mu\mathbb{A}.\mathbb{T}_\mu \mid \{\ell_1 : \mathbb{T}_\mu, \ldots, \ell_n : \mathbb{T}_\mu\}}$$

where $\ell_1, \ldots, \ell_n$ are distinct *labels*, and we assume that $\mathbb{A}$ contains atoms for base types like $\mathbf{nat}, \mathbf{int}, \mathbf{bool}, \ldots$.

Correspondingly, the set of terms is extended with *record structures* of the form $\{\ell_1 = M_1, \ldots \ell_n = M_n\}$ where $M_1, \ldots, M_n$ are terms, and constants of base types.

Record structures are tuples whose components are accessed via their labels: if $\{\ell_1 = M_1, \ldots \ell_n = M_n\}$ is such a structure, then $\{\ell_1 = M_1, \ldots \ell_n = M_n\}.\ell_i$ has the same meaning as M_i.

Recursive record structures can be interpreted as *objects*, along the lines of the work of Cardelli. For example, we may have a point object in plane defined by:

$$\mathsf{Y}\left(\lambda s.\{\mathsf{x} = 1.0, \mathsf{y} = 2.0, \mathsf{dist} = \lambda p.\sqrt{(p.\mathsf{x} - s.\mathsf{x})^2 + (p.\mathsf{y} - s.\mathsf{y})^2}\}\right)$$

with coordinates of type **real**, x and y, and a method dist for calculating its distance from another point. Observe that this point has type

$$\mathsf{Point} \equiv \{\mathsf{x} : \mathbf{real}, \mathsf{y} : \mathbf{real}, \mathsf{dist} : \mathsf{Point} \to \mathbf{real}\},$$

a recursive record type.

Object calculus and object types

This will deal (very sketchily) with the ς-calculus of Abadi and Cardelli in their book *A Theory of Objects*.

Pure subtyping

Consider the set of simple types generated from type variables (ranged over by $s, t, \ldots$) and an unspecified set of type constants like $\mathbf{nat}, \mathbf{int}, \mathbf{bool}, \ldots$, ranged over by κ, described by the syntax:

$$A ::= \kappa \mid t \mid A' \to A''.$$

12.2.2. DEFINITION (Subtyping judgements). 1. A *subtyping judgement* has the form $A \leqslant B$, where A, B are types. A subtyping judgement is *atomic* whenever A, B are type variables or type constants. A subtyping judgement is *inflationary* if it has the form $t \leqslant A$, *deflationary* if it has the

form $A \leqslant t$; in the last two cases, the type variable t is called a *subject* of the judgement.

2. A set $\mathcal{C}$ of subtyping judgements is *atomic* if it consists of atomic subtyping judgements. It is *inflationary* (resp. deflationary) when it consists only of inflationary (resp. deflationary) subtyping judgements.

3. A set $\mathcal{C}$ of subtyping judgements is *homogeneous* in a type variable t if the subtyping judgements of which t is subject are all inflationary or they are all deflationary.

12.2.3. Definition (Entaiment of subtyping judgements). We define the following proof system for proving entaiments of the form

$$\mathcal{C} \vdash A \leqslant B$$

where $\mathcal{C}$ is a finite set of subtyping judgements:

$$\mathcal{C}, A \leqslant B \vdash A \leqslant B \quad \text{(proj)} \qquad \mathcal{C} \vdash A \leqslant A \quad \text{(refl)}$$

$$\frac{\mathcal{C} \vdash A \leqslant A' \quad \mathcal{C} \vdash A' \leqslant A''}{\mathcal{C} \vdash A \leqslant A''} \ \text{(trans)}$$

$$\frac{\mathcal{C} \vdash B' \leqslant A' \quad \mathcal{C} \vdash A'' \leqslant B''}{\mathcal{C} \vdash A' \to A'' \leqslant B' \to B''} \ \text{(arrow)}$$

The rules for assigning types to λ-terms have to take into account also the assumptions concerning the subtyping judgements. They derive judgements of the form:

$$\mathcal{C}, \Gamma \vdash M : A$$

where Γ is a basis, and there is the further rule:

$$\frac{\mathcal{C}, \Gamma \vdash M : A \quad \mathcal{C} \vdash A \leqslant B}{\mathcal{C}, \Gamma \vdash M : B} \ \text{(sub)}$$

12.2.4. Example. (i) We can type self-application by an essential use of rule (sub):

$$\{s \leqslant s \to t\}, \emptyset \vdash \lambda x.xx : s \to t$$

Observe that $\{s \leqslant s \to t\}$ is inflationary, hence homogeneous because it consists of one subtyping judgement.
(ii) Subtyping judgements can have the same effect as simultaneous recursions:

$$\{s \leqslant s \to s, s \to s \leqslant s\}, \emptyset \vdash (\lambda x.xx)(\lambda x.xx) : s$$

Actually, the set $\{s \leqslant s \to s, s \to s \leqslant s\}$ allows to type every pure λ-term, and in this sense is equivalent to the simultaneous recursion $\{s = s \to s\}$. In this case, we note that the set of subtyping judgements is not homogeneous.

We can prove the following relation between typability and normalization properties of λ-terms:

12.2.5. PROPOSITION. *1. Assume that $\mathcal{C}, \Gamma \vdash M : A$. If $\mathcal{C}$ is homogeneous in every type variable which occurs as subject, then M is stronly normalizing.*

2. If the term N is a normal form, then there is a homogeneous, inflationary set $\mathcal{C}$ of subtyping judgements and a basis Γ such that $\mathcal{C}, \Gamma \vdash N : A$ for some type A.

Subtyping recursive types

We discuss the system of Brandt & Henglein for subtyping recursive types. Types are defined by the grammar:

$$A ::= \mathbf{nat} \mid \mathbf{int} \mid t \mid A' \to A'' \mid \mu t.(A' \to A'')$$

Subtyping among recursive types is defined by the system of Brandt & Henglein, with rules:

$$\Gamma \vdash \mathbf{nat} \leqslant \mathbf{int} \quad \text{(const)} \quad \Gamma \vdash t \leqslant t \quad \text{(var)}$$

$$\Gamma \vdash \mu t.A \leqslant A[t := \mu t.A] \quad \text{(unfold)}$$

$$\Gamma \vdash A[t := \mu t.A] \leqslant \mu t.A \quad \text{(fold)}$$

$$\Gamma, A \leqslant B, \Gamma' \vdash A \leqslant B \quad \text{(hyp)}$$

$$\frac{\Gamma, A \to B \leqslant A' \to B' \vdash A' \leqslant A \quad \Gamma, A \to B \leqslant A' \to B' \vdash B \leqslant B'}{\Gamma \vdash A \to B \leqslant A' \to B'} \quad \text{(Arrow/Fix)}$$

where Γ is a set of subtyping assumptions of the form $A_1 \leqslant B_1, \ldots, A_n \leqslant B_n$.

We follow Brandt & Henglein, and Amadio & Cardelli, and introduce also *partial types*, ranged over by $S, T, \ldots$ and inductively defined by the grammar:

$$T ::= \mho \mid \Omega \mid \mathbf{nat} \mid \mathbf{int} \mid T' \to T''$$

The subtyping relation on partial types is defined by

$$\mho \leqslant_{fin} T \qquad T \leqslant_{fin} \Omega \qquad \mathbf{nat} \leqslant_{fin} \mathbf{int}$$

$$\frac{T \leqslant_{fin} T' \quad T' \leqslant_{fin} T''}{T \leqslant_{fin} T''} \qquad \frac{S' \leqslant_{fin} S \quad T \leqslant_{fin} T'}{S \to T \leqslant_{fin} S' \to T'}$$

For a recursive type A, the tree obtained by infinite unfolding of A is denoted by A^*. The Amadio & Cardelli subtyping relation is defined by

$$A \leqslant_{AC} B \Leftrightarrow \forall n \in \omega.(A^*) \downarrow n \leqslant_{fin} (B^*) \downarrow n.$$

where the *lower approximations* $\cdot \downarrow n$ and the *upper approximations* $\cdot \uparrow n$, for all $n \in \omega$, are defined on infinite trees and yield partial types:

12.2.6. DEFINITION (Approximations of an infinite tree). Let τ be an infinite tree, its lower and upper approximations are defined simultaneously as follows, for any $n \in \omega$:

$$\begin{array}{rcl} \tau \downarrow 0 & = & \mho \\ (\tau' \to \tau'') \downarrow (n+1) & = & \tau' \uparrow n \to \tau'' \downarrow n \\ \mho \downarrow (n+1) & = & \mho \\ \Omega \downarrow (n+1) & = & \Omega \\ t \downarrow (n+1) & = & t \end{array} \qquad \begin{array}{rcl} \tau \uparrow 0 & = & \Omega \\ (\tau' \to \tau'') \uparrow (n+1) & = & \tau' \downarrow n \to \tau'' \uparrow n \\ \mho \uparrow (n+1) & = & \mho \\ \Omega \uparrow (n+1) & = & \Omega \\ t \uparrow (n+1) & = & t \end{array}$$

Soundness and completeness

Let D be a domain that solves the equation:

$$D \cong \mathbb{Z}_\perp + [D \to D]_\perp + \{?\}_\perp$$

where $\mathbb{Z}_\perp$ is the flat cpo of integers, $(\cdot)_\perp$ denotes lifting and $+$ is coalesced sum. As in the previous Chapter, types will be interpreted as complete and uniform subsets of D not containing the error element ?. We shall assume that D has been obtained as an inverse limit, hence it is endowed with a notion of approximation, with essentially the same properties as in Definition 11.1.3. We can also modify the definition of application, taking into account the fact that now not all elements are functions; for all $d, e \in D$ we set:

$$d \cdot e = \begin{cases} \perp_D & \text{if } d \in \mathbb{Z}_\perp \\ d(e) & \text{if } d \in [D \to D] \\ ? & \text{if } d = ?. \end{cases}$$

12.2.7. Definition. For $X, Y \in \mathcal{CU}$, define

$$X \to Y =_{\text{def}} \{d \in [D \to D] \mid \forall e \in X. d \cdot e \in Y\} \quad (12.1)$$

$$X \to^{n+1} Y =_{\text{def}} \{d \in ([D \to D])_{n+1} \mid \forall e \in X. d \cdot e \in Y\} \quad (12.2)$$

12.2.8. Proposition. *1. Let $X, Y \in \mathcal{CU}$. Then $X \to Y$ and $X_n \to^{n+1} Y_n$ are also complete and uniform.*

2. If X, Y are ideals of D, then also $X \to Y$ is an ideal of D. If Y, Y are ideals of D_n, then $X \to^{n+1} Y$ is an ideal of D_{n+1}.

The following Lemma states a useful fact concerning the inclusion of complete and uniform subsets of D:

12.2.9. Lemma. *For X, Y complete and uniform subsets of D, $X \subseteq Y$ if and only if $X_n \subseteq Y_n$ for all $n \in \omega$.*

Proof. If $X \subseteq Y$ and $d \in X_n$, then $d \in X$, so $d \in Y$ and $d \in Y_n$ because $d = d_n$. Conversely, if $d \in X$, then for all $n \in \omega$ $d_n \in X_n \subseteq Y_n \subseteq Y$, so $d = \bigsqcup_{n \in \omega} d_n \in Y$. ■

As in the previous Chapter, we define the interpretation of types following the approximation structure of D.

12.2.10. Definition (Approximate Interpretations of Types). For all $n \in \omega$, any type $A \in \mathbb{T}_\mu$ and any type environment $\eta : V \to \mathcal{CU}$, define $\mathcal{I}^n[\![A]\!]_\eta$ as follows, by induction on n and the complexity of the type A:

- $\mathcal{I}^0[\![A]\!]_\eta = \{\perp_D\}$;
- $\mathcal{I}^{n+1}[\![\mathbf{int}]\!]_\eta = \mathbb{Z}_\perp$;
- $\mathcal{I}^{n+1}[\![\mathbf{nat}]\!]_\eta = \{n \in \mathbb{Z}_\perp \mid n \in \omega \vee n = \perp\}$;

- $\mathcal{I}^{n+1}[\![?]\!]_\eta = ?$;
- $\mathcal{I}^{n+1}[\![t]\!]_\eta = (\eta(t))_{n+1}$;
- $\mathcal{I}^{n+1}[\![A_1 \to A_2]\!]_\eta = \mathcal{I}^n[\![A_1]\!]_\eta \to^{n+1} \mathcal{I}^n[\![A_2]\!]_\eta$;
- $\mathcal{I}^{n+1}[\![\mu t.(A' \to A'')]\!]_\eta = \mathcal{I}^{n+1}[\![A' \to A'']\!]_{\eta[t \mapsto \mathcal{I}^n[\![\mu t.(A' \to A'')]\!]_\eta]}$.

The following series of Lemmas are left as exercises for the reader: they are straightforward variations of the corresponding lemmas in the previous Chapter.

12.2.11. Lemma. *For any $n \in \omega$, all types A and any type environment η:*

1. $\mathcal{I}^n[\![A]\!]_\eta \subseteq \mathcal{I}^{n+1}[\![A]\!]_\eta$,
2. $\mathcal{I}^n[\![A]\!]_\eta = (\mathcal{I}^{n+1}[\![A]\!]_\eta)_n$.

12.2.12. Definition (Type Interpretations). For every type A and type environment η,

$$\mathcal{I}[\![A]\!]_\eta =_{\text{def}} \{d \in D \mid \forall n \in \omega. d_n \in \mathcal{I}^n[\![A]\!]_\eta\}.$$

12.2.13. Proposition. *(i)* $\forall n \in \omega. \mathcal{I}^n[\![A]\!]_\eta \subseteq \mathcal{I}[\![A]\!]_\eta$.

(ii) For any type A and environment η, $\mathcal{I}[\![A]\!]_\eta$ is a complete and uniform subset of D. It is an ideal if η maps type variables to ideals of D.

This Proposition entails that, for any type A and environment η,

$$(\mathcal{I}[\![A]\!]_\eta)_n = \mathcal{I}^n[\![A]\!]_\eta,$$

for all $n \in \omega$.

12.2.14. Lemma. *For any type A and environment η:*

$$\mathcal{I}[\![\mu t.A]\!]_\eta = \mathcal{I}[\![A]\!]_{\eta[t \mapsto \mathcal{I}[\![\mu t.A]\!]_\eta]}$$

12.2.15. Lemma. *For any pair of types A, B, any type variable t and any environment η,*

$$\mathcal{I}[\![A[t := B]]\!]_\eta = \mathcal{I}[\![A]\!]_{\eta[t \mapsto \mathcal{I}[\![B]\!]_\eta]}.$$

12.2.16. Theorem (Properties of Type Interpretations). *The following conditions are satisfied, for any type environment η and all types A, B:*

1. $\mathcal{I}[\![t]\!]_\eta = \eta(t)$
2. $\mathcal{I}[\![A \to B]\!]_\eta = \mathcal{I}[\![A]\!]_\eta \to \mathcal{I}[\![B]\!]_\eta$
3. $\mathcal{I}[\![\mu t.A]\!]_\eta = \mathcal{I}[\![A[t := \mu t.A]]\!]_\eta$

Soundness We follow Brandt & Henglein and define the notion of satisfaction for subtyping judgements exploiting the stratification of D into levels:

12.2.17. Definition (Satisfaction). Let η be a type environment, and $k \in \omega$:

1. $\eta \models_k A \leqslant B$ if and only if $\mathcal{I}^k[\![A]\!]_\eta \subseteq \mathcal{I}^k[\![B]\!]_\eta$,
2. $\eta \models_k \Gamma$ if and only if $\eta \models_k A \leqslant B$ for all judgements $A \leqslant B$ in Γ,
3. $\Gamma \models_k A \leqslant B$ if and only if, for all η such that $\eta \models_k \Gamma$, $\eta \models_k A \leqslant B$,
4. $\Gamma \models A \leqslant B$ if and only if, for all $k \in \omega$, $\Gamma \models_k A \leqslant B$.

12.2.18. Lemma. $\Gamma \vdash A \leqslant B$ *implies* $\Gamma \models A \leqslant B$

Proof. By induction on the length of the prof that $\Gamma \vdash A \leqslant B$. This is clear if the last rule applied is (const), (hyp), (fold) of (unfold): for the last two cases use Lemma 12.2.15. Assume that the last rule applied is (Arrox/Fix), then $A \equiv A' \to A''$ and $B \equiv B' \to B''$ and the last inference has the shape:

$$\frac{\Gamma, A \to B \leqslant A' \to B' \vdash A' \leqslant A \quad \Gamma, A \to B \leqslant A' \to B' \vdash B \leqslant B'}{\Gamma \vdash A \to B \leqslant A' \to B'}.$$

By induction hypothesis

$$\Gamma, A \to B \leqslant A' \to B' \models A' \leqslant A$$

$$\Gamma, A \to B \leqslant A' \to B' \models B \leqslant B'$$

and we have to show that, for all $k \in \omega$,

$$\Gamma \models_k A \to B \leqslant A' \to B'.$$

We do this by induction on k. The base case is obvious, as both sides are interpreted as $\{\bot_D\}$. For the induction step, assume that $\Gamma \models_k A \to B \leqslant A' \to B'$ and assume also that $\eta \models_{k+1} \Gamma$. We have to show that $\mathcal{I}^{k+1}[\![A' \to A'']\!]_\eta \subseteq \mathcal{I}^{k+1}[\![B' \to B'']\!]_\eta$. By induction hypothesis (on k) we have $\eta \models_k \Gamma, A \to B \leqslant A' \to B'$, because if $\eta \models_{k+1} \Gamma$, then also $\eta \models_k \Gamma$ by the properties of the approximate interpretation of types and Lemma 12.2.9. hence, by hypothesis of the main induction, $\mathcal{I}^k[\![B']\!]_\eta \subseteq \mathcal{I}^k[\![A']\!]_\eta$ and $\mathcal{I}^k[\![A'']\!]_\eta \subseteq \mathcal{I}^k[\![B'']\!]_\eta$. Now assume $d \in \mathcal{I}^{k+1}[\![A' \to A'']\!]_\eta = \mathcal{I}^k[\![A']\!]_\eta \to^{n+1} \mathcal{I}^k[\![A'']\!]_\eta$, and let $a \in \mathcal{I}^k[\![B']\!]_\eta$. Then $d \cdot a \in \mathcal{I}^k[\![B'']\!]_\eta$, hence $d \in \mathcal{I}^{k+1}[\![B' \to B'']\!]_\eta$, which concludes the proof. ■

12.2.19. Corollary (Soundness). $\vdash A \leqslant B$ *implies* $\models A \leqslant B$

Completeness For proving completeness, we exploit some lemmas proved in the paper of Brandt & Henglein.

12.2.20. Definition. A *simulation* on recursive types is a binary relation $\mathcal{R}$ that satisfies the following properties:

1. $(A' \to A'')\ \mathcal{R}\ (B' \to B'')$ implies $B'\ \mathcal{R}\ A'$ and $A''\ \mathcal{R}\ B''$,

2. $\mu t.A \mathcal{R} B$ implies $A[t := \mu t.A] \mathcal{R} B$,

3. $A \mathcal{R} \mu t.B$ implies $A \mathcal{R} B[t := \mu t.B]$,

4. $A \mathcal{R} B$ implies $\mathcal{L}(A) \leqslant \mathcal{L}(B)$

where $\mathcal{L}(A)$ is the root symbol of the tree A^* and the ordering on labels is defined as:

$$\mho \leqslant \mathbf{nat} \leqslant \mathbf{int} \leqslant \Omega$$

$$\mho \leqslant \rightarrow \leqslant \Omega \qquad \mho \leqslant t \leqslant \Omega$$

12.2.21. LEMMA. *If X', X'', Y', Y'' are ideals of D, then*

$$X' \to X'' \subseteq Y' \to Y'' \quad \textit{implies} \quad Y' \subseteq X', X'' \subseteq Y''.$$

PROOF. Assume $X' \to X'' \subseteq Y' \to Y''$ but $Y' \not\subseteq X'$. Then there is a finite element d of D such that $d \in Y' \setminus X'$. The step function $d \Rightarrow ?$ belongs to $X' \to X''$ because no $x \sqsupseteq d$ is in X', but not to $Y' \to Y''$, against the assumption, hence $Y' \subseteq X'$.

Assume now that $X' \to X'' \subseteq Y' \to Y''$ but $X'' \not\subseteq Y''$. Then there is $e \in X'' \setminus Y''$ and the constant funtion $\bot_D \Rightarrow e$ belongs to $X' \to X''$ but not to $Y' \to Y''$, against the assumption, hence $X'' \subseteq Y''$. ■

12.2.22. COROLLARY. *The relation on recursive types defined by*

$$A \mathcal{R} B \quad \textit{iff} \quad \mathcal{I}[\![A]\!]_\eta \subseteq \mathcal{I}[\![A]\!]_\eta$$

for all type environments η that interpret type variables as ideals, is a simulation.

PROOF. This follows easily from Lemma 12.2.21 and inspection of the structure of D. ■

12.2.23. THEOREM. *Let $A, B \in \mathbb{T}_\mu$. If $\models A \leqslant B$ then $\vdash A \leqslant B$.*

PROOF. By assumption, for all η we have $\mathcal{I}^k[\![A]\!]_\eta \subseteq \mathcal{I}^k[\![A]\!]_\eta$ for all $k \in \omega$ and all η, hence $\mathcal{I}[\![A]\!]_\eta \subseteq \mathcal{I}[\![A]\!]_\eta$ for all η and a fortiori for all environments η that interpret type variables as ideals. Hence the assumption implies $A \mathcal{R} B$ which implies $A \leqslant_{AC} B$ by [Brandt & Henglein, Lemma 2.2] which finally implies $\vdash A \leqslant B$, by [Brandt & Henglein, Theorem 2.6]. ■

We are looking for a more direct proof of the completeness theorem which constructs a derivation in the Brandt & Henglein system from the assumption that $\models A \subseteq B$, exploiting the structure of the domain D and the definition of the type interpretation.

Chapter 13

Further readings 21.2.2008:897

Historical background The origins of recursive types can be traced back to the basic early developments in theoretical computer science. On the semantical side, they appeared in the special form of recursive definitions of sets. For example, **?** discussed the set S of *sequences* of elements of a set A defined recursively by $S = 1 + (A \times S)$ (to be read: "a sequence is either empty, or is a pair whose first component is an A and whose second component is a sequence"). McCarthy also observed that this definition implies that $1 = S - (A \times S) = S \times (1 - A)$, hence $S = 1/(1-A)$, whose expansion gives $S = 1 + A + A^2 + A^3 + \cdots$, which describes a sequence a either empty, or consisting of two elements of A, or consisting of three elements of A,... (The justification of such calculations requires a deep understanding of algebraic facts — see **?** for a discussion, with applications to deciding isomorphisms of recursive types) Of course, the various categories of domains and the methods of solving recursive domain equations over them (**?**) can fairly be regarded as comprehensive semantical universes for recursive types, a fact that has been exploited in §11.2.

On the syntactical side, recursive types appeared in J.H. Morris' thesis **?**, who considered the possibility of allowing, in the simply typed λ-calculus, "circular" type expressions like the solution of the equation $X = X \to \alpha$. At about the same time, a form of recursive type definition was introduced in ALGOL 68 as "recursive mode declarations", e.g.

$$\textbf{mode}\,\texttt{cell} = \textbf{struct}(\textbf{ref}\,\texttt{cell}\,\texttt{next}, \textbf{int}\,\texttt{item})$$

Also in this case, equality of recursive modes can be checked by reducing the problem to that of deciding the equality of infinite regular trees **?** or, equivalently, of components of solutions of finite systems of equations (**?**); the latter method is thoroughly investigated in **?**.

In a different direction, typed functional programming languages in the tradition of ML **?**, including Miranda and Haskell, allow recursive definitions of data-types which the type-checker exploits in inferring type schemes for programs, see **?**. A typical example of such definitions, say, in Haskell **?**, is

```
data Tree a = Leaf a | Branch (Tree a) (Tree a)
```

which declares `Tree` as a (polymorphic) *type* constructor, and `Leaf` and `Branch` as *data* constructors, so that `Leaf` has type `a -> Tree a` and `Branch` has type

`Tree a -> (Tree a -> Tree a)`; an element of this type is a binary tree with leaves of type `a`, where `a` is any type, see **?**.

Type inference and equivalence of recursive types Recursive type equations like those considered by Morris arise naturally when omitting the "occur check" in the unification procedure invoked by the type inference algorithm of **?**, as remarked already in **?** (see **?** for a textbook treatment). The solutions of these equations are best regarded as infinite expressions, equivalently infinite trees. They are always *regular* (or *rational*) trees **?**, a fact that yields efficient decision procedures for checking their equality. This view of recursive types as infinite trees has then been exploited systematically in **?**, and is at the basis of this whole Part.

More recent approaches exploit the fact that the set of finite and infinite trees (over a first-order signature) is the final coalgebra of a polynomial endofuntor over the category of sets canonically associated with the signature, as in Proposition **??**. This is a well-known fact in the theory of coalgebras **?**, and has been exploited in **?**. The view of flat simultaneous recursions as coalgebras of the same functor Remark **??** has been suggested by an analogy with non-well-founded sets **?**; see **?** for an application to decidability of strong equivalence of simultaneous recursions.

The notion of type algebra (Definition **??**) was anticipated in **?** expanding on a remark of **?**; it was taken up in **?** as an alternative to the presentation of recursive types via the μ-operator. The notion of simultaneous recursion and the property of invertibility are introduced in that paper, which also sets a framework that has been used as a unifying theme throughout this Part.

Presentations of the explicitly typed systems In §8.1 we have mentioned two presentations of the explicitly type systems with recursive types. One of these (Definition 8.1.16) identifies equivalent types, and therefore one term can have infinitely many types (though all equivalent to each other) via rule (equiv). Such presentations have been called *equi-recursive* in the literature **?**, and are more interesting both from the practical and the theoretical point of view, especially when designing type checking algorithms for them. The other formulation (Definition **??**) is classified as *iso-recursive*, due to the presence of explicit coercions from a recursive type to its unfolding and conversely. We have not pursued this matter, but refer the reader to what appears to be, to our knowledge, the only study of this issue, in the context of a call-by-value formulation of Plotkin's system FPC **?**.

Models The first model for the implicitly typed systems is given in **?**, where types are interpreted as ideals over a Scott domain of the form $D \cong V + [D \to D]$. Recursive types are restricted to those of the form $\mu t.A$ where A is contractive in t, and are then interpreted as the unique fixed point of the induced contractive mapping over the complete metric spaces of ideals, whose existence is guaranteed by a well-known result of Banach.

The interpetation of recursive types that we have described in §11.1, exploiting the approximation structure of domains like D above induced by their construction as inverse limits according to the technique of Scott **?**, stems from **?**, where also the completeness theorems 11.1.39 and 11.1.40 are proved. The technique described in this paper has been shown to extend to model constructions for various extensions of simple recursive types: on the one hand it applies to many first-order type constructors without the contractiveness requirement, as in **?**. On the other hand, it can be extended to second-order polymorphic types **?** and even to bounded polymorphic types with subtyping **?**. Orthogonally to all these applications, it is possible to adapt the constructions to the explicitly typed systems, by performing them over (complete and uniform) partial equivalence relations. An attempt to view this technique in a more abstract algebraic framework, relating it, as a by-product, to the use of complete metric spaces, and exploiting results on iterative algebras **?**, can be found in **?**.

Concerning the interpretation of the explicitly typed systems, the domain models presented in §11.2 belong to the folklore of the subject; also the genericity lemma 11.2.10 is, to our knowledge, new. The use of closures instead of, say, finitary projections as done in **?** in their construction of a model for the polymorphic λ-calculus, stems from the observation that the interpretation of the type $\mu t.t \to t$ as a finitary projection is $\bot$, the same phenomenon arising in the domain interpretation.

Part III

Intersection types $\lambda^{\mathcal{S}}_{\cap}$ 21.2.2008:897

In a nutshell the intersection type systems considered in this Part form a class of type assignment systems for untyped λ-calculus, extending Curry's *basic functionality* with a new type constructor, *intersection*. This simple move makes it possible to express naturally and in a finitary way many *operational and denotational* properties of terms.

Intersection types have been originally introduced as a language for describing and capturing properties of λ-terms, which had escaped all previous typing disciplines. For instance, they were used in order to give the first type theoretic characterization of *strongly normalizing* terms, and later of *normalizing terms*.

It was realized early on that intersection types also had a distinctive semantical flavour: they express at a syntactical level the fact that a term belongs to suitable compact open sets in a Scott domain. Building on this intuition, intersection types were used in Barendregt et al. [1983] to introduce filter models and give a proof of the completeness of the natural semantics of simple type assignment systems in applicative structures suggested in Scott [1972].

Since then, intersection types have been used as a powerful tool both for the analysis and the synthesis of λ-models. On the one hand, intersection type disciplines provide finitary inductive definitions of interpretation of λ-terms in models. On the other hand, they are suggestive for the shape the domain model has to have in order to exhibit certain properties.

Intersection types can be viewed also as a restriction of the domain theory in logical form, see Abramsky [1991], to the special case of modeling pure lambda calculus by means of ω-algebraic complete lattices. Many properties of these models can be proved using this paradigm, which goes back to Stone duality.

Type assignment using intersection types can be parametrized by intersection type theories or intersection type structures. The various type theories (and corresponding type structures) are introduced together in order to give reasonably uniform proofs of their properties as well of those of the corresponding type assignment systems and filter models.

In the present Part III of this book both these syntactic and semantic aspects will be exploited.

Chapter 14

An Exemplary System

21.2.2008:897

[overview3]

There are several systems that assign intersection types to untyped lambda terms. These will be collectively denoted by $\lambda_\cap$. In this section we consider one particular system of this family, $\lambda_\cap^{\mathrm{BCD}}$ in order to outline the concepts and related properties. Definitions and the statement of theorems will be given, but no proofs. These can be found in the next chapters of Part III.

One motivation for the system presented comes from trying to modify the system $\lambda_\to$ in such a way that not only subject reduction, but also subject expansion holds. The problem of subject expansion is the following. Suppose $\vdash_{\lambda_\to} M : A$ and that $M' \twoheadrightarrow_{\beta\boldsymbol{\eta}} M$. Does one have $\vdash_{\lambda_\to} M' : A$? Let us focus on one β-step. So let $M \equiv (\lambda x.P)Q$ be a redex and suppose

$$\vdash_{\lambda_\to} P[x := Q] : A. \tag{1}$$

Do we have $\vdash_{\lambda_\to} (\lambda x.P)Q : A$? It is tempting to reason as follows. By assumption (1) also Q must have a type, say B. Then $(\lambda x.P)$ has a type $B{\to}A$ and therefore $\vdash_{\lambda_\to} (\lambda x.P)Q : A$. The mistake is that in (1) there may be several occurrences of Q, say $Q_1 \equiv Q_2 \equiv \ldots \equiv Q_n$, having as types respectively $B_1, \ldots, B_n$. It may be impossible to find a single type for all the occurrences of Q and this prevents us from finding a type for the redex. For example

$$\begin{aligned} \vdash_{\lambda_\to} (\lambda x.\mathsf{I}(\mathsf{K}x)(\mathsf{I}x)) &: A{\to}A, \\ \not\vdash_{\lambda_\to} (\lambda xy.x(\mathsf{K}y)(xy))\mathsf{I} &: A{\to}A. \end{aligned}$$

The system introduced in this chapter with intersection types assigned to untyped lambda terms remedies the situation. The idea is that if the several occurrences of Q have to have different types $B_1, \ldots, B_n$, we give them all of these types:

$$\vdash Q : B_1 \cap \ldots \cap B_n,$$

implying that for all i one has $Q : B_i$. Then we will get

$$\begin{aligned} \vdash (\lambda x.P) &: B_1 \cap \ldots \cap B_n{\to}A \qquad \text{and} \\ \vdash ((\lambda x.P)Q) &: A. \end{aligned}$$

There is, however, a second problem. In the $\lambda\mathsf{K}$-calculus, with its terms $\lambda x.P$ such that $x \notin \mathrm{FV}(P)$ there is the extra problem that Q may not be typable at all, as it may not occur in $P[x := Q]$! This is remedied by allowing $B_1 \cap \ldots \cap B_n$ also for $n = 0$ and writing this type as $\mathtt{U}$, to be considered as the universal type, i.e. assigned to all terms. Then in case $x \notin \mathrm{FV}(P)$ one has

$$\begin{array}{rcll} \vdash Q & : & \mathtt{U} & \\ \vdash (\lambda x.P) & : & \mathtt{U}{\to}A & \text{and} \\ \vdash ((\lambda x.P)Q) & : & A. & \end{array}$$

This is the motivation to introduce a $\leq$ relation on types with largest element $\mathtt{U}$ and intersections such that $A \cap B \leq A, A \cap B \leq B$ and the extension of the type assignment by the sub-sumption rule $\Gamma \vdash M : A,\ A \leq B \ \Rightarrow\ \Gamma \vdash M : B$. It has as consequence that terms like $\lambda x.xx$ get as type $((A{\to}B) \cap A){\to}B$, while $(\lambda x.xx)(\lambda x.xx)$ only gets $\mathtt{U}$ as type. Also we have subject conversion

$$\Gamma \vdash M : A \ \&\ M =_\beta N \ \Rightarrow\ \Gamma \vdash N : A.$$

This has as consequence that one can create a so-called filter lambda model in which the meaning of a closed term consists of the collection of types it gets. In this way new lambda models will be obtained and new ways to study classical models as well.

The type assignement system $\lambda^{\mathrm{BCD}}_{\cap}$ will be introduced in Section 14.1 and the correspondig filter model in 14.2.

14.1. The type assignment system $\lambda^{\mathrm{BCD}}_{\cap}$

[lin.bcd] A typical member of the family of intersection type assignment systems is $\lambda^{\mathrm{BCD}}_{\cap}$. This system is introduced in Barendregt et al. [1983] as an extension of the initial system in Coppo and Dezani-Ciancaglini [1980].

14.1.1. Definition. (i) Define the following sets of type atoms.

$$\begin{array}{lcl} \mathbb{A}_\infty & = & \{\mathtt{c}_0, \mathtt{c}_1, \mathtt{c}_2, \ldots\} \\ \mathbb{A}^{\mathtt{U}}_\infty & = & \mathbb{A}_\infty \cup \{\mathtt{U}\}, \\ \mathbb{A}^{\mathrm{BCD}} & = & \mathbb{A}^{\mathtt{U}}_\infty, \end{array}$$

where the type atom $\mathtt{U} \notin \mathbb{A}_\infty$ is a special symbol called *universe* or *universal top*.

(ii) The *intersection type language* over $\mathbb{A}^{\mathrm{BCD}}$, denoted by $\mathbb{T} = \mathbb{T}^{\mathrm{BCD}}$ is defined by the following abstract syntax.

$$\mathbb{T} = \mathbb{A}^{\mathrm{BCD}} \mid \mathbb{T}{\to}\mathbb{T} \mid \mathbb{T} \cap \mathbb{T}.$$

Notation. (i) Greek letters $\alpha, \beta, \ldots$ will denote arbitrary atoms in $\mathbb{A}^{\mathrm{BCD}}$.

(ii) In earlier work, the universe used to be denoted by ω (Barendregt et al. [1983], Coppo et al. [1987]) or Ω (Dezani-Ciancaglini et al. [2003], Alessi et al. [2006]) and not by $\mathtt{U}$.

(iii) A, B, C, D, E range over arbitrary types in $\mathbb{T}$.

(iv) When writing intersection types we shall use the following convention: the constructor $\cap$ takes precedence over the constructor $\to$ and it associates to the right. For example

$$(A{\to}B{\to}C)\cap A{\to}B{\to}C \equiv ((A{\to}(B{\to}C))\cap A){\to}(B{\to}C).$$

14.1.2. Remark. In Part III other sets $\mathbb{T}$ of types will be formed by replacing the set $\mathbb{A}^{\mathrm{BCD}}$ of type atoms by an arbitrary set $\mathbb{A}$ (finite or countably infinite). In this Chapter, however, we take $\mathbb{A} = \mathbb{A}^{\mathrm{BCD}} = \mathbb{A}^{\mathtt{U}}_{\infty}$.

The following deductive system has as intention to introduce an appropriate pre-order on $\mathbb{T}$, compatible with the operator $\to$, such that $A\cap B$ is a greatest lower bound of A and B, for each A, B.

14.1.3. Definition (Intersection type preorder). **[ex.ineq]** The intersection type theory BCD is the set of all statements $A \leq B$ (to be read as "A is a subtype of B"), with $A, B \in \mathbb{T}$, derivable from the following axioms and rules, where $A, B, C, \ldots \in \mathbb{T}$.

(refl)	$A \leq A$
(incl_L)	$A\cap B \leq A$
(incl_R)	$A\cap B \leq B$
(glb)	$\dfrac{C\leq A \quad C\leq B}{C\leq A\cap B}$
(trans)	$\dfrac{A\leq B \quad B\leq C}{A\leq C}$
($\mathtt{U}_{\mathrm{top}}$)	$A \leq \mathtt{U}$
($\mathtt{U}{\to}$)	$\mathtt{U} \leq A{\to}\mathtt{U}$
($\to\cap$)	$(A{\to}B)\cap(A{\to}C) \leq A{\to}(B\cap C)$
($\to$)	$\dfrac{A'\leq A \quad B\leq B'}{(A{\to}B)\leq(A'{\to}B')}$

14.1.4. Notation. **[BCDeq]**

(i) For $(A \leq B) \in \mathrm{BCD}$ we write $A \leq_{\mathrm{BCD}} B$ or $\vdash_{\mathrm{BCD}} A \leq B$ (or often just $A \leq B$ if there is little danger of confusion).

(ii) Write $A{=}_{\mathrm{BCD}}B$ (or $A = B$) for $A \leq_{\mathrm{BCD}} B$ & $B \leq_{\mathrm{BCD}} A$.

(iii) We write $[\mathbb{T}]$ for the set $\mathbb{T}$ modulo $=_{\mathrm{BCD}}$. In BCD we usually work on $[\mathbb{T}]$.

(iv) We write $A \equiv B$ for syntactic identity. E.g. $A\cap B \equiv A\cap B$, but $A\cap B \not\equiv B\cap A$.

14.1.5. Remark. All systems in Part III have the first five axioms and rules of Definition 14.1.3. They differ in the extra axioms and rules and the set of atoms.

14.1.6. PROPOSITION. [**proposition:bcdcompatibility**] *The equivalence relation* $=$ *defined in Notation 14.1.4(ii) is a congruence, i.e.* $=$ *is compatible with* $\cap$ *and* $\to$.

PROOF. (i) By rules (trans), (incl_L), (incl_R) and (glb) one has

$$A = A' \;\&\; B = B' \;\Rightarrow\; (A \cap B) = (A' \cap B').$$

(ii) By rule ($\to$) one has

$$A = A' \;\&\; B = B' \;\Rightarrow\; (A \to B) = (A' \to B'). \;\blacksquare$$

14.1.7. REMARK. [**remark:bcdstructure**] The theory BCD can be seen as a structure with a pre-order

$$\text{BCD} = \langle \mathbb{T}, \leq, \cap, \to, \mathtt{U} \rangle.$$

This means that $\leq$ is reflexive and transitive, but not anti-symmetric

$$A \leq B \;\&\; B \leq A \;\not\Rightarrow\; A \equiv B.$$

One can go over to equivalence classes and define a partial order $\leq$ on $[\mathbb{T}]$ that satisfies antisymmetry.

$$[A] \leq [B] \iff A \leq B.$$

By Proposition 14.1.6, the operators $\cap$ and $\to$ can be defined on $[\mathbb{T}]$ by

$$\begin{aligned} [A] \cap [B] &= [A \cap B]; \\ [A] \to [B] &= [A \to B]. \end{aligned}$$

We obtain a type structure

$$[\text{BCD}] \;=\; \langle [\mathbb{T}], \leq, \cap, \to, [\mathtt{U}] \rangle.$$

In this structure, $[\mathtt{U}]$ is the largest element (also called *top*) and $[A] \cap [B]$ is the greatest lower bound of $[A]$ and $[B]$.

14.1.8. DEFINITION. [**ex.assign**]

(i) A *basis* is a finite set of statements of the shape $x\,B$, where $B \in \mathbb{T}$, with all variables distinct.

(ii) The type assignment system $\lambda^{\text{BCD}}_{\cap}$ for deriving statements of the form $\Gamma \vdash M : A$ with Γ a basis, $M \in \Lambda$ (the set of untyped lambda terms) and $A \in \mathbb{T}$ is defined by the following axioms and rules.

(Ax)	$\Gamma \vdash x\,A$	if $(x\,A) \in \Gamma$
($\to$I)	$\dfrac{\Gamma,\; x\,A \vdash M : B}{\Gamma \vdash (\lambda x.M) : (A \to B)}$	
($\to$E)	$\dfrac{\Gamma \vdash M : (A \to B) \quad \Gamma \vdash N : A}{\Gamma \vdash (MN) : B}$	
($\cap$I)	$\dfrac{\Gamma \vdash M : A \quad \Gamma \vdash M : B}{\Gamma \vdash M : (A \cap B)}$	
($\leq$)	$\dfrac{\Gamma \vdash M : A}{\Gamma \vdash M : B}$	if $A \leq_{\text{BCD}} B$
($\mathtt{U}$)	$\Gamma \vdash M : \mathtt{U}$	

(iii) We say that a term M is *typable* from a given basis Γ, if there is a type $A \in \mathbb{T}$ such that the judgement $\Gamma \vdash M : A$ is derivable in $\lambda^{\mathrm{BCD}}_{\cap}$. In this case we write $\Gamma \vdash^{BCD}_{\cap} M : A$ or just $\Gamma \vdash M : A$, if there is little danger of confusion.

14.1.9. Remark. All systems of type assignment in Part III have the first five axioms and rules of Definition 14.1.8.

In Proposition 14.1.11 we need the notions of admissible and derived rule.

14.1.10. Definition. Consider the rule

$$\frac{\Gamma \vdash M : A}{\Gamma' \vdash M' : A'.} \qquad (R)$$

(i) R is called *admissible* if one has

$$\Gamma \vdash M : A \;\Rightarrow\; \Gamma' \vdash M' : A'.$$

(ii) R is called *derived* if there is a derivation starting from $\Gamma \vdash M : A$ that ends in $\Gamma' \vdash M' : A'$.

A derived rule is always admissible but the converse does not hold. If

$$\frac{\Gamma \vdash M : A}{\Gamma' \vdash N : B}$$

is a derived rule, then for all Δ one has that

$$\frac{\Gamma \cup \Delta \vdash M : A}{\Gamma' \cup \Delta \vdash N : B}$$

is also derived. Hence derived rules are closed under theory extension. We will only be concerned with admissible and derived rules for theories of type assignment.

14.1.11. Proposition. **[adm-derived]**

(i) *The rules* $(\cap E)$

$$\frac{\Gamma \vdash M : (A \cap B)}{\Gamma \vdash M : A} \qquad \frac{\Gamma \vdash M : (A \cap B)}{\Gamma \vdash M : B}$$

are derived in $\lambda^{\mathrm{BCD}}_{\cap}$.

(ii) *The following rules are admissible in the intersection type assignment system* $\lambda^{\mathrm{BCD}}_{\cap}$.

$$(\textit{weakening}) \quad \frac{\Gamma \vdash M : A \quad x \notin \Gamma}{\Gamma, x\, B \vdash M : A}$$

$$(\textit{strengthening}) \quad \frac{\Gamma, x\, B \vdash M : A \quad x \notin FV(M)}{\Gamma \vdash M : A}$$

$$(\textit{cut}) \quad \frac{\Gamma, x\, B \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash (M[x := N]) : A}$$

$$(\leq\text{-L}) \quad \frac{\Gamma, x\, B \vdash M : A \quad C \leq B}{\Gamma, x\, C \vdash M : A}$$

$$(\to\text{L}) \quad \frac{\Gamma, y\, B \vdash M : A \quad \Gamma \vdash N : C \quad x \notin \Gamma}{\Gamma, x\, (C \to B) \vdash (M[y := xN]) : A}$$

$$(\cap\text{L}) \quad \frac{\Gamma, x\, A \vdash M : B}{\Gamma, x\, (A \cap C) \vdash M : B}$$

14.1.12. Theorem. *In (i) assume* $A \neq \mathtt{U}$. *Then*

(i) $\Gamma \vdash x : A \iff \exists B \in \mathbb{T}.[(x{:}B \in \Gamma \;\&\; B \leq A]$.

(ii) $\Gamma \vdash (MN) : A \iff \exists B \in \mathbb{T}.[\Gamma \vdash M : (B \to A) \;\&\; \Gamma \vdash N : B]$.

(iii) $\Gamma \vdash \lambda x.M : A \iff \exists n{>}0 \exists B_1, \ldots, B_n, C_1, \ldots, C_n \in \mathbb{T}$
$\forall i \in \{1, \ldots, n\}.[\Gamma, x\, B_i \vdash M : C_i \;\&$
$(B_1 \to C_1) \cap \ldots \cap (B_n \to C_n) \leq A]$.

(iv) $\Gamma \vdash \lambda x.M : B \to C \iff \Gamma, x\, B \vdash M : C$.

14.1.13. Definition. Let R be a notion of reduction. Consider the following rules.

$$(R\textit{-red}) \quad \frac{\Gamma \vdash M : A \quad M \to_R N}{\Gamma \vdash N : A}$$

$$(R\textit{-exp}) \quad \frac{\Gamma \vdash M : A \quad M \leftarrow_R N}{\Gamma \vdash N : A}$$

14.1.14. Proposition. *The rules* (β*-red*), (β*-exp*) *and* (η*-red*) *are admissible in* $\lambda^{\mathrm{BCD}}_{\cap}$. *The rule* ($\eta$*-exp*) *is not.*

The following result characterizes notions related to normalization in terms of type assignment in the system $\lambda^{\mathrm{BCD}}_{\cap}$. The notation $\mathtt{U} \notin A$ means that $\mathtt{U}$ does not occur in A.

14.1.15. Theorem. *Let* $M \in \Lambda^{\emptyset}$.

(i) *M has a head normal form* $\iff \exists A \in \mathbb{T}.[A \neq_{\mathrm{BCD}} \mathtt{U} \;\&\; \vdash M : A]$.

(ii) *M has a normal form* $\iff \exists A \in \mathbb{T}.[\mathtt{U} \notin A \;\&\; \vdash M : A]$.

Let M be a lambda term. For the notion 'approximant of M', see Barendregt [1984]. The approximants of a term M are roughly obtained from the Böhm tree $\mathrm{BT}(M)$ of M by removing some branches and replacing these by a new symbol $\bot$. The set of approximants of M is denoted by $\mathcal{A}(M)$. We have e.g. for the fixed-point combinator Y

$$\mathcal{A}(\mathsf{Y}) = \{\bot\} \cup \{\lambda f.f^n\bot \mid n{>}0\}.$$

Approximants are being typed by letting the typing rules be valid for approximants. For example one has

$$\begin{array}{l}
\vdash \bot : \mathtt{U}\\
\vdash \lambda f.f\bot : (\mathtt{U}{\to}A_1){\to}A_1\\
\vdash \lambda f.f(f\bot) : (\mathtt{U}{\to}A_1)\cap(A_1{\to}A_2){\to}A_2\\
\ldots\\
\vdash \lambda f.f^n\bot : (\mathtt{U}{\to}A_1)\cap(A_1{\to}A_2)\cap\ldots\cap(A_{n-1}{\to}A_n){\to}A_n\\
\ldots
\end{array}$$

The set of types of a term M coincides with the union of the sets of types of its approximants $P\in\mathcal{A}(M)$. This will give an Approximation Theorem for the filter model of next section.

14.1.16. Theorem. $\Gamma\vdash M:A \iff \exists P\in\mathcal{A}(M).\Gamma\vdash P:A$.

For example since $\lambda f.f^n\bot$ is for all n an approximant of Y, we have that all types of the shape $(\mathtt{U}{\to}A_1)\cap\ldots\cap(A_{n-1}{\to}A_n){\to}A_n$ can be derived for Y.

Finally the question whether an intersection type is inhabited is undecidable.

14.1.17. Theorem. *The set* $\{A\in\mathbb{T} \mid \exists M\in\Lambda^{\emptyset}\vdash M:A\}$ *is undecidable.*

14.2. The filter model $\mathcal{F}^{\mathrm{BCD}}$

[lin.FM]

14.2.1. Definition. **[O.compl.lat]**
(i) A *complete lattice* $(D,\sqsubseteq)$ is a partial order which has arbitrary least upper bounds (sup's) (and hence has arbitrary inf's).

(ii) A subset $Z\subseteq D$ is *directed* if $Z\neq\emptyset$ and

$$\forall x,y\in Z\exists z\in Z.x,y\sqsubseteq z.$$

(iii) An element $c\in D$ is *compact* (in the literature also called *finite*) if for each directed $Z\subseteq D$ one has

$$c\sqsubseteq\bigsqcup Z \;\Rightarrow\; \exists z\in Z.c\sqsubseteq z.$$

Let $\mathcal{K}(D)$ denote the set of compact elements of D.

(iv) A complete lattice is *ω-algebraic* if $\mathcal{K}(D)$ is countable, and for each $d\in D$, the set $\mathcal{K}(d)=\{c\in\mathcal{K}(D)\mid c\sqsubseteq d\}$ is directed and $d=\bigsqcup\mathcal{K}(d)$.

(v) If $D, \mathcal{E}$ are ω-algebraic complete lattices and $f : D \to E$, we say that f is *Scott continuous* (or simply continuous), if for any $Z \subseteq D$ directed, we have

$$f(\bigsqcup Z) = \bigsqcup f(Z).$$

The set $[D{\to}\mathcal{E}]$ consists of the continuous maps from D to $\mathcal{E}$ and can be ordered pointwise

$$f \sqsubseteq g \iff \forall d \in D. f(d) \sqsubseteq g(d).$$

Then $\langle [D{\to}\mathcal{E}], \sqsubseteq \rangle$ is again an ω-algebraic lattice.

(vi) The category **ALG** is the category whose objects are the ω-algebraic complete lattices and whose morphisms are the (Scott) continuous functions.

14.2.2. Definition. (i) A *filter* over $\mathbb{T} = \mathbb{T}^{\mathrm{BCD}}$ is a non-empty set $X \subseteq \mathbb{T}$ such that

(1) $A \in X \;\&\; A \leq_{\mathrm{BCD}} B \;\Rightarrow\; B \in X$;

(2) $A, B \in X \;\Rightarrow\; (A \cap B) \in X$.

(ii) $\mathcal{F}^{\mathrm{BCD}}$, or just $\mathcal{F}$, denotes the set of filters over $\mathbb{T}$.

14.2.3. Definition. (i) If $X \subseteq \mathbb{T}$ is non-empty, then the filter *generated by* X, notation $\uparrow X$, is the smallest filter containing X.

(ii) A *principal* filter is of the form $\uparrow\{A\}$ for some $A \in \mathbb{T}$. We shall denote this simply by $\uparrow A$. Note that $\uparrow A = \{B \mid A \leq B\}$.

14.2.4. Proposition. (i) *$\mathcal{F} = \langle \mathcal{F}, \subseteq \rangle$ is an ω-algebraic complete lattice.*

(ii) *$\mathcal{F}$ has as bottom element $\uparrow\mathtt{U}$ and as top element $\mathbb{T}$.*

(iii) *The compact elements of $\mathcal{F}$ are exactly the principal filters.*

14.2.5. Definition. Let D be an ω-algebraic lattice and let

$$\begin{array}{rcl} F & : & D{\to}[D{\to}D] \\ G & : & [D{\to}D]{\to}D \end{array}$$

be Scott continuous. D is called a *reflexive* via F, G if $F \circ G = \mathtt{id}_{[D\to D]}$.

A reflexive element of **ALG** is also a λ-model in which the term interpretation is naturally defined as follows (see Barendregt [1984], Section 5.4).

14.2.6. Definition (Interpretation of terms). Let D be reflexive via F, G.

(i) A *term environment* in D is a map $\rho : \mathtt{Var}{\to}D$.

(ii) If ρ is a term environment and $d \in D$, then $\rho(x := d)$ is the term environment ρ' defined by

$$\begin{array}{rcll} \rho'(y) & = & \rho(y) & \text{if } y \not\equiv x; \\ \rho'(x) & = & d. & \end{array}$$

(iii) Given a term environment ρ, the interpretation $[\![\ ]\!]_\rho : \Lambda \to D$ is defined as follows.

$$\begin{aligned} [\![x]\!]^D_\rho &= \rho(x); \\ [\![MN]\!]^D_\rho &= F([\![M]\!]^D_\rho)([\![N]\!]^D_\rho); \\ [\![\lambda x.M]\!]^D_\rho &= G(\lambda\!\!\lambda d \in D.[\![M]\!]^D_{\rho(x:=d)}). \end{aligned}$$

(iv) The statement $M = N$, for M, N untyped lambda terms, is *true in* D, notation $D \models M = N$ iff

$$\forall \rho \in \mathsf{Env}_D.[\![M]\!]^D_\rho = [\![N]\!]^D_\rho.$$

14.2.7. Theorem. *Let D be reflexive via F, G. Then D is a λ-model, in particular for all $M, N \in \Lambda$*

$$D \models (\lambda x.M)N = M[x := N].$$

14.2.8. Proposition. *Define maps $F : \mathcal{F} \to [\mathcal{F} \to \mathcal{F}]$ and $G : [\mathcal{F} \to \mathcal{F}] \to \mathcal{F}$ by*

$$\begin{aligned} F(X)(Y) &= \uparrow\{B \mid \exists A \in Y.(A \to B) \in X\} \\ G(f) &= \uparrow\{A \to B \mid B \in f(\uparrow A)\}. \end{aligned}$$

Then $\mathcal{F}$ is reflexive via F, G. Therefore $\mathcal{F}$ is a λ-model.

An important property of the λ-model $\mathcal{F}$ is that the meaning of a term is the set of types which are deducible for it.

14.2.9. Theorem. *For all λ-terms M one has*

$$[\![M]\!]^{\mathcal{F}}_\rho = \{A \mid \exists \Gamma \models \rho.\Gamma \vdash M : A\},$$

where $\Gamma \models \rho$ iff for all $(x\ B) \in \Gamma$ one has $B \in \rho(x)$.

Lastly we notice that all continous functions are representable.

14.2.10. Theorem.

$$[\mathcal{F} \to \mathcal{F}] = \{f : \mathcal{F} \to \mathcal{F} \mid f \text{ is representable}\},$$

where $f \in \mathcal{F} \to \mathcal{F}$ is called representable *iff for some $X \in \mathcal{F}$ one has*

$$\forall Y \in \mathcal{F}.f(Y) = F^{\mathrm{BCD}}(X)(Y).$$

14.3. Completeness of type assignment

14.3.1. Definition (Interpretation of types). Let D be reflexive via F, G and hence a λ-model. For $F(d)(e)$ we also write (as usual) $d \cdot e$.
(i) A *type environment* in D is a map $\xi : \mathbb{A}_\infty \to \mathcal{P}(D)$.

(ii) For $X, Y \in \mathcal{P}(D)$ define

$$X \to Y = \{d \in D \mid d \cdot X \subseteq Y\} = \{d \in D \mid \forall x \in X. d \cdot x \in Y\}.$$

(iii) Given a type environment ξ, the interpretation $[\![\ ]\!]_\xi : \mathbb{T} \to \mathcal{P}(D)$ is defined as follows.

$$\begin{array}{rcll} [\![\mathtt{U}]\!]^D_\xi & = & D; & \\ [\![\alpha]\!]^D_\xi & = & \xi(\alpha), & \text{for } \alpha \in \mathbb{A}_\infty; \\ [\![A \to B]\!]^D_\xi & = & [\![A]\!]^D_\xi \to [\![B]\!]^D_\xi; & \\ [\![A \cap B]\!]^D_\xi & = & [\![A]\!]^D_\xi \cap [\![B]\!]^D_\xi. & \end{array}$$

14.3.2. Definition (Satisfaction). (i) Given a λ-model D, a term environment ρ and a type environment ξ one defines the following.

$$\begin{array}{rcll} D, \rho, \xi \models M : A & \iff & [\![M]\!]^D_\rho \in [\![A]\!]^D_\xi. & \\ D, \rho, \xi \models \Gamma & \iff & D, \rho, \xi \models x : B, & \text{for all } (x\ B) \in \Gamma. \end{array}$$

(ii) $\Gamma \models M : A \iff \forall D, \rho, \xi. [D, \rho, \xi \models \Gamma \ \Rightarrow\ \rho, \xi \models M : A]$.

14.3.3. Theorem (Soundness).

$$\Gamma \vdash M : A \ \Rightarrow\ \Gamma \models M : A.$$

14.3.4. Theorem (Completeness).

$$\Gamma \models M : A \ \Rightarrow\ \Gamma \vdash M : A.$$

The completeness proof is an application of the λ-model $\mathcal{F}$, see Barendregt et al. [1983].

Chapter 15

Type Assignment Systems

21.2.2008:897

This chapter defines a family of systems $\lambda^{\mathcal{T}}_{\cap}$ that assign intersection types to untyped lambda terms. These systems have a common set of typing rules parametric in an *intersection type theory* $\mathcal{T}$. They are obtained as a generalization of the examplary system $\lambda^{\mathrm{BCD}}_{\cap}$ presented in Chapter 14.

In Section 15.1, we start by defining a set $\mathbb{T}^{\mathbb{A}}$ of intersection types similar to $\mathbb{T}^{\mathrm{BCD}}$, where the set of type atoms is now an arbitrary one denoted by $\mathbb{A}$. Then, we define the *intersection type theory* $\mathcal{T}$ as the set of statements of the form $A \leq_{\mathcal{T}} B$ (or just $A \leq B$) with $A, B \in \mathbb{T}^{\mathbb{A}}$ satisfying some logical rules which ensure that $\leq_{\mathcal{T}}$ is a pre-order on $\mathbb{T}^{\mathbb{A}}$. In particular, the logical rules for the intersection will ensure that $A \cap B$ is a greatest lower bound for the types A and B. Since all type theories in Part III of this book are using the intersection operator, we keep this implicit and often simply speak about *type theories* without the word 'intersection' in the front.

For some type theories a particular atom, denoted by $\mathtt{U}$, is selected to act as *universal type*: intended as the type of all lambda terms. The rules of type assignment are such that if $\mathtt{U} \leq A$, then also A is a universal element. So it is natural (but not strictly necessary) to require that $\mathtt{U}$ is the top element. The class of simple intersection type theories without an explicit universal element is denoted by $\mathrm{TT}^{-\mathtt{U}}$ and the one with a universal element by $\mathrm{TT}^{\mathtt{U}}$. For the (disjoint) union we write $\mathrm{TT} = \mathrm{TT}^{-\mathtt{U}} \cup \mathrm{TT}^{\mathtt{U}}$.

Figure 15.1 shows the thirteen specific examples of type theories that will be considered where BCD is amongst them. These theories are denoted by names (respectively acronymes) of the author(s) who have first considered the λ-model induced by such a theory. In this list the given order is logical, rather than historical, and some of the references define the models directly, others deal with the corresponding filter models. In some cases the type theory was modelled after an existing domain model in order to study the image of terms and hence equality of terms into that model; in other cases the type theory came first and created a domain model with certain properties.

The first ten type theories of Fig. 15.1 have the universal type $\mathtt{U}$ and the

remaining three do not, i.e.

$$\begin{array}{ll} \text{Scott, Park, CDZ, HR, DHM, BCD, AO, Plotkin, Engeler, CDS} & \in \mathrm{TT}^{\mathtt{U}} \\ \text{HL, CDV, CD} & \in \mathrm{TT}^{-\mathtt{U}} \end{array}$$

Some of these type theories have other type atoms such as 0 and 1. We will end this section by proving some basic lemmas for these specific type theories. In particular, we will prove that $0 < 1 < \mathtt{U}$.

$\mathcal{T}$	$\lambda^{\mathcal{T}}_{\cap}$	$\mathcal{F}^{\mathcal{T}}$	Reference
Scott	$\lambda^{\mathrm{Scott}}_{\cap}$	$\mathcal{F}^{\mathrm{Scott}}$	Scott [1972]
Park	$\lambda^{\mathrm{Park}}_{\cap}$	$\mathcal{F}^{\mathrm{Park}}$	Park [1976]
CDZ	$\lambda^{\mathrm{CDZ}}_{\cap}$	$\mathcal{F}^{\mathrm{CDZ}}$	Coppo et al. [1987]
HR	$\lambda^{\mathrm{HR}}_{\cap}$	$\mathcal{F}^{\mathrm{HR}}$	Honsell and Ronchi Della Rocca [1992]
DHM	$\lambda^{\mathrm{DHM}}_{\cap}$	$\mathcal{F}^{\mathrm{DHM}}$	Dezani-Ciancaglini et al. [2005a]
BCD	$\lambda^{\mathrm{BCD}}_{\cap}$	$\mathcal{F}^{\mathrm{BCD}}$	Barendregt et al. [1983]
AO	$\lambda^{\mathrm{AO}}_{\cap}$	$\mathcal{F}^{\mathrm{AO}}$	Abramsky and Ong [1993]
Plotkin	$\lambda^{\mathrm{Plotkin}}_{\cap}$	$\mathcal{F}^{\mathrm{Plotkin}}$	Plotkin [1993]
Engeler	$\lambda^{\mathrm{Engeler}}_{\cap}$	$\mathcal{F}^{\mathrm{Engeler}}$	Engeler [1981]
CDS	$\lambda^{\mathrm{CDS}}_{\cap}$	$\mathcal{F}^{\mathrm{CDS}}$	Coppo et al. [1979]
HL	$\lambda^{\mathrm{HL}}_{\cap}$	$\mathcal{F}^{\mathrm{HL}}$	Honsell and Lenisa [1999]
CDV	$\lambda^{\mathrm{CDV}}_{\cap}$	$\mathcal{F}^{\mathrm{CDV}}$	Coppo et al. [1981]
CD	$\lambda^{\mathrm{CD}}_{\cap}$	$\mathcal{F}^{\mathrm{CD}}$	Coppo and Dezani-Ciancaglini [1980]

Figure 15.1: Specific type theories, type assignment systems and filter models

In Section 15.2 we will assign types in $\mathbb{TT}^{\mathbb{A}}$ to lambda terms in Λ. Given a type theory $\mathcal{T}$, we will derive assertions of the form $\Gamma \vdash^{\mathcal{T}}_{\cap} M : A$ where $M \in \Lambda$, $A \in \mathbb{TT}^{\mathbb{A}}$ and Γ is a set of type declarations for the variables in M. For this, we define a set of typing rules parametric in $\mathcal{T}$ denoted by $\lambda^{\mathcal{T}}_{\cap}$ and called *type assignment system over* $\mathcal{T}$. This can be seen as a mapping

$$\mathcal{T} \in \mathrm{TT} \mapsto \text{set } \lambda^{\mathcal{T}}_{\cap} \text{ of typing rules (and axioms) parametric in } \mathcal{T}.$$

The parameter $\mathcal{T}$ appears in rule ($\leq$) and axiom ($\mathtt{U}$). The rule ($\leq$) states that a lambda term has type B if it has type A and $A \leq_{\mathcal{T}} B$. The axiom ($\mathtt{U}$) states that all lambda terms have type $\mathtt{U}$ only if $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$. In particular, the type assignments of the first ten type theories of Fig. 15.1 contain ($\mathtt{U}$) and the remaining three do not.

The systems $\lambda^{\mathcal{T}}_{\cap}$ also share a set of non-parametric rules for assigning lambda terms to the types $(A \cap B)$ and $(A \to B)$. The particular use of intersection is that if the same lambda term has type A and B, then it also has type $(A \cap B)$. The type $(A \to B)$ plays the same role as in the simply typed lambda calculus to cover the abstraction terms.

We have an infinite collection $\{\lambda^{\mathcal{T}}_{\cap} \mid \mathcal{T} \in \mathrm{TT}\}$ of type assignment systems which are defined by giving *only one* set of typing rules parametric in $\mathcal{T}$. Now we mention some of the advantages of having this general and common framework for all these systems:

1. We can capture most of the intersection type assignment systems that appear in the literature as shown in Fig. 15.1.
2. We can study general properties of $\lambda^{\mathcal{T}}_{\cap}$ that hold for all $\mathcal{T} \in \mathrm{TT}$ as it will be done in Chapter **??**.

In Section 15.3 we define the notion of intersection type structure

$$\mathcal{S} = \langle |\mathcal{S}|, \leq, \cap, \to \rangle$$

where $\leq$ is now a partial order (a pre-order that is anti-symmetric) and the greatest lower bound $\cap$ is unique. The collection of type structures is denoted by TS. Given a type theory $\mathcal{T}$, one usually requires that the equivalence relation $=_{\mathcal{T}}$ is a congruence with respect to $\to$. Then we speak of a *compatible* type theory, having a corresponding *type structure*

$$[\mathcal{T}] = \langle [\Pi], \leq, \cap, \to \rangle.$$

Each type structure can be seen as coming from a compatible type theory and compatible type theories and type structures are basically the same. In this section, we also introduce specific categories of lattices and type structures to accommodate them.

Finally in Section 15.4 we introduce the notion of *filter* over $\mathcal{T}$ as a set of types closed under intersection $\cap$ and pre-order $\leq$. If $\mathcal{T} \in \mathrm{TT}^{\,\mathtt{U}}$ then filters are non-empty and the smallest filter (ordered by subset inclusion) is the filter generated by $\{\mathtt{U}\}$. If $\mathcal{T} \in \mathrm{TT}^{\text{-}\mathtt{U}}$ then the empty set is considered to be a filter which in this case is the smallest one. As for the type assignment systems, we also have the mapping

$$\mathcal{T} \in \mathrm{TT} \mapsto \text{ a set } \mathcal{F}^{\mathcal{T}} \text{ of filters over } \mathcal{T}.$$

We also define the notion of filter structure over $\mathcal{T}$ as a triple $\langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle$ where $F^{\mathcal{T}}, G^{\mathcal{T}}$ are operations for interpreting application and abstraction respectively. We also have a mapping

$$\mathcal{T} \in \mathrm{TT} \mapsto \mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle.$$

In Chapter 17 a proper categorical setting is provided to study some interesting properties of these mappings as functors. This will be used to establish equivalences of categories of specific types structures and algebraic lattices.

In Chapter **??** we will study general conditions on $\mathcal{T}$ to ensure that the filter structures $\mathcal{F}^{\mathcal{T}}$ are models of the untyped lambda calculus, i.e. *filter models*. This will cover the thirteen specific cases of filter models of Fig. 15.1 which appear in the literature. The first ten are models of the λ-calculus (when $\mathcal{T} \in \mathrm{TT}^{\,\mathtt{U}}$) and the remaining three are models of the $\lambda\mathsf{I}$-calculus (when $\mathcal{T} \in \mathrm{TT}^{\text{-}\mathtt{U}}$).

15.1. Type theories

15.1.1. Definition. Let $\mathbb{A}$ be a countable (finite or countably infinite) set of symbols, called *(type) atoms*. The set of *intersection types* over $\mathbb{A}$, notation $\Pi^{\mathbb{A}}_{\cap}$

(if there is little danger of confusion also denoted by $\mathbb{T}_\cap$, $\mathbb{T}^\mathbb{A}$ or just $\mathbb{T}$), is defined by the following abstract syntax.

$$\mathbb{T} = \mathbb{A} \mid \mathbb{T}\to\mathbb{T} \mid \mathbb{T}\cap\mathbb{T}.$$

15.1.2. Remark. (i) The set $\mathbb{A}$ varies in the applications of intersection types.

(ii) In Chapter 14 the set of intersection types $\mathbb{T}$ was defined over the set of atoms $\mathbb{A}^{\mathtt{U}}_\infty = \{\mathtt{c}_0, \mathtt{c}_1, \ldots\} \cup \{\mathtt{U}\}$. We write this as

$$\mathbb{A}^{\mathrm{BCD}} = \mathbb{A}^{\mathtt{U}}_\infty \text{ and } \mathbb{T}^{\mathrm{BCD}} = \mathbb{T}^{\mathbb{A}^{\mathtt{U}}_\infty}.$$

(iii) The following are some of the type atoms that will be used in different contexts.

$\mathtt{c}_0, \mathtt{c}_1, \mathtt{c}_2, \ldots$	indiscernible atoms
$0, 1, \mathtt{U}$	atoms with special properties.

(iv) The atom $\mathtt{U}$ is called *universe*. Its intention is to host all lambda terms. The intention of the special atoms 1 varies: sometimes it hosts the strongly normalizing terms, sometimes the terms which reduce to λI-terms. Similarly 0 can host the terms which reduce to closed terms or other sets of terms. This will be determined by the properties of the type theory in which 1 and 0 occur. See Figure 19.1.

Notation. (i) Greek letters $\alpha, \beta, \ldots$ range over arbitrary atoms in $\mathbb{A}$.

(ii) The letters $A, B, C, \ldots$ range over types in $\mathbb{T}^\mathbb{A}$.

(iii) Some papers on intersection types such as Coppo et al. [1979] use the greek letter ω to denote the universal type, while they use the atoms 0 and 1 with the same meaning as here. Other papers such as Dezani-Ciancaglini et al. [2003] use Ω for the universal type and ω and φ for 0 and 1.

15.1.3. Definition. (i) An *intersection type theory over a countable set of type atoms* $\mathbb{A}$ is a set $\mathcal{T}$ of sentences of the form $A \leq B$ (to be read: A is a subtype of B), with $A, B \in \mathbb{T}^\mathbb{A}$, satisfying at least the following axioms and rules.

(refl)	$A \leq A$
(incl_L)	$A \cap B \leq A$
(incl_R)	$A \cap B \leq B$
(glb)	$\dfrac{C \leq A \quad C \leq B}{C \leq A \cap B}$
(trans)	$\dfrac{A \leq B \quad B \leq C}{A \leq C}$

This means that e.g. $(A \leq A) \in \mathcal{T}$ and $(A \leq B), (B \leq C) \in \mathcal{T} \;\Rightarrow\; (A \leq C) \in \mathcal{T}$, for all $A, B, C \in \mathbb{T}^\mathbb{A}$.

(ii) The notion 'intersection type theory' will be abbreviated as 'type theory', as the 'intersection' part is default.

The set of type theories is denoted by TT. In the following definition, we distinguish two disjoint subsets, $\mathrm{TT}^{\mathtt{U}}$ and $\mathrm{TT}^{\text{-}\mathtt{U}}$.

15.1.4. Definition. (i) The collection of *intersection type theories with universe* $\mathtt{U}$, notation $\mathrm{TT}^{\mathtt{U}}$, consists of the type theories $\mathcal{T}$ over a $\mathbb{A}$ such that $\mathtt{U} \in \mathbb{A}$ and $\mathtt{U}$ is the top, i.e. the following holds for all types A

$$\boxed{(\mathtt{U}_{\mathrm{top}}) \quad A \leq \mathtt{U}}$$

In the corresponding type structures treated in 15.3 the universal element will be visible in the signature.

(ii) The collection of *non-universal intersection type theories*, notation $\mathrm{TT}^{\text{-}\mathtt{U}}$, consist of the type theories without such special atom $\mathtt{U}$ that is a top.

15.1.5. Remark. (i) An intersection type theory $\mathcal{T}$ can fail in three ways to be in $\mathrm{TT}^{\text{-}\mathtt{U}}$: (i) there is a atom $\mathtt{U}$, but it not a top; (ii) there is a top $\top$, but it is not declared as a universal element; (iii) there is neither special element $\mathtt{U}$, nor a top $\top$. In all these cases $\mathcal{T} \in \mathrm{TT}^{\text{-}\mathtt{U}}$.

(ii) The intuition behind the special atom $\mathtt{U}$, and its name, will become clear in the next section 15.2 on type assignment. The types in a TT will be assigned to untyped lambda terms. Arrow types like $A \to B$ are assigned to abstraction terms of the form $\lambda x.M$. For $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$ we will postulate the assignment

$$\Gamma \vdash M : \mathtt{U},$$

for all Γ and all $M \in \Lambda$. So $\mathtt{U}$ is *a universal type*, i.e. a type assigned to all terms. Another rule of type assignment is

$$\frac{\Gamma \vdash M : A \quad A \leq B}{\Gamma \vdash M : B}.$$

Hence if $\mathtt{U} \leq A$, then also A is a universal type. Therefore, for type theories in which we wish to have both a universe $\mathtt{U}$ and a top $\top$, we will always choose $\mathtt{U} = \top$, i.e. we have for all types A

$$A \leq \mathtt{U}.$$

There will be $\mathcal{T} \in \mathrm{TT}$ with a top, but without a universe. In principle the converse, a TT with a universe, but without a top could also be considered, but we will not do so in the main theory. The system $\lambda_{\cap}^{\mathrm{Krivine}^{\mathtt{U}}}$ of Krivine, see Exercise **??**, is such an example.

15.1.6. Remark. (i) In this Part of the book $\mathcal{T}$ ranges over elements of TT.

(ii) If $\mathcal{T} \in \mathrm{TT}$ over $\mathbb{A}$, then we also write $\mathbb{T}^{\mathcal{T}} = \mathbb{T}_{\cap}^{\mathbb{A}}$ and $\mathbb{A}^{\mathcal{T}} = \mathbb{A}$.

15.1.7. Remark. [**methodology**] Most $\mathcal{T} \in \mathrm{TT}$ have some extra axioms or rules, the above set in Definition 15.1.3 being the minimum requirement. For example the theory BCD over $\mathbb{A} = \mathbb{A}_{\infty}^{\mathtt{U}}$, introduced in Chapter 14, is a $\mathrm{TT}^{\mathtt{U}}$ and has the extra axioms $(\mathtt{U}\!\to)$ and $(\to\!\cap)$ and rule $(\to)$.

15.1.8. NOTATION. Let $\mathcal{T} \in \mathrm{TT}$. We write the following.
(i) $A \leq_{\mathcal{T}} B$ for $(A \leq B) \in \mathcal{T}$.
(ii) $A =_{\mathcal{T}} B$ for $A \leq_{\mathcal{T}} B \leq_{\mathcal{T}} A$.
(iii) $A <_{\mathcal{T}} B$ for $A \leq B \;\&\; A \neq_{\mathcal{T}} B$.
(iv) If there is little danger of confusion and $\mathcal{T}$ is clear from the context, then we will write $\leq, =, <$ for respectively $\leq_{\mathcal{T}}, =_{\mathcal{T}}, <_{\mathcal{T}}$.
(v) We write $A \equiv B$ for syntactic identity. E.g. $A \cap B \equiv A \cap B$, but $A \cap B \not\equiv B \cap A$. Note that always $A \cap B =_{\mathcal{T}} B \cap A$ in a type theory.
(vi) We write $[\mathbb{T}]$ for $\mathbb{T}$ modulo $=_{\mathcal{T}}$.
(vii) We will use the informal notation $A_1 \cap \ldots \cap A_n$ to denote the intersection of a sequence of n types A_i for $i \in \{1 \ldots, n\}$ associating to the right. If $n = 3$ then $A_1 \cap \ldots \cap A_n$ denotes $(A_1 \cap (A_2 \cap A_3))$. In case $n = 0$ and $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$, we intend that the sequence is empty and $A_1 \cap \ldots \cap A_n$ denotes $\mathtt{U}$.
(viii) For a finite $I = \{1, \ldots, n\}$ we may also use the notation $\bigcap_{i \in I} A_i$ to denote $A_1 \cap \ldots \cap A_n$.

15.1.9. PROPOSITION. *The following rule is derived.*

$$(\mathrm{mon}) \quad \frac{A \leq A' \quad B \leq B'}{A \cap B \leq A' \cap B'}$$

PROOF. By (trans), (incl_L), (incl_R) and (glb). ■

The above proposition implies that for any $\mathcal{T}$, $=_{\mathcal{T}}$ is compatible with the operator $\cap$. For the case that $=_{\mathcal{T}}$ is compatible with $\to$ we define the following.

15.1.10. DEFINITION. $\mathcal{T}$ is called *compatible* iff the following rule is admissible.

$$(\to^{=}) \quad \frac{A = A' \quad B = B'}{(A \to B) = (A' \to B')}$$

This means $A =_{\mathcal{T}} A' \;\&\; B =_{\mathcal{T}} B' \;\Rightarrow\; (A \to B) =_{\mathcal{T}} (A' \to B')$. One way to insure this is to adopt $(\to^{=})$ as rule determining $\mathcal{T}$.

15.1.11. LEMMA. (i) *For any $\mathcal{T}$ one has $A \cap B =_{\mathcal{T}} B \cap A$.*
(ii) *If $\mathcal{T}$ is compatible then $(A \cap B) \to C =_{\mathcal{T}} (B \cap A) \to C$.*

PROOF. (i) By (incl_L), (incl_R) and (glb).
(ii) By (i). ■

15.1.12. REMARK. Similarly to Remark 14.1.7 any $\mathcal{T} \in \mathrm{TT}$ can be seen as a structure with a pre-order $\mathcal{T} = \langle \mathbb{T}, \leq, \cap, \to \rangle$. This means that $\leq$ is reflexive and transitive, but not necessarily anti-symmetric

$$A \leq B \;\&\; B \leq A \;\not\Rightarrow\; A \equiv B.$$

One can go over to equivalence classes and define a partial order on $[\mathbb{T}]$ by

$$[A] \leq [B] \iff A \leq B.$$

By Proposition 15.1.9, $\cap$ is always well defined on $[\mathbb{T}]$ by $[A] \cap [B] = [A \cap B]$. To ensure that $\to$ is well defined by $[A] \to [B] = [A \to B]$, we need to require that $=$ is compatible with $\to$. This is the case if $\mathcal{T}$ is compatible and one obtains a type structure

$$[\mathcal{T}] \quad = \quad \langle [\mathbb{T}], \leq, \cap, \to \rangle.$$

This structure is a meet semi-lattice, i.e. a poset with $[A] \cap [B]$ the greatest lower bound of $[A]$ and $[B]$. If moreover $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$, then $[\mathcal{T}]$ can be enriched to a type structure with universe $[\mathtt{U}]$ of the form

$$[\mathcal{T}] \quad = \quad \langle [\mathbb{T}], \leq, \cap, \to, [\mathtt{U}] \rangle.$$

This will be done in Section 15.3.

Specific intersection type theories

Now we will construct several, in total thirteen, type theories that will play an important role in later chapters, by introducing the following axiom schemes, rule schemes and axioms. Only two of them are non-compatible, so we obtain eleven type structures.

Axioms	
(0_{Scott})	$(\mathtt{U} \to 0) = 0$
(0_{Park})	$(0 \to 0) = 0$
(01)	$0 \leq 1$
$(1 \to 0)$	$(1 \to 0) = 0$
$(0 \to 1)$	$(0 \to 1) = 1$
(I)	$(1 \to 1) \cap (0 \to 0) = 1$

Axiom schemes	
$(\mathtt{U}_{\mathrm{top}})$	$A \leq \mathtt{U}$
$(\mathtt{U} \to)$	$\mathtt{U} \leq (A \to \mathtt{U})$
$(\mathtt{U}_{\mathrm{lazy}})$	$(A \to B) \leq (\mathtt{U} \to \mathtt{U})$
$(\to \cap)$	$(A \to B) \cap (A \to C) \leq A \to B \cap C$
$(\to \cap^{=})$	$(A \to B) \cap (A \to C) = A \to B \cap C$

Rule schemes	
$(\to)$	$\dfrac{A' \leq A \quad B \leq B'}{(A \to B) \leq (A' \to B')}$
$(\to^{=})$	$\dfrac{A' = A \quad B = B'}{(A \to B) = (A' \to B')}$

Figure 15.2: Possible Axioms and Rules concerning $\leq$.

15.1.13. Remark. (i) The possible axiom scheme ($\mathtt{U}_{\rm top}$) states that the universe $\mathtt{U}$ is a top element.

(ii) In the presence of ($\to$) the axiom-scheme ($\mathtt{U}\to$) is equivalent with the axiom $\mathtt{U} \leq (\mathtt{U} \to \mathtt{U})$. Also in that case the axiom-scheme ($\to\cap$) is equivalent with ($\to\cap^{=}$). See Exercise **??**.

15.1.14. Definition. In Figure 15.3 a collection of elements of TT is defined. For each name $\mathcal{T}$ a set $\mathbb{A}^{\mathcal{T}}$ of atoms and a set of rules and axiom(scheme)s are given. The type theory $\mathcal{T}$ is the smallest intersection type theory over $\mathbb{A}^{\mathcal{T}}$ (see Definition 15.1.3) that satisfies the rules and axioms of $\mathcal{T}$ shown in Figure 15.3.

$\mathcal{T}$	$\mathbb{A}^{\mathcal{T}}$	Rules	Axiom Schemes	Axioms
Scott	$\{\mathtt{U}, 0\}$	($\to$)	($\to\cap$), ($\mathtt{U}_{\rm top}$), ($\mathtt{U}\to$)	($0_{\rm Scott}$)
Park	$\{\mathtt{U}, 0\}$	($\to$)	($\to\cap$), ($\mathtt{U}_{\rm top}$), ($\mathtt{U}\to$)	($0_{\rm Park}$)
CDZ	$\{\mathtt{U}, 1, 0\}$	($\to$)	($\to\cap$), ($\mathtt{U}_{\rm top}$), ($\mathtt{U}\to$)	(01), (1→0), (0→1)
HR	$\{\mathtt{U}, 1, 0\}$	($\to$)	($\to\cap$), ($\mathtt{U}_{\rm top}$), ($\mathtt{U}\to$)	(01), (1→0), (I)
DHM	$\{\mathtt{U}, 1, 0\}$	($\to$)	($\to\cap$), ($\mathtt{U}_{\rm top}$), ($\mathtt{U}\to$)	(01), (0→1), ($0_{\rm Scott}$)
BCD	$\mathbb{A}^{\mathtt{U}}_{\infty}$	($\to$)	($\to\cap$), ($\mathtt{U}_{\rm top}$), ($\mathtt{U}\to$)	
AO	$\{\mathtt{U}\}$	($\to$)	($\to\cap$), ($\mathtt{U}_{\rm top}$), ($\mathtt{U}_{\rm lazy}$)	
Plotkin	$\{\mathtt{U}, 0\}$	($\to^{=}$)	($\mathtt{U}_{\rm top}$)	–
Engeler	$\mathbb{A}^{\mathtt{U}}_{\infty}$	($\to^{=}$)	($\to\cap^{=}$), ($\mathtt{U}_{\rm top}$), ($\mathtt{U}\to$)	–
CDS	$\mathbb{A}^{\mathtt{U}}_{\infty}$	–	($\mathtt{U}_{\rm top}$)	–
HL	$\{1, 0\}$	($\to$)	($\to\cap$)	(01), (0→1), (1→0)
CDV	$\mathbb{A}_{\infty}$	($\to$)	($\to\cap$)	–
CD	$\mathbb{A}_{\infty}$	–	–	–

Figure 15.3: Various type theories

15.1.15. Remark. (i) Note that CDS and CD are non-compatible, while the other eleven are compatible.

(ii) The first ten type theories of Figure 15.3 belong to $\mathrm{TT}^{\mathtt{U}}$ and the last three to $\mathrm{TT}^{-\mathtt{U}}$. In Lemma 15.1.22(i) we will see that HL has 1 as top.

(iii) The type theories CDV and CD do not have a top at all, as shown in Lemma 15.1.22(ii) and (iii).

15.1.16. Remark. The expressive power of intersection types is remarkable. This will become apparent when we will use them as a tool for characterizing properties of λ-terms (see Sections 19.1 and 19.4), and for describing different λ-models (see Section 18.3).

Much of this expressive power comes from the fact that they are endowed with a *preorder relation*, $\leq$, which induces, on the set of types modulo $=$, the structure of a meet semi-lattice with respect to $\cap$. This appears natural when we think of types as subsets of a domain of discourse D (the interpretation of

the universe $\mathtt{U}$), which is endowed with a (partial) application $\cdot : D \times D \to D$, and interpret $\cap$ as set-theoretic intersection, $\leq$ as set inclusion, and give $\to$ the *realizability interpretation* $[\![A]\!]_\xi \subseteq D$ for each type A. One starts by interpreting types in $\mathbb{T}_\to$.

$$\begin{array}{rcl} [\![\alpha]\!]_\xi & = & \xi(\alpha), \text{ for } \alpha \not\equiv \mathtt{U}; \\ [\![A \to B]\!]_\xi & = & [\![A]\!]_\xi \to [\![B]\!]_\xi = \{d \in D \mid d \cdot [\![A]\!]_\xi \subseteq [\![B]\!]_\xi\}. \end{array}$$

This semantics, due to Scott, can be extended to intersection types by

$$\begin{array}{rcl} [\![\mathtt{U}]\!]_\xi & = & D; \\ [\![A \cap B]\!]_\xi & = & [\![A]\!]_\xi \cap [\![B]\!]_\xi. \end{array}$$

Given the right TT and right domain the following holds.

$$A \leq B \text{ iff forall } \xi \text{ one has } [\![A]\!]_\xi \subseteq [\![B]\!]_\xi.$$

This type of semantics will be studied in Section 19.2.

The type $\mathtt{U} \to \mathtt{U}$ is the set of functions which applied to an arbitrary element return again an arbitrary element. In that case axiom scheme $(\mathtt{U} \to)$ expresses the fact that all the objects in our domain of discourse are total functions, i.e. that $\mathtt{U}$ is equal to $A \to \mathtt{U}$, hence $A \to \mathtt{U} = B \to \mathtt{U}$ for all A, B (Barendregt et al. [1983]). If now we want to capture only those terms which truly represent functions, as we do for example in the lazy λ-calculus, we cannot assume axiom $(\mathtt{U} \to)$. One still may postulate the weaker property $(\mathtt{U}_{\text{lazy}})$ to make all functions total (Abramsky and Ong [1993]). It simply says that an element which is a function, because it maps A into B, maps also the whole universe into itself.

The intended interpretation of arrow types also motivates axiom $(\to\cap)$, which implies that if a function maps A into B, and the same function maps also A into C, then, actually, it maps the whole A into the intersection between B and C (i.e. into $B \cap C$), see Barendregt et al. [1983].

Rule $(\to)$ is again very natural in view of the set-theoretic interpretation. It implies that the arrow constructor is contravariant in the first argument and covariant in the second one. It is clear that if a function maps A into B, and we take a subset A' of A and a superset B' of B, then this function will map also A' into B', see Barendregt et al. [1983].

The rule $(\to\cap^=)$ is similar to the rule $(\to\cap)$. It capture properties of the graph models for the untyped lambda calculus, see Plotkin [1975] and Engeler [1981], as we shall discuss in Section 18.3.

For Scott, Park, CDZ, HR, DHM, the axioms express peculiar properties of D_∞-like inverse limit models (see Section 18.3). For Park, CDZ, HR, DHM as well as for HL, the axioms also express properties of subsets of λ-terms (see Figure 19.1 and Proposition 19.1.13).

15.1.17. Remark. In Figure 15.4 we have connected $\mathcal{T}_1$ with an edge towards the higher positioned $\mathcal{T}_2$ in case $\mathcal{T}_1 \subset \mathcal{T}_2$. In Exercise **??** it is shown that the inclusions are strict. Above the horizontal line we find the elements of $\mathrm{TT}^{\mathtt{U}}$, below of $\mathrm{TT}^{-\mathtt{U}}$.

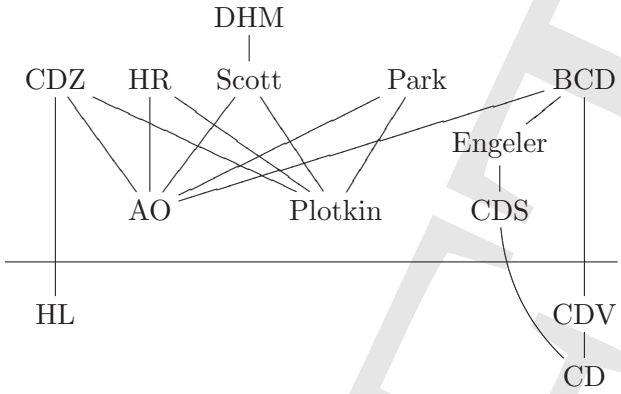


Figure 15.4: Inclusion among some intersection type theories.

Some classes of type theories

Now we will consider some classes of type theories. In order to do this, we list the relevant defining properties.

15.1.18. Definition. We define special subclasses of TT.

Class	Defining axiom(-scheme)(s) or rule(-scheme)(s)
graph	$(\to^{=}), (\mathtt{U}_{\mathrm{top}})$
lazy	$(\to), (\to\cap), (\mathtt{U}_{\mathrm{top}}), (\mathtt{U}_{\mathrm{lazy}})$
natural	$(\to), (\to\cap), (\mathtt{U}_{\mathrm{top}}), (\mathtt{U}\to)$
proper	$(\to), (\to\cap)$

15.1.19. Notation. The sets of graph, lazy, natural and proper type theories are denoted by respectively $\mathrm{GTT}^{\mathtt{U}}$, $\mathrm{LTT}^{\mathtt{U}}$, $\mathrm{NTT}^{\mathtt{U}}$ and PTT. The following subset inclusions are easily deduced from their definition.

$$\begin{array}{ccccccc} \mathrm{NTT}^{\mathtt{U}} & \subset & \mathrm{LTT}^{\mathtt{U}} & \subset & \mathrm{GTT}^{\mathtt{U}} & \subset & \mathrm{TT}^{\mathtt{U}} \\ & & \cap & & & & \cap \\ & & \mathrm{PTT} & & \subset & & \mathrm{TT} \end{array}$$

15.1.20. Remark. The type theories of Figure 15.3 are classified as follows.

non compatible	CD, CDS
graph	Plotkin,Engeler
lazy	AO
natural	Scott, Park, CDZ, HR, DHM, BCD
proper	HL, CDV

This table indicates the typical classification for these type theories: for example, CDZ is typically natural because $\mathrm{NTT}^{\mathtt{U}}$ is the smallest set that contains CDZ.

Some properties about specific TTs

Results about proper type theories

All type theories of Fig. 15.3 are proper, except CD, CDS, Plotkin, Engeler.

15.1.21. Proposition. *Let $\mathcal{T}$ be proper. Then we have*

(i) $(A{\to}B)\cap(A'{\to}B')\leq(A\cap A'){\to}(B\cap B')$;

(ii) $(A_1{\to}B_1)\cap\ldots\cap(A_n{\to}B_n)\leq(A_1\cap\ldots\cap A_n){\to}(B_1\cap\ldots\cap B_n)$;

(iii) $(A{\to}B_1)\cap\ldots\cap(A{\to}B_n)=A{\to}(B_1\cap\ldots B_n)$.

Proof. (i) $\begin{array}{rcl}(A{\to}B)\cap(A'{\to}B') &\leq& ((A\cap A'){\to}B)\cap((A\cap A'){\to}B')\\ &\leq& (A\cap A'){\to}(B\cap B'),\end{array}$

by respectively $(\to)$ and $(\to\cap)$.

(ii) Similarly (i.e. by induction on $n{>}1$, using (i) for the induction step).

(iii) By (ii) one has $(A{\to}B_1)\cap\ldots\cap(A{\to}B_n)\leq A{\to}(B_1\cap\ldots B_n)$. For $\geq$ use $(\to)$ to show that $A{\to}(B_1\cap\ldots\cap B_n)\leq(A{\to}B_i)$, for all i. ■

It follows that the mentioned equality and inequalities hold for Scott, Park, CDZ, HR, DHM, BCD, AO, HL and CDV.

Results about the type theories of Figure 15.3

15.1.22. Lemma. (i) 1 *is the top and* 0 *the bottom element in* HL.

(ii) CDV *has no top element.*

(iii) CDS, CD *have no top element.*

Proof. (i) By induction on the generation of $\mathbb{T}^{\mathrm{HL}}$ one shows that $0\leq A\leq 1$ for all $A\in\mathbb{T}^{\mathrm{HL}}$.

(ii) If α is a fixed atom and

$$\mathcal{B}_\alpha := \alpha \mid \mathcal{B}_\alpha\cap\mathcal{B}_\alpha$$

and $A\in\mathcal{B}_\alpha$, then one can show by induction on the generation of $\leq_{\mathrm{CDV}}$ that $A\leq_{\mathrm{CDV}}B\ \Rightarrow\ B\in\mathcal{B}_\alpha$. Hence if $\alpha\leq_{\mathrm{CDV}}B$, then $B\in\mathcal{B}_\alpha$. Since $\mathcal{B}_{\alpha_1}$ and $\mathcal{B}_{\alpha_2}$ are disjoint when α_1 and α_2 are two different atoms, we conclude that CDV has no top element.

(iii) Similarly to (ii), but easier. ■

15.1.23. Remark. By the above lemma, the atom 1 turns out to be the top element in HL. But 1 is not declared as a universe and hence, HL is not in $\mathrm{TT}^{\mathtt{U}}$.

In the following Lemmas 15.1.24-15.1.28 we study the positions of the atoms 0, and 1 in the TTs introduced in Figure 15.3. The principal result is that $0 < 1$ in HL and, as far as applicable,

$$0 < 1 < \mathtt{U},$$

in the theories Scott, Park, CDZ, HR, DHM and Plotkin.

15.1.24. LEMMA. *Let* $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{BCD}, \text{Engeler}\}$. *Define inductively the following collection of types*

$$\boxed{\mathcal{B} \;:=\; \mathtt{U} \mid \mathbb{T}^{\mathcal{T}} \to \mathcal{B} \mid \mathcal{B} \cap \mathcal{B}}$$

Then $\mathcal{B} = \{A \in \mathbb{T}^{\mathcal{T}} \mid A =_{\mathcal{T}} \mathtt{U}\}$.

PROOF. By induction on the generation of $A \leq_{\mathcal{T}} B$ one proves that $\mathcal{B}$ is closed upwards. This gives $\mathtt{U} \leq_{\mathcal{T}} A \;\Rightarrow\; A \in \mathcal{B}$.

By induction on the definition of $\mathcal{B}$ one shows, using $(\to)$ or $(\to^{=})$, $(\mathtt{U}_{\text{top}})$ and $(\mathtt{U}\to)$, that $A \in \mathcal{B} \;\Rightarrow\; A =_{\mathcal{T}} \mathtt{U}$.

Therefore

$$A =_{\mathcal{T}} \mathtt{U} \iff A \in \mathcal{B}. \;\blacksquare$$

15.1.25. LEMMA. *For* $\mathcal{T} \in \{\text{AO}, \text{Plotkin}\}$ *define inductively*

$$\boxed{\mathcal{B} \;:=\; \mathtt{U} \mid \mathcal{B} \cap \mathcal{B}}$$

Then $\mathcal{B} = \{A \in \mathbb{T}^{\mathcal{T}} \mid A =_{\mathcal{T}} \mathtt{U}\}$, *hence* $\mathtt{U}\to\mathtt{U} \neq_{\mathcal{T}} \mathtt{U}$.

PROOF. Similar to the proof of 15.1.22, but easier. $\blacksquare$

15.1.26. LEMMA. *For* $\mathcal{T} \in \{\text{CDZ}, \text{HR}, \text{DHM}\}$ *define by mutual induction*

$$\begin{array}{rcl} \mathcal{B} & = & 1 \mid \mathtt{U} \mid \mathbb{T}^{\mathcal{T}} \to \mathcal{B} \mid \mathcal{H} \to \mathbb{T}^{\mathcal{T}} \mid \mathcal{B} \cap \mathcal{B} \\ \mathcal{H} & = & 0 \mid \mathcal{B} \to \mathcal{H} \mid \mathcal{H} \cap \mathbb{T}^{\mathcal{T}} \mid \mathbb{T}^{\mathcal{T}} \cap \mathcal{H}. \end{array}$$

Then

$$\begin{array}{rcl} 1 \leq B & \Rightarrow & B \in \mathcal{B}, \\ A \leq 0 & \Rightarrow & A \in \mathcal{H}. \end{array}$$

PROOF. By induction on $\leq_{\mathcal{T}}$ one shows

$$A \leq B \;\Rightarrow\; (A \in \mathcal{B} \;\Rightarrow\; B \in \mathcal{B}) \;\Rightarrow\; (B \in \mathcal{H} \;\Rightarrow\; A \in \mathcal{H}).$$

From this the assertion follows immediately. $\blacksquare$

15.1.27. LEMMA. *We work with the theory* HL.

(i) *Define by mutual induction*

$$\begin{array}{rcl}\mathcal{B} & = & 1 \mid \mathcal{H}{\to}\mathcal{B} \mid \mathcal{B}\cap\mathcal{B}\\ \mathcal{H} & = & 0 \mid \mathcal{B}{\to}\mathcal{H} \mid \mathcal{H}\cap\mathbb{T}^{\mathrm{HL}} \mid \mathbb{T}^{\mathrm{HL}}\cap\mathcal{H}.\end{array}$$

Then

$$\begin{array}{rcl}\mathcal{B} & = & \{A\in\mathbb{T}^{\mathrm{HL}} \mid A =_{\mathrm{HL}} 1\};\\ \mathcal{H} & = & \{A\in\mathbb{T}^{\mathrm{HL}} \mid A =_{\mathrm{HL}} 0\}.\end{array}$$

(ii) $0 \neq_{\mathrm{HL}} 1$ *and hence* $0 <_{\mathrm{HL}} 1$.

PROOF. (i) By induction on $\leq_{\mathcal{T}}$ one shows

$$A \leq B \;\Rightarrow\; (A\in\mathcal{B} \;\Rightarrow\; B\in\mathcal{B}) \;\&\; (B\in\mathcal{H} \;\Rightarrow\; A\in\mathcal{H}).$$

This gives

$$(1 \leq B \;\Rightarrow\; B\in\mathcal{B}) \;\&\; (A \leq 0 \;\Rightarrow\; A\in\mathcal{H}).$$

By simultaneous induction on the generation of $\mathcal{B}$ and $\mathcal{H}$ one shows, using that 0 is the bottom element and 1 is the top element of HL, by Lemma 15.1.22(i),

$$(B\in\mathcal{B} \;\Rightarrow\; B = 1) \;\&\; (A\in\mathcal{H} \;\Rightarrow\; A = 0).$$

Now the assertion follows immediately.

(ii) By (i). ■

15.1.28. PROPOSITION. *In* HL *we have* $0 < 1$. *For the other members of Figure 15.3 as far as applicable*

$$0 < 1 < \mathtt{U}.$$

More precisely, in HL *one has*

(i) $0 < 1$.

In CDZ, HR, DHM *one has*

(ii) $0 < 1 < \mathtt{U}$.

PROOF. (i) By rule (01) and Lemma 15.1.27.

(ii) By rules (01), ($\mathtt{U}_{\mathrm{top}}$)and Lemmas 15.1.24-15.1.26. ■

15.2. Type assignment

Assignment of types from type theories

In this subsection we define an infinite collection $\{\lambda^{\mathcal{T}}_{\cap} \mid \mathcal{T}\in\mathrm{TT}\}$ of type assignment systems by giving a *uniform* set of typing rules parametric in $\mathcal{T}$.

15.2.1. DEFINITION. Let $\mathcal{T}\in\mathrm{TT}$.

(i) A $\mathcal{T}$-*statement* is of the form $M : A$, with $M\in\Lambda$ and $A\in\mathbb{T}^{\mathcal{T}}$.

(ii) A $\mathcal{T}$-*declaration* is a $\mathcal{T}$-statement of the form $x : A$.

(iii) A $\mathcal{T}$-*basis* Γ is a finite set of $\mathcal{T}$-declarations, with all variables distinct.

(iv) A $\mathcal{T}$-*assertion* is of the form

$$\Gamma \vdash M : A,$$

where $M : A$ is a $\mathcal{T}$-statement and Γ is a $\mathcal{T}$-basis.

15.2.2. REMARK. Let $M : A$ be a $\mathcal{T}$-statement.
(i) The term M is called the *subject* of this statement.
(ii) The type A is called its *predicate*.

15.2.3. DEFINITION. Let $\mathcal{T} \in \mathrm{TT}$. The *type assignment system* $\lambda^{\mathcal{T}}_{\cap}$ derives $\mathcal{T}$-assertions by the following axioms and rules.

(Ax)	$\Gamma \vdash x\ A$	if $(x\ A \in \Gamma)$
($\to$I)	$\dfrac{\Gamma, x\ A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}$	
($\to$E)	$\dfrac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$	
($\cap$I)	$\dfrac{\Gamma \vdash M : A \quad \Gamma \vdash M : B}{\Gamma \vdash M : A \cap B}$	
($\leq$)	$\dfrac{\Gamma \vdash M : A \quad A \leq_{\mathcal{T}} B}{\Gamma \vdash M : B}$	
($\mathtt{U}$)	$\Gamma \vdash M : \mathtt{U}$	if $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$

Note that the parameter $\mathcal{T}$ appears only in the last two rules.

15.2.4. NOTATION. (i) We write $\Gamma \vdash^{\mathcal{T}}_{\cap} M : A$ if $\Gamma \vdash M : A$ is derivable in $\lambda^{\mathcal{T}}_{\cap}$.
(ii) The assertion $\vdash^{\mathcal{T}}_{\cap}$ may also be written as $\vdash^{\mathcal{T}}$, as $\vdash_{\cap}$ or simply as $\vdash$, if there is little danger of confusion.
(iii) $\lambda^{\mathcal{T}}_{\cap}$ may be denoted simply by $\lambda_{\cap}$.

15.2.5. REMARK. Given a type theory $\mathcal{T}$, the following two options are mutually exclusive: either the type assignment system $\lambda^{\mathcal{T}}_{\cap}$ contains the axiom ($\mathtt{U}$) or it does not. For the specific type theories in Figure 15.3, the situation is as follows.

1. For the first ten type theories, i.e. Scott, Park, CDZ, HR, DHM, BCD, AO, Plotkin, Engeler and CDS, we only get the type assignment system with the axiom ($\mathtt{U}$), since they all belong to $\mathrm{TT}^{\mathtt{U}}$.

2. For the remaining three type theories, i.e. HL, CDV and CD, we only get the one without this axiom, since they all belong to $\mathrm{TT}^{-\mathtt{U}}$.

15.2.6. REMARK. As suggested in the introduction of Chapter 14, the type assignment systems with the axiom ($\mathtt{U}$) are closed under β-expansions and can be used to construct models of the $\lambda\mathsf{K}$-calculus. On the other hand the systems without this axiom are closed under $\beta\mathsf{I}$-expansions (not necessarily under β) and can be used to construct models specifically for the $\lambda\mathsf{I}$-calculus.

15.2.7. Example. The statements in this remark will be proved in Exercises 16.3.7 and 16.3.8. Define $\boldsymbol{\omega} \equiv \lambda x.xx$, $\boldsymbol{\Omega} = \boldsymbol{\omega}\boldsymbol{\omega}$, $\mathsf{V} \equiv \lambda yz.\mathsf{K}z(yz)$ and $\mathsf{K}_* \equiv \lambda yz.z$. Then $\mathsf{V} \twoheadrightarrow_\beta \mathsf{K}_*$. Write $\vdash^{\mathcal{T}}$ for $\vdash^{\mathcal{T}}_{\cap}$.

(i) We have the following for arbitrary $A, B \in \mathbb{T}$.

(1) For all $\mathcal{T} \in \mathrm{TT}$.

$$\vdash^{\mathcal{T}} \boldsymbol{\omega} : A \cap (A \to B) \to B.$$
$$\vdash^{\mathcal{T}} \mathsf{V} : (B \to A) \to B \to B.$$
$$\vdash^{\mathcal{T}} \mathsf{K}_* : A \to B \to B.$$

(2) For some $\mathcal{T} \in \mathrm{TT}^{-\mathtt{U}}$, for example for CDV and CD, one has $\not\vdash^{\mathcal{T}} \mathsf{V} : \alpha \to \beta \to \beta$. Conclude that

$$M \twoheadrightarrow_\beta N \;\&\; \Gamma \vdash^{\mathcal{T}} N : A \;\not\Rightarrow\; \Gamma \vdash^{\mathcal{T}} M : A,$$

i.e. in the absence of $\mathtt{U}$ subject expansion fails, even if the expanded term does have a type; this phenomenon already occurs in $\lambda_\to$ (van Bakel [1993]).

(3) For some $\mathcal{T} \in \mathrm{TT}^{-\mathtt{U}}$, for example for HL, CDV and CD, one has for all A

$$\not\vdash^{\mathcal{T}} \mathsf{KI}\boldsymbol{\Omega} : A;$$
$$\not\vdash^{\mathcal{T}} \boldsymbol{\Omega} : A.$$

(4) $\not\vdash^{\mathrm{CD}} \mathsf{I} : ((\alpha \cap \beta) \to \gamma) \to ((\beta \cap \alpha) \to \gamma)$.

(ii) Let $\mathcal{T} \in \mathrm{TT}^{\,\mathtt{U}}$. Then

$$\vdash^{\mathcal{T}} \boldsymbol{\omega} : A \cap (A \to B) \to B.$$
$$\vdash^{\mathcal{T}} \boldsymbol{\Omega} : \mathtt{U}.$$
$$\vdash^{\mathcal{T}} \mathsf{KI}\boldsymbol{\Omega} : (A \to A).$$
$$\vdash^{\mathcal{T}} \mathsf{V} : A \to B \to B.$$
$$\vdash^{\mathcal{T}} \mathsf{K}_* : A \to B \to B.$$

15.2.8. Definition. Define the rules ($\cap$E)

$$\frac{\Gamma \vdash M : (A \cap B)}{\Gamma \vdash M : A} \qquad \frac{\Gamma \vdash M : (A \cap B)}{\Gamma \vdash M : B}$$

Notice that these rules are derived in $\lambda^{\mathcal{T}}_{\cap}$ for all $\mathcal{T}$.

15.2.9. Lemma. *For $\mathcal{T} \in \mathrm{TT}^{-\mathtt{U}}$ one has the following.*

(i) $\Gamma \vdash^{\mathcal{T}}_{\cap} M : A \;\Rightarrow\; \mathrm{FV}(M) \subseteq \mathrm{dom}(\Gamma)$.

(ii) $\Gamma \vdash^{\mathcal{T}}_{\cap} M : A \;\Rightarrow\; (\Gamma \restriction \mathrm{FV}(M)) \vdash M : A$.

Proof. (i), (ii) By induction on the derivation. ■

Notice that $\Gamma \vdash M : A \Rightarrow \mathrm{FV}(M) \subseteq \mathrm{dom}(\Gamma)$ does not hold for $\mathcal{T} \in \mathrm{TT}^{\,\mathtt{U}}$, since by axiom ($\mathtt{U}$) we have $\vdash^{\mathcal{T}}_{\cap} M : \mathtt{U}$ for all M.

Admissible rules

15.2.10. PROPOSITION. *The following rules are admissible in* $\lambda^{\mathcal{T}}_{\cap}$.

$$(\textit{weakening}) \quad \frac{\Gamma \vdash M : A \quad x \notin \Gamma}{\Gamma, x\ B \vdash M : A};$$

$$(\textit{strengthening}) \quad \frac{\Gamma, x\ B \vdash M : A \quad x \notin FV(M)}{\Gamma \vdash M : A};$$

$$(\textit{cut}) \quad \frac{\Gamma, x\ B \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash (M[x := N]) : A};$$

$$(\leq\text{-L}) \quad \frac{\Gamma, x\ B \vdash M : A \quad C \leq_{\mathcal{T}} B}{\Gamma, x\ C \vdash M : A};$$

$$(\rightarrow\text{L}) \quad \frac{\Gamma, y\ B \vdash M : A \quad \Gamma \vdash N : C \quad x \notin \Gamma}{\Gamma, x\ (C \rightarrow B) \vdash (M[y := xN]) : A};$$

$$(\cap\text{L}) \quad \frac{\Gamma, x\ A \vdash M : B}{\Gamma, x\ (A \cap C) \vdash M : B}.$$

Figure 15.5: Various admissible rules.

PROOF. By induction on the structure of derivations. ■

Proofs later on in Part III will freely use the rules of the above proposition.

As we remarked earlier, there are various equivalent alternative presentations of intersection type assignment systems. We have chosen a natural deduction presentation, where $\mathcal{T}$-bases are additive. We could have taken, just as well, a sequent style presentation and replace rule ($\rightarrow$E) with the three rules ($\rightarrow$L), ($\cap$L) and (cut) occuring in Proposition 15.2.10, see Barbanera, Dezani-Ciancaglini and de'Liguoro [1995], Barendregt and Ghilezan [n.d.]. Next to this we could have formulated the rules so that $\mathcal{T}$-bases "multiply". Notice that because of the presence of the type constructor $\cap$, a special notion of *multiplication of* $\mathcal{T}$*-bases* is useful.

15.2.11. DEFINITION (Multiplication of $\mathcal{T}$-bases).

$$\begin{array}{rcl} \Gamma \uplus \Gamma' & = & \{x\ A \cap B \mid x\ A \in \Gamma \text{ and } x\ B \in \Gamma'\} \\ & \cup & \{x\ A \mid x\ A \in \Gamma \text{ and } x \notin \Gamma'\} \\ & \cup & \{x\ B \mid x\ B \in \Gamma' \text{ and } x \notin \Gamma\}. \blacksquare \end{array}$$

Accordingly we define:

$$\Gamma \trianglelefteq \Gamma' \ \Leftrightarrow\ \exists \Gamma''.\, \Gamma \uplus \Gamma'' = \Gamma'.$$

For example, $\{x\ A, y\ B\} \uplus \{x\ C, z\ D\} = \{x\ A \cap C, y\ B, z\ D\}$.

15.2.12. PROPOSITION. *The following rules are admissible in all* $\lambda^{\mathcal{T}}_{\cap}$.

$$\begin{array}{ll} (\textit{multiple weakening}) & \dfrac{\Gamma_1 \vdash M : A}{\Gamma_1 \uplus \Gamma_2 \vdash M : A} \\[2ex] (\textit{relevant} \to\text{E}) & \dfrac{\Gamma_1 \vdash M : A \to B \quad \Gamma_2 \vdash N : A}{\Gamma_1 \uplus \Gamma_2 \vdash MN : B} \\[2ex] (\textit{relevant} \cap\text{I}) & \dfrac{\Gamma_1 \vdash M : A \quad \Gamma_2 \vdash M : B}{\Gamma_1 \uplus \Gamma_2 \vdash M : A \cap B} \end{array}$$

PROOF. By induction on derivations. ■

In Exercise 16.3.23, it will be shown that we can replace rule $(\leq)$ with other more perspicuous rules. This is possible as soon as we will have proved appropriate "inversion" theorems for $\lambda^{\mathcal{T}}_{\cap}$. For some very special theories, one can even omit altogether rule $(\leq)$, provided the remaining rules are reformulated "multiplicatively" with respect to $\mathcal{T}$-bases, see e.g. Di Gianantonio and Honsell [1993]. We shall not follow up this line of investigation.

In $\lambda^{\mathcal{T}}_{\cap}$, assumptions are allowed to appear in the basis without any restriction. Alternatively, we might introduce a *relevant* intersection type assignment system, where only "minimal-base" judgements are derivable, (see Honsell and Ronchi Della Rocca [1992]). Rules like (*relevant* $\to$E) and (*relevant* $\cap$ I), which exploit the above notion of multiplication of bases, are essential for this purpose. Relevant systems are necessary, for example, for giving finitary logical descriptions of qualitative domains as defined in Girard et al. [1989]. We will not follow up this line of research either. See Honsell and Ronchi Della Rocca [1992].

Special type assignment for call-by-value λ-calculus

15.2.13. DEFINITION. The type theory EHR with $\mathbb{A}^{\text{EHR}} = \{\mathtt{V}\}$ and the extra rule $(\to)$ and axioms $(\to\cap)$ and

$$A \to B \leq \mathtt{V}.$$

The type assignment system $\lambda^{\text{EHR}}_{\cap\mathtt{V}}$ is defined by the axiom and rules of Definition 15.2.3 with the extra axiom

$$(\mathtt{V}) \qquad \Gamma \vdash (\lambda x.M) : \mathtt{V}.$$

The type theory EHR has a top, namely $\mathtt{V}$, but it is not an element of $\mathrm{TT}^{\mathtt{U}}$. Hence $\lambda^{\text{EHR}}_{\cap\mathtt{V}}$ does not contain the axiom $(\mathtt{U})$. Note also that the axiom $(\mathtt{V})$ is different from $(\mathtt{U})$. This type assignment system has been extensively studied in Ronchi Della Rocca and Paolini [2004].

15.3. Type structures

Intersection type structures

Remember that a type algebra $\mathcal{A}$, see Definition **??**, is of the form $\mathcal{A} = \langle |\mathcal{A}|, \to \rangle$, i.e. just an arbitrary set $|\mathcal{A}|$ with a binary operation $\to$ on it.

15.3.1. DEFINITION. (i) A *meet semi-lattice* (without universe) is a structure

$$\mathcal{M} = \langle |\mathcal{M}|, \leq, \cap \rangle,$$

such that $|\mathcal{M}|$ is a countable set, $\leq$ is a partial order, for all $A, B \in |\mathcal{M}|$ the element $A \cap B$ (meet) is the greatest lower bound of A and B. $\text{MSL}^{-\mathtt{U}}$ is the set of meet semi-lattices.

(ii) A *meet semi-lattice with universe is a similar structure*

$$\mathcal{M} = \langle |\mathcal{M}|, \leq, \cap, \mathtt{U} \rangle,$$

with $\mathtt{U}$ *the (unique) top element of* $\mathcal{M}$. $MSL^{\mathtt{U}}$ *is the set of meet semi-lattices with universe.*

(iii) $MSL = MSL^{\mathtt{U}} \cup MSL^{-\mathtt{U}}$ *is the set of meet semi-lattices with or without universe.*

(iv) *We have* $MSL^{\mathtt{U}} \cap MSL^{-\mathtt{U}} = \emptyset$, *as the signatures are different.*

15.3.2. DEFINITION. (i) An *(intersection) type structure* (without universe) is a type algebra with the additional structure of a meet semi-lattice

$$\mathcal{S} = \langle |\mathcal{S}|, \leq, \cap, \to \rangle.$$

$\text{TS}^{-\mathtt{U}}$ is the set of type structures without universe. The relation $\leq$ and the operation $\to$ have a priori no relation with each other, but in special structures this will be the case.

(ii) A *type structure with universe* $\mathtt{U}$ *is a type algebra that is also a meet semi-lattice with universe.*

$$\mathcal{S} = \langle |\mathcal{S}|, \leq, \cap, \to, \mathtt{U} \rangle.$$

$\text{TS}^{\mathtt{U}}$ *is the set of type structures with universe* $\mathtt{U}$.

(iii) $\text{TS} = \text{TS}^{\mathtt{U}} \cup \text{TS}^{-\mathtt{U}}$ *is the set of type structures with or without universe. As before* $\text{TS}^{-\mathtt{U}} \cap \text{TS}^{\mathtt{U}} = \emptyset$.

NOTATION. (i) As 'intersection' is everywhere in this Part III, we will omit this word and only speak about a *type structure*.

(ii) *Par abus de language* we also use $A, B, C, \ldots$ to denote arbitrary elements of type structures and we write $A \in \mathcal{S}$ for $A \in |\mathcal{S}|$.

If $\mathcal{T}$ is a type theory that is not compatible, like CD and CDS, then $\to$ cannot be defined on the equivalence classes. But if $\mathcal{T}$ is compatible, then one can work on the equivalence classes and obtain a type structure in which $\leq$ is a partial order.

15.3.3. PROPOSITION. *Let $\mathcal{T} \in \mathrm{TT}$ be compatible. Then $\mathcal{T}$ induces a type structure $[\mathcal{T}]$ defined as follows.*

$$[\mathcal{T}] = \langle [\mathbb{T}], \leq, \cap, \to \rangle,$$

by defining on the $=_{\mathcal{T}}$-equivalence classes

$$\begin{array}{rcl} [A] \leq [B] & \iff & A \leq B; \\ [A] \cap [B] & = & [A \cap B]; \\ [A] \to [B] & = & [A \to B]^1 \end{array}$$

where A, B, C range over $\mathbb{T}$. If moreover $\mathcal{T}$ has universe $\mathtt{U}$, then $[\mathcal{T}]$ is a type structure with $[\mathtt{U}]$ as universe.

PROOF. See Remark 15.1.12. ■

Let $\mathcal{T} \in \mathrm{TT}$. Then the type structure $[\mathcal{T}]$ is called a *syntactical* type structure.

15.3.4. PROPOSITION. *Every type structure is isomorphic to a syntactical one.*

PROOF. For a type structure $\mathcal{S}$, define a type theory $\mathrm{Th}(\mathcal{S})$ as follows. Take $\mathbb{A} = \mathbb{A}^{\mathrm{Th}(\mathcal{S})} = \{\underline{c} \mid c \in \mathcal{S}\}$. Then define $g : \mathbb{T}^{\mathbb{A}} \to \mathcal{S}$ by

$$\begin{array}{rcl} g(\underline{c}) & = & c; \\ g(A \to B) & = & g(A) \to g(B); \\ g(A \cap B) & = & g(A) \cap g(B). \end{array}$$

Define $A \leq_{\mathrm{Th}(\mathcal{S})} B \iff g(A) \leq_{\mathcal{S}} g(B)$. Then g induces a bijective morphism $\overline{g} : [\mathrm{Th}(\mathcal{S})] \to \mathcal{S}$ where the inverse f is defined by $f(a) = [\underline{a}]$. Moreover, if $\mathcal{S}$ has a universe $\mathtt{U}$, then $[\mathtt{U}]$ is the universe of $[\mathrm{Th}(\mathcal{S})]$ and $\overline{g}([\underline{\mathtt{U}}]) = \mathtt{U}$. ■

15.3.5. REMARK. (i) Each of the eleven compatible type theories $\mathcal{T}$ in Figure 15.3 may be considered as the intersection type structure $[\mathcal{T}]$. For example Scott can be a name, a type theory or a type structure.

(ii) Although essentially equivalent, type structures and type theories differ in the following. In the theories the types are freely generated from a fixed set of atoms and inequality can be controlled somewhat by choosing the right axioms and rules. This will be exploited in Chapters **??**, **??** and 19. In type structures one has the antisymmetric law $A \leq B \leq A \Rightarrow A = B$, which is in line with the common theory of partial orders. This will be exploited in Chapter 17.

(iii) Note that in a type theories there is up to $=_{\mathcal{T}}$ at most one universe $\mathtt{U}$, as we require it to be the top.

Now the notion of type assignment will also be defined for type structures. These structures arise naturally coming from algebraic lattices that are used towards obtaining a semantics for untyped lambda calculus.

[1] Here we misuse notation in a suggestive way, by using the same notation $\to$ for equivalence classes as for types.

15.3.6. DEFINITION. Let $\mathcal{S} \in \mathrm{TS}$.
(i) The notion of an $\mathcal{S}$-statement $M : A$, an $\mathcal{S}$-declaration $x\ A$, a $\mathcal{S}$-basis and an $\mathcal{S}$-assertion $\Gamma \vdash M : A$ is as in Definition 15.2.1, now for $A \in \mathcal{S}$ an element of the type structure $\mathcal{S}$.

(ii) The notion $\Gamma \vdash^{\mathcal{S}}_{\cap} M : A$ is defined by the same set of axioms and rules of $\lambda^{\mathcal{T}}_{\cap}$ in Definition 15.2.3, where now $\leq_{\mathcal{S}}$ is the inequality of the structure $\mathcal{S}$.

The following result shows that for syntactic type structures type assignment is essentially the same as the one coming from the corresponding lambda theory.

15.3.7. PROPOSITION. *Let $\mathcal{T} \in \mathrm{TT}$ be compatible and write*

$$[\mathcal{T}] = \langle [\mathbb{T}], \leq, \cap, \to(,[\mathtt{U}]) \rangle$$

its corresponding type structure possibly with universe. For a type $A \in \mathcal{T}$ write its equivalence class as $[A] \in [\mathcal{T}]$. For $\Gamma = \{x_1 : B_1, \ldots, x_n : B_n\}$ a $\mathcal{T}$-basis write $[\Gamma] = \{x_1 : [B_1], \ldots, x_n : [B_n]\}$, a $[\mathcal{T}]$-basis. Then

$$\Gamma \vdash^{\mathcal{T}}_{\cap} M : A \iff [\Gamma] \vdash^{[\mathcal{T}]}_{\cap} M : [A].$$

PROOF. ($\Rightarrow$) By induction on the derivation of $\Gamma \vdash^{\mathcal{T}} M : A$. ($\Leftarrow$) Show by induction on the derivation of $[\Gamma] \vdash^{[\mathcal{T}]} M : [A]$ that for all $A' \in [A]$ and $\Gamma' = \{x_1 : B'_1, \ldots, x_n : B'_n\}$, with $B'_i \in [B_i]$ for all $1 \leq i \leq n$, one has

$$\Gamma' \vdash^{\mathcal{T}} M : A'. \blacksquare$$

Using this result we could have defined type assignment first for type structures and then for compatible type theories via translation to the type assignment for its corresponding syntactical type structure, essentially by turning the previous result into a definition.

Categories of meet-semi lattices and type structures

For use in Chapter 17 we will introduce some categories related to given classes of type structures.

15.3.8. DEFINITION. (i) The category $\mathbf{MSL}^{-\mathtt{U}}$ has as objects the elements of $\mathrm{MSL}^{-\mathtt{U}}$ and as morphisms maps $f : \mathcal{M} \to \mathcal{M}'$, preserving $\leq$, $\cap$:

$$\begin{aligned} A \leq B &\Rightarrow f(A) \leq' f(B); \\ f(A \cap B) &= f(A) \cap' f(B). \end{aligned}$$

(ii) $\mathbf{MSL}^{\mathtt{U}}$ has as objects the elements of $\mathrm{MSL}^{\mathtt{U}}$ and as morphisms maps that preserve $\leq$, $\cap$ but also $\mathtt{U}$, i.e. $f(\mathtt{U}) = \mathtt{U}'$.

There is no natural category corresponding to MSL, as it is a hybrid set consisting of structures with and without universe.

15.3.9. Definition. (i) The category $\mathbf{TS}^{-\mathtt{U}}$ has as objects type structures and as morphisms maps $f : \mathcal{S} \to \mathcal{S}'$, preserving $\leq, \cap, \to$:

$$\begin{array}{rcl} A \leq B & \Rightarrow & f(A) \leq' f(B); \\ f(A \cap B) & = & f(A) \cap' f(B); \\ f(A \to B) & = & f(A) \to' f(B). \end{array}$$

(ii) The category $\mathbf{TS}^{\mathtt{U}}$ is as $\mathbf{TS}^{-\mathtt{U}}$, but based on type structures with universe. For morphisms we require preservation of $\leq$, $\cap$, $\to$, but also

$$f(\mathtt{U}) = \mathtt{U}'.$$

Again there is no category corresponding to TS.

The definitions of graph, lazy, natural and proper type theory translate immediately to type structures.

15.3.10. Definition. We define three full subcategories of $\mathbf{TS}^{\mathtt{U}}$ and one full subcategory of $\mathbf{TS}^{-\mathtt{U}}$ by specifying in each case the objects.

(i) $\mathbf{GTS}^{\mathtt{U}}$ whose objects are the graph type structures.
(ii) $\mathbf{LTS}^{\mathtt{U}}$ whose objects are the lazy type structures.
(iii) $\mathbf{NTS}^{\mathtt{U}}$ whose objects are the natural type structures.
(iv) $\mathbf{PTS}^{-\mathtt{U}}$ whose objects are the proper type structures.

15.4. Filters

In this section we define the notions of filter and filter structure.

15.4.1. Definition. (i) Let $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$ and $X \subseteq \mathbb{T}^{\mathcal{T}}$. Then X is a *filter* over $\mathcal{T}$ if the following hold.

(1) $A \in X \;\&\; A \leq B \;\Rightarrow\; B \in X$;
(2) $A, B \in X \;\Rightarrow\; A \cap B \in X$;
(3) X is non-empty.

(ii) Let $\mathcal{T} \in \mathrm{TT}^{-\mathtt{U}}$ and $X \subseteq \mathbb{T}^{\mathcal{T}}$. Then X is a *filter* over $\mathcal{T}$ if the following hold.

(1) $A \in X \;\&\; A \leq B \;\Rightarrow\; B \in X$;
(2) $A, B \in X \;\Rightarrow\; A \cap B \in X$.
(iii) Write $\mathcal{F}^{\mathcal{T}} = \{X \subseteq \mathbb{T}^{\mathcal{T}} \mid X \text{ is a filter over } \mathcal{T}\}$.

If $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$, then filters ar sets of types containing $\mathtt{U}$ and closed under $\leq$ and $\cap$. If $\mathcal{T} \in \mathrm{TT}^{-\mathtt{U}}$, then filters may be empty.

15.4.2. Definition. Let $\mathcal{T} \in \mathrm{TT}$.
(i) For $A \in \mathbb{T}^{\mathcal{T}}$ write $\uparrow A = \{B \in \mathbb{T}^{\mathcal{T}} \mid A \leq B\}$.
(ii) For $X \subseteq \mathbb{T}^{\mathcal{T}}$ define $\uparrow X$ to be the smallest filter over $\mathcal{T}$ containing X.

15.4.3. Remark. (i) If X is non-empty,

$$\uparrow X = \{B \in \mathbb{T}^{\mathcal{T}} \mid \exists n \geq 1\, \exists A_1, \ldots, A_n \in X. A_1 \cap \ldots \cap A_n \leq B\}.$$

(ii) For $X = \emptyset$, we have that

$$\uparrow\emptyset = \begin{cases} \{A \mid A = \mathtt{U}\} = [\mathtt{U}] & \text{if } \mathcal{T} \in \mathrm{TT}^{\,\mathtt{U}} \\ \emptyset & \text{if } \mathcal{T} \in \mathrm{TT}^{\text{-}\mathtt{U}} \end{cases}$$

(iii) $C \in \uparrow\{B_i \mid i \in \mathcal{I} \neq \emptyset\} \iff \exists I \subseteq_{\text{fin}} \mathcal{I}.[I \neq \emptyset \;\&\; \bigcap_{i \in I} B_i \leq C]$.

Complete lattices and the category **ALG** were introduced in Definition 14.2.1.

15.4.4. Proposition. *Let $\mathcal{T} \in \mathrm{TT}$.*

(i) *$\mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, \subseteq \rangle$ is a complete lattice, with for $\mathcal{X} \subseteq \mathcal{F}^{\mathcal{T}}$ the sup is*

$$\bigsqcup \mathcal{X} = \uparrow(\bigcup \mathcal{X})$$

(ii) *For $A \in \mathbb{T}^{\mathcal{T}}$ one has $\uparrow A = \uparrow\{A\}$ and $\uparrow A \in \mathcal{F}^{\mathcal{T}}$.*

(iii) *For $A, B \in \mathbb{T}^{\mathcal{T}}$ one has $\uparrow A \sqcup \uparrow B = \uparrow(A \cap B)$.*

(iv) *For $A_i \in \mathbb{T}^{\mathcal{T}}$ $(i \in \mathcal{I})$ one has $\bigsqcup\{\uparrow A_i \mid i \in \mathcal{I}\} = \uparrow \{A_i \mid i \in \mathcal{I}\}$.*

(v) *For $X \in \mathcal{F}^{\mathcal{T}}$ one has*

$$\begin{array}{rcccl} X & = & \bigsqcup\{\uparrow A \mid A \in X\} & = & \bigsqcup\{\uparrow A \mid \uparrow A \subseteq X\} \\ & = & \bigcup\{\uparrow A \mid A \in X\} & = & \bigcup\{\uparrow A \mid \uparrow A \subseteq X\}. \end{array}$$

(vi) *The set $\mathcal{K}(\mathcal{F}^{\mathcal{T}})$ of finite elements of $\mathcal{F}^{\mathcal{T}}$ is given by*

$$\mathcal{K}(\mathcal{F}^{\mathcal{T}}) = \begin{cases} \{\uparrow A \mid A \in \mathbb{T}^{\mathcal{T}}\} & \text{if } \mathcal{T} \in \mathrm{TT}^{\,\mathtt{U}} \\ \{\uparrow A \mid A \in \mathbb{T}^{\mathcal{T}}\} \cup \emptyset & \text{if } \mathcal{T} \in \mathrm{TT}^{\text{-}\mathtt{U}} \end{cases}$$

(vii) *$\mathcal{F}^{\mathcal{T}} \in$ **ALG**.*

Proof. Easy. ■

Now we introduce the fundamental notion of filter structure. It is of paramount importance in Part III of this book. Since the seminal paper Barendregt et al. [1983], this notion has played a major role in the study of the mathematical semantics of lambda calculus.

15.4.5. Definition. Let $\mathcal{T} \in \mathrm{TT}$. Define

$$\begin{array}{rcll} F^{\mathcal{T}} & \in & [\mathcal{F}^{\mathcal{T}} \to [\mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}]], & \text{and} \\ G^{\mathcal{T}} & \in & [[\mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}] \to \mathcal{F}^{\mathcal{T}}] & \end{array}$$

as follows

$$\begin{array}{rcl} F^{\mathcal{T}}(X)(Y) & = & \uparrow\{B \in \mathbb{T}^{\mathcal{T}} \mid \exists A \in Y.(A \to B) \in X\}; \\ G^{\mathcal{T}}(f) & = & \uparrow\{A \to B \mid B \in f(\uparrow A)\}. \end{array}$$

Then, $\mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle$ is called the *filter structure* over $\mathcal{T}$.

It is easy to show that

$$F^{\mathcal{T}} \in [\mathcal{F}^{\mathcal{T}} \to [\mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}]] \;\&\; G^{\mathcal{T}} \in [[\mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}] \to \mathcal{F}^{\mathcal{T}}].$$

15.4.6. Remark. The items 15.1.18-15.2.12 and 15.4.1-15.4.5 are about type theories, but can be translated immediately to structures and if no $\to$ are involved to meet-semi lattices. For example Proposition 15.1.21 also holds for a proper type structure, hence it holds for Scott, Park, CDZ, HR, DHM, BCD, AO, HL and CDV considered as type structures. Also 15.1.22-15.1.28 immediately yield corresponding valid statements for the corresponding type structures, though the proof for the type theories cannot be translated to proofs for the type structures because they are by induction on the syntactic generation of $\mathbb{T}$ or $\leq$. Also 15.2.7-15.2.12 hold for type structures, as follows immediately from Propositions 15.3.4 and 15.3.7. Finally 15.4.1-15.4.5 can be translated immediately to type structures and meet semi-lattices. In Chapter 17 we work directly with meet semi-lattices and type structures and not with type theories, because there a proper partial order is needed.

An example of the use of this remark is the following easy lemma.

15.4.7. Lemma. [**filters.TT.TS**] *Let $\mathcal{T}$ be a compatible type theory. Then $\mathcal{F}^{\mathcal{T}} \cong \mathcal{F}^{[\mathcal{T}]}$ in the category* **ALG**.

Proof. A filter X over $\mathcal{T}$ is mapped to

$$[X] = \{[A] \mid A \in X\}.$$

This map is 1-1, onto and preserves $\subseteq$. ■

Chapter 16

Basic properties

16.1. Inversion theorems

In the style of Coppo et al. [1984] and Alessi et al. [2003], [2005] we shall isolate special properties which allow to 'reverse' some of the rules of the type assignment system $\vdash^{\mathcal{T}}_{\cap}$, thereby achieving some form of 'generation' and 'inversion' properties. These state necessary and sufficient conditions when an assertion $\Gamma \vdash^{\mathcal{T}} M : A$ holds depending on the form of M and A, see Theorems 16.1.1 and 16.1.9.

16.1.1. Theorem (Inversion Theorem I). *Let $\mathcal{T} \in \mathrm{TT}$ and let $\vdash$ denote $\vdash^{\mathcal{T}}_{\cap}$. If $\mathcal{T} \in \mathrm{TT}^{-\mathtt{U}}$, then the following statements hold unconditionally; if $\mathcal{T} \in \mathrm{TT}^{\,\mathtt{U}}$, then they hold under the assumption that $A \neq \mathtt{U}$ in (i) and (ii).*

$$
\begin{array}{lrcl}
\text{(i)} & \Gamma \vdash x : A & \iff & \Gamma(x) \leq A. \\
\text{(ii)} & \Gamma \vdash MN : A & \iff & \exists k \geq 1\, \exists B_1, \ldots, B_k, C_1, \ldots, C_k \\
 & & & [C_1 \cap \ldots \cap C_k \leq A \;\&\; \forall i \in \{1, \ldots, k\} \\
 & & & \Gamma \vdash M : B_i \to C_i \;\&\; \Gamma \vdash N : B_i]. \\
\text{(iii)} & \Gamma \vdash \lambda x.M : A & \iff & \exists k \geq 1\, \exists B_1, \ldots, B_k, C_1, \ldots, C_k \\
 & & & [(B_1 \to C_1) \cap \ldots \cap (B_k \to C_k) \leq A \\
 & & & \;\&\; \forall i \in \{1, \ldots, k\}. \Gamma, x\, B_i \vdash M : C_i].
\end{array}
$$

Proof. We only treat ($\Rightarrow$) in (i)-(iii), as ($\Leftarrow$) is trivial. Let first consider $\mathcal{T} \in \mathrm{TT}^{-\mathtt{U}}$.

(i) By induction on derivations. We reason according which axiom or rule has been used in the last step. Only axiom (Ax), and rules ($\cap$I), ($\leq$) could have been applied. In the first case one has $\Gamma(x) \equiv A$. In the other two cases the induction hypothesis applies.

(ii) By induction on derivations. By assumption on A and the shape of the term the last applied step has to be rule ($\to$E), ($\leq$) or ($\cap$I). In the first case the last applied rule is

$$
(\to\mathrm{E})\ \frac{\Gamma \vdash M : D \to A \quad \Gamma \vdash N : D}{\Gamma \vdash MN : A}\,.
$$

We can take $k = 1$ and $C_1 \equiv A$ and $B_1 \equiv D$. In the second case the last rule applied is

$$(\leq) \quad \frac{\Gamma \vdash MN : B \quad B \leq A}{\Gamma \vdash MN : A}$$

and the induction hypothesis applies. In the last case $A \equiv A_1 \cap A_2$ and the last applied rule is

$$(\cap\mathrm{I}) \quad \frac{\Gamma \vdash MN : A_1 \quad \Gamma \vdash MN : A_2}{\Gamma \vdash MN : A_1 \cap A_2}.$$

By the induction hypothesis there are B_i, C_i, D_j, E_j, with $1 \leq i \leq k,\ 1 \leq j \leq k'$, such that

$$\begin{array}{ll} \Gamma \vdash M : B_i{\to}C_i, & \Gamma \vdash N : B_i, \\ \Gamma \vdash M : D_j{\to}E_j, & \Gamma \vdash N : D_j, \\ C_1 \cap \ldots \cap C_k \leq A_1, & E_1 \cap \ldots \cap E_{k'} \leq A_2. \end{array}$$

Hence we are done, as $C_1 \cap \ldots \cap C_k \cap E_1 \cap \ldots \cap E_{k'} \leq A$.

(iii) Again by induction on derivations. We only treat the case $A \equiv A_1 \cap A_2$ and the last applied rule is ($\cap$I):

$$(\cap\mathrm{I}) \quad \frac{\Gamma \vdash \lambda x.M : A_1 \quad \Gamma \vdash \lambda x.M : A_2}{\Gamma \vdash \lambda x.M : A_1 \cap A_2}.$$

By the induction hypothesis there are B_i, C_i, D_j, E_j with $1 \leq i \leq k,\ 1 \leq j \leq k'$ such that

$$\begin{array}{ll} \Gamma, x\ B_i \vdash M : C_i, & (B_1{\to}C_1) \cap \ldots \cap (B_k{\to}C_k) \leq A_1, \\ \Gamma, x\ D_j \vdash M : E_j, & (D_1{\to}E_1) \cap \ldots \cap (D_{k'}{\to}E_{k'}) \leq A_2. \end{array}$$

We are done, since $(B_1{\to}C_1){\cap}\ldots{\cap}(B_k{\to}C_k){\cap}(D_1{\to}E_1){\cap}\ldots{\cap}(D_{k'}{\to}E_{k'}) \leq A$.

Now we prove ($\Rightarrow$) in (i)-(iii) for $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$.

(i), (ii) The condition $A \neq \mathtt{U}$ implies that axiom ($\mathtt{U}$) cannot have been used in the last step. Hence the reasoning above suffices.

(iii) If $A = \mathtt{U}$, then ($\Rightarrow$) in (iii) holds as $\mathtt{U} \to \mathtt{U} \leq \mathtt{U}$ and $\Gamma, x\ \mathtt{U} \vdash M : \mathtt{U}$. So we may assume that $A \neq \mathtt{U}$. Then the only interesting rule is ($\cap$I). Condition $A \neq \mathtt{U}$ implies that we cannot have $A_1 = A_2 = \mathtt{U}$. In case $A_1 \neq \mathtt{U}$ and $A_2 \neq \mathtt{U}$ the result follows as above. The other cases are easier. ■

Under some conditions (that will hold for many type theories, notably the ones introduced in Section 15.1), the Inversion Theorem can be restated in a more memorable form. This will be done in Theorem 16.1.9.

16.1.2. Corollary (Subformula property). *Let $\mathcal{T} \in \mathrm{TT}$. Assume*

$$\Gamma \vdash^{\mathcal{T}}_{\cap} M : A \text{ and } N \text{ is a subterm of } M.$$

Then N is typable in an extension $\Gamma^+ = \Gamma, x_1\ B_1, \ldots, x_n\ B_n$ in which also the variables $\{x_1, \ldots, x_n\} = \mathrm{FV}(N) - \mathrm{FV}(M)$ get a type assigned.

Proof. We can write $M \equiv C[N]$. If $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$, then the statement is trivial as $\vdash N : \mathtt{U}$. Otherwise the statement is proved by induction on the structure of $C[\ ]$, using Theorem 16.1.1. ■

16.1.3. Proposition. *Let $\mathcal{T} \in \mathrm{TT}$. Writing $\vdash$ for $\vdash^{\mathcal{T}}_{\cap}$, we have for $y \notin \mathrm{dom}(\Gamma)$*

$$\exists B\,[\Gamma \vdash N : B \;\&\; \Gamma \vdash M[x := N] : A] \;\Rightarrow$$
$$\exists B\,[\Gamma \vdash N : B \;\&\; \Gamma, y\,B \vdash M[x := y] : A].$$

Proof. By induction on the structure of M. ■

In the following definition, the notion of β-soundness is introduced to prove invertibility of the rule ($\to$I) and preservation of β-reduction.

16.1.4. Definition. $\mathcal{T}$ is called *β-sound* if

$\forall k{\geq}1 \forall A_1, \ldots, A_k, B_1, \ldots, B_k, C, D.$

$$(A_1 \to B_1) \cap \ldots \cap (A_k \to B_k) \leq (C \to D) \;\&\; D \neq \mathtt{U} \;\Rightarrow$$
$$C \leq A_{i_1} \cap \ldots \cap A_{i_p} \;\&\; B_{i_1} \cap \ldots \cap B_{i_p} \leq D,$$
$$\text{for some } p \geq 1 \text{ and } 1 \leq i_1, \ldots, i_p \leq k.$$

This definition translates immediately to type structures. The notion of β-soundness is important to prove invertibility of the rule ($\to$I), which is crucial for the next section.

In particular the condition of β-soundness for $k{=}1$ is expressed as follows:

$$B' \neq \mathtt{U} \;\;\&\;\; A \to B \leq A' \to B' \;\Rightarrow\; A' \leq A \;\;\&\;\; B \leq B'.$$

When $B' = \mathtt{U}$ and $A \to B \leq A' \to B'$, the condition of β-soundness does not imply that $A' \leq A$ and $B \leq B'$.

It will be shown that all type theories of Figure 15.3 are β-sound. The proof occupies 16.1.5-16.1.7.

16.1.5. Remark. Note that in a TT every type A can be written uniquely, modulo the order, as

$$A \equiv \alpha_1 \cap \ldots \cap \alpha_n \cap (B_1 \to C_1) \cap \ldots \cap (B_k \to C_k) \qquad (+),$$

i.e. an intersection of atoms ($\alpha_i \in \mathbb{A}$) and arrow types.

For some of our $\mathcal{T}$ the shape (+) in Remark 16.1.5 can be simplified.

16.1.6. Definition. For the type theories $\mathcal{T}$ of Figure 15.3 we define for each $A \in \mathbb{T}^{\mathcal{T}}$ its *canonical form*, notation $\mathrm{cf}(A)$, as follows.

(i) If $\mathcal{T} \in \{\mathrm{BCD}, \mathrm{AO}, \mathrm{Plotkin}, \mathrm{Engeler}, \mathrm{CDV}, \mathrm{CDS}, \mathrm{CD}\}$, then

$$\mathrm{cf}(A) \equiv A.$$

(ii) If $\mathcal{T} \in \{\mathrm{Scott}, \mathrm{Park}, \mathrm{CDZ}, \mathrm{HR}, \mathrm{DHM}, \mathrm{HL}\}$ then the definition is by induction on A. For an atom α the canonical form $\mathrm{cf}(\alpha)$ depends on the type theory

in question; moreover the mapping cf preserves $\to, \cap$ and $\mathtt{U}$.

System $\mathcal{T}$	A	$\mathrm{cf}(A)$
Scott	0	$\mathtt{U}\to 0$
Park	0	$0\to 0$
CDZ, HL	0	$1\to 0$
	1	$0\to 1$
HR	0	$1\to 0$
	1	$(0\to 0)\cap(1\to 1)$
DHM	0	$\mathtt{U}\to 0$
	1	$0\to 1$
All systems	$\mathtt{U}$	$\mathtt{U}$
All systems	$B\to C$	$B\to C$
All systems	$B\cap C$	$\mathrm{cf}(B)\cap\mathrm{cf}(C)$

16.1.7. THEOREM. *All type theories of Figure 15.3 are β-sound.*

PROOF. We prove the following stronger statement (induction loading). Let

$$\begin{array}{rcl} A & \leq & A', \\ \mathrm{cf}(A) & = & \alpha_1\cap\ldots\cap\alpha_n\cap(B_1\to C_1)\cap\ldots\cap(B_k\to C_k), \\ \mathrm{cf}(A') & = & \alpha'_1\cap\ldots\cap\alpha'_{n'}\cap(B'_1\to C'_1)\cap\ldots\cap(B'_{k'}\to C'_{k'}) \end{array}$$

where $n, n' \geq 0, k, k' \geq 1$. Then

$$\begin{array}{l} \forall j\in\{1,\ldots,k'\}.[\,C_{j'}\neq\mathtt{U}\ \Rightarrow \\ \exists p{\geq}1\exists i_1,\ldots,i_p\in\{1,\ldots,k\}.[B'_j\leq B_{i_1}\cap\ldots\cap B_{i_p}\ \&\ C_{i_1}\cap\ldots\cap C_{i_p}\leq C'_j]]. \end{array}$$

The proof of the statement is by induction on the generation of $A\leq A'$. From it β-soundness follows easily. ■

16.1.8. REMARK. From Theorem 16.1.7 it follows immediately that for the compatible theories of Figure 15.3 the corresponding type structures are β-sound.

16.1.9. THEOREM (Inversion Theorem II). *Let $\mathcal{T}\in\mathrm{TT}$. Of the following properties (i) holds in general, (ii) provided that $\mathcal{T}\in\mathrm{PTT}$ and if $\mathcal{T}\in\mathrm{TT}^{\mathtt{U}}$, then $A\neq\mathtt{U}$, and (iii) provided that $\mathcal{T}$ is β-sound.*

$$\begin{array}{lrcl} \text{(i)} & \Gamma, x\,A\vdash x:B & \Longleftrightarrow & A\leq B. \\ \text{(ii)} & \Gamma\vdash(MN):A & \Longleftrightarrow & \exists B\,[\Gamma\vdash M:(B\to A)\ \&\ \Gamma\vdash N:B]. \\ \text{(iii)} & \Gamma\vdash(\lambda x.M):(B\to C) & \Longleftrightarrow & \Gamma, x\,B\vdash M:C. \end{array}$$

PROOF. The proof of each ($\Leftarrow$) is easy. So we only treat ($\Rightarrow$).

(i) If $B\neq\mathtt{U}$, then the conclusion follows from Theorem 16.1.1(i). If $B=\mathtt{U}$, then the conclusion holds trivially.

(ii) Suppose $\Gamma \vdash MN : A$. Then by Theorem 16.1.1(ii) there are $B_1, \ldots, B_k$, $C_1, \ldots, C_k$, with $k \geq 1$, such that $C_1 \cap \ldots \cap C_k \leq A$, $\Gamma \vdash M : B_i \to C_i$ and $\Gamma \vdash N : B_i$ for $1 \leq i \leq k$. Hence $\Gamma \vdash N : B_1 \cap \ldots \cap B_k$ and

$$\begin{aligned} \Gamma \vdash M : \quad & (B_1 \to C_1) \cap \ldots \cap (B_k \to C_k) \\ & \leq (B_1 \cap \ldots \cap B_k) \to (C_1 \cap \ldots \cap C_k) \\ & \leq (B_1 \cap \ldots \cap B_k) \to A, \end{aligned}$$

by Lemma 15.1.21. So we can take $B \equiv (B_1 \cap \ldots \cap B_k)$.

(iii) Suppose $\Gamma \vdash (\lambda x.M) : (B \to C)$. Then Theorem 16.1.1(iii) applies and we have for some $k \geq 1$ and $B_1, \ldots, B_k, C_1, \ldots, C_k$

$$\begin{gathered} (B_1 \to C_1) \cap \ldots \cap (B_k \to C_k) \leq B \to C, \\ \Gamma, x\, B_i \vdash M : C_i \text{ for all } i. \end{gathered}$$

If $C = \mathtt{U}$, then the assertion holds trivially, so let $C \neq \mathtt{U}$. Then by β-soundness there are $1 \leq i_1, \ldots, i_p \leq k$, $p \geq 1$ such that

$$\begin{gathered} B \leq B_{i_1} \cap \ldots \cap B_{i_p}, \\ C_{i_1} \cap \ldots \cap C_{i_p} \leq C. \end{gathered}$$

Applying ($\leq$-L) we get

$$\begin{gathered} \Gamma, x\, B \vdash M : C_{i_j},\ 1 \leq j \leq p, \\ \Gamma, x\, B \vdash M : C_{i_1} \cap \ldots \cap C_{i_p} \leq C. \blacksquare \end{gathered}$$

We give a simple example which shows that in general rule ($\to$E) cannot be reversed, i.e. that if $\Gamma \vdash MN : B$, then it is not always true that there exists A such that $\Gamma \vdash M : A \to B$ and $\Gamma \vdash N : A$.

16.1.10. Example. Let $\mathcal{T}$ = Engeler, one of the intersection type theories of Figure 15.3. Let $\Gamma = \{x\, (\mathtt{c}_0 \to \mathtt{c}_1) \cap (\mathtt{c}_2 \to \mathtt{c}_3), y\, (\mathtt{c}_0 \cap \mathtt{c}_2)\}$. Then one has

$$\Gamma \vdash^{\mathcal{T}}_{\cap} xy : \mathtt{c}_1 \cap \mathtt{c}_3.$$

But for no type B

$$\Gamma \vdash^{\mathcal{T}}_{\cap} x : B \to (\mathtt{c}_1 \cap \mathtt{c}_3) \text{ and } \Gamma \vdash^{\mathcal{T}}_{\cap} y : B.$$

16.1.11. Remark. Nore that in general

$$\Gamma \vdash^{\mathcal{T}}_{\cap} (\lambda x.M) : A \not\Rightarrow \exists B, C.A = (B \to C)\ \&\ \Gamma, x\, B \vdash^{\mathcal{T}}_{\cap} M : C.$$

Consider $\vdash^{\mathrm{BCD}}_{\cap} \mathsf{I} : (\mathtt{c}_1 \to \mathtt{c}_1) \cap (\mathtt{c}_2 \to \mathtt{c}_2)$, with $\mathtt{c}_1, \mathtt{c}_2$ different type atoms.

16.1.12. Proposition. *For all $\mathcal{T}$ in Figure 15.1.14 except* AO *properties (i), (ii) and (iii) of Theorem 16.1.9 hold unconditionally. For $\mathcal{T}$ = AO they hold under the condition that $A \neq \mathtt{U}$ in (ii).*

Proof. Since these $\mathcal{T}$ are proper and β-sound, by Theorem 16.1.7, we can apply Theorem 16.1.9. Moreover, by axiom ($\to$U) we have $\Gamma \vdash^{\mathcal{T}}_{\cap} M : \mathtt{U} \to \mathtt{U}$ for all Γ, M, hence we do not need to assume $A \neq \mathtt{U}$ for $\mathcal{T} \in \{$Scott, Park, CDZ, HR, DHM, BCD$\}$. $\blacksquare$

16.2. Subject reduction and expansion

Various subject reduction and expansion properties are proved, for the classical β, $\beta\mathsf{I}$ and $\boldsymbol{\eta}$ notions of reduction. Other results can be found in Alessi et al. [2003], Alessi et al. [2006]. We consider the following rules.

$$(R\text{-}red) \quad \frac{M \to_R N \quad \Gamma \vdash M : A}{\Gamma \vdash N : A}$$

$$(R\text{-}exp) \quad \frac{M \,{}_R\!\leftarrow N \quad \Gamma \vdash M : A}{\Gamma \vdash N : A}$$

where R is a notion of reduction, notably β-, $\beta\mathsf{I}$, or $\boldsymbol{\eta}$-reduction. If one of these rules holds in $\lambda^{\mathcal{T}}_{\cap}$, we write $\lambda^{\mathcal{T}}_{\cap} \models (R\text{-}\{exp, red\})$, respectively. If both hold we write $\lambda^{\mathcal{T}}_{\cap} \models (R\text{-}cnv)$. These properties will be crucial in Chapters **??** and **??**, where we will discuss (untyped) λ-models induced by these systems.

Recall that $(\lambda x.M)N$ is a $\beta\mathsf{I}$-*redex* if $x \in \mathrm{FV}(M)$, Curry and Feys [1958].

β-conversion

We first investigate when $\lambda^{\mathcal{T}}_{\cap} \models (\beta(\mathsf{I})\text{-}red)$.

16.2.1. Proposition. *Let $\mathcal{T} \in \mathrm{TT}$. Then we have the following.*

(i) $\lambda^{\mathcal{T}}_{\cap} \models (\beta\mathsf{I}\text{-}red) \iff$

$$[\Gamma \vdash^{\mathcal{T}} (\lambda x.M) : (B \to A) \;\&\; x \in \mathrm{FV}(M) \;\Rightarrow\; \Gamma, x\, B \vdash^{\mathcal{T}} M : A].$$

(ii) $\lambda^{\mathcal{T}}_{\cap} \models (\beta\text{-}red) \iff$

$$[\Gamma \vdash^{\mathcal{T}} (\lambda x.M) : (B \to A) \;\Rightarrow\; \Gamma, x\, B \vdash^{\mathcal{T}} M : A].$$

Proof. (i) ($\Rightarrow$) Assume $\Gamma \vdash \lambda x.M : B \to A \;\&\; x \in \mathrm{FV}(M)$, which implies $\Gamma, y\, B \vdash (\lambda x.M)y : A$, by weakening and rule ($\to$E) for a fresh y. Now rule ($\beta\mathsf{I}\text{-}red$) gives us $\Gamma, y\, B \vdash M[x{:=}y] : A$. Hence $\Gamma, x\, B \vdash M : A$.

($\Leftarrow$) Suppose $\Gamma \vdash (\lambda x.M)N : A \;\&\; x \in \mathrm{FV}(M)$, in order to show that $\Gamma \vdash M[x{:=}N] : A$. We may assume $A \neq \mathtt{U}$. Then Theorem 16.1.1(ii) implies $\Gamma \vdash \lambda x.M : B_i \to C_i$, $\Gamma \vdash N : B_i$ and $C_1 \cap \ldots \cap C_k \leq A$, for some $B_1, \ldots, B_k, C_1, \ldots, C_k$. By assumption $\Gamma, x\, B_i \vdash M : C_i$. Hence by rule (*cut*), Proposition 15.2.10, one has $\Gamma \vdash M[x{:=}N] : C_i$. Therefore $\Gamma \vdash M[x{:=}N] : A$, using rules ($\cap\mathsf{I}$) and ($\leq$).

(ii) Similarly. ■

16.2.2. Corollary. *Let $\mathcal{T} \in \mathrm{TT}$ be β-sound. Then $\lambda^{\mathcal{T}}_{\cap} \models (\beta\text{-}red)$.*

Proof. Using Theorem 16.1.9(iii). ■

The converse of Corollary 16.2.2 does not hold. In Definition 18.2.19 we will introduce a type theory that is not β-sound, but nevertheless induces a type assignment system satisfying ($\beta\text{-}red$).

16.2.3. Corollary. *Let $\mathcal{T}$ be one of the* TT *in Figure 15.1.14. Then*

$$\lambda_\cap^{\mathcal{T}} \models (\beta\text{-}red).$$

Proof. By Corollary 16.2.2 and Theorem 16.1.7. ■

Now we investigate when $\lambda_\cap^{\mathcal{T}} \models (\beta\text{-}exp)$. As a warm-up, suppose that $\Gamma \vdash M[x{:=}N] : A$. Then we would like to conclude that N has a type, as it seems to be a subformula, and therefore $\Gamma \vdash (\lambda x.M)N : A$. There are two problems: N may occur several times in $M[x{:=}N]$, so that it has (should have) in fact several types. In the system $\lambda_\rightarrow$ this problem causes the failure of rule $(\beta\text{-}exp)$. But in the intersection type theories one has $N : B_1 \cap \ldots \cap B_k$ if $N : B_1, \ldots, N : B_k$. Therefore $(\lambda x.M)N$ has a type if $M[x{:=}N]$ has one. The second problem arises if N does not occur at all in $M[x{:=}N]$, i.e. if the redex is a $\lambda\mathsf{K}$-redex. We would like to assign as type to N the intersection over an empty sequence, i.e. the universe $\mathtt{U}$. This makes $(\beta\text{-}exp)$ invalid for $\mathcal{T} \in \mathrm{TT}^{-\mathtt{U}}$, but valid for $\mathcal{T} \in \mathrm{TT}^{\,\mathtt{U}}$.

16.2.4. Proposition. *Let $\mathcal{T} \in \mathrm{TT}$. Then we have the following.*

(i) *Suppose $\Gamma \vdash^{\mathcal{T}} M[x{:=}N] : A$. Then*

$$\Gamma \vdash^{\mathcal{T}} (\lambda x.M)N : A \iff N \text{ is typable in context } \Gamma.$$

(ii) $\lambda_\cap^{\mathcal{T}} \models (\beta\mathsf{I}\text{-}exp) \iff \forall \Gamma, M, N, A$ *with* $x \in \mathrm{FV}(M)$

$$[\Gamma \vdash^{\mathcal{T}} M[x{:=}N] : A \Rightarrow N \text{ is typable in context } \Gamma].$$

(iii) $\lambda_\cap^{\mathcal{T}} \models (\beta\text{-}exp) \iff \forall \Gamma, M, N, A$

$$[\Gamma \vdash^{\mathcal{T}} M[x{:=}N] : A \Rightarrow N \text{ is typable in context } \Gamma].$$

Proof. (i) ($\Rightarrow$) By Theorem 16.1.1(ii). ($\Leftarrow$) Let $\Gamma \vdash M[x{:=}N] : A$ and suppose N is typable in context Γ. By Proposition 16.1.3 for some B and a fresh y one has $\Gamma \vdash N : B$ & $\Gamma, y\ B \vdash M[x\ {=}\ y] : A$. Then $\Gamma \vdash \lambda x.M : (B{\rightarrow}A)$ and hence $\Gamma \vdash (\lambda x.M)N : A$.

(ii) Similar and simpler than (iii).

(iii) ($\Rightarrow$) Assume $\Gamma \vdash M[x{:=}N] : A$. Then $\Gamma \vdash (\lambda x.M)N : A$, by $(\beta\text{-}exp)$, hence by (i) we are done. ($\Leftarrow$) Assume $\Gamma \vdash L' : A$, with $L \rightarrow_\beta L'$. By induction on the generation of $L \rightarrow_\beta L'$ we get $\Gamma \vdash L : A$ from (i) and Theorem 16.1.1. ■

16.2.5. Corollary. (i) *Let $\mathcal{T} \in \mathrm{TT}$. Then $\lambda_\cap^{\mathcal{T}} \models (\beta\mathsf{I}\text{-}exp)$.*

(ii) *Let $\mathcal{T} \in \mathrm{TT}^{\,\mathtt{U}}$. Then $\lambda_\cap^{\mathcal{T}} \models (\beta\text{-}exp)$.*

Proof. (i) By the subformula property (Corollary 16.1.2).

(ii) Trivial, since every term has type $\mathtt{U}$. ■

Now we can harvest results towards closure under β-conversion.

16.2.6. Theorem. (i) *Let $\mathcal{T} \in \mathrm{TT}$. Then*

$$\mathcal{T} \text{ is } \beta\text{-sound} \Rightarrow \lambda_\cap^{\mathcal{T}} \models (\beta\mathsf{I}\text{-}cnv).$$

(ii) *Let* $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$. *Then*

$$\mathcal{T} \text{ is } \beta\text{-sound} \Rightarrow \lambda_{\cap}^{\mathcal{T}} \models (\beta\text{-}cnv).$$

PROOF. (i) By Corollaries 16.2.2 and 16.2.5(i).
(ii) By Corollaries 16.2.2 and 16.2.5(ii). ■

16.2.7. COROLLARY. (i) *Let* $\mathcal{T}$ *be a* TT *of Figure 15.1.14. Then*

$$\lambda_{\cap}^{\mathcal{T}} \models (\beta\mathsf{I}\text{-}cnv).$$

(ii) *Let* $\mathcal{T} \in \{\text{Scott, Park, CDZ, HR, DHM, BCD, AO, Plotkin, Engeler, CDS}\}$. *Then* $\lambda_{\cap}^{\mathcal{T}} \models (\beta\text{-}cnv)$.

PROOF. (i) By Theorems 16.1.7 and 16.2.6(i).
(ii) Similarly, by Theorem 16.2.6(ii). ■

$\boldsymbol{\eta}$-conversion

First we give necessary and sufficient conditions for a system $\lambda_{\cap}^{\mathcal{T}}$ to satisfy the rule ($\boldsymbol{\eta}$-*red*).

16.2.8. THEOREM (Characterization of $\boldsymbol{\eta}$-*red*). (i) *Let* $\mathcal{T} \in \mathrm{TT}^{\text{-}\mathtt{U}}$. *Then*

$$\lambda_{\cap}^{\mathcal{T}} \models (\boldsymbol{\eta}\text{-}red) \iff \mathcal{T} \text{ is proper.}$$

(ii) *Let* $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$. *Then*

$$\lambda_{\cap}^{\mathcal{T}} \models (\boldsymbol{\eta}\text{-}red) \iff \mathcal{T} \text{ is natural.}$$

PROOF. (i) Similarly, but simpler than (ii).
(ii) ($\Rightarrow$) Assume $\lambda_{\cap}^{\mathcal{T}} \models (\boldsymbol{\eta}\text{-}red)$ towards ($\to\cap$), ($\to$) and ($\mathtt{U}\to$).
As to ($\to\cap$), one has

$$x\,(A{\to}B) \cap (A{\to}C), y\,A \vdash xy : B \cap C,$$

hence by ($\to$I) it follows that $x\,(A{\to}B) \cap (A{\to}C) \vdash \lambda y.xy : A{\to}(B \cap C)$. Therefore $x\,(A{\to}B) \cap (A{\to}C) \vdash x : A{\to}(B \cap C)$, by ($\boldsymbol{\eta}$-*red*). By Theorem 16.1.9(i) one can conclude $(A{\to}B) \cap (A{\to}C) \leq A{\to}(B \cap C)$.

As to ($\to$), suppose that $A \leq B$ and $C \leq D$, in order to show $B{\to}C \leq A{\to}D$. One has $x\,B{\to}C, y\,A \vdash xy : C \leq D$, so $x\,B{\to}C \vdash \lambda y.xy : A{\to}D$. Therefore by ($\boldsymbol{\eta}$-*red*) it follows that $x\,B{\to}C \vdash x : A{\to}D$ and we are done as before.

As to $\mathtt{U} \leq \mathtt{U}{\to}\mathtt{U}$, notice that $x\,\mathtt{U}, y\,\mathtt{U} \vdash xy : \mathtt{U}$, so we have $x\,\mathtt{U} \vdash \lambda y.xy : \mathtt{U}{\to}\mathtt{U}$. Therefore $x\,\mathtt{U} \vdash x : \mathtt{U}{\to}\mathtt{U}$ and again we are done.

($\Leftarrow$) Let $\mathcal{T}$ be natural. Assume that $\Gamma \vdash \lambda x.Mx : A$, with $x \notin \mathrm{FV}(M)$, in order to show $\Gamma \vdash M : A$. If $A = \mathtt{U}$, we are done. Otherwise,

$$\begin{aligned} \Gamma \vdash \lambda x.Mx : A \quad \Rightarrow \quad & \Gamma, x\,B_i \vdash Mx : C_i, 1 \leq i \leq k, \ \& \\ & (B_1{\to}C_1) \cap \ldots \cap (B_k{\to}C_k) \leq A, \\ & \text{for some } B_1, \ldots, B_k, C_1, \ldots, C_k, \end{aligned}$$

by Theorem 16.1.1(iii). We can suppose that $C_i \neq \mathtt{U}$ for all i. If there exists i such $C_i = \mathtt{U}$ then, by ($\mathtt{U}$) and ($\mathtt{U}\!\to$), we have $(B_i \to C_i) \cap D = \mathtt{U} \cap D = D$, for any type D. On the other hand, there is at least one $C_i \neq \mathtt{U}$, since otherwise $A \geq (B_1\!\to\!\mathtt{U}) \cap \ldots \cap (B_k\!\to\!\mathtt{U}) = \mathtt{U}$, and we would have $A = \mathtt{U}$. Hence by Theorem 16.1.9(ii)

$$\begin{array}{lll}
\Rightarrow & \Gamma, x\ B_i \vdash M : D_i\!\to\!C_i \text{ and} & \\
& \Gamma, x\ B_i \vdash x : D_i, & \text{for some } D_1, \ldots, D_k, \\
\Rightarrow & B_i \leq D_i, & \text{by Theorem 16.1.9(i)}, \\
\Rightarrow & \Gamma \vdash M : (B_i\!\to\!C_i), & \text{by } (\leq\text{-L}) \text{ and } (\to), \\
\Rightarrow & \Gamma \vdash M : ((B_1\!\to\!C_1) \cap \ldots \cap (B_k\!\to\!C_k)) \leq A. \ \blacksquare &
\end{array}$$

16.2.9. Corollary. *Let* $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{BCD}, \text{HL}, \text{CDV}\}$. *Then* $\lambda^{\mathcal{T}}_{\cap} \models (\boldsymbol{\eta}\text{-}red)$.

In order to characterize the admissibility of rule ($\boldsymbol{\eta}$-*exp*), we need to introduce a further condition on type theories. This condition is necessary and sufficient to derive from the basis $x\ A$ the same type A for $\lambda y.xy$, as we will show in the proof of Theorem 16.2.11.

16.2.10. Definition. Let $\mathcal{T} \in \mathrm{TT}$.
(i) $\mathcal{T}$ is called $\boldsymbol{\eta}$-*sound* if for all A there are $k \geq 1$, $m_1, \ldots, m_k \geq 1$ and $B_1, \ldots, B_k$, $C_1, \ldots, C_k$,

$$\begin{pmatrix} D_{11} \ldots D_{1m_1} \\ \ldots \\ D_{k1} \ldots D_{km_k} \end{pmatrix} \text{ and } \begin{pmatrix} E_{11} \ldots E_{1m_1} \\ \ldots \\ E_{k1} \ldots E_{km_k} \end{pmatrix}$$

with

$$\begin{array}{lll}
(B_1\!\to\!C_1) \cap \ldots \cap (B_k\!\to\!C_k) & \leq & A \\
\& \ A & \leq & (D_{11}\!\to\!E_{11}) \cap \ldots \cap (D_{1m_1}\!\to\!E_{1m_1}) \cap \\
& & \ldots \\
& & (D_{k1}\!\to\!E_{k1}) \cap \ldots \cap (D_{km_k}\!\to\!E_{km_k}) \\
\& \ B_i & \leq & D_{i1} \cap \ldots \cap D_{im_i} \ \& \ E_{i1} \cap \ldots \cap E_{im_i} \ \leq \ C_i, \\
& & \text{for } 1 \leq i \leq k.
\end{array}$$

(ii) Let $\mathcal{T} \in \mathrm{TT}^{\,\mathtt{U}}$. Then $\mathcal{T}$ is called $\boldsymbol{\eta}^{\mathtt{U}}$-*sound* if for all $A \neq \mathtt{U}$ at least one of the following two conditions holds.

(1) There are types $B_1, \ldots, B_n$ with $(B_1\!\to\!\mathtt{U}) \cap \ldots \cap (B_n\!\to\!\mathtt{U}) \leq A$;

(2) There are $n \geq k \geq 1$, $m_1, \ldots, m_k \geq 1$ and $B_1, \ldots, B_k$, $C_1, \ldots, C_k$,

$$\begin{pmatrix} D_{11} \ldots D_{1m_1} \\ \ldots \\ D_{k1} \ldots D_{km_k} \end{pmatrix} \text{ and } \begin{pmatrix} E_{11} \ldots E_{1m_1} \\ \ldots \\ E_{k1} \ldots E_{km_k} \end{pmatrix}$$

with

$$\begin{array}{ll} & (B_1{\to}C_1)\cap\ldots\cap(B_k{\to}C_k)\cap \\ & \cap(B_{k+1}{\to}\mathtt{U})\cap\ldots\cap(B_n{\to}\mathtt{U}) \;\leq\; A \\ \& & A \;\leq\; (D_{11}{\to}E_{11})\cap\ldots\cap(D_{1m_1}{\to}E_{1m_1})\cap \\ & \qquad\ldots \\ & \qquad(D_{k1}{\to}E_{k1})\cap\ldots\cap(D_{km_k}{\to}E_{km_k}) \\ \& & B_i \;\leq\; D_{i1}\cap\ldots\cap D_{im_i} \;\&\; E_{i1}\cap\ldots\cap E_{im_i} \;\leq\; C_i, \\ & \text{for } 1\leq i\leq k. \end{array}$$

The validity of $\boldsymbol{\eta}$-expansion can be characterized as follows.

16.2.11. THEOREM (Characterization of $\boldsymbol{\eta}$-*exp*). (i) *Let* $\mathcal{T}\in\mathrm{TT}^{\text{-}\mathtt{U}}$. *Then*

$$\lambda^{\mathcal{T}}_{\cap}\models(\boldsymbol{\eta}\text{-}exp)\iff\mathcal{T}\text{ is }\boldsymbol{\eta}\text{-sound.}$$

(ii) *Let* $\mathcal{T}\in\mathrm{TT}^{\,\mathtt{U}}$. *Then*

$$\lambda^{\mathcal{T}}_{\cap}\models(\boldsymbol{\eta}\text{-}exp)\iff\mathcal{T}\text{ is }\boldsymbol{\eta}^{\mathtt{U}}\text{-sound.}$$

PROOF. (i) ($\Rightarrow$) Assume $\lambda^{\mathcal{T}}_{\cap}\models(\boldsymbol{\eta}\text{-exp})$. As $x\,A\vdash x:A$, by assumption we have $x\,A\vdash\lambda y.xy:A$. From Theorem 16.1.1(iii) it follows that $x\,A,y\,B_i\vdash xy:C_i$ and $(B_1{\to}C_1)\cap\ldots\cap(B_k{\to}C_k)\leq A$ for some B_i,C_i. By Theorem 16.1.1(ii) for each i there exist D_{ij},E_{ij}, such that for each j one has $x\,A,y\,B_i\vdash x:(D_{ij}{\to}E_{ij})$, and $x\,A,y\,B_i\vdash y:D_{ij}$ and $E_{i1}\cap\ldots\cap E_{im_i}\leq C_i$. Hence by Theorem 16.1.1(i) we have $A\leq(D_{ij}{\to}E_{ij})$ and $B_i\leq D_{ij}$ for all i and j. Therefore we obtain the condition of Definition 16.2.10(i).

($\Leftarrow$) Suppose that $\Gamma\vdash M:A$ in order to show $\Gamma\vdash\lambda x.Mx:A$, with x fresh. By assumption A satisfies the condition of Definition 16.2.10(i).

$$\begin{array}{ll} & (B_1{\to}C_1)\cap\ldots\cap(B_k{\to}C_k) \;\leq\; A \\ \& & A \;\leq\; (D_{11}{\to}E_{11})\cap\ldots\cap(D_{1m_1}{\to}E_{1m_1})\cap \\ & \qquad\ldots \\ & \qquad(D_{k1}{\to}E_{k1})\cap\ldots\cap(D_{km_k}{\to}E_{km_k}) \\ \& & B_i \;\leq\; D_{i1}\cap\ldots\cap D_{im_i} \;\&\; E_{i1}\cap\ldots\cap E_{im_i} \;\leq\; C_i, \\ & \text{for } 1\leq i\leq k. \end{array}$$

By rule ($\leq$) for all i,j we have $\Gamma\vdash M:D_{ij}{\to}E_{ij}$ and so $\Gamma,x\,D_{ij}\vdash Mx:E_{ij}$ by rule ($\to$E). From ($\leq$-L), ($\cap$I) and ($\leq$) we get $\Gamma,x\,B_i\vdash Mx:C_i$ and this implies $\Gamma\vdash\lambda x.Mx:B_i{\to}C_i$, using rule ($\to$I). So we can conclude by ($\cap$I) and ($\leq$) that $\Gamma\vdash\lambda x.Mx:A$.

(ii) The proof is nearly the same as for (i). ($\Rightarrow$) Again we get $x\,A,y\,B_i\vdash xy:C_i$ and $(B_1{\to}C_1)\cap\ldots\cap(B_k{\to}C_k)\leq A$ for some B_i,C_i. If all $C_i=\mathtt{U}$, then A satisfies the first condition of Definition 16.2.10(ii). Otherwise, consider the i such that $C_i\neq\mathtt{U}$ and reason as in the proof of ($\Rightarrow$) for (i).

($\Leftarrow$) Suppose that $\Gamma\vdash M:A$ in order to show $\Gamma\vdash\lambda x.Mx:A$, with x fresh. If A satisfies the first condition of Definition 16.2.10(ii), that is $(B_1{\to}\mathtt{U})\cap\ldots\cap(B_n{\to}\mathtt{U})\leq A$, then by ($\mathtt{U}$) it follows that $\Gamma,x\,B_i\vdash Mx:\mathtt{U}$, hence $\Gamma\vdash\lambda x.Mx:(B_1{\to}\mathtt{U})\cap\ldots\cap(B_n{\to}\mathtt{U})\leq A$. Now let A satisfy the second condition. Then the proof is similar to that for ($\Leftarrow$) in (i). ■

For most intersection type theories of interest the condition of $\boldsymbol{\eta}^{(\mathtt{U})}$-soundness is deduced from the following proposition.

16.2.12. PROPOSITION. *Let $\mathcal{T} \in \mathrm{TT}$ be proper, with set $\mathbb{A}$ of atoms.*

(i) *$\mathcal{T}$ is $\boldsymbol{\eta}$-sound* $\iff$ $\forall \alpha \in \mathbb{A}\, \exists k \geq 1\, \exists B_1, \ldots, B_k, C_1, \ldots, C_k$
$\alpha = (B_1 \to C_1) \cap \ldots \cap (B_k \to C_k)$.

(ii) *Let $\mathcal{T} \in \mathrm{TT}^{\,\mathtt{U}}$. Then*

$$\begin{array}{lll} \mathcal{T} \text{ is } \boldsymbol{\eta}^{\mathtt{U}}\text{-sound} & \iff & \forall \alpha \in \mathbb{A}[\mathtt{U} \to \mathtt{U} \leq \alpha \vee \exists k \geq 1 \exists B_1, \ldots, B_k, C_1, \ldots, C_k \\ & & [(B_1 \to C_1) \cap \ldots \cap (B_k \to C_k) \cap (\mathtt{U} \to \mathtt{U}) \leq \alpha \\ & & \;\&\; \alpha \leq (B_1 \to C_1) \cap \ldots \cap (B_k \to C_k)]]. \end{array}$$

(iii) *Let $\mathcal{T} \in \mathrm{NTT}^{\mathtt{U}}$. Then*

$$\mathcal{T} \text{ is } \boldsymbol{\eta}^{\mathtt{U}}\text{-sound} \iff \mathcal{T} \text{ is } \boldsymbol{\eta}\text{-sound}.$$

PROOF. (i) ($\Rightarrow$) Suppose $\mathcal{T}$ is $\boldsymbol{\eta}$-sound. Let $\alpha \in \mathbb{A}$. Then α satisfies the condition of Definition 16.2.10(i), for some $B_1, \ldots, B_k$, $C_1, \ldots, C_k$, $D_{11}, \ldots, D_{1m_1}, \ldots, D_{k1}, \ldots, D_{km_1}$, $E_{11}, \ldots, E_{1m_1}, \ldots, E_{k1}, \ldots, E_{km_k}$. By ($\to\cap$) and ($\to$), using Proposition 15.1.21, it follows that

$$\begin{array}{rcl} A & \leq & (D_{11} \cap \ldots \cap D_{1m_1} \to E_{11} \cap \ldots \cap E_{1m_1}) \cap \ldots \cap \\ & & (D_{k1} \cap \ldots \cap D_{km_k} \to E_{k1} \cap \ldots \cap E_{km_k}) \\ & \leq & (B_1 \to C_1) \cap \ldots \cap (B_k \to C_k), \end{array}$$

hence $A =_{\mathcal{T}} (B_1 \to C_1) \cap \ldots \cap (B_k \to C_k)$.

($\Leftarrow$) By induction on the generation of A one can show that A satisfies the condition of $\boldsymbol{\eta}$-soundness. The case $A_1 \to A_2$ is trivial and the case $A \equiv A_1 \cap A_2$ follows by the induction hypothesis and Rule (mon).

(ii) Similarly. Note that $(\mathtt{U} \to \mathtt{U}) \leq (B \to \mathtt{U})$ for all B.

(iii) Immediately by (ii) using rule $(\mathtt{U} \to)$. ■

16.2.13. COROLLARY. (i) *Let $\mathcal{T} \in \{\mathrm{Scott}, \mathrm{Park}, \mathrm{CDZ}, \mathrm{HR}, \mathrm{DHM}, \mathrm{AO}\}$. Then $\mathcal{T}$ is $\boldsymbol{\eta}^{\mathtt{U}}$-sound.*

(ii) HL *is $\boldsymbol{\eta}$-sound.*

PROOF. Easy. For AO in (i) one applies (ii) of the Proposition. ■

16.2.14. COROLLARY. *Let $\mathcal{T} \in \{\mathrm{Scott}, \mathrm{Park}, \mathrm{CDZ}, \mathrm{HR}, \mathrm{DHM}, \mathrm{AO}, \mathrm{HL}\}$. Then*

$$\lambda^{\mathcal{T}}_{\cap} \models (\boldsymbol{\eta}\text{-}exp).$$

PROOF. By the previous Corollary and Theorem 16.2.11. ■

Exercise 16.3.21 shows that the remaining systems of Figure 15.3 do not satisfy ($\boldsymbol{\eta}$-*exp*).

Now we can harvest results towards closure under $\boldsymbol{\eta}$-conversion.

16.2.15. THEOREM. (i) *Let* $\mathcal{T} \in \mathrm{TT}^{-\mathsf{U}}$. *Then*

$$\lambda_\cap^{\mathcal{T}} \models (\boldsymbol{\eta}\text{-}cnv) \iff \mathcal{T} \text{ is proper and } \boldsymbol{\eta}\text{-sound.}$$

(ii) *Let* $\mathcal{T} \in \mathrm{TT}^{\,\mathsf{U}}$. *Then*

$$\lambda_\cap^{\mathcal{T}} \models (\boldsymbol{\eta}\text{-}cnv) \iff \mathcal{T} \text{ is natural and } \boldsymbol{\eta}^{\mathsf{U}}\text{-sound.}$$

PROOF. (i) By Theorems 16.2.8(i) and 16.2.11(i).
(ii) By Theorems 16.2.8(ii) and 16.2.11(ii). ■

16.2.16. THEOREM. *For* $\mathcal{T} \in \{\mathrm{Scott}, \mathrm{Park}, \mathrm{CDZ}, \mathrm{HR}, \mathrm{DHM}, \mathrm{HL}\}$ *one has*

$$\lambda_\cap^{\mathcal{T}} \models (\boldsymbol{\eta}\text{-}cnv).$$

PROOF. By Corollaries 16.2.9 and 16.2.14. ■

16.3. Exercises

16.3.1. Let $\subset$ be the inclusion relation as considered in Remark 15.1.17. Prove that $\mathcal{T}_1 \subset \mathcal{T}_2 \iff \forall \Gamma, M, A[\Gamma \vdash^{\mathcal{T}_1} M : A \Rightarrow \Gamma \vdash^{\mathcal{T}_2} M : A]$.

16.3.2. Show that for each number $n \in \mathbb{N}$ there is a type $A_n \in \mathbb{T}^{\mathrm{CD}}$ such that for the Church numerals $\mathbf{c}_n$ one has $\Gamma \vdash_\cap^{\mathrm{CD}} \mathbf{c}_{n+1} : A_n$, but $\Gamma \not\vdash_\cap^{\mathrm{CD}} \mathbf{c}_n : A_n$.

16.3.3. Show that $\mathsf{S}(\mathsf{KI})(\mathsf{II})$ and $(\lambda x.xxx)\mathsf{S}$ are typable in $\vdash_\cap^{\mathrm{CD}}$.

16.3.4. Derive $\vdash_\cap^{\mathrm{CDZ}} (\lambda x.xxx)\mathsf{S} : 1$ and $y\,0, z\,0 \vdash_\cap^{\mathrm{CDZ}} (\lambda x.xxx)(\mathsf{S}yz) : 0$.

16.3.5. Using the Inversion Theorems show the following.
(i) $\not\vdash_\cap^{\mathrm{CD}} \mathsf{1} : \alpha \to \alpha$, where α is any constant.
(ii) $\not\vdash_\cap^{\mathrm{HL}} \mathsf{K} : 0$.
(iii) $\not\vdash_\cap^{\mathrm{Scott}} \mathsf{I} : 0$.
(iv) $\not\vdash_\cap^{\mathrm{Plotkin}} \mathsf{I} : 0$.

16.3.6. Let $\vdash$ be $\vdash^{\mathrm{CD}}$ and $\leq$ be $\leq_{\mathrm{CD}}$. Show
(i) Let

$$\begin{aligned} A &\equiv \alpha_1 \cap \ldots \cap \alpha_n \cap (B_1 \to C_1) \cap \ldots \cap (B_m \to C_m), \\ A' &\equiv \alpha'_1 \cap \ldots \cap \alpha'_{n'} \cap (B'_1 \to C'_1) \cap \ldots \cap (B_{m'} \to C'_{m'}). \end{aligned}$$

Suppose $A \leq A'$. Then every α'_i is one of $\alpha_1, \ldots, \alpha_n$ and every $B'_i \to C'_i$ is one of $(B_1 \to C_1), \ldots, (B_m \to C_m)$.
(ii) Let $k \geq 1$. Then

$$\begin{aligned} &(B_1 \to C_1) \cap \ldots \cap (B_k \to C_k) \leq (B \to C) \Rightarrow \\ &(B \to C) \equiv (B_j \to C_j), \text{ for some } 1 \leq j \leq k. \end{aligned}$$

(iii) $\Gamma \vdash \lambda x.M : A \to B \Rightarrow \Gamma, x\,A \vdash M : B$.
(iv) $\Gamma \vdash MN : A \Rightarrow \exists B, C\,[\Gamma \vdash M : B \to C \;\&\; \Gamma \vdash N : B]$.

16.3.7. Let $\boldsymbol{\omega} = \lambda x.xx$ and $\boldsymbol{\Omega} = \boldsymbol{\omega}\boldsymbol{\omega}$. For $\vdash$ being $\vdash^{\mathrm{CD}}$ we want to show

$$\begin{array}{ll} \nvdash \boldsymbol{\Omega} : A, & \text{for all types } A; \\ \nvdash \mathsf{KI}\boldsymbol{\Omega} : A, & \text{for all types } A. \end{array}$$

Prove this using the following steps.
(i) $\vdash \boldsymbol{\Omega} : A \;\Rightarrow\; \exists B, C \vdash \boldsymbol{\omega} : (B \to C) \cap B$.
(ii) $\vdash \boldsymbol{\omega} : (B \to C) \cap B \;\Rightarrow\; \exists B', C'. \vdash \boldsymbol{\omega} : (B' \to C') \cap B'$ & B' is a proper subtype of B.

16.3.8. Let $M = \lambda xy.\mathsf{K}x(xy)$. Show that for $\vdash$ being $\vdash^{\mathrm{CD}}$ one has the following.

$$\begin{array}{l} \nvdash M : \alpha \to \beta \to \alpha; \\ \nvdash \mathsf{I} : ((\alpha \cap \beta) \to \gamma) \to ((\beta \cap \alpha) \to \gamma). \end{array}$$

[Hint. Use 16.3.6.]

16.3.9. We say that M and M' have the same types in Γ, notation $M \sim_\Gamma M'$ if

$$\forall A\, [\Gamma \vdash M : A \iff \Gamma \vdash M' : A].$$

Prove that $M \sim_\Gamma M' \;\Rightarrow\; M\vec{N} \sim_\Gamma M'\vec{N}$ for all $\vec{N}$.

16.3.10. Let $\mathcal{T}$ be a β-sound type theory that satisfies (U) and(U$\to$). Prove that for all A, B we have that $A \to B = \mathtt{U} \iff B = \mathtt{U}$.

16.3.11. Using β-soundness, find out whether the following types are related or not with respect to $\leq_{\mathrm{CDZ}}$.

$$(0 \to (1 \to 1) \to 0) \cap ((1 \to 1) \to 1), 0 \to 0 \to 0, (0 \to 0) \to 0 \text{ and } 1 \to (0 \to 0) \to 1.$$

16.3.12. Let $\mathcal{T} \in \{\mathrm{CDZ}, \mathrm{HR}\}$. Consider the sequence of types defined by $A_0 = 0$ and $A_{n+1} = \mathtt{U} \to A_n$. Using Exercise **??** and β-soundness, prove that $\mathcal{T}$ does not have a bottom element at all.

16.3.13. Show that Plotkin is β-sound by checking that it satisfies the following stronger condition.

$$\begin{array}{l} (A_1 \to B_1) \cap \ldots \cap (A_n \to B_n) \leq C \to D \;\Rightarrow \\ \exists k \neq 0\, \exists i_1, \ldots, i_k . 1 \leq i_j \leq n \;\&\; C = A_{i_j} \;\&\; B_{i_1} \cap \ldots \cap B_{i_k} = D. \end{array}$$

16.3.14. Show that Engeler is β-sound by checking that it satisfies the following stronger condition:

$$\begin{array}{l} (A_1 \to B_1) \cap \ldots \cap (A_n \to B_n) \leq C \to D \& D \neq \mathtt{U} \;\Rightarrow \\ \exists k \neq 0\, \exists i_1, \ldots, i_k . 1 \leq i_j \leq n \;\&\; C = A_{i_j} \;\&\; B_{i_1} \cap \ldots \cap B_{i_k} = D. \end{array}$$

16.3.15. Let $\mathbb{A}^{\mathcal{T}} = \{\mathtt{U}, 0\}$ and $\mathcal{T}$ be defined by the axioms and rules of the theories Scott and Park together. Show that $\mathcal{T}$ is not β-sound [Hint: show that $\mathtt{U} \neq 0$].

16.3.16. Prove that Theorem 16.1.9(ii) still holds if the condition of properness is replaced by the following two conditions

$$A \leq_{\mathcal{T}} B \;\Rightarrow\; C \to A \leq_{\mathcal{T}} C \to B$$

$$(A \to B) \cap (C \to D) \leq_{\mathcal{T}} A \cap C \to B \cap D.$$

16.3.17. Show that for $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$ the following condition

$$A \to B =_{\mathcal{T}} \mathtt{U} \to \mathtt{U} \;\Rightarrow\; B =_{\mathcal{T}} \mathtt{U}$$

is necessary for the admissibility of rule (β-*red*) in $\lambda^{\mathcal{T}}_{\cap}$. [Hint: Use Proposition 16.2.1(ii).]

16.3.18. Remember that the systems $\lambda^{\mathrm{Krivine}}_{\cap}$ and $\lambda^{\mathrm{Krivine}^{\mathtt{U}}}_{\cap}$ are defined in Exercise **??**.

(i) Show that rules (β-*red*) and ($\beta\mathsf{I}$-*exp*) are admissible in $\lambda^{\mathrm{Krivine}}_{\cap}$, while ($\beta$-*exp*) is not admissible.

(ii) Show that rules (β-*red*) and (β-*exp*) are admissible in $\lambda^{\mathrm{Krivine}^{\mathtt{U}}}_{\cap}$.

16.3.19. Show that for $\mathcal{T} \in \{\mathrm{AO}, \mathrm{Plotkin}, \mathrm{Engeler}, \mathrm{CDS}, \mathrm{CD}\}$ one has

$$\lambda^{\mathcal{T}}_{\cap} \not\models (\boldsymbol{\eta}\text{-red}).$$

16.3.20. Verify the following.

(i) Let $\mathcal{T} \in \{\mathrm{BCD}, \mathrm{Plotkin}, \mathrm{Engeler}, \mathrm{CDS}\}$. Then $\mathcal{T}$ is not $\boldsymbol{\eta}^{\mathtt{U}}$-sound.

(ii) Let $\mathcal{T} \in \{\mathrm{CDV}, \mathrm{CD}\}$. Then $\mathcal{T}$ is not $\boldsymbol{\eta}$-sound.

16.3.21. Show that for $\mathcal{T} \in \{\mathrm{BCD}, \mathrm{Plotkin}, \mathrm{Engeler}, \mathrm{CDS}, \mathrm{CDV}, \mathrm{CD}\}$ one has

$$\lambda^{\mathcal{T}}_{\cap} \not\models (\boldsymbol{\eta}\text{-exp}).$$

16.3.22. Show that rules ($\boldsymbol{\eta}$-*red*) and ($\boldsymbol{\eta}$-*exp*) are not admissible in the systems $\lambda^{\mathrm{Krivine}}_{\cap}$ and $\lambda^{\mathrm{Krivine}^{\mathtt{U}}}_{\cap}$ as defined in Exercises **??**.

16.3.23. Let $\vdash$ denote derivability in the system obtained from the system $\lambda^{\mathrm{CDV}}_{\cap}$ by replacing rule ($\leq$) by the rules ($\cap$E), see Definition 15.2.8, and adding the rule

$$(R\eta) \quad \frac{\Gamma \vdash \lambda x.Mx : A}{\Gamma \vdash M : A} \quad \text{if } x \notin \mathrm{FV}(M).$$

Show that $\Gamma \vdash^{\mathrm{CDV}}_{\cap} M : A \iff \Gamma \vdash M : A$.

16.3.24. (Barendregt et al. [1983]) Let $\vdash$ denote derivability in the system obtained from $\lambda^{\mathrm{BCD}}_{\cap}$ by replacing rule ($\leq$) by the rules ($\cap$E) and adding ($R\eta$) as defined in Exercise 16.3.23. Verify that

$$\Gamma \vdash^{\mathrm{BCD}}_{\cap} M : A \iff \Gamma \vdash M : A.$$

16.3.25. Let Δ be a basis that is allowed to be infinite. We define $\Delta \vdash M : A$ iff there exists a finite basis $\Gamma \subseteq \Delta$ such that $\Gamma \vdash M : A$.

(i) Show that all the typability rules are derivable except possibly for ($\to$I).

(ii) Suppose $\mathrm{dom}(\Delta)$ is the set of all the variables. Show that the rule ($\to$I) is derivable if it is reformulated as

$$\Delta_x, x\,A \vdash M : B \;\Rightarrow\; \Delta \vdash (\lambda x.M) : (A \to B),$$

with Δ_x the result of removing any $x\,C$ from Δ.

(iii) Reformulate and prove Propositions 15.2.10, 15.2.12, Theorems 16.1.1 and 16.1.9 for infinite bases.

16.3.26. A *multi-basis* Γ is a set of declarations, in which the requirement that

$$x\ A, y\ B \in \Gamma\ \Rightarrow\ x \equiv y\ \Rightarrow\ A \equiv B$$

is dropped. Let Δ be a (possibly infinite) multi-basis. We define $\Delta \vdash M : A$ iff there exists a singled (only one declaration per variable) basis $\Gamma \subseteq \Delta$ such that $\Gamma \vdash M : A$.

(i) Show that $x : \alpha_1, x : \alpha_2 \not\vdash^{\mathrm{CD}} x : \alpha_1 \cap \alpha_2$.

(ii) Show that $x : \alpha_1 \to \alpha_2, x : \alpha_1 \not\vdash^{\mathrm{CD}} xx : \alpha_2$.

(iii) Consider $\Delta = \{x : \alpha_1 \cap \alpha_2, x : \alpha_1\}$;
$A = \alpha_2$;
$B = (\alpha_1 \to \alpha_2 \to \alpha_3) \to \alpha_3$;
$M = \lambda y.yxx$.
Show that $\Delta, x : A \vdash^{\mathrm{CD}} M : B$, but $\Delta \not\vdash^{\mathrm{CD}} (\lambda x.M) : (A \to B)$.

(iv) We say that a multi-basis is closed under $\cap$ if for all $x \in \mathrm{dom}(\Delta)$ the set $\mathcal{X} = \Delta(x)$ is closed under $\cap$, i.e. $A, B \in \mathcal{X}\ \Rightarrow\ A \cap B \in \mathcal{X}$, up to equality of types in the TT under consideration.
Show that all the typability rules of Definition 15.2.3, except for ($\to$I), are derivable for (possibly infinite) multi-bases that are closed under $\cap$.

(v) Let Δ be closed under $\cap$. We define

$$\Delta[x := X] = \{y : \Delta(y) \mid y \neq x\} \cup \{x : A \mid A \in X\}.$$

Prove that the following reformulation of ($\to$I) using principal filters is derivable

$$\frac{\Delta[x :=\uparrow B] \vdash N : C}{\Delta \vdash \lambda x.N : B \to C}.$$

(vi) Prove Propositions 15.2.10, 15.2.12, Theorems 16.1.1 and 16.1.9 for (possible infinite) multi-bases reformulating the statements whenever it is necessary.

(vii) Prove that if $\Delta(x)$ is a filter then $\{A \mid \Delta \vdash x : A\} = \Delta(x)$.

16.3.27. (i) Prove that $F^{\mathcal{T}} \circ G^{\mathcal{T}} \supseteq \mathsf{Id}_{\mathcal{F}^{\mathcal{T}}}$ for all $\mathcal{T} \in \mathrm{TT}$.

(ii) Prove that $F^{\mathcal{T}} \circ G^{\mathcal{T}} \subseteq \mathsf{Id}_{\mathcal{F}^{\mathcal{T}}}$ iff $\mathcal{T}$ is β-sound.

(iii) Construct a natural type theory $\mathcal{T}$ such that $F^{\mathcal{T}} \circ G^{\mathcal{T}} \neq \mathsf{Id}_{\mathcal{F}^{\mathcal{T}}}$.

16.3.28. Let $\mathcal{T} \in \mathrm{TT}^{-\mathtt{U}}$.

(i) Prove that $G^{\mathcal{T}} \circ F^{\mathcal{T}} \supseteq \mathsf{Id}_{\mathcal{F}^{\mathcal{T}}}$ iff $\mathcal{T}$ is proper.

(ii) Prove that $G^{\mathcal{T}} \circ F^{\mathcal{T}} \subseteq \mathsf{Id}_{\mathcal{F}^{\mathcal{T}}}$ iff $\mathcal{T}$ is $\boldsymbol{\eta}$-sound.

16.3.29. Let $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$.

(i) Prove that $G^{\mathcal{T}} \circ F^{\mathcal{T}} \supseteq \mathsf{Id}_{\mathcal{F}^{\mathcal{T}}}$ iff $\mathcal{T}$ is natural.

(ii) Prove that $G^{\mathcal{T}} \circ F^{\mathcal{T}} \subseteq \mathsf{Id}_{\mathcal{F}^{\mathcal{T}}}$ iff $\mathcal{T}$ is $\boldsymbol{\eta}^{\mathtt{U}}$-sound.

Chapter 17

Type and Lambda Structures

21.2.2008:897

This Chapter makes connections between convenient categories for models of the untyped lambda calculus and those of certain type structures. The main result that is needed later in Section 18.3 is the equivalence between the categories of natural type structures on the one hand and natural lambda structures on the other hand. This can be found in Section 17.2.

As a warm-up, a proto-type result in this direction is Corollary 17.1.23, stating the equivalence between the categories $\mathbf{MSL}^{\mathsf{U}}$ of meet semi-lattices with universe and $\mathbf{ALG}_a$ of algebraic lattices with additive morphisms that preserve compactness. The idea is that by definition in an algebraic lattice D an element d is fully determined by its compact (defined in 14.2.1(iii)) approximations

$$d = \bigsqcup\{a \mid a \sqsubseteq d \;\&\; a \text{ compact}\}.$$

Writing $\mathcal{K}(D) = \{a \in D \mid a \text{ compact}\}$ and $\uparrow a = \{d \in D \mid a \sqsubseteq d\}$ one has

$$a \sqsubseteq b \iff \uparrow b \subseteq \uparrow a.$$

Therefore it is natural to write for $a, b \in \mathcal{K}(D)$

$$b \leq a \iff a \sqsubseteq b.$$

The complete lattice D can be reconstructed from the meet semi-lattice $\mathcal{S}$ by a general construction that assigns to $\mathcal{S}$ the collection of *filters* (defined in 15.4.1 for type theories, see also 15.4.6 for the translation to type structures) $\mathcal{F}^{\mathcal{S}}$, forming an algebraic lattice. We will show the isomorphisms

$$\begin{aligned} \mathcal{S} &\cong \mathcal{K}(\mathcal{F}^{\mathcal{S}}); \\ D &\cong \mathcal{F}^{\mathcal{K}(D)}. \end{aligned}$$

In fact the isomorphisms are functional and constitute an equivalence between the categories $\mathbf{MSL}^{\mathsf{U}}$ and $\mathbf{ALG}_a$. Similarly, for the strict case, the equivalence between $\mathbf{MSL}^{\text{-}\mathsf{U}}$ and $\mathbf{ALG}_a^s$ is proved. See Fig. 17.1. For the injectivity of the map establishing $\mathcal{S} \cong \mathcal{K}(\mathcal{F}^{\mathcal{S}})$ in Proposition 17.1.22 one needs that $\leq$ is a partial order. Hence the results of this section do not work for type theories in general.

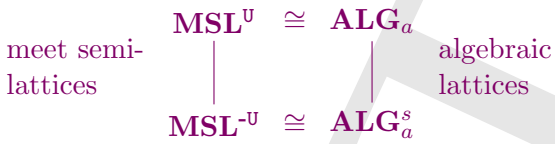


Figure 17.1: Equivalences proved in Section 17.1

To interpret untyped λ-terms the objects D in **ALG** are not enough. We need some more structure, D enriched to $\langle D, F, G\rangle$, where $F : D \to [D \to D]$ and $G : [D \to D] \to D$. These are called lambda structures. There is a canonical way to define such a pair F, G for D being a filter structure obtained from a type theory, see Definition 15.4.5 and Exercise 16.3.27. Then we always have

$$F \circ G \sqsupseteq \mathsf{Id}_D$$

This does not yet imply the validity of the β-axiom

$$(\lambda x.M)N = M[x := N] \qquad (\beta)$$

For β to hold it suffices to require

$$F \circ G = \mathsf{Id}_D \qquad (\text{fun-}\beta)$$

Looking for the 'right' category of type structures corresponding to lambda structures there are other considerations. The axioms $(\rightarrowtail)$, $(\to\cap)$ and $(\mathtt{U}\to)$ are natural considering the intended semantics of $\lambda_\to$ given by Scott [1975b] generalized to $\lambda_\cap$. The 'natural type structures', satisfying $(\to)$, $(\to\cap)$ and $(\mathtt{U}\to)$, exactly correspond to 'natural lambda structures' satisfying

$$\begin{array}{lll} (1) & F \circ G & \sqsupseteq \mathsf{Id}_{[D\to D]} \\ (2) & G \circ F & \sqsubseteq \mathsf{Id}_D \end{array}$$

as suggested in Exercise 16.3.29.Moreover, these axioms suffice to turn the connection between type structures and lambda structures into a categorical equivalence, see Section 17.2.

$$\mathbf{NTS}^{\mathtt{U}} \cong \mathbf{NLS}$$

Figure 17.2: Equivalence proved in Section 17.2

This, however, does not automatically give rise to λ-models. But the lambda structures induced by type structures (arising from the natural type theories) presented in Table 15.3 all satisfy (fun-β) and hence are λ-models. See Exercise 16.3.27 for a natural lambda structure that is not a λ-model.

Weakening one of the axioms for natural type structures, taking $(\mathtt{U}_{\text{lazy}})$ instead of $(\mathtt{U}\to)$, we obtain the 'lazy type structures'. This category is equivalent with that of 'lazy lambda structures' in which there is a weaker version of (2). These give rise to models of the lazy λ-calculus in which the order of a term

(essentially the maximum number of consecutive λ's it can have on its head) is preserved under equality, see Section 18.4.

After this success of the type structures, one may hope to find appropriate ones whose filter models are isomorphic to the well-known graph λ-models $P\omega$ and D_A. This can be done, but we cannot rely on $P\omega$ or D_A as lambda structures for the following reason. The essence of the graph models is that there are maps

$$i : \mathcal{K}(D) \times \mathrm{Prime}(D) \to \mathrm{Prime}(D),$$

where

$$\mathrm{Prime}(D) = \{d \in D \mid \forall A \subseteq D. d \sqsubseteq \bigsqcup A \;\Rightarrow\; \exists a \in A. d \sqsubseteq a\}.$$

In the case of $P\omega$ and D_A the subset $\mathrm{Prime}(D)$ consists of the singletons. Although we can define from these maps i appropriate F_i, G_i, one cannot reconstruct the i from F_i, G_i.

Therefore we want to incorporate i into the definition of a relevant category of structures. In order to do this, we generalize i to a coding of pairs of compact elements

$$Z : \mathcal{K}(D) \times \mathcal{K}(D) \to \mathcal{K}(D)$$

in order to be able to relate it in a simple way to type structures: $Z(a, b)$ roughly corresponds to $(a \to b)$. In this way we obtain the so-called zip structures $\langle D, Z\rangle$. For these structures one can define a corresponding lambda structure $\langle D, F_Z, G_Z\rangle$ by

$$\begin{aligned} F_Z(x)(y) &= \bigsqcup\{b \mid \exists a \sqsubseteq y. Z(a, b) \sqsubseteq x\},\\ G_Z(f) &= \bigsqcup\{Z(a, b) \mid b \sqsubseteq f(a)\}. \end{aligned}$$

Finally the type structures whose filter models are $P\omega$ and D_A, respectively, can defined by weakening the conditions $(\to)$, $(\to\cap)$.

Even if we can establish an equivalence between the categories of type and zip structures, we loose the equivalence between the general categories of type structures and lambda structures. This situation is studied in detail in Sections 17.3 and 17.4. The equivalence between all variants of type structures and zip structures is perfect, but as stated before not between those of zip structures and lambda structures.

Summarizing, there are three groups of categories of structures: the type, zip and lambda structures. The type structures are expansions of meet semi-lattices with the extra operator $\to$;the zip and lambda structures are expansions of algebraic lattices D with respectively a map Z merging ('zipping') two compact elements into one or the extra structure of a pair $\langle F, G\rangle$ with $F : D \to [D \to D]$ and $G : [D \to D] \to D$. Each time we also consider the so called strict case, in which the objects of the categories may miss a top element or where the top element is treated in a special way. Within each group there are five relevant categories, explained above. The categorical equivalences are displayed in Figures 17.1, 17.2 and 17.3.

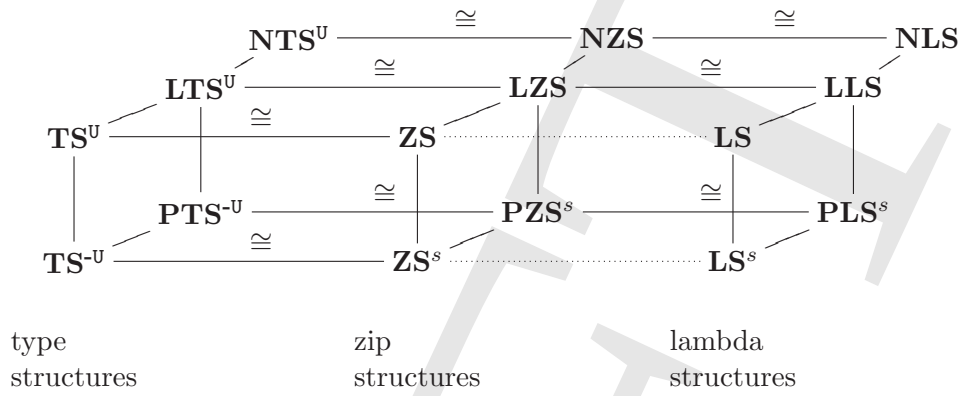


Figure 17.3: Equivalences proved in Sections 17.3 and 17.4

17.1. Meet semi-lattices and algebraic lattices

The results of this section are either known or relatively simple modifications of known ones, see Gierz et al. [1980]???.

Categories of meet semi-lattices

Remember the following notions, see Definitions 15.3.8-15.3.10. The category **MSL** has as objects at most countable meet semi-lattices and as morphisms maps preserving $\leq$ and $\cap$.

The category $\mathbf{MSL}^{\mathtt{U}}$ is as **MSL**, but based on top meet semi-lattices. So now morphisms also should preserve the top.

The category $\mathbf{TS}^{\mathtt{-U}}$ has as objects the at most countable type structures and as morphisms maps $f : \mathcal{S} \to \mathcal{S}'$, preserving $\leq, \cap, \to$.

The category $\mathbf{TS}^{\mathtt{U}}$ is as $\mathbf{TS}^{\mathtt{-U}}$, but based on top type structures. Now also morphisms should preserve the top.

In Definition 15.3.10 we defined three full subcategories of $\mathbf{TS}^{\mathtt{U}}$ and one of $\mathbf{TS}^{\mathtt{-U}}$, by specifying in each case the objects: $\mathbf{GTS}^{\mathtt{U}}$ with as objects the graph top type structures; $\mathbf{LTS}^{\mathtt{U}}$ with as objects the lazy top type structures; $\mathbf{NTS}^{\mathtt{U}}$ with as objects the natural top type structures; $\mathbf{PTS}^{\mathtt{-U}}$ with as objects the proper type structures.

Categories of algebraic lattices

The following has already been given in Definition 14.2.1, but now we treat it in greater detail.

17.1.1. Definition. (i) A *complete lattice* is a poset $D = (D, \sqsubseteq)$ such that for arbitrary $X \subseteq D$ the supremum $\bigsqcup X \in D$ exists. Then one has also a *top* element $\top_D = \bigsqcup D$, a *bottom* element $\bot_D = \bigsqcup \emptyset$, arbitrary *infima*

$$\bigsqcap X = \bigsqcup \{z \mid \forall x \in X. z \sqsubseteq x\}$$

and the *sup* and *inf* of two elements

$$x \sqcup y = \bigsqcup\{x, y\}, \quad x \sqcap y = \bigsqcap\{x, y\}.$$

(ii) A subset $Z \subseteq D$ is called *directed* if Z is non-empty and

$$\forall x, y \in Z\ \exists z \in Z. x \sqsubseteq z\ \&\ y \sqsubseteq z.$$

(iii) An element $d \in D$ is called *compact* (also sometimes called *finite* in the literature) if for every directed $Z \subseteq D$ one has

$$d \sqsubseteq \bigsqcup Z \ \Rightarrow\ \exists z \in Z. d \sqsubseteq z.$$

Note that $\bot_D$ is always compact and if d, e are compact, then so is $d \sqcup e$[1].

(iv) $\mathcal{K}(D) = \{d \in D \mid d \text{ is compact}\}$.

(v) $\mathcal{K}^s(D) = \mathcal{K}(D) - \{\bot_D\}$.

(vi) D is called an *algebraic lattice* if

$$\forall x \in D. x = \bigsqcup\{e \in \mathcal{K}(D) \mid e \sqsubseteq x\}.$$

D is called an *ω-algebraic lattice* if moreover $\mathcal{K}(D)$ is countable.

Instead of $d \in D$ or $X \subseteq D$ we often write $d \in D$ or $X \subseteq D$, respectively. When useful we will decorate $\sqsubseteq, \bigsqcup, \bigsqcap, \bot, \top, \sqcup$ and $\sqcap$ with D, e.g. $\sqsubseteq_D$ etcetera. In this Chapter $a, b, c, d \ldots$ always denote compact elements in lattices. Generic elements are denoted by $x, y, z \ldots$.

17.1.2. Definition. Let D be an ω-algebraic lattice.

(i) On $\mathcal{K}(D)$ define

$$d \leq e \iff e \sqsubseteq d.$$

(ii) For $x \in D$, write

$$K(x) = \{d \in \mathcal{K}(D) \mid d \sqsubseteq x\}.$$

The following connects the notion of a compact element to the notion of compact subset of a topological space.

17.1.3. Lemma. *Let D be a complete lattice.*

(i) *$a \in D$ is compact iff*

$$\forall Z \subseteq D. [d \sqsubseteq \bigsqcup Z \ \Rightarrow\ \exists Z_0 \subseteq Z. [Z_0 \text{ is finite } \&\ a \sqsubseteq \bigsqcup Z_0]].$$

(ii) *If a, b are compact, then $a \sqcup b$ is compact.*

(iii) *For $a, b \in \mathcal{K}(D)$ one has $a \sqcap_{\mathcal{K}(D)} b = a \sqcup_D b$.*

(iv) $(\mathcal{K}(D), \leq) \in \mathbf{MSL}^{\mathtt{U}}$.

[1] In general it is not true that if $d \sqsubseteq e \in \mathcal{K}(D)$, then $d \in \mathcal{K}(D)$; take for example $\omega + 1$ in the ordinal $\omega + \omega = \{0, 1, 2, \ldots\ \omega, \omega + 1, \omega + 2, \ldots\}$. It is compact, but ω $(\sqsubseteq \omega + 1)$ is not.

PROOF. (i) ($\Rightarrow$) Suppose $d \in D$ is compact. Given $Z \subseteq D$, let

$$Z^+ = \{\bigsqcup Z_0 \mid Z_0 \subseteq Z \ \&\ Z_0 \text{ finite}\}.$$

Then $Z \subseteq Z^+, \bigsqcup Z = \bigsqcup Z^+$ and Z^+ is directed. Hence

$$\begin{array}{lll} d \sqsubseteq \bigsqcup Z & \Rightarrow & d \sqsubseteq \bigsqcup Z^+ \\ & \Rightarrow & \exists z^+ \in Z^+ . d \sqsubseteq z^+ \\ & \Rightarrow & \exists Z_0 \subseteq Z . d \sqsubseteq \bigsqcup Z_0 \ \&\ Z_0 \text{ is finite.} \end{array}$$

($\Leftarrow$) Suppose $d \sqsubseteq \bigsqcup Z$ with $Z \subseteq D$ directed. By the condition $d \sqsubseteq \bigsqcup Z_0$ for some finite $Z_0 \subseteq Z$. If Z_0 is non-empty, then by the directedness of Z there exists a $z \in Z$ such that $z \sqsupseteq \bigsqcup Z_0 \sqsupseteq d$. If Z_0 is empty, then $d = \bot_D$ and we can take an arbitrary element z in the non-empty Z satisfying $d \sqsubseteq z$.

(ii) If $a \sqcup b$ is covered by the union of a family Z, then each of a, b is covered, hence by a finite subset of Z, by (i). Therefore also $a \sqcup b$ is covered by a finite subset of Z.

(iii) By (ii) $(a \sqcup_D b) \in \mathcal{K}(D)$; now turn things around.

(iv) Immediate from (iii), noticing that $\mathtt{U}_{\mathcal{K}(D)} = \bot_D \in \mathcal{K}(D)$. ■

Instead of $\sqcap_{\leq}$ we often write $\cap_{\leq}$ or simply $\cap$.

17.1.4. DEFINITION. Let D, D' be complete lattices and $f : D \to D'$.

(i) f is called *(Scott) continuous* iff for all directed $X \subseteq D$ one has

$$f(\bigsqcup X) = \bigsqcup f(X) \ (= \bigsqcup \{f(x) \mid x \in X\}).$$

(ii) $[D \to D'] = \{f : D \to D' \mid f \text{ is Scott continuous function}\}$.

(iii) f is called *strict* iff $f(\bot_D) = \bot_{D'}$.

(iv) Write $[D \to_{\mathrm{s}} D']$ for the collection of continuous strict maps.

17.1.5. PROPOSITION. *Let D, D' be algebraic lattices.*

(i) *Let $f \in [D \to D']$. Then for $x \in D$*

$$f(x) = \bigsqcup \{f(a) \mid a \in \mathcal{K}(x)\}.$$

(ii) *Let $f, g \in [D \to D']$. Suppose $f \restriction \mathcal{K}(D) = g \restriction \mathcal{K}(D)$. Then $f = g$.*

PROOF. (i) Use that $x = \bigsqcup \{a \mid a \in \mathcal{K}(x)\}$ is a directed sup and that f is continuous.

(ii) By (i). ■

17.1.6. DEFINITION. The category **ALG** has as objects the ω-algebraic complete lattices and as morphisms the continuous maps.

17.1.7. DEFINITION. (i) $[D \to D']$ is partially ordered pointwise as follows.

$$f \sqsubseteq g \iff \forall x \in D . f(x) \sqsubseteq g(x).$$

(ii) If $e \in D$, $e' \in D'$, then $e \mapsto e'$ is the *step function* defined by

$$\begin{aligned}(e \mapsto e')(d) &= e', && \text{if } e \sqsubseteq d;\\ &= \bot_{D'}, && \text{otherwise.}\end{aligned}$$

17.1.8. LEMMA. *$[D \to D']$ is a complete lattice with*

$$(\bigsqcup_{f \in X} f)(d) = \bigsqcup_{f \in X} f(d).$$

17.1.9. LEMMA. *For $d, e \in D$, $d', e' \in D'$ and $f \in [D \to D']$ one has*

(i) *d compact $\Rightarrow$ $d \mapsto d'$ is continuous.*
(ii) *$d \mapsto d'$ is continuous and $d' \neq \bot$ $\Rightarrow$ d is compact.*
(iii) *d' compact $\iff$ $d \mapsto d'$ compact.*
(iv) *$d' \sqsubseteq f(d) \iff (d \mapsto d') \sqsubseteq f$.*
(v) *$e \sqsubseteq dd' \sqsubseteq e' \Rightarrow (d \mapsto d') \sqsubseteq (e \mapsto e')$.*
(vi) *$(d \mapsto d') \sqcup (e \mapsto e') \sqsubseteq (d \sqcap e) \mapsto (d' \sqcup e')$.*

PROOF. Easy. ■

17.1.10. LEMMA. *For all $e, d_1, \ldots, d_n \in D$, $e', d'_1, \ldots, d'_n \in D'$*

$$(e \mapsto e') \sqsubseteq (d_1 \mapsto d'_1) \sqcup \ldots \sqcup (d_n \mapsto d'_n) \iff$$
$$\iff \exists I \subseteq \{1, \ldots, n\}\, [\textstyle\bigsqcup_{i \in I} d_i \sqsubseteq e \;\&\; e' \sqsubseteq \bigsqcup_{i \in I} d'_i].$$

Clearly in $(\Rightarrow)$ we have $I \neq \emptyset$ if $e' \neq \bot_{D'}$.

PROOF. Easy. ■

17.1.11. PROPOSITION. *Let $D, D' \in \mathbf{ALG}$.*

(i) *For $f \in [D \to D']$ one has $f = \bigsqcup\{a \mapsto a' \mid a' \sqsubseteq f(a),\ a \in \mathcal{K}(D), a' \in \mathcal{K}(D')\}$.*
(ii) *Let $D \in \mathbf{ALG}$ and let $f : [D \to D']$ be compact. Then*

$$f = (a_1 \mapsto a'_1) \sqcup \ldots \sqcup (a_n \mapsto a'_n),$$

for some $a_1, \ldots, a_n \in \mathcal{K}(D), a'_1, \ldots, a'_n \in \mathcal{K}(D')$.

(iii) *$[D \to D'] \in \mathbf{ALG}$.*

PROOF. (i) It suffices to show that RHS and LHS are equal when applied to an arbitrary element $d \in D$.

$$\begin{aligned}f(d) &= f(\bigsqcup\{a \mid a \sqsubseteq d \;\&\; a \in \mathcal{K}(D)\})\\ &= \bigsqcup\{f(a) \mid a \sqsubseteq d \;\&\; a \in \mathcal{K}(D)\}\\ &= \bigsqcup\{a' \mid a' \sqsubseteq f(a) \;\&\; a \sqsubseteq d \;\&\; a \in \mathcal{K}(D), a' \in \mathcal{K}(D')\}\\ &= \bigsqcup\{(a \mapsto a')(d) \mid a' \sqsubseteq f(a) \;\&\; a \sqsubseteq d \;\&\; a \in \mathcal{K}(D), a' \in \mathcal{K}(D')\}\\ &= \bigsqcup\{(a \mapsto a')(d) \mid a' \sqsubseteq f(a) \;\&\; a \in \mathcal{K}(D), a' \in \mathcal{K}(D')\}\\ &= \bigsqcup\{(a \mapsto a') \mid a' \sqsubseteq f(a) \;\&\; a \in \mathcal{K}(D), a' \in \mathcal{K}(D')\}(d).\end{aligned}$$

(ii) For f compact one has $f = \bigsqcup\{a \mapsto a' \mid a' \sqsubseteq f(a) \;\&\; a \in \mathcal{K}(D), a' \in \mathcal{K}(D')\}$, by (i). Hence by Lemma 17.1.3(i) for some $a_1, \ldots, a_n \in \mathcal{K}(D), a'_1, \ldots, a'_n \in \mathcal{K}(D')$

$$f = (a_1 \mapsto a'_1) \sqcup \ldots \sqcup (a_n \mapsto a'_n). \tag{17.1}$$

(iii) It remains to show that there are only countably many compact elements in $[D \to D]$. Since $\mathcal{K}(D)$ is countable, there are only countably many expressions in the RHS of (17.1). (The cardinality is $\leq \Sigma_n n.\aleph_0^2 = \aleph_0$.) Therefore there are countable many compact $f \in [D \to D]$. (There may be more expressions on the RHS for one f, but this results in less compact elements.) ■

17.1.12. Definition. (i) The category $\mathbf{ALG}_a$ has the same objects as $\mathbf{ALG}$ and as morphisms $\mathbf{ALG}_a(D, D')$ maps $f : D \to D'$ that satisfy the properties 'compactness preserving' and 'additive':

$$\begin{array}{ll} \text{(cmp-pres)} & \forall a \in \mathcal{K}(D).f(a) \in \mathcal{K}(D'); \\ \text{(add)} & \forall X \subseteq D.f(\bigsqcup X) = \bigsqcup f(X). \end{array}$$

(ii) The category $\mathbf{ALG}_a^s$ has the same objects as $\mathbf{ALG}_a$ and as morphisms $\mathbf{ALG}_a^s(D, D')$ maps $f : D \to D'$ satisfying (cmp-pres), (add) and

$$\text{(s)} \quad \forall d \in D.[f(d) = \bot_{D'} \;\Rightarrow\; d = \bot_D].$$

17.1.13. Remark. (i) Note that the requirement (add) implies that a morphism f is continuous (only required to preserve sups for directed subsets X) and strict, i.e. $f(\bot_D) = \bot_{D'}$.

(ii) Remember that $\mathcal{K}^s(D) = \mathcal{K}(D) - \{\bot_D\}$. Note that $\mathbf{ALG}_a^s(D, D')$ consists of maps satisfying

$$\begin{array}{ll} \text{(cmp-pres}^{\text{s}}\text{)} & \forall a \in \mathcal{K}^s(D).f(a) \in \mathcal{K}^s(D'); \\ \text{(add)} & \forall X \subseteq D.f(\bigsqcup X) = \bigsqcup f(X). \end{array}$$

(iii) By contrast to Proposition 17.1.11(iii) $\mathbf{ALG}_a(D, D') \notin \mathbf{ALG}_a$, because from (i) of that Proposition it follows that the set of compactness preserving functions is not closed under taking the supremum.

We need some Lemmas.

17.1.14. Lemma. (i) *Let $f : D \to D'$ be a continuous function with $D, D' \in \mathbf{ALG}$. Then, for any $X \subseteq D$, $b' \in \mathcal{K}(D')$,*

$$b' \sqsubseteq f(\bigsqcup X) \;\Leftrightarrow\; \exists Z \subseteq_{\mathit{fin}} X \cap \mathcal{K}(D).b' \sqsubseteq f(\bigsqcup Z).$$

(ii) *A map $f : D \to D'$ satisfies (add) iff f is Scott continuous and*

$$\forall X \subseteq_{\mathit{fin}} \mathcal{K}(D).m(\bigsqcup X) = \bigsqcup m(X). \tag{1}$$

Proof. (i) Note that $\Xi = \{\bigsqcup Z \mid Z \subseteq_{\text{fin}} X \cap \mathcal{K}(D)\}$ is a directed set and $\bigsqcup \Xi = \bigsqcup X$. Moreover, by monotonicity of f, also the set $\{f(\bigsqcup Z) \mid Z \subseteq_{\text{fin}} X \cap \mathcal{K}(D)\}$ is directed. Therefore

$$\begin{array}{lll} b' \sqsubseteq f(\bigsqcup X) & \Leftrightarrow & b' \sqsubseteq f(\bigsqcup \Xi) \\ & \Leftrightarrow & b' \sqsubseteq \bigsqcup f(\Xi), \qquad \text{since } f \text{ is continuous,} \\ & \Leftrightarrow & b' \sqsubseteq \bigsqcup\{f(\bigsqcup Z) \mid Z \subseteq_{\text{fin}} X \cap \mathcal{K}(D)\}, \quad \text{by definition of } \Xi, \\ & \Leftrightarrow & \exists Z \subseteq_{\text{fin}} X \cap \mathcal{K}(D).b' \sqsubseteq f(\bigsqcup Z), \qquad \text{since } b' \text{ is compact.} \end{array}$$

(ii) The non-trivial direction is to show, assuming f is Scott continuous and satisfies (1), that f is additive. By monotonicity of f we only need to show for all $X \subseteq D$

$$f(\bigsqcup X) \sqsubseteq \bigsqcup f(X).$$

As D' is algebraic, it suffices to assume $b' \sqsubseteq f(\bigsqcup X)$ and conclude $b' \sqsubseteq \bigsqcup f(X)$. By (i) $\exists Z \subseteq_{\text{fin}} X \cap \mathcal{K}(D).b' \sqsubseteq f(\bigsqcup Z) = \bigsqcup f(Z)$, so $b' \sqsubseteq \bigsqcup f(X)$. ■

17.1.15. LEMMA. *Let $\mathcal{S} \in \mathbf{MSL}^{\mathtt{U}}$ be a meet semi-lattice, $I \neq \emptyset$ and $s, t, s_i \in \mathcal{S}$. Then*

(i) *In $\mathcal{F}^{\mathcal{S}}$ we have*

$$\bigsqcup_{i \in I} \uparrow s_i = \uparrow \bigcap_{i \in I} s_i.$$

(ii) *In $\mathcal{K}(\mathcal{F}^{\mathcal{S}})$ we have*

$$\bigcap_{i \in I} \uparrow s_i = \uparrow \bigcap_{i \in I} s_i,$$

where the $\bigcap$ denote the infima in respectively $\mathcal{S}$ and $\mathcal{K}(\mathcal{S})$.

(iii) $\uparrow s \leq_{\mathcal{K}(\mathcal{F}^{\mathcal{S}})} \uparrow t \;\Leftrightarrow\; s \leq_{\mathcal{S}} t$.

PROOF. From the definitions. ■

17.1.16. LEMMA. (i) *Let $\mathcal{X} \subseteq \mathcal{F}^{\mathcal{K}(D)}$. Then taking sups in D one has*

$$\bigsqcup(\bigcup \mathcal{X}) = \bigsqcup_{X \in \mathcal{X}} (\bigsqcup X).$$

(ii) *Let $\theta \subseteq \mathcal{K}(D)$ be non-empty. Then taking the sups in D one has*

$$\bigsqcup(\uparrow\theta) = \bigsqcup \theta.$$

PROOF. (i) Realising that a sup (in D) of a union of $\{Y_i\}_{i \in I} \subseteq \mathcal{K}(D)$ is the sup of the sups $\bigsqcup Y_i$, one has

$$\bigsqcup(\bigcup_{i \in I} Y_i) = \bigsqcup_{i \in I} (\bigsqcup Y_i).$$

The result follows by making an α-conversion $[i := X]$ and taking $I = \mathcal{X}$ and $Y_X = X$.

(ii) $\uparrow\theta$ is obtained from θ by taking extensions and intersections in $\mathcal{K}(D)$. Now the order in this $\mathbf{MSL}^{\mathtt{U}}$ is the reverse of the one induced by D, therefore $\uparrow\theta$ is obtained by taking smaller elements and unions (in D). But then taking the big union the result is the same. ■

We now will establish the following equivalences of categories.

$$\begin{aligned} \mathbf{MSL}^{\mathtt{U}} &\cong \mathbf{ALG}_a; \\ \mathbf{MSL}^{\text{-}\mathtt{U}} &\cong \mathbf{ALG}_a^s. \end{aligned}$$

Equivalence between $\mathbf{MSL}^{\mathsf{U}}$ and $\mathbf{ALG}_a$

We now define the functors establishing the equivalences between categories of meet semi-lattices and complete algebraic lattices. Remember that 15.4.1-15.4.4 can be translated immediately to meet semi-lattices.

17.1.17. DEFINITION. We define a map $\mathsf{Flt} : \mathbf{MSL}^{\mathsf{U}} \to \mathbf{ALG}_a$, that will turn out to be a functor, as follows.

(i) On objects $\mathcal{S} \in \mathbf{MSL}^{\mathsf{U}}$ one defines

$$\mathsf{Flt}(\mathcal{S}) = \langle \mathcal{F}^{\mathcal{S}}, \subseteq \rangle.$$

(ii) On morphisms $f : \mathcal{S} \to \mathcal{S}'$ one defines $\mathsf{Flt}(f) : \mathcal{F}^{\mathcal{S}} \to \mathcal{F}^{\mathcal{S}'}$ by

$$\mathsf{Flt}(f)(X) = \{s' \mid \exists s \in X. f(s) \leq s'\}.$$

17.1.18. LEMMA. *Let* $f \in \mathbf{MSL}^{\mathsf{U}}(\mathcal{S}, \mathcal{S}')$.

(i) *For* $X \subseteq \mathcal{S}$, *one has* $\mathsf{Flt}(f)(\uparrow X) = \uparrow \{f(s) \mid s \in X\}$.

(ii) *For* $s \in \mathcal{S}$ *one has* $\mathsf{Flt}(f)(\uparrow s) = \uparrow f(s)$. ■

PROOF. We only prove (ii).

$$\begin{array}{rcll} \mathsf{Flt}(f)(\uparrow s) & = & \{s' \mid \exists t \in \uparrow s. f(t) \leq s'\}, & \\ & = & \{s' \mid \exists t \geq s. f(t) \leq s'\}, & \\ & = & \{s' \mid f(s) \leq s'\}, & \text{since } f \text{ is monotone,} \\ & = & \uparrow f(s). & \end{array}$$

17.1.19. PROPOSITION. Flt *is a functor from* $\mathbf{MSL}^{\mathsf{U}}$ *to* $\mathbf{ALG}_a$.

PROOF. We have to prove that Flt transforms a morphism in $\mathbf{MSL}^{\mathsf{U}}$ into a morphism in $\mathbf{ALG}_a$. Let $f \in \mathbf{MSL}^{\mathsf{U}}(\mathcal{S}, \mathcal{S}')$, $\uparrow s \in K(\mathcal{F}^{\mathcal{S}})$. By Lemma 17.1.18(ii) $\mathsf{Flt}(f)(\uparrow s) = \uparrow f(s)$, which is compact in $\mathcal{F}^{\mathcal{S}'}$, hence $\mathsf{Flt}(f)$ satisfies (cmp-pres).

$\mathsf{Flt}(f)$ satisfies (add). Indeed, by Lemma 17.1.14(ii) and the fact that $\mathsf{Flt}(f)$ is trivially Scott continuous, it is enough to prove that it commutes with finite joins of compact elements. Let I be non-empty. We have

$$\begin{array}{rcll} \mathsf{Flt}(f)(\bigsqcup_{i \in I} \uparrow s_i) & = & \mathsf{Flt}(f)(\uparrow \bigcap_{i \in I} s_i), & \text{by Lemma 17.1.15(ii),} \\ & = & \uparrow f(\bigcap_{i \in I} s_i), & \text{by Lemma 17.1.18(ii),} \\ & = & \uparrow \bigcap_{i \in I} f(s_i), & \text{since } f \text{ commutes with } \cap, \\ & = & \bigsqcup_{i \in I} \uparrow f(s_i), & \text{by Lemma 17.1.15(ii),} \\ & = & \bigsqcup_{i \in I} \mathsf{Flt}(f)(\uparrow s_i) & \text{by Lemma 17.1.18(ii).} \end{array}$$

If I is empty, and U, U' are respectively the universal tops of $\mathcal{S}, \mathcal{S}'$, then

$$\begin{array}{rcll} \mathsf{Flt}(f)(\bigsqcup_{\emptyset} \uparrow s_i) & = & \mathsf{Flt}(f)(\uparrow \mathsf{U}), & \\ & = & \uparrow f(\mathsf{U}), & \text{by Lemma 17.1.18(ii),} \\ & = & \uparrow \mathsf{U}', & \text{since } f \text{ preserves tops,} \\ & = & \bigsqcup_{\emptyset} \mathsf{Flt}(f)(\uparrow s_i). & \end{array}$$

So $\mathsf{Flt}(f)$ satisfies (add). ■

It is possible to leave out conditions (cmp-pres) and (add), obtaining the category **ALG**. Then one needs to consider *approximable maps* as morphisms in the category $\mathbf{MSL}^{\mathtt{U}}$. See Abramsky [1991].

17.1.20. DEFINITION. We define a map $\mathsf{Cmp} : \mathbf{ALG}_a \to \mathbf{MSL}^{\mathtt{U}}$, that will turn out to be a functor, as follows.

(i) On objects $D \in \mathbf{ALG}_a$ one defines Cmp by

$$\mathsf{Cmp}(D) = (\mathcal{K}(D), \leq).$$

(ii) On morphisms $f \in \mathbf{ALG}_a(D, D')$ one defines $\mathsf{Cmp}(f)$ by

$$\mathsf{Cmp}(f)(d) = f(d).$$

17.1.21. LEMMA. *The map* Cmp *is a functor.*

PROOF. Let $f \in \mathbf{ALG}_a(D, D')$. Note that $\mathsf{Cmp}(f) = f \restriction \mathcal{K}(D) : \mathcal{K}(D) \to \mathcal{K}(D')$, by (cmp-pres). By the fact that f is additive one has $f(\bot_D) = \bot_{D'}$, which is $f(\mathtt{U}_{\mathcal{K}(D)}) = \mathtt{U}_{\mathcal{K}(D')}$, and

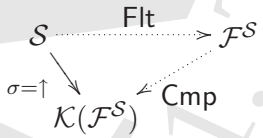
$$f(a \cap_{\mathcal{K}(D)} b) = f(a \sqcup_D b) = f(a) \sqcup_{D'} f(b) = f(a) \cap_{\mathcal{K}(D)} f(b).$$

Also f is monotonic as it is continuous. ■

In the remainder of the present section we write $\leq$ instead of $\leq_{\mathcal{K}(\mathcal{F}^{\mathcal{S}})}$.

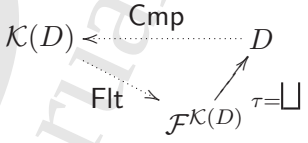
Now we will show that the functors Flt and Cmp establish an equivalence between the categories $\mathbf{MSL}^{\mathtt{U}}$ and $\mathbf{ALG}_a$.

17.1.22. PROPOSITION. (i) *Let* $\mathcal{S} \in \mathbf{MSL}^{\mathtt{U}}$. *Then* $\sigma = \sigma_{\mathcal{S}} : \mathcal{S} \to \mathcal{K}(\mathcal{F}^{\mathcal{S}})$ *defined by* $\sigma_{\mathcal{S}}(s) = \uparrow s$ *is an* $\mathbf{MSL}^{\mathtt{U}}$ *isomorphism.*



Therefore $\mathcal{S} \cong \mathcal{K}(\mathcal{F}^{\mathcal{S}})$.

(ii) *Let* $D \in \mathbf{ALG}_a$. *Then* $\tau = \tau_D : \mathcal{F}^{\mathcal{K}(D)} \to D$ *defined by* $\tau(X) = \bigsqcup X$, *where* $\bigsqcup$ *is taken in* D, *is an* $\mathbf{ALG}_a$ *isomorphism with inverse* $\tau^{-1} : D \to \mathcal{F}^{\mathcal{K}(D)}$ *defined by* $\tau^{-1}(x) = \{c \in \mathcal{K}(D) \mid c \sqsubseteq x\}$.



Therefore $D \cong \mathcal{F}^{\mathcal{K}(D)}$.

PROOF. (i) By Proposition 15.4.4(v) σ is a surjection. It is also 1-1, since $\uparrow s = \uparrow t \;\Rightarrow\; s \leq t \leq s \;\Rightarrow\; s = t$. (This is the place where we need that $\leq$ is a partial order, not only a pre-order.) Moreover, σ preserves $\leq$:

$$\begin{array}{rcll} s \leq t & \iff & \uparrow t \subseteq \uparrow s & \\ & \iff & \uparrow s \leq \uparrow t, & \text{by definition of } \leq \text{ on } \mathcal{K}(\mathcal{F}^{\mathcal{S}}). \end{array}$$

Also σ preserves $\cap$:

$$\begin{array}{rcll} \sigma(s \cap t) & = & \uparrow(s \cap t) & \\ & = & \uparrow s \sqcup \uparrow t & \text{in } \mathcal{F}^{\mathcal{S}}, \text{ by 15.4.4(iii),} \\ & = & \uparrow s \cap \uparrow t & \text{in } \mathcal{K}(\mathcal{F}^{\mathcal{S}}), \text{ by definition of } \leq \text{ on } \mathcal{K}(\mathcal{F}^{\mathcal{S}}), \\ & = & \sigma(s) \cap \sigma(t). & \end{array}$$

Then $\sigma(\mathtt{U}_{\mathcal{S}}) = \uparrow \mathtt{U}_{\mathcal{S}} = \mathtt{U}_{\mathcal{K}(\mathcal{F}^{\mathcal{S}})}$, since $\{\mathtt{U}_{\mathcal{S}}\}$ is the $\subseteq$-smallest filter; hence σ preserves tops.

Finally σ^{-1} preserves $\leq, \sqcap_{\mathcal{K}(\mathcal{F}^{\mathcal{S}})}$ and $\mathtt{U}_{\mathcal{K}(\mathcal{F}^{\mathcal{S}})}$, as by Lemma 15.4.4(v) an element $c \in \mathcal{K}(\mathcal{F}^{\mathcal{S}})$ is of the form $c = \uparrow s$, with $s \in \mathcal{S}$ and $\sigma^{-1}(\uparrow s) = s$.

(ii) We have $\tau \circ \tau^{-1} = \mathbf{1}_D$ and $\tau^{-1} \circ \tau = \mathbf{1}_{\mathcal{F}^{\mathcal{K}(D)}}$:

$$\begin{array}{rcll} \tau(\tau^{-1}(x)) & = & \bigsqcup\{c \in \mathcal{K}(D) \mid c \sqsubseteq x\} & \\ & = & x, & \text{since } D \in \mathbf{ALG}_a. \\ \tau^{-1}(\tau(X)) & = & \{c \mid c \sqsubseteq \bigsqcup X\} & \\ & = & \{c \mid c \in X\}, & \text{since one has} \\ c \sqsubseteq_D \bigsqcup X & \iff & \exists x \in X\ c \sqsubseteq_D x, & \text{as } c \text{ is compact and } X \subseteq \mathcal{K}(D) \subseteq D \text{ is} \\ & & & \text{a filter w.r.t. } \leq, \text{ so directed w.r.t. } \sqsubseteq \\ & \iff & \exists x \in X\ x \leq_{\mathcal{K}(D)} c & \\ & \iff & c \in X, & \text{as } X \text{ is a filter on } \mathcal{K}(D). \end{array}$$

We still have to show that τ and τ^{-1} are morphisms. One easiliy sees that τ satisfies (cmp-pres). The map τ is also additive, i.e. $\bigsqcup \tau(\mathcal{X}) = \tau(\bigsqcup \mathcal{X})$ for arbitrary $\mathcal{X} \subseteq \mathcal{F}^{\mathcal{K}(D)}$. Indeed,

$$\begin{array}{rcll} \bigsqcup \tau(\mathcal{X}) & = & \bigsqcup\limits_{X \in \mathcal{X}} (\bigsqcup X), & \text{by definition of } \tau, \\ & = & \bigsqcup(\bigcup \mathcal{X}), & \text{by Proposition 17.1.16(i),} \\ & = & \bigsqcup(\uparrow(\bigcup \mathcal{X})), & \text{by Proposition 17.1.16(ii),} \\ & = & \tau(\uparrow(\bigcup \mathcal{X})), & \text{by definition of } \tau, \\ & = & \tau(\bigsqcup \mathcal{X}), & \text{by Proposition 15.4.4(i).} \end{array}$$

Now we have to prove that τ^{-1} satisfies (cmp-pres) and (add). As to (cmp-pres), assume that $b \in \mathcal{K}(D)$ and $\tau^{-1}(b) \sqsubseteq \bigsqcup X$, with X directed. Then $b \sqsubseteq \bigsqcup \tau(X)$, since τ satisfies (add). Since b is compact, there exists $x \in X$ such that $b \sqsubseteq \tau(x)$,

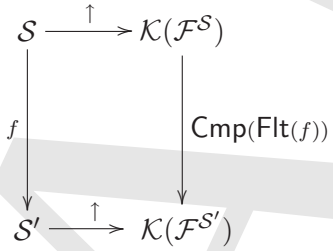
hence $\tau^{-1}(b) \sqsubseteq x$ and we are done. As to (add), let $X \subseteq D$. Then

$$\begin{aligned} \tau^{-1}(\bigsqcup X) &= \tau^{-1}(\bigsqcup\{\tau(\tau^{-1}(x)) \mid x \in X\}), \\ &= \tau^{-1}(\tau(\bigsqcup\{\tau^{-1}(x) \mid x \in X\})), \quad \text{since } \tau \text{ satisfies (add)}, \\ &= \bigsqcup\{\tau^{-1}(x) \mid x \in X\}. \blacksquare \end{aligned}$$

17.1.23. COROLLARY. *The categories* $\mathbf{MSL}^{\mathsf{U}}$ *and* $\mathbf{ALG}_a$ *are equivalent; in fact the isomorphisms in Proposition 17.1.22 form natural isomorphisms showing*

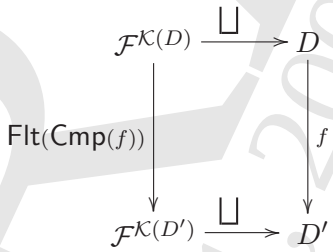
$$\mathsf{Cmp} \circ \mathsf{Flt} \cong \mathsf{Id}_{\mathbf{MSL}^{\mathsf{U}}} \;\&\; \mathsf{Flt} \circ \mathsf{Cmp} \cong \mathsf{Id}_{\mathbf{ALG}_a}.$$

PROOF. First one has to show that in $\mathbf{MSL}^{\mathsf{U}}$ the following diagram commutes.



One must show $\mathsf{Cmp}(\mathsf{Flt}(f))(\uparrow s) = \uparrow(f(s))$. This follows from Lemma 17.1.18(ii) and the definition of Cmp.

Secondly one has to show that in $\mathbf{ALG}_a$ the following diagram commutes.



Now for $X \in \mathcal{F}^{\mathcal{K}(D)}$ one has

$$\begin{aligned} \mathsf{Flt}(\mathsf{Cmp}(f))(X) &= \{d' \in \mathcal{K}(D') \mid \exists d \in X.\, f(d) \leq d'\} \\ &= \{d' \in \mathcal{K}(D') \mid \exists d \in X.\, d' \sqsubseteq f(d)\}. \end{aligned}$$

Hence, using also the continuity of f and the fact that X as subset of $\langle D, \sqsubseteq\rangle$ is directed,

$$\bigsqcup(\mathsf{Flt}(\mathsf{Cmp}(f))(X)) = \bigsqcup_{d \in X} f(d) = f(\bigsqcup X),$$

and the diagram commutes. As before we have $\mathsf{Flt} \circ \mathsf{Cmp} \cong \mathsf{Id}_{\mathbf{ALG}_a}$. $\blacksquare$

This result is a special case of Stone duality, cf. Johnstone [1986] (II, 3.3).

Equivalence between $\mathbf{MSL}^{-\mathsf{U}}$ and $\mathbf{ALG}^s_a$

We prove that $\mathbf{MSL}^{-\mathsf{U}} \cong \mathbf{ALG}^s_a$. The proof uses the functors Flt and a small variation of Cmp, denoted by Cmp_s. The functor Flt is given in Definition 17.1.18 where now $\mathcal{F}^{\mathcal{S}}$ is taken with $\mathcal{S} \in \mathbf{MSL}^{-\mathsf{U}}$. But now $\emptyset \in \mathcal{K}(\mathcal{F}^{\mathcal{S}})$ by Definition 15.4.1 and hence, $\mathcal{S} \not\cong \mathcal{K}(\mathcal{F}^{\mathcal{S}})$. To obtain an isomorphism, the functor Cmp_s is defined by considering the set $\mathcal{K}^s(D)$ of compact elements of D without $\perp$.

17.1.24. DEFINITION. Let $D \in \mathbf{ALG}^s_a$.

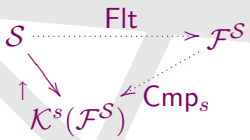
(i) The functor $\mathsf{Cmp}_s : \mathbf{ALG}^s_a \to \mathbf{MSL}^{-\mathsf{U}}$ is defined as follows

$$\mathsf{Cmp}_s(D) = (\mathcal{K}^s(D), \leq).$$

For a morphism f define $\mathsf{Cmp}_s(f)$ by

$$\mathsf{Cmp}_s(f)(d) = f(d).$$

17.1.25. PROPOSITION. (i) *Let $\mathcal{S} \in \mathbf{MSL}^{-\mathsf{U}}$. Then $\sigma : \mathcal{S} \to \mathcal{K}^s(\mathcal{F}^{\mathcal{S}})$ defined by $\sigma(s) = \uparrow s$ is an $\mathbf{MSL}^{-\mathsf{U}}$ isomorphism.*

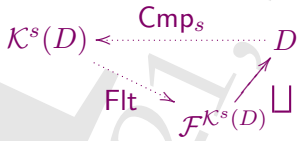


Therefore $\mathcal{S} \cong \mathcal{K}^s(\mathcal{F}^{\mathcal{S}})$.

(ii) *Let $D \in \mathbf{ALG}^s_a$. Then $\tau : \mathcal{F}^{\mathcal{K}^s(D)} \to D$ defined by $\tau(X) = \bigsqcup X$, is an $\mathbf{ALG}^s_a$ isomorphism with inverse $\tau^{-1} : D \to \mathcal{F}^{\mathcal{K}^s(D)}$ defined by*

$$\tau^{-1}(d) = \{c \in \mathcal{K}^s(D) \mid c \sqsubseteq d\}.$$

In diagram,



Therefore $D \cong \mathcal{F}^{\mathcal{K}_s(D)}$.

PROOF. Similar to the proof of Proposition 17.1.22. ■

17.1.26. COROLLARY. *The categories $\mathbf{MSL}^{-\mathsf{U}}$ and $\mathbf{ALG}^s_a$ are equivalent.*

17.1.27. REMARK. The map $\rho : \mathcal{F}^{\mathcal{K}^s(D)} \to \mathcal{F}^{\mathcal{K}(D)}$, given by $\rho(X) = X \cup \{\perp_D\}$ is an isomorphism in the category $\mathbf{ALG}^s_a$, hence also in $\mathbf{ALG}_a$.

17.2. Natural type structures and lambda structures

In this section we prove for the natural type and lambda structures that

$$\begin{array}{rcl} \mathcal{S} & \simeq & \mathcal{K}(\mathcal{F}^{\mathcal{S}}), \\ D & \simeq & \mathcal{F}^{\mathcal{K}(D)}, \end{array}$$

such that there is a congruence between the categories $\mathbf{NTS}^{\mathsf{U}} \cong \mathbf{NLS}$. The results of this Section will be generalized using zip structures in Sections 17.3 and 17.4. Even if the results in this section follow from the mentioned generalization, we decided to keep the proofs here as a warm-up. Moreover, the proofs of the results in this Section are more direct than those obtained as corollaries.

17.2.1. Definition. Let $D, D' \in \mathbf{ALG}$. A *Galois connection* between D and D', notation $\langle \mathsf{m}, \mathsf{n} \rangle : D \to D'$, is a pair of continuous functions $\langle \mathsf{m}, \mathsf{n} \rangle$ with $\mathsf{m} : D \to D'$, $\mathsf{n} : D' \to D$ such that

$$\begin{array}{ll} \text{(Galois-1)} & \mathsf{n} \circ \mathsf{m} \sqsupseteq \mathsf{Id}_D, \\ \text{(Galois-2)} & \mathsf{m} \circ \mathsf{n} \sqsubseteq \mathsf{Id}_{D'}; \end{array}$$

m is said to be the *left adjoint* of the Galois connection, n the *right adjoint*.

A statement equivalent to (Galois-1, Galois-2) is the following.

$$\text{(Galois)} \quad \forall x \in D, x' \in D'. \mathsf{m}(x) \sqsubseteq x' \;\Leftrightarrow\; x \sqsubseteq n(x').$$

See Exercise 17.5.8.

Each adjoint in a Galois connection determines the other, see exercise 17.5.9. For this reason often one writes m^R for n, and n^L for m. From now on $\underline{\mathsf{m}}$ is short for $\langle \mathsf{m}, \mathsf{m}^R \rangle$.

The next lemma provides some properties of Galois connections.

17.2.2. Lemma. *Let $\underline{\mathsf{m}} : D \to D'$ be a Galois connection. Then*

(i) m *is additive:*

$$\forall X \subseteq D,\ \mathsf{m}(\bigsqcup X) = \bigsqcup \mathsf{m}(X).$$

In particular m *is strict,* $\mathsf{m}(\bot) = \bot$, *as* $\bot = \bigsqcup \emptyset$.

(ii) $d \in \mathcal{K}(D) \;\Rightarrow\; \mathsf{m}(d) \in \mathcal{K}(D')$.

(iii) *Let* $d, e \in \mathcal{K}(D)$. *Then*

$$\mathsf{m} \circ (d \mapsto e) \circ \mathsf{m}^R = (\mathsf{m}(d) \mapsto \mathsf{m}(e)).$$

Proof. (i) As m is monotone we have $\bigsqcup \mathsf{m}(X) \sqsubseteq \mathsf{m}(\bigsqcup X)$. On the other hand

$$\begin{array}{rcll} \mathsf{m}(\bigsqcup X) & \sqsubseteq & \mathsf{m}(\bigsqcup \mathsf{m}^R \circ \mathsf{m}(X)), & \text{by (Galois-1)}, \\ & \sqsubseteq & \mathsf{m}(\mathsf{m}^R(\bigsqcup \mathsf{m}(X))), & \text{since } \mathsf{m}^R \text{ is monotone}, \\ & \sqsubseteq & \bigsqcup \mathsf{m}(X), & \text{by (Galois-2)}. \end{array}$$

(ii) Let $d \in \mathcal{K}(D)$ and let $\mathsf{m}(d) \sqsubseteq \bigsqcup Z$, where $Z \subseteq D'$ is any directed set. Then, by (Galois), $d \sqsubseteq \mathsf{m}^R(\bigsqcup Z) = \bigsqcup \mathsf{m}^R(Z)$. Since d is compact, there exists $z \in Z$ such that $d \sqsubseteq \mathsf{m}^R(z)$, hence, by (Galois), $\mathsf{m}(d) \sqsubseteq z$.This proves that $\mathsf{m}(d)$ is compact.

(iii) Let $y \in D'$. We have

$$(\mathsf{m} \circ (d \mapsto e) \circ \mathsf{m}^R)(y) = \begin{cases} \mathsf{m}(e), & \text{if } \mathsf{m}^R(y) \sqsupseteq d; \\ \mathsf{m}(\bot), & \text{otherwise.} \end{cases}$$

Note that by (Galois), $d \sqsubseteq \mathsf{m}^R(y)$ is equivalent to $\mathsf{m}(d) \sqsubseteq y$. Moreover, as previously noted, $\mathsf{m}(\bot) = \bot$. So we have

$$(\mathsf{m} \circ (d \mapsto e) \circ \mathsf{m}^R)(y) = \begin{cases} \mathsf{m}(e), & \text{if } \mathsf{m}(d) \sqsubseteq y; \\ \bot, & \text{otherwise.} \end{cases}$$

The RHS above is the definition of the step function $(\mathsf{m}(d) \mapsto \mathsf{m}(e))$. ■

17.2.3. Definition. (i) Let $D \in \mathbf{ALG}$, $F : D \to [D \to D]$, $G : [D \to D] \to D$. A triple $\mathcal{D} = \langle D, F, G \rangle$ is called a *lambda structure*, notation LS, if F and G are continuous.

(ii) Such a LS $\mathcal{D}$ is *natural* if $\langle F, G \rangle$ is a Galois connection, with $F = G^R$.

Note that we are using the notation $\langle D, F, G \rangle$ coherent with lambda structure notation, but this is in contrast with Galois connection notation, since the left adjoint G is put on the right.

Given a natural lambda structure $\mathcal{D}$, then (Galois) implies

$$\text{(func-Galois)} \quad \forall d, e \in \mathcal{K}(D), x \in D.\ e \sqsubseteq F(x)(d) \ \Leftrightarrow\ G(d \mapsto e) \sqsubseteq x.$$

The next lemma shows how to build Galois connections in **ALG** out of morphisms between $\mathrm{NTS}^{\mathtt{U}}$'s.

17.2.4. Lemma. *Let $f \in \mathbf{NTS}^{\mathtt{U}}(\mathcal{S}, \mathcal{S}')$. Let $\overline{f}$ be short for $\mathsf{Flt}(f)$. Define $\overline{f}^R : \mathbf{ALG}(\mathcal{F}^{\mathcal{S}'}, \mathcal{F}^{\mathcal{S}})$ by*

$$\overline{f}^R(d') \ = \ \{s \mid f(s) \in d'\}.$$

Then $\langle \overline{f}, \overline{f}^R \rangle : \mathcal{F}^{\mathcal{S}} \to \mathcal{F}^{\mathcal{S}'}$ is a Galois connection (so the name $\overline{f}^R$ is appropriate).

Proof. We leave as an exercise to prove that $\overline{f}$ is well-defined and continuous. Note that

$$(1) \quad [\forall s \in \mathcal{S}.\ s \in d \ \Rightarrow f(s) \in d'] \ \Leftrightarrow \ [\forall s' \in \mathcal{S}'.\ (\exists s \in d. f(s) \leq s') \ \Rightarrow s' \in d'].$$

($\Rightarrow$) Suppose that $s' \in \mathcal{S}'$ and there exists $s \in d$ such that $f(s) \leq s'$. Then $f(s) \in d'$ by the LHS and hence $s' \in d'$ as d is a filter.

($\Leftarrow$) Let $s \in d$. Choosing $s' = f(s)$ in the RHS, we get $f(s) \in d'$.

Now we prove that (Galois) holds for $\langle \overline{f}, \overline{f}^R \rangle$.

$$\begin{aligned} \overline{f}(x) \subseteq x' \ &\Leftrightarrow \ \{s' \mid \exists s \in x. f(s) \leq s'\} \subseteq x' \\ &\Leftrightarrow \ \forall s' \in \mathcal{S}'.\ (\exists s \in x. f(s) \leq s') \ \Rightarrow \ s' \in x' \\ &\Leftrightarrow \ \forall s \in \mathcal{S}.\ s \in x \ \Rightarrow \ f(s) \in x', \qquad \text{by (1)}, \\ &\Leftrightarrow \ x \subseteq \{s \mid f(s) \in x'\} \\ &\Leftrightarrow \ x \subseteq \overline{f}^R(x'). \ ■ \end{aligned}$$

From now on $\underline{f}$ denotes for $f \in \mathbf{NTS}^{\mathsf{U}}(\mathcal{S}, \mathcal{S}')$ the Galois connection $\langle \overline{f}, \overline{f}^R \rangle$.

Next definition is necessary for introducing morphisms between lambda structures. We will explain in Sections 17.3 and 17.4 this choice.

17.2.5. DEFINITION. [**special-galois-connections-definition**] Let $\mathcal{D} = \langle D, F, G \rangle$, $\mathcal{D}' = \langle D', F, G \rangle$ be lambda structures. A *lambda Galois connection* is a Galois connection $\underline{\mathsf{m}} = \langle \mathsf{m}, \mathsf{m}^R \rangle : \mathcal{D} \to \mathcal{D}'$ such that

$$\begin{array}{ll} \text{(lambda-gc1)} & \forall f \in [D \to D].\mathsf{m}(G(f)) \sqsubseteq G'(\mathsf{m} \circ f \circ \mathsf{m}^R); \\ \text{(lambda-gc2)} & \forall x' \in D', x \in D.F(m^R(x'))(x) \sqsubseteq m^R(F'(x')(m(x))). \end{array}$$

If we write $f^{\mathsf{m}} = \mathsf{m} \circ f \circ \mathsf{m}^R$, then we can reformulate these conditions as

$$\begin{array}{ll} \text{(lambda-gc1)} & \forall f \in [D \to D].\mathsf{m}(G(f)) \sqsubseteq G'(f^{\mathsf{m}}); \\ \text{(lambda-gc2)} & \forall x' \in D', x \in D.m^R(x') \cdot x \sqsubseteq m^R(x' \cdot (m(x))). \end{array}$$

See also Lemma 18.1.11.

We can dualize both (lambda-gc1) and (lambda-gc2), obtaining the following conditions:

$$\begin{array}{ll} \text{(lambda-gc1}^*\text{)} & \forall f' \in [D' \to D'].\mathsf{m}^R(G'(f')) \sqsupseteq G(\mathsf{m}^R \circ f' \circ \mathsf{m}); \\ \text{(lambda-gc2}^*\text{)} & \forall x \in D, x' \in D'.F'(m(x))(x') \sqsupseteq m(F(x)(m^R(x'))). \end{array}$$

We have the following result, whose proof is left as an exercise.

17.2.6. LEMMA. [**gcR**]
(i) *(lambda-gc1)* $\Leftrightarrow$ *(lambda-gc1*).*
(ii) *(lambda-gc2)* $\Leftrightarrow$ *(lambda-gc2*).*

17.2.7. DEFINITION. (i) The category **LS** consists of lambda structures as objects and lambda Galois connections as morphisms. The composition between morphisms $\langle \mathsf{m}, \mathsf{m}^R \rangle : \mathcal{D} \to \mathcal{D}'$ and $\langle \mathsf{n}, \mathsf{n}^R \rangle : \mathcal{D}' \to \mathcal{D}''$ is given by $\langle \mathsf{n} \circ \mathsf{m}, \mathsf{m}^R \circ \mathsf{n}^R \rangle$.

(ii) The category of *natural lambda structures*, notation **NLS**, is the full subcategory of **LS** which has as objects natural lambda structures.

For $\mathcal{S} \in \mathrm{TS}^{\mathsf{U}}$ the maps $F^{\mathcal{S}} : \mathcal{F}^{\mathcal{S}} \to [\mathcal{F}^{\mathcal{S}} \to \mathcal{F}^{\mathcal{S}}]$ and $G^{\mathcal{S}} : [\mathcal{F}^{\mathcal{S}} \to \mathcal{F}^{\mathcal{S}}] \to \mathcal{F}^{\mathcal{S}}$ are two continuous functions, defined in Definition 15.4.5 as follows.

$$\begin{array}{l} \forall x \in \mathcal{F}^{\mathcal{S}}.\ F^{\mathcal{S}}(x) = \lambda\!\!\lambda y \in \mathcal{S}.\uparrow \{t \mid \exists s \in y.s \to t \in x\}; \\ \forall f \in [\mathcal{F}^{\mathcal{S}} \to \mathcal{F}^{\mathcal{S}}].\ G^{\mathcal{S}}(f) = \uparrow \{s \to t \mid t \in f(\uparrow s)\}. \end{array}$$

Also remember that $x \cdot y := F^{\mathcal{S}}(x)(y)$, for any $x, y \in \mathcal{F}^{\mathcal{S}}$.

17.2.8. LEMMA. *Let $\mathcal{S} \in \mathbf{NTS}^{\mathsf{U}}$, $s, t \in \mathcal{S}$, $x, y \in \mathcal{F}^{\mathcal{S}}$, and $f : \mathcal{F}^{\mathcal{S}} \to \mathcal{F}^{\mathcal{S}}$. Then*
(i) $x \cdot y = \{t \mid \exists s \in y.s \to t \in x\}$.
(ii) $t \in x \cdot \uparrow s \ \Leftrightarrow\ s \to t \in x$.
(iii) $t \in f(\uparrow s) \ \Rightarrow\ (s \to t) \in G^{\mathcal{S}}(f)$.
(iv) $G^{\mathcal{S}}(\uparrow s \mapsto \uparrow t) = \uparrow (s \to t)$.

PROOF. (i) Let $\{x{\cdot}y\} = \{t \mid \exists s \in y.s \to t \in x\}$. Then $\{x{\cdot}y\} \subseteq x{\cdot}y$ by definition. We prove that $\{x \cdot y\}$ is a filter, hence it coincides with $x \cdot y$. Now $\top \in \{x \cdot y\}$ by $(\top \to)$. Moreover, $\{x \cdot y\}$ is upward closed. In fact, let $t \in \{x \cdot y\}$ and $t \leq t'$. Then there exists $s \in y$ such that $(s \to t) \in x$. But $(s \to t) \leq (s \to t')$ by $(\to)$, so this last type is also in the filter x. Therefore $t' \in \{x \cdot y\}$.

Finally, let $t, t' \in \{x \cdot y\}$. Then $(s \to t), (s' \to t') \in x$ for some $s, s' \in y$. Then $(s \to t) \cap (s' \to t') \in x$ and $s \cap s' \in y$, as x, y are filters. By Proposition 15.1.21(ii) one has $(s \to t) \cap (s' \to t') \leq (s \cap s') \to (t \cap t')$, hence this last type is in the filter x. Therefore, by definition of application, $t \cap t' \in \{x \cdot y\}$.

(ii) By (i), $t \in x{\cdot} \uparrow s$ if and only if there exists $s' \in \uparrow s$ such that $(s' \to t) \in x$. By $(\to)$, this is equivalent to $(s \to t) \in x$.

(iii) Easy.

(iv) We have

$$\begin{array}{rcl} G^{\mathcal{S}}(\uparrow s \mapsto \uparrow t) & = & \uparrow \{s' \to t' \mid t' \in (\uparrow s \mapsto \uparrow t)(\uparrow s')\} \\ & = & \uparrow \{s' \to t' \mid [\uparrow s' \supseteq \uparrow s \ \&\ t' \in \uparrow t] \text{ or } t' = \mathtt{U}\} \\ & = & \uparrow (s \to t). \end{array}$$

As to the last equality, $\supseteq$ holds trivially, and $\subseteq$ by $(\to)$, $(\mathtt{U}\to)$. ■

17.2.9. LEMMA. *Let $\mathcal{S}$ be in $\mathbf{NTS}^{\mathtt{U}}$. Write $\mathcal{A}(\mathcal{S}) = \langle \mathcal{F}^{\mathcal{S}}, F^{\mathcal{S}}, G^{\mathcal{S}} \rangle$. Then*

(i) $\mathcal{A}(\mathcal{S}) \in \mathbf{LS}$.

(ii) $\mathcal{A}(\mathcal{S}) \in \mathbf{NLS}$

PROOF. (i) Easy.

(ii) We claim that $\langle F^{\mathcal{S}}, G^{\mathcal{S}} \rangle$ is a Galois connection. As to $F^{\mathcal{S}}(G^{\mathcal{S}}(f)) \sqsupseteq f$, it suffices to prove this on compact elements $\uparrow s \in \mathcal{K}(\mathcal{S})$. Let $f \in [\mathcal{S} \to \mathcal{S}]$, $s \in \mathcal{S}$. We have

$$\begin{array}{rcll} F^{\mathcal{S}}(G^{\mathcal{S}}(f))(\uparrow s) & = & \{t \mid s \to t \in G^{\mathcal{S}}(f)\}, & \text{by Lemma 17.2.8(ii),} \\ & \supseteq & \{t \mid t \in f(\uparrow s)\}, & \text{by Lemma 17.2.8(iii),} \\ & = & f(\uparrow s). & \end{array}$$

On the other hand, let $x \in \mathcal{F}^{\mathcal{S}}$. We have

$$\begin{array}{rcll} G^{\mathcal{S}}(F^{\mathcal{S}}(x)) & = & \uparrow \{s \to t \mid t \in x{\cdot} \uparrow s\} & \\ & = & \uparrow \{s \to t \mid s \to t \in x\}, & \text{by Lemma 17.2.8(ii),} \\ & \subseteq & x. & \blacksquare \end{array}$$

17.2.10. THEOREM. *Define the action of $\mathcal{A}$ on morphisms $f \in \mathbf{NTS}^{\mathtt{U}}(\mathcal{S}, \mathcal{S}')$ by*

$$\mathcal{A}(f) = \underline{f} = \langle \mathsf{Flt}(f), \mathsf{Flt}(f)^R \rangle.$$

Then $\mathcal{A} : \mathbf{NTS}^{\mathtt{U}} \to \mathbf{NLS}$ is a functor.

PROOF. The proof will descend from the results of Sections 17.3 and 17.4 (see in particular Proposition 17.4.31).

17.2.11. DEFINITION. **[cala]**
(i) Given a natural lambda structure $\mathcal{D}$, we define

$$\mathcal{L}(\mathcal{D}) = \langle \mathcal{K}(D), \leq, \cap, \to_D, \mathtt{U} \rangle$$

with $a \leq b \;\Leftrightarrow\; b \sqsubseteq_{\mathcal{K}(D)} a$, $a \cap b \equiv a \sqcup_D b$, $\mathtt{U} = \bot_D$, and $a \to_{\mathcal{K}(D)} b \equiv G(a \mapsto b)$.
(ii) Given $\underline{\mathsf{m}} \in \mathbf{NLS}(\mathcal{D}, \mathcal{D}')$, we define

$$\mathcal{L}(\underline{\mathsf{m}}) = \mathsf{m} \restriction \mathcal{K}(D) : \mathcal{K}(D) \to \mathcal{K}(D').$$

17.2.12. LEMMA. *$\mathcal{L}$ is a functor from* **NLS** *to* $\mathbf{NTS}^{\mathtt{U}}$.

PROOF. The proof will descend from the result of Sections 17.3 and 17.4 (see in particular Theorem 17.4.34).

Now we will prove that $\mathbf{NTS}^{\mathtt{U}}$ and **NLS** are equivalent categories.

17.2.13. PROPOSITION. *Let $\mathcal{S} \in \mathbf{NTS}^{\mathtt{U}}$. Define $\sigma_{\mathcal{S}} : S \to \mathcal{K}(\mathcal{F}^{\mathcal{S}})$ by*

$$\sigma_{\mathcal{S}}(s) = \uparrow s.$$

Then $\sigma_{\mathcal{S}}$ is an isomorphism in $\mathbf{NTS}^{\mathtt{U}}$. Hence $\mathcal{S} \cong \mathcal{K}(\mathcal{F}^{\mathcal{S}})$.

PROOF. We write simply σ for $\sigma_{\mathcal{S}}$. We know from Proposition 17.1.22 that σ is an isomorphism of meet-semilattices. It remains to prove that structure is preserved by σ. It suffices to show that $\sigma(s \to t) = \sigma(s) \to_{\mathcal{K}(\mathcal{F}^{\mathcal{S}})} \sigma(t)$, for any $s, t \in \mathcal{S}$. We have

$$\begin{array}{lcll} \sigma(s) \to_{\mathcal{K}(\mathcal{F}^{\mathcal{S}})} \sigma(t) & = & \uparrow s \to_{\mathcal{K}(\mathcal{F}^{\mathcal{S}})} \uparrow t & \\ & = & G^{\mathcal{S}}(\uparrow s \mapsto \uparrow t), & \text{by Definition 17.2.11,} \\ & = & \uparrow (s \to t), & \text{by Lemma 17.2.9(i),} \\ & = & \sigma(s \to t). & \end{array}$$

Similarly, σ^{-1} preserves structure. ■

17.2.14. PROPOSITION. *Let $\mathcal{D} \in \mathbf{NLS}$. Define $\tau_D : \mathcal{F}^{\mathcal{K}(D)} \to D$ by*

$$\tau_D(x) = \bigsqcup x.$$

Then τ_D is an isomorphism in **NLS**. *Hence $\mathcal{D} \cong \mathcal{F}^{\mathcal{K}(D)}$.*

PROOF. Write $\tau = \tau_D$. By Proposition 17.1.22 τ is an isomorphism of lattices. The pair $\langle \tau, \tau^{-1} \rangle$ is a Galois connection. We show that

$$\langle \tau, \tau^{-1} \rangle : \langle \mathcal{F}^{\mathcal{K}(D)}, F^{\mathcal{K}(D)}, G^{\mathcal{K}(D)} \rangle \to \langle D, F, G \rangle$$

is lambda. Let $f : \mathcal{F}^{\mathcal{K}(D)} \to \mathcal{F}^{\mathcal{K}(D)}$ be a continuous function. We have:

$$\begin{aligned}
\tau(G^{\mathcal{K}(D)}(f)) &= \tau(\uparrow \{a \to_{\mathcal{K}(D)} b \mid b \in f(\uparrow a)\}) \\
&= \bigsqcup\{a \to_{\mathcal{K}(D)} b \mid b \in f(\uparrow a)\}, \\
&\quad \text{since } \bigsqcup \uparrow x = \bigsqcup x, \text{ for any } x \subseteq \mathcal{K}(D), \\
&= \bigsqcup\{G(a \mapsto b \mid b \in f(\uparrow a)\}), \text{ by definition of } \to_{\mathcal{K}(D)}, \\
&= \bigsqcup\{G(a \mapsto b) \mid b \sqsubseteq \bigsqcup(f(\uparrow a))\} \\
&= \bigsqcup\{G(a \mapsto b) \mid b \sqsubseteq \tau \circ f \circ \tau^{-1}(a)\} \\
&= G(\bigsqcup\{a \mapsto b \mid b \sqsubseteq \tau \circ f \circ \tau^{-1}(a)\}), \\
&\quad \text{since } G \text{ is additive by Lemma 17.2.2(i)}, \\
&= G(\tau \circ f \circ \tau^{-1})
\end{aligned}$$

Also $\langle \tau, \tau^{-1} \rangle$ is a lambda Galois connection:

$$\begin{aligned}
\tau^{-1}(G(g)) &= \tau^{-1}(G(\tau \circ \tau^{-1} \circ g \circ \tau \circ \tau^{-1})) \\
&= \tau^{-1} \circ \tau(G^{\mathcal{K}(D)}(\tau^{-1} \circ g \circ \tau)), \quad \text{as above,} \\
&= G^{\mathcal{K}(D)}(\tau^{-1} \circ g \circ \tau). \blacksquare
\end{aligned}$$

17.2.15. THEOREM. *The categories* **NTS**$^{\mathtt{U}}$ *and* **NLS** *are equivalent.*

PROOF. This follows from Propositions 17.2.13 and 17.2.14 almost in the same way as Corollary 17.1.23 follows from Proposition 17.1.22. There is one extra case. If $\langle \mathsf{m}, \mathsf{m}^R \rangle : D \to D'$. then we must show the commutativity of the following diagram

$$\begin{array}{ccc}
\mathcal{D} & \xleftarrow{\bigsqcup} & \mathcal{F}^{\mathcal{K}(D)} \\
{\scriptstyle m^R}\uparrow & & \uparrow{\scriptstyle \overline{m\restriction\mathcal{K}(D)}^R} \\
\mathcal{D}' & \xleftarrow{\bigsqcup} & \mathcal{F}^{\mathcal{K}(D')}
\end{array}$$

This is done in Exercise 17.5.13. ■

17.3. Type and zip structures

[type.zip]

The aim of the two next sections is to compare type and lambda structures, using an intermediate kind of structure, the *zip structures*. A zip structure is a pair $\langle D, Z \rangle$, where D is an object in **ALG** and $Z : \mathcal{K}(D) \times \mathcal{K}(D) \to \mathcal{K}(D)$ is the semantic counterpart of the arrow constructor in type structures, being a set-theoretic function that "zips" the information of two compact elements, not necessarily in such a way that the constituents can be found back. The various categories of zip structures are easily proven to be equivalent to the corresponding ones of type structures. So we can think of zip structures as an alternative way of describing type structures. In the following section, they will be compared with lambda structures.

17.3.1. DEFINITION. [**LS**]

(i) A *zip structure* is a pair $D_Z = \langle D, Z\rangle$ with $D \in \mathbf{ALG}$ and

$$Z : \mathcal{K}(D) \times \mathcal{K}(D) \to \mathcal{K}(D)$$

an arbitrary map.

(ii) The category **ZS** has zip structures as objects and maps $f : D \to D'$ such that ($a, b, c, \ldots$ ranging over $\mathcal{K}(D)$)

$$\begin{array}{ll} \text{(cmp-pres)} & \forall a.f(a) \in \mathcal{K}(D'); \\ \text{(add)} & \forall X \subseteq D.f(\bigsqcup X) = \bigsqcup f(X); \\ \text{(Z-comm)} & \forall a, b.f(Z(a,b)) = Z'(f(a), f(b)) \end{array}$$

as morphisms $\mathbf{ZS}(\langle D, Z\rangle, \langle D', Z'\rangle)$. The second requirement implies that a morphism f is continuous (only required to preserve sups for directed sets X) and strict, i.e. $f(\bot_D) = \bot_{D'}$.

We now specialize this general framework to special zip structures.

17.3.2. DEFINITION. [lls] Let (D, Z) be a zip structure.

(i) Then $\langle D, Z\rangle$ is a *lazy zip structure* if the following holds.

(1) [**LLS1**] (Z-contr) $a \sqsubseteq a' \;\&\; b' \sqsubseteq b \;\Rightarrow\; Z(a', b') \sqsubseteq Z(a, b)$;

(2) [**LLS2**] (Z-add) $Z(a, b_1 \sqcup b_2) = Z(a, b_1) \sqcup Z(a, b_2)$;

(3) [**LLS3**] (Z-lazy) $Z(\bot_D, \bot_D) \sqsubseteq Z(a, b)$.

(ii) **LZS** is the full subcategory of **ZS** consisting of lazy zip structures.

17.3.3. DEFINITION. [nls] Let $(D, Z) \in \mathbf{ZS}$.

(i) Then $\langle D, Z\rangle$ is a *natural zip structure* if $\langle D, Z\rangle \in \mathbf{LZS}$ and moreover

$$\text{(Z-bot)} \quad Z(\bot_D, \bot_D) = \bot_D.$$

(ii) **NZS** is the full subcategory of **LZS** consisting of natural zip structures.

17.3.4. REMARK. [bot=¿bot-lazy] Since condition (Z-bot) is stronger than (Z-lazy), D is natural if it satisfies (Z-contr), (Z-add) and (Z-bot). In fact, (Z-bot) corresponds to ($\mathtt{U}\!\to$), and (Z-lazy) to the weaker ($\mathtt{U}_{\text{lazy}}$).

Strict zip structures

17.3.5. DEFINITION. Let $D \in \mathbf{ALG}$. Remember that $\mathcal{K}^s(D) = \mathcal{K}(D) - \{\bot_D\}$.

(i) A *strict zip structure* is of the form $\langle D, Z\rangle$, with

$$Z : (\mathcal{K}^s(D) \times \mathcal{K}^s(D)) \to \mathcal{K}^s(D).$$

(ii) If we write $Z(a, b)$, then it is always understood that $(a, b) \in \mathrm{dom}(Z)$.

(iii) The category $\mathbf{ZS}^s$ consists of strict zip structures as objects and as morphisms maps f satisfying

$$\begin{array}{ll} \text{(cmp-pres}^s\text{)} & \forall a \in \mathcal{K}^s(D).f(a) \in \mathcal{K}^s(D'); \\ \text{(add)} & \forall X \subseteq D.f(\bigsqcup X) = \bigsqcup f(X); \\ \text{(Z-comm)} & \forall a, b.f(Z(a,b)) = Z'(f(a), f(b)). \end{array}$$

17.3.6. DEFINITION. (i) Let $\langle D, Z\rangle \in \mathbf{ZS}^s$. Then $\langle D, Z\rangle$ is called a *proper strict zip structure*, if it satisfies the following conditions.

$$\begin{array}{ll} (Z\text{-contr}) & a \sqsubseteq a' \;\&\; b' \sqsubseteq b \;\Rightarrow\; Z(a',b') \sqsubseteq Z(a,b); \\ (Z\text{-add}) & Z(a, b_1 \sqcup b_2) = Z(a,b_1) \sqcup Z(a,b_2). \end{array}$$

(ii) $\mathbf{PZS}^s$ is the full subcategory of $\mathbf{ZS}^s$ consisting of proper strict zip structures.

17.3.7. REMARK. In Section 15.3 we introduced the names $\mathrm{MSL}^{\mathtt{U}}, \mathrm{MSL}^{\text{-}\mathtt{U}}, \mathrm{TS}^{\mathtt{U}}$, $\mathrm{TS}^{\text{-}\mathtt{U}}$ for collections of meet semi-lattices and type structures. The superscript $\mathtt{U}$ in these names denotes that the structures in the relevant collections have a top element.

In Definitions 17.1.12, 17.3.1 and 17.3.5, we introduced $\mathbf{ALG}_a$,$\mathbf{ALG}^s_a$,$\mathbf{ZS}$,$\mathbf{ZS}^s$. The superscript s, to be read as 'strict', concerns the top element of $\mathcal{K}(D)_{\leq}$ (see Definition 17.1.2) and suggests that we are not interested in it.

Equivalences between type and zip structures

[**TS-LS**] Now we will extend the equivalences of Section 17.1 to the various categories of type and zip structures. We will show the following equivalences of categories.

$$\begin{array}{rcl} \mathbf{TS}^{\mathtt{U}} & \cong & \mathbf{ZS}; \\ \mathbf{LTS}^{\mathtt{U}} & \cong & \mathbf{LZS}; \\ \mathbf{NTS}^{\mathtt{U}} & \cong & \mathbf{NZS}; \\ \mathbf{TS}^{\text{-}\mathtt{U}} & \cong & \mathbf{ZS}^s; \\ \mathbf{PTS}^{\text{-}\mathtt{U}} & \cong & \mathbf{PZS}^s. \end{array}$$

Note that under this correspondence there is a sort of adjunction in the superscripts of the names due to the fact that in the lefthand side of this table the presence of a top is given explicitly, whereas in the right hand side the name indicates when we are *not* interested in the top (of $\mathcal{K}(D)_{\leq}$). In particular, note that **TS** does not correspond with **ZS**.

Since the proofs are standard, we will give the full details only for the equivalence between $\mathbf{TS}^{\mathtt{U}}$ and **ZS**, whilst for the other cases we just state the results, leaving the easy proofs as exercises.

Equivalence between $\mathbf{TS}^{\mathtt{U}}$ and ZS

First we see how the functors Flt and Cmp between $\mathbf{MSL}^{\mathtt{U}}$ and $\mathbf{ALG}_a$ induce new functors between the richer categories $\mathbf{TS}^{\mathtt{U}}$ and **ZS**.

17.3.8. DEFINITION. [**TS.LS.fnct**]
(i) For $\mathcal{S} \in \mathbf{TS}^{\mathtt{U}}$, define $\mathcal{Q}(\mathcal{S}) \in \mathbf{ZS}$ by

$$\mathcal{Q}(\mathcal{S}) = \langle \mathcal{F}^{\mathcal{S}}, Z^{\mathcal{S}}\rangle,$$

with $Z^{\mathcal{S}} : \mathcal{K}(\mathcal{F}^{\mathcal{S}}) \times \mathcal{K}(\mathcal{F}^{\mathcal{S}}) \to \mathcal{K}(\mathcal{F}^{\mathcal{S}})$ defined by

$$Z^{\mathcal{S}}(\uparrow s, \uparrow t) \;= \uparrow (s \to t).$$

(ii) For $\langle D, Z\rangle \in \mathbf{ZS}$, define $\mathcal{M}(\langle D, Z\rangle) \in \mathbf{TS}^{\mathtt{U}}$ by

$$\mathcal{M}(\langle D, Z\rangle) = \langle \mathcal{K}(D), \leq, \cap, \rightarrow_Z, \mathtt{U}\rangle,$$

with $a \leq b \;\Leftrightarrow\; b \sqsubseteq_D a$, $a \cap b \equiv a \sqcup_D b$, and $\mathtt{U} \equiv \bot_D$, as in Definition 17.1.20 and

$$a \rightarrow_Z b \equiv Z(a, b).$$

(iii) The actions of the maps $\mathcal{M} : \mathbf{ZS} \rightarrow \mathbf{TS}^{\mathtt{U}}$ and $\mathcal{Q} : \mathbf{TS}^{\mathtt{U}} \rightarrow \mathbf{ZS}$ on morphisms are defined as follows. Given $f \in \mathbf{TS}^{\mathtt{U}}(\mathcal{S}, \mathcal{S}')$, $X \in \mathcal{F}^{\mathcal{S}}$ and $g \in \mathbf{ZS}(\langle D, Z\rangle, \langle D', Z'\rangle)$ define

$$\begin{array}{rcl} \mathcal{Q}(f) & = & \lambda X.\{t \mid \exists s \in X. f(s) \leq t\}; \\ \mathcal{M}(g) & = & g \restriction \mathcal{K}(D). \end{array}$$

17.3.9. Lemma. [**up-arr-com**] *If $\mathcal{S} \in \mathbf{TS}^{\mathtt{U}}$ and $Z^{\mathcal{S}}$ are defined as in Definition 17.3.8(i), then $\langle \mathcal{K}(\mathcal{F}^{\mathcal{S}}), Z^{\mathcal{S}}\rangle \in \mathbf{ZS}$. Moreover if $\rightarrow_{Z^{\mathcal{S}}}$ is defined for $\langle \mathcal{K}(\mathcal{F}^{\mathcal{S}}), Z^{\mathcal{S}}\rangle \in \mathbf{ZS}$ as in Definition 17.3.8(ii), then*

$$\uparrow s \rightarrow_{Z^{\mathcal{S}}} \uparrow t = \uparrow (s \rightarrow t).$$

Proof. Immediate from Definition 17.3.8. ■

17.3.10. Proposition. [**ccmp-functor**] *$\mathcal{M}$ is a functor from $\mathbf{ZS}$ to $\mathbf{TS}^{\mathtt{U}}$.*

Proof. We just have to prove that $\mathcal{M}$ transforms a morphism in $\mathbf{ZS}$ into a morphism into $\mathbf{TS}^{\mathtt{U}}$. Let $m : \langle D, Z\rangle \rightarrow \langle D', Z'\rangle$ be a morphism. By Lemma 17.1.21 we only need to show that $\mathcal{M}(m)$ commutes with arrows. Now

$$\begin{array}{rcll} \mathcal{M}(m)(a \rightarrow_Z b) & = & m(a \rightarrow_Z b), & \text{by definition of } \mathcal{M}(m), \\ & = & m(Z(a, b)), & \text{by definition of } \rightarrow_Z, \\ & = & Z'(m(a), m(b)), & \text{since } m \text{ satisfies } (Z\text{-comm}), \\ & = & m(a) \rightarrow_{Z'} m(b)), & \\ & = & \mathcal{M}(m(a)) \rightarrow_{Z'} \mathcal{M}(m(b)). \blacksquare & \end{array}$$

17.3.11. Proposition. [**fflt-functor2**] *$\mathcal{Q}$ is a functor from $\mathbf{TS}^{\mathtt{U}}$ to $\mathbf{ZS}$.*

Proof. We have to prove that $\mathcal{Q}$ transforms a morphism in $\mathbf{TS}^{\mathtt{U}}$ into a morphism in $\mathbf{ZS}$. Let $f \in \mathbf{TS}^{\mathtt{U}}(\mathcal{S}, \mathcal{S}')$ with arrows $\rightarrow_{Z^{\mathcal{S}}}$ and $\rightarrow_{Z^{\mathcal{S}'}}$ corresponding to $Z^{\mathcal{S}}$ and $Z^{\mathcal{S}'}$, respectively. By Proposition 17.1.19 we only need to show that $\mathcal{Q}(f)$ satisfies $(Z\text{-comm})$. Indeed,

$$\begin{array}{rcll} \mathcal{Q}(f)(Z^{\mathcal{S}}(\uparrow s, \uparrow t)) & = & \mathcal{Q}(f)(\uparrow (s \rightarrow_{Z^{\mathcal{S}}} t), & \text{by definition of } Z^{\mathcal{S}}, \\ & = & \uparrow f(s \rightarrow_{Z^{\mathcal{S}}} t), & \text{by Lemma 17.1.18}, \\ & = & \uparrow (f(s) \rightarrow_{Z^{\mathcal{S}'}} f(t)), & \text{since } f \in \mathbf{TS}^{\mathtt{U}}(\mathcal{S}, \mathcal{S}'), \\ & = & Z^{\mathcal{S}'}(\uparrow f(s), \uparrow f(t)), & \text{by definition of } Z^{\mathcal{S}'}, \\ & = & Z^{\mathcal{S}'}(\mathcal{Q}(f)(\uparrow s), \mathcal{Q}(f)(\uparrow t)), & \text{by Lemma 17.1.18. } \blacksquare \end{array}$$

Now we will prove that **ZS** and $\mathbf{TS}^{\mathsf{U}}$ are equivalent. To this aim we show that natural isomorphisms $\mathsf{Id}_{\mathbf{TS}^{\mathsf{U}}} \simeq \mathcal{M} \circ \mathcal{Q}$ and $\mathcal{Q} \circ \mathcal{M} \simeq \mathsf{Id}_{\mathbf{ZS}}$ are given by σ and τ respectively, exactly as in the case of the equivalence between the categories $\mathbf{MSL}^{\mathsf{U}}$ and $\mathbf{ALG}_a$.

17.3.12. PROPOSITION. **[phi-iso]** *Let $\mathcal{S}$ be a top type structure. Let $\sigma : \mathcal{S} \to \mathcal{K}(\mathcal{F}^{\mathcal{S}})$ be the map such that $\sigma(s) = \uparrow s$. Then σ is an isomorphism in $\mathbf{TS}^{\mathsf{U}}$.*

PROOF. By Proposition 17.1.22(i) we only need to show that σ and σ^{-1} commute with arrows. Now σ is a bijection, hence the following suffices.

$$\begin{aligned}
\sigma(s \to t) &= \uparrow (s \to t), \\
&= \uparrow s \to_{Z^{\mathcal{S}}} \uparrow t, \qquad \text{by Lemma 17.3.9,} \\
&= \sigma(s) \to_{Z^{\mathcal{S}}} \sigma(t). \blacksquare
\end{aligned}$$

17.3.13. PROPOSITION. **[tau-iso]** *Let $\langle D, Z\rangle \in \mathbf{ZS}$. Define $\tau : \mathcal{F}^{\mathcal{K}(D)} \to D$ by*

$$\tau(x) = \bigsqcup x, \quad \textit{where the sup is taken in } D.$$

Then τ is an isomorphism in **ZS**.

PROOF. By Proposition 17.1.22(ii) we only need to show that τ and τ^{-1} satisfy (Z-comm). As to τ, we have

$$\begin{aligned}
\tau(Z^{\mathcal{K}(D)}(\uparrow a, \uparrow b)) &= \tau(\uparrow (a \to_Z b)), && \text{by definition of } Z^{\mathcal{K}(D)}, \\
&= \tau(\uparrow Z(a,b)), && \text{by definition of } \to_Z, \\
&= \bigsqcup(\uparrow Z(a,b)) \\
&= Z(a,b) \\
&= Z(\bigsqcup \uparrow a, \bigsqcup \uparrow b) \\
&= Z(\tau(\uparrow a), \tau(\uparrow b)).
\end{aligned}$$

As to τ^{-1}, we must show

$$\tau^{-1}(Z(a,b)) = Z^{\mathcal{K}(D)}(\tau^{-1}(a), \tau^{-1}(b)).$$

This follows immediately from (Z-comm) for τ and $\tau^{-1}(a) = \uparrow a$, $\tau^{-1}(b) = \uparrow b$, $\tau \circ \tau^{-1} = \mathsf{Id}_D$ and $\tau^{-1} \circ \tau = \mathsf{Id}_{\mathcal{F}^{\mathcal{K}(D)}}$. $\blacksquare$

17.3.14. THEOREM. **[LS=TTS]** *The categories $\mathbf{TS}^{\mathsf{U}}$ and* **ZS** *are equivalent.*

PROOF. As in Corollary 17.1.23, using Propositions 17.3.10-17.3.13. $\blacksquare$

The other equivalences between type and zip structures

First we define the functors $\mathcal{M}_s$ and $\mathcal{Q}_s$ for the strict case.

17.3.15. DEFINITION. **[strict-Flt]**

(i) For $\mathcal{S} \in \mathbf{TS}^{\text{-U}}$, define $\mathcal{Q}_s(\mathcal{S}) \in \mathbf{ZS}^s$ by

$$\mathcal{Q}_s(\mathcal{S}) = (\mathcal{F}^{\mathcal{S}}, Z_s^{\mathcal{S}}),$$

with $Z_s^{\mathcal{S}} : \mathcal{K}^s(\mathcal{F}^{\mathcal{S}}) \times \mathcal{K}^s(\mathcal{F}^{\mathcal{S}}) \to \mathcal{K}^s(\mathcal{F}^{\mathcal{S}})$ defined by

$$Z_s^{\mathcal{S}}(\uparrow s, \uparrow t) = \uparrow(s \to t).$$

(ii) For $(D, Z) \in \mathbf{ZS}^s$, define $\mathcal{M}_s(\langle D, Z\rangle) \in \mathbf{TS}^{\text{-U}}$ by

$$\mathcal{M}_s(\langle D, Z\rangle) = \langle K^s(D), \leq, \cap, \to_Z\rangle,$$

with $\leq, \cap, \to_Z$ as in Definition 17.3.8.

(iii) The action of the maps $\mathcal{M}_s$ and $\mathcal{Q}_s$ on morphisms are defined as follows. Given $f \in \mathbf{TS}^{\text{-U}}(\mathcal{S}, \mathcal{S}')$, $X \in \mathcal{F}^{\mathcal{S}}$, and $g \in \mathbf{ZS}^s(\langle D, Z\rangle, \langle D', Z'\rangle)$, define

$$\begin{aligned} \mathcal{Q}_s(f)(X) &= \begin{cases} \{t \mid \exists s \in X. f(s) \leq t\}, & \text{if } X \neq \bot\ (= \emptyset), \\ \bot, & \text{else;} \end{cases} \\ \mathcal{M}_s(g) &= g \restriction \mathcal{K}(D). \end{aligned}$$

Note that indeed $\mathcal{M}_s(\langle D, Z\rangle) \in \mathbf{TS}^{\text{-U}}$ by Lemma **??**.???

17.3.16. THEOREM. [**other-equivalences**]

(i) *The restrictions of $\mathcal{Q}$ and $\mathcal{M}$ set up the following equivalences of categories:*

$$\begin{aligned} \mathbf{LTS}^{\text{U}} &\cong \mathbf{LZS} \\ \mathbf{NTS}^{\text{U}} &\cong \mathbf{NZS} \end{aligned}$$

In both cases, natural isomorphisms are given by the maps $\sigma_{\mathcal{S}} : \mathcal{S} \to \mathcal{K}(\mathcal{F}^{\mathcal{S}})$ and $\tau_D : \mathcal{F}^{\mathcal{K}(D)} \to D$ defined by

$$\begin{aligned} \sigma_{\mathcal{S}}(s) &= \uparrow s \\ \tau_D(x) &= \bigsqcup x \end{aligned}$$

(ii) *The restrictions of $\mathcal{Q}_s$ and $\mathcal{M}_s$ set up the following equivalences of categories:*

$$\begin{aligned} \mathbf{TS}^{\text{-U}} &\cong \mathbf{ZS}^s \\ \mathbf{PTS}^{\text{-U}} &\cong \mathbf{PZS}^s \end{aligned}$$

In both cases, natural isomorphisms are given by the maps $\sigma_{\mathcal{S}} : \mathcal{S} \to \mathcal{K}^s(\mathcal{F}_s^{\mathcal{S}})$ and $\tau_D : \mathcal{F}_s^{\mathcal{K}^s(D)} \to D$ defined by

$$\begin{aligned} \sigma_{\mathcal{S}}(s) &= \uparrow s \\ \tau_D(x) &= \bigsqcup x \end{aligned}$$

17.4. Zip and lambda structures

[zip.lambda]

In this section we do not proceed to a direct comparison between type and lambda structures, but put at work zip structures and compare them with lambda structures. Zip structures and lambda structures have some components in common such as the ω-algebraic lattices and their compact elements. This clarifies the comparision and avoids the confusion which arises with the reversed order and the use of filters when passing from type to lambda structures or vice-versa. We will see that there is no categorical equivalence between zip and lambda structures in the general case and how the correspondence is perfect in the lazy, natural, and proper cases. Then, the relation between type and lambda structures will be a consequence of the combined results of this section and the previous one.

Justifying morphisms in LS

This section justifies the choice of morphisms between lambda structures. We anticipated their definitions in previous section, and in this section we substantiate that choice.

17.4.1. Definition. **[application.TS]** Let $\langle D, Z\rangle \in \mathbf{ZS}$.

(i) $\forall x, y \in D.\Theta_{D_Z}(x,y) = \{b \mid \exists a \sqsubseteq y.Z(a,b) \sqsubseteq x\}$

(ii) $\mathcal{N}(\langle D, Z\rangle)$ is defined as the lambda structure $\langle D, F_Z, G_Z\rangle$, with the two continuous functions $F_Z : D \to [D \to D]$ and $G_Z : [D \to D] \to D$ defined by

$$\begin{aligned} F_Z(x) &= \lambda y.\bigsqcup \Theta_{D_Z}(x,y); \\ G_Z(f) &= \bigsqcup\{Z(a,b) \mid b \sqsubseteq f(a)\}. \end{aligned}$$

Moreover we define $\cdot_Z : D^2 \to D$, by

$$x \cdot_Z y = F_Z(x)(y)$$

17.4.2. Proposition. **[for-choice-mor]** *Let $\underline{m} = \langle m, m^R\rangle \in \mathbf{ZS}(\langle D, Z\rangle, \langle D', Z'\rangle)$. Then $\underline{m}$ satisfies the following properties, for all $f : D \to D$, $x \in D$, $x' \in D'$:*

$$\begin{aligned} m(G_Z(f)) &\sqsubseteq G'_{Z'}(m \circ f \circ m^R) \\ F_Z(m^R(x'))(x) &\sqsubseteq m^R(F'_{Z'}(x')(m(x)) \end{aligned}$$

Proof. In order to simplify notation, we omit all the subscripts. We have

$$\begin{aligned} m(G(f)) &= m(\bigsqcup\{Z(a,b) \mid b \sqsubseteq f(a)\} \\ &= \bigsqcup\{m(Z(a,b)) \mid b \sqsubseteq f(a)\}, && \text{by 17.2.2(i)}, \\ &= \bigsqcup\{Z'(m(a), m(b)) \mid b \sqsubseteq f(a)\}, && \text{by } (Z\text{-comm}), \\ &\sqsubseteq \bigsqcup\{Z'(a',b') \mid \exists a, b.m(a) \sqsubseteq a' \ \&\ b' \sqsubseteq m(b) \ \&\ b \sqsubseteq f(a)\} \\ &= \bigsqcup\{Z'(a',b') \mid \exists a.m(a) \sqsubseteq a' \ \&\ b' \sqsubseteq m(f(a))\}, && \text{since } f \text{ is continuous}, \\ &= \bigsqcup\{Z'(a',b') \mid b' \sqsubseteq m(f(m^R(a')))\}, && \text{by (Galois)}, \\ &= G'(m \circ f \circ m^R). \end{aligned}$$

$$\begin{array}{lcll}
F(m^R(x'))(x) & = & \bigsqcup\{b \mid \exists a \sqsubseteq x.Z(a,b) \sqsubseteq m^R(x')\} & \\
& \sqsubseteq & \bigsqcup\{b \mid \exists a \sqsubseteq x.m(Z(a,b)) \sqsubseteq x'\}, & \text{by (Galois-2),} \\
& = & \bigsqcup\{b \mid \exists a \sqsubseteq x.Z'(m(a),m(b)) \sqsubseteq x'\}, & \text{by } (Z\text{-comm}), \\
& \sqsubseteq & \bigsqcup\{b \mid \exists a' \sqsubseteq m(x).Z'(a',m(b)) \sqsubseteq x'\} & \\
& \sqsubseteq & \bigsqcup\{b \mid m(b) \sqsubseteq \bigsqcup\{b' \mid \exists a' \sqsubseteq m(x).Z'(a',b') \sqsubseteq x'\}\} & \\
& = & m^R(\bigsqcup\{b' \mid \exists a' \sqsubseteq m(x).Z'(a',b') \sqsubseteq x'\}), & \text{see 17.5.9,} \\
& = & m^R(F'(x')(m(x))). \blacksquare &
\end{array}$$

We recall that the category **LS** has been defined in Definition 17.2.7, using Definition 17.2.5. Note that the conditions (lambda-gc1) and (lambda-gc2) of Definition 17.2.5 are expressed in the thesis of previous Proposition 17.4.2. As a consequence, it is immediate that $\mathcal{N}$ can be extended to a functor.

17.4.3. PROPOSITION. **[zzll-functor]** *Given* $\underline{m} \in \mathbf{ZS}(D,D')$, *define* $\mathcal{N}(\underline{m}) = \underline{m}$. *Then* $\mathcal{N} : \mathbf{ZS} \to \mathbf{LS}$ *is a functor.*

PROOF. We only need to show that $\mathcal{N}(\underline{m}) = \underline{m}$ is in $\mathbf{LS}(\mathcal{N}(D),\mathcal{N}(D'))$, when $\underline{m} \in \mathbf{ZS}(D,D')$. This follows from Proposition 17.4.2. ■

As a consequence of last proposition, we obtain that the fundamental operation $\mathcal{A} = \mathcal{N} \circ \mathcal{Q}$ is actually a functor from $\mathbf{TS}^{\mathbf{U}}$ to **LS** (see Theorem 17.4.31). Our notion of morphism between lambda structures guarantees the functoriality of $\mathcal{A}$. In Plotkin [1993] a different definition of morphism between lambda structures is given: a morphism $m : \langle D,F,G\rangle \to \langle D',F',G'\rangle$ between lambda structures consists of a pair of continuous functions, $m : D \to D'$ and $n : D' \to D$ such that

$$(*) \qquad \begin{array}{lcl} m \circ G & = & G' \circ (n \to m) \\ (m \to n) \circ F' & = & F \circ n \end{array}$$

where $n \to m : [D \to D] \to [D' \to D']$ is defined, for any $f : D \to D$, by $(n \to m)(f) = m \circ f \circ n : D' \to D'$ (and similarly is defined $n \circ m : [D' \to D'] \to [D \to D]$). This choice allows to look at lambda structures as *dialgebras*, but does not guarantees a functorial behaviour of $\mathcal{A}$ in the general case.

The conditions (lambda-gc1) and (lambda-gc2) defining our notion of morphism between lambda structures arise from Proposition 17.4.2 and they are obviously weaker than $(*)$.

The equivalence between ZS and LS fails

Since, in a general lambda structure $\mathcal{D} = \langle D,F,G\rangle$, there is no connection between F and G, it is not a surprise that there are lambda structures $\mathcal{D}$ not isomorphic to $\mathcal{N}(D_Z)$, for any zip structure D_Z. An easy counterexample is given.

Consider for instance $\tilde{D}$ as any non-trivial ω-algebraic lattice, and let $F : D \to [D \to D]$, $G : [\tilde{D} \to \tilde{D}] \to \tilde{D}$ defined by:

$$\begin{array}{lcl} F(x) & = & \lambda y.\bot \\ G(f) & = & \bot \end{array}$$

Let $\mathcal{D} = \langle \tilde{D}, F, G\rangle$. Of course if we look for a zip structure D_Z such that $\mathcal{N}(D_Z)$ is isomorphic to $\mathcal{D}$, it is not restrictive to suppose $D_Z = \langle \tilde{D}, Z\rangle$. Now, for no $Z : \tilde{D} \to \tilde{D}$, we have $F_Z = F$ and $G_Z = G$. In fact,

$$\begin{aligned} F_Z(x)(y) &= \bigsqcup\{b \mid \exists a \sqsubseteq y.Z(a,b) \sqsubseteq x\} \\ &= \bigsqcup\{b \mid \exists a \sqsubseteq y.G(a \mapsto b) \sqsubseteq x\} \\ &= \top. \end{aligned}$$

From the equivalence between $\mathbf{TS}^{\mathtt{U}}$ and **ZS** of Theorem 17.3.14 and the previous counterexample, it follows that $\mathbf{TS}^{\mathtt{U}}$ and **LS** are not equivalent.

Various categories of lambda structures

17.4.4. Definition. Let $D = \langle D, F, G\rangle$ be a lambda structure. Write

$$\delta := G(\bot \Rightarrow \bot).$$

We say that D is a *lazy* lambda structure if the following holds.

(i)	(δ-comp)	$\delta \in \mathcal{K}(D)$;
(ii)	(adj1)	$\forall f \in [D \to D].F(G(f)) \sqsupseteq f$;
(iii)	(adj2)	$\forall x \in D.\delta \sqsubseteq x \;\Rightarrow\; G(F(x)) \sqsubseteq x$;
(iv)	($\delta\bot$)	$\forall x \in D.\delta \not\sqsubseteq x \;\Rightarrow\; F(x) = \bot \mapsto \bot$.

17.4.5. Definition. A *strict lambda structure*, notation LS^{s}, is a triple $\langle D, F, G\rangle$, where $D \in \mathbf{ALG}$, and $F : D \to_{\mathrm{s}} [D \to_{\mathrm{s}} D]$ and $G : [D \to_{\mathrm{s}} D] \to_{\mathrm{s}} D$ are continuous.

We now give the definition of various categories of lambda structures. This definition is an expansion of Definition 17.2.7. By sake of completeness, we repeat the definition of the categories of lambda structures and natural lambda structures.

17.4.6. Definition. (i) The category **LS** consists of lambda structures as objects and lambda Galois connections as morphisms. The composition between morphisms $\langle \mathsf{m}, \mathsf{m}^R\rangle : \mathcal{D} \to \mathcal{D}'$ and $\langle \mathsf{n}, \mathsf{n}^R\rangle : \mathcal{D}' \to \mathcal{D}''$ is given by $\langle \mathsf{n} \circ \mathsf{m}, \mathsf{m}^R \circ \mathsf{n}^R\rangle$.

(ii) The category **LLS** is the full subcategory of **LS** which has as objects lazy lambda structures.

(iii) The category of *natural lambda structures*, notation **NLS**, is the full subcategory of **LS** which has as objects natural lambda structures.

(iv) The category of *strict* lambda structures, notation $\mathbf{LS}^s$, has as objects strict lambda structures and as morphisms special Galois connections m such that m and m^R are strict.

(v) A *proper* lambda structure, notation PLS^s is a strict lambda structure $\mathcal{D}$ such that $\langle G, F\rangle$ is a Galois connection.

(vi) $\mathbf{PLS}^s$ is the full subcategory of $\mathbf{LS}^s$ having as objects proper lambda structures.

We will establish the following equivalences of categories.

$$\begin{aligned} \mathbf{LZS} &\cong \mathbf{LLS}; \\ \mathbf{NZS} &\cong \mathbf{NLS}; \\ \mathbf{PZS}^s &\cong \mathbf{PLS}^s. \end{aligned}$$

Isomorphism between LZS and LLS

In this subsection we see how the correspondence between zip structures and lambda structures becomes very smooth (an isomorphism of categories) in the case of lazy structures. We start with some technical preliminary result.

17.4.7. LEMMA. *Let $\langle D, F, G\rangle$ be a lazy lambda structure. Then*

$$\forall f \in [D \to D'] \forall x \in D. f \neq (\bot \mapsto \bot) \;\Rightarrow\; [G(f) \sqsubseteq x \iff f \sqsubseteq F(x)].$$

PROOF. Do Exercise 17.5.14. ■

Recall that Θ_{D_Z} is defined in Definition 17.4.1.

17.4.8. REMARK. Note that by (Z-contr) and (Z-add) one has in **LZS**
(i) $\forall a \in \mathcal{K}(D). Z(a, \bot) = Z(\bot, \bot)$.
(ii) $\Theta_{D_Z}(x, y) \neq \emptyset \iff \Theta_{D_Z}(x, y)$ is directed.

17.4.9. LEMMA. *Let $\langle D, Z\rangle \in$ **LZS** and let $a, b \in \mathcal{K}(D)$, $x \in D$, with $b \neq \bot$. Then*

$$b \sqsubseteq x \cdot a \;\Leftrightarrow Z(a, b) \sqsubseteq x.$$

PROOF. ($\Leftarrow$) follows immediately from the definition of application.
We prove ($\Rightarrow$). We have

$$\begin{array}{lcl} b \sqsubseteq x \cdot a & \Leftrightarrow & b \sqsubseteq \bigsqcup \Theta_{D_Z}(x, a), \\ & \Rightarrow & \exists a_1, b_1. b \sqsubseteq b_1 \;\&\; a_1 \sqsubseteq a \;\&\; Z(a_1, b_1) \sqsubseteq x, \text{by Remark 17.4.7}, \\ & \Rightarrow & Z(a, b) \sqsubseteq x. \blacksquare \end{array}$$

17.4.10. PROPOSITION. *Let $\langle D, Z\rangle$ be a lazy zip structure. Then*
(i) $G_Z(a \mapsto b) = Z(a, b)$.
(ii) $\mathcal{N}(\langle D, Z\rangle) = \langle D, F_Z, G_Z\rangle$ *is a lazy lambda structure.*

PROOF. (i) We have the following.

$$\begin{array}{lcll} G_Z(a \mapsto b) & = & \bigsqcup\{Z(a', b') \mid b' \sqsubseteq (a \mapsto b)a'\} & \\ & = & \bigsqcup\{Z(a', b') \mid a \sqsubseteq a' \;\&\; b' \sqsubseteq b\}, & \text{by Remark 17.4.8(i)}, \\ & = & Z(a, b), & \text{by } (Z\text{-contr}). \end{array}$$

(ii) We prove that $\mathcal{N}(\langle D, Z\rangle)$ satisfies the four points of Definition 17.4.4. We have $G(\bot \mapsto \bot) = Z(\bot, \bot)$, by (i). Therefore ($\delta$-comp) holds, since $Z(\bot, \bot)$ is compact.

As to (adj1), it is sufficient to reason about compact elements, and prove that for any a, b,

$$b \sqsubseteq f(a) \;\Rightarrow b \sqsubseteq F(G(f))(a).$$

Notice that if $b \sqsubseteq f(a)$, then $b \in \Theta_{D_Z}(G(f), a)$, so

$$\begin{array}{lcl} b & \sqsubseteq & \bigsqcup \Theta_{D_Z}(G(f), a), \\ & = & G(f) \cdot a, \\ & = & F(G(f))(a). \end{array}$$

Now we prove (adj2). Suppose $\delta \sqsubseteq x$, that is $Z(a,\bot) \sqsubseteq x$ for any a. Since $G(F(x)) = \bigsqcup\{Z(a,b) \mid b \sqsubseteq x \cdot a\}$, it is enough to prove that $Z(a,b) \sqsubseteq x$ whenever $b \sqsubseteq x \cdot a$. There are two cases. If $b = \bot$, then the thesis follows from the hypothesis. If $b \neq \bot$, then the thesis follows from Lemma 17.4.9.

Finally we prove ($\delta\bot$). By (Z-lazy) it follows that $Z(a,b) \not\sqsubseteq x$, for all a, b. So $F(x)(y) = \bot$, for any y. ■

From Propositions 17.4.10 and 17.4.3 we get the following.

17.4.11. THEOREM. *$\mathcal{N}$ restricts to a functor from* **LZS** *to* **LLS**.

Going the other direction, from any lazy lambda structure one can define a lazy zip structure. Before showing that, we need to extend Lemma 17.2.2(i), (ii) to lazy lambda structures.

17.4.12. LEMMA. *Let $\langle D, F, G\rangle$ be a lazy lambda structure. Then*
(i) *G is additive.*
(ii) *$\forall f \in \mathcal{K}([D \to D]).G(f) \in \mathcal{K}(D)$.*

PROOF. (i) Similar to the proof of Lemma 17.2.2(i).

(ii) If $f = (\bot \mapsto \bot)$ then $G(f) \in \mathcal{K}(D)$ by Definition 17.4.4(i). If on the other hand $f \neq (\bot \mapsto \bot)$, then the proof is similar to that of Lemma 17.2.2(ii), using Lemma 17.4.7. ■

17.4.13. DEFINITION. [**ZFG**] Let $\mathcal{D} = \langle D, F, G\rangle$ be a lazy lambda structure. Then we can define, for any $a, b \in \mathcal{K}(D)$,

$$Z_{F,G}(a,b) = G(a,b)$$

Because of Lemma 17.4.12(ii) we obtain a zip structure $\mathcal{R}(\mathcal{D}) = \langle D, Z_{F,G}\rangle$.

17.4.14. PROPOSITION. *Let $\mathcal{D} = \langle D, F, G\rangle$ be a lazy lambda structure. Then $\mathcal{R}(\mathcal{D})$ is a lazy zip structure.*

PROOF. First of all notice that Z is well-defined since by Lemma 17.4.12(ii), $Z(a,b)$ is a compact element.

We prove (Z-contr). Let $a \sqsubseteq a'$, $b' \sqsubseteq b$. Then, in $[D \to D]$, $a' \mapsto b' \sqsubseteq a \mapsto b$, hence $G(a' \mapsto b') \sqsubseteq G(a \mapsto b)$. By definition of Z, this implies $Z(a',b') \sqsubseteq Z(a,b)$ as desired.

We prove (Z-add). We have

$$\begin{array}{rcll} Z(a, b_1 \sqcup b_2) & = & G(a \mapsto (b_1 \sqcup b_2)), & \text{by definition,} \\ & = & G(a \mapsto b_1) \sqcup G(a \mapsto b_2), & \text{by Lemma 17.4.12(i).} \end{array}$$

Finally, (Z-lazy) is immediate by monotonicity of G and the fact that $(\bot \Rightarrow \bot) \sqsubseteq (a \mapsto b)$, for all $a, b \in \mathcal{K}(D)$. ■

17.4.15. LEMMA. *Let $\underline{m} = \langle m, m^R\rangle \in \mathbf{LLS}(\mathcal{D}, \mathcal{D}'_{F',G'})$, where $\mathcal{D} = \langle D, F, G\rangle$, $\mathcal{D}' = \langle D', F', G'\rangle$. Define $\mathcal{R}(\underline{m}) = m$. Then $\mathcal{R}(\underline{m}) \in \mathbf{LZS}(\mathcal{R}(\mathcal{D}), \mathcal{R}(\mathcal{D}'))$.*

PROOF. $\mathcal{R}(\underline{m}) = m$ satisfies (cmp-pres) and (add) by Lemma 17.4.12. So we are left to prove that m satisfies (Z-comm), that is, for any $a, b \in K(D)$,

$$m(Z(a,b)) = Z'(m(a), m(b))$$

where $Z = Z_{F,G}$, $Z' = Z_{F',G'}$. We have

$$\begin{array}{rcll} m(Z(a,b)) & = & m(G(a \mapsto b)), & \text{by definition of } Z, \\ & \sqsubseteq & G'(m \circ (a \mapsto b) \circ m^R), & \text{by Definition 17.2.5(i)}, \\ & = & G'(m(a) \mapsto m(b)), & \text{by Lemma 17.2.2(iii)}, \\ & = & Z'(m(a), m(b)). & \end{array}$$

On the other hand, being by definition $(a \mapsto b)(a) = b$

$$\begin{array}{rcll} b \sqsubseteq (a \mapsto b)(a) & \Rightarrow & b \sqsubseteq (F(G(a \mapsto b))(a), & \text{by (adj1)}, \\ & \Rightarrow & m(b) \sqsubseteq m((F(G(a \mapsto b))(a)) & \\ & \Rightarrow & m(b) \sqsubseteq F'(m(G(a \mapsto b)))(m(a)), & \text{by Definition 17.2.5(ii)}, \\ & \Rightarrow & m(a) \mapsto m(b) \sqsubseteq F'(m(G(a \mapsto b))) & \\ & \Rightarrow & G'(m(a) \mapsto m(b)) \sqsubseteq m(G(a \mapsto b)), & \text{by (adj2)}, \\ & \Rightarrow & Z'(m(a), m(b)) \sqsubseteq m(Z(a,b)). \blacksquare & \end{array}$$

From Proposition 17.4.14 and Lemma 17.4.15 we obtain the following.

17.4.16. THEOREM. $\mathcal{R} : \mathbf{LLS} \to \mathbf{LZS}$ *is a functor.* ■

Actually, $\mathcal{R}$ and $\mathcal{N}$ set up an isomorphism between **LLS** and **LZS**. So the correspondence between **LLS** and **LZS** is perfect.

17.4.17. THEOREM. (i) $\mathcal{N} \circ \mathcal{R} = \mathsf{Id}_{\mathbf{LLS}}$.
(ii) $\mathcal{R} \circ \mathcal{N} = \mathsf{Id}_{\mathbf{LZS}}$.

PROOF. (i) We have to prove prove that for any lazy lambda structure $\mathcal{D} = \langle D, F, G\rangle$, $\mathcal{N}(\mathcal{R}(\mathcal{D})) = \mathcal{D}$. This is equivalent to prove that

$$F_{Z_{F,G}} = F, \quad G_{Z_{F,G}} = G.$$

The proof is left to the reader.

(ii) For any lazy zip structure $\langle D, Z\rangle$, we have that $\mathcal{R}(\mathcal{A}(\langle D, Z\rangle)) = \langle D, Z\rangle$. For this aim, it is enough to prove that $Z_{G_Z} = Z$.

$$\begin{array}{rcll} Z_{G_Z}(a,b) & = & G_Z(a \mapsto b) & \\ & = & Z(a,b), & \text{by Proposition 17.4.10(i).} \blacksquare \end{array}$$

17.4.18. THEOREM. [**ls-galois-iso**] *The categories* **LZS** *and* **LLS** *are isomorphic.*

Isomorphism between NZS and NLS

In this short subsection we specialize the results of the previous subsection to natural zip structures.

As expected, natural lambda structures are a specialization of the lazy ones: in a lambda structure $\mathcal{D} = \langle D, F, G\rangle$, F and G set up a Galois connection if and only if $\mathcal{D}$ is a lazy lambda structure with $\delta = \bot$.

17.4.19. LEMMA. [**lazy-generalize**] *Let $\mathcal{D} = \langle D, F, G\rangle$ be a lambda structure. Then $\mathcal{D}$ is natural (that is $\langle G, F\rangle$ is a Galois connection) if and only if $\mathcal{D}$ is a lazy lambda structure with $\delta = \bot$.*

17.4.20. PROPOSITION. [**GLS-induce-gc**] *Let $\langle D, Z\rangle$ be a natural zip structure. Then $\mathcal{N}(\langle D, Z\rangle) = \langle D, F_Z, G_Z\rangle$ is a natural lambda structure.*

PROOF. We conclude immediately by Lemma 17.4.19, since $Z(\bot, \bot) = \bot$, hence $G_Z(\bot \mapsto \bot) = \bot$. ■

So we get the following.

17.4.21. THEOREM. [**galoisfunctor-on-galois**][**zccmp-nat**] *$\mathcal{N}$ restricts to a functor from* **NZS** *to* **NLS**.

We now go the other direction.

17.4.22. PROPOSITION. [**lgc-induce-LLS2**] *Let $\mathcal{D} = \langle D, F, G\rangle$ be a natural lambda structure. Then $\mathcal{R}(\mathcal{D}) = \langle D, Z_{F,G}\rangle$ is a natural zip structure.*

PROOF. By Proposition 17.4.14, we just have to prove that $Z_{F,G}(\bot, \bot) = \bot$. By Lemma 17.4.19 we know that $G(\bot \mapsto \bot) = \bot$, so we are done by the definition of $Z_{F,G}$. ■

17.4.23. COROLLARY. [**lsfunctor-on-galois**] *$\mathcal{R}$ restricts to a functor from* **NLS** *to* **NZS**.

17.4.24. THEOREM. [**ls-galois-iso2**] *The categories* **NLS** *and* **NZS** *are isomorphic.*

PROOF. From Theorem 17.4.18 and the fact that **NLS** and **NZS** are full subcategories of **LLS** and **LZS** respectively. ■

Isomorphism between $\mathbf{PZS}^s$ and $\mathbf{PLS}^s$

In this final subsection we specialize the previous results to proper strict zip structures. Most proofs of this section are left to the reader.

First we introduce the functors $\mathcal{N}_s$ and $\mathcal{R}_s$.

17.4.25. LEMMA. *Let $\langle D, Z\rangle \in \mathbf{ZS}^s$. Then $\langle D, Z\rangle$ induces the continuous maps $F_Z : D \to_s [D \to_s D]$ and $G_Z : [D \to_s D] \to_s D$ defined by*

$$\begin{aligned} F_Z(x) &= \lambda y. \bigsqcup \Theta_{D_Z}(x, y); \\ G_Z(f) &= \bigsqcup \{Z(a, b) \mid b \sqsubseteq f(a)\}. \end{aligned}$$

Usually we will omit the Z.

PROOF. (i) We have to prove that F_Z and G_Z are strict. F_Z is strict since $\langle \bot, b\rangle$ is not in the domain of Z.

We prove that G_Z is strict. If $b \sqsubseteq (\bot \Rightarrow \bot)(a)$, then $b = \bot$. Moreover, $\langle a, \bot\rangle$ is not in the domain of Z. ■

17.4.26. DEFINITION. [llzzs]
(i) For $\langle D, Z\rangle \in \mathbf{PZS}^s$, define $\mathcal{N}_s(\langle D, Z\rangle) = \langle D, F_Z, G_Z\rangle$. Given $m \in \mathbf{PZS}^s(\langle D, Z\rangle, \langle D', Z'\rangle)$, define $\mathcal{N}_s(m) = \langle m, m^R\rangle \in \mathbf{PLS}^s(\mathcal{N}_s(\langle D, Z\rangle), \mathcal{N}_s(\langle D', Z'\rangle))$
(ii) For $\mathcal{D} = \langle D, F, G\rangle \in \mathbf{PLS}^s$, define $\mathcal{R}_s(\mathcal{D}) = \langle D, Z_{F,G}\rangle$, where, for any $a, b \in \mathcal{K}^s(D)$, $Z_{F,G}(a, b) = G(a \mapsto b)$. Given $\underline{m} = \langle m, m^R\rangle \in \mathbf{PLS}^s(\mathcal{D}, \mathcal{D}')$,define $\mathcal{R}_s(\underline{m}) = m$.

Actually both $\mathcal{N}_s$ and $\mathcal{R}_s$ are functors.

17.4.27. THEOREM. (i) *$\mathcal{N}_s$ is a functor from $\mathbf{PZS}^s$ to $\mathbf{PLS}^s$.*
(ii) *$\mathcal{R}_s$ is a functor from $\mathbf{PLS}^s$ to $\mathbf{PZS}^s$.*

PROOF. (i) Similar to Proposition 17.4.20.
(ii) Similar to Proposition 17.4.22. ■

17.4.28. THEOREM. *The categories $\mathbf{PZS}^s$ and $\mathbf{PLS}^s$ are isomorphic.*

Equivalences between type and lambda structures

In this subsection we can establish the equivalences between categories of type and lambda structures. The notion of filter structure $\langle \mathcal{F}^{\mathcal{S}}, F^{\mathcal{S}}, G^{\mathcal{S}}\rangle$ over a type structure $\mathcal{S}$ given in Definition 15.4.5 can be seen as a functor $\mathcal{A}$ from $\mathbf{TS}^{\mathsf{U}}$ to $\mathbf{LS}$. Since many classical models of λ-calculus are (or could be) defined as $\mathcal{A}(\mathcal{S})$ for suitable type structures $\mathcal{S}$, one fundamental question is whether it is possible to describe any lambda structure as the filter space of a suitable type structure. The answer is negative for the same reason that the equivalence between zip structures and lazy lambda structures fails. In the general case, lambda structures are not captured by type structures, and no categorical equivalence result is possible. But as far as we restrict to the lazy, natural, or proper case, then the correspondence is perfect, and assumes the shape of categorical equivalences.

17.4.29. DEFINITION. Let $\mathcal{S} \in \mathbf{TS}^{\mathsf{U}}$. We define the operation $\mathcal{A}$ as in Lemma 17.2.9 and Theorem 17.2.10.

1. For $\mathcal{S} \in \mathbf{TS}^{\mathsf{U}}$, define $\mathcal{A}(\mathcal{S}) = \langle \mathcal{F}^{\mathcal{S}}, F^{\mathcal{S}}, G^{\mathcal{S}}\rangle$.
2. For $f \in \mathbf{TS}^{\mathsf{U}}(\mathcal{S}, \mathcal{S}')$, define $\mathcal{A}(f) = \underline{f} = \langle \mathsf{Flt}(f), \mathsf{Flt}(f)^R\rangle$

17.4.30. LEMMA. [**17.4.12b**] *$\mathcal{A}$ is the composition of $\mathcal{Q}$ and $\mathcal{N}$.*

PROOF. Given $f \in \mathbf{TS}^{\mathsf{U}}(\mathcal{S}, \mathcal{S}')$, it is easy to see that $\mathcal{N}(\mathcal{Q}(f)) = \mathcal{A}(f)$. Let $\mathcal{S} \in \mathbf{TS}^{\mathsf{U}}$. We will prove that $\mathcal{N}(\mathcal{Q}(\mathcal{S})) = \mathcal{A}(\mathcal{S})$. By Definitions 17.3.8 and 17.4.1, we have to prove that $F^{\mathcal{S}} = F_{Z^{\mathcal{S}}}$ and $G^{\mathcal{S}} = G_{Z^{\mathcal{S}}}$. Taking the suprema in $\mathcal{F}^{\mathcal{S}}$ one has

$$\begin{aligned} F^{\mathcal{S}}(X)(Y) &= \uparrow\{\uparrow A \mid \exists B \in Y.(B \to A) \in X\} \\ &= \bigsqcup\{\uparrow A \mid \exists B \in Y.\uparrow(B \to A) \subseteq X\} \\ &= \bigsqcup\{\uparrow A \mid \exists \uparrow B \subseteq Y.Z^{\mathcal{S}}(\uparrow B, \uparrow A) \subseteq X\} \\ &= F_{Z^{\mathcal{S}}}(X)(Y). \end{aligned}$$

Moreover,

$$\begin{aligned} G^{\mathcal{S}}(f) &= \uparrow\{B{\to}A \mid A \in f(\uparrow B)\} \\ &= \bigsqcup\{\uparrow(B{\to}A) \mid A \in f(\uparrow B)\} \\ &= \bigsqcup\{Z(\uparrow B, \uparrow A) \mid \uparrow A \subseteq f(\uparrow B)\} \\ &= G_{Z^{\mathcal{S}}}(f). \blacksquare \end{aligned}$$

17.4.31. THEOREM. [galois-functor-on-lazy][galoisfunctor-functor]
(i) $\mathcal{A} : \mathbf{TS}^{\mathtt{U}} \to \mathbf{LS}$ *is a functor.*
(ii) $\mathcal{A}$ *restricts to a functor from* $\mathbf{LTS}^{\mathtt{U}}$ *to* **LLS**.
(iii) $\mathcal{A}$ *restricts to a functor from* $\mathbf{NTS}^{\mathtt{U}}$ *to* **NLS**.

PROOF. (i) By Lemma 17.4.30, $\mathcal{A}$ is a functor since it is the composition of two functors. (ii), (iii) By Theorem 17.3.16(i) along with Theorem 17.4.11 for the lazy case, and Theorem 17.4.21 for the natural case. ■

17.4.32. THEOREM. [ls2] *Define* $\mathcal{L} : \mathbf{LLS} \to \mathbf{LTS}^{\mathtt{U}}$ *as in Definition 17.2.11.*
(i) *Given a lazy lambda structure* $\mathcal{D}$*, we define*

$$\mathcal{L}(\mathcal{D}) = \langle \mathcal{K}(D), \leq, \cap, \to_D, \mathtt{U} \rangle$$

(ii) *Given* $\underline{\mathsf{m}} \in \mathbf{LLS}(\mathcal{D}, \mathcal{D}')$*, we define*

$$\mathcal{L}(\underline{\mathsf{m}}) = \mathsf{m} \restriction \mathcal{K}(D) : \mathcal{K}(D) \to \mathcal{K}(D').$$

Then $\mathcal{L}$ *is a functor. Moreover* $\mathcal{L}$ *restricts to a functor from* **NLS** *to* $\mathbf{NTS}^{\mathtt{U}}$.

PROOF. It follows from the fact that $\mathcal{L}$ is a composition of $\mathcal{R}$ with $\mathcal{M}$, which are functors as proved in Theorem 17.4.16 and Corollary 17.4.23 and 17.3.16(i), both for the lazy and natural case. ■

17.4.33. THEOREM. *The categories* $\mathbf{LTS}^{\mathtt{U}}$ *and* **LLS** *are equivalent.*

PROOF. By Theorem 17.4.18 and Theorem 17.3.16(i). ■

17.4.34. THEOREM. [nts-nls-equ] *The categories* $\mathbf{NTS}^{\mathtt{U}}$ *and* **NLS** *are equivalent.*

PROOF. By Theorem 17.4.24 and Theorem 17.3.16(i). ■

17.4.35. THEOREM. *The categories* $\mathbf{PTS}^{-\mathtt{U}}$ *and* $\mathbf{PLS}^{s}$ *are equivalent.*

PROOF. By Theorem 17.4.28 and Theorem 17.3.16(ii). ■

17.5. Exercises

17.5.1. Let $\mathcal{S} \in \mathrm{TS}^{-\mathtt{U}}$. Show that
$$\begin{array}{rcll} \emptyset \cdot X & = & \emptyset & \text{for all } X \in \mathcal{F}^{\mathcal{S}} \\ X \cdot \emptyset & = & \emptyset & \text{for all } X \in \mathcal{F}^{\mathcal{S}}. \end{array}$$

17.5.2. Let $\mathcal{S}$ be a natural and β-sound type structure. Show that

$$\uparrow \textstyle\bigcap_{i \in I}(A_i \to B_i) \cdot \uparrow C \;=\; \bigcap_{j \in I} B_j,$$

where $J = \{i \in I \mid C \leq A_i\}$.

17.5.3. Let $\mathcal{S}$ be a top type structure. Show that

$$F^{\mathcal{S}}(G^{\mathcal{S}}(\bot \mapsto \bot)) = (\bot \mapsto \bot) \;\Rightarrow$$
$$B = \mathtt{U} \text{ whenever } A \to B = \mathtt{U} \to \mathtt{U}.$$

This exercise is very similar to Ex. 16.3.17 and particular case of next one. Delete???

17.5.4. Let $\mathcal{S}$ be a top type structure. Show that

$$F^{\mathcal{S}}(G^{\mathcal{S}}(\bot \mapsto \bot)) = (\bot \mapsto \bot) \iff$$
$$[\textstyle\bigcap_{i \in I}(C_i \to D_i) \leq A \to B \;\&\; \forall i \in I.D_i = \mathtt{U}] \;\Rightarrow\; B = \mathtt{U}.$$

17.5.5. Let $\mathcal{S}$ be an arbitrary type structure. Show that
$$G^{\mathcal{S}}(\textstyle\bigsqcup_{i \in I}(\uparrow A_i \mapsto \uparrow B_i)) \;\supseteq\; \uparrow \bigcap_{i \in I}(A_i \to B_i).$$

17.5.6. Let $\mathcal{S}$ be a proper type structure. Show that
$$G^{\mathcal{S}}(\textstyle\bigsqcup_{i \in I}(\uparrow A_i \mapsto \uparrow B_i)) \;=\; \uparrow \bigcap_{i \in I}(A_i \to B_i).$$

17.5.7. Let $\mathcal{S}$ be a natural type structure. Show that $G^{\mathcal{S}}(\uparrow \mathtt{U} \mapsto \uparrow \mathtt{U}) = \uparrow \mathtt{U}$.

17.5.8. Show that (Galois-1) and (Galois-2) in Definition 17.2.1 are equivalent with (Galois).

17.5.9. Let $\langle m, m^R \rangle : D \to D'$ be a Galois connection. Show that

(i) $\quad m(x) \;=\; \bigsqcap\{x' \mid x \sqsubseteq m^R(x')\}$;

(ii) $\quad m^R(x') \;=\; \bigsqcup\{x \mid m(x) \sqsubseteq x'\}$.

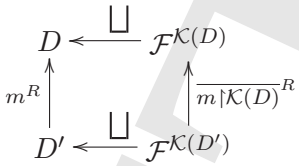
17.5.10. Prove that (i) and (ii) in Lemma **??** follow from (i) and (ii) in Definition 17.2.5, respectively. Paula: which lemma????

17.5.11. Show that for natural lambda structures Definitions 17.2.5 and 17.2.5 are equivalent. Same definition???

17.5.12. Let $f \in \mathbf{NTS}^{\mathtt{U}}(\mathcal{S}, \mathcal{S}')$ and $f' \in \mathbf{NTS}^{\mathtt{U}}(\mathcal{S}', \mathcal{S}'')$. Show that

$$\begin{array}{rcll} \overline{f \circ f'} & = & \overline{f} \circ \overline{f'}; & \\ \overline{f \circ f'}^R & = & \overline{f'}^R \circ \overline{f}^R; & \\ \langle \overline{\mathsf{Id}}, \overline{\mathsf{Id}}^R \rangle & = & \langle \mathsf{Id}, \mathsf{Id} \rangle, & \text{for } \mathsf{Id} = \mathsf{Id}_{\mathcal{S}} : \mathcal{S} \to \mathcal{S}. \end{array}$$

17.5.13. Let $\langle m, m^R\rangle : D{\to}D'$ be a morphism in **LS**. Show the commutativity of the following diagram



Note that the filter X' on $(\mathcal{K}(D'), \leq)$ is a directed subset of $(D' \sqsubseteq)$.

17.5.14. Let $\langle D, F, G\rangle$ be a lazy lambda structure, $f \in [D{\to}D]$ and $x \in D$. Assume $f \neq \bot \mapsto \bot$. Show that

$$G(f) \sqsubseteq x \iff f \sqsubseteq F(x).$$

17.5.15. Let $m \in \mathbf{ZS}(D_Z, D'_{Z'})$, $a, b \in \mathcal{K}(D)$ and $f' \in [D{\to}D']$. Show that

$$(m(a) \mapsto m(b)) \sqsubseteq f' \iff (a \mapsto b) \sqsubseteq (n_m \circ f' \circ m),$$

where n_m is defined as in Definition **??**.

17.5.16. Let $\langle D, F, G\rangle$ be a lazy lambda structure. Show that

$$\text{(func-Galois)} \quad \forall a, b \in \mathcal{K}(D), x \in D.b \sqsubseteq F(x)(a) \iff G(a \mapsto b) \sqsubseteq x.$$

17.5.17. Let $\langle D, F, G\rangle$ and $\langle D', F', G'\rangle$ be natural lambda structures and let $\underline{m} = \langle m, m^R\rangle : D{\to}D'$ be a Galois connection. Show that

$$[\forall x, y \in D.m(Fxy) \sqsubseteq F'(mx)(my)] \Rightarrow [\forall f \in [D{\to}D].m(Gf) \sqsupseteq G'(m \circ f \circ m^R)].$$

17.5.18. Let $\langle D, F, G\rangle$ and $\langle D', F', G'\rangle$ be lambda structures. Assume $\mathsf{m} : D \to D'$ is a bijection such that m and m^{-1} are continuous and the following conditions hold for m.

$$\begin{array}{lrcl} (1) & \mathsf{m}(G(f)) & = & G'(\mathsf{m} \circ f \circ \mathsf{m}^{-1}); \\ (2) & \mathsf{m}(F(d)(e)) & = & F'(\mathsf{m}(d))(\mathsf{m}(e)). \end{array}$$

Prove that m induces an isomorphism of lambda structures $\underline{\mathsf{m}} = \langle \mathsf{m}, \mathsf{m}^{-1}\rangle : \langle D, F, G\rangle \to \langle D', F', G'\rangle$.

17.6. Exercises

17.6.1. Let $\mathcal{S}$ be an arbitrary free type structure. Show that

$$\begin{array}{rcll} \emptyset \cdot X & = & \emptyset & \text{for all } X \in \mathcal{F}^{\mathcal{T}} \\ X \cdot \emptyset & = & \emptyset & \text{for all } X \in \mathcal{F}^{\mathcal{T}}. \end{array}$$

17.6.2. Let $\mathcal{S}$ be a natural and β-sound type structure. Show that

$$\uparrow \textstyle\bigcap_{i \in I}(A_i \to B_i) \cdot \uparrow C \;=\; \bigcap_{j \in J} B_j,$$

where $J = \{i \in I \mid C \leq_{\mathcal{T}} A_i\}$.

17.6.3. Let $\mathcal{S}$ be a top type structure. Show that

$$F^{\mathcal{S}}(G^{\mathcal{S}}(\bot \mapsto \bot)) = (\bot \mapsto \bot) \Rightarrow$$
$$B =_{\mathcal{T}} \mathtt{U} \text{ whenever } A \to B =_{\mathcal{T}} \mathtt{U} \to \mathtt{U}.$$

17.6.4. Let $\mathcal{S}$ be a top type structure. Show that

$$F^{\mathcal{S}}(G^{\mathcal{S}}(\bot \mapsto \bot)) = (\bot \mapsto \bot) \iff$$
$$[\textstyle\bigcap_{i\in I}(C_i \to D_i) \leq_{\mathcal{T}} A \to B \;\&\; \forall i \in I. D_i =_{\mathcal{T}} \mathtt{U}] \;\Rightarrow\; B =_{\mathcal{T}} \mathtt{U}.$$

17.6.5. Let $\mathcal{S}$ be an arbitrary type structure. Show that
$G^{\mathcal{S}}(\bigsqcup_{i\in I}(\uparrow A_i \mapsto \uparrow B_i)) \;\supseteq\; \uparrow \bigcap_{i\in I}(A_i \to B_i)$.

17.6.6. Let $\mathcal{S}$ be a proper type structure. Show that
$G^{\mathcal{S}}(\bigsqcup_{i\in I}(\uparrow A_i \mapsto \uparrow B_i)) \;=\; \uparrow \bigcap_{i\in I}(A_i \to B_i)$.

17.6.7. Let $\mathcal{S}$ be a natural type structure. Show that
$G^{\mathcal{S}}(\bigsqcup_{i\in I}(\uparrow A_i \mapsto \uparrow B_i)) \;=\; \uparrow \bigcap_{i\in I}(A_i \to B_i)$. **Comment:** immediate from previous exercise

17.6.8. Let $\mathcal{S}$ be a natural type structure. Show that $G^{\mathcal{S}}(\uparrow \mathtt{U} \mapsto \uparrow \mathtt{U}) \;=\; \uparrow \mathtt{U}$.

Chapter 18

Models

18.1. Lambda models

In this section we recall some basic notions and properties of λ-models in general that will be used later for filter models, see Barendregt [1984]. Amongst these properties, we show that the only requirement that a (strict) lambda structure misses to be a $\lambda(\mathsf{I})$-model is the satisfiability of $\beta(\mathsf{I})$-conversion. In addition we will characterise extensionality by means of the invariance of typing under η-conversion, given in Section 16.2.

18.1.1. Definition. (i) Let D be a set and Var the set of variables of the untyped lambda calculus. An *environment in* D is a total map

$$\rho : \mathrm{Var} \to D.$$

The set of environments in D is denoted by Env_D.

(ii) If $\rho \in \mathsf{Env}_D$ and $d \in D$, then $\rho[x := d]$ is the $\rho' \in \mathsf{Env}_D$ defined by

$$\begin{array}{lcll} \rho'(x) & = & d; & \\ \rho'(y) & = & \rho(y), & \text{if } y \neq x. \end{array}$$

The definition of a syntactic lambda-models was given in Barendregt [1984] (Definition 5.3.1) or Hindley and Longo [1980]. We simply call these λ-models. We introduce also *applicative structures* (Definition 5.1.1 of Barendregt [1984]) and *quasi λ-models*.

18.1.2. Definition. (i) An *applicative structure* is a pair $\langle D, \cdot \rangle$, where D is a set and $\cdot : D \times D \to D$ is a binary operation on D.

(ii) A *quasi λ-model* is of the form

$$D = \langle D, \cdot, [\![\ ]\!]^D \rangle,$$

where $\langle D, \cdot \rangle$ is an applicative structure and $[\![\ ]\!]^D : \Lambda \times \mathsf{Env}_D \to D$ satisfies the following.

(1) $[\![x]\!]^D_\rho = \rho(x)$;

(2) $[\![MN]\!]^D_\rho = [\![M]\!]^D_\rho \cdot [\![N]\!]^D_\rho$;

(3) $[\![\lambda x.M]\!]^D_\rho = [\![\lambda y.M[x := y]]\!]^D_\rho$, provided $y \notin \mathrm{FV}(M)$; (α)

(4) $\forall d \in D.[\![M]\!]^D_{\rho[x:=d]} = [\![N]\!]^D_{\rho[x:=d]} \Rightarrow [\![\lambda x.M]\!]^D_\rho = [\![\lambda x.N]\!]^D_\rho$; (ξ)

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

(iii) A λ-*model* is a quasi λ-model which safisfies

(6) $[\![\lambda x.M]\!]^D_\rho \cdot d = [\![M]\!]^D_{\rho[x:=d]}$ (β)

(iv) A $\lambda\mathsf{I}$-*model* is a quasi λ-model which safisfies

(6′) $x \in \mathrm{FV}(M) \Rightarrow [\![\lambda x.M]\!]^D_\rho \cdot d = [\![M]\!]^D_{\rho[x:=d]}$ $(\beta\mathsf{I})$

We will write simply $[\![\ ]\!]_\rho$ instead of $[\![\ ]\!]^D_\rho$ when there is no danger of confusion.

We have the following implications.

$$D\ \lambda\text{-model} \implies D\ \lambda\mathsf{I}\text{-model} \implies D \text{ quasi } \lambda\text{-model}.$$

The first class of applicative structures satisfies (β) in general, the second only for $\lambda\mathsf{I}$-redexes and the third class does not need to satisfy (β) at all, but there is an interpretation for λ-terms.

18.1.3. Definition. Let $D = \langle D, \cdot, [\![\ ]\!]\rangle$ be a quasi λ-model.

(i) The statement $M = N$, for M, N untyped lambda terms, is *true in* D, notation $D \models M = N$ iff

$$\forall \rho \in \mathsf{Env}_D.[\![M]\!]_\rho = [\![N]\!]_\rho.$$

(ii) As usual one defines $D \models \chi$, where χ is any statement built up using first order predicate logic from equations between untyped lambda terms.

(iii) A $\lambda(\mathsf{I})$-model D is called *extensional* iff

$$D \models (\forall x.Mx = Nx) \Rightarrow M = N.$$

(iv) A $\lambda(\mathsf{I})$-model D is called an $\boldsymbol{\eta}$-*model* iff

$$D \models \lambda x.Mx = M, \text{ for } x \notin \mathrm{FV}(M). \qquad (\boldsymbol{\eta})$$

We will see now how the notions of $\lambda(\mathsf{I})$-model and (strict) lambda structure are related, see Definition 17.2.3 and Definition 17.4.5.

18.1.4. Definition. (i) Let $\langle D, F, G\rangle$ be a lambda structure. We define the triple $\langle D, \cdot_F, [\![\ ]\!]^{F,G}\rangle$ as follows, with the intention to construct a quasi λ-model.

- First we obtain an applicative structure by setting for $d, e \in D$

$$d \cdot_F e = F(d)(e).$$

- Then the map $[\![\ ]\!]^{F,G} : \Lambda \times \mathsf{Env}_D \to D$ is defined as follows.

$$\begin{aligned} [\![x]\!]^{F,G}_\rho &= \rho(x); \\ [\![MN]\!]^{F,G}_\rho &= F([\![M]\!]^{F,G}_\rho)([\![N]\!]^{F,G}_\rho); \\ [\![\lambda x.M]\!]^{F,G}_\rho &= G(\lambda\!\!\lambda d \in D.[\![M]\!]^{F,G}_{\rho[x:=d]}), \end{aligned}$$

noticing that the map $\lambda\!\!\lambda d \in D.[\![M]\!]_{\rho[x:=d]}$ used for $[\![\lambda x.M]\!]_\rho$ is continuous.

(ii) Let $\langle D, F, G\rangle$ be a *strict* lambda structure $\langle D, F, G\rangle$. We define the triple $\langle D, \cdot_F, [\![\ ]\!]^{F,G}\rangle$ as above, changing the clause for $[\![\lambda x.M]\!]^{F,G}_\rho$ into

$$[\![\lambda x.M]\!]^{F,G}_\rho = G(\lambda\!\!\lambda d \in D.\text{ if } d = \bot_D \text{ then } \bot_D \text{ else } [\![M]\!]^{F,G}_{\rho[x:=d]}).$$

18.1.5. Proposition. *Let $\langle D, F, G\rangle$ be a (strict) lambda structure. Then $\langle D, \cdot_F, [\![\ ]\!]^{F,G}\rangle$ is a quasi λ-model.*

Proof. Easy. ■

18.1.6. Definition. Let $\langle D, F, G\rangle$ be a (strict) lambda structure. Then,

$$D = \langle D, \cdot_F, [\![\ ]\!]^{F,G}\rangle$$

is called *the quasi λ-model induced by $\langle D, F, G\rangle$*. We will sometimes omit the subscript from $\cdot_F$ when there is no danger of confusion.

The only requirement that a (strict) lambda structure misses to be a $\lambda(\mathsf{I})$-model is the axiom $(\beta(\mathsf{I}))$.

18.1.7. Proposition. (i) *Let $D = \langle D, \cdot_F, [\![\ ]\!]^{F,G}\rangle$ be the quasi λ-model induced by the lambda structure $\langle D, F, G\rangle$. Then the following statements are equivalent.*

(1) *$D \models (\lambda x.M)N = M[x := N]$, for all $M, N \in \Lambda$;*

(2) *$[\![\lambda x.M]\!]^{F,G}_\rho \cdot d = [\![M]\!]^{F,G}_{\rho(x:=d)}$, for all $M \in \Lambda$ and $d \in D$;*

(3) *D is a λ-model;*

(4) *$D \models \{M = N \mid \lambda\beta \vdash M = N\}$.*

(ii) *Let $D = \langle D, \cdot_F, [\![\ ]\!]^{F,G}\rangle$ be the quasi λ-model induced by the strict lambda structure $\langle D, F, G\rangle$. Then the following statements are equivalent.*

(1) *$D \models (\lambda x.M)N = M[x := N]$, for all $M, N \in \Lambda$ such that $x \in \mathrm{FV}(M)$;*

(2) *$[\![\lambda x.M]\!]^{F,G}_\rho \cdot d = [\![M]\!]^{F,G}_{\rho(x:=d)}$, for all $M \in \Lambda$ with $x \in \mathrm{FV}(M)$, and $d \in D$;*

(3) *D is a $\lambda\mathsf{I}$-model;*

(4) *$D \models \{M = N \mid \lambda\beta\mathsf{I} \vdash M = N\}$.*

PROOF. (i) (1)⇒(2). By (1) one has $[\![(\lambda x.M)N]\!]^{F,G}_\rho = [\![M[x := N]]\!]^{F,G}_\rho$. Taking $N \equiv x$ and $\rho' = \rho(x := d)$ one obtains

$$[\![(\lambda x.M)x]\!]^{F,G}_{\rho'} = [\![M]\!]^{F,G}_{\rho'},$$

hence

$$[\![\lambda x.M]\!]^{F,G}_\rho \cdot d = [\![M]\!]^{F,G}_{\rho'},$$

as $\rho \restriction \mathrm{FV}(\lambda x.M) = \rho' \restriction \mathrm{FV}(\lambda x.M)$.

(2)⇒(3). By (ii), Definition 18.1.4 and Proposition 18.1.5 all conditions for being a λ-model are fulfilled, see Definition 18.1.2.

(3)⇒(4). By Theorem 5.3.4 in Barendregt [1984].

(4)⇒(1). Trivial.

(ii) Similarly. ■

18.1.8. COROLLARY. *Let D be the $\lambda(\mathsf{I})$-model induced by the (strict) lambda structure $\langle D, F, G\rangle$. Then*

$$D \text{ is a } \lambda(\mathsf{I})\boldsymbol{\eta}\text{-model} \iff D \text{ is an extensional } \lambda(\mathsf{I})\text{-model.}$$

PROOF. (⇒) Suppose that for some ρ one has for all $d \in D$

$$[\![Mx]\!]^{F,G}_{\rho[x:=d]} = [\![Nx]\!]^{F,G}_{\rho[x:=d]}.$$

Then by $(\boldsymbol{\eta})$ and Proposition 18.1.5(ii) one has

$$[\![M]\!]^{F,G}_\rho = [\![\lambda x.Mx]\!]^{F,G}_\rho = [\![\lambda x.Nx]\!]^{F,G}_\rho = [\![N]\!]^{F,G}_\rho.$$

(⇐) Note that by $(\beta(\mathsf{I}))$ one has $D \models (\lambda x.Mx)y = My$, where x is fresh. Hence by extensionality one has $D \models \lambda x.Mx = M$. ■

Isomorphisms of λ-models

This section relates isomorphisms between lambda structures and lambda models.

18.1.9. DEFINITION. We say that D and D' are isomorphic λ-models via m if for all λ-terms M and environments ρ:

$$\mathsf{m}([\![M]\!]^D_\rho) = [\![M]\!]^{D'}_{\mathsf{m}\circ\rho}$$

18.1.10. LEMMA. *If two λ-models D and D' are isomorphic, then they equate the same terms, i.e. $D \models M = N \iff D' \models M = N$.*

PROOF. Easy.

Next lemma is used to prove that an isomorphism of lambda structures is also an isomorphism of λ-models. For the converse of this lemma, see Exercise 17.5.18.

18.1.11. LEMMA. **[ch-iso]** *Let* $\underline{\mathsf{m}} = \langle \mathsf{m}, \mathsf{m}^R \rangle : \langle D, F, G \rangle \to \langle D', F', G' \rangle$ *be an isomorphisms between lambda structures. Then* $\mathsf{m} : D \to D'$ *is a bijective continuous map such that*

$$\begin{array}{lrcl} (1) & \mathsf{m}(G(f)) & = & G'(\mathsf{m} \circ f \circ \mathsf{m}^{-1}); \\ (2) & \mathsf{m}(F(d)(e)) & = & F'(\mathsf{m}(d))(\mathsf{m}(e)). \end{array}$$

If we write $f^{\mathsf{m}} = \mathsf{m} \circ f \circ \mathsf{m}^{-1}$ *then we can reformulate these conditions as*

$$\begin{array}{rcl} \mathsf{m}(G(f)) & = & G'(f^{\mathsf{m}}); \\ \mathsf{m}(d \cdot_F e) & = & \mathsf{m}(d) \cdot_{F'} \mathsf{m}(e). \end{array}$$

PROOF. By Definition 17.2.5 we get:

$$\begin{array}{ll} \text{(lambda-gc1)} & \forall f \in [D \to D].\mathsf{m}(G(f)) \sqsubseteq G'(\mathsf{m} \circ f \circ \mathsf{m}^R); \\ \text{(lambda-gc2)} & \forall x' \in D', x \in D.F(m^R(x'))(x) \sqsubseteq m^R(F'(x')(m(x))). \end{array}$$

As to (1). Since $\underline{\mathsf{m}}$ is an isomorphism, we have that besides the lambda Galois connection

$$\langle \mathsf{m}, \mathsf{m}^R \rangle : D \to D',$$

there is another lambda Galois connection $\underline{\mathsf{m}}^{-1}$, which we call $\underline{\mathsf{n}} = \langle \mathsf{n}, \mathsf{n}^R \rangle : D' \to D$ such that

$$\begin{array}{rcl} \underline{\mathsf{n}} \circ \underline{\mathsf{m}} & = & \langle \mathsf{Id}_D, \mathsf{Id}_D \rangle, \\ \underline{\mathsf{m}} \circ \underline{\mathsf{n}} & = & \langle \mathsf{Id}_{D'}, \mathsf{Id}_{D'} \rangle. \end{array}$$

Using composition between Galois connections, this amounts to saying

$$\begin{array}{rcl} \mathsf{n} \circ \mathsf{m} & = & \mathsf{Id}_D, \\ \mathsf{m}^R \circ \mathsf{n}^R & = & \mathsf{Id}_D, \\ \mathsf{m} \circ \mathsf{n} & = & \mathsf{Id}_{D'}, \\ \mathsf{n}^R \circ \mathsf{m}^R & = & \mathsf{Id}_{D'}. \end{array}$$

We call (†) the set of the four equations above, that we will use in the following.

Looking at (†), we see that compositions of m and n give the identities. So $\mathsf{n} = \mathsf{m}^{-1}$. This implies that n is a right adjoint of m. But the right adjoint is unique, so $\mathsf{m}^{-1} = \mathsf{n} = \mathsf{m}^R$. For the same reason $\mathsf{n}^R = \mathsf{m}$. Therefore we have proved that $(\underline{\mathsf{m}}^{-1} = \underline{\mathsf{n}} =) \langle \mathsf{n}, \mathsf{n}^R \rangle = \langle \mathsf{m}^R, \mathsf{m} \rangle$. Note that, as $\underline{\mathsf{n}}$ is a lambda Galois connection, we have that

$$(a) \quad \begin{cases} \mathsf{m}^R \text{ is (also) the left adjoint of } \mathsf{m}, \text{ and} \\ \mathsf{m} \text{ is (also) the right adjoint of } \mathsf{m}^R. \end{cases}$$

We are now in the position to prove (1). The inequality

$$\mathsf{m}(G(f)) \sqsubseteq G'(\mathsf{m} \circ f \circ \mathsf{m}^R)$$

is (lambda-gc1). As to the other inequality, first of all note that, exploiting (a), we have that conditions (lambda-gc1) and (lambda-gc2) induce, for any $f' : D' \to D'$, and $x \in D, x' \in D'$,

$$\begin{array}{lrcl} (b) & \mathsf{m}^R(G'(f')) & \sqsubseteq & G(\mathsf{m}^R \circ f' \circ \mathsf{m}), \\ (c) & F'(\mathsf{m}(x))(x') & \sqsubseteq & \mathsf{m}(F(x)(\mathsf{m}^R(x'))). \end{array}$$

So we have

$$\begin{array}{rcll} G'(\mathsf{m}\circ f\circ \mathsf{m}^R) & = & \mathsf{m}\circ \mathsf{m}^R\circ G'(\mathsf{m}\circ f\circ \mathsf{m}^R), & \text{by } (\dagger),\\ & \sqsubseteq & \mathsf{m}\circ G(\mathsf{m}^R\circ \mathsf{m}\circ f\circ \mathsf{m}^R\circ \mathsf{m}), & f'=\mathsf{m}\circ f\circ \mathsf{m}^R \text{ in } (b),\\ & = & \mathsf{m}\circ G(f), & \text{since } \mathsf{m}^R=\mathsf{m}^{-1}. \end{array}$$

Therefore we have proved $\mathsf{m}(G(f)) = G'(\mathsf{m}\circ f\circ \mathsf{m}^{-1})$.

As to (2). Notice that $\mathsf{m}(F(d))(e) \sqsubseteq F'(\mathsf{m}(d))(\mathsf{m}(e))$, since

$$\begin{array}{rcll} \mathsf{m}(F(d))(e) & = & \mathsf{m}(F(\mathsf{m}^R(\mathsf{m}(d)))(e)), & \text{by } (\dagger),\\ & \sqsubseteq & \mathsf{m}(\mathsf{m}^R(F'(\mathsf{m}(d))(\mathsf{m}(e)))), & \text{by (lambda-gc2)},\\ & = & F'(\mathsf{m}(d))(\mathsf{m}(e)), & \text{by } (\dagger). \end{array}$$

On the other hand, we have also $F'(\mathsf{m}(d))(\mathsf{m}(e)) \sqsubseteq \mathsf{m}(F(d)(e))$. In fact

$$\begin{array}{rcll} F'(\mathsf{m}(d))(\mathsf{m}(e)) & \sqsubseteq & \mathsf{m}(F(d)(\mathsf{m}^R(\mathsf{m}(e)))), & \text{by } (c),\\ & = & \mathsf{m}(F(d)(e)), & \text{by } (\dagger). \end{array}$$

The following proposition will be used in Corollary 18.3.31 to prove that the models D_∞ and $\mathcal{F}^{\mathrm{Scott}}$ equate the same terms.

18.1.12. PROPOSITION. *Let D and D' be isomorphic as lambda structures. Then,*

(i) *D and D' are isomorphic as λ-models.*
(ii) *They equate the same terms, i.e. $D \models M=N \iff D' \models M=N$.*

PROOF. (i) By induction on M using Lemma 18.1.11.
(ii) Using (i) and Lemma 18.1.10. ■

18.2. Filter models

In this section, we define the notion of filter model and prove that the interpretation of a term is the set of its types (Type-semantics Theorem). Using this theorem and the results in Chapter **??**, we study which conditions have to be imposed on a type theory to induce a filter model. At the end of this section we also study representability of continuous functions.

18.2.1. DEFINITION. Let $\mathcal{T}\in \mathrm{TT}$. The *filter quasi λ-model* of $\mathcal{T}$ is a quasi λ-model, denoted by $\mathcal{F}^{\mathcal{T}}$, where $[\![\ ]\!]^{\mathcal{F}^{\mathcal{T}}}: \Lambda\times \mathsf{Env}_{\mathcal{F}^{\mathcal{T}}}\to \mathcal{F}^{\mathcal{T}}$ is defined by

$$\begin{array}{rcl} [\![x]\!]^{\mathcal{F}^{\mathcal{T}}}_\rho & = & \rho(x);\\ [\![MN]\!]^{\mathcal{F}^{\mathcal{T}}}_\rho & = & \uparrow\{B\in \mathbb{T}^{\mathcal{T}} \mid \exists A\in [\![N]\!]^{\mathcal{F}^{\mathcal{T}}}_\rho.(A\to B)\in [\![M]\!]^{\mathcal{F}^{\mathcal{T}}}_\rho\};\\ [\![\lambda x.M]\!]^{\mathcal{F}^{\mathcal{T}}}_\rho & = & \uparrow\{A\to B \mid B\in [\![M]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho[x:=\uparrow A]}\}. \end{array}$$

The notion of filter structure $\langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle$ given in Definition 15.4.5 contains two operations $F^{\mathcal{T}}$ and $G^{\mathcal{T}}$ that can be used to interpret application and abstraction. These operations coincide with the way application and abstraction are interpreted in Definition 18.2.1. This leads to the following lemma:

18.2.2. LEMMA. *Let $\mathcal{T} \in \mathrm{TT}$. The filter structure $\langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle$ induces a quasi λ-model which coincides with the notion of filter quasi λ-model given in Definition 18.2.1.*

PROOF. Note first that the filter structure $\langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle$ is a lambda structure by Definition 17.2.3 and the Remark just after 15.4.5. In case $\mathcal{T} \in \mathrm{TT}^{-\mathtt{U}}$, we have that $\langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle$ is also a *strict* lambda structure, i.e.

$$F^{\mathcal{T}} \in [\mathcal{F}^{\mathcal{T}} \to_{\mathrm{s}} [\mathcal{F}^{\mathcal{T}} \to_{\mathrm{s}} \mathcal{F}^{\mathcal{T}}]] \text{ and } G^{\mathcal{T}} \in [[\mathcal{F}^{\mathcal{T}} \to_{\mathrm{s}} \mathcal{F}^{\mathcal{T}}] \to_{\mathrm{s}} \mathcal{F}^{\mathcal{T}}],$$

see Definition 17.4.5. Hence $\langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle$ induces a quasi λ-model, by Proposition 18.1.5. It is easy to see that $[\![\ ]\!]^{F^{\mathcal{T}}, G^{\mathcal{T}}} = [\![\ ]\!]^{\mathcal{F}^{\mathcal{T}}}$.

We now define two classes of filter models: filter λ-models and filter $\lambda\mathsf{I}$-models.

18.2.3. DEFINITION. (i) Let $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$. We say that $\mathcal{F}^{\mathcal{T}}$ is *a filter λ-model* if the filter quasi λ-model $\mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, \cdot, [\![\ ]\!]^{\mathcal{F}^{\mathcal{T}}} \rangle$ is a λ-model.

(ii) Let $\mathcal{T} \in \mathrm{TT}^{-\mathtt{U}}$. We say that $\mathcal{F}^{\mathcal{T}}$ is *a filter $\lambda\mathsf{I}$-model* if the filter quasi λ-model $\mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, \cdot, [\![\ ]\!]^{\mathcal{F}^{\mathcal{T}}} \rangle$ is a $\lambda\mathsf{I}$-model.

18.2.4. PROPOSITION. (i) *Let $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$. Then $\mathcal{F}^{\mathcal{T}}$ is a filter λ-model iff for all $M, N \in \Lambda$*

$$[\![(\lambda x.M)N]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho} = [\![M[x := N]]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho}.$$

(ii) *Let $\mathcal{T} \in \mathrm{TT}^{-\mathtt{U}}$. Then $\mathcal{F}^{\mathcal{T}}$ is a filter $\lambda\mathsf{I}$-model iff for all $M, N \in \Lambda$*

$$x \in \mathrm{FV}(M) \ \Rightarrow\ [\![(\lambda x.M)N]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho} = [\![M[x := N]]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho}.$$

PROOF. Both equivalences follow from Proposition 18.1.7. ■

The following *type-semantics theorem* is important. It has as consequence that for a closed untyped lambda term M and a $\mathcal{T} \in \mathrm{TT}$ one has

$$[\![M]\!]^{\mathcal{F}^{\mathcal{T}}} = \{A \mid \vdash^{\mathcal{T}}_{\cap} M : A\},$$

i.e. the semantical meaning of M is the collection of its types.

18.2.5. DEFINITION. Let Γ be a context and $\rho \in \mathsf{Env}_{\mathcal{F}^{\mathcal{T}}}$. Then Γ *agrees with* ρ, notation $\Gamma \models \rho$, if

$$(x : A) \in \Gamma \ \Rightarrow\ A \in \rho(x).$$

18.2.6. PROPOSITION. (i) $\Gamma \models \rho \ \&\ \Gamma' \models \rho \ \Rightarrow\ \Gamma \uplus \Gamma' \models \rho$.

(ii) $\Gamma \models \rho[x :=\uparrow A] \ \Rightarrow\ \Gamma \backslash x \models \rho$.

PROOF. Immediate. ■

18.2.7. THEOREM (Type-semantics Theorem). *Let* $\mathcal{T} \in \mathrm{TT}$ *and* $\langle \mathcal{F}^{\mathcal{T}}, \cdot, [\![\ ]\!]^{\mathcal{F}^{\mathcal{T}}} \rangle$ *its corresponding filter quasi* λ*-model. Then, for any* $M \in \Lambda$ *and* $\rho \in \mathsf{Env}_{\mathcal{F}^{\mathcal{T}}}$,

$$[\![M]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho} = \{A \mid \Gamma \vdash^{\mathcal{T}}_{\cap} M : A \textit{ for some } \Gamma \models \rho\}.$$

PROOF. We have two cases:

(i) Let $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$. By induction on the structure of M.
Case $M \equiv x$. Then

$$\begin{array}{rcl} [\![M]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho} & = & \rho(x) \\ & = & \{A \mid A \in \rho(x)\} \\ & = & \{A \mid A \in \rho(x) x : A \vdash^{\mathcal{T}}_{\cap} x : A\} \\ & = & \{A \mid \Gamma \vdash^{\mathcal{T}}_{\cap} x : A \text{ for some } \Gamma \models \rho\}, \end{array}$$

by Definition 18.2.5 and the Inversion Theorem 16.1.1(i).

Case $M \equiv NL$. Then

$$\begin{array}{rcl} [\![M]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho} & = & [\![N]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho} \cdot [\![L]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho} \\ & = & \uparrow\{A \mid \exists B \in [\![L]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho} . (B \to A) \in [\![N]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho}\} \\ & = & \{A \mid \exists k{>}0 \exists B_1, \ldots, B_k, C_1, \ldots, C_k . \\ & & [(B_i \to C_i) \in [\![N]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho} \; B_i \in [\![L]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho} \; (\textstyle\bigcap_{1 \le i \le k} C_i) \le A]\} \cup \uparrow\{\mathtt{U}\}, \\ & & \text{by definition of } \uparrow, \\ & = & \{A \mid \exists k{>}0 \exists B_1, \ldots, B_k, C_1, \ldots, C_k, \Gamma_1, \ldots, \Gamma_k, \Delta_1, \ldots, \Delta_k \\ & & [\Gamma_i, \Delta_i \models \rho \;\&\; \Gamma_i \vdash^{\mathcal{T}}_{\cap} N : (B_i \to C_i) \\ & & \Delta_i \vdash^{\mathcal{T}}_{\cap} L : B_i C_1 \cap \ldots \cap C_k \le A]\} \cup \uparrow\{\mathtt{U}\}, \\ & & \text{by the induction hypothesis}, \\ & = & \{A \mid \Gamma \vdash^{\mathcal{T}}_{\cap} NL : A \text{ for some } \Gamma \models \rho\}, \\ & & \text{taking } \Gamma = \Gamma_1 \uplus \ldots \uplus \Gamma_k \uplus \ldots \uplus \Delta_1 \uplus \ldots \uplus \Delta_k, \\ & & \text{by Theorem 16.1.1(ii) and Proposition 18.2.6(i)}. \end{array}$$

Case $M \equiv \lambda x.N$. Then

$$\begin{aligned}
[\![\lambda x.N]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho} &= G^{\mathcal{T}}(\lambda\!\!\lambda X \in \mathcal{F}^{\mathcal{T}}.[\![N]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho[x:=X]}\\
&= \uparrow\{(B{\to}C) \mid C \in [\![N]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho[x:=\uparrow B]}\}\\
&= \{A \mid \exists k{>}0 \exists B_1,\ldots,B_k,C_1,\ldots,C_k.\Gamma_1,\ldots,\Gamma_k[\Gamma_i \models \rho[x\ =\uparrow B_i]\\
&\quad \&\ \Gamma_i, x\ B_i \vdash^{\mathcal{T}}_{\cap} N : C_i(B_1{\to}C_1)\cap\ldots\cap(B_k{\to}C_k) \leq A]\},\\
&\quad \text{by the induction hypothesis,}\\
&= \{A \mid \Gamma \vdash^{\mathcal{T}}_{\cap} \lambda x.N : A \text{ for some } \Gamma \models \rho\},\\
&\quad \text{taking } \Gamma = (\Gamma_1 \uplus \ldots \uplus \Gamma_k)\backslash x, \text{ by Theorem 16.1.1(iii), rule } (\leq)\\
&\quad \text{and Proposition 18.2.6(ii).}
\end{aligned}$$

(ii) Let $\mathcal{T} \in \mathrm{TT}^{-\mathtt{U}}$. Similarly. Note that in the case $M = NL$ we drop '$\cup\uparrow\{\mathtt{U}\}$' both times. Delete now if we change def. of quasi l-model: In case $M = \lambda x.N$, using Definition 18.1.4, it follows that

$[\![\lambda x.N]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho} = \uparrow\{(B{\to}C) \mid C \in [\![N]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho[x=\uparrow B]}\}$ holds, because $\uparrow B \neq \emptyset$. ■

18.2.8. Corollary. (i) *Let $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$. Then*

$$\mathcal{F}^{\mathcal{T}} \textit{ is a filter } \lambda\textit{-model} \iff [\Gamma \vdash^{\mathcal{T}}_{\cap} (\lambda x.M) : (B{\to}A) \ \Rightarrow\ \Gamma, x\ B \vdash^{\mathcal{T}}_{\cap} M : A].$$

(ii) *Let $\mathcal{T} \in \mathrm{TT}^{-\mathtt{U}}$. Then*

$$\begin{aligned}
&\mathcal{F}^{\mathcal{T}} \textit{ is a filter } \lambda\mathsf{I}\textit{-model} \iff\\
&[\Gamma \vdash^{\mathcal{T}}_{\cap} (\lambda x.M) : (B{\to}A)\ \&\ x \in \mathrm{FV}(M) \ \Rightarrow\ \Gamma, x\ B \vdash^{\mathcal{T}}_{\cap} M : A].
\end{aligned}$$

Proof. (i) By Propositions 18.2.4(i), 16.2.1(i) and Corollary 16.2.5(i).
(ii) By Propositions 18.2.4(ii), 16.2.1(ii) and Corollary 16.2.5(ii). ■

18.2.9. Corollary. (i) *Let $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$. Then*

$$\mathcal{T} \textit{ is } \beta\textit{-sound } \Rightarrow\ \mathcal{F}^{\mathcal{T}} \textit{ is a filter } \lambda\textit{-model.}$$

(ii) *Let $\mathcal{T} \in \mathrm{TT}^{-\mathtt{U}}$. Then*

$$\mathcal{T} \textit{ is } \beta\textit{-sound } \Rightarrow\ \mathcal{F}^{\mathcal{T}} \textit{ is a filter } \lambda\mathsf{I}\textit{-model.}$$

Proof. By the Corollary above and Theorem 16.1.9(iii). ■

18.2.10. Corollary. (i) *Let $\mathcal{T} \in \{$*Scott, Park, CDZ, HR, DHM, BCD, AO, Plotkin, Engeler, CDS*$\}$. Then*

$$\mathcal{F}^{\mathcal{T}} \textit{ is a filter } \lambda\textit{-model.}$$

(ii) *Let* $\mathcal{T} \in \{\mathrm{HL}, \mathrm{CDV}, \mathrm{CD}\}$. *Then*

$$\mathcal{F}^{\mathcal{T}} \text{ is a filter } \lambda\mathsf{I}\text{-model.}$$

PROOF. (i) By (i) of the previous Corollary and Theorem 16.1.7.
(ii) By (ii) of the Corollary, using Theorem 16.1.7. ■

18.2.11. PROPOSITION. (i) *Let* $\mathcal{T} \in \mathrm{TT}^{\mathsf{U}}$. *Then*

$\mathcal{T}$ *is natural and* β- *and* $\boldsymbol{\eta}^{\mathsf{U}}$*-sound* $\Rightarrow$ $\mathcal{F}^{\mathcal{T}}$ *is an extensional filter* λ*-model.*

(ii) *Let* $\mathcal{T} \in \mathrm{TT}^{-\mathsf{U}}$. *Then*

$\mathcal{T}$ *is proper and* β- *and* $\boldsymbol{\eta}$*-sound* $\Rightarrow$ $\mathcal{F}^{\mathcal{T}}$ *is an extensional filter* $\lambda\mathsf{I}$*-model.*

PROOF. (i) and (ii). $\mathcal{F}^{\mathcal{T}}$ is a $\lambda(\mathsf{I})$-model by Corollary 18.2.9(i)((ii)). As to extensionality it suffices by Corollary 18.1.8 to verify for $x \notin \mathrm{FV}(M)$ that

$$[\![\lambda x.Mx]\!]_\rho = [\![M]\!]_\rho. \qquad (\boldsymbol{\eta})$$

This follows from Theorems 18.2.7, and 16.2.15. ■

18.2.12. COROLLARY. (i) *Let* $\mathcal{T} \in \{\mathrm{Scott}, \mathrm{Park}, \mathrm{CDZ}, \mathrm{HR}, \mathrm{DHM}\}$. *Then*

$$\mathcal{F}^{\mathcal{T}} \text{ is an extensional filter } \lambda\text{-model.}$$

(ii) *Let* $\mathcal{T} = \mathrm{HL}$. *Then*

$$\mathcal{F}^{\mathcal{T}} \text{ is an extensional filter } \lambda\mathsf{I}\text{-model.}$$

PROOF. (i) and (ii) follow from Corollary 16.2.13. ■

As shown in Meyer [1982], see also Barendregt [1984] Ch.4, a lambda structure is a λ-model provided that it contains the combinators K, S and ε, satisfying certain properties. Thus, a condition for being a filter λ-model can be obtained by simply forcing the existence of such combinators. This yields a characterization of the natural type theories which induce λ-models. See Alessi [1991] for the rather technical proof.

18.2.13. THEOREM. *Let* $\mathcal{T} \in \mathrm{NTT}^{\mathsf{U}}$.

(i) *The filter structure* $\mathcal{F}^{\mathcal{T}}$ *is a filter* λ*-model if and only if the following three conditions are fulfilled in* $\mathcal{T}$.

(K) *For all* C, E *one has*

$$\begin{aligned} C \leq E \iff\ & \forall D\, \exists k \geq 1, A_1, \ldots, A_k, B_1, \ldots, B_k. \\ & (A_1 \to B_1 \to A_1) \cap \ldots \cap (A_k \to B_k \to A_k) \leq C \to D \to E. \end{aligned}$$

(S) *For all* D, E, F, G *one has*

$$\exists H.[E \leq F \to H \ \&\ D \leq F \to H \to G] \iff$$

$$\left[\begin{array}{l} \exists k \geq 1, A_1, \ldots, A_k, B_1, \ldots, B_k, C_1, \ldots, C_k. \\ \quad (A_1 \to B_1 \to C_1) \to (A_1 \to B_1) \to (A_1 \to C_1) \cap \\ \quad \ldots \\ \quad (A_k \to B_k \to C_k) \to (A_k \to B_k) \to (A_k \to C_k) \quad \leq D \to E \to F \to G \end{array}\right].$$

(ε) *For all C, D one has*

$$\left[\begin{array}{l}\exists k \geq 1, A_1, \ldots, A_k, B_1, \ldots, B_k. \\ (A_1 \to B_1) \to (A_1 \to B_1) \cap \ldots \cap (A_k \to B_k) \to (A_k \to B_k) \leq (C \to D)\end{array}\right] \Leftrightarrow$$
$$\exists m \geq 1, E_1, \ldots, E_m, F_1, \ldots, F_m. C \leq (E_1 \to F_1) \cap \ldots \cap (E_m \to F_m) \leq D.$$

(ii) *The structure $\mathcal{F}^{\mathcal{T}}$ is an extensional filter λ-model iff the third condition above is replaced by the following two.*

(ε_1) $\forall A\, \exists k \geq 1, A_1, \ldots, A_k, B_1, \ldots, B_k. A =_{\mathcal{T}} (A_1 \to B_1) \cap \ldots \cap (A_k \to B_k)$;

(ε_2) $\forall A, B\, \exists k \geq 1, A_1, \ldots, A_k. [(A_1 \to A_1) \cap \ldots \cap (A_k \to A_k) \leq (A \to B)$
$\Leftrightarrow A \leq B]$ [Correct?]. ■

Representability of continuous functions

In this subsection following Alessi, Barbanera and Dezani-Ciancaglini [2004] we will isolate a number of conditions on a $\mathcal{T} \in \mathrm{NTT}^{\mathtt{U}}$ to characterize properties of the set of *representable functions* in $\langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle$, i.e. the set of functions in the image of $F^{\mathcal{T}}$.

18.2.14. Definition. A function $f : D \to D$ is called *representable* in the lambda structure $\langle D, F, G \rangle$ if $f = F(d)$ for some $d \in D$.

Note that since $F : D \to [D \to D]$, all representable functions are continuous.

18.2.15. Lemma. *Let $\mathcal{T} \in \mathrm{NTT}^{\mathtt{U}}$ and let $f \in [\mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}]$. Then*

$$f \text{ is representable in } \langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle \iff F^{\mathcal{T}} \circ G^{\mathcal{T}}(f) = f.$$

Proof. ($\Leftarrow$) Trivial. ($\Rightarrow$) Suppose $f = F^{\mathcal{T}}(X)$. Claim $F^{\mathcal{T}}(G^{\mathcal{T}}(F^{\mathcal{T}}(X))) = F^{\mathcal{T}}(X)$. One has $G^{\mathcal{T}}(F^{\mathcal{T}}(X)) = \uparrow\{A \to B \mid A \to B \in X\}$. Hence

$$A \to B \in G^{\mathcal{T}}(F^{\mathcal{T}}(X)) \iff A \to B \in X.$$

So $\forall Y. F^{\mathcal{T}}(G^{\mathcal{T}}(F^{\mathcal{T}}(X)))(Y) = F^{\mathcal{T}}(X)(Y)$, hence $F^{\mathcal{T}}(G^{\mathcal{T}}(f)) = f$. ■

18.2.16. Lemma. *Let $\mathcal{T} \in \mathrm{NTT}^{\mathtt{U}}$. Let $A, B \in \mathbb{T}^{\mathcal{T}}$. Then*

$$G^{\mathcal{T}}(\uparrow A \Rightarrow \uparrow B) = \uparrow(A \to B).$$

Proof.

$$\begin{aligned} G^{\mathcal{T}}(\uparrow A \Rightarrow \uparrow B) &= \uparrow\{(C \to D) \mid D \in (\uparrow A \Rightarrow \uparrow B)(\uparrow C)\} \\ &= \uparrow(A \to B). \end{aligned}$$

In the last step the inclusion $\supseteq$ is trivial: $(A \to B)$ is one of the $C \to D$. Now suppose $C \to D$ is such that $D \in (\uparrow A \Rightarrow \uparrow B)(\uparrow C)$. Then there are two cases. Case $\uparrow A \subseteq \uparrow C$. This means $C \leq A$, so $(\uparrow A \Rightarrow \uparrow B)(\uparrow C) = \uparrow B$, so $D \geq B$. Hence in this case $A \to B \leq C \to D$ by rule ($\to$). Case $\uparrow A \not\subseteq \uparrow C$. Then $D = \mathtt{U}$, hence $C \to D = \mathtt{U}$, by rules ($\to$), ($\mathtt{U} \to$), and again $A \to B \leq C \to D$. Therefore also $\subseteq$ holds in the last equation. ■

18.2.17. LEMMA. *Let $\mathcal{T} \in \mathrm{NTT}^{\mathtt{U}}$ and define the function $h : \mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}$ by*

$$h = (\uparrow A_1 \Rightarrow \uparrow B_1) \sqcup \ldots \sqcup (\uparrow A_n \Rightarrow \uparrow B_n).$$

Then, for all $C \in \mathbb{T}^{\mathcal{T}}$ we have that

(i) $h(\uparrow C) = \{D \mid \exists k \geq 1 \exists i_1 \ldots i_k. [B_{i_1} \cap \ldots \cap B_{i_k} \leq DC \leq A_{i_1} \cap \ldots \cap A_{i_k}]\} \cup \uparrow\{\mathtt{U}\}.$

(ii) $(F^{\mathcal{T}} \circ G^{\mathcal{T}})(h)(\uparrow C) = \{D \mid (A_1 \to B_1) \cap \ldots \cap (A_n \to B_n) \leq (C \to D)\}.$

PROOF. (i)
$$\begin{aligned} h(\uparrow C) &= \bigsqcup\{\uparrow B_i \mid \uparrow A_i \subseteq \uparrow C 1 \leq i \leq n\} \\ &= \uparrow B_{i_1} \sqcup \ldots \sqcup \uparrow B_{i_k} \\ &\quad \text{for } \{i_1, \ldots i_k\} = \{i \mid C \leq A_i \& 1 \leq i \leq n\} \\ &= \uparrow(B_{i_1} \cap \ldots \cap B_{i_k}) \cup \uparrow\mathtt{U}, \text{ by 15.4.4(iii)}, \\ &= \{D \mid \exists k \geq 1 \exists i_1 \ldots i_k. \\ &\quad B_{i_1} \cap \ldots \cap B_{i_k} \leq DC \leq A_{i_1} \cap \ldots \cap A_{i_k}\} \cup \uparrow\{\mathtt{U}\}. \end{aligned}$$

(ii) $(F^{\mathcal{T}} \circ G^{\mathcal{T}})(h)(\uparrow C) =$

$$\begin{aligned} &= F^{\mathcal{T}}(G^{\mathcal{T}}(\uparrow A_1 \Rightarrow \uparrow B_1) \sqcup \ldots \sqcup G^{\mathcal{T}}(\uparrow A_n \Rightarrow \uparrow B_n))(\uparrow C), \text{ by 17.2.2(ii)}, \\ &= F^{\mathcal{T}}(\uparrow(A_1 \to B_1) \sqcup \ldots \sqcup \uparrow(A_n \to B_n))(\uparrow C), \text{ by 18.2.16}, \\ &= F^{\mathcal{T}}(\uparrow((A_1 \to B_1) \cap \ldots \cap (A_n \to B_n)))(\uparrow C), \text{ by 15.4.4(iii)}, \\ &= \{D \mid \exists E \in \uparrow C.(E \to D) \in \uparrow((A_1 \to B_1) \cap \ldots \cap (A_n \to B_n))\}, \\ &\quad \text{see Definition 15.4.5(ii)}, \\ &= \{D \mid \exists E \geq C.(A_1 \to B_1) \cap \ldots \cap (A_n \to B_n) \leq (E \to D)\} \\ &= \{D \mid (A_1 \to B_1) \cap \ldots \cap (A_n \to B_n) \leq (C \to D)\}, \text{ by } (\to). \blacksquare \end{aligned}$$

NOTATION. We define the function $\mathsf{K} \in \mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}$ as

$$\mathsf{K} = \lambda\!\!\lambda X \in \mathcal{F}^{\mathcal{T}}.\lambda\!\!\lambda Y \in \mathcal{F}^{\mathcal{T}}.X.$$

For each $X \in \mathcal{F}^{\mathcal{T}}$, $\mathsf{K}(X)$ is a constant function in $[\mathcal{F}^{\mathcal{T}} \to \mathcal{F}^{\mathcal{T}}]$.

18.2.18. THEOREM. *Let $\mathcal{T} \in \mathrm{NTT}^{\mathtt{U}}$. Let $\mathcal{R}$ be the set of representable functions in $\langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle$. Then we have the following.*

(i) *$\mathcal{R}$ contains the bottom function $\mathsf{K}(\bot_{\mathcal{F}^{\mathcal{T}}})$ iff for all C, D*

$$\mathtt{U} \leq C \to D \;\Rightarrow\; \mathtt{U} \leq D.$$

(ii) *$\mathcal{R}$ contains all constant functions iff for all B, C, D*

$$\mathtt{U} \to B \leq C \to D \;\Rightarrow\; B \leq D.$$

(iii) *$\mathcal{R}$ contains all continuous step functions iff for all A, B, C, D*

$$A \to B \leq C \to D \;\&\; D \neq \mathtt{U} \;\Rightarrow\; C \leq A \;\&\; B \leq D.$$

(iv) *$\mathcal{R}$ contains (i.e. is the set of) all continuous functions iff $\mathcal{T}$ is β-sound.*

PROOF. (i) Assume that $\mathsf{K}(\bot_{\mathcal{F}^{\mathcal{T}}}) \in \mathcal{R}$. Then $F^{\mathcal{T}}(G^{\mathcal{T}}(\mathsf{K}(\bot_{\mathcal{F}^{\mathcal{T}}}))) = \mathsf{K}(\bot_{\mathcal{F}^{\mathcal{T}}})$, by Lemma 18.2.15. Observe that

$$\begin{array}{rcll} G^{\mathcal{T}}(\mathsf{K}(\bot_{\mathcal{F}^{\mathcal{T}}})) & = & \uparrow\{A{\to}\mathtt{U} \mid A \in \mathbb{T}\}, & \text{by Definition 15.4.5(i),} \\ & = & \uparrow\mathtt{U}, & \text{since } \mathcal{T} \text{ is natural.} \end{array}$$

Hence $F^{\mathcal{T}}(\bot_{\mathcal{F}^{\mathcal{T}}}) = \mathsf{K}(\bot_{\mathcal{F}^{\mathcal{T}}})$, so in particular $F^{\mathcal{T}}(\bot_{\mathcal{F}^{\mathcal{T}}})(\uparrow C) = \bot_{\mathcal{F}^{\mathcal{T}}}$, and hence

$$\begin{array}{rcll} \{D \mid \mathtt{U} \leq D\} & = & \uparrow\mathtt{U} = \ \bot_{\mathcal{F}^{\mathcal{T}}} & \\ & = & F^{\mathcal{T}}(\bot_{\mathcal{F}^{\mathcal{T}}})(\uparrow C) & \\ & = & F^{\mathcal{T}}(\uparrow\mathtt{U})(\uparrow C) & \\ & = & \{D \mid \mathtt{U} \leq (C{\to}D)\}, & \text{by Definition 15.4.5(i).} \end{array}$$

But then $\mathtt{U} \leq C{\to}D \ \Rightarrow \ \mathtt{U} \leq D$.

($\Leftarrow$) Suppose $\mathtt{U} \leq C{\to}D \ \Rightarrow \ \mathtt{U} \leq D$. We show that $F^{\mathcal{T}}(\bot_{\mathcal{F}^{\mathcal{T}}}) = \mathsf{K}(\bot_{\mathcal{F}^{\mathcal{T}}})$. Indeed,

$$\begin{array}{rcll} F^{\mathcal{T}}(\bot_{\mathcal{F}^{\mathcal{T}}})(X) & = & \{B \mid \exists A \in X.(A{\to}B) \in \bot_{\mathcal{F}^{\mathcal{T}}}\} & \\ & = & \{B \mid \exists A \in X.(A{\to}B) \in \uparrow\mathtt{U}\} & \\ & = & \{B \mid \exists A \in X.\mathtt{U} \leq (A{\to}B)\} & \\ & = & \{B \mid \mathtt{U} \leq B\}, & \text{by the assumption,} \\ & = & \uparrow\mathtt{U} = \bot_{\mathcal{F}^{\mathcal{T}}}. & \end{array}$$

(ii) Suppose that $\mathtt{U}{\to}B \leq C{\to}D \ \Rightarrow \ B \leq D$. We first show that each compact constant function $\mathsf{K}(\uparrow B)$ is represented by $\uparrow(\mathtt{U}{\to}B)$. Indeed,

$$\begin{array}{rcll} D \in \uparrow(\mathtt{U}{\to}B) \cdot \uparrow C & \iff & C{\to}D \in \uparrow(\mathtt{U}{\to}B), & \text{by } (\to), \\ & \iff & \mathtt{U}{\to}B \leq C{\to}D & \\ & \iff & B \leq D, & \text{using the assumption,} \\ & \iff & D \in \uparrow B = \mathsf{K}(\uparrow B)\,(\uparrow C). & \end{array}$$

Now we show that an arbitrary constant function $\mathsf{K}(X)$ is representable. Then $X = \bigcup\{\uparrow B \mid B \in X\}$, where $\{\uparrow B \mid B \in X\}$ is directed. Notice that $\mathsf{K}(X) = \bigsqcup_{B \in X} \mathsf{K}(\uparrow B)$. Therefore by the representability of $\mathsf{K}(\uparrow B)$ just proved, Lemma 18.2.15 and the continuity of $F^{\mathcal{T}} \circ G^{\mathcal{T}}$ we get

$$\begin{array}{rcl} \mathsf{K}(X) & = & \bigsqcup\limits_{B \in X} \mathsf{K}(\uparrow B) \\ & = & \bigsqcup\limits_{B \in X} (F^{\mathcal{T}} \circ G^{\mathcal{T}})(\mathsf{K}(\uparrow B)) \\ & = & (F^{\mathcal{T}} \circ G^{\mathcal{T}})(\bigsqcup\limits_{B \in X} \mathsf{K}(\uparrow B)) \\ & = & (F^{\mathcal{T}} \circ G^{\mathcal{T}})(\mathsf{K}(X)), \end{array}$$

hence again by Lemma 18.2.15 the constant map $\mathsf{K}(X)$ is representable.

Conversely, suppose that all constant functions are representable. Then $F^{\mathcal{T}} \circ G^{\mathcal{T}}(\mathsf{K}(\uparrow B)) = \mathsf{K}(\uparrow B)$, by Lemma 18.2.15. Therefore

$$\begin{array}{lll} \mathtt{U}{\to}B \leq C{\to}D & \Rightarrow & (C{\to}D) \in \uparrow(\mathtt{U}{\to}B) \\ & \Rightarrow & D \in \uparrow(\mathtt{U}{\to}B) \cdot \uparrow C \\ & \Rightarrow & D \in ((F^{\mathcal{T}} \circ G^{\mathcal{T}})(\mathsf{K}(\uparrow B)))(\uparrow C), \\ & & \text{since } \uparrow(\mathtt{U}{\to}B) \subseteq \mathsf{K}(\uparrow B) \text{ by } (\to) \text{ and } (\mathtt{U}), \\ & \Rightarrow & D \in (\mathsf{K}(\uparrow B))(\uparrow C) = \uparrow B \\ & \Rightarrow & B \leq D. \end{array}$$

(iii) ($\Rightarrow$) Suppose all continuous step functions are representable. Suppose $A{\to}B \leq C{\to}D,\ D \neq \mathtt{U}$. Take $h = \uparrow A \Rightarrow \uparrow B$. By Lemma 18.2.17(ii) we have

$$\begin{array}{rcl} (F^{\mathcal{T}} \circ G^{\mathcal{T}})(h)(\uparrow C) & = & \{E \mid A{\to}B \leq C{\to}E\} \\ h(\uparrow C) & = & \{E \mid B \leq EC \leq A\} \cup \uparrow\mathtt{U}. \end{array}$$

By the first assumption these two sets are equal. By the second assumption it follows that $C \leq A \ \&\ B \leq D$.

($\Leftarrow$) Let $h = X \Rightarrow Y$ be continuous. We have to show that

$$(F^{\mathcal{T}} \circ G^{\mathcal{T}})(h) = h \tag{2}$$

By Lemma 17.1.5(ii) it suffices to show this for compact h. If $Y \neq \bot_{\mathcal{F}^{\mathcal{T}}}$, then by 17.1.9 both X, Y are compact, so $h = \uparrow A{\to}\uparrow B$. Then (2) holds by Lemma 18.2.17 and the assumption. If $Y = \bot_{\mathcal{F}^{\mathcal{T}}}$, then h is the bottom function and hence representable (the assumption in (iii) implies the assumption in (i)).

(iv) Let $\mathcal{T} \in \mathrm{NTT}$. Let $h \in \mathcal{K}([\mathcal{F}^{\mathcal{T}} {\to} \mathcal{F}^{\mathcal{T}}])$. By Proposition 17.1.11(ii) it follows that for some $A_1, \ldots, A_n, B_1, \ldots, B_n \in \mathcal{T}$

$$h = (\uparrow A_1 \Rightarrow \uparrow B_1) \sqcup \ldots \sqcup (\uparrow A_n \Rightarrow \uparrow B_n),$$

as a finite element of $\mathcal{F}^{\mathcal{T}}$ is of the form $\uparrow A$.

($\Rightarrow$) Suppose all continuous functions are representable. Then since h above is continuous and by Lemma 18.2.15, one has

$$(F^{\mathcal{T}} \circ G^{\mathcal{T}})(h)(\uparrow C) = h(\uparrow C).$$

It follows from Lemma 18.2.17(i),(ii) that $\mathcal{T}$ is β-sound.

($\Leftarrow$) Suppose $\mathcal{T}$ is β-sound. Again by Lemma 18.2.17 for a compact continuous function h one has that $(F^{\mathcal{T}} \circ G^{\mathcal{T}})(h)$ and h coincide on the compact elements $\uparrow C$. Therefore by Proposition 17.1.5 they coincide everywhere. But then it follows again that $f = (F^{\mathcal{T}} \circ G^{\mathcal{T}})(f)$ for every continuous $f : \mathcal{F}^{\mathcal{T}} {\to} \mathcal{F}^{\mathcal{T}}$. Hence Lemma 18.2.15 applies. ■

To induce a filter model β-soundness is not necessary

The intersection type theories $\mathcal{T} \in \{\mathrm{Scott}, \mathrm{Park}, \mathrm{BCD}, \mathrm{CDZ}, \mathrm{HR}, \mathrm{AO}, \mathrm{DHM}\}$ all induce filter λ-models, by Corollary 18.2.10(i). These type theories are all natural and β-sound. Therefore by Theorem 18.2.18(iv) all continuous functions

are representable in these $\mathcal{F}^{\mathcal{T}}$. In Sections 18.3 and 18.4 we will give many more filter λ-models arising from domain models. It is therefore interesting to ask whether there exist filter λ-models where *not all* continuous functions are representable. We answer affirmatively, and end this section by giving an example of a natural type theory ABD that is not β-sound and nevertheless, it induces a filter λ-model $\mathcal{F}^{\mathrm{ABD}}$. Therefore by the same theorem not all continuous functions are representable in $\mathcal{F}^{\mathrm{ABD}}$. The model builds on an idea in Coppo et al. [1984]. In exercise 18.5.4 another such model, due to Alessi [1993], is constructed. See also Alessi, Barbanera and Dezani-Ciancaglini [2004].

The theory ABD

18.2.19. Definition. Let $\mathbb{A}^{\mathrm{ABD}} = \{\mathtt{U}, \diamondsuit, \heartsuit\}$. We define ABD as the smallest natural type theory [1] that contains the axiom ($\diamondsuit$) where

$$(\diamondsuit)\ A \leq_{\mathrm{ABD}} A[\diamondsuit := \heartsuit].$$

18.2.20. Lemma. (i) $A \leq_{\mathrm{ABD}} B \ \Rightarrow\ A[\diamondsuit := \heartsuit] \leq_{\mathrm{ABD}} B[\diamondsuit := \heartsuit]$.

(ii) $\Gamma \vdash^{\mathrm{ABD}} M : A \ \Rightarrow\ \Gamma[\diamondsuit := \heartsuit] \vdash^{\mathrm{ABD}} M : A[\diamondsuit := \heartsuit]$.

(iii) $\Gamma, \Gamma' \vdash^{\mathrm{ABD}} M : A \ \Rightarrow\ \Gamma, \Gamma'[\diamondsuit := \heartsuit] \vdash^{\mathrm{ABD}} M : A[\diamondsuit := \heartsuit]$.

(iv) $\Gamma, x\ A_i \vdash^{\mathrm{ABD}} M : B_i$ *for* $1 \leq i \leq n$ &
$(A_1 \to B_1) \cap \ldots \cap (A_n \to B_n) \leq_{\mathrm{ABD}} C \to D \ \Rightarrow\ \Gamma, x\ C \vdash^{\mathrm{ABD}} M : D$.

Proof. (i) By induction on the definition of $\leq_{\mathrm{ABD}}$.

(ii) By induction on derivations using (i) for rule ($\leq_{\mathrm{ABD}}$).

(iii) From (ii) and rule ($\leq_{\mathrm{ABD}}$-L), taking into account that if $(x\ B) \in \Gamma$, then $(x\ B[\diamondsuit := \heartsuit]) \in \Gamma[\diamondsuit := \heartsuit]$ and $B \leq_{\mathrm{ABD}} B[\diamondsuit := \heartsuit]$.

(iv) Let $\alpha_1, \ldots, \alpha_n, \alpha'_1, \ldots, \alpha'_{n'} \in \mathbb{A}^{\mathrm{ABD}}$. We show by induction on the definition of $\leq_{\mathrm{ABD}}$ that if the following statements hold

$$\begin{aligned}
&A \leq_{\mathrm{ABD}} A';\\
&A = \alpha_1 \cap \ldots \cap \alpha_n \cap (B_1 \to C_1) \cap \ldots \cap (B_k \to C_k);\\
&A' = \alpha'_1 \cap \ldots \cap \alpha'_{n'} \cap (B'_1 \to C'_1) \cap \ldots \cap (B'_{k'} \to C'_{k'});\\
&\Gamma, x\ B_i \vdash^{\mathrm{ABD}} M : C_i \quad \forall i \in \{1, \ldots, k\}.
\end{aligned}$$

Then

$$\Gamma, x\ B'_j \vdash^{\mathrm{ABD}} M : C'_j \quad \forall j \in \{1, \ldots, k'\}.$$

The only interesting case is when the applied rule is ($\diamondsuit$), i.e. we have

$$\begin{aligned}
&A \leq_{\mathrm{ABD}} A[\diamondsuit := \heartsuit];\\
&A = \alpha_1 \cap \ldots \cap \alpha_n \cap (B_1 \to C_1) \cap \ldots \cap (B_k \to C_k);\\
&A' = A[\diamondsuit := \heartsuit].
\end{aligned}$$

By hypothesis $\Gamma, x\ B_i \vdash^{\mathrm{ABD}} M : C_i$ for all $i \in \{1, \ldots, k\}$, so we are done by (iii). ■

[1] ABD contains the axioms and rules of Definitions 15.1.3 and 15.1.18.

18.2.21. THEOREM. (i) ABD *is a* TT *that is not* β*-sound.*
(ii) *Nevertheless* $\mathcal{F}^{\mathrm{ABD}}$ *is a filter* λ*-model.*

PROOF. (i) By definition ABD is a TT. We have $\diamondsuit\to\diamondsuit \leq_{\mathrm{ABD}} \heartsuit\to\heartsuit$, but $\heartsuit \not\leq_{\mathrm{ABD}} \diamondsuit$, so it is not β-sound.

(ii) To show that $\mathcal{F}^{\mathrm{ABD}}$ is a λ-model, it suffices, by Proposition 18.2.8, to verify that $\Gamma \vdash^{\mathrm{ABD}} \lambda x.M : A\to B \;\Rightarrow\; \Gamma, x\ A \vdash^{\mathrm{ABD}} M : B$. Suppose that $\Gamma \vdash^{\mathrm{ABD}} \lambda x.M : A\to B$. By Lemma 16.1.1(iii), there are $C_1,\ldots,C_n, D_1,\ldots,D_n$ such that

$$(C_1\to D_1)\cap\ldots\cap(C_n\to D_n) \leq_{\mathrm{ABD}} A\to B$$
$$\forall i\in\{1,\ldots,n\}\Gamma, x\ C_i \vdash^{\mathrm{ABD}} M : D_i.$$

So, we are done by Lemma 18.2.20(iv). ■

For example the step function $\uparrow\diamondsuit \Rightarrow \uparrow\diamondsuit$ is not representable in $\mathcal{F}^{\mathrm{ABD}}$.

18.3. D_∞ models as filter models

This section shows the connection between filter models and D_∞ models, see Scott [1972] or Barendregt [1984]. We will work in the category **ALG** of ω-algebraic complete lattices and Scott-continuous maps. In other categories such as the one of Scott domains or stable sets, filter models do not capture the D_∞-models in their full generality.

To begin with, Proposition 17.2.14 states that forall $D\in\mathbf{NLS}$

$$D \;\cong\; \mathcal{F}^{\mathcal{K}(D)},$$

(isomorphism within the category **NLS**). For a given lambda model D_∞ obtained from a $D_0\in\mathbf{ALG}$ we have that $D_\infty\in\mathbf{NLS}$. The construction of D_∞ does not only depend on the initial D_0, but also on the projection pair $\mathsf{i}_0,\mathsf{j}_0$ which gives the start of the D_∞ construction:

$$\mathsf{i}_0 : D_0\to D_1,\ \mathsf{j}_0 : D_1\to D_0,$$

where $D_1 = [D_0\to D_0]$.

Given the triple $t = (D_0,\mathsf{i}_0,\mathsf{j}_0)$, we write $D_\infty = D^t_\infty$ to emphasize the dependency on t. For each t, we associate a type theory CDHL(t) whose type atoms are the compact elements of D_0 and the subtyping relation contains both the order of D_0 and the function i_0. We will prove that

$$\mathcal{K}(D^t_\infty) \cong [\mathrm{CDHL}(t)],$$

where $=$ is the equality of the type theory CDHL(t). Then, using Proposition 17.2.14 and Lemma 15.4.7, it follows that

$$D^t_\infty \cong \mathcal{F}^{\mathcal{K}(D^t_\infty)} \cong \mathcal{F}^{[\mathrm{CDHL}(t)]} \cong \mathcal{F}^{\mathrm{CDHL}(t)}.$$

We conclude that an arbitrary D^t_∞ model can be described as the filter model $\mathcal{F}^{\mathrm{CDHL}(t)}$. The converse is true for the extensional filter λ-models of Fig. **??**.

Given $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{DHM}, \text{HR}\}$, we will associate a triple $\mathsf{init}(\mathcal{T}) = (D_0, \mathsf{i}_0, \mathsf{j}_0)$ such that $\text{CDHL}(\mathsf{init}(\mathcal{T})) = \mathcal{T}$. Then,

$$D_\infty^{\mathsf{init}(\mathcal{T})} \cong \mathcal{F}^{\text{CDHL}(\mathsf{init}(\mathcal{T}))} = \mathcal{F}^{\mathcal{T}}.$$

We will write $D_\infty^{\mathcal{T}} := D_\infty^{\mathsf{init}(\mathcal{T})}$. Then the equation becomes

$$D_\infty^{\mathcal{T}} \cong \mathcal{F}^{\text{CDHL}(\mathsf{init}(\mathcal{T}))} = \mathcal{F}^{\mathcal{T}}.$$

One obtains the following five versions of D_∞,

$$D_\infty^{\text{Scott}}, D_\infty^{\text{Park}}, D_\infty^{\text{CDZ}}, D_\infty^{\text{DHM}} \text{ and } D_\infty^{\text{HR}}.$$

The pleasant fact is that $\mathcal{T}$ and the triple $t = (D_0, \mathsf{i}_0, \mathsf{j}_0)$ correspond to each other in a canonical way. For each of the theories $\mathcal{T} \in \{\text{Scott}, \text{Park}\}$ the model $D_\infty^{\mathcal{T}}$ was constructed first and the natural type theory $\mathcal{T}$ came later. For $\mathcal{T} \in \{\text{CDZ}, \text{DHM}, \text{HR}\}$ one constructed this natural type theory in order to obtain the model $D_\infty^{\mathcal{T}}$ satisfying certain property.

D_∞ models

This subsection recalls some basic concepts of the standard D_∞ construction and fixes some notations, see Scott [1972], Barendregt [1984], Gierz et al. [1980].

18.3.1. Definition. (i) Let D_0 be an ω-algebraic complete lattice and

$$\langle \mathsf{i}_0, \mathsf{j}_0 \rangle$$

be an *embedding-projection* pair between D_0 and $[D_0 \to D_0]$, i.e.

$$\begin{array}{rcl} \mathsf{i}_0 & : & D_0 \to [D_0 \to D_0] \\ \mathsf{j}_0 & : & [D_0 \to D_0] \to D_0 \end{array}$$

are Scott continuous maps satisfying

$$\begin{array}{rcl} \mathsf{i}_0 \circ \mathsf{j}_0 & \sqsubseteq & \mathsf{Id}_{[D_0 \to D_0]} \\ \mathsf{j}_0 \circ \mathsf{i}_0 & = & \mathsf{Id}_{D_0}. \end{array}$$

(ii) Define a *tower* $\langle \mathsf{i}_n, \mathsf{j}_n \rangle : D_n \to D_{n+1}$ in the following way:

- $D_{n+1} = [D_n \to D_n]$;
- $\mathsf{i}_n(f) = \mathsf{i}_{n-1} \circ f \circ \mathsf{j}_{n-1}$ for any $f \in D_n$;
- $\mathsf{j}_n(g) = \mathsf{j}_{n-1} \circ g \circ \mathsf{i}_{n-1}$ for any $g \in D_{n+1}$.

(iii) For $d \in \Pi_{n \in \mathbb{N}} D_n$ write $d_n = d(n)$. The set D_∞ is defined by

$$D_\infty \;=\; \{d \in \Pi_{n \in \mathbb{N}} D_n \mid \forall n \in \mathbb{N}.\ d_n \in D_n \ \&\ \mathsf{j}_n(d_{n+1}) = d_n\},$$

Notation. d_n denotes the projection on D_n, while d^n is an element of D_n.

18.3.2. Definition. (i) The ordering on D_∞ is given by

$$d \sqsubseteq e \;\Leftrightarrow\; \forall k \in \mathbb{N}.\, d_k \sqsubseteq e_k.$$

(ii) Let $\langle \Phi_{m\infty}, \Phi_{\infty m} \rangle$ denote the standard embedding-projection pair between D_m and D_∞ defined as follows. For $d^m \in D_m$, $d \in D_\infty$ write

$$\Phi_{mn}(d^m) = \begin{cases} \mathsf{j}_n(\ldots(\mathsf{j}_{m-1}(d^m))), & \text{if } m > n; \\ d^m & \text{if } m = n; \\ \mathsf{i}_{n-1}(\ldots(\mathsf{i}_m(d^m))), & \text{if } m < n; \end{cases}$$

and take

$$\begin{aligned} \Phi_{m\infty}(d^m) &= \langle \Phi_{m0}(d^m), \Phi_{m1}(d^m), \ldots, \Phi_{mn}(d^m), \ldots \rangle; \\ \Phi_{\infty m}(d) &= d_m = d(m). \end{aligned}$$

18.3.3. Lemma. *$\bigsqcup X$ exists for all $X \subseteq D_\infty$.*

Proof. Let $\begin{aligned} d^n &= \bigsqcup\{x_n | x \in X\}; \\ e^n &= \bigsqcup\{\Phi_{mn}(d^m) | m \in \mathbb{N}\}. \end{aligned}$

The set $\{\Phi_{mn}(d^m) | m \in \mathbb{N}\}$ is directed by monotonicity of $\mathsf{i}_m, \mathsf{j}_m$, and the fact that $\mathsf{i}_m \circ \mathsf{j}_m \sqsubseteq \mathsf{Id}_{D_{m+1}}$ and $\mathsf{j}_m \circ \mathsf{i}_m = \mathsf{Id}_{D_m}$. Define

$$\bigsqcup X = \lambda n \in \mathbb{N}.e^n.$$

Then, by continuity of j_n, we have that $\bigsqcup X \in D_\infty$, if $X \neq \emptyset$. If $X = \emptyset$, then the continuity cannot be applied, but $\bigsqcup \emptyset = \lambda\!\!\lambda n \in \mathbb{N}.\bot_{D_n}$ (use $i_n(\bot_{D_n}) = \bot_{D_{n+1}}$ so that $j_{n+1}(\bot_{D_{n+1}}) = \bot_{D_n}$). ■

18.3.4. Definition. Let

$$\begin{aligned} F_\infty &: D_\infty \to [D_\infty \to D_\infty] \\ G_\infty &: [D_\infty \to D_\infty] \to D_\infty \end{aligned}$$

be defined as follows

$$\begin{aligned} F_\infty(d) &= \bigsqcup_{n \in \mathbb{N}} (\Phi_{n\infty} \circ d_{n+1} \circ \Phi_{\infty n}); \\ G_\infty(f) &= \bigsqcup_{n \in \mathbb{N}} \Phi_{(n+1)\infty}(\Phi_{\infty n} \circ f \circ \Phi_{n\infty}). \blacksquare \end{aligned}$$

18.3.5. Lemma. (i) $\mathsf{i}_n \circ \mathsf{j}_n \sqsubseteq \mathsf{Id}_{[D_n \to D_n]}$, $\mathsf{j}_n \circ \mathsf{i}_n = \mathsf{Id}_{D_n}$.

(ii) $\forall p, q \in D_n\ [\mathsf{i}_{n+1}(p \mapsto q) = (\mathsf{i}_n(p) \mapsto \mathsf{i}_n(q)) \;\&$
$\mathsf{j}_{n+1}(\mathsf{i}_n(p) \mapsto \mathsf{i}_n(q)) = (p \mapsto q)]$.

(iii) $\Phi_{m\infty} \circ \Phi_{\infty m} \sqsubseteq \mathsf{Id}_\infty$ *and* $\Phi_{\infty m} \circ \Phi_{m\infty} = \mathsf{Id}_{D_m}$.

(iv) $\forall e \in \mathcal{K}(D_n)\ [\mathsf{i}_n(e) \in \mathcal{K}(D_{n+1})]$.

(v) $\forall e \in \mathcal{K}(D_n)\ [m \geq n \Rightarrow \Phi_{nm}(e) \in \mathcal{K}(D_m)]$.

(vi) $\forall e \in \mathcal{K}(D_n)\ [\Phi_{n\infty}(e) \in \mathcal{K}(D_\infty)]$.

(vii) *If $n \leq k \leq m$ and $d \in D_n$, $e \in D_k$, then*

$$\Phi_{nk}(d) \sqsubseteq e \iff \Phi_{nm}(d) \sqsubseteq \Phi_{km}(e) \iff \Phi_{n\infty}(d) \sqsubseteq \Phi_{k\infty}(e).$$

(viii) $\Phi_{mn} = \Phi_{\infty n} \circ \Phi_{m\infty}$.

(ix) $\forall a, b \in D_n\ [(\Phi_{n\infty}(a) \mapsto \Phi_{n\infty}(b)) = \Phi_{n\infty} \circ (a \mapsto b) \circ \Phi_{\infty n}]$.

PROOF. (i) and (ii): By induction on n.

(iii) follows from (i).

(iv) and (v) and (vi): By Lemma 17.2.2(ii), observing that the following pairs are all Galois connections:

1. $\langle i_n, j_n \rangle$;
2. $\langle \Phi_{nm}, \Phi_{mn} \rangle$ for $n \leq m$;
3. $\langle \Phi_{n\infty}, \Phi_{\infty n} \rangle$.

(ix) follows from 17.2.2(iii). ■

18.3.6. LEMMA. $\bigsqcup_{n \in \mathbb{N}} \Phi_{n\infty} \circ \Phi_{\infty n} = \mathsf{Id}_{D_\infty}$.

PROOF. Since $\langle \Phi_{n\infty}, \Phi_{\infty n} \rangle$ is an embedding-projection pair, we have for all $n \in \mathbb{N}$ $\Phi_{n\infty} \circ \Phi_{\infty n} \sqsubseteq \mathsf{Id}_{D_\infty}$, hence for all $d \in D_\infty$

$$\bigsqcup_{n \in \mathbb{N}} \Phi_{n\infty} \circ \Phi_{\infty n}(d) \sqsubseteq d.$$

On the other hand, for all $k \in \mathbb{N}$, we have

$$\begin{array}{lll} (\bigsqcup_{n \in N} \Phi_{n\infty} \circ \Phi_{\infty n}(d))_k & \sqsupseteq (\Phi_{k\infty} \circ \Phi_{\infty k}(d))_k, & \text{because } (-)_k \text{ is monotone,} \\ & = \Phi_{\infty k}(d), & \text{as } \forall x \in D_k.(\Phi_{k\infty}(x))_k = x, \\ & = d_k. & \end{array}$$

Therefore also

$$\bigsqcup_{n \in \mathbb{N}} \Phi_{n\infty} \circ \Phi_{\infty n}(d) \sqsupseteq d,$$

and we are done. ■

Next lemma characterizes the compact elements of D_∞ and $[D_\infty \to D_\infty]$.

18.3.7. LEMMA. (i) $d \in \mathcal{K}(D_\infty) \iff \exists k, e \in \mathcal{K}(D_k).\Phi_{k\infty}(e) = d$.

(ii) $f \in \mathcal{K}([D_\infty \to D_\infty]) \iff \exists k, g \in \mathcal{K}(D_{k+1}).f = \Phi_{k\infty} \circ g \circ \Phi_{\infty k}$.

(iii) *If* $f = \Phi_{k\infty} \circ g \circ \Phi_{\infty k}$ *with* $g \in D_{k+1}$, *then* $G_\infty(f) = \Phi_{(k+1)\infty}(g)$.

PROOF. (i) ($\Rightarrow$) Let $d \in \mathcal{K}(D_\infty)$. Then $d = \bigsqcup_{n \in \mathbb{N}} \Phi_{n\infty}(d_n)$, by Lemma 18.3.6. Since d is compact, there exists $k \in \mathbb{N}$ such that $d = \Phi_{k\infty}(d_k)$. Now we prove that $d_k \in \mathcal{K}(D_k)$. Let $X \subseteq D_k$ be directed. Then

$$\begin{array}{lll} d_k \sqsubseteq \bigsqcup X & \Rightarrow & d \sqsubseteq \Phi_{k\infty}(\bigsqcup X) \\ & \Rightarrow & d \sqsubseteq \bigsqcup \Phi_{k\infty}(X), \qquad \text{since } \Phi_{k\infty} \text{ is continuous,} \\ & \Rightarrow & \exists x \in X.d \sqsubseteq \Phi_{k\infty}(x), \qquad \text{for some } k \text{ since } d \text{ is compact,} \\ & \Rightarrow & \Phi_{\infty k}(d) \sqsubseteq \Phi_{\infty k} \circ \Phi_{k\infty}(x) \\ & \Rightarrow & d_k \sqsubseteq x. \end{array}$$

This proves that $d_k \in \mathcal{K}(D_k)$. ($\Leftarrow$) By Lemma 18.3.5(vi).

(ii) ($\Rightarrow$) By Lemma 18.3.6, we have

$$f = \bigsqcup_{n\in\mathbb{N}} \Phi_{n\infty} \circ (\Phi_{\infty n} \circ f \circ \Phi_{n\infty}) \circ \Phi_{\infty n}.$$

Using similar arguments as in the proof of (i), we have that

- $\exists k \in \mathbb{N}. f = \Phi_{k\infty} \circ (\Phi_{\infty k} \circ f \circ \Phi_{k\infty}) \circ \Phi_{\infty k}$,
- $(\Phi_{\infty k} \circ f \circ \Phi_{k\infty}) \in \mathcal{K}(D_{k+1})$.

Put $g = (\Phi_{\infty k} \circ f \circ \Phi_{k\infty})$. ($\Leftarrow$) Easy.

(iii) Let $f = \Phi_{k\infty} \circ g \circ \Phi_{\infty k}$ with $g \in D_{k+1}$. Then

$$\begin{aligned}
G_\infty(f) &= G_\infty(\Phi_{k\infty} \circ g \circ \Phi_{\infty k}) \\
&= \bigsqcup_{n\in\mathbb{N}} \Phi_{(n+1)\infty}(\Phi_{\infty n} \circ (\Phi_{k\infty} \circ g \circ \Phi_{\infty k}) \circ \Phi_{n\infty}) \\
&= \bigsqcup_{n\in\mathbb{N}} \Phi_{(n+1)\infty}(\Phi_{kn} \circ g \circ \Phi_{nk}), && \text{by Definition of } \Phi_{kn}, \\
&= \bigsqcup_{n\in\mathbb{N}} \Phi_{(n+1)\infty}(\Phi_{(k+1)(n+1)}(g)), && \text{by Lemma 18.3.6}, \\
&= \Phi_{(k+1)\infty}(g). \blacksquare
\end{aligned}$$

18.3.8. Lemma. (i) $\forall x \in D_\infty. x = \bigsqcup\{e \in \mathcal{K}(D_\infty) \mid e \sqsubseteq x\}$.

(ii) $\mathcal{K}(D_\infty)$ *is countable.*

(iii) $D_\infty \in \mathbf{ALG}$.

Proof. (i) Let $x \in D_\infty$ and $U_x = \{e \in \mathcal{K}(D_\infty) \mid e \sqsubseteq x\}$. Clearly, $\bigsqcup U_x \sqsubseteq x$. Now let $f = \bigsqcup U_x$ in order to show $x \sqsubseteq f$. By definition of sup in D_∞ we have

$$f_n = \bigsqcup V(n,x), \quad \text{where } V(n,x) = \{e_n \in D_n \mid e \in \mathcal{K}(D_\infty) \,\&\, e \sqsubseteq x\}.$$

Since D_n is algebraic, we have that

$$x_n = \bigsqcup W(n,x), \quad \text{where } W(n,x) = \{d \in \mathcal{K}(D_n) \mid d \sqsubseteq x_n\}.$$

We will prove that $W(n,x) \subseteq V(n,x)$. Suppose $d \in W(n,x)$. Then $d \in \mathcal{K}(D_n)$ and $d \sqsubseteq x_n$. Let $e = \Phi_{n\infty}(d)$. Then,

(1) $d = \Phi_{\infty n} \circ \Phi_{n\infty}(d) = e_n$.

(2) $e = \Phi_{n\infty}(d) \in \mathcal{K}(D_\infty)$, by Lemma 18.3.5(vi).

(3) $e = \Phi_{n\infty}(d) \sqsubseteq \Phi_{n\infty}(x_n) \sqsubseteq x$, by monotonicity of $\Phi_{n\infty}$ and Lemma 18.3.5(iii).

Hence $d \in V(n,x)$. Now indeed $x \sqsubseteq f$, as clearly $x_n \sqsubseteq f_n$.

(ii) By Proposition 17.1.11 one has $D_n \in \mathbf{ALG}$ for each n. Hence $\mathcal{K}(D_n)$ is countable for each n. But then also $\mathcal{K}(D_\infty)$ is countable, by Lemma 18.3.7(i).

(iii) By (i), and (ii). $\blacksquare$

18.3.9. Theorem. (Scott [1972]) *Let D_∞ be constructed from $D_0 \in \mathbf{ALG}$ and a projection pair $\mathsf{i}_0, \mathsf{j}_0$. Then $D_\infty \in \mathbf{ALG}$ and D_∞ with F_∞, G_∞ is reflexive. Moreover,*

$$F_\infty \circ G_\infty = \mathsf{Id}_{[D_\infty \to D_\infty]} \;\&\; G_\infty \circ F_\infty = \mathsf{Id}_{D_\infty}.$$

It follows that D_∞ is an extensional λ-model.

PROOF. See Barendregt [1984] Theorem 18.2.16 for the proof that F and G are each other's inverse. By Proposition 18.1.8 it follows that D_∞ is an extensional λ-model. ■

18.3.10. COROLLARY. *Let D_∞ be constructed from $D_0 \in \mathbf{ALG}$ and a projection pair $\mathsf{i}_0, \mathsf{j}_0$. Then $\langle D_\infty, F_\infty, G_\infty\rangle$ is in* **NLS**.

PROOF. Immediate from the Theorem. ■

D_∞ as a filter λ-model

In this subsection we follow Alessi [1991], Alessi, Dezani-Ciancaglini and Honsell [2004]. Let D_∞ be constructed from the triple $t = (D_0, \mathsf{i}_0, \mathsf{j}_0)$. To emphasize the dependency on t we write $D_\infty = D^t_\infty$. From the previous corollary and Proposition 17.2.14 it follows that $D^t_\infty \cong \mathcal{F}^{\mathcal{K}(D^t_\infty)}$. In this subsection we associate with $t = (D_0, \mathsf{i}_0, \mathsf{j}_0)$ a family of intersection type theories CDHL(t). These type theories are compatible and they can be considered as type structures, denoted also by CDHL(t). We will show that

$$\mathcal{K}(D^t_\infty) \cong \mathrm{CDHL}(t).$$

Hence

$$D^t_\infty \cong \mathcal{F}^{\mathrm{CDHL}(t)}.$$

The name of the family of type theories CDHL(t) is due to Coppo et al. [1984] where this construction was first discussed. Other relevant references are Coppo et al. [1987], which presents the filter λ-model induced by the type theory CDZ, Honsell and Ronchi Della Rocca [1992], where the filter λ-models induced by the type theories Park, HR and other models are considered, and Alessi [1991], Di Gianantonio and Honsell [1993], Plotkin [1993], where the relation between applicative structures and type theories is studied.

18.3.11. DEFINITION. Let $t = (D_0, \mathsf{i}_0, \mathsf{j}_0)$ be given. We define the family of type theories CDHL(t) as follows.

(i) The partial order $\leq_0$ on $\mathcal{K}(D_0)$ is defined by

$$d \leq_0 e \Leftrightarrow d \sqsupseteq e \quad d, e \in \mathcal{K}(D_0);$$

(ii) $\mathbb{T}^{\mathrm{CDHL}(t)} = \mathcal{K}(D_0) \mid \mathbb{T}^{\mathrm{CDHL}(t)} \to \mathbb{T}^{\mathrm{CDHL}(t)} \mid \mathbb{T}^{\mathrm{CDHL}(t)} \cap \mathbb{T}^{\mathrm{CDHL}(t)}$.

(iii) Write U for $\bot_{D_0}$. Let CDHL(t) be the smallest natural type theory[2] on $\mathbb{T}^{\mathrm{CDHL}(t)}$ that contains the following extra axiom and rules.

$$(\sqcup) \quad c \cap d =_{\mathrm{CDHL}(t)} c \sqcup d,$$

$$(\leq_0) \quad \frac{c \leq_0 d}{c \leq_{\mathrm{CDHL}(t)} d} \qquad (\mathsf{i}_0) \quad \frac{\mathsf{i}_0(e) = (c_1 \mapsto d_1) \sqcup \ldots \sqcup (c_n \mapsto d_n)}{e =_{\mathrm{CDHL}(t)} (c_1 \to d_1) \cap \ldots \cap (c_n \to d_n)},$$

where $c, d, e, , c_1, d_1, \ldots, c_n, d_n \in \mathcal{K}(D_0)$ and $\sqcup$ is the lub for the ordering $\sqsubseteq$ on $\mathcal{K}(D_0)$. Note that for $c, d, e \in \mathcal{K}(D)$ one has $(c \mapsto d), \mathsf{i}_0 \in D_1$.

[2] CDHL(t) contains the axiom and rules of Definitions 15.1.3 and 15.1.18.

(iv) Since CDHL(t) is compatible, it can be considered as a type structure denoted by [CDHL(t)]. Note that [CDHL(t)] $\in \mathbf{NTS}^{\mathtt{U}}$. ■

The proof of the next lemma follows easily from Definition 18.3.11.

18.3.12. LEMMA. $c_1 \cap \ldots \cap c_n =_{\mathrm{CDHL}(t)} c_1 \sqcup \ldots \sqcup c_n$. ■

The proof of $\mathcal{K}(D_\infty^t) \cong \mathrm{CDHL}(t)$ occupies 18.3.13-18.3.18. First we classify the types in $\mathbb{T}^{\mathrm{CDHL}(t)}$according to the maximal number of nested arrow occurrences they may contain.

18.3.13. DEFINITION. (i) We define the map $rank : \mathbb{T}^{\mathrm{CDHL}(t)} \to \mathbb{N}$ by:

$$\begin{array}{rcll} rank(c) & = & 0, & \text{for } c \in \mathcal{K}(D_0); \\ rank(A \to B) & = & \max\{rank(A), rank(B)\} + 1; & \\ rank(A \cap B) & = & \max\{rank(A), rank(B)\}. & \end{array}$$

(ii) Let $\mathbb{T}_n^{\mathrm{CDHL}(t)} = \{A \in \mathbb{T}^{\mathrm{CDHL}(t)} \mid rank(A) \leq n\}$. ■

Notice that $rank$ differs from the map rk in Definition 1.1.14.

We can associate to each type in $\mathbb{T}_n^{\mathrm{CDHL}(t)}$ an element in D_n: this will be crucial for defining the required isomorphism (see Definition 18.3.20).

18.3.14. DEFINITION. We define, for each $n \in \mathbb{N}$, a map $w_n : \mathbb{T}_n^{\mathrm{CDHL}(t)} \to \mathcal{K}(D_n^t)$ by a double induction on n and on the construction of types in $\mathbb{T}^{\mathrm{CDHL}(t)}$:

$$\begin{array}{rcl} w_n(c) & = & \Phi_{0n}(c); \\ w_n(A \cap B) & = & w_n(A) \sqcup w_n(B); \\ w_n(A \to B) & = & (w_{n-1}(A) \mapsto w_{n-1}(B)). \ \blacksquare \end{array}$$

18.3.15. REMARK. From Lemma 17.1.9 we get $\forall A \in \mathbb{T}_n^{\mathrm{CDHL}(t)}. w_n(A) \in \mathcal{K}(D_n^t)$.

18.3.16. LEMMA. *Let $n \leq m$ and $A \in \mathbb{T}_n^{\mathrm{CDHL}(t)}$. Then $\Phi_{m\infty}(w_m(A)) = \Phi_{n\infty}(w_n(A))$.*

PROOF. We show by induction on the definition of w_n that $w_{n+1}(A) = \mathsf{i}_n(w_n(A))$. Then the desired equality follows from the definition of the function Φ. The only interesting case is when $A \equiv B \to C$. We get

$$\begin{array}{rcll} w_{n+1}(B \to C) & = & w_n(B) \mapsto w_n(C), & \text{by definition,} \\ & = & \mathsf{i}_{n-1}(w_{n-1}(B)) \mapsto \mathsf{i}_{n-1}(w_{n-1}(C)), & \text{by induction,} \\ & = & \mathsf{i}_n(w_{n-1}(B) \mapsto w_{n-1}(C)), & \text{by Lemma 18.3.5(ii),} \\ & = & \mathsf{i}_n(w_n(B \to C)), & \text{by Definition 18.3.14. } \blacksquare \end{array}$$

The maps w_n reverse the order between types.

18.3.17. LEMMA. *Let $rank(A \cap B) \leq n$. Then*

$$A \leq_{\mathrm{CDHL}(t)} B \ \Rightarrow \ w_n(B) \sqsubseteq w_n(A).$$

PROOF. The proof is by induction on the definition of $\leq_{\mathrm{CDHL}(t)}$. We consider only two cases.

Case $(\to)$. Let $A \leq_{\mathrm{CDHL}(t)} B$ follows from $A \equiv C{\to}D$, $B \equiv E{\to}F$, $E \leq_{\mathrm{CDHL}(t)} C$and $D \leq_{\mathrm{CDHL}(t)} F$. Then

$$\begin{aligned} E \leq_{\mathrm{CDHL}(t)} C \;\&\; D \leq_{\mathrm{CDHL}(t)} F \;\Rightarrow\; & \\ \Rightarrow\;& w_{n-1}(C) \sqsubseteq w_{n-1}(E) \;\&\; w_{n-1}(F) \sqsubseteq w_{n-1}(D), \\ & \text{by the induction hypothesis,} \\ \Rightarrow\;& w_{n-1}(E) \mapsto w_{n-1}(F) \sqsubseteq w_{n-1}(C) \mapsto w_{n-1}(D) \\ \Rightarrow\;& w_n(B) \sqsubseteq w_n(A). \end{aligned}$$

Case $e =_{\mathrm{CDHL}(t)} (c_1{\to}d_1) \cap \ldots \cap (c_k{\to}d_k)$ follows from $i_0(e) = (c_1 \mapsto d_1) \sqcup \ldots \sqcup (c_k \mapsto d_k)$.
We show by induction on $n \geq 1$ the following.

$$w_n(e) = (w_{n-1}(c_1) \mapsto w_{n-1}(d_1)) \sqcup \ldots \sqcup (w_{n-1}(c_k) \mapsto w_{n-1}(d_k)).$$

It trivially holds for $n = 1$, so let $n > 1$.

$$\begin{aligned} w_n(e) &= \mathsf{i}_{n-1}(w_{n-1}(e)) \\ &= \mathsf{i}_{n-1}((w_{n-2}(c_1) \mapsto w_{n-2}(d_1)) \sqcup \ldots \sqcup (w_{n-2}(c_k) \mapsto w_{n-2}(d_k))) \\ &= \mathsf{i}_{n-1}(w_{n-2}(c_1) \mapsto w_{n-2}(d_1)) \sqcup \ldots \sqcup \mathsf{i}_{n-1}(w_{n-2}(c_k) \mapsto w_{n-2}(d_k)) \\ &= (\mathsf{i}_{n-2}(w_{n-2}(c_1)) \mapsto \mathsf{i}_{n-2}(w_{n-2}(d_1))) \sqcup \ldots \\ &\quad \ldots \sqcup (\mathsf{i}_{n-2}(w_{n-2}(c_k)) \mapsto \mathsf{i}_{n-2}(w_{n-2}(d_k)) \\ &= (w_{n-1}(c_1) \mapsto w_{n-1}(d_1)) \sqcup \ldots \sqcup (w_{n-1}(c_k) \mapsto w_{n-1}(d_k)). \blacksquare \end{aligned}$$

Also the reverse implication of Lemma 18.3.17 holds.

18.3.18. LEMMA. *Let* $rank(A \cap B) \leq n$. *Then*

$$w_n(B) \sqsubseteq w_n(A) \;\Rightarrow\; A \leq_{\mathrm{CDHL}(t)} B.$$

PROOF. By induction on $rank(A \cap B)$.
If $rank(A \cap B) = 0$ we have $A \equiv \bigcap_{i \in I} c_i$, $B = \bigcap_{j \in J} d_j$. Then

$$\begin{aligned} w_n(B) \sqsubseteq w_n(A) \;\Rightarrow\;& \bigsqcup_{j \in J} \Phi_{0n}(d_j) \sqsubseteq \bigsqcup_{i \in I} \Phi_{0n}(c_i) \\ \Rightarrow\;& \Phi_{n0}(\bigsqcup_{j \in J} \Phi_{0n}(d_j)) \sqsubseteq \Phi_{n0}(\bigsqcup_{i \in I} \Phi_{0n}(c_i)) \\ \Rightarrow\;& \bigsqcup_{j \in J} (\Phi_{n0} \circ \Phi_{0n})(d_j) \sqsubseteq \bigsqcup_{i \in I} (\Phi_{n0} \circ \Phi_{0n})(c_i) \\ \Rightarrow\;& \bigsqcup_{j \in J} d_j \sqsubseteq \bigsqcup_{i \in I} c_i \\ \Rightarrow\;& A \leq_{\mathrm{CDHL}(t)} B. \end{aligned}$$

Otherwise, let

$$\begin{aligned} A &\equiv (\bigcap_{i\in I} c_i) \cap (\bigcap_{l\in L} (C_l \to D_l)), \\ B &\equiv (\bigcap_{h\in H} d_h) \cap (\bigcap_{m\in M} (E_m \to F_m)). \end{aligned}$$

By Rule (i_0), we have that

$$c_i =_{\mathrm{CDHL}(t)} \bigcap_{j\in J_i} (a_j \to b_j), \;\; d_h =_{\mathrm{CDHL}(t)} \bigcap_{k\in K_h} (e_k \to f_k),$$

where $a_j, b_j, e_k, f_k \in \mathcal{K}(D_0)$. Now for all $n \geq 1$

$$\begin{aligned} w_n(c_i) &= \bigsqcup_{j\in J_i} (w_{n-1}(a_j) \mapsto w_{n-1}(b_j))), \\ w_n(d_h) &= (\bigsqcup_{k\in K_h} (w_{n-1}(e_k) \mapsto w_{n-1}(f_k))), \end{aligned}$$

since by Lemma 18.3.17 the function w_n identifies elements in the equivalence classes of $=_{\mathrm{CDHL}(t)}$. Therefore

$$\begin{aligned} \bigsqcup_{h\in H} (\bigsqcup_{k\in K_h} w_{n-1}(e_k) \mapsto w_{n-1}(f_k)) \sqcup (\bigsqcup_{m\in M} w_{n-1}(E_m) \mapsto w_{n-1}(F_m)) \;\sqsubseteq \\ \bigsqcup_{i\in I} (\bigsqcup_{j\in J_i} w_{n-1}(a_j) \mapsto w_{n-1}(b_j)) \sqcup (\bigsqcup_{l\in L} w_{n-1}(C_l) \mapsto w_{n-1}(D_l)). \end{aligned}$$

Hence for each $h \in H$, $k \in K_h$ we have

$$\begin{aligned} (w_{n-1}(e_k) \mapsto w_{n-1}(f_k)) \;\sqsubseteq\; & \bigsqcup_{i\in I} (\bigsqcup_{j\in J_i} w_{n-1}(a_j) \mapsto w_{n-1}(b_j)) \sqcup \\ & (\bigsqcup_{l\in L} w_{n-1}(C_l) \mapsto w_{n-1}(D_l)). \end{aligned}$$

Suppose $w_{n-1}(f_k) \neq \bot_{D_n}$. By Lemma 17.1.10 there exist $I' \subseteq I$, $J'_i \subseteq J_i$, $L' \subseteq L$ such that

$$\begin{aligned} \bigsqcup_{i\in I'} (\bigsqcup_{j\in J'_i} w_{n-1}(a_j)) \sqcup (\bigsqcup_{l\in L'} w_{n-1}(C_l)) &\sqsubseteq w_{n-1}(e_k), \\ \bigsqcup_{i\in I'} (\bigsqcup_{j\in J'_i} w_{n-1}(b_j)) \sqcup (\bigsqcup_{l\in L'} w_{n-1}(D_l)) &\sqsupseteq w_{n-1}(f_k). \end{aligned}$$

Notice that all types involved in the two above judgments have ranks strictly less than $rank(A \cap B)$:

1. the rank of a_j, b_j, e_k, f_k is 0, since they are all atoms in $\mathcal{K}(D_0)$;

2. the rank of C_l, D_l is stricly smaller than the one of $A \cap B$, since they are subterms of an arrow in A.

Then by induction and by Lemma 18.3.12 we obtain

$$e_k \leq_{\mathrm{CDHL}(t)} \bigcap_{i\in I'}\Big(\bigcap_{j\in J'_i} a_j\Big)\cap \bigcap_{l\in L'} C_l,$$
$$f_k \geq_{\mathrm{CDHL}(t)} \bigcap_{i\in I'}\Big(\bigcap_{j\in J'_i} b_j\Big)\cap \bigcap_{l\in L'} D_l.$$

Therefore, by $(\to)$ and Proposition 15.1.21, we have $A \leq_{\mathrm{CDHL}(t)} e_k \to f_k$. If $w_{n-1}(f_k) = \bot_{D_n}$, then $w_{n-1}(f_k) = \Phi_{0n}(f_k)$ since $f_k \in \mathcal{K}(D_0)$. This gives $f_k = \Phi_{n0}\circ\Phi_{0n}(f_k) = \Phi_{n0}(\bot_{D_n}) = \bot_{D_0}$ because $j_n(\bot_{D_{n+1}}) = \bot_{D_n}$. Since $f_k = \bot_{D_0}$ implies $A \leq_{\mathrm{CDHL}(t)} e_k \to f_k$ we are done.

In a similar way we can prove that $A \leq_{\mathrm{CDHL}(t)} E_m \to F_m$, for any $m \in M$. Putting together these results we get $A \leq_{\mathrm{CDHL}(t)} B$. ■

18.3.19. Proposition. *$\langle \mathcal{K}(D^t_\infty), \leq_\infty, \cap, \to_\infty, \mathtt{U}\rangle$ is a natural type structure, where $\leq_\infty$ is the reverse order on $\mathcal{K}(D^t_\infty)$, $a \to_\infty b = G_\infty(a \mapsto b)$, $\cap$ is the least upper bound of D^t_∞ and $\mathtt{U} = \bot_{D^t_\infty}$.*

Proof. By Lemma 17.2.12. ■

We can now prove the isomorphism in $\mathbf{NTS}^{\mathtt{U}}$ between $\mathcal{K}(D^t_\infty)$ and CDHL(t) seen as type structure, i.e. [CDHL(t)].

18.3.20. Definition. For $A \in \mathbb{T}^{\mathrm{CDHL}(t)}$ write

$$\mathsf{m}([A]) = \Phi_{r\infty}(w_r(A)),$$

where $r \geq \mathit{rank}(A)$.

18.3.21. Theorem. *In $\mathbf{NTS}^{\mathtt{U}}$ one has $\mathcal{K}(D^t_\infty) \cong$ [CDHL(t)] via m.*

Proof. First of all notice that m is well defined, in the sense the it does not depend on either the type chosen in $[A]$ or the rank r. In fact let $B, B' \in [A]$, and let $p \geq \mathit{rank}(B)$, $p' \geq \mathit{rank}(B')$. Fix any $q \geq p, p'$. Then we have

$$\begin{aligned}
\Phi_{p\infty}(w_p(B)) &= \Phi_{q\infty}(w_q(B)), && \text{by Lemma 18.3.16,}\\
&= \Phi_{q\infty}(w_q(B')), && \text{by Lemma 18.3.17,}\\
&= \Phi_{p'\infty}(w_{p'}(B')), && \text{by Lemma 18.3.16.}
\end{aligned}$$

Write $\mathsf{m}(A)$ for $\mathsf{m}([A])$. Now m is injective by Lemma 18.3.18 and monotone by Lemma 18.3.17. From Lemma 17.1.11(ii) we get immediately

$$\mathcal{K}(D_{n+1}) = \{c_1 \mapsto d_1 \sqcup \ldots \sqcup c_n \mapsto d_n | c_i, d_i \in \mathcal{K}(D_n)\}.$$

it is easily proved by induction on n that w_n is surjective on $\mathcal{K}(D_n)$, hence m is surjective by Lemma 18.3.7(i). The function m^{-1} is monotone by Lemma 18.3.18. Taking into account that the order $\leq_\infty$ on $\mathcal{K}(D^t_\infty)$ is the reversed of $\sqsubseteq$ of D^t_∞ and that $d \to_\infty e = G_\infty(d \mapsto e)$, we need to show

$$A \leq_{\mathrm{CDHL}(t)} B \to C \iff \mathsf{m}(A) \sqsupseteq G_\infty(\mathsf{m}(B) \mapsto \mathsf{m}(C)) \tag{18.1}$$

In order to prove (18.1), let $r \geq max\{rank(A), rank(B \to C)\}$ (in particular it follows $rank(B), rank(C) \leq r-1$). We have

$$\begin{aligned} G_\infty(\mathsf{m}(B) \mapsto \mathsf{m}(C)) &= G_\infty(\Phi_{(r-1)\infty}(w_{r-1}(B)) \mapsto \Phi_{(r-1)\infty}(w_{r-1}(C))) \\ &= G_\infty(\Phi_{(r-1)\infty} \circ (w_{r-1}(B) \mapsto w_{r-1}(C)) \circ \Phi_{\infty(r-1)}), \\ &\quad \text{by Lemma 18.3.5(viii),} \\ &= \Phi_{r\infty}(w_{r-1}(B) \mapsto w_{r-1}(C)), \quad \text{by Lemma 18.3.7(iii),} \\ &= \Phi_{r\infty}(w_r(B \to C)), \quad \text{by definition of } w_r. \end{aligned}$$

Finally we have

$$\begin{aligned} A \leq_{\mathrm{CDHL}(t)} B \to C &\iff w_r(A) \sqsupseteq w_r(B \to C), \\ &\quad \text{by Lemmas 18.3.17 and 18.3.18,} \\ &\iff \Phi_{r\infty}(w_r(A)) \sqsupseteq \Phi_{r\infty}(w_r(B \to C)), \\ &\quad \text{since } \Phi_{r\infty} \text{ is an embedding,} \\ &\iff \Phi_{r\infty}(w_r(A)) \sqsupseteq G_\infty(\mathsf{m}(B) \mapsto \mathsf{m}(C)), \quad \text{as above,} \\ &\iff \mathsf{m}(A) \sqsupseteq G_\infty(\mathsf{m}(B) \mapsto \mathsf{m}(C)). \\ &\iff \mathsf{m}(A) \leq_\infty \mathsf{m}(B) \to_\infty \mathsf{m}(C) \end{aligned}$$

So we have proved (18.1) and the proof is complete. ■

18.3.22. THEOREM. $\mathcal{F}^{[\mathrm{CDHL}(t)]} \cong D^t_\infty$ *in* **NLS**, *via the map*

$$\hat{\mathsf{m}}(X) = \bigsqcup\{\mathsf{m}(B) \mid B \in X\},$$

satisfying $\hat{\mathsf{m}}(\uparrow A) = \mathsf{m}(A)$.

PROOF. Let $\hat{\mathsf{m}} : [\mathrm{CDHL}(t)] \to \mathcal{K}(D_\infty)$ be the isomorphism in $\mathbf{NTS}^{\mathtt{U}}$. By Proposition 17.2.10 we know that $\mathcal{A}$ is a functor from $\mathbf{NTS}^{\mathtt{U}}$ to **NLS**. Then

$$\mathcal{A}(\mathsf{m}) : \mathcal{F}^{[\mathrm{CDHL}(t)]} \to \mathcal{F}^{\mathcal{K}(D_\infty)}$$

is an isomorphism in **NLS** where $\mathcal{A}(\mathsf{m})(X) = \{B \mid \exists A \in X.\mathsf{m}(A) \sqsubseteq B\}$.

By Proposition 17.2.14 we have that

$$\tau : \mathcal{F}^{\mathcal{K}(D^t_\infty)} \to D^t_\infty$$

is an isomorphism in **NLS** where $\tau(X) = \bigsqcup X$.

The composition of τ and $\mathcal{A}(\mathsf{m})$ is an isomorphism from $\mathcal{F}^{[\mathrm{CDHL}(t)]}$ to D_∞ explicitly given by

$$\begin{aligned} \tau \circ \mathcal{A}(\mathsf{m})(X) &= \bigsqcup\{B \mid \exists A \in X.\mathsf{m}(A) \sqsubseteq B\} \\ &= \bigsqcup\{\mathsf{m}(A) \mid A \in X\} = \hat{\mathsf{m}}(X). \blacksquare \end{aligned}$$

18.3.23. COROLLARY. $\mathcal{F}^{\mathrm{CDHL}(t)} \cong D^t_\infty$ *in* **NLS**.

PROOF. By Lemma 15.4.7. ■

Specific models D_∞ as filter models

In this subsection we specialize Theorem 18.3.22 to D^t_∞ models constructed from specific triples $t = (D_0, \mathsf{i}_0, \mathsf{j}_0)$, five in total, each satisfying a specific model theoretic property. For each $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{DHM}, \text{HR}\}$, we associate a triple $\mathsf{init}(\mathcal{T}) = (D_0, \mathsf{i}_0, \mathsf{j}_0)$ such that $\text{CDHL}(\mathsf{init}(\mathcal{T})) = \mathcal{T}$. By Corollary 18.3.23,

$$D_\infty^{\mathsf{init}(\mathcal{T})} \cong \mathcal{F}^{\text{CDHL}(\mathsf{init}(\mathcal{T}))} = \mathcal{F}^{\mathcal{T}}.$$

We will write $D^{\mathcal{T}}_\infty := D_\infty^{\mathsf{init}(\mathcal{T})}$, so that this reads smoothly as $D^{\mathcal{T}}_\infty \cong \mathcal{F}^{\mathcal{T}}$.

18.3.24. Remark. (i) Remember the following type theoretic axioms.

(0_{Scott})	$(\mathtt{U}\to 0) = 0$
(0_{Park})	$(0\to 0) = 0$
(01)	$0 \leq 1$
$(1\to 0)$	$(1\to 0) = 0$
$(0\to 1)$	$(0\to 1) = 1$
(I)	$(1\to 1) \cap (0\to 0) = 1$

(ii) In Definition 15.1.14, each $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{DHM}, \text{HR}\}$ has been defined by specifying its set of atoms $\mathbb{A}^{\mathcal{T}}$ and some extra axioms (besides the axioms $(\to\cap)$, $(\mathtt{U}_{\text{top}})$ and $(\mathtt{U}\to)$ and the rule $(\to)$).

$\mathcal{T}$	Atoms $\mathbb{A}^{\mathcal{T}}$	Axioms of $\mathcal{T}$
Scott	$\{\mathtt{U}, 0\}$	(0_{Scott})
Park	$\{\mathtt{U}, 0\}$	(0_{Park})
CDZ	$\{\mathtt{U}, 0, 1\}$	$(01), (1\to 0), (0\to 1)$
DHM	$\{\mathtt{U}, 0, 1\}$	$(01), (0\to 1), (0_{Scott})$
HR	$\{\mathtt{U}, 0, 1\}$	$(01), (1\to 0), (\mathsf{I})$

18.3.25. Definition. For each $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{DHM}, \text{HR}\}$, we associate a triple $\mathsf{init}(t) = (D_0^{\mathcal{T}}, \mathsf{i}_0^{\mathcal{T}}, \mathsf{j}_0^{\mathcal{T}})$ defined as follows.

1. We define $D_0^{\mathcal{T}}$ as either a two point chain or a three point chain depending on $\mathcal{T}$ as follows.

$\mathcal{T}$	$D_0^{\mathcal{T}}$
Scott, Park	$\mathtt{U} \sqsubseteq 0$
CDZ, DHM, HR	$\mathtt{U} \sqsubseteq 1 \sqsubseteq 0$

2. We define $\mathsf{i}_0^{\mathcal{T}} : D_0^{\mathcal{T}} \to [D_0^{\mathcal{T}} \to D_0^{\mathcal{T}}]$ as follows.

$$\mathsf{i}_0^{\mathcal{T}}(\mathtt{U}) = \mathtt{U} \mapsto \mathtt{U} \text{ for any } \mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{DHM}, \text{HR}\}$$

$$\mathsf{i}_0^{\mathcal{T}}(1) = \begin{cases} 0 \mapsto 1 & \text{if } \mathcal{T} \in \{\text{CDZ}, \text{DHM}\} \\ (1 \mapsto 1) \sqcup (0 \mapsto 0) & \text{if } \mathcal{T} \in \{\text{HR}\} \end{cases}$$

$$\mathsf{i}_0^{\mathcal{T}}(0) = \begin{cases} \mathtt{U} \mapsto 0 & \text{if } \mathcal{T} \in \{\text{Scott}, \text{DHM}\} \\ 0 \mapsto 0 & \text{if } \mathcal{T} \in \{\text{Park}\} \\ 1 \mapsto 0 & \text{if } \mathcal{T} \in \{\text{CDZ}, \text{HR}\} \end{cases}$$

3. Then we define $\mathsf{j}_0^{\mathcal{T}} : [D_0^{\mathcal{T}} \to D_0^{\mathcal{T}}] \to D_0^{\mathcal{T}}$ as follows.

$$\mathsf{j}_0^{\mathcal{T}}(f) \;=\; \bigsqcup\{d \in D_0^{\mathcal{T}} \mid \mathsf{i}_0^{\mathcal{T}}(d) \sqsubseteq f\} \qquad \text{for } f \in [D_0^{\mathcal{T}} \to D_0^{\mathcal{T}}]. \;\blacksquare$$

It is easy to prove that $\langle \mathsf{i}_0^{\mathcal{T}}, \mathsf{j}_0^{\mathcal{T}} \rangle$ is an embedding-projection pair from $D_0^{\mathcal{T}}$ to $[D_0^{\mathcal{T}} \to D_0^{\mathcal{T}}]$, so we can build $D_\infty^{\mathsf{init}(\mathcal{T})}$ following the steps outlined in Definition18.3.1.

NOTATION. We abbreviate $D_\infty^{\mathcal{T}} := D_\infty^{\mathsf{init}(\mathcal{T})}$.

18.3.26. LEMMA. *Let* $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{DHM}, \text{HR}\}$ *and* $c_1, \ldots, c_n, d_1, \ldots, d_n$, $e_1, \ldots, e_k, f_1, \ldots, f_k \in D_0^{\mathcal{T}}$ *. Then*

$$(e_1 \to f_1) \cap \ldots \cap (e_k \to f_k) =_{\mathcal{T}} (c_1 \to d_1) \cap \ldots \cap (c_n \to d_n) \iff$$
$$(c_1 \mapsto d_1) \sqcup \ldots \sqcup (c_n \mapsto d_n) = (e_1 \mapsto f_1) \sqcup \ldots \sqcup (e_k \mapsto f_k).$$

PROOF. It suffices to prove

$$(c \mapsto d) \sqsubseteq (e_1 \mapsto f_1) \sqcup \ldots \sqcup (e_k \mapsto f_k) \iff (e_1 \to f_1) \cap \ldots \cap (e_k \to f_k) \leq_{\mathcal{T}} (c \to d).$$

Now, $(c \mapsto d) \sqsubseteq (e_1 \mapsto f_1) \sqcup \ldots \sqcup (e_k \mapsto f_k)$

$\iff$ $\exists I \subseteq \{1, \ldots, k\} \, [\sqcup_{i \in I} e_i \sqsubseteq c \;\&\; d \sqsubseteq \sqcup_{i \in I} f_i]$, by Lemma 17.1.10,
$\iff$ $\exists i_1, \ldots, i_p \in \{1, \ldots, k\} \, [c \leq_{\mathcal{T}} e_{i_1} \cap \ldots \cap e_{i_p} \;\&\; f_{i_1} \cap \ldots \cap f_{i_p} \leq_{\mathcal{T}} d]$,
$\iff$ $(e_1 \to f_1) \cap \ldots \cap (e_k \to f_k) \leq_{\mathcal{T}} (c \to d)$, by β-soundness, $(\to)$ and $(\to\cap)$. $\blacksquare$

18.3.27. COROLLARY. *The definition of* $\mathsf{i}_0^{\mathcal{T}}$ *is canonical. By this we mean that we could have given equivalently the following definition.*

$$\mathsf{i}_o^{\mathcal{T}}(e) = (c_1 \mapsto d_1) \sqcup \ldots \sqcup (c_n \mapsto d_n) \iff e =_{\mathcal{T}} (c_1 \to d_1) \cap \ldots \cap (c_n \to d_n).$$

PROOF. Immediate, by the definition of $\mathsf{i}_0^{\mathcal{T}}$, the axioms $(\mathtt{U}_{\text{top}})$ and $(\mathtt{U} \to)$, the special axioms $(0_{Scott}), (0_{Park}), (1 \to 0), (0 \to 1), (\mathsf{I})$ respectively, and the previous Lemma. $\blacksquare$

18.3.28. PROPOSITION. *For* $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{DHM}, \text{HR}\}$ *one has*

$$\mathcal{T} = \text{CDHL}(\mathsf{init}(\mathcal{T})).$$

PROOF. First of all, we have that $\mathbb{T}^{\mathcal{T}} = \mathbb{T}^{\text{CDHL}(\mathsf{init}(\mathcal{T}))}$ because

$$\begin{array}{rll} \mathbb{A}^{\mathcal{T}} & = \mathcal{K}(D_0^{\mathcal{T}}) & \text{by Definition 18.3.25} \\ & = \mathbb{A}^{\text{CDHL}(\mathsf{init}(\mathcal{T}))} & \text{by Definition 18.3.11} \end{array}$$

since each $D_0^{\mathcal{T}}$ only contains compact elements. It remains to show that

$$A \leq_{\text{CDHL}(\mathsf{init}(\mathcal{T}))} B \iff A \leq_{\mathcal{T}} B.$$

As to $(\Rightarrow)$, this follows by induction on the generation of $\leq_{\text{CDHL}(t)}$ where $t = \mathsf{init}(\mathcal{T})$. Now $\mathcal{T}$ satisfies the axioms $(\to\cap), (\mathtt{U}_{\text{top}})$ and is closed under the

rule $(\to)$, since it is a natural type theory. It remains to show that the extra axiom and rules are valid in this theory.

Axiom $(\sqcup)$. Then, $c \cap d =_{\mathrm{CDHL}(t)} c \sqcup d$, with $c, d \in \mathcal{K}(D_0^{\mathcal{T}}) = D_0^{\mathcal{T}}$, we have, say, $c \sqsubseteq d$. Then $c \sqcup d = d$. Again we have $d \leq_{\mathcal{T}} c$. Therefore $d \leq_{\mathcal{T}} c \cap d \leq_{\mathcal{T}} d$, and hence $c \cap d =_{\mathcal{T}} d = c \sqcup d$.

Rule $(\leq_0)$. Then, $c \leq_{\mathrm{CDHL}(t)} d$ because $d \sqsubseteq c$, we have $d = \mathtt{U}, c = 0$ or $d = c$. Then in all cases $c \leq_{\mathcal{T}} d$, by the axioms $(\mathtt{U}_{\mathrm{top}})$ and (01).

Rule (i_0) where $\mathsf{i}_0 = \mathsf{i}_0^{\mathcal{T}}$. Suppose $e =_{\mathrm{CDHL}(t)} (c_1 \to d_1) \cap \ldots \cap (c_n \to d_n)$, because

$$\mathsf{i}_0^{\mathcal{T}}(e) = (c_1 \mapsto d_1) \sqcup \ldots \sqcup (c_n \mapsto d_n).$$

Then $e =_{\mathcal{T}} (c_1 \to d_1) \cap \ldots \cap (c_n \to d_n)$, by Corollary 18.3.27.

As to $(\Leftarrow)$, the axioms and rules $(\mathtt{U}_{\mathrm{top}}), (\mathtt{U} \to), (\to \cap)$ and $(\to)$ hold for $\mathcal{T}$ by definition. Moreover, all axioms extra of $\mathcal{T}$ hold in $\mathrm{CDHL}(\mathsf{init}(\mathcal{T}))$, by Definition 18.3.25 and the rules $(\leq_0)$, (i_0) of Definition 18.3.11. ■

Now we can obtain the following result

18.3.29. Corollary. *Let $\mathcal{T} \in \{\mathrm{Scott}, \mathrm{Park}, \mathrm{CDZ}, \mathrm{DHM}, \mathrm{HR}\}$. Then in the category* **NLS** *we have*

$$\mathcal{F}^{\mathcal{T}} \cong D_\infty^{\mathcal{T}}.$$

Proof.
$$\begin{array}{lcll} \mathcal{F}^{\mathcal{T}} & = & \mathcal{F}^{\mathrm{CDHL}(\mathsf{init}(\mathcal{T}))}, & \text{by Proposition 18.3.28,} \\ & \cong & D_\infty^{\mathcal{T}}, & \text{by Corollary 18.3.23. ■} \end{array}$$

We will end this subsection by telling what is the interest of the various models D_∞^t. In Barendregt [1984], Theorem 19.2.9, the following result is proved.

18.3.30. Theorem (Hyland and Wadsworth). *Let $t = (D_0, \mathsf{i}_0, \mathsf{j}_0)$, where D_0 is cpo (or object of* **ALG***) with at least two elements and*

$$\begin{array}{lcll} \mathsf{i}_0(d) & = & \lambda\!\!\lambda e \in D_0.d, & \text{for } d \in D_0, \\ \mathsf{j}_0(f) & = & f(\bot_{D_0}), & \text{for } f \in [D_0 \to D_0]. \end{array}$$

Then for $M, N \in \Lambda$ (untyped lambda terms) and $C[\,]$ ranging over contexts

$$D_\infty^t \models M = N \iff \forall C[\,].(C[M] \text{ is solvable} \iff C[N] \text{ is solvable}).$$

In particular, the local structure of D_∞^t (i.e. $\{M = N \mid D_\infty^t \models M = N\}$) is independent of the initial D_0. ■

18.3.31. Corollary. *For t as in the theorem one has for closed terms M, N*

$$D_\infty^t \models M = N \iff \forall A \in \mathbb{T}^{\mathrm{Scott}}[\vdash_\cap^{\mathrm{Scott}} M : A \iff \vdash_\cap^{\mathrm{Scott}} N : A].$$

Proof. Let $M, N \in \Lambda^\emptyset$. Then

$$\begin{array}{lcl} D_\infty^t \models M = N & \iff & D_\infty^{\mathrm{Scott}} \models M = N, \text{ by Theorem 18.3.30,} \\ & \iff & \mathcal{F}^{\mathrm{Scott}} \models M = N, \\ & & \text{by Corollary 18.3.29 and Proposition 18.1.12,} \\ & \iff & \forall A \in \mathbb{T}^{\mathrm{Scott}}[\vdash_\cap^{\mathrm{Scott}} M : A \iff \vdash_\cap^{\mathrm{Scott}} N : A], \\ & & \text{by Theorem 18.2.7. ■} \end{array}$$

The model D_∞^{Park} has been introduced to contrast the following result, see Barendregt [1984], 19.3.6.

18.3.32. THEOREM (Park). *Let t be as in 18.3.30. Then for the untyped λ-term*

$$\mathsf{Y}_{Curry} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$$

one has

$$[\![\mathsf{Y}_{Curry}]\!]^{D_\infty^t} = \mathsf{Y}_{Tarski},$$

where Y_{Tarski} is the least fixed-point combinator on D_∞^t. ■

The model D_∞^{Park} has been constructed to give $\mathsf{Y}_{\mathrm{Curry}}$ a meaning different from $\mathsf{Y}_{\mathrm{Tarski}}$.

18.3.33. THEOREM (Park). $[\![\mathsf{Y}_{Curry}]\!]^{D_\infty^{\mathrm{Park}}} \neq \mathsf{Y}_{Tarski}$. ■

Now this model can be obtained as a simple filter model $D_\infty^{\mathrm{Park}} \cong \mathcal{F}^{\mathrm{Park}}$ and therefore, by Corollory 18.3.29, one has

$$[\![\mathsf{Y}_{\mathrm{Curry}}]\!]^{\mathcal{F}^{\mathrm{Park}}} \neq \mathsf{Y}_{\mathrm{Tarski}}.$$

Other domain equations

Results similar to Theorem 18.3.21 can be given also for other, non-extensional, inverse limit λ-models. These are obtained as solutions of domain equations involving different functors. For instance one can solve the equations

$$\begin{aligned} D &= [D \to D] \times A \\ D &= [D \to D] + A \\ D &= [D \to_\perp D] \times A \\ D &= [D \to_\perp D] + A, \end{aligned}$$

useful for the analysis of models for restricted λ-calculi. In all such cases one gets concise type theoretic descriptions of the λ-models obtained as fixed points of such functors corresponding to suitable choices of the mapping G, see Coppo et al. [1983]. Solutions of these equations will be discussed below. At least the following result is worthwhile mentioning in this respect, see Coppo et al. [1984] for a proof.

18.3.34. PROPOSITION. *The filter λ-model $\mathcal{F}^{\mathrm{BCD}}$ is isomorphic to $\langle D, F, G\rangle$, where D is the initial solution of the domain equation*

$$D = [D \to D] \times \mathsf{P}(\mathbb{A}_\infty)$$

the pair $\langle F, G\rangle$ set up a Galois connection and G is the map that takes the minimal element in the extensionality classes of a function. ■

18.4. Other filter models

Lazy λ-calculus

Intersection types are flexible enough to allow for the description of λ-models which are computationally adequate for the lazy operational semantics (Abramsky and Ong [1993]). Following Berline [2000] we define the notion of lazy λ-model.

18.4.1. Definition. (i) The *order* of an untyped lambda term is

$$\text{order}(M) = sup\{n \mid \exists N.M \twoheadrightarrow_\beta \lambda x_1 \dots x_n.N\},$$

i.e. the upperbound of the number of its initial abstractions modulo β conversion. So $\text{order}(M) \in \mathbb{N} \cup \infty$.

(ii) A λ-model D is *lazy* if

$$[D \models M = N \ \& \ \text{order}(M) = k] \ \Rightarrow \ \text{order}(N) = k.$$

For example $\text{order}(\mathbf{\Omega})$=0, $\text{order}(\mathsf{K})$=2 and $\text{order}(\mathsf{YK}) = \infty$. As usual we have $\mathbf{\Omega} := (\lambda x.xx)(\lambda x.xx)$, $\mathsf{K} := \lambda xy.x$, and $\mathsf{Y} := \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$.

18.4.2. Proposition. *Let $\mathcal{F}^{\mathcal{T}}$ be a filter λ-model. Then $\mathcal{F}^{\mathcal{T}}$ is lazy iff*

$$\forall \Gamma, A.[\Gamma \vdash^{\mathcal{T}}_{\cap} M : A \Leftrightarrow \Gamma \vdash^{\mathcal{T}}_{\cap} N : A] \ \Rightarrow \ order(M) = order(N),$$

i.e. if M and N have the same types, then they have the same order.

Proof. By 18.2.7(i). ■

A very simple type theory is AO, see Figure 15.1.14. This gives a lazy λ-model, which is discussed in Abramsky and Ong [1993]. In this paper the following result is proved, where for a $D \in \mathbf{ALG}$ its lifting $D_\perp$ is defined as the domain obtained by adding a new bottom element.

18.4.3. Theorem (Abramsky and Ong [1993]). *Let D^{lazy}_∞ be the initial solution of the domain equation $D \cong [D \to D]_\perp$ in $\mathbf{ALG}$. Then $D^{lazy}_\infty \cong \mathcal{F}^{\mathrm{AO}}$.* ■

The theory AO can also be used to prove the completeness of the so called *F-semantics*, see Dezani-Ciancaglini and Margaria [1986].

The $\lambda\mathsf{I}$-calculus.

Models of the $\lambda\mathsf{I}$-calculus are considered in Honsell and Lenisa [1993] and[1999]. Like Theorem 18.3.21 one has the following.

18.4.4. Theorem (Honsell and Lenisa [1999]). *Let D^{I}_∞ be the inverse limit solutions of the domain equation $[D \to_\perp D] \cong D$. Then $D^{\mathsf{I}}_\infty \cong \mathcal{F}^{\mathcal{T}}$, for some proper type theory $\mathcal{T}$ with $\mathbb{A}^{\mathcal{T}} = \mathcal{K}(D_0)$.*

Honsell and Lenisa [1999] discusses a filter structure which gives a *computationally adequate* model for the *perpetual* operational semantics and a mathematical model for the maximal sensible $\lambda\mathsf{I}$-theory.

A filter model equating an arbitrary closed term to Ω

In Jacopini [1975] it has been proved by an analysis of conversion that the lambda term $\mathbf{\Omega}$ is easy, i.e. for any closed lambda term M the equation $\mathbf{\Omega} = M$ is consistent. This fact was proved by a Church-Rosser argument by Mitschke, see his[1976] or Barendregt [1984], Proposition 15.3.9. A model theoretical proof was given by Baeten and Boerboom [1979], where it was shown that for any closed M one has

$$\mathcal{P}(\omega) \models \mathbf{\Omega} = M,$$

for a particular way of coding pairs on the set of natural numbers ω. We will now present the proof of this fact from Alessi et al. [2001], using intersection types. For an arbitrary closed λ-term M we will build a filter model $\mathcal{F}^{\mathrm{ADH}(M)}$ such that

$$\mathcal{F}^{\mathrm{ADH}(M)} \models \mathbf{\Omega} = M.$$

We first examine which types can be assigned to $\boldsymbol{\omega} := \lambda x.xx$ and $\mathbf{\Omega} := \boldsymbol{\omega}\boldsymbol{\omega}$.

18.4.5. Lemma. *Let $\mathcal{T}$ be a natural type theory that is β-sound.*

(i) $\vdash^{\mathcal{T}}_{\cap} \boldsymbol{\omega} : A \to B \iff A \leq_{\mathcal{T}} A \to B$.

(ii) $\vdash^{\mathcal{T}}_{\cap} \mathbf{\Omega} : B \iff \exists A \in \mathbb{T}^{\mathcal{T}}. \vdash^{\mathcal{T}}_{\cap} \boldsymbol{\omega} : A \leq_{\mathcal{T}} (A \to B)$.

Proof. (i) ($\Leftarrow$) Suppose $A \leq_{\mathcal{T}} (A \to B)$. Then

$$\begin{array}{lll} x\ A & \vdash^{\mathcal{T}}_{\cap} & x : (A \to B) \\ x\ A & \vdash^{\mathcal{T}}_{\cap} & xx : B \\ & \vdash^{\mathcal{T}}_{\cap} & \lambda x.xx : (A \to B). \end{array}$$

($\Rightarrow$) Suppose $\vdash^{\mathcal{T}}_{\cap} \boldsymbol{\omega} : (A \to B)$. If $B =_{\mathcal{T}} \mathtt{U}$, then $A \leq_{\mathcal{T}} \mathtt{U} =_{\mathcal{T}} (A \to B)$, by axiom $(\mathtt{U} \to)$. Otherwise, by Theorem 16.1.9,

$$\begin{array}{lll} \vdash^{\mathcal{T}}_{\cap} \lambda x.xx : (A \to B) & \Rightarrow & x\ A \vdash^{\mathcal{T}}_{\cap} xx : B, \\ & \Rightarrow & x\ A \vdash^{\mathcal{T}}_{\cap} x : C, \quad x\ A \vdash^{\mathcal{T}}_{\cap} x : (C \to B), \text{ for some } C, \\ & \Rightarrow & A \leq_{\mathcal{T}} (C \to B) \leq_{\mathcal{T}} (A \to B), \qquad \text{by } (\to). \end{array}$$

(ii) ($\Leftarrow$) Immediate. ($\Rightarrow$) If $B =_{\mathcal{T}} \mathtt{U}$, then $\vdash^{\mathcal{T}}_{\cap} \boldsymbol{\omega} : \mathtt{U} \leq_{\mathcal{T}} \mathtt{U} \to B$. If $B \neq_{\mathcal{T}} \mathtt{U}$, then by Theorem 16.1.9(ii) one has $\vdash^{\mathcal{T}}_{\cap} \boldsymbol{\omega} : (A \to B)$, $\vdash^{\mathcal{T}}_{\cap} \boldsymbol{\omega} : A$, for some A. By (i) one has $A \leq_{\mathcal{T}} A \to B$.

We associate to each type the maximum number of nested arrows in the leftmost path.

18.4.6. Definition. Let $\mathcal{T}$ be a type theory. For $A \in \mathbb{T}^{\mathcal{T}}$ its *type nesting*, notation $\#(A)$, is defined inductively on types as follows:

$$\begin{array}{lll} \#(A) & = & 0 \qquad\qquad\qquad \text{if } A \in \mathbb{A}^{\mathcal{T}}; \\ \#(A \to B) & = & \#(A) + 1; \\ \#(A \cap B) & = & \max\{\#(A), \#(B)\}. \end{array}$$

For $\boldsymbol{\eta}^{\mathtt{U}}$-sound and natural type theories Lemma 18.4.5(ii) can be strengthened using type nesting. First we need the following lemma that shows that in a β-sound type theory, Definition 16.1.4, any type A with $\#(A) \geq 1$ is equivalent to an intersection of arrows with the same type nesting.

18.4.7. LEMMA. *Let $\mathcal{T}$ be $\boldsymbol{\eta}^{\mathtt{U}}$-sound and natural type theory. Then for all $A \in \mathbb{T}^{\mathcal{T}}$ with $\#(A) \geq 1$, there exist $C_1, \ldots, C_m, D_1, \ldots, D_m$ such that*

$$\begin{array}{rcl} A & =_{\mathcal{T}} & (C_1 \to D_1) \cap \ldots \cap (C_m \to D_m); \\ \#(A) & = & \#((C_1 \to D_1) \cap \ldots \cap (C_m \to D_m)). \end{array}$$

PROOF. Every type A is an intersection of arrow types and atoms. Since $\mathcal{T}$ is $\boldsymbol{\eta}^{\mathtt{U}}$-sound and natural, the atoms can be replaced by an intersection of arrows between atoms. As $\#(A) \geq 1$ this does not increase the type nesting. ■

18.4.8. LEMMA. *Let $\mathcal{T}$ be a natural type theory which is β and $\boldsymbol{\eta}^{\mathtt{U}}$-sound. Then*

$$\vdash^{\mathcal{T}}_{\cap} \boldsymbol{\Omega} : B \;\Rightarrow\; \exists A \in \mathbb{T}^{\mathcal{T}} [\vdash^{\mathcal{T}}_{\cap} \boldsymbol{\omega} : A \leq_{\mathcal{T}} A \to B \;\&\; \#(A) = 0].$$

PROOF. Let $\vdash^{\mathcal{T}}_{\cap} \boldsymbol{\Omega} : B$. If $B =_{\mathcal{T}} \mathtt{U}$ take $A \equiv \mathtt{U}$. Otherwise, by Lemma 18.4.5(ii), there exists $A \in \mathbb{T}^{\mathcal{T}}$ such that $\vdash^{\mathcal{T}}_{\cap} \boldsymbol{\omega} : A$ and $A \leq_{\mathcal{T}} A \to B$. We show by course of value induction on $n = \#(A)$ that we can take an alternative A' with $\#(A') = 0$. If $n = 0$ we are done, so suppose $n \geq 1$. By Lemma 18.4.7, we may assume that A is of the form $A \equiv (C_1 \to D_1) \cap \ldots \cap (C_m \to D_m)$. Now $A \leq_{\mathcal{T}} A \to B$, hence $A \leq_{\mathcal{T}} C_{i_1} \cap \ldots \cap C_{i_p}$ and $D_{i_1} \cap \ldots \cap D_{i_p} \leq_{\mathcal{T}} B$, with $p > 0$ and $1 \leq i_1, \ldots, i_p \leq m$, since $\mathcal{T}$ is β-sound. Hence,

$$\begin{array}{rcl} \vdash^{\mathcal{T}}_{\cap} \boldsymbol{\omega} : A & \Rightarrow & \vdash^{\mathcal{T}}_{\cap} \boldsymbol{\omega} : (C_{i_k} \to D_{i_k}), 1 \leq k \leq p, \\ & \Rightarrow & C_{i_k} \leq_{\mathcal{T}} (C_{i_k} \to D_{i_k}), \text{ by 18.4.5(i)}, \\ & \Rightarrow & C_{i_1} \cap \ldots \cap C_{i_p} \leq_{\mathcal{T}} (C_{i_1} \to D_{i_1}) \cap \ldots \cap (C_{i_p} \to D_{i_p}) \\ & & \leq_{\mathcal{T}} C_{i_1} \cap \ldots \cap C_{i_p} \to D_{i_1} \cap \ldots \cap D_{i_p}, \text{ since } \mathcal{T} \text{ is natural}, \\ & & \leq_{\mathcal{T}} (C_{i_1} \cap \ldots \cap C_{i_p} \to B), \text{ as } D_{i_1} \cap \ldots \cap D_{i_p} \leq_{\mathcal{T}} B. \end{array}$$

Now take $A' \equiv C_{i_1} \cap \ldots \cap C_{i_p}$. Then $\#(A') < n$ and we are done by the IH. ■

Now let $M \in \Lambda^{\emptyset}$. We will build the desired model satisfying $\models \boldsymbol{\Omega} = M$ by taking the union of a countable sequence of type theories $\mathrm{ADH}_n(M)$ defined in a suitable way to force the final interpretation of M to coincide with the interpretation of $\boldsymbol{\Omega}$. In the following $\langle \cdot, \cdot \rangle$ denotes any bijection between $\mathbb{N} \times \mathbb{N}$ and $\mathbb{N}$.

18.4.9. DEFINITION. (i) Define the growing sequence of intersection type structures $\mathcal{T}_n$ by induction on $n \in \mathbb{N}$, specifying the atoms, axioms and rules.

- $\mathbb{A}^{\mathrm{ADH}_0(M)} = \{\mathtt{U}, 0\}$;
- $\mathrm{ADH}_0(M)$ is the smallest natural type theory that contains (0_{Scott});
- $\mathbb{A}^{\mathrm{ADH}_{n+1}(M)} = \mathbb{A}^{\mathrm{ADH}_n(M)} \cup \{\xi_{\langle n,m \rangle} \mid m \in \mathbb{N}\}$;

- $\mathrm{ADH}_{n+1}(M)$ is the smallest natural type theory that contains $\mathrm{ADH}_n(M)$ and the following infinite set of axioms

$$\xi_{\langle n,m\rangle} = (\xi_{\langle n,m\rangle} \to W_{\langle n,m\rangle})\} \text{ for } m \in \mathbb{N},$$

where $\langle W_{\langle n,m\rangle}\rangle_{m\in\mathbb{N}}$ is any enumeration of the set

$$\{A \mid \vdash_{\cap}^{\mathrm{ADH}_n(M)} M : A\}.$$

(ii) We define $\mathrm{ADH}(M)$ as follows:

$$\mathbb{A}^{\mathrm{ADH}(M)} = \bigcup_{n\in\mathbb{N}} \mathbb{A}^{\mathrm{ADH}_n(M)}; \quad \mathrm{ADH}(M) = \bigcup_{n\in\mathbb{N}} \mathrm{ADH}_n(M).$$

18.4.10. Proposition. $\mathrm{ADH}(M)$ *is a* $\beta, \boldsymbol{\eta}^{\mathtt{U}}$*-sound natural type theory.*

Proof. It is immediate to check that $\mathrm{ADH}(M)$ is β and $\boldsymbol{\eta}^{\mathtt{U}}$-sound. Clearly the axioms $(\mathtt{U}_{\mathrm{top}}), (\mathtt{U} \to)$ and $(\to\cap)$ are valid in $\mathrm{ADH}(M)$, as they are already in $\mathrm{ADH}_0(M)$. The validity of rule $(\to)$ in $\mathrm{ADH}(M)$ follows by a "compactness" argument: if $A' \leq_{\mathrm{ADH}(M)} A \;\&\; B \leq_{\mathrm{ADH}(M)} B'$, then $A' \leq_{\mathrm{ADH}_n(M)} A \;\&\; B \leq_{\mathrm{ADH}_m(M)} B'$; but then $(A{\to}B) \leq_{\mathrm{ADH}(\max(n,m))M} (A'{\to}B')$ and hence $(A{\to}B) \leq_{\mathrm{ADH}(M)} (A'{\to}B')$. Therefore the type theory is natural. ■

18.4.11. Theorem. $\mathcal{F}^{\mathrm{ADH}(M)}$ *is an extensional filter* λ*-model.*

Proof. By Propositions 18.4.10 and 18.2.11. ■

We now need to show that some types cannot be deduced for $\boldsymbol{\omega}$.

18.4.12. Lemma. $\not\vdash_{\cap}^{\mathrm{ADH}(M)} \boldsymbol{\omega} : 0$ *and* $\not\vdash_{\cap}^{\mathrm{ADH}(M)} \boldsymbol{\omega} : (0 \to 0) \to 0 \to 0$.

Proof. Define the set $\mathcal{E}_{\mathtt{U}} \subseteq \mathbb{T}(\mathbb{A}^{\mathrm{ADH}(M)})$ as the minimal set such that:

$$\begin{array}{lll} \mathtt{U} \in \mathcal{E}_{\mathtt{U}}; & & \\ A \in \mathbb{T}^{\mathrm{ADH}(M)}, B \in \mathcal{E}_{\mathtt{U}} & \Rightarrow & (A \to B) \in \mathcal{E}_{\mathtt{U}}; \\ A, B \in \mathcal{E}_{\mathtt{U}} & \Rightarrow & (A \cap B) \in \mathcal{E}_{\mathtt{U}}; \\ W_i \in \mathcal{E}_{\mathtt{U}} & \Rightarrow & \xi_i \in \mathcal{E}_{\mathtt{U}}. \end{array}$$

Claim: $A \in \mathcal{E}_{\mathtt{U}} \iff A =_{\mathrm{ADH}(M)} \mathtt{U}$.

($\Rightarrow$) By induction on the definition of $\mathcal{E}_{\mathtt{U}}$, using axiom $(\mathtt{U} \to)$.

($\Leftarrow$) By induction on $\leq_{\mathrm{ADH}(M)}$ it follows that

$$\mathcal{E}_{\mathtt{U}} \ni B \leq_{\mathrm{ADH}(M)} A \;\Rightarrow\; A \in \mathcal{E}_{\mathtt{U}}.$$

Hence if $A =_{\mathrm{ADH}(M)} \mathtt{U}$, one has $\mathcal{E}_{\mathtt{U}} \ni \mathtt{U} \leq_{\mathrm{ADH}(M)} A$ and thus $A \in \mathcal{E}_{\mathtt{U}}$.

As $0 \notin \mathcal{E}_{\mathtt{U}}$, it follows by the claim that

$$\mathtt{U} \neq_{\mathrm{ADH}(M)} 0. \tag{18.2}$$

Suppose towards a contradiction that $\vdash_{\cap}^{\mathrm{ADH}(M)} \boldsymbol{\omega} : 0$. Then $\vdash^{\mathrm{ADH}(M)} \boldsymbol{\omega} : \mathtt{U} \to 0$, by (0_{Scott}). By Lemma 18.4.5(i) we get $\mathtt{U} \leq_{\mathrm{ADH}(M)} (\mathtt{U} \to 0) =_{\mathrm{ADH}(M)} 0 \leq_{\mathrm{ADH}(M)} \mathtt{U}$, i.e. $\mathtt{U} =_{\mathrm{ADH}(M)} 0$, contradicting (18.2).

Similarly from $\vdash_{\cap}^{\mathrm{ADH}(M)} \boldsymbol{\omega} : (0 \to 0) \to 0 \to 0$, by Lemma 18.4.5(i), we get $0 \to 0 \leq_{\mathrm{ADH}(M)} (0 \to 0) \to (0 \to 0)$, which implies $0 \to 0 \leq_{\mathrm{ADH}(M)} 0 \leq_{\mathrm{ADH}(M)} \mathtt{U} \to 0$, by β-soundness and (0_{Scott}). Therefore $\mathtt{U} =_{\mathrm{ADH}(M)} 0$, contradicting (18.2). ■

We finally are able to prove the main theorem.

18.4.13. Theorem. *Let $M \in \Lambda^{\emptyset}$. Then $\mathcal{F}^{\mathrm{ADH}(M)}$ is a non-trivial extensional λ-model such that $\mathcal{F}^{\mathrm{ADH}(M)} \models M = \boldsymbol{\Omega}$.*

Proof. The model is non-trivial since clearly $\vdash_{\cap}^{\mathrm{ADH}(M)} \mathsf{I} : (0 \to 0) \to 0 \to 0$ and by Lemma 18.4.12 $\nvdash_{\cap}^{\mathrm{ADH}(M)} \boldsymbol{\omega} : (0 \to 0) \to 0 \to 0$, therefore it follows that $\mathcal{F}^{\mathrm{ADH}(M)} \not\models \mathsf{I} = \boldsymbol{\omega}$.

We must show that $[\![M]\!] = [\![\boldsymbol{\Omega}]\!]$. Suppose that $W \in [\![M]\!]$. Then

$$\begin{array}{lll} \vdash_{\cap}^{\mathrm{ADH}(M)} M : W & \Rightarrow & \vdash_{\cap}^{\mathrm{ADH}_n(M)} M : W, \qquad \text{for some } n, \\ & \Rightarrow & \xi_i =_{\mathrm{ADH}_{n+1}(M)} (\xi_i \to W), \qquad \text{for some } i, \\ & \Rightarrow & \vdash_{\cap}^{\mathrm{ADH}(M)} \boldsymbol{\Omega} : W, \\ & \Rightarrow & W \in [\![\boldsymbol{\Omega}]\!]. \end{array}$$

This proves $[\![M]\!] \subseteq [\![\boldsymbol{\Omega}]\!]$.

Now suppose $B \in [\![\boldsymbol{\Omega}]\!]$, i.e. $\vdash_{\cap}^{\mathrm{ADH}(M)} \boldsymbol{\Omega} : B$. Then by Lemma 18.4.8 there exists A such that $\#(A) = 0$ and $\vdash_{\cap}^{\mathrm{ADH}(M)} \boldsymbol{\omega} : A \leq_{\mathrm{ADH}(M)} A \to B$. Let $A \equiv \bigcap_{i \in I} \psi_i$, with $\psi_i \in \mathbb{A} = \{\mathtt{U}, 0, \xi_0, \ldots\}$. By Lemma 18.4.12 $\psi_i \neq_{\mathrm{ADH}(M)} 0$. Hence it follows that $A =_{\mathrm{ADH}(M)} \mathtt{U}$ or $A =_{\mathrm{ADH}(M)} \bigcap_{j \in J}(\xi_j)$, for some finite $J \subseteq \mathbb{N}$. Since $\mathtt{U} =_{\mathrm{ADH}(M)} (\mathtt{U} \to \mathtt{U})$ and $\xi_j =_{\mathrm{ADH}(M)} (\xi_j \to W)$ we get $A =_{\mathrm{ADH}(M)} (\mathtt{U} \to \mathtt{U})$ or $A =_{\mathrm{ADH}(M)} \bigcap_{j \in J}(\xi_j \to W_j)$. Since $A \leq_{\mathrm{ADH}(M)} A \to B$ it follows by β-soundness that in the first case $\mathtt{U} \leq_{\mathrm{ADH}(M)} B$ or in the second case $\bigcap_{j \in L} W_j \leq_{\mathrm{ADH}(M)} B$, for some $L \subseteq J$. Since each W_j is in $[\![M]\!]$, we have in both cases $B \in [\![M]\!]$. This shows $[\![\boldsymbol{\Omega}]\!] \subseteq [\![M]\!]$ and we are done. ■

Graph models as filter models

Engeler's Model

18.4.14. Definition (Engeler [1981]). Let $\mathbb{A}_\infty$ be a countable set of atoms.

(i) Define Em as the least set satisfying $\mathsf{Em} = \mathbb{A}_\infty \cup (\mathsf{P}_{\mathit{fin}}(\mathsf{Em}) \times \mathsf{Em})$

(ii) Define $F_{\mathsf{Em}} : \mathsf{P}(\mathsf{Em}) \to [\mathsf{P}(\mathsf{Em}) \to \mathsf{P}(\mathsf{Em})]$ by
$G_{\mathsf{Em}} : [\mathsf{P}(\mathsf{Em}) \to \mathsf{P}(\mathsf{Em})] \to \mathsf{P}(\mathsf{Em})$

$$\begin{array}{rcl} F_{\mathsf{Em}}(X) & = & \bigsqcup\{u \mapsto e \mid \langle u, e\rangle \in X\} \\ G_{\mathsf{Em}}(f) & = & \{\langle u, e\rangle \mid e \in f(u)\}. \blacksquare \end{array}$$

18.4.15. THEOREM. *$F_{\mathsf{Em}}, G_{\mathsf{Em}}$ satisfy $F_{\mathsf{Em}} \circ G_{\mathsf{Em}} = \mathbf{1}$, making $\mathsf{P}(\mathsf{Em})$ a λ-model.* ■

PROOF. See Engeler [1981]. ■

18.4.16. THEOREM. *Let* Engeler *be as defined in Figure 15.1.14. Then $\mathsf{P}(\mathsf{Em}) \cong \mathcal{F}^{\text{Engeler}}$ are isomorphic as natural λ-structures (λ-models).* ■

PROOF. See Plotkin [1993]. ■

Scott's $\mathsf{P}(\omega)$ model

Following the original notation by Scott, in the $\mathsf{P}(\omega)$ model ω denotes the set of natural numbers. $\mathsf{P}_{\text{fin}}(\omega)$ denotes the set of finite subsets of ω.

NOTATION. (i) Let $\lambda nm.\langle n,m\rangle : \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ be a bijection, e.g. the well-known one defined by $\langle n,m\rangle = \frac{1}{2}(n+m)(n+m+1)+m$.

(ii) Let $\lambda n.e_n : \mathbb{N}\to\mathsf{P}_{\text{fin}}(\omega)$ be a bijection, e.g. the well-known one defined by

$$e_n = \{k_0,\ldots,k_{m-1}\} \text{ with } k_0 < k_1 < \ldots < k_{m-1} \iff n = \Sigma_{i<m}2^i.$$

18.4.17. DEFINITION. [Scott [1972]] Let $\gamma : \mathsf{P}_{fin}(\omega)\times\omega\to\omega$ be the bijection defined by

$$\gamma(e_n,m) = \langle n,m\rangle.$$

(i) Define $F_\omega : \mathsf{P}(\omega)\to[\mathsf{P}(\omega)\to\mathsf{P}(\omega)]$ by

$$F_\omega(X) = \bigsqcup\{u \mapsto i \mid \gamma(u,i)\in X\}.$$

(ii) $G_\omega : [\mathsf{P}(\omega)\to\mathsf{P}(\omega)]\to\mathsf{P}(\omega)$ by

$$G_\omega(f) = \{\gamma(u,i) \mid i\in f(u)\}$$

for all $f\in[\mathsf{P}(\omega)\to\mathsf{P}(\omega)]$. ■

18.4.18. PROPOSITION. *Define for $X,Y\in\mathsf{P}(\omega)$ the application*

$$X\cdot_{\mathsf{P}(\omega)}Y = \{m \mid \exists e_n\subseteq Y\ \langle n,m\rangle\in X\}.$$

Then $F_\omega(X)(Y) = X\cdot_{\mathsf{P}(\omega)}Y$ is a (more common) equivalent definition for F_ω.

PROOF. Do exercise 18.5.5. ■

18.4.19. THEOREM. *$\mathsf{P}(\omega)$ is a λ-model via F_ω, G_ω.*

PROOF. See Scott [1972]. ■

18.4.20. THEOREM. *Define*

$$\begin{array}{lcl} \mathbb{A}^{\text{Alessi}} & = & \omega \\ \text{Alessi} & = & \text{Engeler}\cup\{\bigcap_{k\in e}(k\to n) = \gamma(e,n) \mid e\in\mathsf{P}_{fin}(\omega), n\in\omega\}. \end{array}$$

Then $\mathsf{P}(\omega)\cong\mathcal{F}^{\text{Alessi}}$ are isomorphic as natural λ-structures (λ-models).

PROOF. See Alessi [1991]. ■

Plotkin's Model

18.4.21. DEFINITION (Plotkin [1993]). Let ω be an atom.

(i) Define Pm as the least set such that

$$\mathsf{Pm} = \{\omega\} \cup (\mathsf{P}_{fin}(\mathsf{Pm}) \times \mathsf{P}_{fin}(\mathsf{Pm})).$$

(ii) Define $F_{\mathsf{Pm}} : \mathsf{P}(\mathsf{Pm}) \to [\mathsf{P}(\mathsf{Pm}) \to \mathsf{P}(\mathsf{Pm})]$ by
$G_{\mathsf{Pm}} : [\mathsf{P}(\mathsf{Pm}) \to \mathsf{P}(\mathsf{Pm})] \to \mathsf{P}(\mathsf{Pm})$

$$\begin{aligned} F_{\mathsf{Pm}}(X) &= \bigsqcup\{u \mapsto v \mid \langle u, v\rangle \in X\} \\ G_{\mathsf{Pm}}(f) &= \{\langle u, v\rangle \mid v \subseteq f(u)\}. \blacksquare \end{aligned}$$

18.4.22. THEOREM. *$F_{\mathsf{Pm}}, G_{\mathsf{Pm}}$ satisfy $F_{\mathsf{Pm}} \circ G_{\mathsf{Pm}} = Id$, turning $\mathsf{P}(\mathsf{Pm})$ into a λ-model.* ■

PROOF. See Plotkin [1993]. ■

18.4.23. THEOREM. *Let* Plotkin *be as defined in Figure 15.1.14. Then $\mathsf{P}(\mathsf{Pm}) \cong \mathcal{F}^{\text{Plotkin}}$ are isomorphic as natural λ-structures (λ-models).*

PROOF. See Plotkin [1993]. ■

18.5. Exercises

18.5.1. Check the following equalities:

$$\begin{aligned} [\![\lambda x.y]\!]^{\mathcal{F}^{\text{CDV}}}_{\rho_0} &= \emptyset; \\ [\![\lambda x.y]\!]^{\mathcal{F}^{\text{HL}}}_{\rho_1} &= \uparrow 0; \\ [\![\lambda x.y]\!]^{\mathcal{F}^{\text{AO}}}_{\rho_0} &= \uparrow (\mathtt{U}\to\mathtt{U}); \\ [\![\lambda x.y]\!]^{\mathcal{F}^{\text{EHR}}}_{\rho_0} &= \uparrow \mathtt{V}, \end{aligned}$$

where $\rho_0(y) = \uparrow \emptyset$ and $\rho_1(y) = \uparrow 0$.

18.5.2. (i) Define $\mathsf{K}^\infty \equiv \mathsf{YK}$. This term is called the "ogre". Find a type for it in the system $\lambda\cap^{\text{AO}}$.

(ii) Show that "ogre" inhabits all types in $\lambda\cap^{\text{AO}}$? [Hint. Use (i) and Exercise **??**.]

18.5.3. Prove using the results of Exercise 18.5.2 that $[\![\mathsf{K}^\infty]\!]^{\mathcal{F}^{\text{AO}}}_\rho = \mathcal{F}^{\text{AO}}$.

18.5.4. Define $\mathtt{t} : \mathbb{T}(\{0,1,\mathtt{U}\}) \to \mathbb{T}(\{0,1,\mathtt{U}\})$ inductively:

$$\begin{aligned} \mathtt{t}(\alpha) &= \alpha, && \text{where } \alpha \in \{\mathtt{U}, 0\}; \\ \mathtt{t}(1) &= \mathtt{U}; \\ \mathtt{t}(A\to B) &= A\to\mathtt{t}(B); \\ \mathtt{t}(A\cap B) &= \mathtt{t}(A)\cap\mathtt{t}(B). \end{aligned}$$

The intersection type theory AABD is axiomatized by rule $(\to)$ and axioms $(\to\cap)$, $(\mathtt{U}_{\rm top})$, $(\mathtt{U}\to)$, (01), $(1\to0)$, $(0\to1)$, see Fig. 15.1.14, and $(\mathtt{t})$, $(\mathtt{t}\to)$, where

$$\begin{array}{ll}(\mathtt{t}) & A\leq \mathtt{t}(A);\\ (\mathtt{t}\to) & A\to B\leq \mathtt{t}(A)\to\mathtt{t}(B).\end{array}$$

If $\Gamma=\{x_1\ A_1,\dots,x_n\ A_n\}$, then write $\mathtt{t}(\Gamma)=\{x_1\ \mathtt{t}(A_1),\dots,x_n\ \mathtt{t}(A_n)\}$. Show the following.

(i) The map $\mathtt{t}$ is idempotent, i.e. $\mathtt{t}(\mathtt{t}(A))=\mathtt{t}(A)$.
(ii) $A\to\mathtt{t}(B)=_{\rm AABD}\mathtt{t}(A)\to\mathtt{t}(B)$.
(iii) $A\leq_{\rm AABD}B\Rightarrow \mathtt{t}(A)\leq_{\rm AABD}\mathtt{t}(B)$.
(iv) $\Gamma\vdash^{\rm AABD}M:A\Rightarrow \mathtt{t}(\Gamma)\vdash^{\rm AABD}M:\mathtt{t}(A)$.
(v) $\Gamma,\Gamma'\vdash^{\rm AABD}M:A\Rightarrow\ \Gamma,\mathtt{t}(\Gamma')\vdash^{\rm AABD}M:\mathtt{t}(A)$.
(vi) $\forall i\in I.\,\Gamma,x\ A_i\vdash^{\rm AABD}M:B_i\ \&\ \bigcap_{i\in I}(A_i\to B_i)\leq_{\rm AABD}C\to D$ $\Rightarrow\ \Gamma,x\ C\vdash^{\rm AABD}M:D$.
(vii) AABD is not β-sound. [Hint. $1\to0\leq_{\rm AABD}\mathtt{U}\to0$.]
(viii) $\mathcal{F}^{\rm AABD}$ is a filter λ-model. [Hint. Modify Theorem 18.2.21, and Lemma 18.2.20(vi).]
(ix) The step function $\uparrow 1\Rightarrow\uparrow 0$ is not representable in $\mathcal{F}^{\rm AABD}$.

Actually, $\mathcal{F}^{\rm AABD}$ is the inverse limit solution of the domain equation $D\simeq[D\to D]$ taken in the category of $\mathtt{t}$-*lattices*, whose objects are ω-algebraic lattices D endowed with a finitary additive projection $\delta:D\to D$ and whose morphisms $f:(D,\delta)\to(D',\delta')$ are continuous functions such that $\delta'\circ f\sqsubseteq f\circ\delta$. See Alessi [1993], Alessi, Barbanera and Dezani-Ciancaglini [2004] for details.

18.5.5. Show Proposition 18.4.18.

18.5.6. Show Theorem 18.4.20 using the mapping $\mathsf{m}:\mathsf{P}(\omega)\to\mathbb{T}^{\rm Alessi}$ defined as

$$\begin{array}{lcl}\mathsf{m}(\emptyset) & = & \mathtt{U},\\ \mathsf{m}(\{i\}) & = & i,\\ \mathsf{m}(u\cup\{i\}) & = & \mathsf{m}(u)\cap i.\end{array}$$

where $u\in\mathsf{P}_{\mathit{fin}}(\omega),i\in\omega$.

18.5.7. Show Theorem 18.4.23 using the mapping $\mathsf{m}:\mathsf{P}_{\mathit{fin}}(\mathsf{Pm})\to\Pi^{\mathsf{Pm}}$ defined as

$$\begin{array}{lcl}\mathsf{m}(\emptyset) & = & \mathtt{U},\\ \mathsf{m}(\{\omega\}) & = & \omega,\\ \mathsf{m}(u\cup\{a\}) & = & \mathsf{m}(u)\cap\mathsf{m}(\{a\}),\\ \mathsf{m}(\{\langle u,v\rangle\}) & = & \mathsf{m}(u)\to\mathsf{m}(v).\end{array}$$

where $u,v\in\mathsf{P}_{\mathit{fin}}(\mathsf{Pm})$, $a\in\mathsf{Pm}$.

18.5.8. Show Theorem 18.4.16 using the mapping $\mathsf{m}:\mathsf{P}_{\mathit{fin}}(\mathsf{Em})\to\mathbb{T}^{\mathsf{Em}}$ defined as

$$\begin{array}{lcl}\mathsf{m}(\emptyset) & = & \mathtt{U},\\ \mathsf{m}(\{\mathsf{a}\}) & = & \mathsf{a},\\ \mathsf{m}(u\cup\{e\}) & = & \mathsf{m}(u)\cap\mathsf{m}(\{e\})\\ \mathsf{m}(\{\langle u,e\rangle\}) & = & \mathsf{m}(u)\to\mathsf{m}(\{e\}),\end{array}$$

where $u \in \mathsf{P}_{fin}(\mathsf{Em}), e \in \mathsf{Em}, \mathsf{a} \in \mathbb{A}_\infty$.

18.5.9. Let $\mathcal{T} \in \mathrm{TT}^{-\mathtt{U}}$. Prove the following statements.

1. The filter quasi λ-model $\mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, \cdot, [\![\ ]\!]^{\mathcal{F}^{\mathcal{T}}} \rangle$ is not a λ-model [Hint. Consider the constant function $\lambda x.y$.]
2. The filter quasi λ-model $\mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, \cdot, [\![\ ]\!]^{\mathcal{F}^{\mathcal{T}}} \rangle$ does not satisfy β-conversion.
3. The filter structure $\mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle$ is not reflexive.
4. Not all continuous functions are representable in $\mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle$.

Chapter 19

Advanced Properties

21.2.2008:897

This chapter proves some properties of intersection types in relation to terms and models.

Section 19.2 defines a realizability interpretation of types (Barendregt et al. [1983]). Types are interpreted as subsets of a domain of discourse D. Assuming a (partial) application $\cdot : D \times D \to D$, and a type environment ξ, i.e. a mapping from type atoms to subsets of D, we can define an interpretation $[\![A]\!]_\xi = [\![A]\!]^{\mathcal{T}}_\xi \subseteq D$ for each type A in $\mathbb{T}_\to$ by giving $\to$ the *realizability interpretation*, i.e.

$$\begin{array}{rcl} [\![\alpha]\!]_\xi & = & \xi(\alpha); \\ [\![A{\to}B]\!]_\xi & = & [\![A]\!]_\xi {\to} [\![B]\!]_\xi = \{d \in D \mid d \cdot [\![A]\!]_\xi \subseteq [\![B]\!]_\xi\}. \end{array}$$

This semantics, due to Scott [1975b], can be extended to intersection types by interpreting $\mathtt{U}$ as the domain of discourse and the intersection $\cap$ on types as set-theoretic intersection, i.e.

$$\begin{array}{rcl} [\![\mathtt{U}]\!]_\xi & = & D; \\ [\![A \cap B]\!]_\xi & = & [\![A]\!]_\xi \cap [\![B]\!]_\xi. \end{array}$$

The first requirement will be met by considering only ξ with $\xi(\mathtt{U}) = D$. Then, $\leq$ is interpreted as inclusion between sets. For interpreting λ-terms, it is enough to require that D is a quasi λ-model, depending on a valuation ρ of the term variables in D. One says that D satisfies $M : A$ under ρ, ξ, notation $D, \rho, \xi \models^{\mathcal{T}}_\cap M : A$ or simply $D, \rho, \xi \models M : A$, if the following holds

$$D, \rho, \xi \models^{\mathcal{T}}_\cap M : A \iff [\![M]\!]_\rho \in [\![A]\!]^{\mathcal{T}}_\xi.$$

Then Γ satisfies $M : A$, notation $\Gamma \models M : A$, is defined by

$$\Gamma \models M : A \iff \forall \rho, \xi. [D, \rho, \xi \models \Gamma \Rightarrow D, \rho, \xi \models M : A].$$

First soundness is proved, i.e.

$$\Gamma \vdash^{\mathcal{T}}_\cap M : A \Rightarrow \Gamma \models^{\mathcal{T}}_\cap M : A.$$

Completeness is the converse implication. Not all type theories satisfy completeness. We will prove that the natural type theories are exactly the ones that satisfy completeness.

$$[\Gamma \models^{\mathcal{T}}_{\cap} M : A \;\Rightarrow\; \Gamma \vdash^{\mathcal{T}}_{\cap} M : A] \text{ iff } \mathcal{T} \in \mathrm{NTT}^{\mathtt{U}}.$$

The proof of completness for a natural type theory $\mathcal{T}$ follows by taking $D = \mathcal{F}^{\mathcal{T}}$ where $\mathcal{F}^{\mathcal{T}}$ is the filter quasi λ-model over $\mathcal{T}$.

In Sections 19.1 and 19.4 intersection types will be used to characterize some properties of λ-terms. We consider the following properties (subsets) of λ-terms: strong normalization, normalization, head normalization and the persistent variants of these. A term has *persistently* property P if for all appropriate (to be defined in Section 19.1) arguments $\vec{N}$ the term $M\vec{N}$ has property P. The sets of untyped lambda terms having these properties are denoted respectively by $\mathsf{SN}, \mathsf{N}, \mathsf{HN}, \mathsf{PSN}, \mathsf{PN}, \mathsf{PHN}$. For a set of terms $X \subseteq \Lambda$ write

$$X{\uparrow}\beta = \{M \mid \exists N \in X. M \twoheadrightarrow_\beta N\}.$$

We denote by Γ_α the set of type declarations which associate α to all variables. For $\mathcal{T} \in \{\mathrm{Park}, \mathrm{CDZ}, \mathrm{HR}, \mathrm{DHM}, \mathrm{HL}\}$ Figure 19.1 defines a context $\mathsf{Ctx}^{\mathcal{T}}$ and a subset $\mathsf{Set}^{\mathcal{T}}(c)$ of lambda terms for each type atom $c \in \{0, 1, \mathtt{U}\}$.

$\mathcal{T}$	$\mathsf{Ctx}^{\mathcal{T}}$	$\mathsf{Set}^{\mathcal{T}}(0)$	$\mathsf{Set}^{\mathcal{T}}(1)$	$\mathsf{Set}^{\mathcal{T}}(\mathtt{U})$
Park	$\emptyset$	$\Lambda^{\emptyset}{\uparrow}\beta$	$-$	Λ
CDZ	Γ_0	PN	N	Λ
HR	Γ_1	$\emptyset$	$\Lambda^{\mathsf{I}}{\uparrow}\beta$	Λ
DHM	Γ_0	PHN	HN	Λ
HL	Γ_0	PSN	SN	$-$

Figure 19.1: Context and Set associated to 0, 1 and $\mathtt{U}$

We will show the following characterizations.

$$M \in \mathsf{Set}^{\mathcal{T}}(c) \text{ iff } \mathsf{Ctx}^{\mathcal{T}} \vdash^{\mathcal{T}} M : c.$$

The characterisations for $\mathtt{U}$ are immediate; those for 1 are given in Theorems 19.1.15 (CDZ, DHM, HL) and 19.4.10 (HR); those for 0 in Theorem 19.4.4 (Park), Lemma 19.4.7 (HR); the remaining ones can be found in Dezani-Ciancaglini et al. [2005b] (CDZ, DHM), and Tatsuta and Dezani-Ciancaglini [2006] (HL). Two methods are used to prove the characterizations, explaining why they are located in different sections.

1. Sections 19.1 uses the type interpretation defined in Section 19.2 and the standard technique of type stable sets.

2. Section 19.4 uses the Approximation Theorem presented in Section 19.3.

In Section 19.3, we introduce appropriate notions of approximants for almost all the type theories of Figure 15.3. Intuitively, an approximant is a partial term in the computation that does not contain redexes. In some cases, the approximants are obtained by replacing redexes by $\bot$ and in other cases by just freezing them. In case of Scott, CDZ, DHM and BCD, the whole context containing a redex in a head position is replaced by $\bot$. In case of AO, the notion of approximant is relaxed and abstractions are not replaced by $\bot$. In

case of Park and HR, the redexes are frozen by inserting a constant before the abstraction. We will show that a type can be derived for a term *if and only if* it can be derived for an approximant of that term (Approximation Theorem). A common and uniform proof is given of this theorem for all the type theories mentioned above (Dezani-Ciancaglini et al. [2001]). The proof technique used is a variant of stable sets over a Kripke applicative structure. In Section 19.4 some applications of the Approximation Theorem are given. Amongst these, the characterizations for Park and HL, mentioned in Figure 19.1.

Finally in Section 19.5 it will be shown that given Γ, A one cannot decide inhabitation for the type theory CDV, i.e. the existence of an M such that $\Gamma \vdash^{\mathrm{CDV}} M : A$. Also for BCD the inhabitation is undecidable, see Urzyczyn [1999]. On the other hand, in the type theory AO all types are inhabited, see Exercise 18.5.2, therefore inhabitation is decidable. The question of decidability of inhabitation for several other type theories remains open.

19.1. Characterizing syntactic properties

In this section we will see the intersection type systems at work in the characterization of properties of λ-terms. Since types are preserved by reduction, we can characterize only properties which induce equivalences that are preserved by reduction. In particular we will consider some normalization properties of λ-terms, i.e. the standard properties of having a head normal form or a normal form, and of being strongly normalizable. First we recall some basic definitions.

19.1.1. Definition. (i) A lambda term M is called *(β-)strongly normalizing* if there is no infinite (β-)reduction starting with M.

(ii) $\mathsf{SN} = \{M \mid M \text{ is strongly normalizing}\}$.

For example $\mathsf{SK} \in \mathsf{SN}$, but $\mathbf{\Omega}, \mathsf{SK}\mathbf{\Omega} \notin \mathsf{SN}$, even if the last term has a normal form.

19.1.2. Lemma (Raamsdonk and Severi [1995]). *The set* SN *is the smallest set of terms closed under the following rules.*

$$\frac{M_1 \in \mathsf{SN}, \dots, M_n \in \mathsf{SN}}{xM_1 \dots M_n \in \mathsf{SN}}{\scriptstyle n\geq 0}$$

$$\frac{M \in \mathsf{SN}}{\lambda x.M \in \mathsf{SN}}$$

$$\frac{M[x := N]M_1 \dots M_n \in \mathsf{SN} \qquad N \in \mathsf{SN}}{(\lambda x M)NM_1 \dots M_n \in \mathsf{SN}}{\scriptstyle n\geq 0}$$

Proof. Let $\mathcal{SN}$ be the set defined by these rules. We show

$$M \in \mathcal{SN} \iff M \in \mathsf{SN}.$$

($\Rightarrow$) By induction on the generation of $\mathcal{SN}$.

($\Leftarrow$) Suppose that M is strongly normalizing. Let $||M||$, the *norm* of M, be the length of the longest reduction path starting with M. We prove that

$M \in \mathcal{SN}$ by induction on the pair $(||M||, M)$, lexicographically ordered by the usual ordering on natural numbers and the subterm ordering. If M is a nf, then $M \in \mathcal{SN}$. In the case $||M|| = n > 0$, we have three cases, being $x\vec{M}, \lambda x.N$ or $(\lambda x.P)Q\vec{M}$. In the first two cases, the result follows by the induction hypothesis applied to subterms, where the norm is the same or has decreased. In the last case, the induction hypothesis is applied to $M[x := N]M_1 \dots M_n$ and N, where the norm strictly decreases. ■

19.1.3. DEFINITION. (i) A term M is *persistently head normalizing* iff $M\vec{N}$ has a head normal form for all terms $\vec{N}$.

(ii) A term M is *persistently normalizing* iff $M\vec{N}$ has a normal form for all normalizable terms $\vec{N}$.

(iii) A term M is *persistently strongly normalising* iff $M\vec{N}$ is strongly normalising for all strongly normalising terms $\vec{N}$.

The notion of persistently normalising terms has been introduced in Böhm and Dezani-Ciancaglini [1975].

19.1.4. NOTATION. Several classes of lambda terms are denoted by an acronym.

$$\begin{aligned}
\mathsf{HN} &= \{M \mid M \text{ has a head normal form}\}.\\
\mathsf{PHN} &= \{M \mid M \text{ is persistently head normalizing}\}.\\
\mathsf{N} &= \{M \mid M \text{ has a normal form}\}.\\
\mathsf{PN} &= \{M \mid M \text{ is persistently normalizing}\}.\\
\mathsf{SN} &= \{M \mid M \text{ is strongly normalizing}\}.\\
\mathsf{PSN} &= \{M \mid M \text{ is persistently strongly normalizing}\}.
\end{aligned}$$

The following inclusions follow immediately by definition, except the inclusions $\mathsf{PSN} \subseteq \mathsf{PN} \subseteq \mathsf{PHN}$, which are proved in Exercise 19.6.6. The inclusion $\mathsf{PSN} \subseteq \mathsf{PN}$ follows also comparing Figures 15.4 and 19.1. It is easy to find examples to show that all these inclusions are strict.

$$\begin{array}{ccccc}
\mathsf{SN} & \subset & \mathsf{N} & \subset & \mathsf{HN}\\
\cup & & \cup & & \cup\\
\mathsf{PSN} & \subset & \mathsf{PN} & \subset & \mathsf{PHN}
\end{array}$$

19.1.5. EXAMPLE. (i) $\mathsf{K}x\mathbf{\Omega} \in \mathsf{PN}$ but not in SN, hence not in PSN.

(ii) $(\lambda x.\mathsf{K}x\mathbf{\Omega}) \in \mathsf{N}$ but not in SN nor in PHN, hence not in PN.

(iii) $x\mathbf{\Omega} \in \mathsf{PHN}$ but not in N, hence not in PN.

(iv) $x \in \mathsf{PSN}$.

(v) $\mathsf{I} \in \mathsf{SN}$, but not in PNF.

Stable sets

We will use the standard proof technique of type stable sets (Krivine [1990]).

19.1.6. Definition. The open term model of the lambda calculus consists of open λ-terms modulo β-conversion. That is $\Lambda(\beta) = \langle \Lambda, \cdot, [\![\ ]\!]^\Lambda \rangle$, with

$$[\![M]\!]^\Lambda_\rho \quad = \quad \{N \mid N =_\beta M[\vec{x} := \rho(\vec{x})]\}, \qquad \text{where } \vec{x} = FV(M).$$

Then

$$[\![M]\!]^\Lambda_\rho \cdot [\![N]\!]^\Lambda_\rho = [\![MN]\!]^\Lambda_\rho.$$

19.1.7. Remark. In $\Lambda(\beta) = \langle \Lambda, \cdot, [\![\ ]\!]^\Lambda \rangle$ one has

$$X \to Y = \{M \in \Lambda \mid \forall N \in X\ MN \in Y\}.$$

19.1.8. Definition. Let $X \subseteq \Lambda$.
(i) X is called *closed under head expansion of redexes*, notation h$\uparrow$-closed, if

$$P[x := Q]\vec{R} \in X \text{ implies } (\lambda x.P)Q\vec{R} \in X.$$

The term Q is called the *argument* of the head expansion.
(ii) X is HN-*stable* if $X \subseteq \mathsf{HN}$, it contains $x\vec{M}$ for all $\vec{M} \in \Lambda$ and is h$\uparrow$-closed.
(iii) X is N-*stable* if $X \subseteq \mathsf{N}$, it contains $x\vec{M}$ for all $\vec{M} \in \mathsf{N}$ and is h$\uparrow$-closed.
(iv) A set X is SN-*stable* if $X \subseteq \mathsf{SN}$, it contains $x\vec{M}$ for all $\vec{M} \in \mathsf{SN}$ and is closed under head expansion of redexes, whose arguments are in SN.

From the above definition and Lemma 19.1.2 we easily get the following.

19.1.9. Proposition. *Let* $\mathsf{S} \in \{\mathsf{HN}, \mathsf{N}, \mathsf{SN}\}$ *and* $X, Y \subseteq \Lambda$.
(i) S *is* S-*stable.*
(ii) PHN *is* HN-*stable and* PN *is* N-*stable.*
(iii) *If* X, Y *are* S-*stable, then* $(X \to Y)$ *and* $(X \cap Y)$ *are* S-*stable.*
(iv) *If* Y *is* HN-*stable and* $X \neq \emptyset$, *then* $(X \to Y)$ *is* HN-*stable.* ■

19.1.10. Definition (Type environments). (i) The *type environment* $\xi = \xi^1_{\mathrm{BCD}} : \mathbb{A}_\infty \to \mathcal{P}(\Lambda)$ in the open term model $\mathcal{M}(\beta)$ is defined as follows.

$$\xi(\alpha) \quad = \quad \mathsf{HN}, \qquad \text{if } \alpha \in \mathbb{A}_\infty.$$

(ii) The *type environment* $\xi = \xi^2_{\mathrm{BCD}}$ in $\mathcal{M}(\beta)$ is defined as follows.

$$\xi(\alpha) \quad = \quad \mathsf{N}, \qquad \text{if } \alpha \in \mathbb{A}_\infty.$$

(iii) The *type environment* $\xi = \xi_{\mathrm{DHM}}$ in $\mathcal{M}(\beta)$ is defined as follows.

$$\begin{aligned} \xi(0) &= \mathsf{PHN}; \\ \xi(1) &= \mathsf{HN}. \end{aligned}$$

(iv) The *type environment* $\xi = \xi_{\mathrm{CDZ}}$ in $\mathcal{M}(\beta)$ is defined as follows.

$$\begin{aligned} \xi(0) &= \mathsf{PN}; \\ \xi(1) &= \mathsf{N}. \end{aligned}$$

(v) The *type environment* $\xi = \xi_{\mathrm{CDV}}$ in $\mathcal{M}(\beta)$ is defined as follows.

$$\xi(\alpha) \;=\; \mathsf{SN}, \qquad \text{if } \alpha \in \mathbb{A}_\infty.$$

(vi) The *type environment* $\xi = \xi_{\mathrm{HL}}$ in $\mathcal{M}(\beta)$ is defined as follows.

$$\begin{aligned} \xi(0) &= \mathsf{PSN}; \\ \xi(1) &= \mathsf{SN}. \end{aligned}$$

19.1.11. LEMMA. (i) $[\![A]\!]_{\xi^1_{\mathrm{BCD}}}$ *and* $[\![A]\!]_{\xi_{\mathrm{DHM}}}$ *are* HN*-stable.*
(ii) $[\![A]\!]_{\xi^2_{\mathrm{BCD}}}$ *and* $[\![A]\!]_{\xi_{\mathrm{CDZ}}}$ *are* N*-stable.*
(iii) $[\![A]\!]_{\xi_{\mathrm{CDV}}}$ *and* $[\![A]\!]_{\xi_{\mathrm{HL}}}$ *are* SN*-stable.*

PROOF. All points follow easily from Proposition 19.1.9. ■

We shall show that for each type environment $\xi_{\mathcal{T}}$ of Definition 19.1.10 $(\Lambda(\beta), \xi_{\mathcal{T}})$ are $\mathcal{T}$-good and preserve $\leq_{\mathcal{T}}$. The proof occupies 19.1.12-19.1.14.

19.1.12. LEMMA. (i) $M \in \mathsf{SN}, N \in \mathsf{PSN} \;\Rightarrow\; M[x := N] \in \mathsf{SN}$.
(ii) $M \in \mathsf{SN}, N \in \mathsf{PSN} \;\Rightarrow\; MN \in \mathsf{SN}$.
(iii) $M \in \mathsf{N}, N \in \mathsf{PN} \;\Rightarrow\; M[x := N] \in \mathsf{N}$.
(iv) $M \in \mathsf{N}, N \in \mathsf{PN} \;\Rightarrow\; MN \in \mathsf{N}$.
(v) $M \in \mathsf{HN}, N \in \mathsf{PHN} \;\Rightarrow\; M[x := N] \in \mathsf{N}$.
(vi) $M \in \mathsf{HN}, N \in \mathsf{PHN} \;\Rightarrow\; MN \in \mathsf{HN}$.

PROOF. The first two statements follow using the inductive definition of SN given in 19.1.2. The rest follow by an easy induction on the (head) normal form of M. ■

19.1.13. PROPOSITION. (i) $\mathsf{PSN} = (\mathsf{N} \mapsto \mathsf{PSN})$.
(ii) $\mathsf{SN} = (\mathsf{PSN} \mapsto \mathsf{SN})$.
(iii) $\mathsf{PN} = (\mathsf{N} \mapsto \mathsf{PN})$.
(iv) $\mathsf{N} = (\mathsf{PN} \mapsto \mathsf{N})$.
(v) $\mathsf{PHN} = (\mathsf{HN} \mapsto \mathsf{PHN})$.
(vi) $\mathsf{HN} = (\mathsf{PHN} \mapsto \mathsf{HN})$.

PROOF. All cases are immediate except the inclusions $\mathsf{SN} \subseteq (\mathsf{PSN} \mapsto \mathsf{SN})$, $\mathsf{N} \subseteq (\mathsf{PN} \mapsto \mathsf{N})$ and $\mathsf{HN} \subseteq (\mathsf{PHN} \mapsto \mathsf{HN})$. These follow easily from Lemma 19.1.12 (ii), (iv) and (vi). ■

19.1.14. LEMMA. *For all* $\xi_{\mathcal{T}}$ *of Definition 19.1.10 we have the following.*
(i) $\forall N \in [\![B]\!]_{\xi_{\mathcal{T}}}, M[x := N] \in [\![A]\!]_{\xi_{\mathcal{T}}}$ *implies* $(\lambda x.M) \in [\![B \to A]\!]_{\xi_{\mathcal{T}}}$.
(ii) $A \leq_{\mathcal{T}} B \;\Rightarrow\; [\![A]\!]_{\xi_{\mathcal{T}}} \subseteq [\![B]\!]_{\xi_{\mathcal{T}}}$.
I.e. for all $\xi_{\mathcal{T}}$ *of Definition 19.1.10* $(\Lambda(\beta), \xi_{\mathcal{T}})$ *are* $\to$*-good and preserve* $\leq_{\mathcal{T}}$.

PROOF. (i) If either $\mathcal{T} \neq \mathrm{CDV}$ or $\mathcal{T} = \mathrm{CDV}$ and $N \in \mathsf{SN}$ one easily shows that $M[x := N] \in [\![A]\!]_{\xi_{\mathcal{T}}}$ implies $(\lambda x.M)N \in [\![A]\!]_{\xi_{\mathcal{T}}}$ by induction on A using Proposition 19.1.9. The conclusion from the definition of $\to$.

(ii) By induction on the generation of $\leq_{\mathcal{T}}$, using Proposition 19.1.13. ■

In the following result several important syntactic properties of lambda terms are characterized by typability with respect to some intersection type theory. We define $\Gamma_0^M = \{x_1\, 0, \ldots, x_n\, 0\}$, where $\{x_1, \ldots, x_n\} = \mathrm{FV}(M)$.

19.1.15. Theorem (Characterization Theorems).

(i) $M \in \mathsf{N} \iff \forall \mathcal{T} \in \mathrm{TT}^{\,\mathtt{U}}\, \exists \Gamma, A. \Gamma \vdash^{\mathcal{T}}_{\cap} M : A \;\&\; \mathtt{U} \notin \Gamma, A$

$\iff \exists \Gamma, A. \Gamma \vdash^{\mathrm{BCD}}_{\cap} M : A \;\&\; \mathtt{U} \notin \Gamma, A.$

$\iff \Gamma_0^M \vdash^{\mathrm{CDZ}}_{\cap} M : 1.$

(ii) $M \in \mathsf{HN} \iff \forall \mathcal{T} \in \mathrm{TT}^{\,\mathtt{U}}\, \exists \Gamma\, \exists n, m \in \mathbb{N}. \Gamma \vdash^{\mathcal{T}}_{\cap} M : (\mathtt{U}^m \to A)^n \to A$

$\iff \exists \Gamma, A. \Gamma \vdash^{\mathrm{BCD}}_{\cap} M : A \;\&\; A \neq_{\mathrm{BCD}} \mathtt{U}$

$\iff \Gamma_0^M \vdash^{\mathrm{DHM}}_{\cap} M : 1.$

(iii) $M \in \mathsf{SN} \iff \forall \mathcal{T} \in \mathrm{TT}\, \exists \Gamma, A. \Gamma \vdash^{\mathcal{T}}_{\cap} M : A.$

$\iff \exists \Gamma, A. \Gamma \vdash^{\mathrm{CDV}}_{\cap} M : A.$

$\iff \Gamma_0^M \vdash^{\mathrm{HL}}_{\cap} M : 1.$

Proof. We first prove ($\Rightarrow$) for (i)-(iii).

(i) By Corollary 16.2.5(ii) it suffices to consider M in normal form. The proof is by induction on M.

For the first and second equivalence, the only interesting case is $M \equiv x\vec{M}$, where $\vec{M} \equiv M_1 \ldots M_m$. By the induction hypothesis we have $\Gamma_j \vdash^{\mathcal{T}} M_j : A_j$, for some Γ_j, A_j not containing $\mathtt{U}$ and for $j \leq m$. This implies that

$$\uplus_{j \leq m} \Gamma_j \uplus \{x\, A_1 \to \ldots \to A_m \to A\} \vdash^{\mathcal{T}}_{\cap} x\vec{M} : A.$$

Therefore, $\forall \mathcal{T} \in \mathrm{TT}\, \exists \Gamma, A. \Gamma \vdash^{\mathcal{T}}_{\cap} M : A \;\&\; \mathtt{U} \notin \Gamma, A$ in particular for $\mathcal{T} = \mathrm{BCD}$.

For $\lambda^{\mathrm{CDZ}}_{\cap}$ we also show by induction on M in normal form that $\Gamma_0^M \vdash M : 1$. If $M \equiv x\vec{M}$ then $\Gamma_0^M \vdash^{\mathrm{CDZ}}_{\cap} M_j : 1$ by the induction hypothesis and weakening. As $0 = 1 \to 0$ in CDZ, this implies $\Gamma_0^M \vdash^{\mathrm{CDZ}}_{\cap} x\vec{M} : 0$. By rule ($\leq_{\mathrm{CDZ}}$) we conclude $\Gamma_0^M \vdash^{\mathrm{CDZ}}_{\cap} M : 1$.

If $M \equiv \lambda y.N$ then by the induction hypothesis we have $\Gamma_0^M, y : 0 \vdash^{\mathrm{CDZ}}_{\cap} N : 1$ and this implies $\Gamma_0^M \vdash^{\mathrm{CDZ}}_{\cap} M : 0 \to 1$. Hence $\Gamma_0^M \vdash^{\mathrm{CDZ}}_{\cap} M : 1$ by rule ($\leq_{\mathrm{CDZ}}$).

(ii) Again assume $M \equiv \lambda x_1 \ldots x_n.xM_1 \ldots M_m$ is in head normal form.

$x\,(\mathtt{U}^m \to A) \;\vdash^{\mathcal{T}}_{\cap}\; xM_1 \ldots M_m : A,$ by ($\to$E), hence

$x\,(\mathtt{U}^m \to A) \;\vdash^{\mathcal{T}}_{\cap}\; M : (\mathtt{U}^m \to A)^n \to A,$ by (*weakening*) and ($\to$I).

As A is arbitrary we can get the type $\neq \mathtt{U}$ in $\mathcal{T} = \mathrm{BCD}$. We get

$$x\,(\mathtt{U}^m \to 0) \vdash^{\mathrm{DHM}}_{\cap} M : (\mathtt{U}^m \to 0)^n \to 0,$$

taking $\mathcal{T} = \mathrm{DHM}$ and $A \equiv 0$. This implies

$$x\,0 \;\vdash^{\mathrm{DHM}}_{\cap}\; M : 1,$$

using $(\mathtt{U}^m \to 0) =_{\mathrm{DHM}} 0$, as $(\mathtt{U} \to 0) =_{\mathrm{DHM}} 0$, and $((\mathtt{U}^m \to 0)^n \to 0) \leq_{\mathrm{DHM}} 1$, as $0 \leq_{\mathrm{DHM}} 1$ and $1 =_{\mathrm{DHM}} 0 \to 1$.

(iii) By induction on the structure of strongly normalizing terms following Definition 19.1.2. We only consider the case $M \equiv (\lambda x.R)N\vec{M}$ with $\vec{M} \equiv M_1 \dots M_n$ and both $R[x := N]\vec{M}$ and N are strongly normalizing. By the induction hypothesis there are Γ, A, Γ', B such that $\Gamma \vdash^{\mathcal{T}}_{\cap} R[x := N]\vec{M} : A$ and $\Gamma' \vdash^{\mathcal{T}}_{\cap} N : B$. We get $\Gamma \uplus \Gamma' \vdash^{\mathcal{T}}_{\cap} R[x := N]\vec{M} : A$ and $\Gamma \uplus \Gamma' \vdash^{\mathcal{T}}_{\cap} N : B$, so if $n = 0$ we are done by Theorem 16.2.4(i). If $n > 0$, then by iterated applications of Theorem 16.1.1(ii) to $\Gamma \vdash^{\mathcal{T}}_{\cap} R[x := N]\vec{M} : A$ we obtain

$$\Gamma \vdash^{\mathcal{T}}_{\cap} R[x := N] : B^{(i)}_1 \to \dots \to B^{(i)}_n \to B^{(i)} \quad \Gamma \vdash^{\mathcal{T}}_{\cap} M_j : B^{(i)}_j,\ (j \leq n)$$

and $\bigcap_{i \in I} B^{(i)} \leq_{\mathcal{T}} A$ for some $I, B^{(i)}_j (j \leq n), B^{(i)} \in \mathbb{TT}^{\mathcal{T}}$. As in case $n = 0$ we obtain $\Gamma \uplus \Gamma' \vdash^{\mathcal{T}}_{\cap} (\lambda x.R)N : B^{(i)}_1 \to \dots \to B^{(i)}_m \to B^{(i)}$. So we can conclude $\Gamma \uplus \Gamma' \vdash^{\mathcal{T}}_{\cap} (\lambda x.R)N\vec{M} : A$. Finally, $\Gamma^M_0 \vdash^{\mathrm{HL}}_{\cap} M : 1$ follows from the observation that 1 is the top and 0 the bottom element in HL, see Lemma 15.1.22(i).

($\Leftarrow$) Now we show the converse implication. Let $\rho_0(x) = x$ for all $x \in \mathrm{Var}$.

(i) Suppose $\Gamma \vdash^{\mathrm{BCD}}_{\cap} M : A$ and $\mathtt{U} \notin A, \Gamma$. By soundness (Theorem 19.2.4) it follows that $\Gamma \models^{\mathrm{BCD}}_{\cap} M : A$. By Lemmas 19.1.14 and 19.1.11(ii) one has $\Lambda(\beta), \rho_0, \xi^2_{\mathrm{BCD}} \models \Gamma$. Hence, $M \in [\![A]\!]_{\xi^2_{\mathrm{BCD}}} \subseteq \mathsf{N}$, again by Lemma 19.1.11(ii).

Suppose $\Gamma^M_0 \vdash^{\mathrm{CDZ}}_{\cap} M : 1$. By soundness it follows that $\Gamma^M_0 \models^{\mathrm{CDZ}}_{\cap} M : 1$. By Lemmas 19.1.14 and 19.1.11(ii) one has $\Lambda(\beta), \rho_0, \xi_{\mathrm{CDZ}} \models \Gamma$. Hence, $M \in [\![1]\!]_{\xi_{\mathrm{CDZ}}} = \mathsf{N}$, by Definition 19.1.10(iv).

(ii) Suppose $\Gamma \vdash^{\mathrm{BCD}}_{\cap} M : A \neq \mathtt{U}$. Then $\Gamma \models^{\mathrm{BCD}}_{\cap} M : A$ by soundness. By Lemmas 19.1.14 and 19.1.11(i) one has $\Lambda(\beta), \rho_0, \xi^1_{\mathrm{BCD}} \models \Gamma$. Therefore we have $M \in [\![A]\!]_{\xi^1_{\mathrm{BCD}}} \subseteq \mathsf{HN}$, again by Lemma 19.1.11(ii).

Suppose $\Gamma^M_0 \vdash^{\mathrm{DHM}}_{\cap} M : 1$. Again by soundness $\Gamma^M_0 \models^{\mathrm{DHM}}_{\cap} M : 1$. By Lemmas 19.1.14 and 19.1.11(i) one has $\Lambda(\beta), \rho_0, \xi_{\mathrm{DHM}} \models \Gamma$. Hence, by Definition 19.1.10(iii) one has $M \in [\![1]\!]_{\xi_{\mathrm{DHM}}} = \mathsf{HN}$.

(iii) Let $\Gamma \vdash^{\mathrm{CDV}}_{\cap} M : A$. Again by soundness $\Gamma \models^{\mathrm{CDV}}_{\cap} M : A$. By Lemmas 19.1.14 and 19.1.11(iii) one has $\Lambda(\beta), \rho_0, \xi_{\mathrm{CDV}} \models \Gamma$. Hence $M \in [\![A]\!]_{\xi_{\mathrm{CDV}}} \subseteq \mathsf{SN}$, by Lemma 19.1.11(iii). ■

19.1.16. Remark. (i) For $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$ one has

$$\exists A, \Gamma. \Gamma \vdash^{\mathcal{T}}_{\cap} M : A \ \&\ \mathtt{U} \neq_{\mathcal{T}} A, \Gamma \not\Rightarrow M \in \mathsf{HN}.$$

Take for example $\mathcal{T} = \mathrm{Park}$, then $\vdash^{\mathrm{Park}}_{\cap} \mathbf{\Omega} : 0 \neq_{\mathrm{Park}} \mathtt{U}$, by Theorem 19.4.4, but this term is unsolvable, hence without hnf, see Barendregt [1984], so $M \notin \mathsf{HN}$.

(ii) There are many proofs of Point Theorem 19.1.15(iii) in the literature: Pottinger [1981], Leivant [1986], van Bakel [1992], Krivine [1990], Ghilezan [1996], Amadio and Curien [1998]. As observed in Venneri [1996] all but Amadio and Curien [1998] contain some bugs, which in Krivine [1990] can be easily remedied with a suitable non-standard notion of length of reduction path.

(iii) In Coppo et al. [1987] persistently normalizing normal forms have been given a similar characterization using the notion of replaceable variable (Coppo

et al. [1987]). Other classes of terms are characterized in Dezani-Ciancaglini et al. [2005a] and Tatsuta and Dezani-Ciancaglini [2006].

19.2. Realizability interpretation of types

The natural set-theoretic semantics for type assignment in $\lambda_{\to}$ based on untyped λ-models is given in Scott [1975b] where it was shown that

$$\Gamma \vdash_{\lambda_{\to}} M : A \;\Rightarrow\; \Gamma \models M : A.$$

Scott asked whether the converse (completeness) holds. In Barendregt et al. [1983] the notion of semantics was extended to intersection types and completeness was proved for $\lambda_{\cap}^{\mathrm{BCD}}$ via the corresponding filter model. Completeness for $\lambda_{\to}$ follows by a conservativity result. In Hindley [1983] an alternative proof of completeness for $\lambda_{\to}$ was given directly, using a term model. Variations of the semantics are presented in Dezani-Ciancaglini et al. [2003].
Recall that quasi λ-models are defined in 18.1.2, with (β) being not required.

19.2.1. Definition (Type Interpretation). Let $D = \langle D, \cdot, [\![\,]\!]^D \rangle$ be a quasi λ-model and let $\mathcal{T}$ be an intersection type theory over the atoms $\mathbb{A}^{\mathcal{T}}$.

(i) For $X, Y \subseteq D$ define

$$X{\to}Y = \{d \in D \mid \forall e \in X. d \cdot e \in Y\}.$$

(ii) The *type interpretation* induced by the type environment $\xi : \mathbb{A}^{\mathcal{T}} \to \mathsf{P}(D)$ with $\xi(\mathtt{U}) = D$ is the map $[\![\,]\!]^D_\xi : \mathbb{T}^{\mathcal{T}} \to \mathsf{P}(D)$ defined as follows.

(1) $[\![\alpha]\!]^D_\xi = \xi(\alpha)$;
(2) $[\![A{\to}B]\!]^D_\xi = [\![A]\!]^D_\xi {\to} [\![B]\!]^D_\xi$;
(3) $[\![A \cap B]\!]^D_\xi = [\![A]\!]^D_\xi \cap [\![B]\!]^D_\xi$.

The above definition is the extension to intersection-types of the *simple semantics* for simple types of Scott [1975b], generalized by allowing D to be just a quasi λ-model instead of a λ-model.

It is easy to verify that $[\![\mathtt{U}{\to}\mathtt{U}]\!]^D_\xi = D$ for all D, ξ.

In order to prove soundness, we have to check that the interpretation preserves the typability rules. We already know that the interpretation preserves the typing rules for application and intersection. This is because $\cap$ is interpreted as intersection on sets and $\to$ is interpreted as the arrow induced by the application $\cdot$ on D. The following definition is necessary to require that the interpretation preserves the remaining two typability rules: abstraction and subtyping.

19.2.2. Definition. Let $D = \langle D, \cdot, [\![\,]\!]^D \rangle$ be a quasi λ-model and $\xi : \mathbb{A}^{\mathcal{T}} \to \mathsf{P}(D)$ be a type environment.

(i) The pair (D, ξ) is *$\to$-good* if for all $A, B \in \mathbb{T}^{\mathcal{T}}$ for all environments ρ, terms M and variables x

$$[\forall d \in [\![A]\!]^D_\xi .\, [\![M]\!]^D_{\rho[x:=d]} \in [\![B]\!]^D_\xi] \;\Rightarrow\; [\![\lambda x.M]\!]^D_\rho \in [\![A]\!]^D_\xi {\to} [\![B]\!]^D_\xi;$$

(ii) The pair (D,ξ) *preserves* $\leq_{\mathcal{T}}$ *iff for all* $A,B\in\mathbb{T}^{\mathcal{T}}$*:*

$$A\leq_{\mathcal{T}} B \;\Rightarrow\; [\![A]\!]^D_\xi\subseteq[\![B]\!]^D_\xi.$$

We now introduce the semantics of type assignment.

19.2.3. Definition (Semantic Satisfiability). Let $\mathcal{T}\in\mathrm{TT}^{\mathtt{U}}$.

(i) Let $D=\langle D,\cdot,[\![\ ]\!]^D\rangle$ be a quasi λ-model. Define

$$\begin{array}{lll} D,\rho,\xi\models M:A & \iff & [\![M]\!]^D_\rho\in[\![A]\!]^D_\xi;\\ D,\rho,\xi\models\Gamma & \iff & D,\rho,\xi\models x:B, \text{for all } (x\ B)\in\Gamma.\end{array}$$

(ii) We say that Γ satisfies $M:A$, notation $\Gamma\models^{\mathcal{T}}_{\cap} M:A$, iff

$$D,\rho,\xi\models\Gamma\;\Rightarrow\; D,\rho,\xi\models M:A,$$

for all D,ξ,ρ such that (D,ξ) is $\to$-good and preserves $\leq_{\mathcal{T}}$.

Derivability in the type system implies semantic satisfiability, as shown in the next theorem.

19.2.4. Theorem (Soundness). *For all* $\mathcal{T}\in\mathrm{TT}^{\mathtt{U}}$ *one has*

$$\Gamma\vdash^{\mathcal{T}}_{\cap} M:A\;\Rightarrow\;\Gamma\models^{\mathcal{T}}_{\cap} M:A.$$

Proof. By induction on the derivation of $\Gamma\vdash^{\mathcal{T}}_{\cap} M:A$. Rules ($\to$E), ($\cap$I) and ($\mathtt{U}$) are sound by the definition of type interpretation (Definition 19.2.1).

As to the soundness of rule ($\to$I), assume $\Gamma,x\ A\vdash^{\mathcal{T}}_{\cap} M:B$ in order to show $\Gamma\models^{\mathcal{T}}(\lambda x.M):(A\to B)$. Assuming $D,\rho,\xi\models\Gamma$ we have to show

$$[\![\lambda x.M]\!]^D_\rho\in[\![A]\!]^D_\xi\to[\![B]\!]^D_\xi.$$

Let $d\in[\![A]\!]^D_\xi$. We are done if we can show

$$[\![M]\!]^D_{\rho[x=d]}\in[\![B]\!]^D_\xi,$$

because (D,ξ) are $\to$-good. Now $D,\rho[x=d],\xi\models\Gamma,xA$, hence $[\![M]\!]^D_{\rho[x=d]}\in[\![B]\!]^D_\xi$, by the induction hypothesis for $\Gamma,x\ A\vdash^{\mathcal{T}}_{\cap} M:B$.

Rule ($\leq$) is sound, as we consider only (D,ξ) that preserve $\leq_{\mathcal{T}}$. ■

Completeness

Now we characterize the complete theories.

Notation. Let $\mathcal{T}\in\mathrm{NTT}^{\mathtt{U}}$ and $\mathcal{F}^{\mathcal{T}}=\langle\mathcal{F}^{\mathcal{T}},\cdot,[\![\ ]\!]^{\mathcal{F}^{\mathcal{T}}}\rangle$ its corresponding filter quasi λ-model, see Definition 18.2.1.

(i) Let $\xi^{\mathcal{T}}:\mathbb{A}^{\mathcal{T}}\to\mathsf{P}(\mathcal{F}^{\mathcal{T}})$ be the type environment defined by

$$\xi^{\mathcal{T}}(\alpha)=\{X\in\mathcal{F}^{\mathcal{T}}\mid\alpha\in X\}.$$

(ii) Let $[\![\ ]\!]^{\mathcal{T}} : \mathbb{T}^{\mathcal{T}} \to \mathsf{P}(\mathcal{F}^{\mathcal{T}})$ be the mapping $[\![\ ]\!]^{\mathcal{F}^{\mathcal{T}}}_{\xi^{\mathcal{T}}}$.

The mapping $[\![\ ]\!]^{\mathcal{T}} : \mathbb{T}^{\mathcal{T}} \to \mathsf{P}(\mathcal{F}^{\mathcal{T}})$ turns out to have the property of associating to each type A the set of filters which contain A (thus preserving the property which defines $\xi^{\mathcal{T}}$ in the basic case of type atoms).

19.2.5. Proposition. *Let $\mathcal{T} \in \mathrm{NTT}^{\mathtt{U}}$. Then we have*

$$[\![A]\!]^{\mathcal{T}} \quad = \quad \{X \in \mathcal{F}^{\mathcal{T}} \mid A \in X\}.$$

Proof. By induction on A. The only interesting case is when A is an arrow type. If $A \equiv B \to C$ we have

$$\begin{array}{lcll}
[\![B \to C]\!]^{\mathcal{T}} & = & \{X \in \mathcal{F}^{\mathcal{T}} \mid \forall Y \in [\![B]\!]^{\mathcal{T}}.\ X \cdot Y \in [\![C]\!]^{\mathcal{T}}\}, & \text{by definition,} \\
& = & \{X \in \mathcal{F}^{\mathcal{T}} \mid \forall Y.\ B \in Y \Rightarrow C \in X \cdot Y\}, & \text{by induction,} \\
& = & \{X \in \mathcal{F}^{\mathcal{T}} \mid C \in X \cdot \uparrow B\}, & \text{by monotonicity,} \\
& = & \{X \in \mathcal{F}^{\mathcal{T}} \mid C \in \uparrow\{C' \mid \exists B' \in \uparrow B.B' \to C' \in X\}\}, & \text{by the definition of} \\
& & & \text{filter application,} \\
& = & \{X \in \mathcal{F}^{\mathcal{T}} \mid B \to C \in X\}, & \text{by } (\to) \text{ and } (\mathtt{U} \to). \ \blacksquare
\end{array}$$

19.2.6. Lemma. *Let $\mathcal{T} \in \mathrm{NTT}^{\mathtt{U}}$. Then $(\mathcal{F}^{\mathcal{T}}, \xi^{\mathcal{T}})$ is $\to$-good and preserves $\leq_{\mathcal{T}}$.*

Proof. Suppose that $X \in [\![A]\!]^{\mathcal{T}}$ is such that

$$[\![M]\!]^{\mathcal{T}}_{\rho[x:=X]} \in [\![B]\!]^{\mathcal{T}},$$

in order to show $[\![\lambda x.M]\!]^{\mathcal{T}}_{\rho} \cdot X \in [\![B]\!]^{\mathcal{T}}$, which establishes that $(\mathcal{F}^{\mathcal{T}}, \xi^{\mathcal{T}})$ is $\to$-good. By Proposition 19.2.5 we have $B \in [\![M]\!]^{\mathcal{T}}_{\rho[x:=X]}$, hence $B \in f(X)$, where we have put $f = \lambda\!\!\lambda d.[\![M]\!]^{\mathcal{T}}_{\rho[x:=d]}$. Since by Lemma 17.2.9(ii) one has $f \sqsubseteq F^{\mathcal{T}}(G^{\mathcal{T}}(f))$, it follows that $B \in F^{\mathcal{T}}(G^{\mathcal{T}}(f))(X)$. Hence $[\![\lambda x.M]\!]^{\mathcal{T}}_{\rho} \cdot X = F^{\mathcal{T}}(G^{\mathcal{T}}(f))(X) \in [\![B]\!]^{\mathcal{T}}$, by Definition 18.1.4(i) and Proposition 19.2.5.

As an immediate consequence of the Proposition 19.2.5 we get

$$A \leq_{\mathcal{T}} B \Leftrightarrow \forall X \in \mathcal{F}^{\mathcal{T}}.[A \in X \Rightarrow B \in X] \Leftrightarrow [\![A]\!]^{\mathcal{T}} \subseteq [\![B]\!]^{\mathcal{T}}_{,}$$

and therefore $(\mathcal{F}^{\mathcal{T}}, \xi^{\mathcal{T}})$ preserves $\leq_{\mathcal{T}}$. ■

Now we can prove the desired completeness result.

19.2.7. Theorem. *(Completeness) Let $\mathcal{T} \in \mathrm{TT}^{\mathtt{U}}$.*
(i) $[\Gamma \models^{\mathcal{T}}_{\cap} M : A \ \Rightarrow\ \Gamma \vdash^{\mathcal{T}}_{\cap} M : A]$ *iff* $\mathcal{T} \in \mathrm{NTT}^{\mathtt{U}}$.
(ii) *Let $\mathcal{T} \in \mathrm{NTT}^{\mathtt{U}}$. Then*

$$\Gamma \models^{\mathcal{T}}_{\cap} M : A \iff \Gamma \vdash^{\mathcal{T}}_{\cap} M : A.$$

Proof. (i) ($\Rightarrow$) It is easy to verify that all type interpretations validate rule $(\to)$ and the axioms $(\to\cap)$, and $(\mathtt{U}_{\mathrm{top}})$. As to axiom $(\to\cap)$, consider the $\mathcal{T}$-basis $\Gamma = \{x\ (A \to B) \cap (A \to C)\}$. From Definition 19.2.1 we get

$$\Gamma \models^{\mathcal{T}}_{\cap} x : A \to (B \cap C).$$

Hence, by hypothesis, we have $\Gamma \vdash^{\mathcal{T}}_{\cap} x : A{\to}(B \cap C)$. Using Theorem 16.1.9(i) it follows that $(A{\to}B){\cap}(A{\to}C) \leq_{\mathcal{T}} A{\to}B \cap C$. Therefore axiom $(\to\cap)$ holds.

As to axiom $(\mathtt{U} \to)$

$$\begin{array}{lcll} \models^{\mathcal{T}}_{\cap} x : \mathtt{U} \to \mathtt{U} & \Rightarrow & \vdash^{\mathcal{T}}_{\cap} x : (\mathtt{U}{\to}\mathtt{U}) & \\ & \Rightarrow & x\,\mathtt{U} \vdash^{\mathcal{T}}_{\cap} x : (\mathtt{U}{\to}\mathtt{U}) & \\ & \Rightarrow & \mathtt{U} \leq_{\mathcal{T}} (\mathtt{U}{\to}\mathtt{U}), & \text{by Theorem 16.1.9(i).} \end{array}$$

This proves $(\Rightarrow)$.

$(\Leftarrow)$ Now suppose $\Gamma \models^{\mathcal{T}} M : A$ towards $\Gamma \vdash^{\mathcal{T}}_{\cap} M : A$. We use the filter quasi λ-model $\langle \mathcal{F}^{\mathcal{T}}, \cdot, [\![\ ]\!]^{\mathcal{T}} \rangle$. By Lemma 19.2.6 we have that $\Gamma \models^{\mathcal{T}}_{\cap} M : A$ implies $[\![M]\!]^{\mathcal{T}}_{\rho_\Gamma} \in [\![A]\!]^{\mathcal{T}}$, where

$$\rho_\Gamma(x) = \begin{cases} \uparrow A & \text{if } x\,A \in \Gamma, \\ \uparrow \mathtt{U} & \text{otherwise.} \end{cases}$$

We conclude $\Gamma \vdash^{\mathcal{T}}_{\cap} M : A$, using Proposition 19.2.5 and Theorem 18.2.7(i).

(ii) By Proposition 19.2.4 and (i). ■

Similar results for $\mathcal{T} \in \mathrm{PTT}$ can be found in Dezani-Ciancaglini et al. [2003]. See also Exercises 19.6.3 and 19.6.4.

19.3. Approximation theorems

Crucial results for the study of the equational theory of ω-algebraic λ-models are the *Approximation Theorems*, see e.g. Hyland [1975/76], Wadsworth [1976], Barendregt [1984], Longo [1987], Ronchi Della Rocca [1988], Honsell and Ronchi Della Rocca [1992]. An Approximation Theorem expresses the interpretation of any λ-term, even a non terminating one, as the supremum of the interpretations of suitable *normal forms*, called the *approximants* of the term, in an appropriate *extended language*. Approximation Theorems are very useful in proving, for instance, *Computational Adequacy* of models with respect to *operational semantics*, see e.g. Barendregt [1984], Honsell and Ronchi Della Rocca [1992]. There are other possible methods of showing computational adequacy, both semantical and syntactical, e.g. Hyland [1975/76], Wadsworth [1976], Honsell and Ronchi Della Rocca [1992], Abramsky and Ong [1993], but the method based on Approximation Theorems is usually the most straightforward. However, proving an Approximation Theorem for a given model theory is usually rather difficult. Most of the proofs in the literature are based on the technique of *indexed reduction*, see Wadsworth [1976], Abramsky and Ong [1993], Honsell and Ronchi Della Rocca [1992]. However, when the model in question is a filter model, by applying duality, the Approximation Theorem can be rephrased as follows: the types of a given term are all and only the types of its approximants. This change in perspective opens the way to proving Approximation Theorems using the syntactical machinery of proof theory, such as *logical predicates* and *computability* techniques.

The aim of the present section is to show in a uniform way that all the type assignment systems which induce filter models isomorphic to the models

in Scott [1972], Park [1976], Coppo et al. [1987], Honsell and Ronchi Della Rocca [1992], Dezani-Ciancaglini et al. [2005a], Barendregt et al. [1983], Abramsky and Ong [1993] satisfy the Approximation Theorem. To this end following Dezani-Ciancaglini et al. [2001] we use a technique which can be constructed as a version of stable sets over a Kripke applicative structure. In Ronchi Della Rocca and Paolini [2004] approximation theorems are given also for the type assignment system $\lambda^{\mathrm{EHR}}_{\cap\mathtt{V}}$ defined in Definition 15.2.13.

For almost all the type theories of Figure 15.3 which induce λ-models we introduce appropriate notions of *approximants* which agree with the λ-theories of different models and therefore also with the type theories describing these models. Then we will prove that all types of an approximant of a given term (with respect to the appropriate notion of approximants) are also types of the given term. Finally we show the converse, namely that the types which can be assigned to a term can also be assigned to at least one approximant of that term. Hence a type can be derived for a term *if and only if* it can be derived for an approximant of that term. We end this section showing some applications of the Approximation Theorem.

Approximate normal forms

In this section we consider two extensions of λ-calculus both obtained by adding one constant. The first one is the well known language $\lambda\bot$, see Barendregt [1984]. The other extension is obtained by adding the constant Φ and is discussed in Honsell and Ronchi Della Rocca [1992].

19.3.1. Definition. (i) The set $\Lambda\bot$ of $\lambda\bot$-terms is obtained by adding the constant $\bot$ to the formation rules of terms.

(ii) The set $\Lambda\Phi$ of $\lambda\Phi$-terms is obtained by adding the constant Φ to the formation rules of terms.

We consider two mappings ($\Box_\bot$ and $\Box_L$) from λ-terms to $\lambda\bot$-terms and one mapping ($\Box_\Phi$) from λ-terms to $\lambda\Phi$-terms. These mappings differ in the translation of β-redexes. Clearly the values of these mappings are β-irreducible terms, i.e. normal forms for an extended language. As usual we call such a term an *approximate normal form* or abbreviated an *anf*.

19.3.2. Definition. The mappings $\Box_\bot : \Lambda\to\Lambda\bot$, $\Box_L : \Lambda\to\Lambda\bot$, $\Box_\Phi : \Lambda\to\Lambda\Phi$ are inductively defined as follows.

$$\begin{aligned}
\Box(\lambda\vec{x}.y\vec{M}) &= \lambda\vec{x}.y\Box(M_1)\dots\Box(M_m);\\
\Box_\bot(\lambda\vec{x}.(\lambda y.R)N\vec{M}) &= \bot;\\
\Box_L(\lambda\vec{x}.(\lambda y.R)N\vec{M}) &= \lambda\vec{x}.\bot;\\
\Box_\Phi(\lambda\vec{x}.(\lambda y.R)N\vec{M}) &= \lambda\vec{x}.\Phi\Box_\Phi(\lambda y.R)\Box_\Phi(N)\Box_\Phi(M_1)\dots\Box_\Phi(M_m),
\end{aligned}$$

where $\Box\in\{\Box_\bot,\Box_L,\Box_\Phi\}$, $\vec{M}\equiv M_1\dots M_m$ and $m\geq 0$.

The mapping $\Box_\bot$ is related to the Böhm-tree of untyped lambda terms, whereas $\Box_L$ to the Lévy-Longo trees, see van Bakel et al. [2002], where these trees are

related to intersection types.
In order to give the appropriate Approximation Theorem we will use the mapping $\Box_\perp$ for the type assignment systems $\lambda^{\text{Scott}}_\cap$, $\lambda^{\text{CDZ}}_\cap$, $\lambda^{\text{DHM}}_\cap$, $\lambda^{\text{BCD}}_\cap$ the mapping $\Box_L$ for the type assignment system $\lambda^{\text{AO}}_\cap$, and the mapping $\Box_\Phi$ for the type assignment systems $\lambda^{\text{Park}}_\cap$, $\lambda^{\text{HR}}_\cap$. Each one of the above mappings associates a set of approximants to each λ-term in the standard way.

19.3.3. Definition. Let $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{BCD}, \text{AO}\}$.
The set $\mathcal{A}_\mathcal{T}(M)$ of $\mathcal{T}$*-approximants* of M is defined by

$$\mathcal{A}_\mathcal{T}(M) = \{P \mid \exists M'.\, M \twoheadrightarrow_\beta M' \text{ and } P \equiv \Box(M')\},$$

where

$$\begin{array}{lcll} \Box & = & \Box_\perp, & \text{for } \mathcal{T} \in \{\text{Scott}, \text{CDZ}, \text{DHM}, \text{BCD}\}, \\ \Box & = & \Box_L, & \text{for } \mathcal{T} \in \{\text{AO}\}, \\ \Box & = & \Box_\Phi, & \text{for } \mathcal{T} \in \{\text{Park}, \text{HR}\}. \ \blacksquare \end{array}$$

We extend the typing to $\lambda\perp$-terms and to $\lambda\Phi$-terms by adding two different axioms for Φ and nothing for $\perp$.

19.3.4. Definition. (i) Let $\mathcal{T} \in \{\text{Scott}, \text{CDZ}, \text{DHM}, \text{BCD}, \text{AO}\}$. We extend the definition of type assignment $\Gamma \vdash^\mathcal{T}_\cap M : A$ to $\lambda\perp$-terms by letting M, N in Definition 15.2.3 range over $\Lambda\perp$.
(ii) We extend the type assignment $\lambda^{\text{Park}}_\cap$ to $\lambda\Phi$-terms by adding the axiom
(Ax-Φ-Park) $\Gamma \vdash^{\text{Park}}_\cap \Phi : 0$.
(iii) We extend the type assignment $\lambda^{\text{HR}}_\cap$ to $\lambda\Phi$-terms by adding the axiom
(Ax-Φ-HR) $\Gamma \vdash^{\text{HR}}_\cap \Phi : 1$. ■

We do not introduce different notations for these extended type assignment systems concerning terms in $\Lambda\perp\Phi$. It is easy to verify that the Inversion Theorems (Theorems 16.1.1 and 16.1.9) remain valid. In addition to these the following result is relevant.

19.3.5. Proposition. (i) *Let* $\mathcal{T} \in \{\text{Scott}, \text{CDZ}, \text{DHM}, \text{BCD}, \text{AO}\}$. *Then*

$$\Gamma \vdash^\mathcal{T}_\cap \perp : A \iff A =_\mathcal{T} \mathtt{U}.$$

(ii) $\Gamma \vdash^{\text{Park}}_\cap \Phi : A \iff 0 \leq_{\text{Park}} A$.
(iii) $\Gamma \vdash^{\text{HR}}_\cap \Phi : A \iff 1 \leq_{\text{HR}} A$. ■

19.3.6. Lemma. *Let* $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{BCD}, \text{AO}\}$.
(i) $M_1 \twoheadrightarrow_\beta M_2$ & $\Gamma \vdash^\mathcal{T}_\cap \Box(M_1) : A \Rightarrow \Gamma \vdash^\mathcal{T}_\cap \Box(M_2) : A$.
(ii) *If* $P, P' \in \mathcal{A}_\mathcal{T}(M)$, $\Gamma \vdash^\mathcal{T}_\cap P : A$ *and* $\Gamma \vdash^\mathcal{T}_\cap P' : B$, *then*

$$\exists P'' \in \mathcal{A}_\mathcal{T}(M).\Gamma \vdash^\mathcal{T}_\cap P'' : A \cap B.$$

PROOF. (i). For $\mathcal{T} \in \{\text{Scott}, \text{CDZ}, \text{DHM}, \text{BCD}, \text{AO}\}$ the proof follows by induction on the structure of the term distinguishing cases (being or not in head normal form) and using the Inversion Theorems.
For $\mathcal{T} \in \{\text{Park}, \text{HR}\}$ it sufficies to consider the case $M_1 \equiv (\lambda x.M)N$ and $M_2 \equiv M[x := N]$. Notice that $\Box_\Phi(M[x := N])$ is $\Box_\Phi(M)$ where the occurrences of x have been replaced by $\Phi\Box_\Phi(N)$ if they are functional and N is an abstraction, and by $\Box_\Phi(N)$ otherwise. More formally, define the mapping $\overline{\Box_\Phi} : \Lambda \to \Lambda\Phi$ by

$$\overline{\Box_\Phi}(M) = \begin{cases} \Phi\Box_\Phi(M) & \text{if } M \equiv \lambda x.M' \\ \Box_\Phi(M) & \text{otherwise} \end{cases}$$

and the mapping $\{\ \}^x_y : \Lambda \to \Lambda$ by

$$\begin{array}{lcl} \{z\}^x_y & = & z \\ \{M_1M_2\}^x_y & = & \begin{cases} y\{M_2\}^x_y & \text{if } M_1 \equiv x \\ \{M_1\}^x_y\{M_2\}^x_y & \text{otherwise} \end{cases} \\ \{\lambda z.M\}^x_y & = & \lambda z.\{M\}^x_y. \end{array}$$

Then $\Box_\Phi(M_1M_2) = \overline{\Box_\Phi}(M_1)\Box_\Phi(M_2)$ and one can check, by induction on M, that $\Box_\Phi(M[x := N]) \equiv \Box_\Phi(\{M\}^x_y)[x := \Box_\Phi(N)][y := \overline{\Box_\Phi}(N)]$ for y fresh.

We may assume $A \neq_\mathcal{T} \mathtt{U}$. Then from $\Gamma \vdash^\mathcal{T}_\cap \Phi(\lambda x.\Box_\Phi(M))\Box_\Phi(N) : A$ we get $\Gamma \vdash^\mathcal{T}_\cap \Phi(\lambda x.\Box_\Phi(M)) : C \to A$, $\Gamma \vdash^\mathcal{T}_\cap \Box_\Phi(N) : C$ for some C, by Theorem 16.1.9(ii). By Lemma 15.1.24 we have $C \to A \neq_\mathcal{T} \mathtt{U}$, so again by Theorem 16.1.9(ii) $\Gamma \vdash^\mathcal{T}_\cap \Phi : B \to C \to A$, $\Gamma \vdash^\mathcal{T}_\cap \lambda x.\Box_\Phi(M) : B$, for some B.

For $\mathcal{T} = \text{Park}$ we get $0 \leq_\text{Park} B \to C \to A$ from $\Gamma \vdash^\text{Park}_\cap \Phi : B \to C \to A$ by Proposition 19.3.5(ii). This implies $B \leq_\text{Park} 0$, $C \leq_\text{Park} 0$, and $0 \leq_\text{Park} A$, since $0 =_\mathcal{T} 0 \to 0$, since Park is β-sound by Theorem 16.1.7, $(C{\to}A) \neq_\mathcal{T} \mathtt{U}$ and $A \neq_\mathcal{T} \mathtt{U}$. We obtain by rule $(\leq)$ $\Gamma \vdash^\text{Park}_\cap \lambda x.\Box_\Phi(M) : 0$ and $\Gamma \vdash^\text{Park}_\cap \Box_\Phi(N) : 0$. We get $\Gamma, x0 \vdash^\text{Park}_\cap \Box_\Phi(M) : 0$ (by Theorem 16.1.9 (iii)) and $\Gamma \vdash^\text{Park}_\cap \Phi\Box_\Phi(N) : 0$ since $0 =_\text{Park} 0 \to 0$. Now $\Box_\Phi(\{M\}^x_y)$ equals $\Box_\Phi(M)$ with some occurrences of x replaced by the fresh variable y. Hence $\Gamma, y\,0, x\,0 \vdash^\text{Park}_\cap \Box_\Phi(\{M\}^x_y)\,0$. So we conclude $\Gamma \vdash^\text{Park}_\cap \Box_\Phi(\{M\}^x_y)[x := \Box_\Phi(N)][y := \overline{\Box_\Phi}(N)] : A$ by rules (cut) and $(\leq)$.

For $\mathcal{T} = \text{HR}$ we get $1 \leq_\text{HR} B \to C \to A$ from $\Gamma \vdash^\text{HR}_\cap \Phi : B \to C \to A$ by Theorem 19.3.5. This implies either ($B \leq_\text{HR} 1$ and $1 \leq_\text{HR} C \to A$) or ($B \leq_\text{HR} 0$ and $0 \leq_\text{HR} C \to A$) since $1 =_\text{HR} (1 \to 1){\cap}(0 \to 0)$ and HR is β-sound by Theorem 16.1.7, $C{\to}A \neq_\text{HR} \mathtt{U}$ and $A \neq_\text{HR} \mathtt{U}$ (notice that $1{\cap}0 = 0$). Similarly in the first case from $1 \leq_\text{HR} C \to A$ we get either $C \leq_\text{HR} 1$ and $1 \leq_\text{HR} A$ or $C \leq_\text{HR} 0$ and $0 \leq_\text{HR} A$. In the second case from $0 \leq_\text{HR} C \to A$ we get $C \leq_\text{HR} 1$ and $0 \leq_\text{HR} A$ since $0 =_\text{HR} 1 \to 0$.

To sum up, using rule $(\leq)$ we have the following alternative cases.

- $\Gamma \vdash^\text{HR}_\cap \lambda x.\Box_\Phi(M) : 1$, $\Gamma \vdash^\text{HR}_\cap \Box_\Phi(N) : 1$, and $1 \leq_\text{HR} A$;
- $\Gamma \vdash^\text{HR}_\cap \lambda x.\Box_\Phi(M) : 1$, $\Gamma \vdash^\text{HR}_\cap \Box_\Phi(N) : 0$, and $0 \leq_\text{HR} A$;
- $\Gamma \vdash^\text{HR}_\cap \lambda x.\Box_\Phi(M) : 0$, $\Gamma \vdash^\text{HR}_\cap \Box_\Phi(N) : 1$, and $0 \leq_\text{HR} A$.

From Theorem 16.1.9 (iii) we get alternatively:

- $\Gamma, x\,1 \vdash^{\mathrm{HR}}_{\cap} \Box_\Phi(M) : 1$, and $\Gamma \vdash^{\mathrm{HR}}_{\cap} \Phi\Box_\Phi(N) : 1$;
- $\Gamma, x\,0 \vdash^{\mathrm{HR}}_{\cap} \Box_\Phi(M) : 0$, and $\Gamma \vdash^{\mathrm{HR}}_{\cap} \Phi\Box_\Phi(N) : 0$;
- $\Gamma, x\,1 \vdash^{\mathrm{HR}}_{\cap} \Box_\Phi(M) : 0$, and $\Gamma \vdash^{\mathrm{HR}}_{\cap} \Phi\Box_\Phi(N) : 1$,

so we can conclude as in the previous case.

(ii). By hypotheses there are M_1, M_2 such that $M \twoheadrightarrow_\beta M_1$, $M \twoheadrightarrow_\beta M_2$ and $P \equiv \Box(M_1)$, $P' \equiv \Box(M_2)$. By the Church-Rosser property of $\to_\beta$ we can find M_3 such that $M_1 \twoheadrightarrow_\beta M_3$ and $M_2 \twoheadrightarrow_\beta M_3$. By (i) we can choose $P'' \equiv \Box(M_3)$. ■

Approximation Theorem - Part 1

It is useful to introduce the following definition.

19.3.7. Definition. Let $\mathcal{T} \in \{\mathrm{Scott}, \mathrm{Park}, \mathrm{CDZ}, \mathrm{HR}, \mathrm{DHM}, \mathrm{BCD}, \mathrm{AO}\}$. Write

$$[A]^{\mathcal{T}}_\Gamma = \{M \mid \exists P \in \mathcal{A}_{\mathcal{T}}(M).\ \Gamma \vdash^{\mathcal{T}}_{\cap} P : A\}.$$

By definition we get that $M \in [A]^{\mathcal{T}}_\Gamma$ and $N \twoheadrightarrow_\beta M$ imply $N \in [A]^{\mathcal{T}}_\Gamma$. Moreover $\Gamma \subseteq \Gamma'$ implies $[A]^{\mathcal{T}}_\Gamma \subseteq [A]^{\mathcal{T}}_{\Gamma'}$ for all types $A \in \mathbb{T}^{\mathcal{T}}$.

In this subsection we prove that, if $M \in [A]^{\mathcal{T}}_\Gamma$, then there exists a derivation of $\Gamma \vdash^{\mathcal{T}}_{\cap} M : A$.

19.3.8. Proposition. *Let* $\mathcal{T} \in \{\mathrm{Scott}, \mathrm{Park}, \mathrm{CDZ}, \mathrm{HR}, \mathrm{DHM}, \mathrm{BCD}, \mathrm{AO}\}$.

$$M \in [A]^{\mathcal{T}}_\Gamma \Rightarrow \Gamma \vdash^{\mathcal{T}}_{\cap} M : A.$$

Proof. Write $P \equiv \Box(M)$ with $\Box = \Box_\bot$ for $\mathcal{T} \in \{\mathrm{Scott}, \mathrm{CDZ}, \mathrm{DHM}, \mathrm{BCD}\}$, $\Box = \Box_L$ for $\mathcal{T} = \mathrm{AO}$, $\Box = \Box_\Phi$ for $\mathcal{T} \in \{\mathrm{Park}, \mathrm{HR}\}$.

By Corollary 16.2.5 (ii) it is sufficient to show that for all mentioned $\mathcal{T}$ one has

$$\Gamma \vdash^{\mathcal{T}}_{\cap} P : A \Rightarrow \Gamma \vdash^{\mathcal{T}}_{\cap} M : A. \tag{19.1}$$

For $\vdash^{\mathcal{T}}_{\cap}$ just write $\vdash$ in this proof.

For $\mathcal{T} \in \{\mathrm{Scott}, \mathrm{CDZ}, \mathrm{DHM}, \mathrm{BCD}, \mathrm{AO}\}$ the implication (19.1) follows from Proposition 19.3.5(i) and the definition of the mappings $\Box_\bot$ and $\Box_L$.

For $\mathcal{T} \in \{\mathrm{Park}, \mathrm{HR}\}$ we prove (19.1) by induction on M, assuming $A \neq_{\mathcal{T}} \mathtt{U}$.

Case $M \equiv x$. Trivial.

Case $M \equiv \lambda x.M'$. Then $P \equiv \lambda x.P'$ where $P' \equiv \Box_\Phi(M')$. By Theorem 16.1.1(iii) from $\Gamma \vdash P : A$ we get $\Gamma, x\ B_i \vdash P' : C_i$ and $\bigcap_{i\in I}(B_i \to C_i) \leq A$ for some I, B_i, C_i. We get by induction $\Gamma, x\ B_i \vdash M' : C_i$ and so we conclude $\Gamma \vdash M : A$ using rules ($\to$I), ($\cap$I) and ($\leq$).

Case $M \equiv M_1 M_2$, where M_1 is not an abstraction. Then $P \equiv P_1 P_2$ where $P_1 \equiv \Box_\Phi(M_1)$ and $P_2 \equiv \Box_\Phi(M_2)$. By Theorem 16.1.9(ii) from $\Gamma \vdash P : A$ we get $\Gamma \vdash P_1 : B \to A$, $\Gamma \vdash P_2 : B$ for some B. By induction this implies $\Gamma \vdash M_1 : B \to A$ and $\Gamma \vdash M_2 : B$, hence $\Gamma \vdash M \equiv M_1 M_2 : A$.

Case $M \equiv M_1 M_2$, where M_1 is an abstraction. Then $P \equiv \Phi P_1 P_2$ where $P_1 \equiv \Box_\Phi(M_1)$ and $P_2 \equiv \Box_\Phi(M_2)$. As in the proof of Lemma 19.3.6(i) from $\Gamma \vdash P : A$, where $A \neq_{\mathcal{T}} \mathtt{U}$, we get $\Gamma \vdash \Phi : B \to C \to A$, $\Gamma \vdash P_1 : B$, $\Gamma \vdash P_2 : C$ for some B, C. By induction this implies $\Gamma \vdash M_1 : B$ and $\Gamma \vdash M_2 : C$.

For $\mathcal{T} = \text{Park}$, as in the proof of Lemma 19.3.6(i), we get $\Gamma \vdash M_1 : 0$ and $\Gamma \vdash M_2 : 0$. We can conclude $\Gamma \vdash M : A$ using rules ($\leq_{\text{Park}}$) and ($\to$E) since $0 =_{\text{Park}} 0 \to 0$.

For $\mathcal{T} = \text{HR}$ as in the proof of Lemma 19.3.6(i) we have the following alternative cases.

- $\Gamma \vdash^{\text{HR}}_{\cap} M_1 : 1$, $\Gamma \vdash^{\text{HR}}_{\cap} M_2 : 1$, and $1 \leq_{\text{HR}} A$;
- $\Gamma \vdash^{\text{HR}}_{\cap} M_1 : 1$, $\Gamma \vdash^{\text{HR}}_{\cap} M_2 : 0$, and $0 \leq_{\text{HR}} A$;
- $\Gamma \vdash^{\text{HR}}_{\cap} M_1 : 0$, $\Gamma \vdash^{\text{HR}}_{\cap} M_2 : 1$, and $0 \leq_{\text{HR}} A$.

It is easy to verify that in all cases we can derive $\Gamma \vdash M : A$ from (I) and ($1 \to 0$) using rules ($\leq_{\text{HR}}$) and ($\to$E). ■

Approximation Theorem - Part 2

In order to prove the converse of Proposition 19.3.8 we will use a Kripke-like version of stable sets Mitchell [1996]. First we need a technical result.

19.3.9. LEMMA. *Let* $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{BCD}, \text{AO}\}$. *Write* $\Gamma' = \Gamma, z : B$, *with* $z \notin \text{FV}(M)$, *and assume* $A \neq_{\mathcal{T}} \mathtt{U}$ *for* $\mathcal{T} = \text{AO}$. *Then*

$$Mz \in [A]^{\mathcal{T}}_{\Gamma'} \;\Rightarrow\; M \in [B \to A]^{\mathcal{T}}_{\Gamma}.$$

PROOF. Let $P \in \mathcal{A}_{\mathcal{T}}(Mz)$ and $\Gamma' \vdash P : A$. We show by cases on P and M that there is $\hat{P} \in \mathcal{A}_{\mathcal{T}}(M)$ such that $\Gamma \vdash \hat{P} : B \to A$.

There are two possibilities.

- $Mz \twoheadrightarrow_\beta M'z$ and $P \equiv \Box(M'z)$;
- $Mz \twoheadrightarrow_\beta (\lambda x.M')z \to_\beta M'[x := z]$ and $P \in \mathcal{A}_{\mathcal{T}}(M'[x := z])$.

In the first case again there are two possibilities.

- $M' \equiv yM_1 \dots M_m, \quad m \geq 0$;
- $M' \equiv (\lambda y.M_0)M_1 \dots M_m, \quad m \geq 0$.

In total there are 4 cases:

- $P \equiv P'z$ and $P' \equiv y\Box(M_1) \dots \Box(M_m) \in \mathcal{A}_{\mathcal{T}}(M)$;
- $P \equiv \bot$ and $\mathcal{T} \in \{\text{Scott}, \text{CDZ}, \text{DHM}, \text{BCD}, \text{AO}\}$;
- $P \equiv \Phi P'z$, $P' \equiv \Box(\lambda y.M_0)\Box(M_1) \dots \Box(M_m) \in \mathcal{A}_{\mathcal{T}}(M)$ and $\mathcal{T} \in \{\text{Park}, \text{HR}\}$;
- $M \twoheadrightarrow_\beta \lambda x.M'$ and $P \in \mathcal{A}_{\mathcal{T}}(M'[x := z])$.

Case $P \equiv P'z$, where $P' \in \mathcal{A}_{\mathcal{T}}(M)$. Then we can choose $\hat{P} \equiv P'$. This is clear if $A =_{\mathcal{T}} \mathtt{U}$ because by assumption $\mathcal{T} \neq \mathrm{AO}$, hence we have $(\mathtt{U}\!\to)$. Now let $A \neq_{\mathcal{T}} \mathtt{U}$. Then by Theorem 16.1.9(ii) from $\Gamma' \vdash P : A$ we get $\Gamma' \vdash P' : C \to A$, $\Gamma' \vdash z : C$ for some C. By Theorem 16.1.9(i) $B \leq C$ and we conclude using $(\leq)$ and (*strenghtening*) $\Gamma \vdash P' : B \to A$.

Case $P \equiv \bot$. By Proposition 19.3.5(i) $A =_{\mathcal{T}} \mathtt{U}$. By assumption, $\mathcal{T} \neq \mathrm{AO}$. Hence, we have rule $(\mathtt{U}\!\to)$.

Case $P \equiv \Phi P'z$, where $P' \in \mathcal{A}_{\mathcal{T}}(M)$ and $\mathcal{T} \in \{\mathrm{Park}, \mathrm{HR}\}$. Now we show that we can choose $\hat{P} \equiv P'$. Again let $A \neq_{\mathcal{T}} \mathtt{U}$. Then from $\Gamma' \vdash P : A$ we get by Theorem 16.1.9(ii) and (i) $\Gamma' \vdash \Phi : C \to D \to A$, and $\Gamma' \vdash P' : C$, and $\Gamma' \vdash z : D$, for some C, D with $B \leq D$. For $\mathcal{T} = \mathrm{Park}$, using Proposition 19.3.5(ii) as in the proof of Lemma 19.3.6(i), we get $C \leq_{\mathrm{Park}} 0$, $D \leq_{\mathrm{Park}} 0$, and $0 \leq_{\mathrm{Park}} A$ (remember that $0 =_{\mathrm{Park}} 0\!\to\!0$). Similarly for $\mathcal{T} = \mathrm{HR}$, using Proposition 19.3.5(iii), we get either $C \leq_{\mathrm{HR}} 1$, $D \leq_{\mathrm{HR}} 1$, and $1 \leq_{\mathrm{HR}} A$ or $C \leq_{\mathrm{HR}} 1$,$D \leq_{\mathrm{HR}} 0$, and $0 \leq_{\mathrm{HR}} A$ or $C \leq_{\mathrm{HR}} 0$, $D \leq_{\mathrm{HR}} 1$, and $0 \leq_{\mathrm{Park}} A$ (remember that $1 =_{\mathrm{HR}} (1\!\to\!1)\cap(0\!\to\!0)$ and $0 =_{\mathrm{HR}} 1\!\to\!0$). In all cases we can conclude $C \leq D \to A \leq B \to A$ and therefore by $(\leq)$ and (*strenghtening*) $\Gamma \vdash P' : B \to A$.

Case $M \twoheadrightarrow_\beta \lambda x.M'$ and $P \in \mathcal{A}_{\mathcal{T}}(M'[x := z])$.If $\Box = \Box_\bot$ and $P \equiv \bot$, then we choose $\hat{P} \equiv P$, otherwise $\hat{P} \equiv \lambda z.P$. ■

The following crucial definition is somewhat involved. It amounts essentially to the definition of the natural set-theoretic semantics of intersection types over a suitable Kripke applicative structure, where bases play the role of worlds.[1] In order to keep the treatment elementary we don't develop the full theory of the natural semantics of intersection types in Kripke applicative structures. The definition below is rather long, since we have different cases for the type 0 and for arrow types according to the different type theories under consideration.

19.3.10. Definition (Kripke type interpretation).
Let $\mathcal{T} \in \{\mathrm{Scott}, \mathrm{Park}, \mathrm{CDZ}, \mathrm{HR}, \mathrm{DHM}, \mathrm{BCD}, \mathrm{AO}\}$. Define

$$
\begin{array}{rcl}
[\![\alpha]\!]^{\mathcal{T}}_\Gamma & = & [\alpha]^{\mathcal{T}}_\Gamma, \qquad \text{for } \alpha \in \mathbb{A}_\infty \cup \{\mathtt{U}, 1\};\\
[\![0]\!]^{\mathcal{T}}_\Gamma & = & \begin{cases} \{M \mid \forall \vec{N}.M\vec{N} \in [0]^{\mathcal{T}}_\Gamma\}, & \text{for } \mathcal{T} \in \{\mathrm{Scott}, \mathrm{DHM}\};\\ \{M \mid \forall \Gamma' \supseteq \Gamma\, \forall \vec{N} \in [1]^{\mathcal{T}}_{\Gamma'}.M\vec{N} \in [0]^{\mathcal{T}}_{\Gamma'}\}, & \text{for } \mathcal{T} \in \{\mathrm{CDZ}, \mathrm{HR}\};\\ [0]^{\mathcal{T}}_\Gamma, & \text{for } \mathcal{T} = \mathrm{Park}; \end{cases}\\
[\![A\!\to\!B]\!]^{\mathcal{T}}_\Gamma & = & \begin{cases} \{M \mid \forall \Gamma' \supseteq \Gamma\, \forall N \in [\![A]\!]^{\mathcal{T}}_{\Gamma'}.MN \in [\![B]\!]^{\mathcal{T}}_{\Gamma'}\}, & \text{if } \mathcal{T} \neq \mathrm{AO} \text{ or } B \neq_{\mathrm{AO}} \mathtt{U};\\ [A\!\to\!B]^{\mathcal{T}}_\Gamma, & \text{if } \mathcal{T} = \mathrm{AO} \;\&\; B =_{\mathrm{AO}} \mathtt{U}; \end{cases}\\
[\![A \cap B]\!]^{\mathcal{T}}_\Gamma & = & [\![A]\!]^{\mathcal{T}}_\Gamma \cap [\![B]\!]^{\mathcal{T}}_\Gamma.
\end{array}
$$

The definition of $[\![A\!\to\!B]\!]^{\mathcal{T}}_\Gamma$ is somewhat involved in order to make Lemma 19.3.12(ii) valid for $\mathcal{T} = \mathrm{AO}$.

[1] As already observed in Berline [2000] we cannot use here stable sets as we will do in section 19.1 since we need to take into account also the $\mathcal{T}$-bases, not only the λ-terms and their types.

19.3.11. Proposition. (i) $M \in [\![A]\!]^{\mathcal{T}}_{\Gamma}$ *and* $N \twoheadrightarrow_{\beta} M$ *imply* $N \in [\![A]\!]^{\mathcal{T}}_{\Gamma}$.

(ii) $\Gamma \subseteq \Gamma'$ *implies* $[\![A]\!]^{\mathcal{T}}_{\Gamma} \subseteq [\![A]\!]^{\mathcal{T}}_{\Gamma'}$ *for all types* $A \in \mathbb{T}^{\mathcal{T}}$.

Proof. Easy. ■

The Lemmas 19.3.12, 19.3.15 and the final theorem are standard.

19.3.12. Lemma. *Let* $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{BCD}, \text{AO}\}$. *Then*

(i) $x\vec{M} \in [A]^{\mathcal{T}}_{\Gamma} \;\Rightarrow\; x\vec{M} \in [\![A]\!]^{\mathcal{T}}_{\Gamma}$.

(ii) $[\![A]\!]^{\mathcal{T}}_{\Gamma} \subseteq [A]^{\mathcal{T}}_{\Gamma}$.

Proof. (i) and (ii) are proved simultaneously by induction on A. We consider only some interesting cases.

(i) Case $A \equiv 0$ and $\mathcal{T} = \text{CDZ}$. Let $\Gamma' \supseteq \Gamma$ and $\vec{N} \in [1]^{\text{CDZ}}_{\Gamma'}$.
Clearly $P \in \mathcal{A}_{\text{CDZ}}(x\vec{M})$, $\vec{Q} \in \mathcal{A}_{\text{CDZ}}(\vec{N}) \Rightarrow P\vec{Q} \in \mathcal{A}_{\text{CDZ}}(x\vec{M}\vec{N})$. Hence

$$\begin{array}{rcll} x\vec{M} \in [0]^{\text{CDZ}}_{\Gamma} & \Rightarrow & x\vec{M}\vec{N} \in [0]^{\text{CDZ}}_{\Gamma'} & \text{by rules } (\leq_{\text{CDZ}}) \text{ and } (\to\text{E}) \\ & & & \text{since } 0 =_{\text{CDZ}} 1 \to 0, \\ & \Rightarrow & x\vec{M} \in [\![0]\!]^{\text{CDZ}}_{\Gamma} & \text{by Definition 19.3.10.} \end{array}$$

Case $A \equiv B \to C$. Let $\Gamma' \supseteq \Gamma$ and $\mathcal{T} \neq \text{AO}$ or $C \neq_{\text{AO}} \mathtt{U}$ and let $N \in [\![B]\!]^{\mathcal{T}}_{\Gamma'}$.
$[\![B]\!]^{\mathcal{T}}_{\Gamma'} \subseteq [B]^{\mathcal{T}}_{\Gamma'}$ by induction on (ii). Hence

$$\begin{array}{rcll} x\vec{M} \in [A]^{\mathcal{T}}_{\Gamma} & \Rightarrow & x\vec{M}N \in [C]^{\mathcal{T}}_{\Gamma'} & \text{by rule } (\to\text{E}), \\ & \Rightarrow & x\vec{M}N \in [\![C]\!]^{\mathcal{T}}_{\Gamma'} & \text{by induction on (i)}, \\ & \Rightarrow & x\vec{M} \in [\![B \to C]\!]^{\mathcal{T}}_{\Gamma} & \text{by Definition 19.3.10.} \end{array}$$

(ii) Case $A \equiv B \to C$ and $\mathcal{T} \neq \text{AO}$ or $C \neq_{\text{AO}} \mathtt{U}$. Let $\Gamma' = \Gamma, z{:}B$ with z fresh, and suppose $M \in [\![B \to C]\!]^{\mathcal{T}}_{\Gamma}$; as $z \in [\![B]\!]^{\mathcal{T}}_{\Gamma, z:B}$ by induction on (i), we have

$$\begin{array}{rcll} M \in [\![B \to C]\!]^{\mathcal{T}}_{\Gamma} \text{ and } z \in [\![B]\!]^{\mathcal{T}}_{\Gamma, z:B} & \Rightarrow & Mz \in [\![C]\!]^{\mathcal{T}}_{\Gamma'} & \text{by Definition 19.3.10,} \\ & \Rightarrow & Mz \in [C]^{\mathcal{T}}_{\Gamma'} & \text{by induction on (ii),} \\ & \Rightarrow & M \in [B \to C]^{\mathcal{T}}_{\Gamma} & \text{by Lemma 19.3.9.} \end{array}$$

Case $A \equiv B \cap C$. This follows from $[\![B \cap C]\!]^{\mathcal{T}}_{\Gamma} = [\![B]\!]^{\mathcal{T}}_{\Gamma} \cap [\![C]\!]^{\mathcal{T}}_{\Gamma}$ and the induction hypothesis using Lemma 19.3.6(ii). ■

The following lemma essentially states that the Kripke type interpretations agree with the corresponding type theories.

19.3.13. Lemma. (i) *Let* $\mathcal{T} \in \{\text{CDZ}, \text{DHM}\}$. *Then*

$$M \in [A]^{\mathcal{T}}_{\Gamma, z\,0} \;\&\; N \in [\![0]\!]^{\mathcal{T}}_{\Gamma} \;\Rightarrow\; M[z := N] \in [A]^{\mathcal{T}}_{\Gamma}.$$

(ii) *Let* $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{BCD}, \text{AO}\}$. *Then*

$$\forall A, B \in \mathbb{T}^{\mathcal{T}}[A \leq_{\mathcal{T}} B \;\Rightarrow\; [\![A]\!]^{\mathcal{T}}_{\Gamma} \subseteq [\![B]\!]^{\mathcal{T}}_{\Gamma}].$$

PROOF. (i) We may assume $A \neq_{\mathcal{T}} \mathtt{U}$.

Let $\mathcal{T} = \mathrm{CDZ}$. If $M \in [A]^{\mathrm{CDZ}}_{\Gamma, z\,0}$, then there is a $P \in \mathcal{A}_{\mathrm{CDZ}}(M)$ such that $\Gamma, z\,0 \vdash^{\mathrm{CDZ}}_{\cap} P : A$. This is proved by induction on P.

Case $P \equiv \bot$. Trivial.

Case $P \equiv \lambda x.P'$. Then $M \twoheadrightarrow_\beta \lambda x.M'$ and $P' \in \mathcal{A}_{\mathrm{CDZ}}(M')$. From $\Gamma, z0 \vdash^{\mathrm{CDZ}}_{\cap} P : A$ we get $\Gamma, z\,0, x\,B_i \vdash^{\mathrm{CDZ}}_{\cap} P' : C_i$ and $\bigcap_{i\in I}(B_i{\to}C_i) \leq_{\mathrm{CDZ}} A$ for some I and $B_i, C_i \in \mathbb{T}^{\mathrm{CDZ}}$ by Theorem 16.1.1(iii). By induction for each $i \in I$ there is a $P_i \in \mathcal{A}_{\mathrm{CDZ}}(M'[z := N])$ such that $\Gamma, x\,B_i \vdash^{\mathrm{CDZ}}_{\cap} P_i : C_i$. Let $P_i = \Box(M_i)$, where $M'[z := N] \twoheadrightarrow_\beta M_i$ and let M'' be a common reduct of the M_i and $P'' \equiv \Box(M'')$. Then $P'' \in \mathcal{A}_{\mathrm{CDZ}}(M'[z := N])$ and $\Gamma, x\,B_i \vdash^{\mathrm{CDZ}}_{\cap} P'' : C_i$, for all $i \in I$, by lemma 19.3.6(i). Clearly $\lambda x.P'' \in \mathcal{A}_{\mathrm{CDZ}}(M[z := N])$ and by construction $\Gamma \vdash^{\mathrm{CDZ}}_{\cap} \lambda x.P'' : A$.

Case $P \equiv x\vec{P}$, then $M \twoheadrightarrow_\beta x\vec{M}$ and $\vec{P} \in \mathcal{A}_{\mathrm{CDZ}}(\vec{M})$. From $\Gamma, z\,0 \vdash^{\mathrm{CDZ}}_{\cap} P : A$ we get $\Gamma, z\,0 \vdash^{\mathrm{CDZ}}_{\cap} x : \vec{B} \to A$ and $\Gamma, z\,0 \vdash^{\mathrm{CDZ}}_{\cap} \vec{P} : \vec{B}$ by Theorem 16.1.9(ii) and Lemma 15.1.24. By induction there are $\vec{P'} \in \mathcal{A}_{\mathrm{CDZ}}(\vec{M}[z := N])$ such that $\Gamma \vdash^{\mathrm{CDZ}}_{\cap} \vec{P'} : \vec{B}$. If $x \neq z$ we are done since $x\vec{P'} \in \mathcal{A}_{\mathrm{CDZ}}(M[z := N])$ and we can derive $\Gamma \vdash^{\mathrm{CDZ}}_{\cap} x\vec{P'} : A$ using $(\to\mathrm{E})$. Otherwise $\Gamma, z\,0 \vdash^{\mathrm{CDZ}}_{\cap} z : \vec{B} \to A$ implies $0 \leq_{\mathrm{CDZ}} \vec{B} \to A$ by Theorem 16.1.9(i). Being CDZ β-sound by Theorem 16.1.7 from $0 =_{\mathrm{CDZ}} \vec{1} \to 0$ we obtain $\vec{B} \leq_{\mathrm{CDZ}} \vec{1}$ and $0 \leq_{\mathrm{CDZ}} A$ by Lemma 15.1.24. So we get $\Gamma \vdash^{\mathrm{CDZ}}_{\cap} \vec{P'} : \vec{1}$, i.e. $\vec{M}[z := N] \in [1]^{\mathrm{CDZ}}_{\Gamma}$. Now

$$N \in [\![0]\!]^{\mathrm{CDZ}}_{\Gamma} \;\&\; \vec{M}[z := N] \in [1]^{\mathrm{CDZ}}_{\Gamma} \;\Rightarrow\; M[z := N] \in [0]^{\mathrm{CDZ}}_{\Gamma},$$

by Definition 19.3.10. Since $0 \leq_{\mathrm{CDZ}} A$ we get $M[z := N] \in [A]^{\mathrm{CDZ}}_{\Gamma}$.

Let $\mathcal{T} = \mathrm{DHM}$. Then the proof is similar but easier. In the case $P \equiv z\vec{P}$ it follows from Definition 19.3.10 that $N \in [\![0]\!]^{\mathrm{DHM}}_{\Gamma} \;\Rightarrow\; M[z := N] \in [A]^{\mathrm{DHM}}_{\Gamma}$.

(ii) We treat the cases related to $A{\to}B{\leq}\mathtt{U}{\to}\mathtt{U}$ in AO, $(0{\to}1){=}1, 0{=}(1{\to}0)$ in CDZ, $(1{\to}1) \cap (0{\to}0){=}1$ in HR, and $(0{\to}0){=}0$ in Park.

Proof of $[\![A{\to}B]\!]^{\mathrm{AO}}_{\Gamma} \subseteq [\![\mathtt{U}{\to}\mathtt{U}]\!]^{\mathrm{AO}}_{\Gamma}$. If $B =_{\mathrm{AO}} \mathtt{U}$, then

$$[\![A{\to}B]\!]^{\mathrm{AO}}_{\Gamma} = [A{\to}B]^{\mathrm{AO}}_{\Gamma} \subseteq [\mathtt{U}{\to}\mathtt{U}]^{\mathrm{AO}}_{\Gamma} = [\![\mathtt{U}{\to}\mathtt{U}]\!]^{\mathrm{AO}}_{\Gamma}.$$

If, on the other hand, $B \neq_{\mathrm{AO}} \mathtt{U}$, then $M \in [\![A{\to}B]\!]^{\mathrm{AO}}_{\Gamma}$. Write $\Gamma' = \Gamma, z{:}A$. Then $z \in [A]^{\mathrm{AO}}_{\Gamma'}$, hence by Lemma 19.3.12(i) $z \in [\![A]\!]^{\mathrm{AO}}_{\Gamma'}$. So $Mz \in [\![B]\!]^{\mathrm{AO}}_{\Gamma'} \subseteq [B]^{\mathrm{AO}}_{\Gamma'}$, by Lemma 19.3.12(ii), and therefore $M \in [A{\to}B]^{\mathrm{AO}}_{\Gamma} \subseteq [\mathtt{U}{\to}\mathtt{U}]^{\mathrm{AO}}_{\Gamma} = [\![\mathtt{U}{\to}\mathtt{U}]\!]^{\mathrm{AO}}_{\Gamma}$, by Lemma 19.3.9.

Proof of $[\![0{\to}1]\!]^{\mathrm{CDZ}}_{\Gamma} \subseteq [\![1]\!]^{\mathrm{CDZ}}_{\Gamma}$. We have

$$\begin{aligned}
[\![0{\to}1]\!]^{\mathrm{CDZ}}_{\Gamma} &\subseteq [0{\to}1]^{\mathrm{CDZ}}_{\Gamma}, && \text{by Lemma 19.3.12(ii),}\\
&= [1]^{\mathrm{CDZ}}_{\Gamma}, && \text{since } 0{\to}1 =_{\mathrm{CDZ}} 1,\\
&= [\![1]\!]^{\mathrm{CDZ}}_{\Gamma}, && \text{by Definition 19.3.10.}
\end{aligned}$$

Proof of $[\![1]\!]^{\mathrm{CDZ}}_{\Gamma} \subseteq [\![0{\to}1]\!]^{\mathrm{CDZ}}_{\Gamma}$. Suppose $\Gamma' \supseteq \Gamma$, $M \in [\![1]\!]^{\mathrm{CDZ}}_{\Gamma}$ and $N \in [\![0]\!]^{\mathrm{CDZ}}_{\Gamma'}$, in order to show $MN \in [\![1]\!]^{\mathrm{CDZ}}_{\Gamma'}$. By Definition 19.3.10 $[\![1]\!]^{\mathrm{CDZ}}_{\Gamma} = [1]^{\mathrm{CDZ}}_{\Gamma}$. If

$M \in [1]^{\mathrm{CDZ}}_{\Gamma}$, then there is $P \in \mathcal{A}_{\mathrm{CDZ}}(M)$ such that $\Gamma \vdash^{\mathrm{CDZ}}_{\cap} P : 1$. We will show $MN \in [\![1]\!]^{\mathrm{CDZ}}_{\Gamma'}$ by distinguishing cases of P.

Case $P \equiv \bot$. By Proposition 19.3.5(i) one has $1 =_{\mathrm{CDZ}} \mathtt{U}$, contradicting Proposition 15.1.28. So this case is impossible.

Case $P \equiv \lambda z.P'$. Then $M \twoheadrightarrow_\beta \lambda z.M'$ and $P' \in \mathcal{A}_{\mathrm{CDZ}}(M')$. From $\Gamma \vdash^{\mathrm{CDZ}}_{\cap} P : 1$ we get $\Gamma, z\,0 \vdash^{\mathrm{CDZ}}_{\cap} P' : 1$ by Theorem 16.1.9(iii), since $1 =_{\mathrm{CDZ}} 0{\to}1$. This implies $M' \in [1]^{\mathrm{CDZ}}_{\Gamma, z\,0}$. We may assume that $z \notin \mathrm{dom}(\Gamma')$. Then also $M' \in [1]^{\mathrm{CDZ}}_{\Gamma', z\,0}$. Therefore

$$\begin{array}{lll} MN & \twoheadrightarrow_\beta & (\lambda z.M')N \\ & \to_\beta & M'[z := N] \\ & \in & [1]^{\mathrm{CDZ}}_{\Gamma'}, \qquad \text{by (i)}, \\ & = & [\![1]\!]^{\mathrm{CDZ}}_{\Gamma'}. \end{array}$$

Case $P \equiv x\vec{P}$. Notice that $\Gamma \vdash^{\mathrm{CDZ}}_{\cap} P : 1$ implies $\Gamma \vdash^{\mathrm{CDZ}}_{\cap} P : 0{\to}1$, since $1 =_{\mathrm{CDZ}} 0{\to}1$. By Lemma 19.3.12(ii) $[\![0]\!]^{\mathrm{CDZ}}_{\Gamma'} \subseteq [0]^{\mathrm{CDZ}}_{\Gamma'}$, hence there is $P' \in \mathcal{A}_{\mathrm{CDZ}}(N)$ such that $\Gamma' \vdash^{\mathrm{CDZ}}_{\cap} P' : 0$. We get $\Gamma' \vdash^{\mathrm{CDZ}}_{\cap} PP' : 1$. As $PP' \in \mathcal{A}_{\mathrm{CDZ}}(MN)$ we conclude that $MN \in [\![1]\!]^{\mathrm{CDZ}}_{\Gamma'}$.

Proof of $[\![1{\to}0]\!]^{\mathrm{CDZ}}_{\Gamma} \subseteq [\![0]\!]^{\mathrm{CDZ}}_{\Gamma}$. We have $[\![1{\to}0]\!]^{\mathrm{CDZ}}_{\Gamma} \subseteq [1{\to}0]^{\mathrm{CDZ}}_{\Gamma}$, by Lemma 19.3.12(ii) and $[1{\to}0]^{\mathrm{CDZ}}_{\Gamma} = [0]^{\mathrm{CDZ}}_{\Gamma}$, as $1{\to}0 =_{\mathrm{CDZ}} 0$, using Definition 19.3.7. Moreover using Definition 19.3.10 it follows that

$$\begin{array}{lll} [\![1{\to}0]\!]^{\mathrm{CDZ}}_{\Gamma} & = & \{M \mid \forall \Gamma' \supseteq \Gamma, \forall N \in [\![1]\!]^{\mathrm{CDZ}}_{\Gamma'}.MN \in [\![0]\!]^{\mathrm{CDZ}}_{\Gamma'}\} \\ & = & \{M \mid \forall \Gamma' \supseteq \Gamma, \forall N \in [1]^{\mathrm{CDZ}}_{\Gamma'}.MN \in [\![0]\!]^{\mathrm{CDZ}}_{\Gamma'}\} \\ & \subseteq & \{M \mid \forall \Gamma' \supseteq \Gamma, \forall N, \vec{N} \in [1]^{\mathrm{CDZ}}_{\Gamma'}.MN\vec{N} \in [0]^{\mathrm{CDZ}}_{\Gamma'}\}. \end{array}$$

From $[\![1{\to}0]\!]^{\mathrm{CDZ}}_{\Gamma} \subseteq [0]^{\mathrm{CDZ}}_{\Gamma}$ and

$$[\![1{\to}0]\!]^{\mathrm{CDZ}}_{\Gamma} \subseteq \{M \mid \forall \Gamma' \supseteq \Gamma, \forall N, \vec{N} \in [1]^{\mathrm{CDZ}}_{\Gamma'}.MN\vec{N} \in [0]^{\mathrm{CDZ}}_{\Gamma'}\}$$

we can conclude

$$[\![1{\to}0]\!]^{\mathrm{CDZ}}_{\Gamma} \subseteq \{M \mid \forall \Gamma' \supseteq \Gamma, \vec{N} \in [1]^{\mathrm{CDZ}}_{\Gamma'}.M\vec{N} \in [0]^{\mathrm{CDZ}}_{\Gamma'}\} = [\![0]\!]^{\mathrm{CDZ}}_{\Gamma}.$$

Proof of $[\![0]\!]^{\mathrm{CDZ}}_{\Gamma} \subseteq [\![1{\to}0]\!]^{\mathrm{CDZ}}_{\Gamma}$. Again using Definition 19.3.10 one has

$$\begin{array}{lll} M \in [\![0]\!]^{\mathrm{CDZ}}_{\Gamma} & \Rightarrow & \forall \Gamma' \supseteq \Gamma, \forall N, \vec{N} \in [1]^{\mathrm{CDZ}}_{\Gamma'}.MN\vec{N} \in [0]^{\mathrm{CDZ}}_{\Gamma'} \\ & \Rightarrow & \forall \Gamma' \supseteq \Gamma, \forall N \in [1]^{\mathrm{CDZ}}_{\Gamma'}.MN \in [\![0]\!]^{\mathrm{CDZ}}_{\Gamma'} \\ & \Rightarrow & M \in [\![1{\to}0]\!]^{\mathrm{CDZ}}_{\Gamma}. \end{array}$$

Proof of $[\![(1{\to}1){\cap}(0{\to}0)]\!]^{\mathrm{HR}}_{\Gamma} \subseteq [\![1]\!]^{\mathrm{HR}}_{\Gamma}$. By Lemma 19.3.12(ii) one has

$$\begin{array}{rcl} [\![(1{\to}1){\cap}(0{\to}0)]\!]^{\mathrm{HR}}_{\Gamma} & \subseteq & [(1{\to}1){\cap}(0{\to}0)]^{\mathrm{HR}}_{\Gamma} \\ & = & [1]^{\mathrm{HR}}_{\Gamma} \\ & = & [\![1]\!]^{\mathrm{HR}}_{\Gamma}, \end{array}$$

by Definition 19.3.7, $(1{\to}1){\cap}(0{\to}0) = 1$ and Definition 19.3.10.

Proof of $[\![1]\!]^{\mathrm{HR}}_{\Gamma} \subseteq [\![(1{\to}1){\cap}(0{\to}0)]\!]^{\mathrm{HR}}_{\Gamma}$. Let $\Gamma' \supseteq \Gamma$.

$$\begin{array}{l} M \in [\![1]\!]^{\mathrm{HR}}_{\Gamma} \Rightarrow M \in [1]^{\mathrm{HR}}_{\Gamma} \\ \qquad\qquad \Rightarrow \exists P \in \mathcal{A}_{\mathrm{HR}}(M)\ \Gamma \vdash^{\mathrm{HR}}_{\cap} P : 1, \qquad \text{by Definition 19.3.7.} \end{array} \tag{19.2}$$

$$\begin{array}{l} N \in [\![1]\!]^{\mathrm{HR}}_{\Gamma'} \Rightarrow N \in [1]^{\mathrm{HR}}_{\Gamma'} \\ \qquad\qquad \Rightarrow \exists P' \in \mathcal{A}_{\mathrm{HR}}(N)\ \Gamma' \vdash^{\mathrm{HR}}_{\cap} P' : 1, \text{ by Definition 19.3.7.} \end{array} \tag{19.3}$$

Let $\hat{P} \equiv \Phi P P'$ if P is a lambda abstraction and $\hat{P} \equiv P P'$ otherwise.

$$\begin{array}{rcll} \text{(19.2) and (19.3)} & \Rightarrow & \Gamma' \vdash^{\mathrm{HR}} \hat{P} : 1, & \text{by (Ax-}\Phi\text{-HR), } (\leq_{\mathcal{T}}), (\to\text{E}), \\ & \Rightarrow & MN \in [1]^{\mathrm{HR}}_{\Gamma'}, & \text{since } \hat{P} \in \mathcal{A}_{\mathrm{HR}}(MN), \\ & \Rightarrow & MN \in [\![1]\!]^{\mathrm{HR}}_{\Gamma'} & \\ & \Rightarrow & M \in [\![1 \to 1]\!]^{\mathrm{HR}}_{\Gamma}. & \end{array}$$

$$\begin{array}{l} N \in [\![0]\!]^{\mathrm{HR}}_{\Gamma'} \Rightarrow N \in [0]^{\mathrm{HR}}_{\Gamma'} \\ \qquad\qquad \Rightarrow \exists P' \in \mathcal{A}_{\mathrm{HR}}(N)\ \Gamma' \vdash^{\mathrm{HR}}_{\cap} P' : 0, \text{ by Definition 19.3.7.} \end{array} \tag{19.4}$$

$$\begin{array}{l} \vec{N} \in [\![1]\!]^{\mathrm{HR}}_{\Gamma'} \Rightarrow \vec{N} \in [1]^{\mathrm{HR}}_{\Gamma'} \\ \qquad\qquad \Rightarrow \exists \vec{P} \in \mathcal{A}_{\mathrm{HR}}(\vec{N})\ \Gamma' \vdash^{\mathrm{HR}}_{\cap} \vec{P} : \vec{1}, \qquad \text{by Definition 19.3.7.} \end{array} \tag{19.5}$$

Let $\hat{P} \equiv \Phi P P' \vec{P}$ if P is a lambda-abstraction and $\hat{P} \equiv P P' \vec{P}$ otherwise.

$$\begin{array}{rcll} \text{(19.2), (19.4) and (19.5)} & \Rightarrow & \Gamma' \vdash^{\mathrm{HR}} \hat{P} : 0, & \text{by (Ax-}\Phi\text{-HR), } (\leq_{\mathcal{T}}), (\to\text{E}) \\ & \Rightarrow & MN\vec{N} \in [0]^{\mathrm{HR}}_{\Gamma'} & \text{since } \hat{P} \in \mathcal{A}_{\mathrm{HR}}(MN\vec{N}) \\ & \Rightarrow & MN \in [\![0]\!]^{\mathrm{HR}}_{\Gamma'} & \\ & \Rightarrow & M \in [\![0 \to 0]\!]^{\mathrm{HR}}_{\Gamma}. & \end{array}$$

Proof of $[\![0{\to}0]\!]^{\mathrm{Park}}_{\Gamma} \subseteq [\![0]\!]^{\mathrm{Park}}_{\Gamma}$. Let $M \in [\![0{\to}0]\!]^{\mathrm{Park}}_{\Gamma}$ and $\Gamma' = \Gamma, z : 0$, where $z \notin \mathrm{FV}(M)$.

$$\begin{array}{rcll} z \in [0]^{\mathrm{Park}}_{\{z:0\}} & \Rightarrow & z \in [\![0]\!]^{\mathrm{Park}}_{\{z:0\}} & \\ & \Rightarrow & Mz \in [\![0]\!]^{\mathrm{Park}}_{\Gamma'} & \\ & \Rightarrow & Mz \in [0]^{\mathrm{Park}}_{\Gamma'} & \\ & \Rightarrow & M \in [0]^{\mathrm{Park}}_{\Gamma}, & \text{by Lemma 19.3.9 and } (0 \to 0) \leq_{\mathrm{Park}} 0, \\ & \Rightarrow & M \in [\![0]\!]^{\mathrm{Park}}_{\Gamma}. & \end{array}$$

Proof of $[\![0]\!]^{\mathrm{Park}}_{\Gamma} \subseteq [\![0{\to}0]\!]^{\mathrm{Park}}_{\Gamma}$. Let $\Gamma' \supseteq \Gamma$. Then we have

$$\begin{aligned} M \in [\![0]\!]^{\mathrm{Park}}_{\Gamma} &\Rightarrow M \in [0]^{\mathrm{Park}}_{\Gamma} \\ &\Rightarrow \exists P \in \mathcal{A}_{\mathrm{Park}}(M)\ \Gamma \vdash^{\mathrm{Park}}_{\cap} P : 0, \quad \text{by Definition 19.3.7.} \end{aligned} \tag{19.6}$$

$$\begin{aligned} N \in [\![0]\!]^{\mathrm{Park}}_{\Gamma'} &\Rightarrow N \in [0]^{\mathrm{Park}}_{\Gamma'} \\ &\Rightarrow \exists P' \in \mathcal{A}_{\mathrm{Park}}(N)\ \Gamma' \vdash^{\mathrm{Park}}_{\cap} P' : 0, \text{ by Definition 19.3.7.} \end{aligned} \tag{19.7}$$

Let $\hat{P} \equiv \Phi P P'$ if P is a lambda-abstraction and $\hat{P} \equiv PP'$ otherwise.

$$\begin{aligned} \text{(19.6) and (19.7)} \ &\Rightarrow\ \Gamma' \vdash^{\mathrm{Park}}_{\cap} \hat{P} : 0, && \text{by (Ax-}\Phi\text{), } (\leq_{\mathrm{Park}}), (\to\mathrm{E}) \\ &\Rightarrow\ MN \in [0]^{\mathrm{Park}}_{\Gamma'}, && \text{since } \hat{P} \in \mathcal{A}_{\mathrm{Park}}(MN) \\ &\Rightarrow\ MN \in [\![0]\!]^{\mathrm{Park}}_{\Gamma'} \\ &\Rightarrow\ M \in [\![0 \to 0]\!]^{\mathrm{Park}}_{\Gamma}. \ \blacksquare \end{aligned}$$

19.3.14. Definition (Semantic Satisfiability). Let ρ be a mapping from term variables to terms and write $[\![M]\!]_\rho = M[\vec{x} := \rho(\vec{x})]$, where $\vec{x} = \mathrm{FV}(M)$.

(i) $\mathcal{T}, \rho, \Gamma \models M : A \iff [\![M]\!]_\rho \in [\![A]\!]^{\mathcal{T}}_{\Gamma}$.
(ii) $\mathcal{T}, \rho, \Gamma' \models \Gamma \iff \mathcal{T}, \rho, \Gamma' \models x : B$, for all $(x\ B) \in \Gamma$;
(iii) $\Gamma \models^{\mathcal{T}} M : A \iff \mathcal{T}, \rho, \Gamma' \models \Gamma \Rightarrow \mathcal{T}, \rho, \Gamma' \models M : A$, for all ρ, Γ'.

In line with the previous remarks, the following result can be constructed also as the soundness of the natural semantics of intersection types over a particular Kripke applicative structure, where bases play the role of worlds.

19.3.15. Lemma. *Let* $\mathcal{T} \in \{\mathrm{Scott}, \mathrm{Park}, \mathrm{CDZ}, \mathrm{HR}, \mathrm{DHM}, \mathrm{BCD}, \mathrm{AO}\}$. *Then*

$$\Gamma \vdash^{\mathcal{T}}_{\cap} M : A \ \Rightarrow\ \Gamma \models^{\mathcal{T}} M : A.$$

Proof. The proof is by induction on the derivation of $\Gamma \vdash^{\mathcal{T}}_{\cap} M : A$.

Cases (Ax), (Ax-U). Immediate.

Cases ($\to$E), ($\cap$I). By induction.

Case ($\leq$). By Lemma 19.3.13(ii).

Case ($\to$I). Suppose $M \equiv \lambda y.R$, $A \equiv B{\to}C$ and $\Gamma, y : B \vdash^{\mathcal{T}}_{\cap} R : C$.

Subcase $\mathcal{T} \neq$ AO or $C \neq_{\mathrm{AO}} \mathtt{U}$. Suppose $\mathcal{T}, \rho, \Gamma' \models \Gamma$ in order to show $[\![\lambda y.R]\!]_\rho \in [\![B{\to}C]\!]^{\mathcal{T}}_{\Gamma'}$. Let $\Gamma'' \supseteq \Gamma'$ and $T \in [\![B]\!]^{\mathcal{T}}_{\Gamma''}$. Then by the induction hypothesis $[\![R]\!]_{\rho[y:=T]} \in [\![C]\!]^{\mathcal{T}}_{\Gamma''}$. We may assume $y \notin \rho(x)$ for all $x \in \mathrm{dom}(\Gamma)$. Then one has $[\![\lambda y.R]\!]_\rho T \to_\beta [\![R]\!]_{\rho[y:=T]}$ and hence $[\![\lambda y.R]\!]_\rho T \in [\![C]\!]^{\mathcal{T}}_{\Gamma''}$, by Proposition 19.3.11. Therefore $[\![\lambda y.R]\!]_\rho \in [\![B{\to}C]\!]^{\mathcal{T}}_{\Gamma'}$.

Subcase $\mathcal{T} =$ AO and $C =_{\mathrm{AO}} \mathtt{U}$. The result follows easily observing that $\lambda x.\bot \in \mathcal{A}_{\mathrm{AO}}([\![\lambda y.R]\!]_\rho)$ for all ρ and we can derive $\vdash^{\mathrm{AO}}_{\cap} \lambda x.\bot : B{\to}C$ using (Ax-U), ($\to$I) and ($\leq_{\mathrm{AO}}$). Therefore $[\![M]\!]_\rho \in [A]^{\mathcal{T}}_{\Gamma}$, implying $[\![M]\!]_\rho \in [\![A]\!]^{\mathcal{T}}_{\Gamma}$ by Definition 19.3.10. $\blacksquare$

Now we can prove the converse of Proposition 19.3.8.

19.3.16. Proposition. *Let* $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{BCD}, \text{AO}\}$. *Then*

$$\Gamma \vdash^{\mathcal{T}}_{\cap} M : A \Rightarrow M \in [A]^{\mathcal{T}}_{\Gamma}.$$

Proof. Let $\rho_0(x) = x$. By Lemma 19.3.12(i) $\mathcal{T}, \rho_0, \Gamma \models \Gamma$. Then $\Gamma \vdash^{\mathcal{T}}_{\cap} M : A$ implies $M = [\![M]\!]_{\rho_0} \in [\![A]\!]^{\mathcal{T}}_{\Gamma}$ by Lemma 19.3.15. So we conclude $M \in [A]^{\mathcal{T}}_{\Gamma}$ by Lemma 19.3.12(ii). ■

19.3.17. Theorem (Approximation Theorem). *Let* $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{BCD}, \text{AO}\}$. *Then*

$$\Gamma \vdash^{\mathcal{T}}_{\cap} M : A \iff \exists P \in \mathcal{A}_{\mathcal{T}}(M).\Gamma \vdash^{\mathcal{T}}_{\cap} P : A.$$

Proof. By Propositions 19.3.8 and 19.3.16. ■

19.3.18. Corollary. *Let* $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{BCD}, \text{AO}\}$. *Let* M *be an untyped lambda term. Then*

$$[\![M]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho} = \{A \mid \Gamma_{\rho} \vdash^{\mathcal{T}}_{\cap} P : A \text{ for some } P \in \mathcal{A}_{\mathcal{T}}(M) \text{ and some } \Gamma \models \rho\}.$$

Proof. By Theorem 18.2.7 and the Approximation Theorem. ■

Another way of writing this is

$$\begin{aligned} [\![M]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho} &= \bigcup_{P \in \mathcal{A}_{\mathcal{T}}(M)} [\![P]\!]^{\mathcal{F}^{\mathcal{T}}}_{\rho} \\ &= \bigcup_{P \in \mathcal{A}_{\mathcal{T}}(M)} \{A \mid \Gamma \vdash^{\mathcal{T}}_{\cap} P : A \text{ for some } \Gamma \models \rho\}. \end{aligned}$$

This gives the motivation for the name 'Approximation Theorem'. Theorem 19.3.17 was first proved for $\mathcal{T} = \text{BCD}$ in Barendregt et al. [1983], for $\mathcal{T} = \text{Scott}$ in Ronchi Della Rocca [1988], for $\mathcal{T} = \text{CDZ}$ in Coppo et al. [1987], for $\mathcal{T} = \text{AO}$ in Abramsky and Ong [1993], for $\mathcal{T} = \text{Park}$ and $\mathcal{T} = \text{HR}$ in Honsell and Ronchi Della Rocca [1992].

19.4. Applications of the approximation theorem

As discussed in Section 18.2 type theories give rise in a natural way to *filter* λ*-models*. Properties of $\mathcal{F}^{\mathcal{T}}$ with $\mathcal{T} \in \{\text{Scott}, \text{CDZ}, \text{AO}, \text{BCD}, \text{Park}, \text{HR}\}$ can be easily derived using Theorem 19.3.17. For instance, one can check the following.

- The models $\mathcal{F}^{\text{Scott}}, \mathcal{F}^{\text{CDZ}}, \mathcal{F}^{\text{DHM}}, \mathcal{F}^{\text{BCD}}$ are sensible.
- The top element in $\mathcal{F}^{\text{AO}}$ is the interpretation of the terms of order ∞.
- The model $\mathcal{F}^{\text{Park}}$ characterizes the terms reducible to closed terms.
- The model $\mathcal{F}^{\text{HR}}$ characterizes the terms reducible to λI-terms.

The rest of this section is devoted to the proof of these properties. Other uses of the Approximation Theorem can be found in the corresponding relevant papers, i.e. Barendregt et al. [1983], Coppo et al. [1987], Ronchi Della Rocca [1988], Abramsky and Ong [1993], Honsell and Ronchi Della Rocca [1992].

19.4.1. THEOREM. *The models $\mathcal{F}^{\mathcal{T}}$ with $\mathcal{T} \in \{\text{Scott}, \text{CDZ}, \text{DHM}, \text{BCD}\}$ are sensible, i.e. for all unsolvable terms M, N one has*

$$[\![M]\!]_\rho^{\mathcal{F}^{\mathcal{T}}} = [\![N]\!]_\rho^{\mathcal{F}^{\mathcal{T}}}.$$

PROOF. It follows immediately from Corollary 19.3.18 of the Approximation Theorem, the fact that $\bot$ is the only approximant of an unsolvable term for the mapping $\Box_\bot$, and Proposition 19.3.5(i). ■

Let us recall the definition of term of order ∞, see Longo [1983].

19.4.2. DEFINITION. An untyped lambda term M is *of order* ∞ iff

$$\forall n \exists M'. M \twoheadrightarrow_\beta \lambda x_1 \ldots \lambda x_n. M'.$$

19.4.3. THEOREM. *Let M be an untyped lambda term. Then the following are equivalent.*

(i) *M is of order ∞.*
(ii) *$\vdash_\cap^{\text{AO}} M : A$ for all types $A \in \mathbb{T}^{\text{AO}}$.*
(iii) *$[\![M]\!]_\rho^{\mathcal{F}^{\text{AO}}} = \top = \mathbb{T}^{\text{AO}} \in \mathcal{F}^{\text{AO}}$ for all valuations ρ.*

PROOF. Write $\vdash, \leq$ for $\vdash_\cap^{\text{AO}}, \leq_{\text{AO}}$, respectively. ((i)⇔(ii)). It is easy to check by structural induction on types (see Exercise **??**) that

$$\forall A \in \mathbb{T}^{\text{AO}} \exists n \in \mathbb{N}. (\mathtt{U}^n \to \mathtt{U}) \leq A.$$

So by the Approximation Theorem it suffices to show that, if $P \in \Lambda\bot$ is an approximate normal form, then we have

$$\vdash P : (\mathtt{U}^n \to \mathtt{U}) \iff P \equiv \lambda x_1 \ldots \lambda x_n. P' \text{ for some } P'.$$

(⇐) By axiom (U) and rule (→I). (⇒) Assume towards a contradiction that $P \equiv \lambda x_1 \ldots \lambda x_m. P'$ for $m < n$ and P' is of the form $\bot$ or $x\vec{P}$. Then by Theorem 16.1.9(iii)

$$\vdash P : (\mathtt{U}^n \to \mathtt{U}) \Rightarrow \{x_1\, \mathtt{U}, \ldots, x_m\, \mathtt{U}\} \vdash P' : (\mathtt{U}^{n-m} \to \mathtt{U}).$$

But this latter judgment can neither be derived if $P' \equiv \bot$, by Proposition 19.3.5(i) and Lemma 15.1.25, nor if $P' \equiv x\vec{P}$, by Theorem 16.1.1(i) and (ii) and Lemma 15.1.25.

((ii)⇔(iii)). Suppose $\vdash M : A$ for all A. Then by Theorem 18.2.7

$$[\![M]\!]_\rho^{\mathcal{F}^{\text{AO}}} = \{A \mid \Gamma \vdash M : A \text{ for some } \Gamma \models \rho\} = \mathbb{T}^{\text{AO}},$$

for all ρ. This is the top element in $\mathcal{F}^{\mathrm{AO}}$. Conversely, if $[\![M]\!]^{\mathcal{F}^{\mathrm{AO}}}_{\rho} = \top = \mathbb{T}^{\mathrm{AO}}$, for all ρ, then take $\rho_0(x) = \uparrow\mathtt{U}$. Then $\Gamma_{\rho_0} = \{x\,\mathtt{U} \mid x \in \mathrm{Var}\}$. Hence

$$\begin{array}{rcll} \mathbb{T}^{\mathrm{AO}} & = & [\![M]\!]^{\mathcal{F}^{\mathrm{AO}}}_{\rho_0} & \\ & = & \{A \mid \Gamma \vdash M : A \text{ for some } \Gamma \models \rho_0\}, & \text{by Theorem 18.2.7,} \\ & = & \{A \mid \ \vdash M : A\}, & \text{by Exercise ??.} \end{array}$$

Therefore $\vdash M : A$ for all $A \in \mathbb{T}^{\mathrm{AO}}$. ■

We denote by $\Lambda_{\downarrow\Lambda^{\emptyset}}$ the set of terms which reduce to a closed term.

19.4.4. Theorem. *A term $M \in \Lambda_{\downarrow\Lambda^{\emptyset}}$ iff $\vdash^{\mathrm{Park}}_{\cap} M : 0$.*

Proof. By the Approximation Theorem it suffices to check that if $P \in \Lambda\Phi$ is an anf and $\mathcal{V}$ is a finite set of term variables:

$$\{x\,0 \mid x \in \mathcal{V}\} \vdash^{\mathrm{Park}}_{\cap} P : 0 \text{ iff } FV(P) \subseteq \mathcal{V}.$$

($\Leftarrow$) By an easy induction on P, using that $0 = 0{\to}0$.

($\Rightarrow$) By induction on P. Lemma 15.1.24 shows $\vec{B}{\to}0 \neq \mathtt{U}$.

Case $P \equiv \lambda y.P'$. By Theorem 16.1.9(iii) and the induction hypothesis for P'.

Case $P \equiv y\vec{P}$. By Theorem 16.1.9(ii) we have $\Gamma \vdash^{\mathrm{Park}}_{\cap} y : \vec{B}{\to}0$ and $\Gamma \vdash^{\mathrm{Park}}_{\cap} \vec{P} : \vec{B}$. Hence by Theorem 16.1.9(i) one has $y \in \mathcal{V}$ and $0 \leq \vec{B}{\to}0$. By β-soundness and $0 = 0{\to}0$ we get $B_i \leq 0$. Thus $\Gamma \vdash^{\mathrm{Park}}_{\cap} P_i : 0$ and hence $\mathrm{FV}(P_i) \subseteq \mathcal{V}$, by the induction hypothesis.

Case $P \equiv \Phi\vec{P}$. Similar to the previous case. ■

Lastly we work out the characterization of terms reducible to $\lambda\mathsf{I}$-terms.

19.4.5. Lemma. *Let $m > 0$. Then*

$$1 \leq_{\mathrm{HR}} A_1 \to \ldots A_m \to 0 \;\Rightarrow\; [A_i = 0, \text{ for some } i].$$

Proof. By induction on m.

Case $m = 1$. Then $1 = (1 \to 1) \cap (0 \to 0) \leq A_1 \to 0$. By β-soundness, the only possible case is $A_1 \leq 0$. Since 0 is the least element, we have that $A_1 = 0$.

Case $m > 1$. Similarly, using the induction hypothesis. ■

19.4.6. Notation. Write $\Gamma^P_1 = \{x\,1 \mid x \in FV(P)\}$.

19.4.7. Lemma. *There is no anf P such that $\Gamma^P_1 \vdash^{\mathrm{HR}}_{\cap} P : 0$.*

Proof. Suppose towards a contradiction there exists such a P. By induction on P, using the Inversion Lemmas, the result will be shown.

Case $P \equiv \lambda z.P'$. Then $\Gamma \vdash P : 0$ implies $\Gamma, z\,1 \vdash P' : 0$ and the induction hypothesis applies.

Case $P \equiv P_0 P_1 ... P_m$, where P_0 is either a variable z or Φ. Then

$$\Gamma \vdash P_0 : A_1 \to \ldots A_m \to 0 \tag{19.8}$$

and $\Gamma \vdash P_i : A_i$ for some A_i. Then $1 \leq A_1 \to \ldots A_m \to 0$, by (19.8). By Lemma 19.4.5, there exists an i such that $A_i = 0$. Then $\Gamma \vdash P_i : 0$. By the induction hypothesis, this is impossible. ■

19.4.8. Lemma. *Let $P \in \Lambda\Phi$ be an anf. If $\Gamma_1^P \vdash^{\mathrm{HR}} P : 1$, then P is a λI-term.*

Proof. By induction on P.

Case $P \equiv \lambda z.P'$. By Inversion Theorem 16.1.9(iii), $\Gamma_1^{P'} \vdash P' : 1$ and $\Gamma_1^{P'} \uplus z : 0 \vdash P' : 0$. By the induction hypothesis, we have that P' is a λI-term. It remains to prove that $z \in FV(P')$. Suppose that $z \notin FV(P')$. Then we could remove it from the context and get $\Gamma_1^{P'} \vdash P' : 0$, contradicting Lemma 19.4.7.

Case $P \equiv P_0 P_1 \ldots P_n$ where $P_0 = x$ or $P_0 = \Phi$. By the Inversion Theorem 16.1.9 (ii), $\Gamma_1^P \vdash P_0 : A_1 \to \ldots \to A_n \to 1$ and $\Gamma_1^P \vdash P_i : A_i$ for all $i > 0$. By the Inversion Theorem 16.1.9(i) for $P_0 = x$ or Proposition 19.3.5(iii) for $P_0 = \Phi$, we get $1 \leq A_1 \to \ldots \to A_n \to 1$. Since $0 \leq 1$ and $0 = 1 \to 0$, we get that $1 \to \ldots \to 1 \to 0 \leq A_1 \to \ldots \to A_n \to 1$. So $A_i \leq 1$, by β-soundness. Hence, by rules (*strengthening*) and $(\leq)$, $\Gamma_1^{P_i} \vdash P_i : 1$ for all $i > 0$. Therefore, by the induction hypothesis, all P_i's are λI-terms. ■

19.4.9. Lemma. *Let $P \in \Lambda\Phi$ be an anf.*

(i) *If P is a λI-term, then $\Gamma_1^P \vdash^{\mathrm{HR}} P : 1$.*
(ii) *If P is a λI-term and $x \in FV(P)$, then $\Gamma_1^P \uplus \{x : 0\} \vdash^{\mathrm{HR}} P : 0$.*

Proof. By simultaneous induction on P.

(i) Case $P \equiv y$. Trivial.

Case $P \equiv \lambda z.P'$. If P is a λI-term, so is P' and $z \in FV(P')$. By the induction hypothesis (i), we have that $\Gamma_1^{P'} \vdash P' : 1$. By the induction hypothesis (ii), we have that $\Gamma_1^{P'} \uplus \{z : 0\} \vdash P' : 0$. Since $1 \cap 0 = 0$ and $1 = (1 \to 1) \cap (0 \to 0)$ we get $\Gamma_1^P \vdash P : 1$, by rules $(\to\mathrm{I})$,$(\cap\mathrm{I})$, $(\leq)$.

Case $P = P'P''$. Then $\Gamma_1^{P'} \vdash P' : 1 \leq 1 \to 1$ and $\Gamma_1^{P''} \vdash P'' : 1$, by the induction hypothesis (i). Hence $\Gamma_1^P \vdash P : 1$, by rules (*weakening*), $(\leq)$ and $(\to\mathrm{E})$.

(ii) Case $P \equiv x$. Trivial.

Case $P \equiv \lambda z.P'$. As $x \in FV(P)$, also $x \in FV(P')$. Then

$$\Gamma_1^{P'} \uplus \{x : 0\} \vdash P' : 0,$$

by the induction hypothesis (ii). By rules $(\to\mathrm{I})$, $(\leq)$ with $1 \to 0 = 0$ we get $\Gamma_1^P \uplus \{x : 0\} \vdash P : 0$.

Case $P \equiv P'P''$. We have two subcases.

Subcase $x \in FV(P')$. By the induction hypothesis (ii) it follows that

$$\Gamma_1^{P'} \uplus \{x : 0\} \vdash P' : 0 = 1 \to 0.$$

By the induction hypothesis (i) $\Gamma_1^{P''} \vdash P'' : 1$. Hence by (*weakening*) and ($\to$E) we conclude $\Gamma_1^P \uplus \{x : 0\} \vdash P : 0$.

Subcase $x \in FV(P'')$. Again $\Gamma_1^{P''} \uplus \{x : 0\} \vdash P'' : 0$. By the induction hypothesis (i) we have $\Gamma_1^{P'} \vdash P' : 1 \leq 0 \to 0$. Hence by (*weakening*) and ($\to$E) we conclude $\Gamma_1^P \uplus \{x : 0\} \vdash P : 0$. ■

Now we can characterize the set $\Lambda_{\downarrow\Lambda^{\mathsf{I}}}$ of terms which reduce to $\lambda\mathsf{I}$-terms in a type theoretic way.

19.4.10. Theorem. *Let M be a lambda term. Then*

$$M \in \Lambda_{\downarrow\Lambda^{\mathsf{I}}} \iff \Gamma_1^M \vdash_{\cap}^{\mathrm{HR}} M : 1.$$

Proof. By Theorem 19.3.17 the types of M are all and only the types of its approximants. By Lemmas 19.4.8 and 19.4.9(i) an anf is a $\lambda\mathsf{I}$-term iff it has the type 1 in HL from the basis which gives type 0 to all its free variables. We conclude observing that if $\Box_\Phi(M)$ is a $\lambda\mathsf{I}$-term, then M is a $\lambda\mathsf{I}$-term as well. ■

Theorem 19.4.10 was first proved in Honsell and Ronchi Della Rocca [1992] by purely semantic means.

The following results are translations of the results in Theorems 19.4.4, 19.4.10 and 19.1.15 to filter models.

19.4.11. Proposition. *Let $M \in \Lambda^\emptyset$ be a closed lambda term. Then*

(i) *M has a normal form* $\iff$ $[\![M]\!]^{D_\infty^{\mathrm{CDZ}}} \sqsupseteq 1$.

(ii) *M is solvable* $\iff$ $[\![M]\!]^{D_\infty^{\mathrm{DHM}}} \sqsupseteq 1$.

(iii) *M reduces to a $\lambda\mathsf{I}$-term* $\iff$ $[\![M]\!]^{D_\infty^{\mathrm{HR}}} \sqsupseteq 1$.

(iv) *M is strongly normalizing* $\iff$ $[\![M]\!]^{\mathcal{F}^{\mathrm{CDV}}} \neq \emptyset$.

Let $M \in \Lambda$ be an (open) lambda term. Then

(v) *M reduces to a closed term* $\iff$ $[\![M]\!]_\rho^{D_\infty^{\mathrm{Park}}} \sqsupseteq 0$, *for all ρ.*

Proof. (i)-(ii) We explain the situation for (i), the other case being similar.

$$\begin{array}{lll}
M \text{ has a nf} & \iff \vdash_{\cap}^{\mathrm{CDZ}} M : 1, & \text{by Theorem 19.1.15(i),} \\
& \iff [\![M]\!]^{\mathcal{F}^{\mathrm{CDZ}}} \ni 1, & \text{by 18.2.7,} \\
& \iff [\![M]\!]^{\mathcal{F}^{\mathrm{CDZ}}} \sqsupseteq \uparrow 1, & \text{by the definition of filters,} \\
& \iff [\![M]\!]^{D_\infty^{\mathrm{CDZ}}} \sqsupseteq \hat{\mathsf{m}}(\uparrow 1) = \mathsf{m}(1), & \text{by Theorem 18.3.22,} \\
& \iff [\![M]\!]^{D_\infty^{\mathrm{CDZ}}} \sqsupseteq \Phi_{0,\infty}(1), & \text{since } 1 \in D_0, \\
& \iff [\![M]\!]^{D_\infty^{\mathrm{CDZ}}} \sqsupseteq 1, &
\end{array}$$

by the identification of D_0 as a subset of D_∞.

(iii) Similarly, using Theorem 19.4.10.

(iv) As (i), but simpler as the step towards D_∞ is not made. Notice that $\mathcal{F}^{\mathrm{CDV}}$ gives rise to a $\lambda\mathsf{I}$-model, not to a λ-model.

(v) Similarly, using Theorem 19.4.4. ■

We do not have a characterization like (i) in the Proposition for D_∞^{Scott}, as it was shown in Wadsworth [1976] that there is a closed term J without a normal form such that $D_\infty^{\text{Scott}} \models \mathsf{I} = \mathsf{J}$.

19.5. Undecidability of inhabitation

In this section we consider type theories with infinitely many type atoms, as described in Section 15.1. To fix ideas, we are concerned here with the theory $\mathcal{T} = \text{CDV}$. Since we do not consider other type theories, in this section the symbols $\vdash$ and $\leq$ stand for $\vdash_\cap^{\text{CDV}}$ and $\leq_{\text{CDV}}$, respectively. Moreover $\mathbb{T} = \mathbb{T}^{\text{CDV}}$.

We investigate the *inhabitation problem* for this type theory, which is to determine, for a given a type A, if there exists a closed term of type A (the inhabitant). In symbols, the problem can be presented as follows:

$$\vdash ? : A$$

A slightly more general variant of the problem is the inhabitation problem *relativized* to a given context Γ:

$$\Gamma \vdash ? : A$$

It is however not difficult to show that these two problems are equivalent.

19.5.1. Lemma. *Let $\Gamma = \{x_1\, A_1, \ldots, x_n\, A_n\}$. Then the following are equivalent.*

1. *There exists a term $M \in \Lambda$ such that $\Gamma \vdash M : A$.*
2. *There exists a term $N \in \Lambda$ such that $\vdash N : A_1 \to \cdots \to A_n \to A$.*

Proof. (1) $\Rightarrow$ (2) Define $N \equiv \lambda x_1 \ldots x_n.\, M$. Apply n times rule ($\to$I).
(2) $\Rightarrow$ (1) Take $M \equiv N x_1 \cdots x_n$ and apply n times ($\to$E) and (weakening). ■

The main result of the present section (Theorem 19.5.30) is that type inhabitation is undecidable for $\mathcal{T} = \text{CDV}$. Compare this to Statman [1979], stating that for simple types the problem is decidable in polynomial space.

By Theorem 19.1.15 and Corollary 16.2.3 we need to consider only inhabitants in normal form. The main idea of the undecidability proof is based on the following observation. The process of solving an instance of the inhabitation problem can be seen as a certain (solitary) game of building trees. In this way, one can obtain a combinatorial representation of the computational contents of the inhabitation problem (for a restricted class of types). We call this model a "tree game". In order to win a tree game, the player may be forced to execute a computation of a particular automaton (a "typewriter automaton", TWA). Thus, the global strategy of the proof is as follows. We make the following abbreviations.

EQA := Emptiness Problem for Queue Automata;

ETW := Emptiness Problem for Typewriter Automata;

WTG := Problem of determining whether one can Win a Tree Game;

IHP := Inhabitation Problem in $\lambda_\cap^{\text{CDV}}$.

If we write $P_1 \leq_T P_2$, this means that problem P_1 is (Turing) reducible to P_2, and hence that undecidability of P_1 implies that of P_2. It is well known that EQA is undecidable, see e.g. Kozen [1997]. The following inequalities show that IHP is undecidable.

EQA	$\leq_T$	ETW	(Lemma 19.5.25);
ETW	$\leq_T$	WTG	(Proposition 19.5.29);
WTG	$\leq_T$	IHP	(Corollary 19.5.23).

Basic properties

We begin with some basic observations concerning the relation $\leq$.

19.5.2. LEMMA. *Let $n > 0$.*

(i) *Let $\alpha \in \mathbb{A}_\infty$, and none of the $A_1, \ldots, A_n \in \mathbb{T}$ be an intersection. Then*

$$A_1 \cap \ldots \cap A_n \leq \alpha \;\Rightarrow\; \exists i.\alpha \equiv A_i.$$

(ii) *Let $\alpha_1, \ldots, \alpha_n \in \mathbb{A}_\infty$ and $A \in \mathbb{T}$ be not an intersection. Then*

$$\alpha_1 \cap \ldots \cap \alpha_n \leq A \;\Rightarrow\; \exists i.A \equiv \alpha_i.$$

(iii) *Let $\alpha_1, \ldots, \alpha_n \in \mathbb{A}_\infty$ and $A \in \mathbb{T}$. Then*

$$\alpha_1 \cap \ldots \cap \alpha_n \leq A \;\Rightarrow\; A \equiv \alpha_{i_1} \cap \ldots \cap \alpha_{i_k},$$

for some $k \geq 0$ and $1 \leq i_1 < \ldots < i_k \leq n$.

PROOF. (i), (ii) Exercise 19.6.18.

(iii) Let $A \equiv B_1 \cap \ldots \cap B_k$ with $k > 0$ and the B_j not intersections. Then for each j one has $\alpha_1 \cap \ldots \cap \alpha_n \leq B_j$ and can apply (ii) to show that $B_j \equiv \alpha_{i_j}$. ■

19.5.3. LEMMA. *Let $A_1, \ldots, A_n, B \in \mathbb{T}$ and $\alpha_1, \ldots, \alpha_n, \beta \in \mathbb{A}_\infty$, with $n > 0$. Then*

$$(A_1 \to \alpha_1) \cap \ldots \cap (A_n \to \alpha_n) \leq (B \to \beta) \;\Rightarrow\; \exists i.\beta \equiv \alpha_i \;\&\; B \leq A_i.$$

PROOF. By Theorem 16.1.7 CDV is β-sound. Hence the assumption implies that $B \leq (A_{i_1} \cap \ldots \cap A_{i_k})$ and $(\alpha_{i_1} \cap \ldots \cap \alpha_{i_k}) \leq \beta$. By Lemma 19.5.2(ii) one has $\beta \equiv \alpha_{i_p}$, for some $1 \leq p \leq k$, and the conclusion follows. ■

19.5.4. LEMMA. *If $\Gamma \vdash \lambda x.M : A$ then $A \notin \mathbb{A}_\infty$.*

PROOF. Suppose $\Gamma \vdash \lambda x.M : \alpha$. By Lemma 16.1.1(iii) it follows that there are $n > 0$ and $B_1, \ldots, B_n, C_1, \ldots, C_n$ such that $\Gamma, x : B_i \vdash M : C_i$, for $1 \leq i \leq n$, and $(B_1 \to C_1) \cap \ldots \cap (B_n \to C_n) \leq \alpha$. This is impossible by Lemma 19.5.2(i). ■

Game contexts

In order to prove that a general decision problem is undecidable, it is enough to identify a "sufficiently difficult" fragment of the problem and prove undecidability of that fragment. Such an approach is often useful. This is because restricting the consideration to specific instances may simplify the analysis of the problem. Of course the choice should be done in such a way that the "core" of the problem remains within the selected special case. This is the strategy we are applying for our inhabitation problem. Namely, we restrict our analysis to the following special case of relativized inhabitation.

$$\Gamma \vdash ? : \alpha,$$

where α is a type atom, and Γ is a "game context", the notion of game context being defined as follows.

19.5.5. Definition. (i) If $\mathcal{X}, \mathcal{Y} \subseteq \mathbb{T}$ are sets of types, then

$$\begin{array}{rcl} \mathcal{X}{\to}\mathcal{Y} & = & \{X{\to}Y \mid X \in \mathcal{X},\ Y \in \mathcal{Y}\}. \\ \mathcal{X} \cap \mathcal{Y} & = & \{X \cap Y \mid X \in \mathcal{X},\ Y \in \mathcal{Y}\}. \\ \mathcal{X}^{\cap} & = & \{A_1 \cap \ldots \cap A_n \mid n \geq 1 \ \&\ A_1, \ldots, A_n \in \mathcal{X}\}. \end{array}$$

If $A \equiv A_1 \cap \ldots \cap A_n$, and each A_i is not an intersection, then the A_i are called the *components* of A.

(ii) We consider the following sets of types.

(1) $\mathcal{A} = \mathbb{A}_{\infty}^{\cap}$.

(2) $\mathcal{B} = (\mathbb{A}_{\infty}{\to}\mathbb{A}_{\infty})^{\cap}$.

(3) $\mathcal{C} = (\mathcal{D}{\to}\mathbb{A}_{\infty})^{\cap}$.

(4) $\mathcal{D} = (\mathcal{B}{\to}\mathbb{A}_{\infty}) \cap (\mathcal{B}{\to}\mathbb{A}_{\infty})$.

(iii) Types in $\mathcal{A} \cup \mathcal{B} \cup \mathcal{C}$ are called *game types*.

(iv) A type context Γ is a *game context* iff all types in Γ are game types.

We show some properties of type judgements involving game types.

19.5.6. Lemma. *For a game context Γ the following hold.*

(i) $\Gamma \vdash xM : A \ \Rightarrow\ A \in \mathcal{A}\ \&\ \Gamma(x) \equiv (E_1{\to}\alpha_1) \cap \ldots \cap (E_n{\to}\alpha_n),\ n > 0,$
$\&\ A \equiv \alpha_{i_1} \cap \ldots \cap \alpha_{i_k}\ \&\ \Gamma \vdash M : E_{i_1} \cap \ldots \cap E_{i_k},$
for some $k > 0\ \&\ 1 \leq i_1 < \ldots < i_k \leq n$.

(ii) $\Gamma \vdash xM : \alpha \ \Rightarrow\ \Gamma(x) \equiv (E_1{\to}\alpha_1) \cap \ldots \cap (E_n{\to}\alpha_n),\ n > 0,$
$\&$ *for some* $1 \leq i \leq n.[\alpha \equiv \alpha_i\ \&\ \Gamma \vdash M : E_i]$.

(iii) $\Gamma \not\vdash xMN : A$.

Proof. (i) Suppose $\Gamma \vdash xM : A$. By Lemma 16.1.9(ii) we have for some type B that $\Gamma \vdash x : (B{\to}A)$ and $\Gamma \vdash M : B$. Then $\Gamma(x) \leq B \to A$, by Lemma 16.1.1(i). By Lemma 19.5.2(ii), this cannot happen if $\Gamma(x)$ is in $\mathcal{A}$. Thus $\Gamma(x)$, being a game type, is of the form $(E_1{\to}\alpha_1) \cap \ldots \cap (E_n{\to}\alpha_n)$. Then,

$$(E_1{\to}\alpha_1) \cap \ldots \cap (E_n{\to}\alpha_n) \equiv \Gamma(x) \leq (B{\to}A),$$

Since CDV is β-sound and by Lemma 19.5.2(iii), we have $B \leq E_{i_1} \cap \ldots \cap E_{i_k}$ and $\alpha_{i_1} \cap \ldots \cap \alpha_{i_k} \equiv A$, for some $k > 0$ and i_j such that $1 \leq i_j \leq n$.

(ii) By (i) and Lemma 19.5.2(ii).
(iii) By (i), using that $B{\to}A \neq \alpha_{i_1} \cap \ldots \cap \alpha_{i_k}$, by Lemma 19.5.2(ii). ■

19.5.7. LEMMA. *If A is a game type and $D \in \mathcal{D}$, then $A \not\leq D$.*

PROOF. Suppose $A \leq D \leq (B{\to}\alpha)$, with $B \in \mathcal{B}$. The case $A \in \mathcal{A}$, is impossible by Lemma 19.5.2(ii). If $A \in \mathcal{B}$, then $(\alpha_1{\to}\beta_1){\cap}\ldots{\cap}(\alpha_n{\to}\beta_n) \leq B{\to}\alpha$ and hence $B \leq \alpha_i$ for some i, by Lemma 19.5.3. By Lemma 19.5.2(i) this is also impossible. If $A \in \mathcal{C}$, then $(D_1{\to}\beta_1) \cap \ldots \cap (D_n{\to}\beta_n) \leq B{\to}\alpha$ and hence $B \leq D_i \in \mathcal{D}$ for some i, by Lemma 19.5.3. We have already shown that this is impossible. ■

For game contexts the Generation Lemma 16.1.9 can be extended as follows.

19.5.8. LEMMA. *Let Γ be a game context, and let M be in normal form.*
(i) *If $\Gamma \vdash M : (B_1 \to \alpha_1) \cap (B_2 \to \alpha_2) \in \mathcal{D}$, with $B_i \in \mathcal{B}$, then $M \equiv \lambda y.N$, and $\Gamma, y\, B_i \vdash N : \alpha_i$ for $i = 1, 2$.*
(ii) *If $\Gamma \vdash M : \alpha$, with $\alpha \in \mathbb{A}_\infty$, then there are two exclusive possibilities.*

- *M is a variable z and $\Gamma(z)$ is in $\mathcal{A}$, where α is one of the components.*
- *$M \equiv xN$, where $\Gamma(x) \equiv (E_1 \to \beta_1) \cap \cdots \cap (E_n \to \beta_n)$, and $\alpha \equiv \beta_i$ and $\Gamma \vdash N : E_i$, for some $1 \leq i \leq n$.*

PROOF. (i) Notice first that by Lemma 19.5.6 the term M cannot be an application. If it is a variable x, then $\Gamma(x) \leq (B_1 \to \alpha_1) \cap (B_2 \to \alpha_2)$, by Lemma 16.1.1(i). This contradicts Lemma 19.5.7, because $\Gamma(x)$ is a game type. It follows that $M = \lambda y.N$ and $\Gamma \vdash \lambda y.N : (B_i{\to}\alpha_i)$. Then for $i = 1, 2$ one has $\Gamma, y\, B_i \vdash N_i : \alpha_i$, by Lemma 16.1.9(iii).

(ii) M is not an abstraction by Lemma 19.5.4. If $M \equiv z$, then $\Gamma(z) \leq \alpha$ and $\Gamma(z)$ is a game type, i.e. in $\mathcal{A}{\cup}\mathcal{B}{\cup}\mathcal{C}$. By Lemma 19.5.2(i) one has $\Gamma(z) \in \mathcal{A}$, with α as one of the components. If M is an application, $M \equiv zM_1 \ldots M_m$, $m > 0$, write $\Gamma(z) = (E_1 \to \beta_1){\cap}\cdots{\cap}(E_n \to \beta_n)$, as it is a game type and by Theorem 16.1.9(ii) it cannot be in $\mathcal{A}$. Then $M \equiv zN$, by Lemma 19.5.6(iii). By (ii) of the same Lemma one has $\alpha \equiv \beta_i$ and $\Gamma \vdash N : E_i$ for some i. ■

Tree games

In order to show the undecidability of inhabitation for CDV we will introduce a certain class of *tree games*. 'Rounds' in a tree game are an intermediate step in our construction. The idea of a tree game is to represent, in an abstract way, the crucial combinatorial behaviour of proof search in CDV. We will first show how inhabitation problems can be represented by tree games and then how tree games can represent computations of certain machines called typewriter automata (TWA).

19.5.9. DEFINITION. Let Σ be a finite alphabet; its elements are called *labels*.

1. A *local move* (over Σ) is a finite nonempty set B of pairs of labels.

2. A *global move* (over Σ) is a finite nonempty set C of triples of the form

$$\langle\, \langle\, X, b\,\rangle, \langle\, Y, c\,\rangle, d\,\rangle,$$

where $b, c, d \in \Sigma$ and X, Y are local moves.

3. A *tree game* (over Σ) is a triple of the form

$$G = \langle\, a, A, \{C_1, \ldots, C_n\}\,\rangle,$$

where $a \in \Sigma$, $A \subseteq \Sigma$ and $C_1, \ldots, C_n$ are global moves. We call a the *initial label* and A the set of *final labels*.

Before we explain the rules of the game, we give an interpretation of the constituents of the tree games in terms of types.

19.5.10. Definition. Let Σ be a finite subset of $\mathbb{A}_\infty$, the infinite set of type atoms, and let G be a tree game over Σ. Moves of G, and the set of final labels, can be interpreted as types of CDV as follows.

1. If $A = \{a_1, \ldots, a_n\}$, then $\tilde{A} = a_1 \cap \ldots \cap a_n$.
2. If $B = \{\langle\, a_1, b_1\,\rangle, \ldots, \langle\, a_n, b_n\,\rangle\}$, then $\tilde{B} = (a_1 \to b_1) \cap \ldots \cap (a_n \to b_n)$.
3. If $C = \{\langle\, \langle\, B_1, b_1\,\rangle, \langle\, B_1', b_1'\,\rangle, c_1\,\rangle, \ldots, \langle\, \langle\, B_n, b_n\,\rangle, \langle\, B_n', b_n'\,\rangle, c_n\,\rangle\}$, then

$$\tilde{C} = (((\tilde{B_1} \to b_1) \cap (\tilde{B_1'} \to b_1')) \to c_1) \cap \ldots \cap (((\tilde{B_n} \to b_n) \cap (\tilde{B_n'} \to b_n')) \to c_n).$$

Notice that $\tilde{A} \in \mathcal{A}, \tilde{B} \in \mathcal{B}$ and $\tilde{C} \in \mathcal{C}$.

A tree game is a solitary game, i.e., there is only one player. Starting from an initial position, the player can nondeterministically choose a sequence of moves, and wins if (s)he can manage to reach a final position. Every position (configuration) of the game is a finite labelled tree, and at every step the depth of the tree is increasing.

19.5.11. Definition. Let $G = \langle a, A, \{C_1, \ldots, C_n\}\rangle$ be a tree game over Σ. A *position* T of G is a finite labelled tree, satisfying the following conditions.

- The root is labelled by the initial symbol a;
- Every node has at most two children;
- Nodes at the same level (the same distance from the root) have the same number of children (in particular all leaves are at the same level);
- All nodes are labelled by elements of Σ;
- In addition, if a node v has two children v' and v'', then the branches $\langle\, v, v'\,\rangle$ and $\langle\, v, v''\,\rangle$ are labelled by local moves.

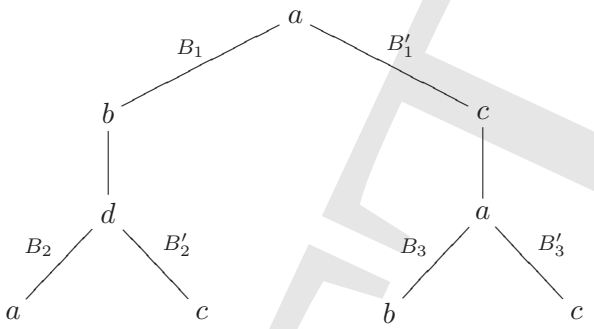


Figure 19.2: An example position

19.5.12. DEFINITION. Let $G = \langle a, A, \{C_1, \ldots, C_n\}\rangle$ be a tree game.

1. The *initial position* of G is the tree with a unique node labelled a.
2. A position T is *winning* iff all labels of the leaves of T are in A.

19.5.13. DEFINITION. Let T be a position in a game $G = \langle a, A, \{C_1, \ldots, C_n\}\rangle$, and let v be a leaf of T. Let k be such that all nodes in T at level $k-1$ have two children as shown in Figure 19.3. There is a node u at level $k-1$ which is an ancestor of v, one of the children of u, say u', is also an ancestor of v (possibly improper, i.e., it may happen that $u' = v$). Assume that B is the label of the branch $\langle u, u' \rangle$. Then we say that B is the *k-th local move associated to v*, and we write $B = B_{v,k}$.

Now we can finally describe the rules of the game.

19.5.14. DEFINITION. (i) Let $G = \langle a, A, \{C_1, \ldots, C_n\} \rangle$ and let T be a (current) position in G. There are two possibilities to obtain a next position.

(1) The player can perform a "global" step, by first selecting one of the global moves C_i and then performing the following actions for each leaf v of T.
   - Choose a triple $\langle \langle B, b \rangle, \langle B', b' \rangle, c \rangle \in C_i$ such that c is the label of v;
   - Create two children of v, say v' and v'', labelled b and b', respectively;
   - Label the branch $\langle v, v' \rangle$ by B and the branch $\langle v, v'' \rangle$ by B'.

   The step is only allowed if the resulting tree is *legal*, i.e. satisfies the conditions of definition 19.5.11.

(2) The player can also perform a "local" step. This begins with a choice of a level $k > 0$ of T such that each node at level $k-1$ has two children. Then, for each leaf v of T, the player executes the following actions.
   - Choose a pair $\langle a, b \rangle \in B_{v,k}$ such that b is the label of v;
   - Create a single child of v, labelled a.

   Again the step is only allowed the appropriate actions can be performed at every leaf, otherwise the resulting tree is not a position.

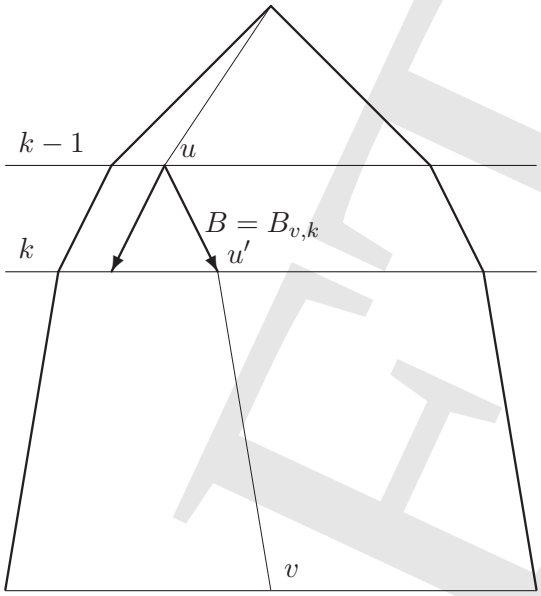


Figure 19.3: Local move X associated to node v

(ii) If a position T' is reachable from T with help of one global step C_i, we write

$$T \Rightarrow^{C_i} T'.$$

If T' is obtained from T by a local step defined via level k, we write

$$T \Rightarrow^{k} T'.$$

If T' is reachable from T in one step (global or local), then we write $T \Rightarrow T'$.

(iii) A position T in G is called *favorable* iff there is a position T' with

$$T \Rightarrow_T^* T' \text{ and } T' \text{ is winning,}$$

where $\Rightarrow_T^*$ is the reflexive transitive closure of $\Rightarrow_T$.

(iv) The game G *can be won by the player*, notation $\mathrm{sol}(G)$, iff the initial position is favorable.

The following example gives an idea of the tree games and moreover it is important for our principal construction.

19.5.15. Example. Consider the tree game $G_0 = \langle 1, \{c\}, \{C_1, C_2\} \rangle$, over the alphabet $\Sigma = \{1, 2, a, b, c\}$, where

- $C_1 = \{\langle \langle \{\langle a, a\rangle\}, 1 \rangle, \langle \{\langle b, a\rangle\}, 2 \rangle, 1 \rangle, \langle \langle \{\langle a, b\rangle\}, 1 \rangle, \langle \{\langle b, b\rangle\}, 2 \rangle, 2 \rangle\}$;
- $C_2 = \{\langle \langle \{\langle c, a\rangle\}, a \rangle, \langle \{\langle c, a\rangle\}, a \rangle, 1 \rangle, \langle \langle \{\langle c, b\rangle\}, a \rangle, \langle \{\langle c, b\rangle\}, a \rangle, 2 \rangle\}$.

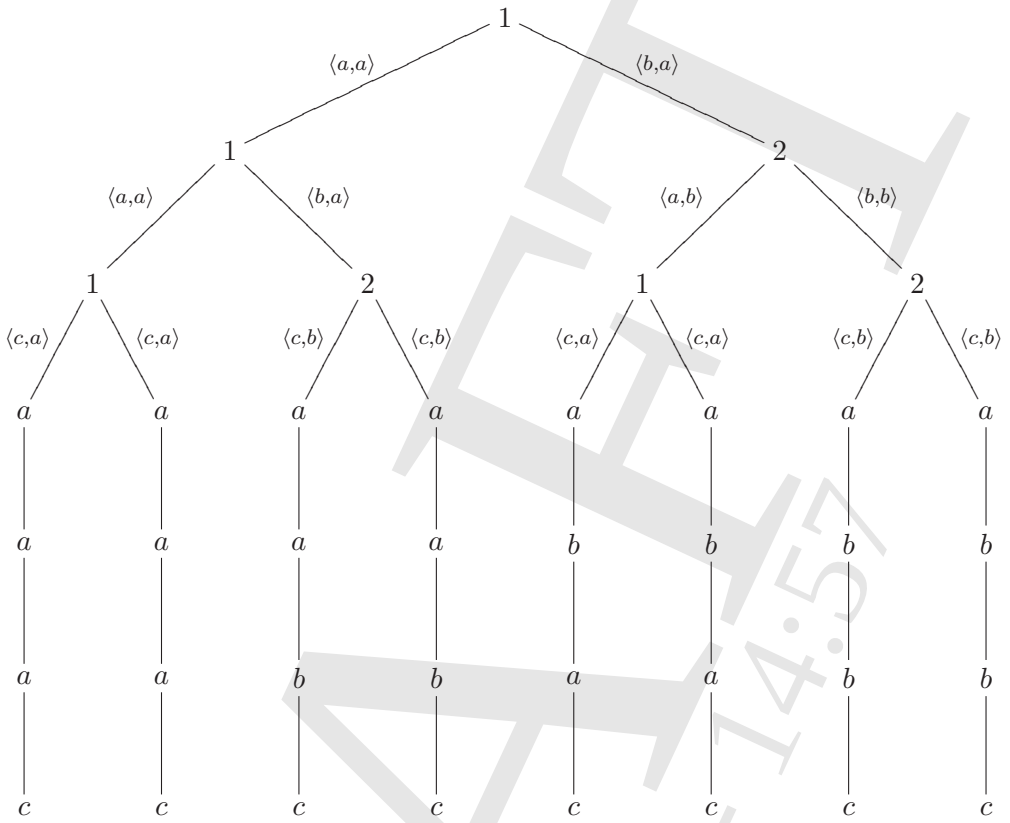


Figure 19.4: A winning position in G_0 of Example 19.5.15.

Figure 19.4 demonstrates a possible winning position T of the game. Note that this position can actually be reached from the initial one in 6 steps, so that the player can win G_0. These 6 steps are as follows

$$T_0 \Rightarrow^{C_1} T_1 \Rightarrow^{C_1} T_2 \Rightarrow^{C_2} T_3 \Rightarrow^{1} T_4 \Rightarrow^{2} T_5 \Rightarrow^{3} T_6 = T,$$

where each T_i is of depth i. The reader should observe that every sequence of steps leading from T_0 to a winning position must obey a similar pattern:

$$T_0 \Rightarrow^{C_1} T_1 \Rightarrow^{C_1} \cdots \Rightarrow^{C_1} T_{n-1} \Rightarrow^{C_2} T_n \Rightarrow^{1} T_{n+1} \Rightarrow^{2} \cdots \Rightarrow^{n} T_{2n},$$

where T_{2n} is winning. Thus, the game must consist of two phases: first a number of applications of C_1, then a single application of C_2 and then a sequence of steps using only local moves. What is important in our example is that the order of local steps is fully determined. Indeed, at the position T_{n+k-1} the only action possible is "$\Rightarrow^k$". That is, one must apply the k-th local moves associated to the leaves (the moves labelling branches at depth k). This is forced by the distribution of symbols a, b at depth $n + k - 1$.

Let us emphasize a few properties of our games. First, a game is a non-deterministic process, and there are various sequences of steps possible. We

can have winning sequences (reaching a winning position), and infinitely long sequences, but also "deadlocks" when no rule is applicable. Note that there are various levels of nondeterminism here: we can choose between C_i's and k's and then between various elements of the chosen set C_i (respectively $B_{v,k}$). It is an important property of the game that the actions performed at various leaves during a local step may be different, as different moves $B_{v,k}$ were "declared" before at the corresponding branches of the tree.

We now explain the relationship between tree games and term search in CDV. Since we deal with intersection types, it is not unexpected that we need sometimes to require one term to have many types, possibly within different contexts. This leads to the following definition.

19.5.16. Definition. Let $k, n > 0$. Let $\vec{A}_1, \ldots, \vec{A}_n$ be n sequences of k types:

$$\vec{A}_i = A_{i1}, \ldots, A_{ik}.$$

Let $\Gamma_i = \{x_1\ A_{i1}, \ldots, x_k\ A_{ik}\}$ and let $\alpha_i \in \mathbb{A}_\infty$, for $1 \leq i \leq n$. A *generalized inhabitation problem (gip)* is a finite set of pairs $P = \{\langle\, \Gamma_i, \alpha_i \,\rangle \mid i = 1, \ldots, n\}$, where each Γ_i and α_i are as above. A *solution* of P is a term M such that $\Gamma_i \vdash M : \alpha_i$ holds for each i. We say *'M solves P'*. This is equivalent to requiring that

$$\vdash \lambda\vec{x}.M : (\vec{A}_1 \to \alpha_1) \cap \ldots \cap (\vec{A}_n \to \alpha_n).$$

19.5.17. Definition. Let $G = \langle\, a, A, \{C_1, \ldots, C_n\} \,\rangle$ be a tree game and let T be a position in G.

(i) We have $\Gamma_G = \{x_0\ \tilde{A}, x_1\ \tilde{C}_1, \ldots, x_n\ \tilde{C}_n\}$.

(ii) Let J be the set of all numbers k such that every node in T at level $k-1$ has two children. We associate a new variable y_k to each $k \in J$. Define for a leaf v of T the basis

$$\Gamma_v = \{y_k\ \tilde{B}_{v,k} \mid k \in J\}.$$

(iii) Define the gip $P_T^G = \{\langle \Gamma_G \cup \Gamma_v, a_v \rangle \mid v \text{ is a leaf of } T \text{ with label } a_v\}$.

The following lemma states the exact correspondence between inhabitation and games. Let us make first one comment that perhaps may help to avoid confusion. We deal here with quite a restricted form of inhabitation problem (only game contexts), which implies that the lambda terms to be constructed have a "linear" shape (Exercise 19.6.20). Thus we have to deal with trees not because of the shape of lambda terms (as often happens with proof search algorithms) but exclusively because the nature of intersection types, and because we need to solve various inhabitation problem uniformly (i.e., to solve *gip*).

19.5.18. Lemma. *Let $G = \langle\, a, A, \{C_1, \ldots, C_n\} \,\rangle$ be a tree game.*

(i) *If T is a winning position of G, then x_0 solves P_T^G.*

(ii) *If $T_1 \Rightarrow^{C_i} T_2$ and N solves $P_{T_2}^G$, then $x_i(\lambda y_k.N)$ solves $P_{T_1}^G$, where k is the depth of T_1 and $i > 0$.*

(iii) *If $T_1 \Rightarrow^{k} T_2$ and N solves $P_{T_2}^G$, then $y_k N$ solves $P_{T_1}^G$.*

PROOF. (i) T is winning, hence $a_v \in A$ for each leaf v. Hence $x_0\ \tilde{A} \vdash x_0 : a_v$ and therefore $\Gamma_G \cup \Gamma_v \vdash x_0 : a_v$.

(ii) $T_1 \Rightarrow^{C_i} T_2$, hence each leaf v of T_1 has two children v' and v'' in T_2 and the branches $\langle v, v'\rangle$ and $\langle v, v''\rangle$ are labelled by B and B' such that $\langle\langle B, a_{v'}, a_v\rangle, \langle B', a_{v''}\rangle\rangle \in T_i$. Let $k = \text{level}(v)$. As N solves $P^G_{T_2}$ one has

$$\begin{array}{lll} \Gamma_G \cup \Gamma_v, y_k\ \tilde{B} & \vdash & N : a_{v'}; \\ \Gamma_G \cup \Gamma_v, y_k\ \tilde{B}' & \vdash & N : a_{v''}. \end{array}$$

So $\Gamma_G \cup \Gamma_v \vdash \lambda y_k.N : (\tilde{B} \to a_{v'}) \cap (\tilde{B}' \to a_{v''})$. Therefore $\Gamma_G \cup \Gamma_v \vdash x_i(\lambda y_k.N) : a_v$.

(iii) $T_1 \Rightarrow^k T_2$, hence each leaf v of T_1 has a child v' in T_2 such that $\langle a_{v'}, a_v\rangle \in B_{v,k}$. Then $y_k\ \tilde{B}_{v,k} \vdash y_k : a'_v \to a_v$ and $\Gamma_G \cup \Gamma_v \vdash N : a_{v'}$, by assumption. Therefore $\Gamma_G \cup \Gamma_v \vdash y_k N : a_v$. ■

19.5.19. COROLLARY. *Let G be a tree game. For positions T one has*

$$T \text{ is favorable } \Rightarrow\ P^G_T \text{ has a solution.}$$

PROOF. By induction on the number of steps needed to reach a winning position and the lemma. ■

For the converse we need the following result.

19.5.20. LEMMA. *Let T_1 be a position in a tree game G and let M be a solution in β-nf of $P^G_{T_1}$. Then we have one of the following cases.*

1. *$M \equiv x_0$ and T_1 is winning.*
2. *$M \equiv y_k N$ and N is the solution of $P^G_{T_2}$, for some T_2 with $T_1 \Rightarrow^k T_2$.*
3. *$M \equiv x_i(\lambda y_k.N)$ and N is the solution of $P^G_{T_2}$, for some T_2 with $T_1 \Rightarrow^{C_i} T_2$.*

PROOF. Case $M \equiv z$. As M is a solution of $P^G_{T_1}$, one has

$$\Delta_v = \Gamma_G \cup \Gamma_v \vdash z : a_v,$$

for all leaves v of T_1. Then by Lemma 16.1.1(i) one has $\Delta_v(z) \leq a_v$. Hence by Lemma 19.5.2(i) $a_v \in A$ and $z = x_0$. Therefore T_1 is winning.

Case M is an application. Then for all leaves v of T_1

$$\Delta_v = \Gamma_G \cup \Gamma_v \vdash M : a_v.$$

By Lemma 19.5.8(ii) $M \equiv zN$ and $\Delta_v(z) \equiv (E_1 \to \beta_1) \cap \cdots \cap (E_n \to \beta_n)$, with $\Delta_v \vdash N : E_j$ and $a_v = \beta_j$, for some j. Now choose a leaf v of T_1. As Δ_v is a game context there are only two possibilities

$$E_j \in \mathbb{A}_\infty \text{ or } E_j \in \mathcal{D}.$$

Subcase $E_j \in \mathbb{A}_\infty$. Let $E_j \equiv \alpha_j$. Now $z \notin \text{dom}(\Gamma_G)$, hence $z \in \text{dom}(\Gamma_v)$ and $z\ (\alpha_1 \to \beta_1) \cap \ldots \cap (\alpha_n \to \beta_n)$ being $y_k : \tilde{B}_{v,k}$, for some k. Also for each leaf w of T_1 one has $z \in \text{dom}(\Gamma_w)$ and $z\ \Gamma_w(z)$ is $y_k\ \tilde{B}_{w,k}$. Define $T_1 \Rightarrow^k T_2$ by giving

each leaf v of T_1 with label β_j a child v' with label α_j. We have $\Delta_v \vdash N : \alpha_j$ and $y_k\, \tilde{B}_{v,k} \vdash y_k : \alpha_j \to \beta_j$. Hence $\Delta_{v'} \vdash M : a_{v'}$.

Subcase $E_j \in \mathcal{D}$. Then $\Delta_v(z) \equiv (E_1 \to \beta_1) \cap \ldots \cap (E_n \to \beta_n)$. Hence $z : \Delta_v(z)$ is $x_i \tilde{C}_i$ for some i. So $z \in \mathrm{dom}(\Gamma_G)$ and therefore $\Delta_w(z) = \Delta_v(z)$ for all leaves w of T_1. Let $C_i = \{\ldots, \langle\langle B_j, \alpha_j\rangle, \langle B'_j \alpha'_j\rangle, \beta_j\rangle, \ldots\}$ and $E_j = ((\tilde{B}_j \to \alpha_j) \cap (\tilde{B}'_j \to \alpha'_j))$. Define the move $T_1 \Rightarrow^{C_i} T_2$ as follows. Give each leaf v with label β_j two children v' and v'' with labels α_j and α'_j, respectively. Label the branches with B_j and B'_j, respectively. Let $k = \mathrm{depth}(T_1)$. One has $\Delta_v \vdash N : E_j$, hence by Lemma 19.5.8(i) one has $N \equiv \lambda y_k.N'$, with

$$\Delta_v, y_k\, \tilde{B}_j \vdash N'\, \alpha_j \;\&\; \Delta_v, y_k\, \tilde{B}'_j \vdash N' : \alpha'_j.$$

Therefore $\Delta_{v'} \vdash N' : a_{v'}$ and $\Delta_{v''} \vdash N' : a_{v''}$.

The case that M is an abstraction is impossible, by Lemma 19.5.4. ■

19.5.21. Corollary. *Let G be a tree game. For positions T one has*

$$T \text{ is favorable} \iff P_T^G \text{ has a solution.}$$

Proof. ($\Rightarrow$) This was Corollary 19.5.19. ($\Leftarrow$) Let M be a solution of P_T^G. By Theorem 19.1.15(ii) one may assume that M is in normal form. The conclusion follows from the previous lemma by induction on the size of M. ■

19.5.22. Theorem. *Let G be a tree game with initial position $\{a\}$. Then*

$$sol(G) \iff P_{\{a\}}^G \text{ has a solution.}$$

Proof. Immediate from the previous Corollary. ■

19.5.23. Corollary. *WTG $\leq_T$ IHP, i.e. winning a tree-game can be reduced to the inhabitation problem.*

Proof. By the theorem, as $P_{T_0}^G = P_a^G$ is an inhabitation problem. ■

Typewriters

In order to simplify our construction we introduce an auxiliary notion of a typewriter automaton. Informally, a typewriter automaton is just a reusable finite-state transducer. At each step, it reads a symbol, replaces it by a new one and changes the internal state. But at the end of the word, our automaton moves its reading and printing head back to the beginning of the tape and continues. This goes on until a final state is reached. That is, a typewriter automaton is a special case of a linear bounded automaton, see Kozen [1997]. A formal definition follows.

19.5.24. Definition. (i) A (deterministic) *typewriter automaton* $\mathcal{A}$ is a tuple of the form

$$\mathcal{A} = \langle\, \Sigma, Q, q_0, F, \varrho \,\rangle,$$

where Σ is a finite alphabet, Q is a finite set of states, $q_0 \in Q$ is an initial state and $F \subseteq Q$ is a set of final states. The last component is a transition function $\varrho : (Q - F) \times (\Sigma \cup \{\varepsilon\}) \to Q \times (\Sigma \cup \{\varepsilon\})$, which must satisfy the following condition: whenever $\varrho(q, a) = (p, b)$, then either $a, b \in \Sigma$ or $a = b = \varepsilon$.

(ii) A configuration (instantaneous description, ID) of $\mathcal{A}$ is represented by a triple $\langle\, w, q, v \,\rangle$, where (as usual) $wv \in \Sigma^*$ is the tape contents, $q \in Q$ is the current state, and the machine head points at the first symbol of v.

(iii) The next ID function $\overline{\varrho}$ is defined as follows:

- $\overline{\varrho}(\langle\, w, q, av \,\rangle) = \langle\, wb, p, v \,\rangle$, if $a \neq \varepsilon$ and $\varrho(q, a) = (p, b)$;
- $\overline{\varrho}(\langle\, w, q, \varepsilon \,\rangle) = \langle\, \varepsilon, p, w \,\rangle$, if $\varrho(q, \varepsilon) = (p, \varepsilon)$.

(iv) The language $L^{\mathcal{A}}$ accepted by $\mathcal{A}$ is the set of all $w \in \Sigma^*$, such that $\overline{\varrho}^k(\langle\, \varepsilon, q_0, w \,\rangle) = \langle\, u, q, v \,\rangle$, for some k and $q \in F, uv \in \Sigma^*$.

(v) ETW is the emptiness problem for typewriter automata.

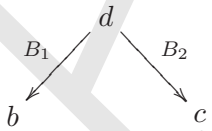
Recall that EQA, the emptiness problem for queue automata, is undecidable (see Kozen [1997]). We need the following.

19.5.25. Lemma. EQA $\leq_T$ ETW.

Proof. Exercise 19.6.22. ■

It follows that also ETW is undecidable.

Our goal is now to represent typewriters as games, in order to establish ETW$\leq$WTG. We begin with a refinement of Example 19.5.15. In what follows, triples of the form $\langle\, \langle\, B_1, b \,\rangle, \langle\, B_2, c \,\rangle, d \,\rangle$ will be represented graphically as



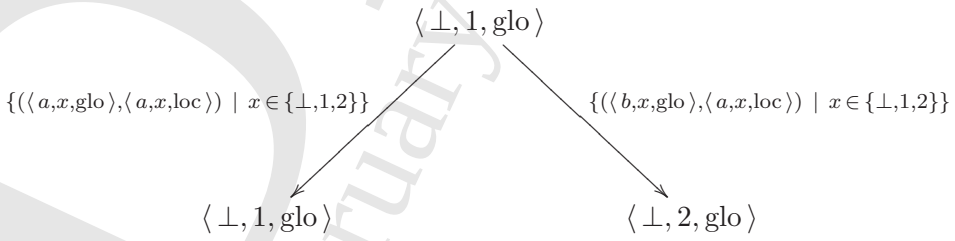
in order to enhance readability.

19.5.26. Definition. The alphabet Σ_1 is the following Cartesian product.

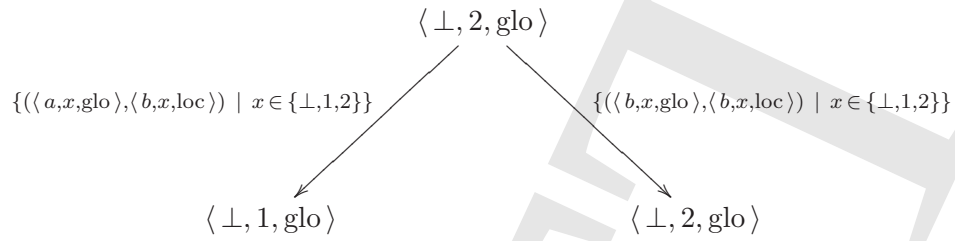
$$\Sigma_1 = \{\bot, a, b\} \times \{\bot, 1, 2\} \times \{\mathrm{loc}, \mathrm{glo}\}.$$

We define a tree game $G_1 = \langle\, \langle\, \bot, 1, C \,\rangle, \Sigma_1, \{C_1, C_2, C_3\} \,\rangle$. The set of accepting labels is Σ_1, because we are interested in all possible 'rounds' (i.e. instances) of the game. The moves are defined as follows.

(i) Global move C_1 consists of the following two triples:

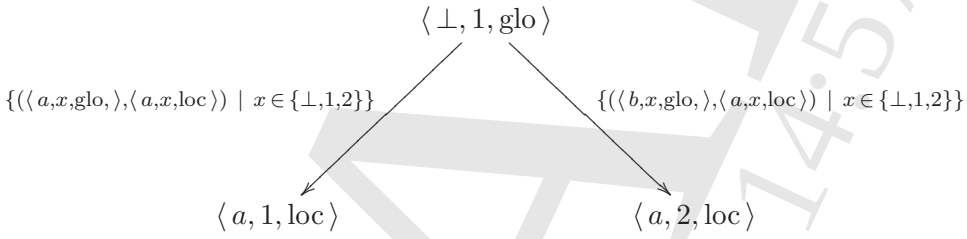


and

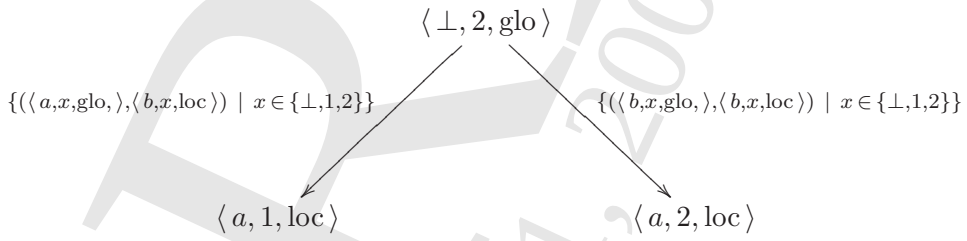


Before we define other global moves, let us point out that C_1 is very similar to rule C_1 of the game G_0 in Example 19.5.15 (observe the a and b in the first component and 1 and 2 in the second one).

(ii) Global move C_2 consists again of two triples:

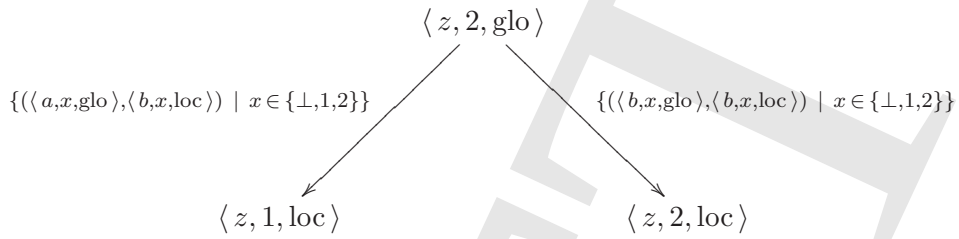


and



(iii) Global move C_3 consists of the triples (where $z \in \{a, b\}$, i.e., $z \neq \bot$)

and



19.5.27. LEMMA. *Every round of G_1 must have the following sequence of moves.*

$$\begin{array}{llllllllll} & C_1; & & C_1; & & C_1; & \dots & & C_1; & (m\times\, C_1) \\ C_2; & 1; & C_3; & 2; & C_3; & 3; & \dots & C_3; & m; & C_3; \\ m+1; & C_3; & m+3; & C_3; & m+5; & C_3; & \dots & & & \end{array}$$

That is, the game starts with m times a C_1 move (possibly $m = 0$ or $m = \infty$). After that, from the $m+1$-st position, the global and local moves alternate and the local move declared at every alternating step C_3 is executed exactly $2m+1$ steps later.

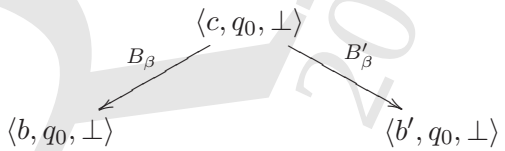
PROOF. Exercise 19.6.23. ■

19.5.28. DEFINITION. Let G_1 be as above and let $\mathcal{A} = \langle\, \Sigma^{\mathcal{A}}, Q, q_0, F^{\mathcal{A}}, \varrho \,\rangle$ be a typewriter automaton. We define a game $G^{\mathcal{A}}$ as follows.

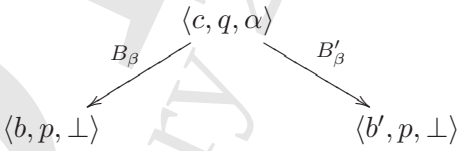
(i) The alphabet of $G^{\mathcal{A}}$ is the cartesian product $\Sigma_1 \times Q \times (\Sigma^{\mathcal{A}} \cup \{\bot, \varepsilon\})$.

(ii) For each local move B of G_1 and each $\beta \in \Sigma^{\mathcal{A}} \cup \{\varepsilon\}$, we define a local move $B_\beta = \{(\langle\, a, q, \beta \,\rangle, \langle\, b, q, \bot \,\rangle) \mid q \in Q \text{ and } \langle\, a, b \,\rangle \in B\}$.

(iii) If $\Delta = \langle\, \langle\, B, b \,\rangle, \langle\, B', b' \,\rangle, c \,\rangle \in C_1 \cup C_2$, then we define Δ_β as the triple



(iv) For each triple $\Delta = \langle\, \langle\, B, b \,\rangle, \langle\, B', b' \,\rangle, c \,\rangle \in C_3$ we define a set $C^{\mathcal{A}}_\Delta$ consisting of all triples of the form



where $\varrho(q, \alpha) = (p, \beta)$.

(v) Define
$$\begin{aligned} C^{\mathcal{A}}_1(\beta) &= \{\Delta_\beta \mid \Delta \in C_1\}, \qquad \text{for } \beta \in \Sigma^{\mathcal{A}}; \\ C^{\mathcal{A}}_2 &= \{\Delta_\varepsilon \mid \Delta \in C_2\}; \\ C^{\mathcal{A}}_3 &= \bigcup\{C^{\mathcal{A}}_\Delta \mid \Delta \in C_3\}. \end{aligned}$$

(vi) The initial symbol of $G^{\mathcal{A}}$ is $a = \langle\, a_1, q_0, \bot \,\rangle$, where $a_1 = \langle \bot, 1, G\rangle$, the initial symbol of G_1.

(vii) The set of final symbols is $A = \Sigma_1 \times (\Sigma^A \cup \{\bot, \varepsilon\}) \times F^{\mathcal{A}}$;

(viii) Finally, we take $G^{\mathcal{A}} = \langle\, a, A, \{C_1^{\mathcal{A}}(\beta) \mid \beta \in \Sigma^{\mathcal{A}}\} \cup \{C_2^{\mathcal{A}}, C_3^{\mathcal{A}}\} \,\rangle$

19.5.29. Proposition. *Let $\mathcal{A}$ be a typewriter that accepts the language $L_{\mathcal{A}}$. Then*

$$L_{\mathcal{A}} \neq \emptyset \iff G^{\mathcal{A}} \text{ is solvable.}$$

Hence ETW $\leq_T$ WTG.

Proof. Our game $G^{\mathcal{A}}$ behaves as a "cartesian product" of G_1 and $\mathcal{A}$. Informally speaking, there is no communication between the first component and the other two. In particular we have the following.

1. Lemma 19.5.27 remains true with G_1 replaced by $G^{\mathcal{A}}$ and C_1, C_2, C_3 replaced respectively by $C_1^{\mathcal{A}}(\beta)$, $C_2^{\mathcal{A}}$ and $C_3^{\mathcal{A}}$. That is, a legitimate round of $G^{\mathcal{A}}$ must look as follows.

$$\begin{array}{cccccccc} & & C_1^{\mathcal{A}}(\beta_1); & & C_1^{\mathcal{A}}(\beta_2); & \dots & & C_1^{\mathcal{A}}(\beta_m); \\ & C_2^{\mathcal{A}}; & 1; & C_3^{\mathcal{A}}; & 2; & \dots & C_3^{\mathcal{A}}; & m; \\ C_3^{\mathcal{A}}; & m+1; & C_3^{\mathcal{A}}; & m+3; & C_3^{\mathcal{A}}; & \dots & & \end{array}$$

2. If a position T of $G^{\mathcal{A}}$ can be reached from the initial position, then the second and third component of labels are always the same for all nodes at every fixed level of T.

Consider a round of the game $G^{\mathcal{A}}$ as in 1 above. Note that this sequence is fully determined by the choice of m and $\beta_1, \dots, \beta_m$. Also observe that β_i, for $i = 1, \dots, m$ are the third components of the labels of all leaves of T_{m+2i}. Let w denote the word $\beta_1\beta_2 \dots \beta_m$ and $O^{\mathcal{A}}(w)$ the 'opening' $C_1^{\mathcal{A}}(\beta_1), \dots, C_1^{\mathcal{A}}(\beta_m)$ in the game $G^{\mathcal{A}}$.

Claim. Typewriter $\mathcal{A}$ accepts w iff $O^{\mathcal{A}}(w)$ leads to a winning position in $G^{\mathcal{A}}$.

We shall now prove the claim. Let β_j^k denote the symbol contained in the j-th cell of the tape of $\mathcal{A}$, after the machine has completed $k-1$ full phases (the tape has been fully scanned $k-1$ times). That is, β_j^k is the symbol to be read during the k-th phase. Of course β_j^1 is β_j. For uniformity write $\beta_{m+1}^k = \varepsilon$. Note that the $q_0, \dots, q_m$ are renamed as $q_1^1, \dots, q_{m+1}^1$ and the $\beta_1, \dots, \beta_m, \epsilon$ as $\beta_1^1, \dots, \beta_{m+1}^1$. Further, let q_j^k be the internal state of the machine, just before it reads the j-th cell for the k-th time (i.e., after $k-1$ full phases). The reader will easily show that for all k and all $j = 1, \dots, m+1$ the following holds.

(i) The third component of labels of all leaves of $T_{(2k-1)(m+1)+2j-1}$ is β_j^k,

(ii) The second component of labels of all leaves of $T_{(2k-1)(m+1)+2j-2}$ is q_j^k.

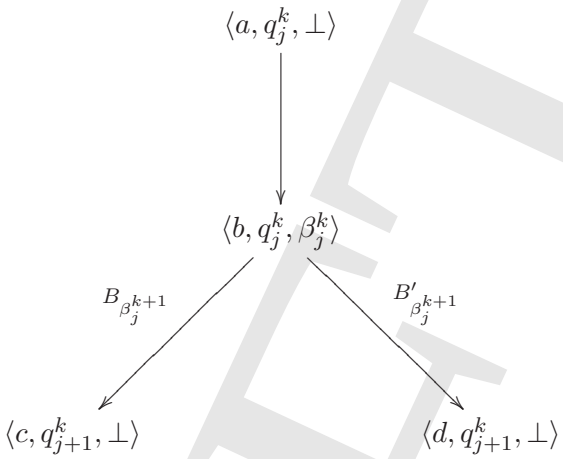


Figure 19.5: Simulation of a single machine step

Figure 19.5 illustrates the induction hypothesis by showing labels of a node at depth $(2k-1)(m+1)+2j-2$ together with her daughter and grandchildren. The claim follows when (ii) is applied to the final states.

It follows immediately from 1 and the claim that $L_{\mathcal{A}} \neq \emptyset$ iff there is a strategy to win $G^{\mathcal{A}}$. Hence the emptiness problem for typewriter automata can be reduced to the problem of winning tree games. ■

19.5.30. THEOREM. *The inhabitation problem for* $\lambda_{\cap}^{\mathrm{CDV}}$ *is undecidable.*

PROOF. By Corollary 19.5.23, Lemma 19.5.25 and Proposition 19.5.29. ■

Remarks

The proof of undecidability of the inhabitation problem presented in this section is a modified version of the original proof in Urzyczyn [1999].

The following notion of rank has been given for intersection types by Leivant [1983]. We denote it by *i-rank* in order to distinguish it from the notion defined in Definition 1.1.14.

$$\begin{aligned}
\text{i-rank}(A) &= 0, && \text{for simple types } A;\\
\text{i-rank}(A \cap B) &= \max(1, \text{i-rank}(A), \text{i-rank}(B));\\
\text{i-rank}(A \to B) &= \max(1 + \text{i-rank}(A), \text{i-rank}(B)), && \text{if } \cap \text{ occurs in } A \to B.
\end{aligned}$$

It should be observed that all types in game contexts are of i-rank at most 3. Thus, the relativized inhabitation problem is undecidable for contexts of i-rank 3, and the inhabitation problem (with empty contexts) is undecidable for types of i-rank 4. It is an open problem whether inhabitation is decidable for i-rank 3. But it is decidable for i-rank 2 (Exercise 19.6.24). Some other decidable cases are discussed in Kurata and Takahashi [1995].

From the point of view of the the *formulae as types* principle, (the "Curry-Howard isomorphism"), the inhabitation problem should correspond to a provability problem for a certain logic. It is however not at all obvious what should

be the logic corresponding to intersection types. We deal here with a "proof-functional" rather than "truth-functional" connective $\cap$, which is called sometimes "strong conjunction": a proof of $A \cap B$ must *be* a proof of both A and B, rather than merely *contain* two separate proofs of A and B. See Lopez-Escobar [n.d.], Mints [1989], and Alessi and Barbanera [1991] for the discussion of strong conjunction logics.

Various authors defined Church-style calculi for intersection types, which is another way leading to understand the logic of intersection types. We refer the reader to Venneri [1994], Dezani-Ciancaglini et al. [1997], Wells et al. [1997], Ronchi Della Rocca [2002], Liquori and Ronchi Della Rocca [2005], and Pimentel et al. [2005] for this approach.

19.6. Exercises

19.6.1. Show by means of examples that the type theories Plotkin and Engeler are not complete, i.e. we do not have $\Gamma \models^{\mathcal{T}} M : A \iff \Gamma \vdash^{\mathcal{T}}_{\cap} M : A$, for $\mathcal{T} \in \{\text{Plotkin}, \text{Engeler}\}$.

19.6.2. Show that $\Sigma(\{0\}, \mathrm{CDV})$ is the smallest complete type theory (w.r.t. the order of Figure 15.4).

19.6.3. Show that for all $\mathcal{T} \in \mathrm{PTT}$ one has

$$\Gamma \vdash^{\mathcal{T}}_{\cap} M : A \;\Rightarrow\; \Gamma \models^{\mathcal{T}}_{\cap} M : A.$$

[Hint. Adapting Definition 19.2.1 to proper intersection type theories in the obvious way.]

19.6.4. Let $\mathcal{T} \in \mathrm{TT}^{\text{-}\mathtt{U}}$. Take $\mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, \cdot, [\![\ ]\!]^{\mathcal{F}^{\mathcal{T}}} \rangle$ as in 18.2.1; define the type environment $\xi^{\mathcal{T}} : \mathbb{A}^{\mathcal{T}} \to \mathsf{P}(\mathcal{F}^{\mathcal{T}})$ by

$$\xi^{\mathcal{T}}(\alpha) = \{X \in \mathcal{F}^{\mathcal{T}} \mid \alpha \in X\};$$

and let $[\![\ ]\!]^{\mathcal{T}} : \Pi\!\!\Pi^{\mathcal{T}} \to \mathsf{P}(\mathcal{F}^{\mathcal{T}})$ be the map $[\![\ ]\!]^{\mathcal{F}^{\mathcal{T}}}_{\xi^{\mathcal{T}}}$ defined by deleting the first clause from Definition 19.2.1. Show the following.

(i) $[\![A]\!]^{\mathcal{T}} = \{X \in \mathcal{F}^{\mathcal{T}} \mid A \in X\}$.

(ii) $\mathcal{F}^{\mathcal{T}}$, $\xi^{\mathcal{T}}$ are $\to$-good and preserve $\leq_{\mathcal{T}}$.

(iii) $[\Gamma \models^{\mathcal{T}}_{\cap} M : A \;\Rightarrow\; \Gamma \vdash^{\mathcal{T}}_{\cap} M : A] \iff \mathcal{T} \in \mathrm{PTT}$.

(iv) $\Gamma \models^{\mathcal{T}}_{\cap} M : A \iff \Gamma \vdash^{\mathcal{T}}_{\cap} M : A$.

19.6.5. Show that all quasi λ-models and all type environments preserve $\leq_{\mathrm{BCD}}$.

19.6.6. Dezani-Ciancaglini et al. [2005b], Tatsuta and Dezani-Ciancaglini [2006].

1. The terms that are in PHN reduce to terms of the form $\lambda\vec{x}.y\vec{M}$ where $y \notin \{\vec{x}\}$. Are these all of them? Is this enough to characterize them?
2. The terms that are in PN reduce to terms of the form $\lambda\vec{x}.y\vec{M}$ where $y \notin \{\vec{x}\}$ and $\vec{M} \in \mathsf{N}$. Are these all of them? Is this enough to characterize them?

3. The terms that are in PSN strongly reduce to terms of the form $\lambda\vec{x}.y\vec{M}$ where $y \notin \{\vec{x}\}$ and $\vec{M} \in \mathsf{N}$. Are these all of them? Is this enough to characterize them?
4. Conclude that $\mathsf{PSN} \subset \mathsf{PN} \subset \mathsf{PHN}$.

19.6.7. Show that PHN is HN-stable and PN is N-stable.

19.6.8. Let $\Lambda(\beta) = \langle \Lambda, \cdot, [\![\ ]\!]^{\Lambda} \rangle$ be the term model of β-equality and $[M]$ the equivalence class of M under β-equality. Let a term M be persistently head normalizing iff $M\vec{N}$ has a head normal form for all terms $\vec{N}$ (see Definition 19.1.3). Define the type environment

$$\xi_{\mathrm{Scott}}(0) = \{[M] \mid M \text{ is persistently head normalizing}\}.$$

Prove that $(\Lambda(\beta), \xi_{\mathrm{Scott}})$ preserves $\leq_{\mathrm{Scott}}$.

19.6.9. Let $\Lambda(\beta)$ and $[M]$ be as in Exercise 19.6.8. Define the type environment

$$\xi_{\mathrm{Park}}(0) = \{[M] \mid M \text{ reduces to a closed term}\}.$$

Prove that $(\Lambda(\beta), \xi_{\mathrm{Park}})$ preserves $\leq_{\mathrm{Park}}$.

19.6.10. A term $(\lambda x.M)N$ is a $\beta\mathsf{N}$-*redex* if $x \notin FV(M)$ or [N is either a variable or a closed SN (strongly normalizing) term] (Honsell and Lenisa [1999]). We denote by $\twoheadrightarrow_{\beta\mathsf{N}}$ the induced reduction. Show that if Γ assigns types to all free variables in N, i.e. $x \in \mathrm{dom}(\Gamma)$ for all $x \in \mathrm{FV}(N)$, then

$$\Gamma \vdash^{\mathrm{HL}}_{\cap} M : A \ \&\ N \twoheadrightarrow_{\beta\mathsf{N}} M \ \Rightarrow\ \Gamma \vdash^{\mathrm{HL}}_{\cap} N : A.$$

[Hint: use Theorem 19.1.15(iii).]

19.6.11. Show that in the system induced by the type theory AO, the terms typable with type $\mathtt{U} \to \mathtt{U}$ for a suitable context are precisely the lazy normalizing ones, i.e. the terms which reduce either to an abstraction or to a (λ-free) head normal form.

19.6.12. Show that in the system induced by the type theory EHR, as defined in definition 15.2.13, the terms typable with type $\mathtt{V}$ in the context all whose predicates are $\mathtt{V}$ are precisely the terms which reduce either to an abstraction or to a variable using the call-by-value β-reduction rule.

19.6.13. Show that all type interpretation domains and all type environments agree with AO.

19.6.14. Using the Approximation Theorem, show that

- there is no type deducible for $\boldsymbol{\Omega}$ in the system $\vdash^{\mathrm{CDV}}_{\cap}$;
- $\vdash^{\mathrm{BCD}}_{\cap} \boldsymbol{\Omega} : A$ iff $A =_{\mathrm{BCD}} \mathtt{U}$;
- $\vdash^{\mathrm{Park}}_{\cap} \boldsymbol{\Omega} : A$ iff $0 \leq_{\mathrm{Park}} A$.

19.6.15. Using the Approximation Theorem, show that in the system $\lambda^{\mathrm{AO}}_{\cap}$ the set of types deducible for $\boldsymbol{\omega\omega}$ is stricty included in the set of types deducible for $\mathsf{K}(\boldsymbol{\Omega})$.

19.6.16. Using the Approximation Theorem, show that in the system $\lambda_{\cap}^{\mathrm{Scott}}$ the set of types deducible for J and I coincide.

19.6.17. Prove that the set $\{M \mid \exists \Gamma, A \,.\, \Gamma \vdash_{\cap}^{\mathcal{T}} M : A\}$ is computable. [Hint. This is obvious.]

19.6.18. Prove Lemma 19.5.2(i) and (ii). [Hint. Similar to the proof of Lemma 15.1.22.]

19.6.19. Consider the type assignment systems $\lambda_{\cap}^{\mathrm{Krivine}}$ and $\lambda_{\cap}^{\mathrm{Krivine^U}}$ as defined in Exercise **??**.

(i) Prove an analogue of Lemma 19.5.8.

(ii) Prove that if Γ is a game context then $\Gamma \vdash^{\mathrm{Krivine}} M : \alpha$ and $\Gamma \vdash^{\mathrm{Krivine^U}} M : \alpha$ are equivalent to $\Gamma \vdash M : \alpha$, for all type variables α. Conclude that type inhabitation remains undecidable without $(\leq)$.

(iii) Prove that the type $\delta \cap (\alpha \to \beta) \cap (\alpha \to \gamma) \to \delta \cap (\alpha \to \beta \cap \gamma)$ is inhabited in $\lambda_{\cap}^{\mathrm{BCD}}$ but is not inhabited in $\lambda_{\cap}^{\mathrm{Krivine^U}}$.

19.6.20. Let Γ be a game context and $\alpha \in \mathbb{A}_\infty$. Prove that if $\Gamma \vdash M : \alpha$ then every node in the Böhm tree of M (see Barendregt [1984], Ch. 10) has at most one branch.

19.6.21. Complete the proofs of Proposition 19.5.22 and Lemma 19.5.23.

19.6.22. Prove Lemma 19.5.25. [Hint. Encode a queue automaton (also called a Post machine, i.e. a deterministic finite automaton with a queue) into a typewriter, thus reducing the halting problem for queue automata to the emptiness problem for typewriters. One possible way of doing it is as follows. Represent a queue, say "011100010", as a string of the form

$$\$\$\cdots\$<011100010>\sharp\cdots\sharp\sharp,$$

with a certain number of the \$'s and $\sharp$'s. The initial empty queue is just "$<>\sharp\cdots\sharp\sharp$". Now an *insert* instruction means: *replace ">" with a digit and replace the first "$\sharp$" with ">"*, and similarly for a *remove*. The number of \$'s increases after each *remove*, while the suffix of $\sharp$'s shrinks after each *insert*, so that the queue "moves to the right". If the number of the initial suffix of $\sharp$'s is sufficiently large, then a typewriter automaton can verify the queue computation.]

19.6.23. Prove Lemma 19.5.27. [Hint. Compare G_1 to the game G_0 of Example 19.5.15. Observe that in each sequence of positions

$$T_{(2k+1)n}, \ldots, T_{(2k+3)n},$$

the odd steps behave as an initial phase of G_0, while the even steps behave as a final phase of G_0. Writing

$$\begin{aligned} \bot_i &= \langle \bot, i, G\rangle; \\ A &= \langle a, 1, \{G, L\}\rangle\langle a, 2, \{G, L\}\rangle; \\ B &= \langle b, 1, \{G, L\}\rangle\langle b, 2, \{G, L\}\rangle \end{aligned}$$

we have the following (the canope of a tree is the collection of its leaves) for the case of two initial C_1 steps.

position #	via move	canope of position
0		$\bot_1$
1	C_1	$\bot_1\bot_2$
2	C_1	$(\bot_1\bot_2)^2$
3	C_2	A^2A^2
4	1	A^2B^2
5	C_3	A^4B^4
6	2	$(A^2B^2)^2$
7	C_3	$(A^4B^4)^2$
8	3	$(A^2B^2)^4$
9	C_3	$(A^4B^4)^4$
10	5	$(A^2B^2)^8$
11	C_3	$(A^4B^4)^8$
12	7	$(A^2B^2)^{16}$
...	...	...
$4+2k$		$(A^2B^2)^{2^k}$

If one starts with m moves of C_1 ($0<m<\infty$), then the canope of position $m+2+2k$ will be $(A^{2^{m-1}}B^{2^{m-1}})^{2^k}$. Note that $m=0$ or $m=\infty$ yield possible plays of the game.]

19.6.24. Kuśmierek [2007]. Prove that in $\lambda^{\mathrm{CDV}}_{\cap}$ the inhabitation problem for types of i-rank at most 2 is decidable. See Definition of i-rank after Theorem 19.5.30. Note that if the i-rank of $B{\to}C$ is at most 2 then the i-rank of B is at most 1.
[Hint. Start by dividing the task of finding an inhabitant of $A_1\cap\ldots\cap A_n$ in n tasks where each A_i is either a type atom or an arrow. Construct a non-deterministic procedure to find n inhabitants of

$$\Gamma_1\vdash ?:A_1\ \ \ldots\ \ \Gamma_n\vdash ?:A_n$$

where A_i is a type atom or an arrow and the i-rank of the types of Γ_i is at most 1 for all $i=\{1,\ldots,.n\}$.]

19.6.25. Kuśmierek [2007].

(i) Let $\iota=(\alpha\to\alpha)\cap(\beta\to\beta)$ and define for $k=0,\ldots,n$ the type

$$A_k=\alpha\to\iota^k\to(\alpha\to\beta)\to(\beta\to\alpha)^{n-k}\to\beta.$$

Prove that the shortest inhabitant of $A_1,\ldots,A_n$ is of length exponential in n. Can you modify the example so that the shortest inhabitant is of double exponential length?

(ii)* How long (in the worst case) is the shortest inhabitant (if it exists) of a given type whose i-rank is 2?

Chapter 20

Further Readings 21.2.2008:897

Comment: the actual further reading chapter is a waste of time and paper! Henk: One should include only papers with lasting value, as far as we can see.

Intersection types have been originally introduced in a number of papers, Coppo and Dezani-Ciancaglini [1980], Coppo et al. [1980] and Barendregt et al. [1983], as a language for describing and capturing properties of λ-terms, which had escaped all previous typing disciplines. For instance, they were used in Pottinger [1980] in order to give the first type theoretic characterization of *strongly normalizable* terms, and later in Coppo et al. [1987] in order to capture *persistently normalizing terms* and *normalizing terms*.

Since then, intersection types have been used as a powerful tool both for the analysis and the synthesis of λ-models, see *e.g.* Barendregt et al. [1983], Coppo et al. [1984], Coppo et al. [1987], Alessi [1991], Egidi et al. [1992], Honsell and Ronchi Della Rocca [1992], Di Gianantonio and Honsell [1993], Plotkin [1993], Honsell and Lenisa [1999], Pravato et al. [1999]. On the one hand, intersection type disciplines provide finitary inductive definitions of interpretation of λ-terms in models. On the other hand, they are suggestive for the shape the domain model has to have in order to exhibit certain properties, Coppo et al. [1987], Honsell and Ronchi Della Rocca [1992].

Citare:
-lavori di Kfoury, Wells, che usano unione,intersezione per i compilatori
- libro di Hankin
- lavoro di Hindley (intersection types: an introduction)

Intersection type assignment systems have been developed in a number of papers which have been cited in the previous sections. An extensive study of the syntactical properties of the several systems for intersection types is **?** (see also **?**, **?**). In particular, the papers **?** define principal type schemes for various intersection type disciplines. An exposition of the proof-theoretic properties of intersection type systems used as tools for the syntactical study of type-free λ-calculus can be found in **?**, which makes extensive use of the notion of saturated sets, called type stable sets in Section 19.1.

The paper **?** gave rise to a number of studies relating intersection type theories to domain theory and λ-models (see **?**). Many of these results are given in Section 19.

In his PhD thesis **?** Alessi gives an account of filter models as representations of domains. Moreover he studies certain pathologies of domains with respect to their representations, by defining a more basic category of models, which admit a good description in terms of intersection types. See also **?**. In the same thesis, and in **?**, other categories of domains, like Girard's qualitative and quantitative domains, are studied using the intersection types.

All the above-mentioned models are models of the classical λ-calculus, but intersection-type disciplines are also suitable for describing models of the $\lambda\mathbf{I}$-calculus **?**, of the lazy λ-calculus **?**, of the call-by-value λ-calculus (introduced by Plotkin in **?**) **?**, and of some extensions of the λ-calculus which include parallel features **?**.

The finitary descriptions of domains have been a major point of interest. Scott pioneered this topic with his Information Systems in **?**. Abramsky extended the theory to SFP domains in **?**, framing it into the paradigm of Stone duality. A strikingly short, but good introduction to this subject is Chapter 10 of **?**. We also suggest the reading of **?**: it is not exactly about filter models, but it gives a perspective to the webbed models (originated with Engeler models **?**), which are an alternative, though related, finitary description of λ-models. See also **?**.

Since, given an arbitrary λ-term, it is undecidable whether it can be typed using intersection types, it is interesting to look for decidable restrictions. For this topic see **?**.

Extensions of intersection types with union and quantified types, both universal and existential, have been studied in **?**. The completeness problem for such systems has been recently settled by Yokouchi in **?**.

Systems of intersection types à la Church are not trivial to define: they have been extensively studied in **?** and applied to type disciplines for object oriented languages in **?**.

Lastly we mention that intersection and union types have been successfully used to prove properties of programs, like strictness analysis, detection of dead code, etc. Pioneers in this field were the PhD thesis of Benton **?** and Jensen **?** and the paper **?**. The recent PhD thesis by Mossin **?** and Damiani **?** contain the last developments and the references to the large literature on the subject.

Part IV

Indices and references

21.2.2008:897

Chapter 21

Index

Chapter 22

References

21.2.2008:897

Bibliography

Abramsky, S., Dov M. Gabbay and T.S.E. Maibaum (eds.) [1992]. *Handbook of Logic in Computer Science, Volume 2: Background: Computational Structures*, Oxford Science Publications.

Abramsky, Samson [1991]. Domain theory in logical form, *Ann. Pure Appl. Logic* **51**(1-2), pp. 1–77. Second Annual IEEE Symposium on Logic in Computer Science (Ithaca, NY, 1987).

Abramsky, Samson and C.-H. Luke Ong [1993]. Full abstraction in the lazy lambda calculus, *Inform. and Comput.* **105**(2), pp. 159–267.

Ackermann, W. [1928]. Zum Hilbertschen Aufbau der reellen Zahlen, *Mathematische Annalen* **99**, pp. 118–133.

Alessi, F. [1991]. *Strutture di tipi, teoria dei domini e modelli del lambda calcolo*, Dissertation, Torino University.

Alessi, Fabio [1993]. The p model, Internal Report, Udine University.

Alessi, Fabio and Franco Barbanera [1991]. Strong conjunction and intersection types, *in:* A. Tarlecki (ed.), *Proc. MFCS 1991*, LNCS 520, pp. 64–73.

Alessi, Fabio, Franco Barbanera and Mariangiola Dezani-Ciancaglini [2003]. Intersection types and computational rules, *in:* Ruy de Queiroz, Elaine Pimentel and Lucilia Figueiredo (eds.), *WoLLIC'03*, ENTCS 84, Elsevier.

Alessi, Fabio, Franco Barbanera and Mariangiola Dezani-Ciancaglini [2004]. Tailoring filter models, *in:* Stefano Berardi, Mario Coppo and Ferruccio Damiani (eds.), *Types '03*, Lecture Notes in Computer Science 3085, Springer-Verlag, pp. 17–33.

Alessi, Fabio, Franco Barbanera and Mariangiola Dezani-Ciancaglini [2005]. Intersection types and lambda models, *Theoret. Comput. Sci.* to appear.

Alessi, Fabio, Franco Barbanera and Mariangiola Dezani-Ciancaglini [2006]. Intersection Types and Lambda Models, *Theoretical Compututer Science*. to appear.

Alessi, Fabio, Mariangiola Dezani-Ciancaglini and Furio Honsell [2004]. Inverse limit models as filter models, *in:* Delia Kesner, Femke van Raamsdonk and J. B. Wells (eds.), *HOR '04*, RWTH Aachen, Aachen, pp. 3–25. ISSN 09353232, AIB-2004-03.

Alessi, Fabio, Mariangiola Dezani and Furio Honsell [2001]. Filter models and easy terms, *in:* Simona Ronchi and Antonio Restivo (eds.), *In ICTCS'01*, LNCS 2202, Springer, pp. 17–37.

Amadio, Roberto M. and Pierre-Louis Curien [1998]. *Domains and lambda-calculi*, Cambridge University Press, Cambridge.

Aspinall, D. and A. Compagnoni [1996]. Subtyping dependent types, *in:* E. Clarke (ed.), *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, IEEE Computer Society Press, New Brunswick, New Jersey, pp. 86–97.

Baeten, J. and B. Boerboom [1979]. ω can be anything it shouldn't be, *Indag.Math.* **41**, pp. 111–120.

van Bakel, S., F. Barbanera, M. Dezani-Ciancaglini and F.-J. de Vries [2002]. Intersection types for lambda-trees, *Theoret. Comput. Sci.* **272**(1–2), pp. 3–40.

van Bakel, Steffen [1992]. Complete restrictions of the intersection type discipline, *Theoret. Comput. Sci.* **102**(1), pp. 135–163.

van Bakel, Steffen [1993]. Principal type schemes for the strict type assignment system, *J. Logic Comput.* **3**(6), pp. 643–670.

Barbanera, F., M. Dezani-Ciancaglini and U. de' Liguoro [1995]. Intersection and union types: syntax and semantics, *Information and Computation* **119**, pp. 202–230.

Barbanera, Franco, Mariangiola Dezani-Ciancaglini and Ugo de'Liguoro [1995]. Intersection and union types: syntax and semantics, *Inform. and Comput.* **119**(2), pp. 202–230.

Barendregt, H. P. [1974]. Pairing without conventional restraints, *Z. Math. Logik Grundlagen Math.* **20**, pp. 289–306.

Barendregt, H. P. [1984]. *The Lambda Calculus: its Syntax and Semantics*, revised edition, North-Holland, Amsterdam.

Barendregt, H. P. and E. Barendsen [1997]. Efficient computations in formal proofs. to appear.

Barendregt, H. P. and S. Ghilezan [n.d.]. Lambda terms for natural deduction, sequent calculus and cut-elimination, *Journal of Functional Programming*. To appear.

Barendregt, Henk, Mario Coppo and Mariangiola Dezani-Ciancaglini [1983]. A filter lambda model and the completeness of type assignment, *J. Symbolic Logic* **48**(4), pp. 931–940 (1984).

Barendregt, H.P. [1992]. Lambda calculi with types, Abramsky et al. [1992], Oxford University Press, pp. 117–309.

Barendregt, H.P. [1996]. The quest for correctness, *Images of SMC Research*, Stichting Mathematisch Centrum, P.O. Box 94079, 1090 GB Amsterdam, pp. 39–58.

Barendregt, H.P., M. Bunder and W. Dekkers [1993]. Systems of illative combinatory logic complete for first order propositional and predicate calculus, *The Journal of Symbolic Logic* **58**(3), pp. 89–108.

Berarducci, A. and C. Bohm [1993]. A self-interpreter of lambda calculus having a normal form, *Lecture Notes in Computer Science* **702**, pp. 85–99.

Berline, Chantal [2000]. From computation to foundations via functions and application : the lambda-calculus and its webbed models, *Theoret. Comput. Sci.* **249**, pp. 81–161.

Bezem, M. and J.F. Groote (eds.) [1993]. *Typed Lambda Calculi and Applications, TLCA'93*, number 664 in *Lecture Notes in Computer Science*, Springer Verlag, Berlin.

Bezem, M.A. [1985]. Strong normalization of bar recursive terms without using infinite terms, *Archiv für Mathematische Logik und Grundlagenforschung* **25**, pp. 175–181.

Böhm, C. [1963]. The cuch as a formal and description language, *in:* Richard Goodman (ed.), *Annual Review in Automatic Programming, Vol. 3*, Pergamon Press, Oxford, pp. 179–197.

Böhm, C. and A. Berarducci [1985]. Automatic synthesis of typed Λ-programs on term algebras, *Theoretical Computer Science* **39**, pp. 135–154.

Böhm, C. and W. Gross [1966]. Introduction to the CUCH, *in:* E.R. Caianiello (ed.), *Automata Theory*, Academic Press, New York, pp. 35–65.

Böhm, C., A. Piperno and S. Guerrini [1994]. lambda-definition of function(al)s by normal forms, *in:* D. Sanella (ed.), *ESOP'94*, Vol. 788, Springer, Berlin, pp. 135–154.

Böhm, Corrado and Mariangiola Dezani-Ciancaglini [1975]. λ-terms as total or partial functions on normal forms, pp. 96–121. Lecture Notes in Comput. Sci., Vol. 37.

de Bruijn, N. G. [1970]. The mathematical language AUTOMATH, its usage and some of its extensions, *in:* M. Laudet, D. Lacombe and M. Schuetzenberger (eds.), *Symposium on Automatic Demonstration*, Springer Verlag, Berlin, 1970, IRIA, Versailles, pp. 29–61. Lecture Notes in Mathematics **125**; also in Nederpelt et al. [1994].

de Bruijn, N. G. [1972]. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem, *Nederl. Akad. Wetensch. Proc. Ser. A* **75**=*Indag. Math.* **34**, pp. 381–392.

de Bruijn, N.G. [1990]. Reflections on Automath, Eindhoven University of Technology. Also in Nederpelt et al. [1994], pp 201–228.

Capretta, V. and S. Valentini [1998]. A general method for proving the normalization theorem for first and second order typed λ-calculi, *Mathematical Structures in Computer Science*. To appear.

Church, A. [1940]. A formulation of the simple theory of types, *The Journal of Symbolic Logic* **5**, pp. 56–68.

Church, Alonzo [1936]. An unsolvable problem of elementary number theory, *American Journal of Mathematics* **58**, pp. 354–363.

Comon, Hubert and Yan Jurski [1998]. Higher-order matching and tree automata, *Computer science logic (Aarhus, 1997)*, Lecture Notes in Comput. Sci. 1414, Springer, Berlin, pp. 157–176.

Coppo, M., M. Dezani-Ciancaglini, F. Honsell and G. Longo [1983]. Applicative information systems, *CAAP '83 (L'Aquila, 1983)*, Springer, Berlin, pp. 35–64.

Coppo, M., M. Dezani-Ciancaglini, F. Honsell and G. Longo [1984]. Extended type structures and filter lambda models, *Logic colloquium '82 (Florence, 1982)*, North-Holland, Amsterdam, pp. 241–262.

Coppo, Mario and Mariangiola Dezani-Ciancaglini [1980]. An extension of the basic functionality theory for the λ-calculus, *Notre Dame J. Formal Logic* **21**(4), pp. 685–693.

Coppo, Mario, Mariangiola Dezani-Ciancaglini and Patrick Sallé [1979]. Functional characterization of some semantic equalities inside lambda-calculus., *in:* Hermann A. Maurer (ed.), *ICALP'79*, pp. 133–146.

Coppo, Mario, Mariangiola Dezani-Ciancaglini and B. Venneri [1980]. Principal type schemes and λ-calculus semantics, *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, Academic Press, London, pp. 535–560.

Coppo, Mario, Mariangiola Dezani-Ciancaglini and B. Venneri [1981]. Functional characters of solvable terms, *Z. Math. Logik Grundlag. Math.* **27**(1), pp. 45–58.

Coppo, Mario, Mariangiola Dezani-Ciancaglini and Maddalena Zacchi [1987]. Type theories, normal forms, and D_∞-lambda-models, *Inform. and Comput.* **72**(2), pp. 85–116.

Coquand, Thierry and Gérard Huet [1988]. The Calculus of Constructions, *Information and Computation* **76**(2/3), pp. 95–120.

Curry, H.B. [1934]. Functionality in combinatory logic, *Proceedings of the National Academy of Science of the USA* **20**, pp. 584–590.

Curry, H.B. and R. Feys [1958]. *Combinatory Logic*, Studies in Logic and the Foundations of Mathematics I, North-Holland, Amsterdam.

Davis, M. [1973]. Hilbert's tenth problem is unsolvable, *American Mathematical Monthly* **80**, pp. 233–269.

Davis, M., J. Robinson and H. Putnam [1960]. The decision problem for exponential Diophantine equations, *Annals of Mathematics*.

Dekkers, W., M. Bunder and H.P. Barendregt [1997]. Completeness of the propositions-as-types interpretation of intuitionistic logic into illative combinatory logic, *The Journal of Symbolic Logic*.

Dekkers, Wil [1988]. Reducibility of types in typed lambda calculus. Comment on: "On the existence of closed terms in the typed λ-calculus, I" (Statman [1980a]), *Inform. and Comput.* **77**(2), pp. 131–137.

Dezani-Ciancaglini, M., S. Ghilezan and B. Venneri [1997]. The "relevance" of intersection and union types, *Notre Dame Journal of Formal Logic* **38**(2), pp. 246–269.

Dezani-Ciancaglini, Maraingiola, Furio Honsell and Fabio Alessi [2003]. A complete characterization of complete intersection-type preorders, *ACM TOCL* **4**(1), pp. 120–147.

Dezani-Ciancaglini, Mariangiola and Ines Margaria [1986]. A characterization of F-complete type assignments, *Theoret. Comput. Sci.* **45**(2), pp. 121–157.

Dezani-Ciancaglini, Mariangiola, Furio Honsell and Yoko Motohama [2001]. Approximation theorems for intersection type systems, *Journal of Logic and Computation* **11**(3), pp. 395–417.

Dezani-Ciancaglini, Mariangiola, Furio Honsell and Yoko Motohama [2005a]. Compositional characterization of λ-terms using intersection types, *Theoret. Comput. Sci.* **340**(3), pp. 459–495. An extended abstract appeared in *MFCS'00*, volume 1893 of *LNCS*, pages 304–313. Springer-Verlag, 2000.

Dezani-Ciancaglini, Mariangiola, Furio Honsell and Yoko Motohama [2005b]. Compositional Characterization of λ-terms using Intersection Types, *Theoretical Computer Science* **340**(3), pp. 459–495.

Di Gianantonio, Pietro and Furio Honsell [1993]. An abstract notion of application, *Typed lambda calculi and applications (Utrecht, 1993)*, Springer, Berlin, pp. 124–138.

Dowek, Gilles [1994]. Third order matching is decidable, *Ann. Pure Appl. Logic* **69**(2-3), pp. 135–155. Invited papers presented at the 1992 IEEE Symposium on Logic in Computer Science (Santa Cruz, CA).

van Draanen, J.-P. [1995]. *Models for simply typed lambda-calculi with fixed point combinators and enumerators*, Dissertation, Catholic University of Nijmegen.

Dyckhoff, R. and L. Pinto [1997]. Permutability of proofs in intuitionistic sequent calculi, *Theoretical Computer Science*. To appear.

Egidi, Lavinia, Furio Honsell and Simona Ronchi Della Rocca [1992]. Operational, denotational and logical descriptions: a case study, *Fund. Inform.* **16**(2), pp. 149–169. Mathematical foundations of computer science '91 (Kazimierz Dolny, 1991).

Elbers, H. [1996]. Personal communication.

Engeler, Erwin [1981]. Algebras and combinators, *Algebra Universalis* **13**(3), pp. 389–392.

Euclid [n.d.]. The Elements, 325 B.C. English translation in Heath [1956].

Flajolet, P. and R. Sedgewick [1993]. The average case analysis of algorithms: counting and generating functions, *Technical Report 1888*, INRIA.

Friedman, H.M. [1975]. Equality between functionals, *in:* R. Parikh (ed.), *Logic Colloqium*, Lecture Notes in Mathematics 453, Springer, Berlin, New York.

Gandy, R. [1980a]. An early proof of normalization by A. M. Turing, *in:* J. P. Seldin and J. R. Hindley (eds.), *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press, New York, pp. 453–457.

Gandy, R.O. [1980b]. Church's Thesis and principles for mechanisms, *The Kleene Symposium*, North-Holland Publishing Company, Amsterdam, pp. 123–148.

Gentzen, G. [1936]. Untersuchungen über das logischen Schliessen, *Mathematische Zeitschrift* **39**, pp. 405–431. Translation in: *Collected papers of Gerhard Gentzen*, ed. M. E. Szabo, North-Holland, Amsterdam [1969], pp. 68–131.

Gentzen, Gerhard [1969]. Investigations into logical deduction, *in:* M. E. Szabo (ed.), *The Collected Papers of Gerhard Gentzen*, North-Holland.

Ghilezan, Silvia [1996]. Strong normalization and typability with intersection types, *Notre Dame J. Formal Logic* **37**(1), pp. 44–52.

Gierz, G.K., Karl Heinrich Hofmann, Klaus Keimel, Lawson Jimmie D., M.W. Mislove and D.S. Scott [1980]. *A Compendium of Continuous Lattices*, Springer, Berlin and New York.

Girard, J.-Y. [1972]. Interpretation fonctionelle et elimination des coupures de l'arithmetique d'ordre superieur, These D'Etat, Universite Paris VII.

Girard, Jean-Yves [1971]. Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types, *Proceedings of the Second Scandinavian Logic Symposium (Univ. Oslo, Oslo, 1970)*, North-Holland, Amsterdam, pp. 63–92. Studies in Logic and the Foundations of Mathematics, Vol. 63.

Girard, Jean-Yves, Yves G. A. Lafont and Paul Taylor [1989]. *Proofs and Types*, Cambridge Tracts in Theoretical Computer Science 7, Cambridge University Press.

Gödel, K. [1931]. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I, *Monatshefte für Mathematik und Physik* **38**, pp. 173–198. German; English translation in van Heijenoort [1967], pages 592–618.

Gödel, K. [1958]. Ueber eine bisher noch nicht benützte Erweiterung des finiten Standpunktes, *Dialectica* **12**, pp. 280–287.

Goldfarb, Warren D. [1981]. The undecidability of the second-order unification problem, *Theoret. Comput. Sci.* **13**(2), pp. 225–230.

Grabmayer, Clemens [2005]. *Relating Proof Systems for Recursive Types*, Dissertation, Vrije Universiteit Amsterdam.

Grabmayer, Clemens [2007]. A duality between proof systems for cyclic term graphs, *Mathematical Structures in Computer Science* **17**, pp. 439–484.

Harrington, L. A., M. D. Morley, A. Ščedrov and S. G. Simpson (eds.) [1985]. *Harvey Friedman's research on the foundations of mathematics*, Studies in Logic and the Foundations of Mathematics 117, North-Holland Publishing Co., Amsterdam.

Heath, T.L. [1956]. *The Thirteen Books of Euclid's Elements*, Dover Publications, Inc., New York.

van Heijenoort, J. (ed.) [1967]. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879 –1931*, Harvard University Press, Cambridge, Massachusetts.

Herbelin, H. [1995]. A lambda calculus structure isomorphic to Gentzen-style sequent calculus structure, *Computer Science Logic (CSL'94)*, Lecture Notes in Computer Science **933**, Springer Verlag, Berlin, pp. 61–75.

Hindley, J.R. [1997]. *Basic Simple Type Theory*, Cambridge University Press, Cambridge, UK.

Hindley, J.Roger [1983]. The completeness theorem for typing λ-terms, *Theoret. Comput. Sci.* **22**, pp. 1–17.

Hindley, R. and G. Longo [1980]. Lambda-calculus models and extensionality, *Z. Math. Logik Grundlag. Math.* **26**(4), pp. 289–310.

Hofmann, M. [1977]. A simple model for quotient types, *Typed lambda calculi and applications*, Lecture Notes in Computer Science, Springer, Berlin and New York, pp. 216–234.

Honsell, Furio and Marina Lenisa [1993]. Some results on the full abstraction problem for restricted lambda calculi, *Mathematical foundations of computer science 1993 (Gdańsk, 1993)*, Springer, Berlin, pp. 84–104.

Honsell, Furio and Marina Lenisa [1999]. Semantical analysis of perpetual strategies in λ-calculus, *Theoret. Comput. Sci.* **212**(1-2), pp. 183–209. Gentzen (Rome, 1996).

Honsell, Furio and Simona Ronchi Della Rocca [1992]. An approximation theorem for topological lambda models and the topological incompleteness of lambda calculus, *J. Comput. System Sci.* **45**(1), pp. 49–75.

Howard, W. A. [1970]. Assignment of ordinals to terms for primitive recursive functionals of finite type, *in:* J. Myhill A. Kino and R.E. Vesley (eds.), *Intuitionism and Proof Theory, Proceedings of the summer conference at Buffalo N.Y. 1968*, Studies in logic and the foundations of mathematics, North-Holland, Amsterdam, pp. 443–458.

Hyland, Martin [1975/76]. A syntactic characterization of the equality in some models for the lambda calculus, *J. London Math. Soc. (2)* **12**(3), pp. 361–370.

Jacopini, G. [1975]. A condition for identifying two elements of whatever model of combinatory logic, *λ-calculus and computer science theory (Proc. Sympos., Rome, 1975)*, Springer, Berlin, pp. 213–219. Lecture Notes in Comput. Sci., Vol. 37.

Johnstone, Peter T. [1986]. *Stone spaces*, Cambridge University Press, Cambridge. Reprint of the 1982 edition.

Joly, Th. [2001]. Constant time parallel computations in lambda-calculus, *TCS* **266**(1), pp. 975–985.

Joly, Th. [2002]. About λ-definability and combinatoric generation of types, Thierry.Joly@pps.jussieu.fr, to be submitted as On λ-definability II, the Fixed Type Problem and Finite Generation of Types.

Joly, Th. [2005]. On λ-Definability I: the Fixed Model Problem and Generalizations of the Matching Problem., *Fundamenta Informaticae* **66**, pp. 1–17.

Jones, J. P. [1982]. Universal Diophantine equation, *J. Symbolic Logic* **47**(3), pp. 549–571.

Jutting, L.S. van Benthem [1977]. *Checking Landau's "Grundlagen" in the AUTOMATH System*, Dissertation, Eindhoven University of Technology.

Kleene, S. C. [1959a]. Countable functionals, *Constructivity in mathematics: Proceedings of the colloquium held at Amsterdam, 1957 (edited by A. Heyting)*, Studies in Logic and the Foundations of Mathematics, North-Holland Publishing Co., Amsterdam, pp. 81–100.

Kleene, S. C. [1959b]. Recursive functionals and quantifiers of finite types. I, *Trans. Amer. Math. Soc.* **91**, pp. 1–52.

Klop, J.W. [1980]. *Combinatory Reduction Systems*, Dissertation, Utrecht University. Appeared as Mathematical Centre Tracts 127, Kruislaan 413, 1098 SJ Amsterdam. Available at URL ⟨`web.mac.com/janwillemklop/iWeb/Site/Bibliography.html`⟩.

Kozen, Dexter [1997]. *Automata and Computability*, Undergraduate Texts in Computer Science, Springer-Verlag, New York. first edition.

Kreisel, G. [1959]. Interpretation of analysis by means of constructive functionals of finite types, *in:* A. Heyting (ed.), *Constructivity in Mathematics*, North-Holland, Amsterdam, pp. 101–128. Studies in Logic and the Foundations of Mathematics.

Krivine, Jean-Louis [1990]. *Lambda-calcul Types et modèles*, Masson, Paris.

Kurata, T. and M. Takahashi [1995]. Decidable properties of intersection type systems, *in:* G. Plotkin M. Dezani-Ciancaglini (ed.), *Proc. Typed Lambda Calculi and Applications '95*, LNCS, Springer, pp. 297–311.

Kuśmierek, Dariusz [2007]. The inhabitation problem for rank two intersection types, *in:* Simona Ronchi Della Rocca (ed.), *Proc. Typed Lambda Calculi and Applications '07*, LNCS 4583, pp. 240–254.

Lambek, J. [1980]. From λ-calculus to Cartesian closed categories, *in: Seldin and (Eds.) [1980]*, Academic Press, London, pp. 375–402.

Landau, E. [1960]. *Grundlagen der Analysis*, 3-rd edition edition, Chelsea Publishing Company.

Leivant, D. [1983]. Polymorphic type inference, *Proc. of 10-th ACM Symposium on Principles of Programming Languages*, Austin, Texas, pp. 88–98.

Leivant, Daniel [1986]. Typing and computational properties of lambda expressions, *Theoret. Comput. Sci.* **44**(1), pp. 51–68.

Liquori, Luigi and Simona Ronchi Della Rocca [2005]. Towards an intersection typed system a la Church, *in:* Mario Coppo and Ferruccio Damiani (eds.), *Workshop on Intersection Types and Related Systems*, ENTCS 136, Elsevier.

Loader, R. [1997]. An algorithm for the minimal model, Unpublished manuscript, obtainable from ⟨`homepages.ihug.co.nz/~suckfish/papers/papers.html`⟩.

Loader, R. [2001a]. Finitary PCF is not decidable, *TCS* **266**(1-2), pp. 341–364.

Loader, R. [2001b]. The undecidability of lambda definability, *Church memorial volume*, Reidel.

Loader, R. [2003]. Higher order β matching is undecidable, *Log. J. IGPL* **11**(1), pp. 51–68.

Longo, G. [1983]. Set-theoretical models of λ-calculus: theories, expansions, isomorphisms, *Ann. Pure Appl. Logic* **24**(2), pp. 153–188.

Longo, G. [1987]. On Church's formal theory of functions and functionals, *Technical Report 9*, Department of Computing Science, Pisa University.

Lopez-Escobar, E.G.K. [n.d.]. Proof functional connectives, *Proc. Methods in Mathematical Logic*, number 1130 in *LNiM*, Berlin, pp. 208–221.

Luo, Zhaohui and Robert Pollack [1992]. The LEGO proof development system: A user's manual, *Technical Report ECS-LFCS-92-211*, University of Edinburgh.

Mairson, Harry G. [1992]. A simple proof of a theorem of Statman, *Theoret. Comput. Sci.* **103**(2), pp. 387–394.

Makanin, G.S. [1977]. The problem of solvability of equations in a free semigroup, *Mathematics of the USSR-Sbornik* **32**, pp. 129–198.

Martin-Löf, P. [1984]. *Intuitionistic Type Theory*, Studies in Proof Theory, Bibliopolis, Napoli.

Matijasevič, Yu. V. [1971]. Diophantine representation of recursively enumerable predicates, *Proceedings of the Second Scandinavian Logic Symposium (Univ. Oslo, Oslo, 1970)*, Studies in Logic and the Foundations of Mathematics, Vol. 63, North-Holland, Amsterdam, pp. 171–177.

Matiyasevič, Yuri V. [1993]. *Hilbert's tenth problem*, Foundations of Computing Series, MIT Press, Cambridge, MA. Translated from the 1993 Russian original by the author, With a foreword by Martin Davis.

Mayr, Richard and Tobias Nipkow [1998]. Higher-order rewrite systems and their confluence, *Theoret. Comput. Sci.* **192**(1), pp. 3–29. Rewriting systems and applications (Kaiserslautern, 1995).

Meyer, Albert R. [1982]. What is a model of the lambda calculus?, *Inform. and Control* **52**(1), pp. 87–122.

Mints, G. [1996]. Normal forms for sequent derivations, *in:* P. Odifreddi (ed.), *Kreiseliana. About and Around Georg Kreisel*, A.K. Peters, Wellesley, Massachusetts, pp. 469–492.

Mints, G. E. [1989]. The completeness of provable realizability, *Notre Dame J. Formal Logic* **30**, pp. 420–441.

Mitchell, John [1996]. *Foundation for Programmimg Languages*, MIT Press.

Mitschke, Gerd [1976]. λ-kalkül, δ-konversion und axiomatische rekursionstheorie, *Technical Report Preprint 274*, Technische Hochschule, Darmstadt.

Nederpelt, R.P., J.H. Geuvers and R.C. de Vrijer (eds.) [1994]. *Selected Papers on Automath*, Studies in Logic and the Foundations of Mathematics **133**, North-Holland, Amsterdam.

Oostdijk, M. [1996].
Proof by calculation, Master's thesis, 385, Universitaire School voor Informatica, Catholic University Nijmegen.

Padovani, V. [1996]. *Filtrage d'ordre supérieur*, Dissertation, Université de Paris VII. Thèse de Doctorat.

Padovani, V. [2000]. Decidability of fourth order matching, *Mathematical Structures in Computer Science* **3**(10), pp. 361 – 372.

Park, D. [1976]. The Y-combinator in Scott's λ-calculus models (revised version), Theory of Computation Report 13, Department of Computer Science, University of Warick.

Paulin-Mohring, C. [1994]. Inductive definitions in the system coq; rules and properties, *Bezem and Groote [1993]*, number 664 in *Lecture Notes in Computer Science*, Springer Verlag, Berlin, pp. 328–345.

Péter, R. [1967]. *Recursive functions*, third revised edition edition, Academic Press, New York.

Pimentel, Elaine, Simona Ronchi Della Rocca and Luca Roversi [2005]. Intersection types: a proof-theoretical approach, To appear in Proceedings of STRUCTURES AND DEDUCTION - ICALP Workshop Lisbon July 16-17.

Platek, R.A. [1966]. *Foundations of recursions theory*, Dissertation, Stanford University.

Plotkin, G. [1977]. LCF considered as a programming language, *Theoretical Computer Science* **5**, pp. 225–255.

Plotkin, G. [1985?]. Personal communication, email.

Plotkin, G. D. [1975]. Call-by-name, call-by-value and the λ-calculus, *Theoret. Comput. Sci.* **1**(2), pp. 125–159.

Plotkin, Gordon D. [1993]. Set-theoretical and other elementary models of the λ-calculus, *Theoret. Comput. Sci.* **121**(1-2), pp. 351–409. A collection of contributions in honour of Corrado Böhm on the occasion of his 70th birthday.

Poincaré, H. [1902]. *La Science et l'Hypothèse*, Flammarion, Paris.

Post, E. [1947]. Recursive unsolvability of a problem of thue, *Journal of Symbolic Logic* **12**(1), pp. 1–11.

Pottinger, G. [1977]. Normalization as a homomorphic image of cut–elimination, *Annals of Mathematical Logic* **12**, pp. 323–357.

Pottinger, G. [1981]. The church-rosser theorem for the typed λ-calculus with surjective pairing, *Notre Dame J. Formal Logic* **22**(3), pp. 264–268.

Pottinger, Garrel [1980]. A type assignment for the strongly normalizable λ-terms, *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, Academic Press, London, pp. 561–577.

Pravato, A., S. Ronchi and L. Roversi [1999]. The call-by-value lambda calculus: a semantic investigation, *Mathematical structures in computer science* **9**(5), pp. 617–650.

Prawitz, D. [1965]. *Natural Deduction*, Almqvist & Wiksell, Stockholm.

Prawitz, D. [1971]. Ideas and results in proof theory, *in:* J. E. Fenstad (ed.), *Proceedings of the 2nd Scandinavian Logic Symposium*, North-Holland, Amsterdam, pp. 235–307.

Raamsdonk, F. van [1996]. *Confluence and Normalisation for Higher-Order Rewriting*, Dissertation, Vrije Universiteit, Amsterdam, The Netherlands.

Raamsdonk, F. van and P. Severi [1995]. On normalisation, *Technical Report 20*, Computing Science Notes, Eindhoven University of Technology.

Ramsey, F.P. [1925]. The foundations of mathematics, *Proceedings of the London Mathematical Society* pp. 338–384.

Ronchi Della Rocca, Simona [1988]. Lecture notes on semantics and types. Internal Report, Torino University.

Ronchi Della Rocca, Simona [2002]. Intersection typed lambda-calculus, *ITRS 2002*, ENTCS 70.1, Elsevier.

Ronchi Della Rocca, Simona and Luca Paolini [2004]. *THE PARAMETRIC LAMBDA CALCULUS A Metamodel for Computation*, Texts in Theoretical Computer Science. An EATCS Series XIII, Springer-Verlag.

Russell, Bertrand A.W. and Alfred North Whitehead [1910–13]. *Principia Mathematica*, Cambridge University Press.

Schmidt-Schauß, Manfred [1999]. Decidability of behavioural equivalence in unary PCF, *Theoret. Comput. Sci.* **216**(1-2), pp. 363–373.

Schwichtenberg, H. [1997]. Termination of permutative conversion inintuitionistic Gentzen calculi, *Theoretical Computer Science*. To appear.

Schwichtenberg, H. and U. Berger [1991]. An inverse of the evaluation functional for typed λ-calculus, *Proceedings of Sixth Annual IEEE Symposium on Logic in Computer Science*, IEEE, pp. 203–211.

Schwichtenberg, Helmut [1975]. Elimination of higher type levels in definitions of primitive recursive functionals by means of transfinite recursion, *Logic Colloquium '73 (Bristol, 1973)*, North-Holland, Amsterdam, pp. 279–303. Studies in Logic and the Foundations of Mathematics, Vol. 80.

Scott, Dana [1972]. Continuous lattices, *Toposes, algebraic geometry and logic (Conf., Dalhousie Univ., Halifax, N. S., 1971)*, Springer, Berlin, pp. 97–136. Lecture Notes in Math., Vol. 274.

Scott, Dana [1975a]. Some philosophical issues concerning theories of combinators, pp. 346–366.

Scott, D.S. [1970]. Constructive validity, *in:* D. Lacombe M. Laudet and M. Schuetzenberger (eds.), *Symposium on Automated Demonstration*, Lecture Notes in Mathematics 125, Springer, Berlin, pp. 237–275.

Scott, D.S. [1975b]. Open problem, *Lambda Calculus and Computer Science Theory*, LNCS 37, Springer, Berlin, p. 368.

Scott, D.S. [1980]. Relating theories of the λ-calculus, *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, Academic Press, pp. 403–450.

Seldin, J.P. and J.R. Hindley (Eds.) [1980]. *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press.

Severi, P. and E. Poll [1994]. Pure type systems with definitions, *in:* A. Nerode and Yu.V. Matiyasevich (eds.), *Proceedings of LFCS'94 (LNCS* **813***)*, LFCS'94, St. Petersburg, Russia, Springer Verlag, New York, pp. 316–328.

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, Proc. Symp. in pure mathematics V*, AMS, Providence, pp. 1–27.

Statman, R. [1982]. Completeness, invariance and λ-definability, *J. Symbolic Logic* **47**(1), pp. 17–26.

Statman, R. [1985]. Equality between functionals revisited, in *Harrington et al. [1985]*, North-Holland, Amsterdam, pp. 331–338.

Statman, R. [2000]. Church's lambda delta calculus, *Logic for programming and automated reasoning (Reunion Island, 2000)*, Lecture Notes in Comput. Sci. 1955, Springer, Berlin, pp. 293–307.

Statman, Richard [1979]. The typed λ-calculus is not elementary recursive, *Theoret. Comput. Sci.* **9**(1), pp. 73–81.

Statman, Richard [1980a]. On the existence of closed terms in the typed λ-calculus. I, in *Seldin and (Eds.) [1980]*, Academic Press, London, pp. 511–534.

Statman, Richard [1980b]. On the existence of closed terms in the typed λ-calculus. III, Dept. of Mathematics, CMU, Pittsburgh, USA.

Tait, W. W. [1967]. Intensional interpretations of functionals of finite type. I, *J. Symbolic Logic* **32**, pp. 198–212.

Tait, William W. [1965]. Infinitely long terms of transfinite type I, *in:* J. Crossley and M. Dummett (eds.), *Formal Systems and Recursive Functions*, North-Holland, pp. 176–185.

Tait, W.W. [1971]. Normal form theorem for barrecursive functions of finite type, *in:* J.E. Fenstad (ed.), *Proceedings of the 2nd Scandinavian Logic Symposium*, North-Holland, Amsterdam, pp. 353–367.

Tatsuta, Makoto and Mariangiola Dezani-Ciancaglini [2006]. Normalisation is Insensible to Lambda-term Identity or Difference, *in:* Rajeev Alur (ed.), *LICS'06*, IEEE Computer Society, pp. 327–336.

Terese [2003]. *Term Rewriting Systems*, Cambridge University Press.

Terlouw, Jan [1982]. On definition trees of ordinal recursive functionals: reduction of the recursion orders by means of type level raising, *J. Symbolic Logic* **47**(2), pp. 395–402.

Thatcher, J.W. [1973]. *Tree automata: an informal survey*, Vol. Currents in the Theory of Computing, Prentice-Hall, chapter 4, pp. 143–172.

Troelstra, A. S. [1998]. Marginalia on sequent calculi, *Studia Logica*. To appear.

Troelstra, A. S. and H. Schwichtenberg [1996]. *Basic Proof Theory*, Cambridge University Press, Cambridge, UK.

Troelstra, A.S. [1973]. *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis*, number 344 in *Lecture Notes in Mathematics*, Springer-Verlag, Berlin.

Urzyczyn, Pawel [1999]. The emptiness problem for intersection types, *J. Symbolic Logic* **64**(3), pp. 1195–1215.

Venneri, B. [1994]. Intersection types as logical formulae, *J. Logic Computat.* **4**(2), pp. 109–124.

Venneri, Betti [1996]. Private communication, Florence University.

Vogel, H. [1976]. Ein starker Normalisationssatz für die barrekursiven Funktionale, *Archiv für Mathematische Logik und Grundlagenforschung* **18**, pp. 81–84.

de Vrijer, R. [1987]. Exactly estimating functionals and strong normalization, *Indagationes Mathematicae* **49**, pp. 479–493.

Wadsworth, Christopher P. [1976]. The relation between computational and denotational properties for Scott's D_∞-models of the lambda-calculus, *SIAM J. Comput.* **5**(3), pp. 488–521. Semantics and correctness of programs.

Waite, W. and G. Goos [1984]. *Compiler Construction*, Springer.

Wells, J. B. [1999]. Typability and type checking in system F are equivalent and undecidable, *Annals of Pure and Applied Logic* **98**(1-3), pp. 111–156.

Wells, J.B., A. Dimock, R. Muller and F. Turbak [1997]. A typed intermediate language for flow-directed compilation, *Proc. 7th International Joint Conference on the Theory and Practice of Software Development, Lille, France*, number 1214 in *LNCS*, pp. 757–771.

van Wijngaarden, A., B.J. Mailloux, J.E.L. Peck, C.H.A. Koster, M. Sintzoff, C.H. Lindsey, L.G.L.T. Meertens and R.G. Fisker (eds.) [1976]. *Revised Report on the Algorithmic Language* ALGOL 68, Springer Verlag, Berlin.

Zucker, J. [1974]. Cut–elimination and normalization, *Annals of Mathematical Logic* **7**, pp. 1–112.