

LAMBDA CALCULUS WITH TYPES

$\lambda_{\rightarrow}^{\mathbb{A}}$ $\lambda_{=}^{\mathcal{A}}$ $(\lambda_{\leq}^{\mathcal{S}})$ $\lambda_{\cap}^{\mathcal{S}}$

HENK BARENDEGT

WIL DEKKERS

RICHARD STATMAN

PERSPECTIVES IN LOGIC
CAMBRIDGE UNIVERSITY PRESS
ASSOCIATION OF SYMBOLIC LOGIC

September 1, 2010

Preface

This book is about typed lambda terms using *simple*, *recursive* and *intersection* types. In some sense it 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 in the period the book was written, notably the undecidability of lambda definability in finite models, the undecidability of second order typability, the decidability of the unique maximal theory extending $\beta\eta$ -conversion and the fact that the collection of closed terms of not every simple type is finitely generated, and the decidability of matching at arbitrary types higher than order 4. The book is not written as an encyclopedic monograph: many topics are only partially treated. For example reducibility among types is analyzed only for simple types built up from only one atom.

One of the recurring distinctions made in the book 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. The book is mainly about Curry typing, although some chapters treat the equivalent Church variant.

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 book will attract readers and inspire them to pursue the topic.

Acknowledgments

Many thanks are due to many people and institutions. The first author obtained substantial support in the form of a generous personal research grant by the Board of Directors of Radboud University, and the Spinoza Prize by The Netherlands Organisation for Scientific Research (NWO). Not all of these means were used to produce this book, but they have been important. The Mathematical Forschungsinstitut at Oberwolfach, Germany, provided hospitality through their ‘Research in Pairs’ program. The Residential Centre at Bertinoro of the University of Bologna hosted us in their stunning castle. The principal regular sites where the work was done have been the Institute for Computing and Information Sciences of Radboud University at Nijmegen, The Netherlands, the Department of Mathematics of Carnegie-Mellon University at Pittsburgh, USA, the Departments of Informatics at the Universities of Torino and Udine, both Italy.

The three main authors wrote the larger part of Part I and thoroughly edited Part II, written by Mario Coppo and Felice Cardone, and Part III, written by Mariangiola Dezani-Ciancaglini, Fabio Alessi, Furio Honsell, and Paula Severi. Some Chapters or Sections have been written by other authors as follows: Chapter 4 by Gilles Dowek, Sections 5C-5E by Marc Bezem, Section 6D by Michael Moortgat and Section 17E by Paweł Urzyczyn, while Section 6C was coauthored by Silvia Ghilezan. This ‘thorough editing’ consisted of rewriting the material to bring all in one style, but in many cases also in adding results and making corrections. It was agreed upon beforehand with all coauthors that this could happen.

Since 1974 Jan Willem Klop has been a close colleague and friend for many years and we engaged with him many inspiring discussions on λ -calculus and types.

Several people helped during the later phases of writing the book. The reviewer Roger Hindley gave invaluable advise. Vincent Padovani carefully read Section 4C. Other help came from Jörg Endrullis, Clemens Grabmeyer, Thierry Joly, Jan Willem Klop, Pieter Koopman, Dexter Kozen, Giulio Manzonetto, James McKinna, Vincent van Oostrom, Rinus Plasmeijer, [Arnoud van Rooij](#), Jan Rutten, [Sylvain Salvati](#), [Christian Urban](#), [Bas Westerbaan](#), and [Bram Westerbaan](#).

Use has been made of the following macro packages: ‘prooftree’ of Paul Taylor, ‘xypic’ of Kristoffer Rose, ‘robustindex’ of Wilberd van der Kallen, and several lay-out commands of Erik Barendsen.

At the end producing this book turned out a time consuming enterprise. But that seems to be the way: while the production of the content of [Barendregt \[1984\]](#) was thought to last two months, it took fifty months; for this book the initial estimation was four years, while it turned out to be eighteen years(!).

Our partners were usually patiently understanding when we spent [yet](#) another period of writing and rewriting. We cordially thank them for their continuous and continuing support and [love](#).

Nijmegen and Pittsburgh

September 1, 2010

Henk Barendregt^{1,2}

Wil Dekkers¹

Rick Statman²

¹Faculty of Science
Radboud University, Nijmegen, The Netherlands
²Departments of Mathematics and Computer Science
Carnegie-Mellon University, Pittsburgh, USA

The founders of the topic of this book are Alonzo Church (1903-1995), who invented the lambda calculus ([Church \[1932\]](#), [Church \[1933\]](#)), and Haskell Curry (1900-1982), who invented ‘notions of functionality’ ([Curry \[1934\]](#)) that later got transformed into types for the hitherto untyped lambda terms. As a tribute to Church and Curry the next pages show pictures of them at an early stage of their careers. Church and Curry have been honored jointly for their timeless invention by the Association for Computing Machinery in 1982.



Alonzo Church (1903-1995)
Studying mathematics at Princeton University (1922 or 1924).
Courtesy of Alonzo Church and Mrs. Addison-Church.



Haskell B. Curry (1900-1982)
BA in mathematics at Harvard (1920).
Courtesy of Town & Gown, Penn State.

Contributors

Fabio Alessi Department of Mathematics and Computer Science Udine University	Part 3, except §17E
Henk Barendregt Institute of Computing & Information Science Radboud University Nijmegen	All parts, except §§5C, 5D, 5E, 6D
Marc Bezem Department of Informatics Bergen University	§§5C, 5D, 5E
Felice Cardone Department of Informatics Torino University	Part 2
Mario Coppo Department of Informatics Torino University	Part 2
Wil Dekkers Institute of Computing & Information Science Radboud University Nijmegen	All parts, except §§5C, 5D, 5E, 6C, 6D, 17E
Mariangiola Dezani-Ciancaglini Department of Informatics Torino University	Part 3, except §17E
Gilles Dowek Department of Informatics École Polytechnique and INRIA	Chapter 4
Silvia Ghilezan Center for Mathematics & Statistics University of Novi Sad	§6C
Furio Honsell Department of Mathematics and Computer Science Udine University	Part 3, except §17E
Michael Moortgat Department of Modern Languages Utrecht University	§6D
Paula Severi Department of Computer Science University of Leicester	Part 3, except §17E
Richard Statman Department of Mathematics Carnegie-Mellon University	Parts 1, 2, except §§5C, 5D, 5E, 6D
Paweł Urzyczyn Institute of Informatics Warsaw University	§17E.

Contents in short

PREFACE	ii
CONTRIBUTORS	viii
INTRODUCTION	xv
Part 1. Simple types λ_{\rightarrow}^A.	1
CHAPTER 1. THE SIMPLY TYPED LAMBDA CALCULUS.....	5
CHAPTER 2. PROPERTIES.....	45
CHAPTER 3. TOOLS	75
CHAPTER 4. DEFINABILITY, UNIFICATION AND MATCHING.....	151
CHAPTER 5. EXTENSIONS	185
CHAPTER 6. APPLICATIONS	245
Part 2. Recursive types λ_{\equiv}^A.	289
CHAPTER 7. THE SYSTEMS λ_{\equiv}^A	291
CHAPTER 8. PROPERTIES OF RECURSIVE TYPES.....	349
CHAPTER 9. PROPERTIES OF TERMS WITH TYPES	383
CHAPTER 10. MODELS	403
CHAPTER 11. APPLICATIONS	431
Part 3. Intersection types λ_{\cap}^S.	449
CHAPTER 12. AN EXEMPLARY SYSTEM.....	451
CHAPTER 13. TYPE ASSIGNMENT SYSTEMS	461
CHAPTER 14. BASIC PROPERTIES.....	483
CHAPTER 15. TYPE AND LAMBDA STRUCTURES	501
CHAPTER 16. FILTER MODELS.....	533
CHAPTER 17. ADVANCED PROPERTIES AND APPLICATIONS	571
BIBLIOGRAPHY	623
INDICES	657
Definitions	658
Names.....	669
Symbols	675

Contents

Preface	ii
Contributors	viii
Contents in short	ix
Introduction	xv

Part 1. Simple types $\lambda_{\rightarrow}^{\mathbb{A}}$

CHAPTER 1. THE SIMPLY TYPED LAMBDA CALCULUS.....	
1A	The systems $\lambda_{\rightarrow}^{\mathbb{A}}$
1B	First properties and comparisons
1C	Normal inhabitants.....
1D	Representing data types
1E	Exercises
CHAPTER 2. PROPERTIES	
2A	Normalization
2B	Proofs of strong normalization
2C	Checking and finding types
2D	Checking inhabitation
2E	Exercises
CHAPTER 3. TOOLS	
3A	Semantics of λ_{\rightarrow}
3B	Lambda theories and term models
3C	Syntactic and semantic logical relations.....
3D	Type reducibility
3E	The five canonical term-models
3F	Exercises
CHAPTER 4. DEFINABILITY, UNIFICATION AND MATCHING	
4A	Undecidability of lambda definability
4B	Undecidability of unification
4C	Decidability of matching of rank 3
4D	Decidability of the maximal theory
4E	Exercises
CHAPTER 5. EXTENSIONS	
	185

5A	Lambda delta	185
5B	Surjective pairing	194
5C	Gödel's system \mathcal{T} : higher-order primitive recursion	215
5D	Spector's system \mathcal{B} : bar recursion	229
5E	Platek's system \mathcal{Y} : fixed point recursion	236
5F	Exercises	238
CHAPTER 6. APPLICATIONS		245
6A	Functional programming	245
6B	Logic and proof-checking	259
6C	Proof theory	267
6D	Grammars, terms and types	276
Part 2. Recursive Types λ_{\equiv}^A		
CHAPTER 7. THE SYSTEMS λ_{\equiv}^A		291
7A	Type-algebras and type assignment	291
7B	More on type algebras	300
7C	Recursive types via simultaneous recursion	305
7D	Recursive types via μ -abstraction	313
7E	Recursive types as trees	326
7F	Special views on trees	336
7G	Exercises	341
CHAPTER 8. PROPERTIES OF RECURSIVE TYPES		349
8A	Simultaneous recursions vs μ -types	349
8B	Properties of μ -types	352
8C	Properties of types defined by an sr over \mathbb{T}	368
8D	Exercises	380
CHAPTER 9. PROPERTIES OF TERMS WITH TYPES		383
9A	First properties of λ_{\equiv}^A	383
9B	Finding and inhabiting types	385
9C	Strong normalization	393
9D	Exercises	401
CHAPTER 10. MODELS		403
10A	Interpretations of type assignments in λ_{\equiv}^A	403
10B	Interpreting \mathbb{T}_μ and \mathbb{T}_μ^*	407
10C	Type interpretations in systems with explicit typing	419
10D	Exercises	425
CHAPTER 11. APPLICATIONS		431
11A	Subtyping	431
11B	The principal type structures	441
11C	Recursive types in programming languages	443
11D	Further reading	446
11E	Exercises	448

Part 3. Intersection types λ_{\cap}^S

CHAPTER 12. AN EXEMPLARY SYSTEM	451
12A The type assignment system λ_{\cap}^{BCD}	452
12B The filter model \mathcal{F}^{BCD}	457
12C Completeness of type assignment	458
CHAPTER 13. TYPE ASSIGNMENT SYSTEMS	461
13A Type theories	463
13B Type assignment	473
13C Type structures	476
13D Filters	480
13E Exercises	481
CHAPTER 14. BASIC PROPERTIES	483
14A Inversion lemmas	485
14B Subject reduction and expansion	490
14C Exercises	495
CHAPTER 15. TYPE AND LAMBDA STRUCTURES	501
15A Meet semi-lattices and algebraic lattices	504
15B Natural type structures and lambda structures	513
15C Type and zip structures	518
15D Zip and lambda structures	522
15E Exercises	529
CHAPTER 16. FILTER MODELS	533
16A Lambda models	535
16B Filter models	540
16C \mathcal{D}_{∞} models as filter models	549
16D Other filter models	562
16E Exercises	568
CHAPTER 17. ADVANCED PROPERTIES AND APPLICATIONS	571
17A Realizability interpretation of types	573
17B Characterizing syntactic properties	577
17C Approximation theorems	582
17D Applications of the approximation theorem	594
17E Undecidability of inhabitation	597
17F Exercises	617
BIBLIOGRAPHY	623
INDICES	657
Index of definitions	658
Index of names	669
Index of symbols	675

Introduction

The rise of lambda calculus

Lambda calculus started as a formalism introduced by Church in 1932 intended to be used as a foundation for mathematics, including the computational aspects. Supported by his students Kleene and Rosser—who showed that the prototype system was inconsistent—Church distilled a consistent computational part and ventured in 1936 the Thesis that exactly the intuitively computable functions can be defined in it. He also presented a function that could not be captured by the λ -calculus. In that same year Turing introduced another formalism, describing what are now called Turing Machines, and formulated the related Thesis that exactly the mechanically computable functions can be captured by these machines. Turing also showed in the same paper that the question whether a given statement could be proved (from a given set of axioms) using the rules of any reasonable system of logic is not computable in this mechanical way. Finally Turing showed that the formalism of λ -calculus and Turing machines define the same class of functions.

Together Church's Thesis, concerning computability by *homo sapiens*, and Turing's Thesis, concerning computability by mechanical devices, using formalisms that are equally powerful but having their computational limitations, made a deep impact on the philosophy in the 20th century concerning the power and limitations of the human mind. So far, cognitive neuropsychology has not been able to refute the combined Church-Turing Thesis. On the contrary, also this discipline shows the limitation of human capacities. On the other hand, the analyses of Church and Turing indicate an element of reflection (universality) in both Lambda Calculus and Turing Machines, that according to their combined thesis is also present in humans.

Turing Machine computations are relatively easy to implement on electronic devices, as started to happen soon in the 1940s. The mentioned universality was employed by von Neumann¹ enabling to construct not only ad hoc computers but even a universal one, capable of performing different tasks depending on a program. This resulted in what is called now *imperative programming*, with the language C presently as the most widely used one for programming in this paradigm. Like with Turing Machines a computation consists of repeated modifications of some data stored in memory. The essential difference between a modern computer and a Turing Machine is that the former has random access memory².

Functional programming

The computational model of Lambda Calculus, on the other hand, has given rise to *functional programming*. The input M becomes part of an expression FM to be evaluated, where F represents the intended function to be computed on M . This expression is

¹It was von Neumann who visited Cambridge UK in 1935 and invited Turing to Princeton during 1936-1937, so he probably knew Turing's work.

²Another difference is that the memory on a TM is infinite: Turing wanted to be technology independent, but was restricting a computation with given input to one using finite memory and time.

reduced (rewritten) according to some rules (indicating the possible computation steps) and some strategy (indicating precisely which steps should be taken).

To show the elegance of functional programming, here is a short functional program generating primes using Eratosthenes sieve (Miranda program by D. Turner):

```
primes = sieve [2..]
  where
    sieve (p:x) = p : sieve [n | n<-x ; n mod p > 0]
```

```
primes_up to n = [p | p<- primes ; p<n]
```

while a similar program expressed in an imperative language looks like (Java program from rosettacode.org)

```
public class Sieve{
  public static LinkedList<Integer> sieve(int n){
    LinkedList<Integer> primes = new LinkedList<Integer>();
    BitSet nonPrimes = new BitSet(n+1);

    for (int p = 2; p <= n; p = nonPrimes.nextClearBit(p+1)){
      for (int i = p * p; i <= n; i += p)
        nonPrimes.set(i);
      primes.add(p);
    }
    return primes;
  }
}
```

Of course the algorithm is extremely simple, one of the first ever invented. However, the gain for more complex algorithms remains, as functional programs do scale up.

The power of functional programming languages derives from several facts.

1. All expressions of a functional programming language have a constant meaning (i.e. independent of a hidden state). This is called ‘referential transparency’ and makes it easier to reason about functional programs and to make versions for parallel computing, important for quality and efficiency.
2. Functions may be arguments of other functions, usually called ‘functionals’ in mathematics and higher order functions in programming. There are functions acting on functionals, etcetera; in this way one obtains functions of arbitrary order. Both in mathematics and in programming higher order functions are natural and powerful phenomena. In functional programming this enables the flexible composition of algorithms.
3. Algorithms can be expressed in a clear goal-directed mathematical way, using various forms of recursion and flexible data structures. The bookkeeping needed for the storage of these values is handled by the language compiler instead of the user of the functional language³.

³In modern functional languages there is a palette of techniques (like overloading, type classes and generic programming) to make algorithms less dependent of specific data types and hence more reusable. If desired the user of the functional language can help the compiler to achieve a better allocation of values.

Types

The formalism as defined by Church is untyped. Also the early functional languages, of which Lisp (McCarthy, Abrahams, Edwards, Hart, and Levin [1962]) and Scheme (Abelson, Dybvig, Haynes, Rozas, IV, Friedman, Kohlbecker, Jr., Bartley, Halstead, [1991]) are best known, are untyped: arbitrary expressions may be applied to each other. Types first appeared in Principia Mathematica, Whitehead and Russell [1910-1913]. In Curry [1934] types are introduced and assigned to expressions in ‘combinatory logic’, a formalism closely related to lambda calculus. In Curry and Feys [1958] this type assignment mechanism was adapted to λ -terms, while in Church [1940] λ -terms were ornamented by fixed types. This resulted in the closely related systems $\lambda_{\rightarrow}^{\text{Cu}}$ and $\lambda_{\rightarrow}^{\text{Ch}}$ treated in Part I.

Types are being used in many, if not most programming languages. These are of the form

bool, nat, real, ...

and occur in compounds like

nat \rightarrow bool, array(real), ...

Using the formalism of types in programming, many errors can be prevented if terms are required to be typable: arguments and functions should match. For example M of type A can be an argument only of a function of type $A \rightarrow B$. Types act in a way similar to the use of dimensional analysis in physics. Physical constants and data obtain a ‘dimension’. Pressure p , for example, is expressed as

$$g/m^2$$

giving the constant R in the law of Boyle

$$\frac{pV}{T} = R$$

a dimension that prevents one from writing an equation like $E = TR^2$. By contrast Einstein’s famous equation

$$E = mc^2$$

is already meaningful from the viewpoint of its dimension.

In most programming languages the formation of function space types is usually not allowed to be iterated like in

$$\begin{aligned} (\text{real} \rightarrow \text{real}) \rightarrow (\text{real} \rightarrow \text{real}) & \quad \text{for indefinite integrals } \int f(x)dx; \\ (\text{real} \rightarrow \text{real}) \times \text{real} \times \text{real} \rightarrow \text{real} & \quad \text{for definite integrals } \int_a^b f(x)dx; \\ ([0, 1] \rightarrow \text{real}) \rightarrow (([0, 1] \rightarrow \text{real}) \rightarrow \text{real}) \rightarrow (([0, 1] \rightarrow \text{real}) \rightarrow \text{real}), \end{aligned}$$

where the latter is the type of a map occurring in functional analysis, see Lax [2002]. Here we wrote “[0, 1] \rightarrow real” for what should be more accurately the set $C[0, 1]$ of continuous functions on [0, 1].

Because there is the Hindley-Milner algorithm (see Theorem 2C.14 in Chapter 2) that decides whether an untyped term does have a type and computes the most general type types found their way to functional programming languages. The first such language to incorporate the types of the simply typed λ -calculus is ML (Milner, Tofte, Harper,

and McQueen [1997]). An important aspect of typed expressions is that if a term M is correctly typed by type A , then also during the computation of M the type remains the same (see Theorem 1B.6, the ‘subject reduction theorem’). This is expressed as a feature in functional programming: one only needs to check types during compile time.

In functional programming languages, however, types come of age and are allowed in their full potential by giving a precise notation for the type of data, functions, functionals, higher order functionals, ... up to arbitrary degree of complexity. Interestingly, the use of higher order types given in the mathematical examples is modest compared to higher order types occurring in a natural way in programming situations.

$$\begin{aligned} & [(a \rightarrow ([([b], c)] \rightarrow ([([b], c)])) \rightarrow ([([b], c)] \rightarrow [b] \rightarrow ([([b], c)])) \rightarrow \\ & ([([b], c)] \rightarrow ([([b], c)])) \rightarrow ([([b], c)] \rightarrow [b] \rightarrow ([([b], c)])) \rightarrow \\ & [a \rightarrow (d \rightarrow ([([b], c)] \rightarrow ([([b], c)])) \rightarrow ([([b], c)] \rightarrow [b] \rightarrow ([([b], c)])) \rightarrow \\ & ([([b], c)] \rightarrow ([([b], c)])) \rightarrow ([([b], c)] \rightarrow [b] \rightarrow ([([b], c)])) \rightarrow \\ & [d \rightarrow ([([b], c)] \rightarrow ([([b], c)])) \rightarrow ([([b], c)] \rightarrow [b] \rightarrow ([([b], c)])) \rightarrow \\ & ([([b], c)] \rightarrow ([([b], c)])) \rightarrow ([([b], c)] \rightarrow [b] \rightarrow ([([b], c)])] \end{aligned}$$

This type (it does not actually occur in this form in the program, but is notated using memorable names for the concepts being used) is used in a functional program for efficient parser generators, see Koopman and Plasmeijer [1999]. The type $[a]$ denotes that of lists of type a and (a, b) denotes the ‘product’ $a \times b$. Product types can be simulated by simple types, while for list types one can use the recursive types developed in Part II of this book.

Although in the pure typed λ -calculus only a rather restricted class of terms and types is represented, relatively simple extensions of this formalism have universal computational power. Since the 1970s the following programming languages appeared: ML (not yet purely functional), Miranda (Thompson [1995], www.cs.kent.ac.uk/people/staff/dat/miranda/) the first purely functional typed programming language, well-designed, but slowly interpreted; Clean (van Eekelen and Plasmeijer [1993], Plasmeijer and van Eekelen [2002], wiki.clean.cs.ru.nl/Clean) and Haskell (Hutton [2007], Peyton Jones [2003], www.haskell.org); both Clean and Haskell are state of the art pure functional languages with fast compiler generating fast code). They show that functional programming based on λ -calculus can be efficient and apt for industrial software. Functional programming languages are also being used for the design (Sheeran [2005]) and testing (Koopman and Plasmeijer [2006]) of hardware. In both cases it is the compact mathematical expressivity of the functional languages that makes them fit for the description of complex functionality.

Semantics of natural languages

Typed λ -calculus has also been employed in the semantics of natural languages (Montague [1973], van Benthem [1995]). An early indication of this possibility can already be found in Curry and Feys [1958], Section 8S2.

Certifying proofs

Next to its function for designing, the λ -calculus has also been used for verification, not only for the correctness of IT products, but also of mathematical proofs. The underlying idea is the following. Ever since Aristotle's formulation of the axiomatic method and Frege's formulation of predicate logic one could write down mathematical proofs in full detail. Frege wanted to develop mathematics in a fully formalized way, but unfortunately started from an axiom system that turned out to be inconsistent, as shown by the Russell paradox. In Principia Mathematica Whitehead and Russell used types to prevent the paradox. They had the same formalization goal in mind and developed some elementary arithmetic. Based on this work, Gödel could state and prove his fundamental incompleteness result. In spite of the intention behind Principia Mathematica, proofs in the underlying formal system were not fully formalized. Substitution was left as an informal operation and in fact the way Principia Mathematica treated free and bound variables was implicit and incomplete. Here starts the role of the λ -calculus. As a formal system dealing with manipulating formulas, being careful with free and bound variables, it was the missing link towards a full formalization. Now, if an axiomatic mathematical theory is fully formalized, a computer can verify the correctness of the definitions and proofs. The reliability of computer verified theories relies on the fact that logic has only about a dozen rules and their implementation poses relatively little problems. This idea was pioneered since the late 1960s by N. G. de Bruijn in the proof-checking language and system Automath ([Nederpelt, Geuvers, and de Vrijer \[1994\]](#), <www.win.tue.nl/automath>).

The methodology has given rise to proof-assistants. These are computer programs that help the human user to develop mathematical theories. The initiative comes from the human who formulates notions, axioms, definitions, proofs and computational tasks. The computer verifies the well-definedness of the notions, the correctness of the proofs, and performs the computational tasks. In this way arbitrary mathematical notions can be represented and manipulated on a computer. Many of the mathematical assistants are based on extensions of typed λ -calculus. See Section 6B for more information.

What this book is and is not about

None of the mentioned fascinating applications of lambda calculus with types are treated in this book. We will study the formalism for its mathematical beauty. In particular this monograph focuses on mathematical properties of three classes of typing for lambda terms.

Simple types, constructed freely from type atoms, cause strong normalization, subject reduction, decidability of typability and inhabitation, undecidability of lambda definability. There turn out to be five canonical term models based on closed terms. Powerful extensions with respectively a discriminator, surjective pairing, operators for primitive recursion, bar recursion, and a fixed point operator are being studied. Some of these extensions remain constructive, other ones are utterly non-constructive, and some will be at the edge between these two realms.

Recursive types allow functions to fit as input for themselves, losing strong normalization (restored by allowing only positive recursive types). Typability remains decidable.

Unexpectedly α -conversion, dealing with a hygienic treatment of free and bound variables among recursive types has interesting mathematical properties.

Intersection types allow functions to take arguments of different types simultaneously. Under certain mild conditions this leads to subject conversion, turning the filters of types of a given term into a lambda model. Classical lattice models can be described as intersection type theories. Typability and inhabitation now become undecidable, the latter being equivalent to undecidability of lambda definability for models of simple types.

A flavour of some of the applications of typed lambda calculus is given: functional programming (Section 6A), proof-checking (Section 6B), and formal semantics of natural languages (Section 6C).

What this book could have been about

This book could have been also about dependent types, higher order types and inductive types, all used in some of the mathematical assistants. Originally we had planned a second volume to do so. But given the effort needed to write this book, we will probably not do so. Higher order types are treated in [Girard, Lafont, and Taylor \[1989\]](#), and [Sørensen and Urzyczyn \[2006\]](#). Research monographs on dependent and inductive types are lacking. This is an invitation to the community of next generations of researchers.

Some notational conventions

A *partial function* from a set X to a set Y is a collection of ordered pairs $f \subseteq X \times Y$ such that $\forall x \in X, y, y' \in Y. [\langle x, y \rangle \in f \& \langle x, y' \rangle \in f \Rightarrow y = y']$.

The set of partial functions from a set X to a set Y is denoted by $X \nrightarrow Y$. If $f \in (X \nrightarrow Y)$ and $x \in X$, then $f(x)$ is *defined*, notation $f(x) \downarrow$ or $x \in \text{dom}(f)$, if for some y one has $\langle x, y \rangle \in f$. In that case one writes $f(x) = y$. On the other hand $f(x)$ is *undefined*, notation $f(x) \uparrow$, means that for no $y \in Y$ one has $\langle x, y \rangle \in f$. An expression E in which partial functions are involved, may be defined or not. If two such expressions are compared, then, following [Kleene \[1952\]](#), we write $E \simeq E_2$ for

if $E_1 \downarrow$, then $E_2 \downarrow$ and $E_1 = E_2$, and vice versa.

The set of natural numbers is denoted by \mathbb{N} . In proofs formula numbers like (1), (2), etcetera, are used to indicate formulas locally: different proofs may use the same numbers. The notation \triangleq is used for “equality by definition”. Similarly ‘ $\xleftarrow{\triangleq}$ ’ is used for the definition of a concept. By contrast $::=$ stands for the more specific introduction of a syntactic category defined by the Backus-Naur form. The notation \equiv stands for syntactic equality (for example to remember the reader that the LHS was defined previously as the RHS). In a definition we do not write ‘ M is *closed* iff $\text{FV}(M) = \emptyset$ ’ but ‘ M is *closed* if $\text{FV}(M) = \emptyset$ ’. The end of a proof is indicated by ‘■’.

Part 1

SIMPLE TYPES $\lambda_{\rightarrow}^{\mathbb{A}}$

The systems of *simple types* considered in Part I are built up from atomic types \mathbb{A} using as only operator the constructor \rightarrow of forming function spaces. For example, from the atoms $\mathbb{A} = \{\alpha, \beta\}$ one can form types $\alpha \rightarrow \beta$, $(\alpha \rightarrow \beta) \rightarrow \alpha$, $\alpha \rightarrow (\alpha \rightarrow \beta)$ and so on. Two choices of the set of atoms that will be made most often are $\mathbb{A} = \{\alpha_0, \alpha_1, \alpha_2, \dots\}$, an infinite set of type variables giving $\lambda_{\rightarrow}^{\infty}$, and $\mathbb{A} = \{0\}$, consisting of only one atomic type giving λ_{\rightarrow}^0 . 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.

Requiring terms to have simple types implies that they are strongly normalizing. 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. Insights coming from this reducibility provide five canonical term models of λ_{\rightarrow}^0 . See next two pages for types and terms involved in this analysis.

The problem of unification

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

is for complex enough A undecidable. That of pattern matching

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

will be shown to be decidable for A up to ‘rank 3’. The recent proof by Stirling of general decidability of matching is not included. The terms of finite type are extended by δ -functions, functionals for primitive recursion (Gödel) and bar recursion (Spector). Applications of the theory in computing, proof-checking and semantics of natural languages will be presented.

Other expositions of the simply typed lambda calculus are [Church \[1941\]](#), [Lambek and Scott \[1981\]](#), [Girard, Lafont, and Taylor \[1989\]](#), [Hindley \[1997\]](#), and [Nerode, Odifreddi, and Platek \[In preparation\]](#). Part of the history of the topic, including the untyped lambda calculus, can be found in [Crossley \[1975\]](#), [Rosser \[1984\]](#), [Kamareddine, Laan, and Nederpelt \[2004\]](#) and [Cardone and Hindley \[2009\]](#).

Sneak preview of λ_{\rightarrow} (Chapters 1, 2, 3)

Terms

Term <i>variables</i> $V \triangleq \{c, c', c'', \dots\}$ $Terms \Lambda \quad \left\{ \begin{array}{l} x \in V \Rightarrow x \in \Lambda \\ M, N \in \Lambda \Rightarrow (MN) \in \Lambda \\ M \in \Lambda, x \in V \Rightarrow (\lambda x M) \in \Lambda \end{array} \right.$
Notations for terms $x, y, z, \dots, F, G, \dots, \Phi, \Psi, \dots$ range over V M, N, L, \dots range over Λ Abbreviations $\begin{aligned} N_1 \cdots N_n &\triangleq (\cdots (MN_1) \cdots N_n) \\ \lambda x_1 \cdots x_n.M &\triangleq (\lambda x_1 (\cdots (\lambda x_n.M) \cdots)) \end{aligned}$
Standard terms: combinators $I \triangleq \lambda x.x$ $K \triangleq \lambda xy.x$ $S \triangleq \lambda xyz.xz(yz)$

Types

Type <i>atoms</i> $\mathbb{A}_{\infty} \triangleq \{c, c', c'', \dots\}$ $Types \mathbb{T} \quad \left\{ \begin{array}{l} \alpha \in \mathbb{A} \Rightarrow \alpha \in \mathbb{T} \\ A, B \in \mathbb{T} \Rightarrow (A \rightarrow B) \in \mathbb{T} \end{array} \right.$
Notations for types $\alpha, \beta, \gamma, \dots$ range over \mathbb{A}_{∞} A, B, C, \dots range over \mathbb{T} Abbreviation $A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_n \triangleq (A_1 \rightarrow (A_2 \rightarrow \cdots (A_{n-1} \rightarrow A_n) \cdots))$
Standard types: each $n \in \mathbb{N}$ is interpreted as type $n \in \mathbb{T}$ $\begin{aligned} 0 &\triangleq c \\ n+1 &\triangleq n \rightarrow 0 \\ (n+1)_2 &\triangleq n \rightarrow n \rightarrow 0 \end{aligned}$

Assignment of types to terms $\vdash M : A$ ($M \in \Lambda, A \in \mathbb{T}$)

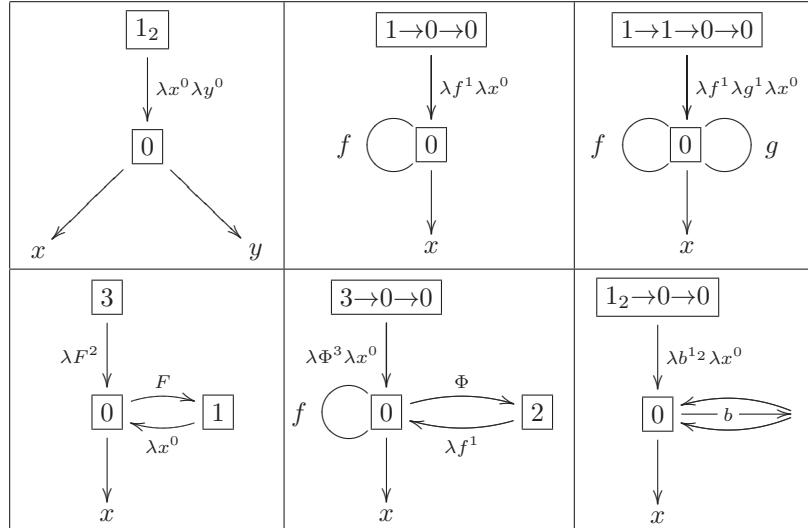
<i>Basis:</i> a set $\Gamma = \{x_1:A_1, \dots, x_n:A_n\}$, with $x_i \in V$ distinct Type <i>assignment</i> (relative to a basis Γ) axiomatized by $\left\{ \begin{array}{l} (x:A) \in \Gamma \Rightarrow \Gamma \vdash x : A \\ \Gamma \vdash M : (A \rightarrow B), \Gamma \vdash N : A \Rightarrow \Gamma \vdash (MN) : B \\ \Gamma, x:A \vdash M : B \Rightarrow \Gamma \vdash (\lambda x.M) : (A \rightarrow B) \end{array} \right.$
Notations for assignment $'x:A \vdash M : B'$ stands for ' $\{x:A\} \vdash M : B$ ' $'\Gamma, x:A'$ for ' $\Gamma \cup \{x:A\}$ ' and ' $\vdash M : A$ ' for ' $\emptyset \vdash M : A$ ' Standard assignments: for all $A, B, C \in \mathbb{T}$ one has $\begin{aligned} \vdash I &: A \rightarrow A && \text{as } x:A \vdash x : A \\ \vdash K &: A \rightarrow B \rightarrow A && \text{as } x:A, y:B \vdash x : A \\ \vdash S &: (A \rightarrow B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow A \rightarrow C && \text{similarly} \end{aligned}$

Canonical term-models built up from constants

The following types A play an important role in Sections 3D, 3E. Their normal inhabitants (i.e. terms M in normal form such that $\vdash M : A$) can be enumerated by the following schemes.

Type	Inhabitants (all possible $\beta\eta^{-1}$ -normal forms are listed)
1_2	$\lambda xy.x, \lambda xy.y.$
$1 \rightarrow 0 \rightarrow 0$	$\lambda fx.x, \lambda fx.fx, \lambda fx.f(fx), \lambda fx.f^3x, \dots$; general pattern: $\lambda fx.f^n x.$
3	$\lambda F.F(\lambda x.x), \lambda F.F(\lambda x.F(\lambda y.x)), \dots ; \lambda F.F(\lambda x_1.F(\lambda x_2. \dots F(\lambda x_n.x_i) \dots)).$
$1 \rightarrow 1 \rightarrow 0 \rightarrow 0$	$\lambda fgx.x, \lambda fgx.fx, \lambda fgx.gx,$ $\lambda fgx.f(gx), \lambda fgx.g(fx), \lambda fgx.f^2x, \lambda fgx.g^2x,$ $\lambda fgx.f(g^2x), \lambda fgx.f^2(gx), \lambda fgx.g(f^2x), \lambda fgx.g^2(fx), \lambda fgx.f(g(fx)), \dots$; $\lambda fgx.w_{\{f,g\}}x,$ where $w_{\{f,g\}}$ is a ‘word over $\Sigma = \{f, g\}$ ’ which is ‘applied’ to x by interpreting juxtaposition ‘ fg ’ as function composition ‘ $f \circ g = \lambda x.f(gx)$ ’.
$3 \rightarrow 0 \rightarrow 0$	$\lambda \Phi x.x, \lambda \Phi x.\Phi(\lambda f.x), \lambda \Phi x.\Phi(\lambda f.fx), \lambda \Phi x.\Phi(\lambda f.f(\Phi(\lambda g.g(fx)))), \dots$ $\lambda \Phi x.\Phi(\lambda f_1.w_{\{f_1\}}x), \lambda \Phi x.\Phi(\lambda f_1.w_{\{f_1\}}\Phi(\lambda f_2.w_{\{f_1,f_2\}}x)), \dots$; $\lambda \Phi x.\Phi(\lambda f_1.w_{\{f_1\}})\Phi(\lambda f_2.w_{\{f_1,f_2\}}) \dots \Phi(\lambda f_n.w_{\{f_1,\dots,f_n\}}x) \dots).$
$1_2 \rightarrow 0 \rightarrow 0$	$\lambda bx.x, \lambda bx.bxx, \lambda bx.bx(bxx), \lambda bx.b(bxx)x, \lambda bx.b(bxx)(bxx), \dots ; \lambda bx.t,$ where t is an element of the context-free language generated by the grammar $\text{tree} ::= x \mid (\text{b tree tree}).$

This follows by considering the inhabitation machine, see Section 1C, for each mentioned type.



We have juxtaposed the machines for types $1 \rightarrow 0 \rightarrow 0$ and $1 \rightarrow 1 \rightarrow 0 \rightarrow 0$, as they are similar, and also those for 3 and $3 \rightarrow 0 \rightarrow 0$. According to the type reducibility theory of Section 3D the types $1 \rightarrow 0 \rightarrow 0$ and 3 are equivalent and therefore they are presented together in the statement.

From the types 1_2 , $1 \rightarrow 0 \rightarrow 0$, $1 \rightarrow 1 \rightarrow 0 \rightarrow 0$, $3 \rightarrow 0 \rightarrow 0$, and $1_2 \rightarrow 0 \rightarrow 0$ five canonical λ -theories and term-models will be constructed, that are strictly increasing (decreasing). The smallest theory is the good old simply typed $\lambda\beta\eta$ -calculus, and the largest theory corresponds to the minimal model, Definition 3E.46, of the simply typed λ -calculus.

CHAPTER 1

THE SIMPLY TYPED LAMBDA CALCULUS

1A. The systems $\lambda_{\rightarrow}^{\mathbb{A}}$

Untyped lambda calculus

Remember the untyped lambda calculus denoted by λ , see e.g. B[1984]⁴.

1A.1. DEFINITION. The set of untyped λ -terms Λ is defined by the following so called ‘*simplified syntax*’. This basically means that parentheses are left implicit.

$V ::= c \mid V'$
$\Lambda ::= V \mid \lambda V \Lambda \mid \Lambda \Lambda$

FIGURE 1. Untyped lambda terms

This makes $V = \{c, c', c'', \dots\}$.

1A.2. NOTATION. (i) $x, y, z, \dots, x_0, y_0, z_0, \dots, x_1, y_1, z_1, \dots$ denote arbitrary variables.

(ii) M, N, L, \dots denote arbitrary lambda terms.

(iii) $MN_1 \dots N_k \triangleq (\dots(MN_1) \dots N_k)$, *association to the left*.

(iv) $\lambda x_1 \dots x_n M \triangleq (\lambda x_1 (\dots (\lambda x_n (M)) \dots))$, *association to the right*.

1A.3. DEFINITION. Let $M \in \Lambda$.

(i) The set of *free variables* of M , notation $\text{FV}(M)$, is defined as follows.

M	$\text{FV}(M)$
x	$\{x\}$
PQ	$\text{FV}(P) \cup \text{FV}(Q)$
$\lambda x.P$	$\text{FV}(P) - \{x\}$

The variables in M that are not free are called *bound variables*.

(ii) If $\text{FV}(M) = \emptyset$, then we say that M is *closed* or that it is a *combinator*.

$$\Lambda^{\emptyset} \triangleq \{M \in \Lambda \mid M \text{ is closed}\}.$$

Well known combinators are $I \triangleq \lambda x.x$, $K \triangleq \lambda xy.y$, $S \triangleq \lambda xyz.xz(yz)$, $\Omega \triangleq (\lambda x.xx)(\lambda x.xx)$, and $Y \triangleq \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$. Officially $S \equiv (\lambda c(\lambda c'(\lambda c''((cc'')(c'c'')))))$, according to Definition 1A.1, so we see that the effort learning the notation 1A.2 pays.

⁴This is an abbreviation for the reference Barendregt [1984].

1. THE SIMPLY TYPED LAMBDA CALCULUS

1A.4. DEFINITION. On Λ the following equational theory $\lambda\beta\eta$ is defined by the usual equality axiom and rules (reflexivity, symmetry, transitivity, congruence), including congruence with respect to abstraction:

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

and the following special axiom(schemes)

$(\lambda x.M)N$	$= M[x := N]$	$(\beta\text{-rule})$
$\lambda x.Mx$	$= M,$ if $x \notin \text{FV}(M)$	$(\eta\text{-rule})$

FIGURE 2. The theory $\lambda\beta\eta$

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

1A.5. DEFINITION. On Λ we define the following notions of β -reduction and η -reduction

$(\lambda x.M)N$	$\rightarrow M[x := N]$	(β)
$\lambda x.Mx$	$\rightarrow M,$ if $x \notin \text{FV}(M)$	(η)

FIGURE 3. $\beta\eta$ -contraction rules

As usual, see B[1984], these notions of reduction generate the corresponding reduction relations $\rightarrow_\beta, \rightarrow_\eta, \rightarrow_{\beta\eta}, \rightarrow_{\eta\beta}, \rightarrow_{\beta\eta\beta}$ and $\rightarrow_{\eta\beta\eta}$. Also there are the corresponding conversion relations $=_\beta, =_\eta$ and $=_{\beta\eta}$. Terms in Λ will often be considered modulo $=_\beta$ or $=_{\beta\eta}$.

1A.6. NOTATION. If we write $M = N$, then we mean $M =_{\beta\eta} N$ by default, the *extensional* version of equality. This by contrast with B[1984], where the default was $=_\beta$.

1A.7. REMARK. Like in B[1984], Convention 2.1.12. we will not be concerned with α -conversion, renaming bound variables in order to avoid confusion between free and bound occurrences of variables. So we write $\lambda x.x \equiv \lambda y.y$. We do this by officially working on the α -equivalence classes; when dealing with a concrete term as representative of such a class the bound variables will be chosen maximally fresh: different from the free variables and from each other. See, however, Section 7D, in which we introduce α -conversion on recursive types and show how it can be avoided in a way that is more effective than for terms.

1A.8. PROPOSITION. *For all $M, N \in \Lambda$ one has*

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

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

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

1A.9. PROPOSITION (Church-Rosser theorem for $\lambda\beta$ and $\lambda\beta\eta$). *For the notions of reduction \rightarrow_β and $\rightarrow_{\beta\eta}$ one has the following.*

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

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

One also says that the reduction relations \rightarrow_R , for $R \in \{\beta, \beta\eta\}$ are confluent.

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

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

PROOF. See Theorems 3.2.8 and 3.3.9 in B[1984]. ■

1A.10. DEFINITION. (i) Let T be a set of equations between λ -terms. Write

$$T \vdash_{\lambda\beta\eta} M = N, \text{ or simply } T \vdash M = N$$

if $M = N$ is provable in $\lambda\beta\eta$ plus the additional equations in T added as axioms.

(ii) T is called *inconsistent* if T proves every equation, otherwise consistent.

(iii) The equation $P = Q$, with $P, Q \in \Lambda$, is called *inconsistent*, notation $P \# Q$, if $\{P = Q\}$ is inconsistent. Otherwise $P = Q$ is *consistent*.

The set $T = \emptyset$, i.e. the $\lambda\beta\eta$ -calculus itself, is consistent, as follows from the Church-Rosser theorem. Examples of inconsistent equations: $K \# I$ and $I \# S$. On the other hand $\Omega = I$ is consistent.

Simple types

Types in this part, also called *simple types*, are syntactic objects built from atomic types using the operator \rightarrow . 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 $A \rightarrow B$ 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 which terms fit together.

1A.11. DEFINITION. (i) Let \mathbb{A} be a non-empty set. An element of \mathbb{A} is called a *type atom*. The set of *simple types* over \mathbb{A} , notation $\mathbb{T} = \mathbb{T}^{\mathbb{A}}$, is inductively defined as follows.

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

We assume that no relations like $\alpha \rightarrow \beta = \gamma$ hold between type atoms: $\mathbb{T}^{\mathbb{A}}$ is freely generated. Often one finds $\mathbb{T} = \mathbb{T}^{\mathbb{A}}$ given by a simplified syntax.

$$\boxed{\mathbb{T} ::= \mathbb{A} \mid \mathbb{T} \rightarrow \mathbb{T}}$$

FIGURE 4. Simple types

(ii) Let $\mathbb{A}_0 = \{0\}$. Then we write $\mathbb{T}^0 \triangleq \mathbb{T}^{\mathbb{A}_0}$.

(iii) Let $\mathbb{A}_{\infty} = \{c, c', c'', \dots\}$. Then we write $\mathbb{T}^{\infty} \triangleq \mathbb{T}^{\mathbb{A}_{\infty}}$

We usually take $0 = c$. Then $\mathbb{T}^0 \subseteq \mathbb{T}^{\infty}$. If we write simply \mathbb{T} , then this refers to $\mathbb{T}^{\mathbb{A}}$ for an unspecified \mathbb{A} .

1A.12. NOTATION. (i) If $A_1, \dots, A_n \in \mathbb{T}$, then

$$A_1 \rightarrow \dots \rightarrow A_n \triangleq (A_1 \rightarrow (A_2 \rightarrow \dots \rightarrow (A_{n-1} \rightarrow A_n))).$$

That is, we use association to the right.

(ii) $\alpha, \beta, \gamma, \dots, \alpha_0, \beta_0, \gamma_0, \dots, \alpha', \beta', \gamma', \dots$ denote arbitrary elements of \mathbb{A} .

(iii) A, B, C, \dots denote arbitrary elements of \mathbb{T} .

1. THE SIMPLY TYPED LAMBDA CALCULUS

1A.13. DEFINITION (Type substitution). Let $A, C \in \mathbb{T}^{\mathbb{A}}$ and $\alpha \in \mathbb{A}$. The result of substituting C for the occurrences of α in A , notation $A[\alpha := C]$, is defined as follows.

$$\begin{aligned}\alpha[\alpha := C] &\triangleq C; \\ \beta[\alpha := C] &\triangleq \beta, && \text{if } \alpha \not\equiv \beta; \\ (A \rightarrow B)[\alpha := C] &\triangleq (A[\alpha := C]) \rightarrow (B[\alpha := C]).\end{aligned}$$

Assigning simple types

1A.14. DEFINITION ($\lambda_{\rightarrow}^{\text{Cu}}$). (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

$$\Gamma \vdash_{\lambda_{\rightarrow}}^{\text{Cu}} M : A$$

(or $\Gamma \vdash_{\lambda_{\rightarrow}} M : A$, 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.

$$\begin{aligned}(x:A) \in \Gamma &\Rightarrow \Gamma \vdash x : A; \\ \Gamma \vdash M : (A \rightarrow B), \quad \Gamma \vdash N : A &\Rightarrow \Gamma \vdash (MN) : B; \\ \Gamma, x:A \vdash M : B &\Rightarrow \Gamma \vdash (\lambda x.M) : (A \rightarrow B).\end{aligned}$$

In the last rule $\Gamma, x:A$ is required to be a basis.

These rules are usually written as follows.

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

FIGURE 5. The system $\lambda_{\rightarrow}^{\text{Cu}}$ à la Curry

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

1A.15. DEFINITION. Let $\Gamma = \{x_1:A_1, \dots, x_n:A_n\}$. Then

- (i) $\text{dom}(\Gamma) \triangleq \{x_1, \dots, x_n\}$, the *domain* of Γ .
- (ii) $x_1:A_1, \dots, x_n:A_n \vdash_{\lambda_{\rightarrow}} M : A$ denotes $\Gamma \vdash_{\lambda_{\rightarrow}} M : A$.

- (iii) In particular $\vdash_{\lambda_{\rightarrow}} M : A$ stands for $\emptyset \vdash_{\lambda_{\rightarrow}} M : A$.
(iv) $x_1, \dots, x_n : A \vdash_{\lambda_{\rightarrow}} M : B$ stands for $x_1 : A, \dots, x_n : A \vdash_{\lambda_{\rightarrow}} M : B$.

1A.16. EXAMPLE. (i) $\vdash_{\lambda_{\rightarrow}} I : A \rightarrow A$;

$$\vdash_{\lambda_{\rightarrow}} K : A \rightarrow B \rightarrow A;$$

$$\vdash_{\lambda_{\rightarrow}} S : (A \rightarrow B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow A \rightarrow C.$$

(ii) Also one has

$$\begin{array}{c} x : A \quad \vdash_{\lambda_{\rightarrow}} Ix : A; \\ x : A, y : B \quad \vdash_{\lambda_{\rightarrow}} Ky : A; \\ x : (A \rightarrow B \rightarrow C), y : (A \rightarrow B), z : A \quad \vdash_{\lambda_{\rightarrow}} Szxyz : C. \end{array}$$

(iii) The terms Y, Ω do not have a type. This is obvious after some trying. A systematic reason is that all typable terms have a nf, as we will see later, but these two do not have a nf.

(iv) The term $\omega \triangleq \lambda x. xx$ is in nf but does not have a type either.

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

Introduction rule	$\frac{x : A \quad \vdots \quad M : B}{\lambda x. M : (A \rightarrow B)}$
Elimination rule	$\frac{M : (A \rightarrow B) \quad N : A}{MN : B}$

$\lambda_{\rightarrow}^{\text{Cu}}$ alternative version

In this version the basis 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 $x : A$ and the ‘axioms’ in the basis. Striking through $x : A$ means that for the conclusion $\lambda x. M : A \rightarrow B$ the assumption $x : A$ is no longer needed; it is *discharged*.

1A.17. EXAMPLE. (i) $\vdash (\lambda xy. x) : (A \rightarrow B \rightarrow A)$ for all $A, B \in \mathbb{T}$.

We will use the notation of version 1 of $\lambda_{\rightarrow}^{\mathbb{A}}$ for a derivation of this statement.

$$\frac{x : A, y : B \vdash x : A}{\frac{x : A \vdash (\lambda y. x) : B \rightarrow A}{\vdash (\lambda x \lambda y. x) : A \rightarrow B \rightarrow A}}$$

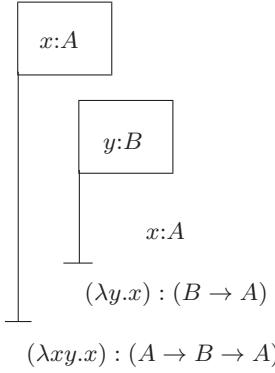
Note that $\lambda xy. x \equiv \lambda x \lambda y. x$ by definition.

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

$$\frac{\frac{\frac{x : A \quad y : B}{x : A}}{(\lambda y. x) : (B \rightarrow A)} 1}{(\lambda xy. x) : (A \rightarrow B \rightarrow A)} 2$$

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 $A \in \mathbb{T}$

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

(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 $\lambda_{\rightarrow}^{\mathbb{A}}$.

1A.18. DEFINITION. Let Γ be a basis and $A \in \mathbb{T} = \mathbb{T}^{\mathbb{A}}$. Then write

- (i) $\Lambda_{\rightarrow}^{\Gamma}(A) \triangleq \{M \in \Lambda \mid \Gamma \vdash_{\lambda_{\rightarrow}^{\mathbb{A}}} M : A\}$.
- (ii) $\Lambda_{\rightarrow}^{\Gamma} \triangleq \bigcup_{A \in \mathbb{T}} \Lambda_{\rightarrow}^{\Gamma}(A)$.
- (iii) $\Lambda_{\rightarrow}(A) \triangleq \bigcup_{\Gamma} \Lambda_{\rightarrow}^{\Gamma}(A)$.
- (iv) $\Lambda_{\rightarrow} \triangleq \bigcup_{A \in \mathbb{T}} \Lambda_{\rightarrow}(A)$.

(v) Emphasizing the dependency on \mathbb{A} we write $\Lambda_{\rightarrow}^{\mathbb{A}}(A)$ or $\Lambda_{\rightarrow}^{\mathbb{A}, \Gamma}(A)$, etcetera.

1A.19. DEFINITION. Let Γ be a basis, $A \in \mathbb{T}$ and $M \in \Lambda$. Then

(i) If $M \in \Lambda_{\rightarrow}^{\emptyset}(A)$, then we say that

M has type A or A is inhabited by M.

- (ii) If $M \in \Lambda_{\rightarrow}^{\emptyset}$, then M is called *typable*.
- (iii) If $M \in \Lambda_{\rightarrow}^{\Gamma}(A)$, then M *has type A relative to Γ* .
- (iv) If $M \in \Lambda_{\rightarrow}^{\Gamma}$, then M is called *typable relative to Γ* .
- (v) If $\Lambda_{\rightarrow}^{\Gamma}(A) \neq \emptyset$, then A is *inhabited relative to Γ* .

1A.20. EXAMPLE. We have

$$\begin{aligned} K &\in \Lambda_{\rightarrow}^{\emptyset}(A \rightarrow B \rightarrow A); \\ Kx &\in \Lambda_{\rightarrow}^{\{x:A\}}(B \rightarrow A). \end{aligned}$$

1A.21. DEFINITION. Let $A \in \mathbb{T}$.

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

$$\begin{aligned}\text{dpt}(\alpha) &\triangleq 1; \\ \text{dpt}(A \rightarrow B) &\triangleq \max\{\text{dpt}(A), \text{dpt}(B)\} + 1.\end{aligned}$$

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

$$\begin{aligned}\text{rk}(\alpha) &\triangleq 0; \\ \text{rk}(A \rightarrow B) &\triangleq \max\{\text{rk}(A) + 1, \text{rk}(B)\}.\end{aligned}$$

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

$$\begin{aligned}\text{ord}(\alpha) &\triangleq 1; \\ \text{ord}(A \rightarrow B) &\triangleq \max\{\text{ord}(A) + 1, \text{ord}(B)\}.\end{aligned}$$

(iv) The *depth* of a basis Γ is

$$\text{dpt}(\Gamma) \triangleq \max_i \{\text{dpt}(A_i) \mid (x_i : A_i) \in \Gamma\}.$$

Similarly we define $\text{rk}(\Gamma)$ and $\text{ord}(\Gamma)$. Note that $\text{ord}(A) = \text{rk}(A) + 1$.

The notion of ‘order’ comes from logic, where dealing with elements of type 0 is done in ‘first order’ predicate logic. 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 notion of ‘rank’ comes from computer science.

1A.22. DEFINITION. For $A \in \mathbb{T}$ we define $A^k \rightarrow B$ by recursion on k :

$$\begin{aligned}A^0 \rightarrow B &\triangleq B; \\ A^{k+1} \rightarrow B &\triangleq A \rightarrow A^k \rightarrow B.\end{aligned}$$

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

Several properties can be proved by induction on the depth of a type. This holds for example for Lemma 1A.25(i).

The asymmetry in the definition of rank is intended because the meaning of a type like $(0 \rightarrow 0) \rightarrow 0$ is more complex than that of $0 \rightarrow 0 \rightarrow 0$, as can be seen by looking to the inhabitants of these types: functionals with functions as arguments versus binary functions. Some authors use the name *type level* instead of ‘rank’.

The minimal and maximal systems λ_{\rightarrow}^0 and $\lambda_{\rightarrow}^{\infty}$

The collection \mathbb{A} of type variables serves as set of base types from which other types are constructed. We have $\mathbb{A}_0 = \{0\}$ with just one type atom and $\mathbb{A}_{\infty} = \{\alpha_0, \alpha_1, \alpha_2, \dots\}$ 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.

1A.23. DEFINITION. We define the following systems of type assignment.

- (i) $\lambda_{\rightarrow}^0 \triangleq \lambda_{\rightarrow}^{\mathbb{A}_0}$.
- (ii) $\lambda_{\rightarrow}^{\infty} \triangleq \lambda_{\rightarrow}^{\mathbb{A}_{\infty}}$.

Focusing on \mathbb{A}_0 or \mathbb{A}_∞ we write $\Lambda_\rightarrow^0(A) \triangleq \Lambda_\rightarrow^{\mathbb{A}_0}(A)$ or $\Lambda_\rightarrow^\infty(A) \triangleq \Lambda_\rightarrow^{\mathbb{A}_\infty}(A)$ respectively.

Many of the interesting features of the ‘larger’ $\lambda_\rightarrow^\infty$ are already present in the minimal version λ_\rightarrow^0 .

1A.24. DEFINITION. (i) The following types of $\mathbb{T}^0 \subseteq \mathbb{T}^\mathbb{A}$ are often used.

$$0 \triangleq c, 1 \triangleq 0 \rightarrow 0, 2 \triangleq (0 \rightarrow 0) \rightarrow 0, \dots.$$

In general

$$0 \triangleq c \text{ and } k+1 \triangleq k \rightarrow 0.$$

Note that $\text{rk}(n) = n$. That overloading of n as element of \mathbb{N} and as type will usually be disambiguated by stating ‘the type n ’ for the latter case.

(ii) Define n_k by cases on n .

$$\begin{aligned} 0_k &\triangleq 0; \\ (n+1)_k &\triangleq n^k \rightarrow 0. \end{aligned}$$

For example

$$\begin{aligned} 1_0 &\equiv 0; \\ 1_2 &\equiv 0 \rightarrow 0 \rightarrow 0; \\ 2_3 &\equiv 1 \rightarrow 1 \rightarrow 1 \rightarrow 0; \\ 1^2 \rightarrow 2 \rightarrow 0 &\equiv (0 \rightarrow 0) \rightarrow (0 \rightarrow 0) \rightarrow ((0 \rightarrow 0) \rightarrow 0) \rightarrow 0. \end{aligned}$$

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

The notation n_k is used only for $n \in \mathbb{N}$. In the following lemma the notation $A_1 \cdots A_a$ with subscripts denotes as usual a sequence of types.

1A.25. LEMMA. (i) Every type A of $\lambda_\rightarrow^\infty$ is of the form

$$A \equiv A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_a \rightarrow \alpha.$$

(ii) Every type A of λ_\rightarrow^0 is of the form

$$A \equiv A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_a \rightarrow 0.$$

$$(iii) \text{rk}(A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_a \rightarrow \alpha) = \max\{\text{rk}(A_i) + 1 \mid 1 \leq i \leq a\}.$$

PROOF. (i) By induction on the structure (depth) of A . If $A \equiv \alpha$, then this holds for $a = 0$. If $A \equiv B \rightarrow C$, then by the induction hypothesis one has

$C \equiv C_1 \rightarrow \cdots \rightarrow C_c \rightarrow \gamma$. Hence $A \equiv B \rightarrow C_1 \rightarrow \cdots \rightarrow C_c \rightarrow \gamma$.

(ii) Similar to (i).

(iii) By induction on a . ■

1A.26. NOTATION. Let $A \in \mathbb{T}^\mathbb{A}$ and suppose $A \equiv A_1 \rightarrow A_2 \rightarrow \cdots \rightarrow A_a \rightarrow \alpha$. Then the A_i are called the *components* of A . We write

$$\begin{aligned} \text{arity}(A) &\triangleq a, \\ A(i) &\triangleq A_i, \quad \text{for } 1 \leq i \leq a; \\ \text{target}(A) &\triangleq \alpha. \end{aligned}$$

Iterated components are denoted as follows

$$A(i, j) \triangleq A(i)(j).$$

1A.27. REMARK. We usually work with $\lambda_{\rightarrow}^{\mathbb{A}}$ for an unspecified \mathbb{A} , but will be more specific in some cases.

Different versions of $\lambda_{\rightarrow}^{\mathbb{A}}$

We will introduce several variants of $\lambda_{\rightarrow}^{\mathbb{A}}$.

The Curry version of $\lambda_{\rightarrow}^{\mathbb{A}}$

1A.28. DEFINITION. The system $\lambda_{\rightarrow}^{\mathbb{A}}$ that was introduced in Definition 1A.14 assigns types to untyped lambda terms. To be explicit it will be referred to as the *Curry version* and be denoted by $\lambda_{\rightarrow}^{\mathbb{A},\text{Cu}}$ or $\lambda_{\rightarrow}^{\text{Cu}}$, as the set \mathbb{A} often does not need to be specified.

The Curry version of $\lambda_{\rightarrow}^{\mathbb{A}}$ is called *implicitly typed* because an expression like

$$\lambda x.xK$$

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 $\lambda_{\rightarrow}^{\mathbb{A}}$ 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), see [Girard, Lafont, and Taylor \[1989\]](#), [Barendregt \[1992\]](#) or [Sørensen and Urzyczyn \[2006\]](#) for a description of that system and [Wells \[1999\]](#) for the undecidability.

The Church version $\lambda_{\rightarrow}^{\text{Ch}}$ of $\lambda_{\rightarrow}^{\mathbb{A}}$

The first variant of $\lambda_{\rightarrow}^{\text{Cu}}$ is the *Church version* of $\lambda_{\rightarrow}^{\mathbb{A}}$, denoted by $\lambda_{\rightarrow}^{\mathbb{A},\text{Ch}}$ or $\lambda_{\rightarrow}^{\text{Ch}}$. 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

$$\begin{array}{ll} \vdash_{\lambda_{\rightarrow}}^{\text{Cu}} (\lambda x.x) : A \rightarrow A & (1_{\text{Cu}}) \\ y:A \quad \vdash_{\lambda_{\rightarrow}}^{\text{Cu}} (\lambda x.xy) : (A \rightarrow B) \rightarrow B & (2_{\text{Cu}}) \end{array}$$

now become

$$(\lambda x^A.x^A) \in \Lambda_{\rightarrow}^{\text{Ch}}(A \rightarrow A) \quad (1_{\text{Ch}})$$

$$(\lambda x^{A \rightarrow B}.x^{A \rightarrow B}y^A) \in \Lambda_{\rightarrow}^{\text{Ch}}((A \rightarrow B) \rightarrow B) \quad (2_{\text{Ch}})$$

1A.29. DEFINITION. Let \mathbb{A} be a set of type atoms. The Church version of $\lambda_{\rightarrow}^{\mathbb{A}}$, notation $\lambda_{\rightarrow}^{\mathbb{A},\text{Ch}}$ or $\lambda_{\rightarrow}^{\text{Ch}}$ if \mathbb{A} is not emphasized, is defined as follows. The system has the same set of types $\mathbb{T}^{\mathbb{A}}$ as $\lambda_{\rightarrow}^{\mathbb{A},\text{Cu}}$.

(i) The set of term variables is different: each such variable is coupled with a unique type. This in such a way that every type has infinitely many variables coupled to it. So we take

$$V^{\mathbb{T}} \triangleq \{x^{t(x)} \mid x \in V\},$$

where $t : V \rightarrow \mathbb{T}^{\mathbb{A}}$ is a fixed map such that $t^{-1}(A)$ is infinite for all $A \in \mathbb{T}^{\mathbb{A}}$. So we have

$$\begin{aligned} \{x^A, y^A, z^A, \dots\} \subseteq V^{\mathbb{T}} &\text{ is infinite for all } A \in \mathbb{T}^{\mathbb{A}}; \\ x^A, x^B \in V^{\mathbb{T}} &\Rightarrow A \equiv B, \text{ for all } A, B \in \mathbb{T}^{\mathbb{A}}. \end{aligned}$$

(ii) The set of *terms of type A*, notation $\Lambda_{\rightarrow}^{\text{Ch}}(A)$, is defined as follows.

$\begin{array}{c} x^A \in \Lambda_{\rightarrow}^{\text{Ch}}(A); \\ M \in \Lambda_{\rightarrow}^{\text{Ch}}(A \rightarrow B), N \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \Rightarrow (MN) \in \Lambda_{\rightarrow}^{\text{Ch}}(B); \\ M \in \Lambda_{\rightarrow}^{\text{Ch}}(B) \Rightarrow (\lambda x^A.M) \in \Lambda_{\rightarrow}^{\text{Ch}}(A \rightarrow B). \end{array}$

FIGURE 6. The system $\lambda_{\rightarrow}^{\text{Ch}}$ of typed terms *á la* Church

(iii) The set of *terms of $\lambda_{\rightarrow}^{\text{Ch}}$* , notation $\Lambda_{\rightarrow}^{\text{Ch}}$, is defined as

$$\Lambda_{\rightarrow}^{\text{Ch}} \triangleq \bigcup_{A \in \mathbb{T}} \Lambda_{\rightarrow}^{\text{Ch}}(A).$$

For example

$$\begin{aligned} y^{B \rightarrow A} x^B &\in \Lambda_{\rightarrow}^{\text{Ch}}(A); \\ \lambda x^A.y^{B \rightarrow A} &\in \Lambda_{\rightarrow}^{\text{Ch}}(A \rightarrow B \rightarrow A); \\ \lambda x^A.x^A &\in \Lambda_{\rightarrow}^{\text{Ch}}(A \rightarrow A). \end{aligned}$$

1A.30. DEFINITION. On $\Lambda_{\rightarrow}^{\text{Ch}}$ we define the following notions of reduction.

$\begin{array}{lcl} (\lambda x^A.M)N &\rightarrow& M[x^A := N] \\ \lambda x^A.Mx^A &\rightarrow& M, \end{array}$	$\begin{array}{ll} &\text{if } x^A \notin \text{FV}(M) \\ &\text{(}\beta\text{)} \\ &\text{(}\eta\text{)} \end{array}$
--	--

FIGURE 7. $\beta\eta$ -contraction rules for $\lambda_{\rightarrow}^{\text{Ch}}$

It will be shown in Proposition 1B.10 that $\Lambda_{\rightarrow}^{\text{Ch}}(A)$ is closed under $\beta\eta$ -reduction; i.e. this reduction preserves the type of a typed term.

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

1A.31. DEFINITION (Type substitution). For $M \in \Lambda_{\rightarrow}^{\text{Ch}}$, $\alpha \in \mathbb{A}$, and $B \in \mathbb{T}^{\mathbb{A}}$ we define the result of substituting B for α in M , notation $M[\alpha := B]$, inductively as follows.

M	$M[\alpha := B]$
x^A	$x^{A[\alpha:=B]}$
PQ	$(P[\alpha := B])(Q[\alpha := B])$
$\lambda x^A.P$	$\lambda x^{A[\alpha:=B]}.P[\alpha := B]$

1A.32. NOTATION. A term like $(\lambda f^1 x^0.f^1(f^1 x^0)) \in \Lambda_{\rightarrow}^{\text{Ch}}(1 \rightarrow 0 \rightarrow 0)$ will also be written as

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

just indicating the types of the bound variables. This notation is analogous to the one in the de Bruijn version of $\lambda_{\rightarrow}^{\mathbb{A}}$ that follows. Sometimes we will even write $\lambda f x.f(fx)$. We will come back to this notational issue in section 1B.

The de Bruijn version $\lambda_{\rightarrow}^{\text{dB}}$ of $\lambda_{\rightarrow}^{\mathbb{A}}$

There is the following disadvantage about the Church systems. Consider

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

In the next volume we will consider dependent types coming from the Automath language family, see [Nederpelt, Geuvers, and de Vrijer \[1994\]](#), designed for formalizing arguments and proof-checking⁵. 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 $P : A \rightarrow \mathbb{T}$. In this way types may contain redexes and we may have the following reduction

$$\mathsf{I} \equiv (\lambda x^A.x^A) \rightarrow_{\beta} (\lambda x^{A'}.x^A),$$

in case $A \rightarrow_{\beta} A'$, by reducing only the first A to A' . The question now is whether $\lambda x^{A'}$ binds the x^A . If we write I as

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

then this problem disappears

$$\lambda x:A.x \rightarrow \lambda x:A'.x.$$

As the second occurrence of x is implicitly typed with the same type as the first, the intended meaning is correct. In the following system $\lambda_{\rightarrow}^{\mathbb{A}, \text{dB}}$ this idea is formalized.

1A.33. DEFINITION. The second variant of $\lambda_{\rightarrow}^{\text{Cu}}$ is the de *Bruijn version* of $\lambda_{\rightarrow}^{\mathbb{A}}$, denoted by $\lambda_{\rightarrow}^{\mathbb{A}, \text{dB}}$ or $\lambda_{\rightarrow}^{\text{dB}}$. Now only bound variables get ornamented with types, but only at the binding stage. The examples (1_{Cu}), (2_{Cu}) now become

$$\begin{aligned} \vdash_{\lambda_{\rightarrow}}^{\text{dB}} & (\lambda x:A.x) : A \rightarrow A & (1_{\text{dB}}) \\ y:A \quad \vdash_{\lambda_{\rightarrow}}^{\text{dB}} & (\lambda x:(A \rightarrow B).xy) : (A \rightarrow B) \rightarrow B & (2_{\text{dB}}) \end{aligned}$$

1A.34. DEFINITION. The system $\lambda_{\rightarrow}^{\text{dB}}$ starts with a collection of *pseudo-terms*, notation $\Lambda_{\rightarrow}^{\text{dB}}$, defined by the following simplified syntax.

$$\boxed{\Lambda_{\rightarrow}^{\text{dB}} ::= \mathsf{V} \mid \Lambda_{\rightarrow}^{\text{dB}} \Lambda_{\rightarrow}^{\text{dB}} \mid \lambda \mathsf{V} : \mathbb{T}. \Lambda_{\rightarrow}^{\text{dB}}}$$

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 typable, term in $\lambda_{\rightarrow}^{\mathbb{A}, \text{dB}}$, whereas the second one is not.

1A.35. DEFINITION. (i) A **basis** Γ consists 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 $\lambda_{\rightarrow}^{\mathbb{A}, \text{Cu}}$.

(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.

⁵The proof-assistant [Coq](#), see the URL coq.inria.fr and [Bertot and Castéran \[2004\]](#), is a modern version of Automath in which one uses for formal proofs typed lambda terms in the de Bruijn style.

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

FIGURE 8. The system $\lambda_{\rightarrow}^{\text{dB}}$ à la de Bruijn

Provability in $\lambda_{\rightarrow}^{\text{dB}}$ is denoted by $\vdash_{\lambda_{\rightarrow}}^{\text{dB}}$. Thus the legal terms of $\lambda_{\rightarrow}^{\text{dB}}$ are defined by making a selection from the context-free language $\Lambda_{\rightarrow}^{\text{dB}}$. That $\lambda x:\alpha.x$ is legal follows from $x:\alpha \vdash_{\lambda_{\rightarrow}}^{\text{dB}} x : \alpha$ using the \rightarrow -introduction rule. That $(\lambda x:\alpha.x)(\lambda y:\beta.y)$ is not legal follows from Proposition 1B.12. These legal terms do not form a context-free language, do exercise 1E.7. For closed terms the Church and the de Bruijn notation are isomorphic.

1B. First properties and comparisons

In this section we will present simple properties of the systems $\lambda_{\rightarrow}^{\text{A}}$. Deeper properties, like normalization of typable terms, will be considered in Sections 2A, 2B.

Properties of $\lambda_{\rightarrow}^{\text{Cu}}$

We start with properties of the system $\lambda_{\rightarrow}^{\text{Cu}}$.

1B.1. PROPOSITION (Weakening lemma for $\lambda_{\rightarrow}^{\text{Cu}}$).

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 $\Gamma \vdash M : A$. ■

1B.2. LEMMA (Free variable lemma for $\lambda_{\rightarrow}^{\text{Cu}}$). For a set \mathcal{X} of variables write

$$\Gamma \upharpoonright \mathcal{X} = \{x:A \in \Gamma \mid x \in \mathcal{X}\}.$$

(i) Suppose $\Gamma \vdash M : A$. Then $\text{FV}(M) \subseteq \text{dom}(\Gamma)$.

(ii) If $\Gamma \vdash M : A$, then $\Gamma \upharpoonright \text{FV}(M) \vdash M : A$.

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

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

1B.3. PROPOSITION (Inversion Lemma for $\lambda_{\rightarrow}^{\text{Cu}}$).

- (i) $\Gamma \vdash x : A \Rightarrow (x:A) \in \Gamma.$
- (ii) $\Gamma \vdash MN : A \Rightarrow \exists B \in \mathbb{T} [\Gamma \vdash M : B \rightarrow A \& \Gamma \vdash N : B].$
- (iii) $\Gamma \vdash \lambda x.M : A \Rightarrow \exists B, C \in \mathbb{T} [A \equiv B \rightarrow C \& \Gamma, x:B \vdash M : C].$

PROOF. (i) Suppose $\Gamma \vdash x : A$ holds in λ_{\rightarrow} . 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. ■

1B.4. COROLLARY. Let $\Gamma \vdash_{\lambda}^{Cu} xN_1 \cdots N_k : B$. Then there exist unique $A_1, \dots, A_k \in \mathbb{T}$ such that

$$\Gamma \vdash_{\lambda}^{Cu} N_i : A_i, 1 \leq i \leq k, \text{ and } x:(A_1 \rightarrow \cdots \rightarrow A_k \rightarrow B) \in \Gamma.$$

PROOF. By applying k -times (ii) and then (i) of the proposition. ■

1B.5. PROPOSITION (Substitution lemma for λ^{Cu}).

- (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. (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 $y \neq x$ 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 $\Gamma, x:A \vdash F : C \rightarrow B$, $\Gamma, x:A \vdash G : C$ and $FG \equiv M$. By the induction hypothesis one has $\Gamma \vdash F^* : C \rightarrow B$ and $\Gamma \vdash G^* : C$. Hence $\Gamma \vdash (FG)^* \equiv F^*G^* : B$.

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

(ii) Similarly. ■

1B.6. PROPOSITION (Subject reduction property for λ^{Cu}).

$$\Gamma \vdash M : A \& M \rightarrow_{\beta\eta} N \Rightarrow \Gamma \vdash N : A.$$

PROOF. It suffices to show this for a one-step $\beta\eta$ -reduction, denoted by \rightarrow . Suppose $\Gamma \vdash M : A$ and $M \rightarrow_{\beta\eta} N$ in order to show that $\Gamma \vdash N : A$. We do this 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 $M \rightarrow N$. Hence this case cannot occur.

Case 2. $\Gamma \vdash M : A$ is $\Gamma \vdash FP : A$ and is a direct consequence of $\Gamma \vdash F : B \rightarrow A$ and $\Gamma \vdash P : B$. Since $FP \equiv M \rightarrow N$ we can have three subcases.

Subcase 2.1. $N \equiv F'P$ with $F \rightarrow F'$.

Subcase 2.2. $N \equiv FP'$ with $P \rightarrow P'$.

In these two subcases it follows that $\Gamma \vdash N : A$, by using twice the IH.

Subcase 2.3. $F \equiv \lambda x.G$ and $N \equiv G[x := P]$. Since

$$\Gamma \vdash \lambda x.G : B \rightarrow A \& \Gamma \vdash P : B,$$

it follows by the inversion Lemma 1B.3 for λ that

$$\Gamma, x \vdash G : A \& \Gamma \vdash P : B.$$

Therefore by the substitution Lemma 1B.5 for λ it follows that

$$\Gamma \vdash G[x := P] : A, \text{ i.e. } \Gamma \vdash N : A.$$

Case 3. $\Gamma \vdash M : A$ is $\Gamma \vdash \lambda x.P : B \rightarrow C$ and follows from $\Gamma, x \vdash P : C$.

Subcase 3.1. $N \equiv \lambda x.P'$ with $P \rightarrow P'$. One has $\Gamma, x:B \vdash P' : C$ by the induction hypothesis, hence $\Gamma \vdash (\lambda x.P') : (B \rightarrow C)$, i.e. $\Gamma \vdash N : A$.

Subcase 3.2. $P \equiv Nx$ and $x \notin \text{FV}(N)$. Now $\Gamma, x:B \vdash Nx : C$ follows by Lemma 1B.3(ii) from $\Gamma, x:B \vdash N : (B' \rightarrow C)$ and $\Gamma, x:B \vdash x : B'$, for some B' . Then $B = B'$, by Lemma 1B.3(i), hence by Lemma 1B.2(ii) we have $\Gamma \vdash N : (B \rightarrow C) = A$. ■

The following result also holds for $\lambda_{\rightarrow}^{\text{Ch}}$ and $\lambda_{\rightarrow}^{\text{dB}}$, see Proposition 1B.28 and Exercise 2E.4.

1B.7. COROLLARY (Church-Rosser theorem for $\lambda_{\rightarrow}^{\text{Cu}}$). *On typable terms of $\lambda_{\rightarrow}^{\text{Cu}}$ the Church-Rosser theorem holds for the notions of reduction \rightarrow_{β} and $\rightarrow_{\beta\eta}$.*

(i) *Let $M, N_1, N_2 \in \Lambda_{\rightarrow}^{\Gamma}(A)$. Then*

$$M \rightarrow_{\beta(\eta)} N_1 \& M \rightarrow_{\beta(\eta)} N_2 \Rightarrow \exists Z \in \Lambda_{\rightarrow}^{\Gamma}(A). N_1 \rightarrow_{\beta(\eta)} Z \& N_2 \rightarrow_{\beta(\eta)} Z.$$

(ii) *Let $M, N \in \Lambda_{\rightarrow}^{\Gamma}(A)$. Then*

$$M =_{\beta(\eta)} N \Rightarrow \exists Z \in \Lambda_{\rightarrow}^{\Gamma}(A). M \rightarrow_{\beta(\eta)} Z \& N \rightarrow_{\beta(\eta)} Z.$$

PROOF. By the Church-Rosser theorems for \rightarrow_{β} and $\rightarrow_{\beta\eta}$ on untyped terms, Theorem 1A.9, and Proposition 1B.6. ■

Properties of $\lambda_{\rightarrow}^{\text{Ch}}$

Not all the properties of $\lambda_{\rightarrow}^{\text{Cu}}$ are meaningful for $\lambda_{\rightarrow}^{\text{Ch}}$. Those that are have to be reformulated slightly.

1B.8. PROPOSITION (Inversion Lemma for $\lambda_{\rightarrow}^{\text{Ch}}$).

- (i) $x^B \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \Rightarrow B = A.$
- (ii) $(MN) \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \Rightarrow \exists B \in \mathbb{T}. [M \in \Lambda_{\rightarrow}^{\text{Ch}}(B \rightarrow A) \& N \in \Lambda_{\rightarrow}^{\text{Ch}}(B)].$
- (iii) $(\lambda x^B. M) \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \Rightarrow \exists C \in \mathbb{T}. [A = (B \rightarrow C) \& M \in \Lambda_{\rightarrow}^{\text{Ch}}(C)].$

PROOF. As before. ■

Substitution of a term $N \in \Lambda_{\rightarrow}^{\text{Ch}}(B)$ for a typed variable x^B is defined as usual. We show that the resulting term keeps its type.

1B.9. PROPOSITION (Substitution lemma for $\lambda_{\rightarrow}^{\text{Ch}}$). *Let $A, B \in \mathbb{T}$. Then*

- (i) $M \in \Lambda_{\rightarrow}^{\text{Ch}}(A), N \in \Lambda_{\rightarrow}^{\text{Ch}}(B) \Rightarrow (M[x^B := N]) \in \Lambda_{\rightarrow}^{\text{Ch}}(A).$
- (ii) $M \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \Rightarrow M[\alpha := B] \in \Lambda_{\rightarrow}^{\text{Ch}}(A[\alpha := B]).$

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

1B.10. PROPOSITION (Closure under reduction for $\lambda_{\rightarrow}^{\text{Ch}}$). *Let $A \in \mathbb{T}$. Then*

- (i) $M \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \& M \rightarrow_{\beta} N \Rightarrow N \in \Lambda_{\rightarrow}^{\text{Ch}}(A).$
- (ii) $M \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \& M \rightarrow_{\eta} N \Rightarrow N \in \Lambda_{\rightarrow}^{\text{Ch}}(A).$
- (iii) $M \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \text{ and } M \rightarrow_{\beta\eta} N. \text{ Then } N \in \Lambda_{\rightarrow}^{\text{Ch}}(A).$

PROOF. (i) Suppose $M \equiv (\lambda x^B. P)Q \in \Lambda_{\rightarrow}^{\text{Ch}}(A)$. Then by Proposition 1B.8(ii) one has $\lambda x^B. P \in \Lambda_{\rightarrow}^{\text{Ch}}(B' \rightarrow A)$ and $Q \in \Lambda_{\rightarrow}^{\text{Ch}}(B')$. Then $B = B'$, and $P \in \Lambda_{\rightarrow}^{\text{Ch}}(A)$, by Proposition 1B.8(iii). Therefore $N \equiv P[x^B := Q] \in \Lambda_{\rightarrow}^{\text{Ch}}(A)$, by Proposition 1B.9.

(ii) Suppose $M \equiv (\lambda x^B. Nx^B) \in \Lambda_{\rightarrow}^{\text{Ch}}(A)$. Then $A = B \rightarrow C$ and $Nx^B \in \Lambda_{\rightarrow}^{\text{Ch}}(C)$, by Proposition 1B.8(iii). But then $N \in \Lambda_{\rightarrow}^{\text{Ch}}(B \rightarrow C)$ by Proposition 1B.8(i) and (ii).

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

The Church-Rosser theorem holds for $\beta\eta$ -reduction on $\Lambda_{\rightarrow}^{\text{Ch}}$. The proof is postponed until Proposition 1B.28.

PROPOSITION [Church-Rosser theorem for $\lambda_{\rightarrow}^{\text{Ch}}$] On typable terms of $\lambda_{\rightarrow}^{\text{Ch}}$ the CR property holds for the notions of reduction \rightarrow_{β} and $\rightarrow_{\beta\eta}$.

(i) Let $M, N_1, N_2 \in \Lambda_{\rightarrow}^{\text{Ch}}(A)$. Then

$$M \rightarrow_{\beta(\eta)} N_1 \& M \rightarrow_{\beta(\eta)} N_2 \Rightarrow \exists Z \in \Lambda_{\rightarrow}^{\text{Ch}}(A). N_1 \rightarrow_{\beta(\eta)} Z \& N_2 \rightarrow_{\beta(\eta)} Z.$$

(ii) Let $M, N \in \Lambda_{\rightarrow}^{\text{Ch}}(A)$. Then

$$M =_{\beta(\eta)} N \Rightarrow \exists Z \in \Lambda_{\rightarrow}^{\text{Ch}}(A). M \rightarrow_{\beta(\eta)} Z \& N \rightarrow_{\beta(\eta)} Z.$$

The following property called uniqueness of types does not hold for $\lambda_{\rightarrow}^{\text{Cu}}$. It is instructive to find out where the proof breaks down for that system.

1B.11. PROPOSITION (Unicity of types for $\lambda_{\rightarrow}^{\text{Ch}}$). *Let $A, B \in \mathbb{T}$. Then*

$$M \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \& M \in \Lambda_{\rightarrow}^{\text{Ch}}(B) \Rightarrow A = B.$$

PROOF. By induction on the structure of M , using the inversion lemma 1B.8. ■

Properties of $\lambda_{\rightarrow}^{\text{dB}}$

We mention the first properties of $\lambda_{\rightarrow}^{\text{dB}}$, the proofs being similar to those for $\lambda_{\rightarrow}^{\text{Ch}}$.

1B.12. PROPOSITION (Inversion Lemma for $\lambda_{\rightarrow}^{\text{dB}}$).

- (i) $\Gamma \vdash x : A \Rightarrow (x:A) \in \Gamma$.
- (ii) $\Gamma \vdash MN : A \Rightarrow \exists B \in \mathbb{T} [\Gamma \vdash M : B \rightarrow A \& \Gamma \vdash N : B]$.
- (iii) $\Gamma \vdash \lambda x:B.M : A \Rightarrow \exists C \in \mathbb{T} [A \equiv B \rightarrow C \& \Gamma, x:B \vdash M : C]$.

1B.13. PROPOSITION (Substitution lemma for $\lambda_{\rightarrow}^{\text{dB}}$).

- (i) $\Gamma, x:A \vdash M : B \& \Gamma \vdash N : A \Rightarrow \Gamma \vdash M[x := N] : B$.
- (ii) $\Gamma \vdash M : A \Rightarrow \Gamma[\alpha := B] \vdash M : A[\alpha := B]$.

1B.14. PROPOSITION (Subject reduction property for $\lambda_{\rightarrow}^{\text{dB}}$).

$$\Gamma \vdash M : A \& M \rightarrow_{\beta} N \Rightarrow \Gamma \vdash N : A.$$

1B.15. PROPOSITION (Church-Rosser theorem for $\lambda_{\rightarrow}^{\text{dB}}$). $\lambda_{\rightarrow}^{\text{dB}}$ satisfies CR.

(i) Let $M, N_1, N_2 \in \Lambda_{\rightarrow}^{\text{dB}, \Gamma}(A)$. Then

$$M \rightarrow_{\beta(\eta)} N_1 \& M \rightarrow_{\beta(\eta)} N_2 \Rightarrow \exists Z \in \Lambda_{\rightarrow}^{\text{dB}, \Gamma}(A). N_1 \rightarrow_{\beta(\eta)} Z \& N_2 \rightarrow_{\beta(\eta)} Z.$$

(ii) Let $M, N \in \Lambda_{\rightarrow}^{\text{dB}, \Gamma}(A)$. Then

$$M =_{\beta(\eta)} N \Rightarrow \exists Z \in \Lambda_{\rightarrow}^{\text{dB}, \Gamma}(A). M \rightarrow_{\beta(\eta)} Z \& N \rightarrow_{\beta(\eta)} Z.$$

PROOF. Do Exercise 2E.4. ■

It is instructive to see why the following result fails if the two contexts are different.

1B.16. PROPOSITION (Unicity of types for $\lambda_{\rightarrow}^{\text{dB}}$). *Let $A, B \in \mathbb{T}$. Then*

$$\Gamma \vdash M : A \& \Gamma \vdash M : B \Rightarrow A = B.$$

Equivalence of the systems

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

The Curry version inspired some implicitly typed programming languages like ML, Miranda, Haskell and Clean. Types are being derived. Since implicit typing makes programming easier, we want to consider this system.

The use of explicit typing becomes essential for extensions of $\lambda_{\rightarrow}^{\text{Cu}}$. For example in the system $\lambda 2$, also called system F , 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 $\lambda_{\rightarrow}^{\text{Ch}}$ and $\lambda_{\rightarrow}^{\text{dB}}$ are basically isomorphic as shown above. These systems have a very canonical semantics if the version $\lambda_{\rightarrow}^{\text{Ch}}$ is used.

We want two versions because the version $\lambda_{\rightarrow}^{\text{dB}}$ 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, see e.g. [Barendregt \[1992\]](#)) 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 λ_{\rightarrow} gives problems. For example, if $A \rightarrow B$, does one have that $\lambda x^A.x^A \rightarrow \lambda x^A.x^B$? 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 $\lambda_{\rightarrow}^{\text{Ch}}$, these problems do not arise. The possibility that types reduce is so important, that for explicitly typed extensions of λ_{\rightarrow} one needs to use the dB-versions.

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.

$$\begin{aligned}\lambda x.xy &\in \Lambda_{\rightarrow}^{\text{Cu},\{y:0\}}((0 \rightarrow 0) \rightarrow 0) & (\text{Curry}); \\ \lambda x:(0 \rightarrow 0).xy &\in \Lambda_{\rightarrow}^{\text{dB},\{y:0\}}((0 \rightarrow 0) \rightarrow 0) & (\text{de Bruijn}); \\ \lambda x^{0 \rightarrow 0}.x^{0 \rightarrow 0}y^0 &\in \Lambda_{\rightarrow}^{\text{Ch}}((0 \rightarrow 0) \rightarrow 0) & (\text{Church}).\end{aligned}$$

Hence for good reasons one finds all the three versions of λ_{\rightarrow} 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 $(0 \rightarrow 0) \rightarrow 0 \rightarrow 0$, but also $(A \rightarrow A) \rightarrow A \rightarrow A$ for any type A . Also there is a simple algorithm to find all possible types for an untyped lambda term, see Section 2C.

We are interested in typable terms M , among the untyped lambda terms Λ , using 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 M not containing a K-redex (of the form $(\lambda x.M)N$ with $x \notin \text{FV}(M)$). If M does contain a K-redex, then the type of the β -nf M^{nf} of M is still uniquely determined by A . For example the Church typing of $M \equiv \text{K}ly$ of type $\alpha \rightarrow \alpha$ is $(\lambda x^{\alpha \rightarrow \alpha}y^{\beta}.x^{\alpha \rightarrow \alpha})(\lambda z^{\alpha}.z^{\alpha})y^{\beta}$. The type β is not determined. But for the β -nf of M , the term I , the Church typing can only be $\text{I}_{\alpha} \equiv \lambda z^{\alpha}.z^{\alpha}$. See Exercise 2E.3.

If a type is not explicitly given, then possible types for M can be obtained schematically from groundtypes. By this we mean that e.g. the term $\mathbf{I} \equiv \lambda x.x$ has a Church version $\lambda x^\alpha.x^\alpha$ and type $\alpha \rightarrow \alpha$, where one can substitute any $A \in \mathbb{T}^{\mathbb{A}}$ for α . We will study this in greater detail in Section 2C.

Comparing $\lambda_{\rightarrow}^{\text{Cu}}$ and $\lambda_{\rightarrow}^{\text{Ch}}$

There are canonical translations between $\lambda_{\rightarrow}^{\text{Ch}}$ and $\lambda_{\rightarrow}^{\text{Cu}}$.

1B.17. DEFINITION. There is a *forgetful map* $|\cdot| : \Lambda_{\rightarrow}^{\text{Ch}} \rightarrow \Lambda$ defined as follows:

$$\begin{aligned} |x^A| &\triangleq x; \\ |MN| &\triangleq |M||N|; \\ |\lambda x:A.M| &\triangleq \lambda x.|M|. \end{aligned}$$

The map $|\cdot|$ just erases all type ornamentations of a term in $\Lambda_{\rightarrow}^{\text{Ch}}$. The following result states that terms in the Church version ‘project’ to legal terms in the Curry version of $\lambda_{\rightarrow}^{\text{A}}$. Conversely, legal terms in $\lambda_{\rightarrow}^{\text{Cu}}$ can be ‘lifted’ to terms in $\lambda_{\rightarrow}^{\text{Ch}}$.

1B.18. DEFINITION. Let $M \in \Lambda_{\rightarrow}^{\text{Ch}}$. Then we write

$$\Gamma_M \triangleq \{x:A \mid x^A \in \text{FV}(M)\}.$$

1B.19. PROPOSITION. (i) Let $M \in \Lambda_{\rightarrow}^{\text{Ch}}$. Then

$$M \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \Rightarrow \Gamma_M \vdash_{\lambda_{\rightarrow}}^{\text{Cu}} |M| : A,$$

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

$$\Gamma \vdash_{\lambda_{\rightarrow}}^{\text{Cu}} M : A \Leftrightarrow \exists M' \in \Lambda_{\rightarrow}^{\text{Ch}}(A). |M'| \equiv M.$$

PROOF. (i) By induction on the generation of $\Lambda_{\rightarrow}^{\text{Ch}}$. Since variables have a unique type Γ_M is well-defined and $\Gamma_P \cup \Gamma_Q = \Gamma_{PQ}$.

(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 1B.19(i) is not true: one has

$$\vdash_{\lambda_{\rightarrow}}^{\text{Cu}} |\lambda x^A.x^A| \equiv (\lambda x.x) : (A \rightarrow B) \rightarrow (A \rightarrow B),$$

but $(\lambda x^A.x^A) \notin \Lambda^{\text{Ch}}((A \rightarrow B) \rightarrow (A \rightarrow B))$.

1B.20. COROLLARY. In particular, for a type $A \in \mathbb{T}$ one has

$$A \text{ is inhabited in } \lambda_{\rightarrow}^{\text{Cu}} \Leftrightarrow A \text{ is inhabited in } \lambda_{\rightarrow}^{\text{Ch}}.$$

PROOF. Immediate. ■

For normal terms one can do better than Proposition 1B.19. First a structural result.

1B.21. PROPOSITION. Let $M \in \Lambda$ be in nf. Then $M \equiv \lambda x_1 \cdots x_n.y M_1 \cdots M_m$, 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], Corollary 8.3.8 for some details if necessary. ■

In order to prove results about the set NF of β -nfs, it is useful to introduce the subset vNF of β -nfs not starting with a λ , but with a free variable. These two sets can be defined by a simultaneous recursion known from context-free languages.

1B.22. DEFINITION. The sets **vNF** and **NF** of Λ are defined by the following grammar.

$$\boxed{\begin{array}{l} \text{vNF} ::= x \mid \text{vNF NF} \\ \text{NF} ::= \text{vNF} \mid \lambda x. \text{NF} \end{array}}$$

1B.23. PROPOSITION. For $M \in \Lambda$ one has

$$M \text{ is in } \beta\text{-nf} \Leftrightarrow M \in \text{NF}.$$

PROOF. By simultaneous induction it follows easily that

$$\begin{aligned} M \in \text{vNF} &\Rightarrow M \equiv x\vec{N} \text{ & } M \text{ is in } \beta\text{-nf;} \\ M \in \text{NF} &\Rightarrow M \text{ is in } \beta\text{-nf.} \end{aligned}$$

Conversely, for M in β -nf by Proposition 1B.21 one has $M \equiv \lambda\vec{x}.yN_1 \cdots N_k$, with the N_i all in β -nf. It follows by induction on the structure of such M that $M \in \text{NF}$. ■

1B.24. PROPOSITION. Assume that $M \in \Lambda$ is in β -nf. Then $\Gamma \vdash_{\lambda\rightarrow}^{\text{Cu}} M : A$ implies that there is a unique $M^{A;\Gamma} \in \Lambda_{\rightarrow}^{\text{Ch}}(A)$ such that $|M^{A;\Gamma}| \equiv M$ and $\Gamma_{M^{A;\Gamma}} \subseteq \Gamma$.

PROOF. By induction on the generation of nfs given in Definition 1B.22.

Case $M \equiv x\vec{N}$, with N_i in β -nf. By Proposition 1B.4 one has $(x:A_1 \rightarrow \cdots \rightarrow A_k \rightarrow A) \in \Gamma$ and $\Gamma \vdash_{\lambda\rightarrow}^{\text{Cu}} N_i : A_i$. As $\Gamma_{M^{A;\Gamma}} \subseteq \Gamma$, we must have $x^{A_1 \rightarrow \cdots \rightarrow A_k \rightarrow A} \in \text{FV}(M^{A;\Gamma})$. By the IH there are unique $N_i^{A_i,\Gamma}$ for the N_i . Then $M^{A;\Gamma} \equiv x^{A_1 \rightarrow \cdots \rightarrow A_k \rightarrow A} N_1^{A_1,\Gamma} \cdots N_k^{A_k,\Gamma}$ is the unique way to type M .

Case $M \equiv \lambda x.N$, with N in β -nf. Then by Proposition 1B.3 we have $\Gamma, x:B \vdash_{\lambda\rightarrow}^{\text{Cu}} N : C$ and $A = B \rightarrow C$. By the IH there is a unique $N^{C;\Gamma,x:B}$ for N . It is easy to verify that $M^{A;\Gamma} \equiv \lambda x^B. N^{C;\Gamma,x:B}$ is the unique way to type M . ■

NOTATION. If M is a closed β -nf, then we write M^A for $M^{A;\emptyset}$.

1B.25. COROLLARY. (i) Let $M \in \Lambda_{\rightarrow}^{\text{Ch}}$ be a closed β -nf. Then $|M|$ is a closed β -nf and

$$M \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \Rightarrow [\vdash_{\lambda\rightarrow}^{\text{Cu}} |M| : A \text{ & } |M|^A \equiv M].$$

(ii) Let $M \in \Lambda^{\emptyset}$ be a closed β -nf and $\vdash_{\lambda\rightarrow}^{\text{Cu}} M : A$. Then M^A is the unique term satisfying

$$M^A \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \text{ & } |M^A| \equiv M.$$

(iii) The following two sets are ‘isomorphic’

$$\begin{aligned} &\{M \in \Lambda \mid M \text{ is closed, in } \beta\text{-nf, and } \vdash_{\lambda\rightarrow}^{\text{Cu}} M : A\}; \\ &\{M \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \mid M \text{ is closed and in } \beta\text{-nf}\}. \end{aligned}$$

PROOF. (i) By the unicity of M^A .

(ii) By the Proposition.

(iii) By (i) and (ii). ■

The applicability of this result will be enhanced once we know that every term typable in $\lambda_{\rightarrow}^{\text{A}}$ (whatever version) has a $\beta\eta$ -nf.

The translation \parallel preserves reduction and conversion.

1B.26. PROPOSITION. Let $R = \beta, \eta$ or $\beta\eta$. Then

(i) Let $M, N \in \Lambda_{\rightarrow}^{\text{Ch}}$. Then $M \rightarrow_R N \Rightarrow |M| \rightarrow_R |N|$. In diagram

$$\begin{array}{ccc} M & \xrightarrow{R} & N \\ || & & || \\ |M| & \xrightarrow[R]{} & |N| \end{array}$$

(ii) Let $M, N \in \Lambda_{\rightarrow}^{\text{Cu}, \Gamma}(A)$, $M = |M'|$, with $M' \in \Lambda_{\rightarrow}^{\text{Ch}}(A)$. Then

$$\begin{aligned} M \rightarrow_R N &\Rightarrow \exists N' \in \Lambda_{\rightarrow}^{\text{Ch}}(A). \\ |N'| &\equiv N \ \& \ M' \rightarrow_R N'. \end{aligned}$$

In diagram

$$\begin{array}{ccc} M' & \xrightarrow[R]{} & N' \\ || & & || \\ M & \xrightarrow{R} & N \end{array}$$

(iii) Let $M, N \in \Lambda_{\rightarrow}^{\text{Cu}, \Gamma}(A)$, $N = |N'|$, with $N' \in \Lambda_{\rightarrow}^{\text{Ch}}(A)$. Then

$$\begin{aligned} M \rightarrow_R N &\Rightarrow \exists M' \in \Lambda_{\rightarrow}^{\text{Ch}}(A). \\ |M'| &\equiv M \ \& \ M' \rightarrow_R N'. \end{aligned}$$

In diagram

$$\begin{array}{ccc} M' & \xrightarrow[R]{} & N' \\ || & & || \\ M & \xrightarrow{R} & N \end{array}$$

(iv) The same results hold for \rightarrow_R and R-conversion.

PROOF. Easy. ■

1B.27. COROLLARY. Define the following two statements.

$$\text{SN}(\lambda_{\rightarrow}^{\text{Cu}}) \triangleq \forall \Gamma \forall M \in \Lambda_{\rightarrow}^{\text{Cu}, \Gamma}. \text{SN}(M).$$

$$\text{SN}(\lambda_{\rightarrow}^{\text{Ch}}) \triangleq \forall M \in \Lambda_{\rightarrow}^{\text{Ch}}. \text{SN}(M).$$

Then

$$\text{SN}(\lambda_{\rightarrow}^{\text{Cu}}) \Leftrightarrow \text{SN}(\lambda_{\rightarrow}^{\text{Ch}}).$$

In fact we will prove in Section 2B that both statements hold.

1B.28. PROPOSITION (*Church-Rosser theorem for $\lambda_{\rightarrow}^{\text{Ch}}$*). On typable terms of $\lambda_{\rightarrow}^{\text{Ch}}$ the Church-Rosser theorem holds for the notions of reduction \rightarrow_{β} and $\rightarrow_{\beta\eta}$.

(i) Let $M, N_1, N_2 \in \Lambda_{\rightarrow}^{\text{Ch}}(A)$. Then

$$M \rightarrow_{\beta\eta} N_1 \ \& \ M \rightarrow_{\beta(\eta)} N_2 \Rightarrow \exists Z \in \Lambda_{\rightarrow}^{\text{Ch}}(A). N_1 \rightarrow_{\beta(\eta)} Z \ \& \ N_2 \rightarrow_{\beta(\eta)} Z.$$

(ii) Let $M, N \in \Lambda_{\rightarrow}^{\text{Ch}}(A)$. Then

$$M =_{\beta(\eta)} N \Rightarrow \exists Z \in \Lambda_{\rightarrow}^{\text{Ch}}(A). M \rightarrow_{\beta(\eta)} Z \ \& \ N \rightarrow_{\beta(\eta)} Z.$$

PROOF. (i) We give two proofs, both borrowing a result from Chapter 2.

Proof 1. We use that every term of $\Lambda_{\rightarrow}^{\text{Ch}}$ has a β -nf, Theorem 2A.13. Suppose $M \rightarrow_{\beta\eta} N_i$, $i \in \{1, 2\}$. Consider the β -nfs N_i^{nf} of N_i . Then $|M| \rightarrow_{\beta\eta} |N_i^{\text{nf}}|$, $i \in \{1, 2\}$. By the CR for untyped lambda terms one has $|N_1^{\text{nf}}| \equiv |N_2^{\text{nf}}|$, and is also in β -nf. By Proposition 1B.24 there exists unique $Z_i \in \Lambda_{\rightarrow}^{\text{Ch}}$ such that $M \rightarrow_{\beta\eta} Z_i$ and $|Z_i| \equiv |N_i^{\text{nf}}|$. But then $Z_1 \equiv Z_2$ and we are done.

Proof 2. Now we use that every term of $\Lambda_{\rightarrow}^{\text{Ch}}$ is β -SN, Theorem 2B.1. It is easy to see that $\rightarrow_{\beta\eta}$ satisfies the weak diamond property; then we are done by Newman's lemma. See e.g. B[1984], Definition 3.1.24 and Proposition 3.1.25.

(ii) As usual from (i). See e.g. B[1984], Theorem 3.1.12. ■

Comparing $\lambda_{\rightarrow}^{\text{Ch}}$ and $\lambda_{\rightarrow}^{\text{dB}}$

There is a close connection between $\lambda_{\rightarrow}^{\text{Ch}}$ and $\lambda_{\rightarrow}^{\text{dB}}$. First we need the following.

1B.29. LEMMA. Let $\Gamma \subseteq \Gamma'$ be bases of $\lambda_{\rightarrow}^{\text{dB}}$. Then

$$\Gamma \vdash_{\lambda_{\rightarrow}}^{\text{dB}} M : A \Rightarrow \Gamma' \vdash_{\lambda_{\rightarrow}}^{\text{dB}} M : A.$$

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

1B.30. DEFINITION. (i) Let $M \in \Lambda_{\rightarrow}^{\text{dB}}$ and suppose $\text{FV}(M) \subseteq \text{dom}(\Gamma)$. Define M^{Γ} inductively as follows.

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

(ii) Let $M \in \Lambda_{\rightarrow}^{\text{Ch}}(A)$ in $\lambda_{\rightarrow}^{\text{Ch}}$. Define M^- , a pseudo-term of $\lambda_{\rightarrow}^{\text{dB}}$, as follows.

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

1B.31. EXAMPLE. To get the (easy) intuition, consider the following.

$$\begin{aligned} (\lambda x:A.x)^{\emptyset} &\equiv (\lambda x^A.x^A); \\ (\lambda x^A.x^A)^- &\equiv (\lambda x:A.x); \\ (\lambda x:A \rightarrow B.xy)^{\{y:A\}} &\equiv \lambda x^{A \rightarrow B}.x^{A \rightarrow B}y^A; \\ \Gamma_{(\lambda x^{A \rightarrow B}.x^{A \rightarrow B}y^A)} &= \{y:A\}, \quad \text{cf. Definition 1B.18.} \end{aligned}$$

1B.32. PROPOSITION. (i) Let $M \in \Lambda_{\rightarrow}^{\text{Ch}}$ and Γ be a basis of $\lambda_{\rightarrow}^{\text{dB}}$. Then

$$M \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \Leftrightarrow \Gamma_M \vdash_{\lambda_{\rightarrow}}^{\text{dB}} M^- : A.$$

$$(ii) \quad \Gamma \vdash_{\lambda_{\rightarrow}}^{\text{dB}} M : A \Leftrightarrow M^{\Gamma} \in \Lambda_{\rightarrow}^{\text{Ch}}(A).$$

PROOF. (i), (ii)(\Rightarrow) By induction on the definition or the proof 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 1B.29. ■

1B.33. COROLLARY. *In particular, for a type $A \in \mathbb{T}$ one has*

$$A \text{ is inhabited in } \lambda_{\rightarrow}^{\text{Ch}} \Leftrightarrow A \text{ is inhabited in } \lambda_{\rightarrow}^{\text{dB}}.$$

PROOF. Immediate. ■

Again the translation preserves reduction and conversion

1B.34. PROPOSITION. (i) *Let $M, N \in \Lambda_{\rightarrow}^{\text{dB}}$. Then*

$$M \rightarrow_R N \Leftrightarrow M^{\Gamma} \rightarrow_R N^{\Gamma},$$

where $R = \beta, \eta$ or $\beta\eta$.

(ii) *Let $M_1, M_2 \in \Lambda_{\rightarrow}^{\text{Ch}}(A)$ and R as in (i). Then*

$$M_1 \rightarrow_R M_2 \Leftrightarrow M_1^- \rightarrow_R M_2^-.$$

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

PROOF. Easy. ■

Comparing $\lambda_{\rightarrow}^{\text{Cu}}$ and $\lambda_{\rightarrow}^{\text{dB}}$

1B.35. PROPOSITION. (i) $\Gamma \vdash_{\lambda_{\rightarrow}}^{\text{dB}} M : A \Rightarrow \Gamma \vdash_{\lambda_{\rightarrow}}^{\text{Cu}} |M| : A$,

here $|M|$ is defined by leaving out all ‘ $: A$ ’ immediately following binding lambdas.

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

$$\Gamma \vdash_{\lambda_{\rightarrow}}^{\text{Cu}} M : A \Leftrightarrow \exists M'. |M'| \equiv M \& \Gamma \vdash_{\lambda_{\rightarrow}}^{\text{dB}} M' : A.$$

PROOF. As for Proposition 1B.19. ■

Again the implication in (i) cannot be reversed.

The three systems compared

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

1B.36. THEOREM. *Let $M \in \Lambda_{\rightarrow}^{\text{Ch}}$ be in β -nf. Then the following are equivalent.*

- (i) $M \in \Lambda_{\rightarrow}^{\text{Ch}}(A)$.
- (ii) $\Gamma_M \vdash_{\lambda_{\rightarrow}}^{\text{dB}} M^- : A$.
- (iii) $\Gamma_M \vdash_{\lambda_{\rightarrow}}^{\text{Cu}} |M| : A$.
- (iv) $|M|^{A; \Gamma_M} \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \& |M|^{A; \Gamma_M} \equiv M$.

PROOF. By Propositions 1B.32(i), 1B.35, and 1B.24 and the fact that $|M^-| = |M|$ we have

$$\begin{aligned} M \in \Lambda_{\rightarrow}^{\text{Ch}}(A) &\Leftrightarrow \Gamma_M \vdash_{\lambda_{\rightarrow}}^{\text{dB}} M^- : A \\ &\Rightarrow \Gamma_M \vdash_{\lambda_{\rightarrow}}^{\text{Cu}} |M| : A \\ &\Rightarrow |M|^{A; \Gamma_M} \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \& |M|^{A; \Gamma_M} \equiv M \\ &\Rightarrow M \in \Lambda_{\rightarrow}^{\text{Ch}}(A). \blacksquare \end{aligned}$$

1C. Normal inhabitants

In this section we will give an algorithm that enumerates the set of closed inhabitants in β -nf 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. The algorithm will be used by concluding that a certain type A is uninhabited or more generally that a certain class of terms exhausts all inhabitants of A .

Because the various versions of $\lambda \rightarrow^{\mathbb{A}}$ are equivalent as to inhabitation of closed β -nfs, we flexibly jump between the set

$$\{M \in \Lambda \rightarrow^{\text{Ch}}(A) \mid M \text{ closed and in } \beta\text{-nf}\}$$

and

$$\{M \in \Lambda \mid M \text{ closed, in } \beta\text{-nf, and } \vdash_{\lambda \rightarrow}^{\text{Cu}} M : A\},$$

thereby we often write a Curry context $\{x_1:A_1, \dots, x_n:A_n\}$ as $\{x_1^{A_1}, \dots, x_n^{A_n}\}$ and a Church term $\lambda x^0.x^0$ as $\lambda x^0.x$, an intermediate form between the Church and the de Bruijn versions.

We do need to distinguish various kinds of nfs.

1C.1. DEFINITION. Let $A = A_1 \rightarrow \dots \rightarrow A_n \rightarrow \alpha$ and suppose $M \in \Lambda \rightarrow^{\text{Ch}}(A)$.

(i) Then M is in *long-nf*, notation *lnf*, if $M \equiv \lambda x_1^{A_1} \dots x_n^{A_n}.x M_1 \dots M_n$ 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 1E.14 it is proved that if M has a β -nf, which according to Theorem 2B.4 is always the case, then it also has a unique lnf and this will be its unique $\beta\eta^{-1}$ nf. Here η^{-1} is the notion of reduction that is the converse of η .

1C.2. EXAMPLES. (i) $\lambda x^0.x$ is both in $\beta\eta$ -nf and lnf.

(ii) $\lambda f^1.f$ is a $\beta\eta$ -nf but not a lnf.

(iii) $\lambda f^1 x^0.f x$ is a lnf but not a $\beta\eta$ -nf; its $\beta\eta$ -nf is $\lambda f^1.f$.

(iv) The β -nf $\lambda F_2^2 \lambda f^1.F f(\lambda x^0.f x)$ is neither in $\beta\eta$ -nf nor lnf.

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

(vi) A variable $f^{1 \rightarrow 1}$ has as lnf $\lambda g^1 x^0.f(\lambda y^0.g y)x =_{\eta} f^{1 \rightarrow 1}$.

1C.3. PROPOSITION. Every β -nf M has a lnf M^ℓ such that $M^\ell \rightarrow_{\eta} M$.

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

$$M^\ell \equiv (\lambda \vec{x}.y M_1 \dots M_n)^\ell \triangleq \lambda \vec{x} \vec{z}.y M_1^\ell \dots M_n^\ell \vec{z}^\ell$$

where \vec{z} is the longest vector that preserves the type. Then M^ℓ does the job. ■

We will define a *2-level grammar*, see van Wijngaarden [1981], for obtaining all closed inhabitants in lnf of a given type A . We do this via the system $\lambda \rightarrow^{\text{Cu}}$.

1C.4. DEFINITION. Let $\mathcal{L} = \{L(A; \Gamma) \mid A \in \mathbb{T}^{\mathbb{A}}; \Gamma \text{ a context of } \lambda \rightarrow^{\text{Cu}}\}$. Let Σ be the alphabet of the untyped lambda terms. Define the following two-level grammar as a notion of reduction over words over $\mathcal{L} \cup \Sigma$. The elements of \mathcal{L} are the non-terminals (unlike in

a context-free language there are now infinitely many of them) of the form $L(A; \Gamma)$.

$$\begin{aligned} L(\alpha; \Gamma) &\xrightarrow{\quad} xL(B_1; \Gamma) \cdots L(B_n; \Gamma), \quad \text{if } (x:\vec{B} \rightarrow \alpha) \in \Gamma; \\ L(A \rightarrow B; \Gamma) &\xrightarrow{\quad} \lambda x^A.L(B; \Gamma, x^A). \end{aligned}$$

Typical productions of this grammar are the following.

$$\begin{aligned} L(3; \emptyset) &\xrightarrow{\quad} \lambda F^2.L(0; F^2) \\ &\xrightarrow{\quad} \lambda F^2.FL(1; F^2) \\ &\xrightarrow{\quad} \lambda F^2.F(\lambda x^0.L(0; F^2, x^0)) \\ &\xrightarrow{\quad} \lambda F^2.F(\lambda x^0.x). \end{aligned}$$

But one has also

$$\begin{aligned} L(0; F^2, x^0) &\xrightarrow{\quad} FL(1; F^2, x^0) \\ &\xrightarrow{\quad} F(\lambda x_1^0.L(0; F^2, x^0, x_1^0)) \\ &\xrightarrow{\quad} F(\lambda x_1^0.x_1). \end{aligned}$$

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

$$L(3; \emptyset) \Rightarrow\!\Rightarrow \lambda F^2.F(\lambda x^0.F(\lambda x_1^0.x_1)).$$

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

1C.5. PROPOSITION. *Let Γ, M, A be given. Then*

$$L(A; \Gamma) \Rightarrow\!\Rightarrow M \Leftrightarrow \Gamma \vdash M : A \ \& \ M \text{ is in lnf.}$$

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

1C.6. DEFINITION. The 2-level grammar N is defined as follows.

$$\begin{aligned} N(A; \Gamma) &\xrightarrow{\quad} xN(B_1; \Gamma) \cdots N(B_n; \Gamma), \quad \text{if } (x:\vec{B} \rightarrow A) \in \Gamma; \\ N(A \rightarrow B; \Gamma) &\xrightarrow{\quad} \lambda x^A.N(B; \Gamma, x^A). \end{aligned}$$

Now the β -nfs are being produced. As an example we make the following production. Remember that $1 = 0 \rightarrow 0$.

$$\begin{aligned} N(1 \rightarrow 0 \rightarrow 0; \emptyset) &\xrightarrow{\quad} \lambda f^1.N(0 \rightarrow 0; f^1) \\ &\xrightarrow{\quad} \lambda f^1.f. \end{aligned}$$

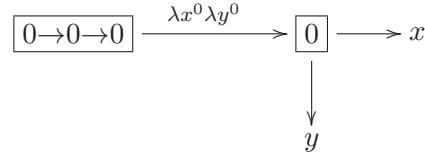
1C.7. PROPOSITION. *Let Γ, M, A be given. Then*

$$N(A; \Gamma) \Rightarrow\!\Rightarrow M \Leftrightarrow \Gamma \vdash M : A \ \& \ M \text{ is in } \beta\text{-nf.}$$

Inhabitation machines

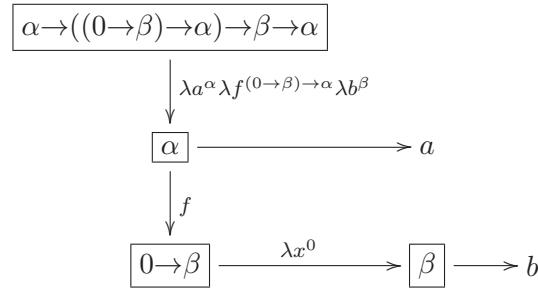
Inspired by this proposition one can introduce for each type A a machine M_A producing the set of closed lnfs of that type. If one is interested in terms containing free variables $x_1^{A_1}, \dots, x_n^{A_n}$, then one can also find these terms by considering the machine for the type $A_1 \rightarrow \dots \rightarrow A_n \rightarrow A$ and looking at the sub-production at node A . This means that a normal inhabitant M_A of type A can be found as a closed inhabitant $\lambda \vec{x}.M_A$ of type $A_1 \rightarrow \dots \rightarrow A_n \rightarrow A$.

1C.8. EXAMPLES. (i) $A = 0 \rightarrow 0 \rightarrow 0$. Then M_A is



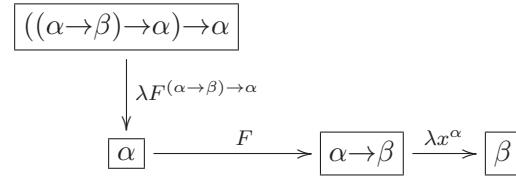
This shows that the type 1_2 has two closed inhabitants: $\lambda xy.x$ and $\lambda xy.y$. We see that the two arrows leaving $\boxed{0}$ represent a choice.

(ii) $A = \alpha \rightarrow ((0 \rightarrow \beta) \rightarrow \alpha) \rightarrow \beta \rightarrow \alpha$. Then M_A is



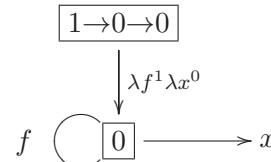
Again there are only two inhabitants, but now the production of them is rather different: $\lambda a f b . a$ and $\lambda a f b . f(\lambda x^0.b)$.

(iii) $A = ((\alpha \rightarrow \beta) \rightarrow \alpha) \rightarrow \alpha$. Then M_A is



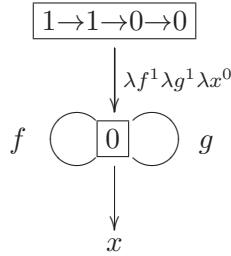
This type, corresponding to Peirce's law, does not have any inhabitants.

(iv) $A = 1 \rightarrow 0 \rightarrow 0$. Then M_A is



This is the type Nat having the Church's numerals $\lambda f^1 x^0. f^n x$ as inhabitants.

(v) $A = 1 \rightarrow 1 \rightarrow 0 \rightarrow 0$. Then M_A is

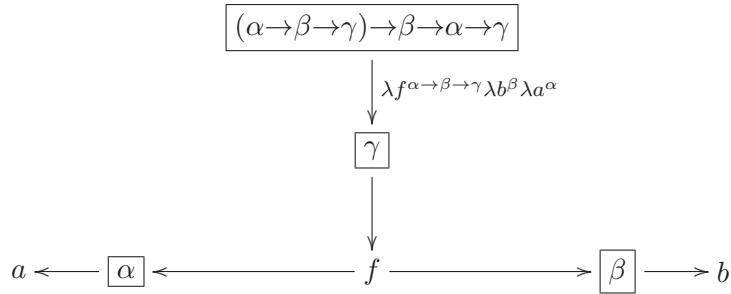


Inhabitants of this type represent words over the alphabet $\Sigma = \{f, g\}$, for example

$$\lambda f^1 g^1 x^0 . f g f f g f g g x,$$

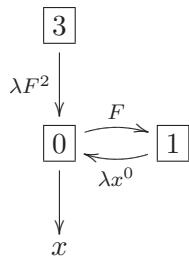
where we have to insert parentheses associating to the right.

(vi) $A = (\alpha \rightarrow \beta \rightarrow \gamma) \rightarrow \beta \rightarrow \alpha \rightarrow \gamma$. Then M_A is



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

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

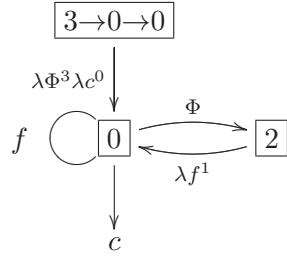


This type 3 has inhabitants having more and more binders:

$$\lambda F^2 . F(\lambda x_0^0 . F(\lambda x_1^0 . F(\dots(\lambda x_n^0 . x_i)))).$$

The novel phenomenon that the binder λx^0 may go round and round forces us to give new incarnations $\lambda x_0^0, \lambda x_1^0, \dots$ 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) $A = 3 \rightarrow 0 \rightarrow 0$. Then M_A is

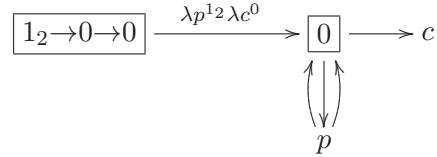


This type, called the *monster* M, does have a potential infinite amount of binding, having as terms e.g.

$$\lambda \Phi^3 c^0. \Phi(\lambda f_1^1.f_1 \Phi(\lambda f_2^1.f_2 f_1 \Phi(\dots(\lambda f_n^1.f_n \dots f_2 f_1 c)\dots))),$$

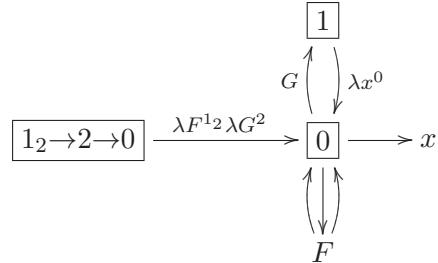
again with inserted parentheses associating to the right. Now a proper bookkeeping of incarnations (of f^1 in this case) becomes necessary, as the f going from $\boxed{0}$ to itself needs to be one that has already been incarnated.

(ix) $A = 1_2 \rightarrow 0 \rightarrow 0$. Then M_A is



This is the type of binary trees, having as elements, e.g. $\lambda p^{1_2} c^0.c$ and $\lambda p^{1_2} c^0.pc(pcc)$. Again, as in example (vi) the outgoing arcs from p (to $\boxed{0}$) should be completed both separately in order to give p its two arguments.

(x) $A = 1_2 \rightarrow 2 \rightarrow 0$. Then M_A is



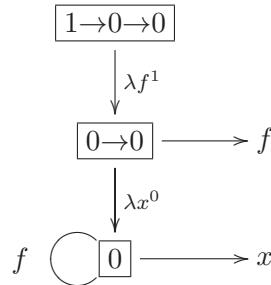
The inhabitants of this type, which we call L , can be thought of as codes for untyped lambda terms. For example the untyped terms $\omega \equiv \lambda x.xx$ and $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$ can be translated to $(\omega)^t \equiv \lambda F^{1_2}G^2.G(\lambda x^0.Fxx)$ and

$$\begin{aligned} (\Omega)^t &\equiv \lambda F^{1_2}G^2.F(G(\lambda x^0.Fxx))(G(\lambda x^0.Fxx)) \\ &=_{\beta} \lambda FG.F((\omega)^tFG)((\omega)^tFG) \\ &=_{\beta} (\omega)^t \cdot_L (\omega)^t, \end{aligned}$$

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 in this example.

Following the 2-level grammar N one can make inhabitation machines for β -nfs M_A^β .

1C.9. EXAMPLE. We show how the production machine for β -nfs differs from the one for lnfs. Let $A = 1 \rightarrow 0 \rightarrow 0$. 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_A^β .



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

1D. Representing data types

In this section it will be shown that first order algebraic data types can be represented in Λ_{\rightarrow}^0 . This means that an algebra \mathcal{A} can be embedded into the set of closed terms in β -nf in $\Lambda_{\rightarrow}^{\text{Cu}}(A)$. That we work with the Curry version is as usual not essential.

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 $\text{Bool} = \text{Bool}_A$ etcetera. Often one takes $A = 0$.

Booleans

1D.1. DEFINITION. Define $\text{Bool} \equiv \text{Bool}_A$

$$\begin{aligned}\text{Bool} &\triangleq A \rightarrow A \rightarrow A; \\ \text{true} &\triangleq \lambda xy.x; \\ \text{false} &\triangleq \lambda xy.y.\end{aligned}$$

Then $\text{true} \in \Lambda_{\rightarrow}^0(\text{Bool})$ and $\text{false} \in \Lambda_{\rightarrow}^0(\text{Bool})$.

1D.2. PROPOSITION. *There are terms **not**, **and**, **or**, **imp**, **iff** with the expected behavior on Booleans. For example $\text{not} \in \Lambda_{\rightarrow}^0(\text{Bool} \rightarrow \text{Bool})$ and*

$$\begin{aligned}\text{not true} &=_{\beta} \text{false}, \\ \text{not false} &=_{\beta} \text{true}.\end{aligned}$$

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

$$\text{imp} \triangleq \lambda ab.\text{or}(\text{not } a)b.$$

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

Natural numbers

1D.3. DEFINITION. The set of natural numbers can be represented as a type

$$\mathbf{Nat} \triangleq (A \rightarrow A) \rightarrow A \rightarrow A.$$

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

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

where

$$\begin{aligned} f^0 x &\triangleq x; \\ f^{n+1} x &\triangleq f(f^n x). \end{aligned}$$

Then $\mathbf{c}_n \in \Lambda_{\rightarrow}^{\emptyset}(\mathbf{Nat})$ for every $n \in \mathbb{N}$. The representation \mathbf{c}_n of $n \in \mathbb{N}$ is called *Church's numeral*. In B[1984] another representation of numerals was used.

1D.4. PROPOSITION. (i) *There exists a term $S^+ \in \Lambda_{\rightarrow}^{\emptyset}(\mathbf{Nat} \rightarrow \mathbf{Nat})$ such that*

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

(ii) *There exists a term $\text{zero?} \in \Lambda_{\rightarrow}^{\emptyset}(\mathbf{Nat} \rightarrow \mathbf{Bool})$ such that*

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

PROOF. (i) Take $S^+ \triangleq \lambda n \lambda f x. f(n f x)$. Then

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

(ii) Take $\text{zero?} \equiv \lambda n \lambda a b. n(\mathbf{K} b) a$. Then

$$\begin{aligned} \text{zero?} \mathbf{c}_0 &=_{\beta} \lambda a b. \mathbf{c}_0(\mathbf{K} b) a \\ &=_{\beta} \lambda a b. a \\ &\equiv \text{true}; \\ \text{zero?}(S^+ x) &=_{\beta} \lambda a b. S^+ x(\mathbf{K} b) a \\ &=_{\beta} \lambda a b. (\lambda f y. f(x f y))(\mathbf{K} b) a \\ &=_{\beta} \lambda a b. \mathbf{K} b(x(\mathbf{K} b) a) \\ &=_{\beta} \lambda a b. b \\ &\equiv \text{false.} \blacksquare \end{aligned}$$

1D.5. DEFINITION. (i) A function $f : \mathbb{N}^k \rightarrow \mathbb{N}$ is called *λ -definable* with respect to \mathbf{Nat} if there exists a term $F \in \Lambda_{\rightarrow}$ such that $F \mathbf{c}_{n_1} \cdots \mathbf{c}_{n_k} = \mathbf{c}_{f(n_1, \dots, n_k)}$ for all $\vec{n} \in \mathbb{N}^k$.

(ii) For different data types represented in λ_{\rightarrow} one defines λ -definability similarly.

Addition and multiplication are λ -definable in λ_{\rightarrow} .

1D.6. PROPOSITION. (i) *There is a term $\text{plus} \in \Lambda_{\rightarrow}^{\emptyset}(\mathbf{Nat} \rightarrow \mathbf{Nat} \rightarrow \mathbf{Nat})$ satisfying*

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

(ii) There is a term $\text{times} \in \Lambda_{\rightarrow}^{\emptyset}(\text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat})$ such that

$$\text{times } \mathbf{c}_n \mathbf{c}_m =_{\beta} \mathbf{c}_{n+m}.$$

PROOF. (i) Take $\text{plus} \triangleq \lambda nm \lambda fx.nf(mfx)$. Then

$$\begin{aligned} \text{plus } \mathbf{c}_n \mathbf{c}_m &=_{\beta} \lambda fx. \mathbf{c}_n f(\mathbf{c}_m f x) \\ &=_{\beta} \lambda fx. f^n(f^m x) \\ &\equiv \lambda fx. f^{n+m} x \\ &\equiv \mathbf{c}_{n+m}. \end{aligned}$$

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

$$\begin{aligned} \text{times } \mathbf{c}_n \mathbf{c}_m &=_{\beta} \lambda fx. \mathbf{c}_m (\lambda y. \mathbf{c}_n fy) x \\ &=_{\beta} \lambda fx. \mathbf{c}_m (\lambda y. f^n y) x \\ &=_{\beta} \lambda fx. (\underbrace{f^n(f^n(\cdots(f^n x)..))}_{m \text{ times}}) \\ &\equiv \lambda fx. f^{n+m} x \\ &\equiv \mathbf{c}_{n+m}. \blacksquare \end{aligned}$$

1D.7. COROLLARY. For every polynomial $p \in \mathbb{N}[x_1, \dots, x_k]$ there is a closed term $M_p \in \Lambda_{\rightarrow}^{\emptyset}(\text{Nat}^k \rightarrow \text{Nat})$ such that $\forall n_1, \dots, n_k \in \mathbb{N}. M_p \mathbf{c}_{n_1} \cdots \mathbf{c}_{n_k} =_{\beta} \mathbf{c}_{p(n_1, \dots, n_k)}$. \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 $\lambda_{\rightarrow}^{\mathbb{A}}$. In Schwichtenberg [1976] or Statman [1982] it is proved that exactly these so-called extended polynomials are definable in $\lambda_{\rightarrow}^{\mathbb{A}}$. Hence primitive recursion cannot be defined in $\lambda_{\rightarrow}^{\mathbb{A}}$; in fact not even the predecessor function, see Proposition 2D.21.

Words over a finite alphabet

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

1D.8. DEFINITION. (i) The *type for words* in Σ^* is

$$\text{Sigma}^* \triangleq (0 \rightarrow 0)^k \rightarrow 0 \rightarrow 0.$$

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

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

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

$$\begin{aligned} \underline{\epsilon} &\triangleq \lambda a_1 \cdots a_k x. x \\ &= K^k I. \end{aligned}$$

Now we show that the operation concatenation is λ -definable with respect to Sigma^* .

1D.9. PROPOSITION. There exists a term $\text{concat} \in \Lambda_{\rightarrow}^{\emptyset}(\text{Sigma}^* \rightarrow \text{Sigma}^* \rightarrow \text{Sigma}^*)$ such that for all $w, v \in \Sigma^*$

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

PROOF. Define

$$\text{concat} \triangleq \lambda wv.\vec{a}x.w\vec{a}(v\vec{a}x).$$

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

1D.10. PROPOSITION. (i) *There exists a term $\text{empty}_? \in \Lambda_{\rightarrow}^{\emptyset}(\Sigma^*)$ such that*

$$\begin{aligned}\text{empty}_? \underline{\epsilon} &= \text{true}; \\ \text{empty}_? \underline{w} &= \text{false}, \quad \text{if } w \neq \epsilon.\end{aligned}$$

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

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

PROOF. (i) Take $\text{empty}_? \equiv \lambda wpq.w(Kq)^{\sim k}p$.

(ii) Take $F \equiv \lambda w\lambda x\vec{a}.\text{empty}_? w(w_0\vec{a}x)(Gw\vec{a}x)$. ■

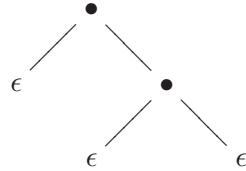
One cannot define terms ‘car’ or ‘cdr’ such that $\text{car } \underline{aw} = \underline{a}$ and $\text{cdr } \underline{aw} = \underline{w}$.

Trees

1D.11. DEFINITION. The set of *binary trees*, notation T^2 , is defined by the following simplified syntax

$$t ::= \epsilon \mid p(t, t)$$

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 \mathbb{T}^0 .

1D.12. DEFINITION. (i) The set T^2 will be represented by the type

$$\textcolor{blue}{T^2} \triangleq (0^2 \rightarrow 0) \rightarrow 0 \rightarrow 0.$$

(ii) Define for $t \in T^2$ its representation \underline{t} inductively as follows.

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

(iii) Write

$$\begin{aligned}\textcolor{blue}{E} &\triangleq \lambda pe.e; \\ \textcolor{blue}{P} &\triangleq \lambda tspe.p(tpe)(spe).\end{aligned}$$

Note that for $t \in T^2$ one has $\underline{t} \in \Lambda_{\rightarrow}^{\emptyset}(\textcolor{blue}{T}^2)$

The following follows immediately from this definition.

1D.13. PROPOSITION. *The map $\underline{_} : T^2 \rightarrow \top^2$ can be defined inductively as follows*

$$\begin{aligned}\underline{\epsilon} &\triangleq E; \\ \underline{p(t,s)} &\triangleq Pt\underline{s}.\end{aligned}$$

Interesting functions, like the one that selects one of the two branches of a tree cannot be defined in λ_{\rightarrow}^0 . The type \top^2 will play an important role in Section 3D.

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 λ_{\rightarrow}^0 . 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 special cases one can do better. Every finite algebra can be represented with all possible functions on them. Pairing with projections can be represented.

1D.14. DEFINITION. (i) An *algebra* is a set A with a specific finite set of operators of different arity:

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

(ii) An n-ary function $F : A^n \rightarrow A$ is called *algebraic* if F can be defined explicitly from the given constructors by composition. For example

$$F = \lambda a_1 a_2. g_1(a_1, (g_2(f_1(a_2), c_2)))$$

is a binary algebraic function, usually specified as

$$F(a_1, a_2) = g_1(a_1, (g_2(f_1(a_2), c_2))).$$

(iii) An element a of A is called *algebraic* if 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 called *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.

1D.15. THEOREM. *For a free algebra \mathcal{A} there is a $\underline{\mathcal{A}} \in \top^0$ and $\lambda a. \underline{a} : \mathcal{A} \rightarrow \Lambda_{\rightarrow}^0(\underline{\mathcal{A}})$ satisfying the following.*

- (i) \underline{a} is a lnf, for every $a \in \mathcal{A}$.
- (ii) $\underline{a} =_{\beta\eta} \underline{b} \Leftrightarrow a = b$.
- (iii) $\Lambda_{\rightarrow}^0(\underline{\mathcal{A}}) = \{\underline{a} \mid a \in \mathcal{A}\}$, up to $\beta\eta$ -conversion.

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

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

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

$$\begin{aligned} \text{test } \underline{c} &= \mathbf{c}_0; \\ \text{test } \underline{f}_1(a) &= \mathbf{c}_1; \\ \text{test } \underline{f}_2(a, b) &= \mathbf{c}_2; \\ &\dots \\ \text{test } \underline{f}_n(a_1, \dots, a_n) &= \mathbf{c}_n. \end{aligned}$$

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

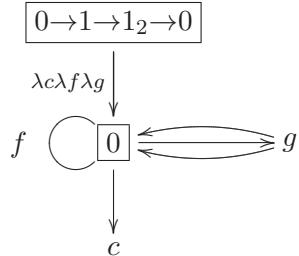
$$a = g(c, f(c))$$

is represented by

$$\underline{a} = \lambda c f g. g c (f c) \in \Lambda(0 \rightarrow 1 \rightarrow 1_2 \rightarrow 0).$$

Taking $\underline{\mathcal{A}} = 0 \rightarrow 1 \rightarrow 1_2 \rightarrow 0$ we will verify the claims. First realize that \underline{a} is constructed from a via $a^\sim = g c (f c)$ and then taking the closure $\underline{a} = \lambda c f g. a^\sim$.

- (i) Clearly the \underline{a} are in Infs .
- (ii) If a and b are different, then their representations $\underline{a}, \underline{b}$ are different Infs , hence $\underline{a} \neq_{\beta\eta} \underline{b}$.
- (iii) The inhabitation machine $M_{\underline{\mathcal{A}}} = M_{0 \rightarrow 1 \rightarrow 1_2 \rightarrow 0}$ looks like



It follows that for every $M \in \Lambda_{\rightarrow}^{\phi}(\underline{\mathcal{A}})$ one has $M =_{\beta\eta} \lambda c f g. a^\sim = \underline{a}$ for some $a \in \mathcal{A}$. This shows that $\Lambda_{\rightarrow}^{\phi}(\underline{\mathcal{A}}) \subseteq \{\underline{a} \mid a \in \mathcal{A}\}$. The converse inclusion is trivial. In the general case (for other data types \mathcal{A}) one has that $\text{rk}(\underline{\mathcal{A}}) = 2$. Hence the Inf inhabitants of $\underline{\mathcal{A}}$ have for example the form $\lambda c f_1 f_2 g_1 g_2. P$, where P is a typable combination of the variables $c, f_1^1, f_2^1, g_1^{12}, g_2^{12}$. This means that the corresponding inhabitation machine is similar and the argument generalizes.

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

$$\begin{aligned} \underline{c} &\triangleq \lambda c f g. c && : \underline{\mathcal{A}}; \\ \underline{f} &\triangleq \lambda a c f g. f(a c f g) && : \underline{\mathcal{A}} \rightarrow \underline{\mathcal{A}}; \\ \underline{g} &\triangleq \lambda a b c f g. g(a c f g)(b c f g) && : \underline{\mathcal{A}}^2 \rightarrow \underline{\mathcal{A}}. \end{aligned}$$

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

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

$$\underline{h} \triangleq \lambda a b. \underline{g} a (\underline{f} b) : \mathcal{A}^2 \rightarrow \mathcal{A}.$$

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

(v) Take $\text{test} \triangleq \lambda a f c. a(\mathbf{c}_0 f c)(\lambda x. \mathbf{c}_1 f c)(\lambda x y. \mathbf{c}_2 f c)$. ■

1D.16. 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

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

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 1E.13.

1D.17. COROLLARY. *Every freely generated multi-sorted first-order algebra can be represented in a way similar to that in Theorem 1D.15.*

PROOF. Similar to that of the Theorem. ■

Finite Algebras

For finite algebras one can do much better.

1D.18. THEOREM. *For every finite set $X = \{a_1, \dots, a_n\}$ there exists a type $\underline{X} \in \mathbb{T}^0$ and elements $\underline{a}_1, \dots, \underline{a}_n \in \Lambda_{\rightarrow}^{\phi}(\underline{X})$ such that the following holds.*

- (i) $\Lambda_{\rightarrow}^{\phi}(\underline{X}) = \{\underline{a} \mid a \in X\}$.
- (ii) *For all k and $f : X^k \rightarrow X$ there exists an $\underline{f} \in \Lambda_{\rightarrow}^{\phi}(\underline{X}^k \rightarrow \underline{X})$ such that*

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

PROOF. Take $\underline{X} = 1_n = 0^n \rightarrow 0$ and $\underline{a}_i = \lambda b_1 \cdots b_n. b_i \in \Lambda_{\rightarrow}^{\phi}(1_n)$.

(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 X , say $f = a_i$. Take $\underline{f} = \underline{a}_i$.

Now suppose we can represent all k -ary functions. Given $f : X^{k+1} \rightarrow X$, define for $b \in X$

$$\underline{f}_b(b_1, \dots, b_k) \triangleq f(b, b_1, \dots, b_k).$$

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

$$\underline{f} \triangleq \lambda b\vec{b}. b(\underline{f}_{a_1}\vec{b}) \cdots (\underline{f}_{a_n}\vec{b}),$$

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

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

One even can faithfully represent the full type structure over X as closed terms of Λ_{\rightarrow}^0 , see Exercise 2E.22.

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 $\Sigma = \{a_1, \dots, a_n\}$ can be seen as an algebra with constant ϵ and further constructors $f_{a_i} = \lambda w.a_iw$. The representations given are particular cases of the theorems about free and finite algebras.

Pairing

In the untyped lambda calculus there exists a way to store two terms in such a way that they can be retrieved.

$$\begin{aligned} \text{pair} &\triangleq \lambda abz.zab; \\ \text{left} &\triangleq \lambda z.z(\lambda xy.x); \\ \text{right} &\triangleq \lambda z.z(\lambda xy.y). \end{aligned}$$

These terms satisfy

$$\begin{aligned} \text{left}(\text{pair } MN) &=_{\beta} (\text{pair } MN)(\lambda xy.x) \\ &=_{\beta} (\lambda z.zMN)(\lambda xy.x) \\ &=_{\beta} M; \\ \text{right}(\text{pair } MN) &=_{\beta} N. \end{aligned}$$

The triple of terms $\langle \text{pair}, \text{left}, \text{right} \rangle$ is called a (notion of) ‘ β -pairing’.

We will translate these notions to Λ_{\rightarrow}^0 . We work with the Curry version.

1D.19. DEFINITION. Let $A, B \in \mathbb{T}$ and let R be a notion of reduction on Λ .

(i) A *product* with *R-pairing* is a type $A \times B \in \mathbb{T}$ together with terms

$$\begin{aligned} \text{pair} &\in \Lambda_{\rightarrow}(A \rightarrow B \rightarrow (A \times B)); \\ \text{left} &\in \Lambda_{\rightarrow}((A \times B) \rightarrow A); \\ \text{right} &\in \Lambda_{\rightarrow}((A \times B) \rightarrow B), \end{aligned}$$

satisfying for variables x, y

$$\begin{aligned}\mathbf{left}(\mathbf{pair} xy) &=_R x; \\ \mathbf{right}(\mathbf{pair} xy) &=_R y.\end{aligned}$$

(ii) The type $A \underline{\times} B$ is called the *product* and the triple $\langle \mathbf{pair}, \mathbf{left}, \mathbf{right} \rangle$ is called the *R-pairing*.

(iii) An *R-Cartesian product* is a product with R -pairing satisfying moreover for variables z

$$\mathbf{pair}(\mathbf{left} z)(\mathbf{right} z) =_R z.$$

In that case the pairing is called a *surjective R-pairing*.

This pairing cannot be translated to a β -pairing in λ^0_{\rightarrow} with a product $A \underline{\times} B$ for arbitrary types, see [Barendregt \[1974\]](#). But for two equal types one can form the product $A \underline{\times} A$. This makes it possible to represent also heterogeneous products using $\beta\eta$ -conversion.

1D.20. LEMMA. *For every type $A \in \mathbb{T}^0$ there is a product $A \underline{\times} A \in \mathbb{T}^0$ with β -pairing $\mathbf{pair}_0^A, \mathbf{left}_0^A$ and \mathbf{right}_0^A .*

PROOF. Take

$$\begin{aligned}A \underline{\times} A &\triangleq (A \rightarrow A \rightarrow A) \rightarrow A; \\ \mathbf{pair}_0^A &\triangleq \lambda mnz.zmn; \\ \mathbf{left}_0^A &\triangleq \lambda p.pK; \\ \mathbf{right}_0^A &\triangleq \lambda p.pK_*.\blacksquare\end{aligned}$$

1D.21. PROPOSITION ([Grzegorczyk \[1964\]](#)). *Let $A, B \in \mathbb{T}^0$ be arbitrary types. Then there is a product $A \underline{\times} B \in \mathbb{T}^0$ with $\beta\eta$ -pairing $\langle \mathbf{pair}_0^{A,B}, \mathbf{left}_0^{A,B}, \mathbf{right}_0^{A,B} \rangle$ such that*

$$\begin{aligned}\mathbf{pair}_0^{A,B} &\in \Lambda_0, \\ \mathbf{left}_0^{A,B}, \mathbf{right}_0^{A,B} &\in \Lambda_0^{\{z:0\}},\end{aligned}$$

and

$$\text{rk}(A \underline{\times} B) = \max\{\text{rk}(A), \text{rk}(B), 2\}.$$

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

$$A \underline{\times} B \triangleq A(1) \rightarrow \cdots \rightarrow A(n) \rightarrow B(1) \rightarrow \cdots \rightarrow B(m) \rightarrow 0 \underline{\times} 0,$$

where $0 \underline{\times} 0 \triangleq (0 \rightarrow 0 \rightarrow 0) \rightarrow 0$. Then

$$\begin{aligned}\text{rk}(A \underline{\times} B) &= \max_{i,j}\{\text{rk}(A_i) + 1, \text{rk}(B_j) + 1, \text{rk}(0^2 \rightarrow 0) + 1\} \\ &= \max\{\text{rk}(A), \text{rk}(B), 2\}.\end{aligned}$$

Define z_A inductively: $z_0 \triangleq z; z_{A \rightarrow B} \triangleq \lambda a.z_B$. Then $z_A \in \Lambda_0^{z:0}(A)$. Write $\vec{x} = x_1, \dots, x_n, \vec{y} = y_1, \dots, y_m, \vec{z}_A = z_{A(1)}, \dots, z_{A(n)}$ and $\vec{z}_B = z_{B(1)}, \dots, z_{B(m)}$. Now define

$$\begin{aligned}\mathbf{pair}_0^{A,B} &\triangleq \lambda mn.\lambda \vec{x} \vec{y}.\mathbf{pair}_0^0(m\vec{x})(n\vec{y}); \\ \mathbf{left}_0^{A,B} &\triangleq \lambda p.\lambda \vec{x}.\mathbf{left}_0^0(p\vec{x}\vec{z}_B); \\ \mathbf{right}_0^{A,B} &\triangleq \lambda p.\lambda \vec{x}.\mathbf{right}_0^0(p\vec{z}_A\vec{y}).\end{aligned}$$

Then e.g.

$$\begin{aligned}\text{left}_0^{A,B}(\text{pair}_0^{A,B} MN) &=_{\beta} \lambda \vec{x}. \text{left}_0^0(\text{pair}_0^0 MN \vec{x} \vec{z}_B) \\ &=_{\beta} \lambda \vec{x}. \text{left}_0^0[\text{pair}_0^0(M \vec{x})(N \vec{z}_B)] \\ &=_{\beta} \lambda \vec{x}.(M \vec{x}) \\ &=_\eta M. \blacksquare\end{aligned}$$

In Barendregt [1974] it is proved that η -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 $\beta\eta$ -conversion. In Section 5B we will discuss systems extended with surjective pairing. With similar techniques as in mentioned paper it can be shown that in $\lambda_\rightarrow^\infty$ there is no $\beta\eta$ -pairing function $\text{pair}_0^{\alpha,\beta}$ for base types. In section 2.3 we will encounter other differences between $\lambda_\rightarrow^\infty$ and λ_\rightarrow^0 .

1D.22. PROPOSITION. Let $A_1, \dots, A_n \in \mathbb{T}^0$. There are closed terms

$$\begin{aligned}\text{tuple}^n : A_1 \rightarrow \dots \rightarrow A_n \rightarrow (A_1 \times \dots \times A_n), \\ \text{proj}_k^n : A_1 \times \dots \times A_n \rightarrow A_k,\end{aligned}$$

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

$$\text{proj}_k^n(\text{tuple}^n M_1 \dots M_n) =_{\beta\eta} M_k.$$

PROOF. By iterating pairing. \blacksquare

1D.23. NOTATION. If there is little danger of confusion and the \vec{M}, N are of the right type we write

$$\begin{aligned}\langle M_1, \dots, M_n \rangle &\triangleq \text{tuple}^n M_1 \dots M_n; \\ N \cdot k &\triangleq \text{proj}_k^n N.\end{aligned}$$

Then $\langle M_1, \dots, M_n \rangle \cdot k = M_k$, for $1 \leq k \leq n$.

1E. Exercises

1E.1. Find types for

$$\begin{aligned}\text{B} &\triangleq \lambda xyz.x(yz); \\ \text{C} &\triangleq \lambda xyz.xzy; \\ \text{C}_* &\triangleq \lambda xy.yx; \\ \text{K}_* &\triangleq \lambda xy.y; \\ \text{W} &\triangleq \lambda xy.xyy.\end{aligned}$$

- 1E.2. Find types for $\text{SKK}, \lambda xy.y(\lambda z.zxx)x$ and $\lambda fx.f(f(fx))$.
- 1E.3. Show that $\text{rk}(A \rightarrow B \rightarrow C) = \max\{\text{rk}(A) + 1, \text{rk}(B) + 1, \text{rk}(C)\}$.
- 1E.4. Show that if $M \equiv P[x := Q]$ and $N \equiv (\lambda x.P)Q$, then M may have a type in $\lambda_\rightarrow^{\text{Cu}}$ but N not. A similar observation can be made for pseudo-terms of $\lambda_\rightarrow^{\text{dB}}$.
- 1E.5. Show the following.
 - (i) $\lambda xy.(xy)x \notin \Lambda_\rightarrow^{\text{Cu}, \phi}$.
 - (ii) $\lambda xy.x(yx) \in \Lambda_\rightarrow^{\text{Cu}, \phi}$.
- 1E.6. Find inhabitants of $(A \rightarrow B \rightarrow C) \rightarrow B \rightarrow A \rightarrow C$ and $(A \rightarrow A \rightarrow B) \rightarrow A \rightarrow B$.

- 1E.7. [van Benthem] Show that $\Lambda_{\rightarrow}^{\text{Ch}}(A)$ and $\Lambda_{\rightarrow}^{\text{Cu},\phi}(A)$ are for some $A \in \mathbb{T}^{\mathbb{A}}$ not a context-free language.
- 1E.8. Define in λ_{\rightarrow}^0 the *pseudo-negation* $\sim A \triangleq A \rightarrow 0$. Construct an inhabitant of $\sim\sim\sim A \rightarrow \sim A$.
- 1E.9. Prove the following, see definition 1B.30.
- Let $M \in \Lambda_{\rightarrow}^{\text{dB}}$ with $\text{FV}(M) \subseteq \text{dom}(\Gamma)$, then $(M^{\Gamma})^- \equiv M$ and $\Gamma_{M^{\Gamma}} \subseteq \Gamma$.
 - Let $M \in \Lambda_{\rightarrow}^{\text{Ch}}$, then $(M^-)^{\Gamma_M} \equiv M$.
- 1E.10. Construct a term F with $\vdash_{\lambda_{\rightarrow}^0} F : \top_2 \rightarrow \top_2$ such that for trees t one has $Ft =_{\beta} t^{\text{mir}}$, where t^{mir} is the mirror image of t , defined by

$$\begin{aligned}\epsilon^{\text{mir}} &\triangleq \epsilon; \\ (p(t, s))^{\text{mir}} &\triangleq p(s^{\text{mir}}, t^{\text{mir}}).\end{aligned}$$

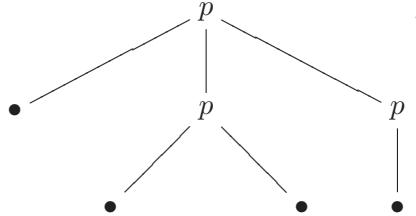
- 1E.11. A term M is called *proper* if all λ 's appear in the prefix of M , i.e. $M \equiv \lambda \vec{x}.N$ and there is no λ occurring in N . Let A be a type such that $\Lambda_{\rightarrow}^{\phi}(A)$ is not empty. Show that

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

- 1E.12. Determine the class of closed inhabitants of the types 4 and 5.
- 1E.13. The collection of multi-ary trees can be seen as part of a multi-sorted algebra with sorts MTree and L_{MTree} as follows.

$$\begin{aligned}\text{nil} &\in L_{\text{Mtree}}; \\ \text{cons} &\in \text{Mtree} \rightarrow L_{\text{Mtree}} \rightarrow L_{\text{Mtree}}; \\ p &\in L_{\text{Mtree}} \rightarrow \text{Mtree}.\end{aligned}$$

Represent this multi-sorted free algebra in λ_{\rightarrow}^0 . Construct the lambda term representing the tree



- 1E.14. 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 $\lambda x_1 \cdots x_n.yM_1 \cdots M_m$ or $\lambda x_1 \cdots x_n.(\lambda x.M_0)M_1 \cdots M_m$. Then $yM_1 \cdots M_m$ (or $(\lambda x.M_0)M_1 \cdots M_m$) is the *matrix* of M and the (M_0, M_1, \dots, 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 0 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.)

- M is in lnf iff M is a β -nf and f.e. expanded.
- If $M =_{\beta\eta} N_1 =_{\beta\eta} N_2$ 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 $N_i \rightarrow_\eta N\downarrow$ and $N\uparrow \rightarrow_\eta N_i$, for $i = 1, 2$. [Hint. Show that both \rightarrow_η 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 $N \rightarrow_\beta N'$, then N' is f.e. expanded.
- (vi) For all M there is a f.e. expanded M^* such that $M^* \rightarrow_\eta M$.
- (vii) If M has a β -nf, then the lnf of M is the β -nf of M^* , its f.e. expansion.

1E.15. For which types $A \in \mathbb{T}^0$ and $M \in \Lambda_{\rightarrow}(A)$ does one have

$$M \text{ in } \beta\text{-nf} \Rightarrow M \text{ in lnf?}$$

1E.16. (i) Let $M = \lambda x_1 \cdots x_n.x_i M_1 \cdots M_m$ be a β -nf. Define by induction on the length of M its **Φ -normal form**, notation $\Phi(M)$, as follows.

$$\Phi(\lambda \vec{x}.x_i M_1 \cdots M_m) \triangleq \lambda \vec{x}.x_i(\Phi(\lambda \vec{x}.M_1)\vec{x}) \cdots (\Phi(\lambda \vec{x}.M_m)\vec{x}).$$

- (ii) Compute the Φ -nf of $S = \lambda xyz.xz(yz)$.
- (iii) Write $\Phi^{n,m,i} \triangleq \lambda y_1 \cdots y_m \lambda x_1 \cdots x_n.x_i(y_1 \vec{x}) \cdots (y_m \vec{x})$. Then

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

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

- (iv) Show that every closed nf of type A is up to $=_{\beta\eta}$ a product of the $\Phi^{n,m,i}$.
- (v) Write S in such a manner.

1E.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 $\vdash_{\lambda_{\rightarrow}}^{\text{Cu}}$ 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.

$$\begin{array}{c} \{x:A\} \vdash x : A; \\ \\ \frac{\Gamma_1 \vdash M : (A \rightarrow B) \quad \Gamma_2 \vdash N : A}{\Gamma_1 \cup \Gamma_2 \vdash (MN) : B}, \quad \text{provided } \Gamma_1 \cup \Gamma_2 \text{ is a basis;} \\ \\ \frac{\Gamma \vdash M : B}{\Gamma - \{x:A\} \vdash (\lambda x.M) : (A \rightarrow B)}. \end{array}$$

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

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

$$\Gamma \vdash' M' : A \Leftrightarrow \exists M \equiv_{\alpha} M'. \Gamma \vdash M : A.$$

1E.18. Elements in Λ are considered in this book modulo α -conversion, by working with α -equivalence classes. If instead one works with α -conversion, as in [Church \[1941\]](#), then one can consider the following problems on elements M of Λ^φ .

1. Given M , find an α -convert of M with a smallest number of distinct variables.
2. Given $M \equiv_\alpha N$, find a shortest α -conversion from M to N .
3. Given $M \equiv_\alpha N$, find an α -conversion from M to N , which uses the smallest number of variables possible along the way.

Study [Statman \[2007\]](#) for the proofs of the following results.

- (i) There is a polynomial time algorithm for solving problem (1). It is reducible to vertex coloring of chordal graphs.
- (ii) Problem (2) is co-NP complete (in recognition form). The general feedback vertex set problem for digraphs is reducible to problem (2).
- (iii) At most one variable besides those occurring in both M and N is necessary. This appears to be the folklore but the proof is not familiar. A polynomial time algorithm for the α -conversion of M to N using at most one extra variable is given.

CHAPTER 2

PROPERTIES

2A. Normalization

For several applications, for example for the problem of finding all possible inhabitants of a given type, we will need the weak normalization theorem, stating that all typable terms do have a $\beta\eta$ -nf (normal form). The result is valid for all versions of $\lambda_{\rightarrow}^{\mathbb{A}}$ and *a fortiori* for the subsystems λ_{\rightarrow}^0 . The proof is due to Turing and is published posthumously in Gandy [1980b]. In fact all typable terms in these systems are $\beta\eta$ strongly normalizing, which means that all $\beta\eta$ -reductions are terminating. This fact requires more work and will be proved in Section 2B.

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

2A.1. DEFINITION. An *abstract reduction system* (ARS) is a pair (X, \rightarrow_R) , where X is a set and \rightarrow_R is a binary relation on X .

We usually will consider $\Lambda, \Lambda_{\rightarrow}^{\mathbb{A}}$ with reduction relations $\rightarrow_{\beta(\eta)}$ as examples of an ARS.

In the following definition WN, weak normalization, stands for having a nf, while SN, strong normalization, stands for not having infinite reduction paths. A typical example in $(\Lambda, \rightarrow_{\beta})$ is the term $KI\Omega$ that is WN but not SN.

2A.2. DEFINITION. Let (X, R) be an ARS.

- (i) An element $x \in X$ is *in R-normal form* (*R-nf*) if for no $y \in X$ one has $x \rightarrow_R y$.
- (ii) An element $x \in X$ is *R-weakly normalizing* (*R-WN*), notation $x \models R\text{-WN}$ (or simply $x \models \text{WN}$), if for some $y \in X$ one has $x \rightarrow_R y$ and y is in *R-nf*.
- (iii) (X, R) is called *WN*, notation $(X, R) \models \text{WN}$, if

$$\forall x \in X. x \models R\text{-WN}.$$

- (iv) An element $x \in X$ is said to be *R-strongly normalizing* (*R-SN*), notation $x \models R\text{-SN}$ (or simply $x \models \text{SN}$), if every *R*-reduction path starting with x

$$x \rightarrow_R x_1 \rightarrow_R x_2 \rightarrow_R \dots$$

is finite.

- (v) (X, R) is said to be *strongly normalizing*, notation $(X, R) \models R\text{-SN}$ or simply $(X, R) \models \text{SN}$, if

$$\forall x \in X. x \models \text{SN}.$$

One reason why the notion of ARS is interesting is that some properties of reduction can be dealt with in ample generality.

2A.3. DEFINITION. Let (X, R) be an ARS.

(i) We say that (X, R) is *confluent* or satisfies the *Church-Rosser property*, notation $(X, R) \models \text{CR}$, if

$$\forall x, y_1, y_2 \in X. [x \rightarrow_R y_1 \& x \rightarrow_R y_2 \Rightarrow \exists z \in X. y_1 \rightarrow_R z \& y_2 \rightarrow_R z].$$

(ii) We say that (X, R) is *weakly confluent* or satisfies the *weak Church-Rosser property*, notation $(X, R) \models \text{WCR}$, if

$$\forall x, y_1, y_2 \in X. [x \rightarrow_R y_1 \& x \rightarrow_R y_2 \Rightarrow \exists z \in X. y_1 \rightarrow_R z \& y_2 \rightarrow_R z].$$

It is not the case that $\text{WCR} \Rightarrow \text{CR}$, do Exercise 2E.18. However, one has the following result.

2A.4. PROPOSITION (Newman's Lemma). *Let (X, R) be an ARS. Then for (X, R)*

$$\text{WCR} \& \text{SN} \Rightarrow \text{CR}.$$

PROOF. See B[1984], Proposition 3.1.25 or Lemma 5C.8 below, for a slightly stronger localized version. ■

In this section we will show $(\Lambda_{\rightarrow}^{\mathbb{A}}, \rightarrow_{\beta\eta}) \models \text{WN}$.

2A.5. DEFINITION. (i) A *multiset* over \mathbb{N} 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. We also may write this multiset as

$$S = \{3^2, 1^1, 0^1\} = \{3^2, 2^0, 1^1, 0^1\}.$$

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

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

for $k > 3$. Such an S is finite if f has finite *support*, where

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

(ii) Let $\mathcal{S}(\mathbb{N})$ be the collection of all finite multisets over \mathbb{N} . $\mathcal{S}(\mathbb{N})$ can be identified with $\{f \in \mathbb{N}^{\mathbb{N}} \mid \text{support}(f) \text{ is finite}\}$. To each f in this set we let correspond the multiset intuitively denoted by

$$S_f = \{n^{f(n)} \mid n \in \text{support}(f)\}.$$

2A.6. 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 element (just one occurrence) by finitely many lower elements (in the usual order of \mathbb{N}). For example

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

The transitive closure of \rightarrow_S , not required to be reflexive, is called the *multiset order*⁶ and is denoted by $>$. (Another notation for this relation is \rightarrow_S^+ .) So for example

$$\{3, 3, 1, 0\} > \{3, 2, 2, 1, 1, 0, 1, 1, 0\}.$$

In the following result it is shown that $(\mathcal{S}(\mathbb{N}), \rightarrow_S)$ is WN, using an induction up to ω^2 .

2A.7. 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 S-reduction sequence*

$$S \rightarrow_S F(S) \rightarrow_S F^2(S) \rightarrow_S \dots$$

is terminating.

PROOF. By induction on the highest number n occurring 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 $(\mathcal{S}(\mathbb{N}), \rightarrow_S)$ is SN. Although we do not strictly need this fact in this Part, we will give even two proofs of it. It will be used in Part II of this book. 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 $\lambda \rightarrow^\mathbb{A}$ satisfies SN.

2A.8. LEMMA. *The reduction system $(\mathcal{S}(\mathbb{N}), \rightarrow_S)$ is SN.*

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

PROOF₁. Assign to every $S \in \mathcal{S}(\mathbb{N})$ an ordinal $\#S < \omega^\omega$ as suggested by the following examples.

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

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

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

Notice that

$$S_1 \rightarrow_S S_2 \Rightarrow \#S_1 > \#S_2$$

(in the example because $\omega^3 > 3\omega^2 + \omega$). Hence by the well-foundedness of the ordinals the result follows. ■₁

⁶We consider both irreflexive, usually denoted by $<$ or its converse $>$, and reflexive order relations, usually denoted by \leq or its converse \geq . From $<$ we can define the reflexive version \leq by

$$a \leq b \Leftrightarrow a = b \text{ or } a < b.$$

Conversely, from \leq we can define the irreflexive version $<$ by

$$a < b \Leftrightarrow a \leq b \text{ & } a \neq b.$$

Also we consider partial and total (or linear) order relations for which we have for all a, b

$$a \leq b \text{ or } b \leq a.$$

If nothing is said the order relation is total, while partial order relations are explicitly said to be partial.

PROOF₂. Viewing multisets as functions with finite support, define

$$\begin{aligned}\mathcal{F}_k &\triangleq \{f \in \mathbb{N}^{\mathbb{N}} \mid \forall n \geq k. f(n) = 0\}; \\ \mathcal{F} &\triangleq \bigcup_{k \in \mathbb{N}} \mathcal{F}_k.\end{aligned}$$

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

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

It is easy to see that $(\mathcal{F}, >)$ is a linear order. We will show that it is even a well-order, 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

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

element-wise. This will be proved by induction on k . If $k = 0$, then this is trivial, since $\mathcal{F}_0 = \{\lambda n. 0\}$. 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

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

where

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

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 . ■₂

2A.9. REMARK. The second proof shows in fact that if $(D, >)$ is a well-ordered set, then so is $(S(D), >)$, defined analogously to $(S(\mathbb{N}), >)$. In fact the argument can be carried out in Peano Arithmetic, showing

$$\vdash_{\mathbf{PA}} \text{TI}_\alpha \rightarrow \text{TI}_{\alpha^\omega},$$

where TI_α is the principle of transfinite induction for the ordinal α . Since TI_ω is in fact ordinary induction we have in PA (in an iterated exponentiation parenthesizing is to the right: for example $\omega^{\omega^\omega} = \omega^{(\omega^\omega)}$)

$$\text{TI}_\omega, \text{TI}_{\omega^\omega}, \text{TI}_{\omega^{\omega^\omega}}, \dots.$$

This implies that the proof of TI_α can be carried out in Peano Arithmetic for every $\alpha < \epsilon_0$. Gentzen [1936] shows that TI_{ϵ_0} , where

$$\epsilon_0 = \omega^{\omega^{\omega^{\omega^{\dots}}}},$$

cannot be carried out in PA.

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

2A.10. DEFINITION. (i) Let $R \equiv (\lambda x^A.M^B)^{A \rightarrow B} N^A$ be a redex. The *depth* of R , notation $\text{dpt}R$, is defined as

$$\text{dpt}(R) \triangleq \text{dpt}(A \rightarrow B),$$

where dpt on types is defined in Definition 1A.21.

(ii) To each M in $\lambda_{\rightarrow}^{\text{Ch}}$ we assign a multi-set S_M as follows

$$S_M \triangleq \{\text{dpt}(R) \mid R \text{ is a redex occurrence in } M\},$$

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.

2A.11. EXAMPLE. (i) Let R be a redex occurrence in a typed term M . Assume

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

i.e. N results from 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)

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

duplicates the other redex R_1 .

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

$$\begin{aligned} & (\lambda x^{A \rightarrow B}.M[x^{A \rightarrow B}P^A]^C)^{(A \rightarrow B) \rightarrow C}(\lambda y^A.Q^B) \rightarrow \beta M[(\lambda y^A.Q^B)^{A \rightarrow B}P^A]^C; \\ & (\lambda x^A.(\lambda y^B.M[x^A, y^B]^C)^{B \rightarrow C})^{A \rightarrow (B \rightarrow C)}P^A Q^B \rightarrow \beta (\lambda y^B.M[P^A, y^B]^C)^{B \rightarrow C}Q^B; \\ & (\lambda x^{A \rightarrow B}.x^{A \rightarrow B})^{(A \rightarrow B) \rightarrow (A \rightarrow B)}(\lambda y^A.P^B)^{A \rightarrow B}Q^A \rightarrow \beta (\lambda y^A.P^B)^{A \rightarrow B}Q^A. \end{aligned}$$

In Lévy [1978], 1.8.4., Lemme 3, it is proved (for the untyped λ -calculus) that the three ways of creating redexes in example 2A.11(ii) are the only possibilities. It is also given as Exercise 14.5.3 in B[1984].

2A.12. LEMMA. Assume $M \xrightarrow{R} \beta N$ and let R_1 be a created redex in N . Then

$$\text{dpt}(R) > \text{dpt}(R_1).$$

PROOF. In each of three cases we can inspect that the statement holds. ■

2A.13. THEOREM (Weak normalization theorem for $\lambda_{\rightarrow}^{\mathbb{A}}$). If $M \in \Lambda$ is typable in $\lambda_{\rightarrow}^{\mathbb{A}}$, then M is $\beta\eta$ -WN, i.e. has a $\beta\eta$ -nf. In short $\lambda_{\rightarrow}^{\mathbb{A}} \models \text{WN}$ (or more explicitly $\lambda_{\rightarrow}^{\mathbb{A}} \models \beta\eta\text{-WN}$).

PROOF. By Proposition 1B.26(ii) it suffices to show this for terms in $\lambda_{\rightarrow}^{\text{Ch}}$. Note that η -reductions decrease the length of a term; moreover, for β -normal terms η -contractions do not create β -redexes. Therefore in order to establish $\beta\eta$ -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 $F(M) \triangleq M$. Otherwise, let R be the *rightmost redex* of maximal depth n in M . A redex occurrence $(\lambda_1 x_1.P_1)Q_1$ is called *to the right* of an other one $(\lambda_2 x_2.P_2)Q_2$, if the occurrence of its λ , viz. λ_1 , is to the right of the other redex λ , viz. λ_2 .

Then

$$F(M) \triangleq N$$

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 2A.12 redexes created in $F(M)$ by the contraction $M \rightarrow \beta F(M)$ are also of depth $< n$. Therefore in case M is not in β -nf we have

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

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

$$M \rightarrow \beta F(M) \rightarrow \beta F^2(M) \rightarrow \beta F^3(M) \rightarrow \beta \cdots$$

must terminate in a β -nf. ■

2A.14. COROLLARY. *Let $A \in \mathbb{T}^{\mathbb{A}}$ and $M \in \Lambda_{\rightarrow}(A)$. Then M has a lnf.*

PROOF. Let $M \in \Lambda_{\rightarrow}(A)$. Then M has a β -nf by Theorem 2A.13, hence by Exercise 1E.14 also a lnf. ■

For β -reduction this weak normalization theorem was first proved by Turing, see Gandy [1980a]. The proof does not really need SN for \mathcal{S} -reduction, requiring transfinite induction up to ω^ω . The simpler result Lemma 2A.7, using induction up to ω^2 , suffices.

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

$$\begin{aligned} & (\lambda x^A.y^{A \rightarrow A \rightarrow A}x^A x^A)^{A \rightarrow A}((\lambda x^A.x^A)^{A \rightarrow A}x^A) \rightarrow \beta \\ & y^{A \rightarrow A \rightarrow A}((\lambda x^A.x^A)^{A \rightarrow A}x^A)((\lambda x^A.x^A)^{A \rightarrow A}x^A) \end{aligned}$$

give the multisets $\{1, 1\}$ and $\{1, 1\}$. Nevertheless, SN does hold for all systems $\lambda_{\rightarrow}^{\mathbb{A}}$, as will be proved in Section 2B. It is an open problem whether ordinals can be assigned in a natural and simple way to terms of $\lambda_{\rightarrow}^{\mathbb{A}}$ such that

$$M \rightarrow \beta N \Rightarrow \text{ord}(M) > \text{ord}(N).$$

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

Applications of normalization

We will show that β -normal terms inhabiting the represented data types (Bool, Nat, Σ^* and T^2) all are standard, i.e. correspond to the intended elements. From WN for $\lambda_{\rightarrow}^{\mathbb{A}}$ and the subject reduction theorem it then follows that all inhabitants of the mentioned data types are standard. The argumentation is given by a direct argument using basically the Generation Lemma. It can be streamlined, as will be done for Proposition 2A.18, by following the inhabitation machines, see Section 1C, for the types involved. For notational convenience we will work with $\lambda_{\rightarrow}^{\text{Cu}}$, but we could equivalently work with $\lambda_{\rightarrow}^{\text{Ch}}$ or $\lambda_{\rightarrow}^{\text{dB}}$, as is clear from Corollary 1B.25(iii) and Proposition 1B.32.

2A.15. PROPOSITION. *Let $\text{Bool} \equiv \text{Bool}_{\alpha}$, with α a type atom. Then for M in nf one has*

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

PROOF. By repeated use of Proposition 1B.21, the free variable Lemma 1B.2 and the generation Lemma for $\lambda_{\rightarrow}^{\text{Cu}}$, proposition 1B.3, one has the following.

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

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

2A.16. PROPOSITION. Let $\text{Nat} \equiv \text{Nat}_{\alpha} = (\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha$. Then for M in nf one has

$$\vdash M : \text{Nat} \Rightarrow M \in \{\mathbf{c}_n \mid n \in \mathbb{N}\}.$$

PROOF. Again we have

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

Now we have

$$\begin{aligned}f:\alpha \rightarrow \alpha, x:\alpha, \vdash M_2 : \alpha &\Rightarrow [M_2 \equiv x \vee \\ &\quad [M_2 \equiv fM_3 \ \& \ f:\alpha \rightarrow \alpha, x:\alpha \vdash M_3 : \alpha]].\end{aligned}$$

Therefore by induction on the structure of M_2 it follows that

$$f:\alpha \rightarrow \alpha, x:\alpha \vdash M_2 : \alpha \Rightarrow M_2 \equiv f^n x,$$

with $n \geq 0$. So $M \equiv \lambda fx.f^n x \equiv \mathbf{c}_n$. ■

2A.17. PROPOSITION. Let $\text{Sigma}^* \equiv \text{Sigma}_{\alpha}^*$. Then for M in nf one has

$$\vdash M : \text{Sigma}^* \Rightarrow M \in \{\underline{w} \mid w \in \Sigma^*\}.$$

PROOF. Again we have

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

A more streamlined proof will be given for the data type of trees T^2 .

2A.18. PROPOSITION. Let $\top \equiv \top_\alpha^2 \triangleq (\alpha \rightarrow \alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha$ and $M \in \Lambda_{\rightarrow}^\varnothing(\top^2)$.

- (i) If M is in lnf , then $M \equiv \underline{t}$, for some $t \in T^2$.
- (ii) Then $M = \beta\eta t$ for some tree $t \in T^2$.

PROOF. (i) For M in lnf use the inhabitation machine for \top^2 to show that $M \equiv \underline{t}$ for some $t \in T^2$.

(ii) For a general M there is by Corollary 2A.14 an M' in lnf such that $M = \beta\eta M'$. Then by (i) applied to M' we are done. ■

This proof raises the question what terms in β -nf are also in lnf , do Exercise 1E.15.

2B. Proofs of strong normalization

We now will give two proofs showing that $\lambda_{\rightarrow}^{\mathbb{A}} \models \text{SN}$. 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.

2B.1. THEOREM ([Strong normalization theorem](#) for $\lambda_{\rightarrow}^{\text{Ch}}$). For all $A \in \mathbb{T}^\infty$, $M \in \Lambda_{\rightarrow}^{\text{Ch}}(A)$ one has $\beta\eta\text{-SN}(M)$.

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

Writing SN for $\text{SN}_{\beta\eta}$ one first defines for $A \in \mathbb{T}^\infty$ the following class \mathcal{C}_A of [computable](#) terms of type A .

$$\begin{aligned}\mathcal{C}_\alpha &\triangleq \{M \in \Lambda_{\rightarrow}^{\text{Ch},\emptyset}(\alpha) \mid \text{SN}(M)\}; \\ \mathcal{C}_{A \rightarrow B} &\triangleq \{M \in \Lambda_{\rightarrow}^{\text{Ch},\emptyset}(A \rightarrow B) \mid \forall Q \in \mathcal{C}_A. M Q \in \mathcal{C}_B\}; \\ \mathcal{C} &\triangleq \bigcup_{A \in \mathbb{T}^\infty} \mathcal{C}_A.\end{aligned}$$

Then one defines the classes \mathcal{C}_A^* of terms that are [computable under substitution](#)

$$\mathcal{C}_A^* \triangleq \{M \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \mid \forall \vec{P} \in \mathcal{C}. [M[\vec{x}: = \vec{P}] \in \Lambda_{\rightarrow}^{\text{Ch},\emptyset}(A) \Rightarrow M[\vec{x}: = \vec{P}] \in \mathcal{C}_A]\}.$$

Write $\mathcal{C}^* \triangleq \bigcup \{\mathcal{C}_A^* \mid A \in \mathbb{T}^\infty\}$. For $A \equiv A_1 \rightarrow \dots \rightarrow A_n \rightarrow \alpha$ define

$$d_A \triangleq \lambda x_1^{A_1} \dots \lambda x_n^{A_n}. d_\alpha.$$

Then for A one has

$$M \in \mathcal{C}_A \Leftrightarrow \forall \vec{Q} \in \mathcal{C}. M \vec{Q} \in \text{SN}, \tag{0}$$

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

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

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

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

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

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

Now one shows by induction on M that

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

We distinguish cases and use (1).

Case $M \equiv x$. Then for $P, Q \in \mathcal{C}$ one has $M[x := P]Q \equiv PQ \in \mathcal{C} \subseteq \text{SN}$, 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

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

By the IH one has $N \in \mathcal{C}^* \subseteq \text{SN}$; therefore, if $\vec{P}, Q, \vec{R} \in \mathcal{C} \subseteq \text{SN}$, then

$$N[x := Q, \vec{y} := \vec{P}]Q\vec{R} \in \text{SN}. \quad (7)$$

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 $\lambda_{\rightarrow}^{\text{Ch+}}$, hence of $\lambda_{\rightarrow}^{\mathbb{A}}$, 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 $M\vec{P} \in \text{SN}$. For a typed system with only combinators this is sufficient and is covered by 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], for natural deduction, Girard [1971] for the second order typed lambda calculus $\lambda 2$, and Stenlund [1972] for λ_{\rightarrow} .

2B.2. COROLLARY (SN for $\lambda_{\rightarrow}^{\text{Cu}}$). $\forall A \in \text{TF}^{\infty} \forall M \in \Lambda_{\rightarrow}^{\text{Cu}, \Gamma}(A). \text{SN}_{\beta\eta}(M)$.

PROOF. Suppose $M \in \Lambda$ has type A with respect to Γ and has an infinite reduction path σ . By repeated use of Proposition 1B.26(ii) lift M to $M' \in \Lambda_{\rightarrow}^{\text{Ch}}$ with an infinite reduction path (that projects to σ), contradicting the Theorem. ■

An elementary proof of strong normalization

Now we present an elementary proof, due to Statman, of strong normalization of $\lambda_{\rightarrow}^{\mathbb{A}, \text{Ch}}$, where $\mathbb{A} = \{0\}$. Inspiration came from Nederpelt [1973], Gandy [1980b] and Klop [1980]. The point of this proof is that in this reduction system strong normalizability follows from normalizability by local structure arguments similar to and in many cases identical to those presented for the untyped lambda calculus in B[1984]. These include analysis of redex creation, permutability 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 use some results in the untyped lambda calculus

2B.3. DEFINITION. (i) Let $R \equiv (\lambda x.X)Y$ be a β -redex. Then R is

- (1) an **I-redex** if $x \in \text{FV}(X)$;
- (2) a **K-redex** if $x \notin \text{FV}(X)$;
- (3) a **\mathbf{K}^o -redex** if R is a K-redex and $x = x^0$ and $X \in \Lambda_{\rightarrow}^{\text{Ch}}(0)$;
- (4) a **\mathbf{K}^+ -redex** if R is a K-redex and is not a \mathbf{K}^o -redex.

(ii) A term M is said to have the λK^o -property, if every abstraction $\lambda x.X$ in M with $x \notin \text{FV}(X)$ satisfies $x = x^0$ and $X \in \Lambda_{\rightarrow}^{\text{Ch}}(0)$.

NOTATION. (i) $\rightarrow_{\beta l}$ is reduction of l -redexes.

(ii) $\rightarrow_{\beta lK^+}$ is reduction of l - or K^+ -redexes.

(iii) $\rightarrow_{\beta K^o}$ is reduction of K^o -redexes.

2B.4. THEOREM. Every $M \in \Lambda_{\rightarrow}^{\text{Ch}}$ is $\beta\eta$ -SN.

PROOF. The result is proved in several steps.

(i) Every term is $\beta\eta$ -normalizable and therefore has a hnf. This is Theorem 2A.13.

(ii) There are no β -reduction cycles. Consider a shortest term M at the beginning of a cyclic reduction. Then

$$M \rightarrow_{\beta} M_1 \rightarrow_{\beta} \cdots \rightarrow_{\beta} M_n \equiv M,$$

where, by minimality of M , at least one of the contracted redexes is a head-redex. Then M has an infinite quasi-head-reduction consisting of $\rightarrow_{\beta} \circ \rightarrow_h \circ \rightarrow_{\beta}$ steps. Therefore M has an infinite head-reduction, as internal (i.e. non-head) redexes can be postponed. (This is Exercise 13.6.13 [use Lemma 11.4.5] in B[1984].) This contradicts (i), using B[1984], Corollary 11.4.8 to the standardization Theorem.

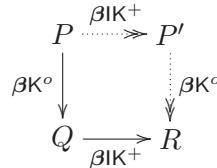
(iii) $M \rightarrow_{\eta} N \rightarrow_{\beta}^+ L \Rightarrow \exists P.M \rightarrow_{\beta}^+ P \rightarrow_{\eta} N$. This is a strengthening of η -postponement, B[1984] Corollary 15.1.6, and can be proved in the same way.

(iv) $\beta\text{-SN} \Rightarrow \beta\eta\text{-SN}$. Take an infinite $\rightarrow_{\beta\eta}$ sequence. Make a diagram with β -steps drawn horizontally and η -steps vertically. These vertical steps are finite, as $\eta \models \text{SN}$. Apply (iii) at each $\rightarrow_{\eta} \circ \rightarrow_{\beta}^+$ -step. The result yields a horizontal infinite \rightarrow_{β} sequence.

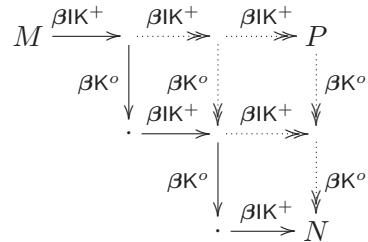
(v) We have $\lambda_{\rightarrow}^{\mathbb{A}} \models \beta l\text{-WN}$. By (i).

(vi) $\lambda_{\rightarrow}^{\mathbb{A}} \models \beta l\text{-SN}$. By Church's result in B[1984], Conservation Theorem for λl , 11.3.4.

(vii) $M \rightarrow_{\beta} N \Rightarrow \exists P.M \rightarrow_{\beta lK^+} P \rightarrow_{\beta K^o} N$ (βK^o -postponement). When contracting a K^o redex, no redex can be created. Realizing this, one has



From this the statement follows by a simple diagram chase, that w.l.o.g. looks like



(viii) Suppose M has the λK^o -property. Then M β -reduces to only finitely many N . First observe that $M \rightarrow_{\beta lK^+} N \Rightarrow M \rightarrow_{\beta l} N$, as a contraction of an l -redex cannot create a K^+ -redex. (But a contraction of a K redex can create a K^+ redex.) Hence by

(vi) the set $\mathcal{X} = \{P \mid M \rightarrow_{\beta\text{IK}^+} P\}$ is finite. Since K -redexes shorten terms, also the set of K^o -reducts of elements of \mathcal{X} form a finite set. Therefore by (vii) we are done.

(ix) *If M has the λK^o -property, then $M \models \beta\text{-SN}$.* By (viii) and (ii).

(x) *If M has the λK^o -property, then $M \models \beta\eta\text{-SN}$.* By (iv) and (ix).

(xi) *For each M there is an N with the λK^o -property such that $N \rightarrow_{\beta\eta} M$.* Let $R \equiv \lambda x^A.P^B$ a subterm of M , making it fail to be a term with the λK^o -property. Write $A = A_1 \rightarrow \dots \rightarrow A_a \rightarrow 0$, $B = B_1 \rightarrow \dots \rightarrow B_b \rightarrow 0$. Then replace mentioned subterm by

$$R' \equiv \lambda x^A \lambda y_1^{B_1} \dots y_b^{B_b} . (\lambda z^0 . (Py_1^{B_1} \dots y_b^{B_b})) (x^A u_1^{A_1} \dots u_a^{A_a}),$$

which $\beta\eta$ -reduces to R , but does not violate the λK^o -property. That R' contains the free variables \vec{u} does not matter. Treating each such subterm this way, N is obtained.

(xii) $\lambda_{\rightarrow}^{\mathbb{A}} \models \beta\eta\text{-SN}$. By (x) and (xi). ■

Other proofs of SN from WN are in [de Vrijer \[1987\]](#), [Kfoury and Wells \[1995\]](#), [Sørensen \[1997\]](#), and [Xi \[1997\]](#). In the proof of de Vrijer a computation is given of the longest reduction path to β -nf for a typed term M .

2C. Checking and finding types

There are several natural problems concerning type systems.

2C.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 *typability* consists of determining for a given term M 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.

$$\begin{array}{lll} \Gamma \vdash_{\lambda_{\rightarrow}} M : A ? & \text{type checking;} \\ \exists A, \Gamma [\Gamma \vdash_{\lambda_{\rightarrow}} M : A] ? & \text{typability;} \\ ? \vdash_{\lambda_{\rightarrow}} M : ? & \text{type reconstruction;} \\ \exists M [\Gamma \vdash_{\lambda_{\rightarrow}} M : A] ? & \text{inhabitation;} \\ \Gamma \vdash_{\lambda_{\rightarrow}} ? : A & \text{enumeration.} \end{array}$$

In another notation this is the following.

$$\begin{array}{lll} M \in \Lambda_{\rightarrow}^{\Gamma}(A) ? & \text{type checking;} \\ \exists A, \Gamma M \in \Lambda_{\rightarrow}^{\Gamma}(A) ? & \text{typability;} \\ M \in \Lambda_{\rightarrow}^{\text{?}}(\text{?}) & \text{type reconstruction;} \\ \Lambda_{\rightarrow}^{\Gamma}(A) \neq \emptyset ? & \text{inhabitation;} \\ ? \in \Lambda_{\rightarrow}^{\Gamma}(A) & \text{enumeration.} \end{array}$$

In this section we will treat the problems of type checking, typability and type reconstruction for the three versions of λ_{\rightarrow} . It turns out that these problems are decidable for all versions. The solutions are essentially simpler for $\lambda_{\rightarrow}^{\text{Ch}}$ and $\lambda_{\rightarrow}^{\text{dB}}$ than for $\lambda_{\rightarrow}^{\text{Cu}}$. 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

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

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

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

Therefore

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

On the other hand the question

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

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 $\lambda_{\rightarrow}^{\text{dB}}$ and $\lambda_{\rightarrow}^{\text{Ch}}$

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

2C.2. PROPOSITION (Type checking for $\lambda_{\rightarrow}^{\text{dB}}$). *Let Γ be a basis of $\lambda_{\rightarrow}^{\text{dB}}$. Then there is a computable function $\text{type}_{\Gamma} : \Lambda_{\rightarrow}^{\text{dB}} \rightarrow \mathbb{T} \cup \{\text{error}\}$ such that*

$$M \in \Lambda_{\rightarrow}^{\text{dB}, \Gamma}(A) \Leftrightarrow \text{type}_{\Gamma}(M) = A.$$

PROOF. Define

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

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

2C.3. COROLLARY. *Typability and type reconstruction for $\lambda_{\rightarrow}^{\text{dB}}$ are computable. In fact one has the following.*

- (i) $M \in \Lambda_{\rightarrow}^{\text{dB}, \Gamma} \Leftrightarrow \text{type}_{\Gamma}(M) \neq \text{error}.$
- (ii) *Each $M \in \Lambda_{\rightarrow}^{\text{dB}, \Gamma}(\text{type}_{\Gamma})$ has a unique type; in particular*

$$M \in \Lambda_{\rightarrow}^{\text{dB}, \Gamma}(\text{type}_{\Gamma}(M)).$$

PROOF. By the proposition. ■

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

2C.4. PROPOSITION (Type checking for $\lambda_{\rightarrow}^{\text{Ch}}$). *There is a computable function type : $\Lambda_{\rightarrow}^{\text{Ch}} \rightarrow \mathbb{T}$ such that*

$$M \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \Leftrightarrow \text{type}(M) = A.$$

PROOF. Define

$$\begin{aligned} \text{type}(x^A) &\triangleq A; \\ \text{type}(MN) &\triangleq B, \quad \text{if } \text{type}(M) = \text{type}(N) \rightarrow B, \\ \text{type}(\lambda x^A.M) &\triangleq A \rightarrow \text{type}(M). \end{aligned}$$

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

2C.5. COROLLARY. *Typability and type reconstruction for $\lambda_{\rightarrow}^{\text{Ch}}$ are computable. In fact one has the following. Each $M \in \Lambda_{\rightarrow}^{\text{Ch}}$ has a unique type; in particular $M \in \Lambda_{\rightarrow}^{\text{Ch}}(\text{type}(M))$.*

PROOF. By the proposition. ■

Checking and finding types for $\lambda_{\rightarrow}^{\text{Cu}}$

We now will show the computability of the three questions for $\lambda_{\rightarrow}^{\text{Cu}}$. This occupies 2C.6 - 2C.16 and in these items \vdash stands for $\vdash_{\lambda_{\rightarrow}^{\text{Cu}}}$ over a general $\mathbb{T}^{\mathbb{A}}$.

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

2C.6. DEFINITION. (i) A *substitutor* is an operation $*$: $\mathbb{T} \rightarrow \mathbb{T}$ such that

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

(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

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

In that case we write

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

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

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

and

$$* = []$$

for the identity substitution.

2C.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 $E = \{A_1 = B_1, \dots, A_n = B_n\}$ 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 $A_1^* \equiv B_1^* \& \dots \& A_n^* \equiv B_n^*$. In that case one writes $* \models E$. Similarly one defines the notion of a most general unifier for E .

2C.8. EXAMPLES. The types $\beta \rightarrow (\alpha \rightarrow \beta)$ and $(\gamma \rightarrow \gamma) \rightarrow \delta$ have a unifier. For example $* = [\beta := \gamma \rightarrow \gamma, \delta := \alpha \rightarrow (\gamma \rightarrow \gamma)]$ or $*_1 = [\beta := \gamma \rightarrow \gamma, \alpha := \varepsilon \rightarrow \varepsilon, \delta := \varepsilon \rightarrow \varepsilon \rightarrow (\gamma \rightarrow \gamma)]$. The unifier $*$ is most general, $*_1$ is not.

2C.9. DEFINITION. A is a *variant* of B if for some $*_1$ and $*_2$ one has

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

2C.10. EXAMPLE. $\alpha \rightarrow \beta \rightarrow \beta$ is a variant of $\gamma \rightarrow \delta \rightarrow \delta$ but not of $\alpha \rightarrow \beta \rightarrow \alpha$.

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 (in the first-order⁷ case) *unification* is decidable.

2C.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*

$$\begin{aligned} A \text{ and } B \text{ have a unifier } &\Rightarrow U(A, B) \text{ is a most general unifier} \\ &\quad \text{for } A \text{ and } B; \end{aligned}$$

$$A \text{ and } B \text{ have no unifier } \Rightarrow U(A, B) = \text{fail}.$$

(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*

$$E \text{ has a unifier } \Rightarrow U(E) \text{ is a most general unifier for } E;$$

$$E \text{ has no unifier } \Rightarrow U(E) = \text{fail}.$$

PROOF. Note that $A_1 \rightarrow A_2 \equiv B_1 \rightarrow B_2$ 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.

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

$$U(A_1 \rightarrow A_2, \alpha) = U(\alpha, A_1 \rightarrow A_2);$$

$$U(A_1 \rightarrow A_2, B_1 \rightarrow B_2) = U(A_1^{U(A_2, B_2)}, B_1^{U(A_2, B_2)}) \circ U(A_2, B_2),$$

where this last expression is considered to be **fail** if one of its parts is. Let

$$\#_{var}(A, B) = \text{'the number of variables in } A \rightarrow B',$$

$$\#\rightarrow(A, B) = \text{'the number of arrows in } A \rightarrow B'.$$

By induction on $(\#_{var}(A, B), \#\rightarrow(A, B))$ ordered lexicographically one can show that $U(A, B)$ is always defined. Moreover U satisfies the specification.

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

⁷That is, for the algebraic signature $\langle \mathbb{T}, \rightarrow \rangle$. Higher-order unification is undecidable, see Section 4B.

See Baader and Nipkow [1998] and Baader and Snyder [2001] 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 typability for λ_{\rightarrow} .

2C.12. PROPOSITION. *For every basis Γ , term $M \in \Lambda$ and $A \in \mathbb{T}$ such that $\text{FV}(M) \subseteq \text{dom}(\Gamma)$ there is a finite set of equations $E = E(\Gamma, M, A)$ such that for all substitutors $*$ one has*

$$* \models E(\Gamma, M, A) \Rightarrow \Gamma^* \vdash M : A^*, \quad (1)$$

$$\Gamma^* \vdash M : A^* \Rightarrow *_1 \models E(\Gamma, M, A), \quad (2)$$

*for some $*_1$ such that $*$ and $*_1$ have the same effect on the type variables in Γ and A .*

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

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

By induction on M one can show (using the generation Lemma (1B.3)) that (1) and (2) hold. ■

2C.13. DEFINITION. (i) Let $M \in \Lambda$. Then (Γ, A) is a *principal pair* for M , notation $\text{pp}(M)$, if

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

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

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

(ii) Let $M \in \Lambda$ be closed. Then A is a *principal type*, notation $\text{pt}(M)$, 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 $\text{FV}(M) = \text{dom}(\Gamma)$.

The following result is independently due to Curry [1969], Hindley [1969], and Milner [1978]. It shows that for λ_{\rightarrow} the problems of type checking and typability are decidable. One usually refers to it as the '*Hindley-Milner algorithm*'.

2C.14. THEOREM (*Principal type theorem* for $\lambda_{\rightarrow}^{\text{Cu}}$). (i) *There exists a computable function pp such that one has*

$$M \text{ has a type} \Rightarrow \text{pp}(M) = (\Gamma, A), \text{ where } (\Gamma, A) \text{ is a pp for } M;$$

$$M \text{ has no type} \Rightarrow \text{pp}(M) = \text{fail}.$$

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

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

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

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

Define

$$\begin{aligned} pp(M) &\triangleq (\Gamma_0^*, A_0^*), & \text{if } U(E(\Gamma_0, M, A_0)) = *; \\ &\triangleq \text{fail}, & \text{if } U(E(\Gamma_0, M, A_0)) = \text{fail}. \end{aligned}$$

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 2C.12. To show that (Γ_0^*, A_0^*) is a pp, suppose that also $\Gamma' \vdash M : A'$. Let $\tilde{\Gamma} = \Gamma' \upharpoonright \text{FV}(M)$; write $\tilde{\Gamma} = \Gamma_0^{*0}$ and $A' = A_0^{*0}$. Then also $\Gamma_0^{*0} \vdash M : A_0^{*0}$. Hence by (2) in proposition 2C.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 2C.11) one has $*_1 = *_2 \circ *$ for some $*_2$. Now indeed

$$(\Gamma_0^*)^{*2} = \Gamma_0^{*1} = \Gamma_0^{*0} = \tilde{\Gamma} \subseteq \Gamma'$$

and

$$(A_0^*)^{*2} = A_0^{*1} = A_0^{*0} = A'.$$

If M has no type, then $\neg \exists * \quad * \models E(\Gamma_0, M, A_0)$ hence

$$U(\Gamma_0, M, A_0) = \text{fail} = pp(M).$$

(ii) Let M be closed and $pp(M) = (\Gamma, A)$. Then $\Gamma = \emptyset$ and we can put $pt(M) = A$. ■

2C.15. COROLLARY. Type checking and typability for $\lambda_{\rightarrow}^{\text{Cu}}$ are decidable.

PROOF. As to type checking, let M and A be given. Then

$$\vdash M : A \Leftrightarrow \exists * [A = pt(M)^*].$$

This is decidable (as can be seen using an algorithm—*pattern matching*—similar to the one in Theorem 2C.11).

As to typability, let M be given. Then M has a type iff $pt(M) \neq \text{fail}$. ■

The following result is due to Hindley [1969] and Hindley [1997], Thm. 7A2.

2C.16. THEOREM (*Second principal type theorem* for $\lambda_{\rightarrow}^{\text{Cu}}$). (i) For every $A \in \mathbb{T}$ one has

$$\vdash M : A \Rightarrow \exists M' [M' \rightarrow_{\beta} M \& \text{pt}(M') = A].$$

(ii) For every $A \in \mathbb{T}$ there exists a basis Γ and $M \in \Lambda$ 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 2E.5 for the general proof.

Case $M \equiv \lambda x.x$ and $A \equiv (\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta$. Then $\text{pt}(M) \equiv \alpha \rightarrow \alpha$. Take $M' \equiv \lambda x y. x y$. The η -expansion of $\lambda x.x$ to $\lambda x y. x y$ makes subtypes of A correspond to unique subterms of M' .

Case $M \equiv \lambda x y. y$ and $A \equiv (\alpha \rightarrow \gamma) \rightarrow \beta \rightarrow \beta$. Then $\text{pt}(M) \equiv \alpha \rightarrow \beta \rightarrow \beta$. Take $M' \equiv \lambda x y. K y(\lambda z. x z)$. The β -expansion forces x to have a functional type.

Case $M \equiv \lambda x y. x$ and $A \equiv \alpha \rightarrow \alpha \rightarrow \alpha$. Then $\text{pt}(M) \equiv \alpha \rightarrow \beta \rightarrow \alpha$. Take $M' \equiv \lambda x y. K x(\lambda f. [f x, f y])$. The β -expansion forces x and y to have the same types.

(ii) Let A be given. We know that $\vdash I : A \rightarrow A$. Therefore by (i) there exists an $I' \xrightarrow{\beta\eta} I$ such that $\text{pt}(I') = A \rightarrow A$. Then take $M \equiv I' x$. We have $\text{pp}(I' x) = (\{x:A\}, A)$. ■ It is an open problem whether the result also holds in the λI -calculus.

Complexity

A closer look at the proof of Theorem 2C.14 reveals that the typability and type-checking problems (understood as yes or no decision problems) reduce to solving first-order unification, a problem known to be solvable in polynomial time, see Baader and Nipkow [1998]. Since the reduction is also polynomial, we conclude that typability and type-checking are solvable in polynomial time as well.

However, the actual type reconstruction may require exponential space (and thus also exponential time), just to write down the result. Indeed, Exercise 2E.21 demonstrates that the length of a shortest type of a given term may be exponential in the length of the term. The explanation of the apparent inconsistency between the two results is this: long types can be represented by small graphs.

In order to decide whether for two typed terms $M, N \in \Lambda_{\rightarrow}(A)$ 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 2E.20 it will be shown that the time and space costs of solving this conversion problem is hyper-exponential (in the sum of the sizes of M, N). The reason is that there are short terms having very long normal forms. For instance, the type-free application of Church numerals

$$\mathbf{c}_n \mathbf{c}_m = \mathbf{c}_{m^n}$$

can be typed, even when applied iteratively

$$\mathbf{c}_{n_1} \mathbf{c}_{n_2} \cdots \mathbf{c}_{n_k}.$$

In Exercise 2E.19 it is shown that the costs of this typability problem are also at most hyper-exponential. The reason is that Turing's proof of normalization for terms in λ_{\rightarrow} uses a successive 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 $M, N \in \Lambda_{\rightarrow}^{\text{Cu}}(A)$, as the cost of finding types only adds 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 convertibility problem for terms of some type in λ_{\rightarrow}^0 . A shorter proof of this result can be found in [Mairson \[1992\]](#).

2D. Checking inhabitation

In this section we study for $\lambda_{\rightarrow}^{\mathbb{A}}$ the problem of inhabitation. In Section 1C we wanted to enumerate all possible normal terms in a given type A . Now we study mere existence of a term M such that in the empty context $\vdash_{\lambda_{\rightarrow}^{\mathbb{A}}} M : A$. By Corollaries 1B.20 and 1B.33 it does not matter whether we work in the system à la Curry, Church or de Bruijn. Therefore we will focus on $\lambda_{\rightarrow}^{C_u}$. Note that by Proposition 1B.2 the term M must be closed. From the normalization theorem 2A.13 it follows that we may limit ourselves to find a term M in β -nf.

For example, if $A = \alpha \rightarrow \alpha$, 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 $A = \alpha \rightarrow \alpha \rightarrow \alpha$ there are two inhabitants: $M_1 \equiv \lambda x_1 x_2.x_1 \equiv K$ and $M_2 \equiv \lambda x_1 x_2.x_2 \equiv K_*$. 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 \rightarrow as the two-ary function on \mathbf{B} defined by

$$x \rightarrow y = 1 - x + xy$$

(classical implication). This makes every type A into a Boolean function. More formally this is done as follows.

2D.1. DEFINITION. (i) A *Boolean valuation* is a map $\rho : \mathbb{A} \rightarrow \mathbf{B}$.

(ii) Let ρ be a Boolean valuation. The *Boolean interpretation under ρ* of a type $A \in \mathbb{T}$, notation $\llbracket A \rrbracket_{\rho}$, is defined inductively as follows.

$$\begin{aligned} \llbracket \alpha \rrbracket_{\rho} &\triangleq \rho(\alpha); \\ \llbracket A_1 \rightarrow A_2 \rrbracket_{\rho} &\triangleq \llbracket A_1 \rrbracket_{\rho} \rightarrow \llbracket A_2 \rrbracket_{\rho}. \end{aligned}$$

(iii) A Boolean valuation ρ *satisfies a type A* , notation $\rho \models A$, if $\llbracket A \rrbracket_{\rho} = 1$. Let $\Gamma = \{x_1 : A_1, \dots, x_n : A_n\}$, then ρ *satisfies Γ* , notation $\rho \models \Gamma$, if

$$\rho \models A_1 \& \dots \& \rho \models A_n.$$

(iv) A type A is *classically valid*, notation $\models A$, iff for all Boolean valuations ρ one has $\rho \models A$.

2D.2. PROPOSITION. *Let $\Gamma \vdash_{\lambda_{\rightarrow}^{\mathbb{A}}} M : A$. Then for all Boolean valuations ρ one has*

$$\rho \models \Gamma \Rightarrow \rho \models A.$$

PROOF. By induction on the derivation in $\lambda_{\rightarrow}^{\mathbb{A}}$. ■

From this it follows that inhabited types are classically valid. This in turn implies that the type α is not inhabited.

2D.3. COROLLARY. (i) *If A is inhabited, then $\models A$.*

(ii) *A type variable α is not inhabited.*

PROOF. (i) Immediate by Proposition 2D.2, by taking $\Gamma = \emptyset$.

(ii) Immediate by (i), by taking $\rho(\alpha) = 0$. ■

One may wonder whether the converse of 2D.3(i), i.e.

$$\models A \Rightarrow A \text{ is inhabited} \quad (1)$$

holds. We will see that in $\lambda_{\rightarrow}^{\mathbb{A}}$ this is not the case. For λ_{\rightarrow}^0 (having only one base type 0), however, the implication (1) is valid.

2D.4. PROPOSITION (Statman [1982]). *Let $A = A_1 \rightarrow \dots \rightarrow A_n \rightarrow 0$, with $n \geq 1$ be a type of λ_{\rightarrow}^0 . Then*

$$\begin{aligned} A \text{ is inhabited} &\Leftrightarrow \text{for some } i \text{ with } 1 \leq i \leq n \text{ the type} \\ &A_i \text{ is not inhabited.} \end{aligned}$$

PROOF. (\Rightarrow) Assume $\vdash_{\lambda_{\rightarrow}^0} M : A$. Suppose towards a contradiction that all A_i are inhabited, i.e. $\vdash_{\lambda_{\rightarrow}^0} N_i : A_i$. Then $\vdash_{\lambda_{\rightarrow}^0} MN_1 \dots N_n : 0$, contradicting 2D.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. $A_i = 0$. Then

$$x_1 : A_1, \dots, x_n : A_n \vdash x_i : 0$$

so

$$\vdash (\lambda x_1 \dots x_n. x_i) : A_1 \rightarrow \dots \rightarrow A_n \rightarrow 0,$$

i.e. A is inhabited.

Case 2. $A_i = B_1 \rightarrow \dots \rightarrow B_m \rightarrow 0$. 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

$$\begin{aligned} x_1 : A_1, \dots, x_n : A_n \vdash x_i : A_i &= B_1 \rightarrow \dots \rightarrow B_m \rightarrow 0 \\ \Rightarrow x_1 : A_1, \dots, x_n : A_n \vdash x_i M_1 \dots M_m : 0 \\ \Rightarrow \vdash \lambda x_1 \dots x_n. x_i M_1 \dots M_m : A_1 \rightarrow \dots \rightarrow A_n \rightarrow 0 &= A. \blacksquare \end{aligned}$$

From the proposition it easily follows that inhabitation of types in λ_{\rightarrow}^0 is decidable with a linear time algorithm.

2D.5. COROLLARY. *In λ_{\rightarrow}^0 one has for all types A*

$$A \text{ is inhabited} \Leftrightarrow \models A.$$

PROOF. (\Rightarrow) By Proposition 2D.3(i). (\Leftarrow) Assume $\models A$ and that A is not inhabited. Then $A = A_1 \rightarrow \dots \rightarrow A_n \rightarrow 0$ with each A_i inhabited. But then for $\rho_0(0) = 0$ one has

$$\begin{aligned} 1 &= \llbracket A \rrbracket_{\rho_0} \\ &= \llbracket A_1 \rrbracket_{\rho_0} \rightarrow \dots \rightarrow \llbracket A_n \rrbracket_{\rho_0} \rightarrow 0 \\ &= 1 \rightarrow \dots \rightarrow 1 \rightarrow 0, \text{ since } \models A_i \text{ for all } i, \\ &= 0, \text{ since } 1 \rightarrow 0 = 0, \end{aligned}$$

contradiction. ■

Corollary 2D.5 does not hold for $\lambda \rightarrow^\infty$. In fact the type $((\alpha \rightarrow \beta) \rightarrow \alpha) \rightarrow \alpha$ (corresponding to Peirce's law) is a valid type that is not inhabited, as we will see soon.

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 become clear in more detail in Sections 6C, 6D. The book Sørensen and Urzyczyn [2006] is devoted to this correspondence.

2D.6. DEFINITION (Implicational propositional logic). (i) The set of formulas of the *implicational propositional logic*, notation $\text{form}(\text{PROP})$, is defined by the following simplified syntax. Define $\text{form} = \text{form}(\text{PROP})$ as follows.

$$\begin{array}{lcl} \text{form} & ::= & \text{var} \mid \text{form} \supset \text{form} \\ \text{var} & ::= & p \mid \text{var}' \end{array}$$

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 $\Gamma \vdash_{\text{PROP}} A$, if $\Gamma \vdash A$ can be produced by the following formal system.

$$\begin{array}{c} A \in \Gamma \Rightarrow \Gamma \vdash A \\ \Gamma \vdash A \supset B, \Gamma \vdash A \Rightarrow \Gamma \vdash B \\ \Gamma, A \vdash B \Rightarrow \Gamma \vdash A \supset B \end{array}$$

NOTATION. (i) q, r, s, t, \dots stand for arbitrary propositional variables.

(ii) As usual $\Gamma \vdash A$ stands for $\Gamma \vdash_{\text{PROP}} A$ if there is little danger for confusion. Moreover, $\vdash A$ stands for $\emptyset \vdash A$.

2D.7. EXAMPLE. (i) $\vdash A \supset A$;
(ii) $A \vdash B \supset A$;
(iii) $\vdash A \supset (B \supset A)$;
(iv) $A \supset (A \supset B) \vdash A \supset B$.

2D.8. DEFINITION. Let $A \in \text{form}(\text{PROP})$ and $\Gamma \subseteq \text{form}(\text{PROP})$.

(i) Define $[A] \in \mathbb{T}^\infty$ and $\Gamma_A \subseteq \mathbb{T}^\infty$ as follows.

A	$[A]$	Γ_A
p	p	\emptyset
$P \supset Q$	$[P] \rightarrow [Q]$	$\Gamma_P \cup \Gamma_Q$

It so happens that $\Gamma_A = \emptyset$ and $[A]$ is A with the \supset replaced by \rightarrow . But the setup will be needed for more complex logics and type theories.

(ii) Moreover, we set $[\Gamma] = \{x_A : A \mid A \in \Gamma\}$.

2D.9. PROPOSITION. Let $A \in \text{form}(\text{PROP})$ and $\Delta \subseteq \text{form}(\text{PROP})$. Then

$$\Delta \vdash_{\text{PROP}} A \Rightarrow [\Delta] \vdash_{\lambda \rightarrow} M : [A], \text{ for some } M.$$

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 $[\Delta] \vdash P : [B] \rightarrow [A]$ 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 $[\Delta] \vdash (\lambda x_B. M) : [B] \rightarrow [C] \equiv [B \supset C] \equiv [A]$. ■

Conversely we have the following.

2D.10. PROPOSITION. *Let $\Delta, A \subseteq \text{form}(\text{PROP})$. Then*

$$[\Delta] \vdash_{\lambda\rightarrow} M : [A] \Rightarrow \Delta \vdash_{\text{PROP}} A.$$

PROOF. By induction on the structure of M .

Case 1. $M \equiv x$. Then by the generation Lemma 1B.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 $C \in \mathbb{T}$ one has $[\Delta] \vdash P : C \rightarrow [A]$ and $[\Delta] \vdash Q : C$. Clearly, for some $C' \in \text{form}$ one has $C \equiv [C']$. Then $C \rightarrow [A] \equiv [C' \supset A]$. By the induction hypothesis one has $\Delta \vdash C' \rightarrow A$ 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 $[A] \equiv B \rightarrow C$ 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 $\Delta \vdash B \rightarrow C \equiv A$. ■

Although intuitionistic logic gives a complete characterization of those types that are inhabited, this does not answer immediately the question whether the type $((\alpha \rightarrow \beta) \rightarrow \alpha) \rightarrow \alpha$ corresponding to Peirce's law is inhabited.

Kripke models

Remember that a type $A \in \mathbb{T}$ is inhabited iff it is the translation of a $B \in \text{form}(\text{PROP})$ that is intuitionistically provable. This explains why

$$A \text{ inhabited } \Rightarrow \models A,$$

but not conversely, since $\models A$ corresponds to classical validity. A common tool to prove that types are not inhabited or that formulas are not intuitionistically derivable consists of the notion of *Kripke model*, that we will introduce now.

2D.11. DEFINITION. (i) A *Kripke model* is a tuple $\mathcal{K} = \langle K, \leq, \odot, F \rangle$, such that

(1) $\langle K, \leq, \odot \rangle$ is a partially ordered set with least element \odot ;

(2) $F : K \rightarrow \wp(\text{var})$ is a monotonic map from K to the powerset of the set of type-variables; that is $\forall k, k' \in K [k \leq k' \Rightarrow F(k) \subseteq F(k')]$.

We often just write $\mathcal{K} = \langle K, F \rangle$.

(ii) Let $\mathcal{K} = \langle K, F \rangle$ be a Kripke model. For $k \in K$ define by induction on the structure of $A \in \mathbb{T}$ the notion k forces A , notation $k \Vdash_{\mathcal{K}} A$. We often omit the subscript.

$$\begin{aligned} k \Vdash \alpha &\Leftrightarrow \alpha \in F(k); \\ k \Vdash A_1 \rightarrow A_2 &\Leftrightarrow \forall k' \geq k [k' \Vdash A_1 \Rightarrow k' \Vdash A_2]. \end{aligned}$$

(iii) \mathcal{K} forces A , notation $\mathcal{K} \Vdash A$, is defined as $\odot \Vdash_{\mathcal{K}} A$.

- (iv) Let $\Gamma = \{x_1:A_1, \dots, x_n:A_n\}$. Then \mathcal{K} forces Γ , notation $\mathcal{K} \Vdash \Gamma$, if
- $$\mathcal{K} \Vdash A_1 \& \dots \& \mathcal{K} \Vdash A_n.$$

We say Γ forces A , notation $\Gamma \Vdash A$, iff for all Kripke models \mathcal{K} one has

$$\mathcal{K} \Vdash \Gamma \Rightarrow \mathcal{K} \Vdash A.$$

In particular forced A , notation $\Vdash A$, if $\mathcal{K} \Vdash A$ for all Kripke models \mathcal{K} .

2D.12. LEMMA. Let \mathcal{K} be a Kripke model. Then for all $A \in \mathbb{T}$ one has

$$k \leq k' \& k \Vdash_{\mathcal{K}} A \Rightarrow k' \Vdash_{\mathcal{K}} A.$$

PROOF. By induction on the structure of A . ■

2D.13. PROPOSITION. $\Gamma \vdash_{\lambda \rightarrow} M : A \Rightarrow \Gamma \Vdash A$.

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 $\Gamma \vdash F : B \rightarrow A$ and $\Gamma \vdash P : B$, then the conclusion follows from the induction hypothesis and the fact that $k \Vdash B \rightarrow A \& k \Vdash B \Rightarrow k \Vdash A$. In the case that $\Gamma \vdash M : A$ is $\Gamma \vdash \lambda x.N : A_1 \rightarrow A_2$ 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}

$$\mathcal{K} \Vdash \Gamma, A_1 \Rightarrow \mathcal{K} \Vdash A_2. \quad (2)$$

We must show $\Gamma \Vdash A_1 \rightarrow A_2$, i.e. $\mathcal{K} \Vdash \Gamma \Rightarrow \mathcal{K} \Vdash A_1 \rightarrow A_2$ for all \mathcal{K} .

Given \mathcal{K} and $k \in K$, define

$$\mathcal{K}_k \triangleq \langle \{k' \in K \mid k \leq k'\}, \leq, k, F \rangle,$$

(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

$$k \Vdash_{\mathcal{K}} A \Leftrightarrow \mathcal{K}_k \Vdash A. \quad (3)$$

Now suppose $\mathcal{K} \Vdash \Gamma$ in order to show $\mathcal{K} \Vdash A_1 \rightarrow A_2$, i.e. for all $k \in K$

$$k \Vdash_{\mathcal{K}} A_1 \Rightarrow k \Vdash_{\mathcal{K}} A_2.$$

Indeed,

$$\begin{aligned} k \Vdash_{\mathcal{K}} A_1 &\Rightarrow \mathcal{K}_k \Vdash A_1, && \text{by (3)} \\ &\Rightarrow \mathcal{K}_k \Vdash A_2, && \text{by (2), since by Lemma 2D.12 also } \mathcal{K}_k \Vdash \Gamma, \\ &\Rightarrow k \Vdash_{\mathcal{K}} A_2. \blacksquare \end{aligned}$$

2D.14. COROLLARY. Let $A \in \mathbb{T}$. Then

$$A \text{ is inhabited} \Rightarrow \Vdash A.$$

PROOF. Take $\Gamma = \emptyset$. ■

Now it can be proved, see exercise 2E.8, that (the type corresponding to) Peirce's law $P = ((\alpha \rightarrow \beta) \rightarrow \alpha) \rightarrow \alpha$ is not forced in some Kripke model. Since $\not\models P$ it follows that P is not inhabited, in spite of the fact that $\models P$.

We also have a converse to corollary 2D.14 which theoretically answers the inhabitation question for $\lambda \rightarrow^{\mathbb{A}}$.

2D.15. REMARK. [Completeness for Kripke models]

(i) The usual formulation is for provability in intuitionistic logic:

$$A \text{ is inhabited} \Leftrightarrow \Vdash A.$$

The proof is given by constructing for a type that is not inhabited a Kripke ‘counter-model’ \mathcal{K} , i.e. $\mathcal{K} \not\models A$, see [Kripke \[1965\]](#).

(ii) In [Harrop \[1958\]](#) it is shown that these Kripke counter-models can be taken to be finite. This solves the decision problem for inhabituation in $\lambda_{\rightarrow}^{\infty}$.

(iii) In [Statman \[1979a\]](#) 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 exist terms satisfying certain properties. For example making it possible to take as product $A \times A$ just the type A itself.

2D.16. DEFINITION. Let $A \in \mathbb{T}^{\mathbb{A}}$. An $A \times A \rightarrow A$ *pairing* is a triple $\langle \text{pair}, \text{left}, \text{right} \rangle$ such that

$$\begin{aligned} \text{pair} &\in \Lambda_{\rightarrow}^{\emptyset}(A \rightarrow A \rightarrow A); \\ \text{left}, \text{right} &\in \Lambda_{\rightarrow}^{\emptyset}(A \rightarrow A); \\ \text{left}(\text{pair } x^A y^A) &=_{\beta\eta} x^A \& \text{right}(\text{pair } x^A y^A) =_{\beta\eta} y^A. \end{aligned}$$

The definition is formulated for $\lambda_{\rightarrow}^{\text{Ch}}$. The existence of a similar $A \times A \rightarrow A$ pairing in $\lambda_{\rightarrow}^{\text{Cu}}$ (leave out the superscripts in x^A, y^A) is by Proposition 1B.26 equivalent to that in $\lambda_{\rightarrow}^{\text{Ch}}$. We will show using a set-theoretic model that for all types $A \in \mathbb{T}$ there does not exist an $A \times A \rightarrow A$ pairing. We take $\mathbb{T} = \mathbb{T}^0$, but the argument for an arbitrary $\mathbb{T}^{\mathbb{A}}$ is the same.

2D.17. DEFINITION. (i) Let X be a set. The *full type structure* (for types in \mathbb{T}^0) over X , notation $\mathcal{M}_X = \{X(A)\}_{A \in \mathbb{T}^0}$, is defined as follows. For $A \in \mathbb{T}^0$ let $X(A)$ be defined inductively as follows.

$$\begin{aligned} X(0) &\triangleq X; \\ X(A \rightarrow B) &\triangleq X(B)^{X(A)}, \text{ the set of functions from } X(A) \text{ into } X(B). \end{aligned}$$

$$(ii) \quad \mathcal{M}_n \triangleq \mathcal{M}_{\{0, \dots, n\}}.$$

In order to use this model, we will use the Church version $\lambda_{\rightarrow}^{\text{Ch}}$, as terms from this system are naturally interpreted in \mathcal{M}_X .

2D.18. 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 $A \in \mathbb{T}^0$.

(ii) Let ρ be a valuation in \mathcal{M}_X . The *interpretation under ρ* of a $\lambda_{\rightarrow}^{\text{Ch}}$ -term into \mathcal{M}_X , notation $\llbracket M \rrbracket_{\rho}$, is defined as follows.

$$\begin{aligned} \llbracket x^A \rrbracket_{\rho} &\triangleq \rho(x^A); \\ \llbracket MN \rrbracket_{\rho} &\triangleq \llbracket M \rrbracket_{\rho} \llbracket N \rrbracket_{\rho}; \\ \llbracket \lambda x^A . M \rrbracket_{\rho} &\triangleq \lambda d \in X(A) . \llbracket M \rrbracket_{\rho(x^A := d)}, \end{aligned}$$

where $\rho(x^A := d) = \rho'$ with $\rho'(x^A) \triangleq d$ and $\rho'(y^B) \triangleq \rho(y^B)$ if $y^B \not\equiv x^A$.⁸

(iii) Define

$$\mathcal{M}_X \models M = N \Leftrightarrow \forall \rho \llbracket M \rrbracket_\rho = \llbracket N \rrbracket_\rho.$$

Before proving properties about the models it is good to do exercises 2E.11 and 2E.12.

2D.19. PROPOSITION. (i) $M \in \Lambda_{\rightarrow}^{\text{Ch}}(A) \Rightarrow \llbracket M \rrbracket_\rho \in X(A)$.

(ii) $M =_{\beta\eta} N \Rightarrow \mathcal{M}_X \models M = N$.

PROOF. (i) By induction on the structure of M .

(ii) By induction on the ‘proof’ of $M =_{\beta\eta} N$, using

$$\llbracket M[x := N] \rrbracket_\rho = \llbracket M \rrbracket_{\rho(x := \llbracket N \rrbracket_\rho)}, \text{ for the } \beta\text{-rule};$$

$$\rho \upharpoonright \text{FV}(M) = \rho' \upharpoonright \text{FV}(M) \Rightarrow \llbracket M \rrbracket_\rho = \llbracket M \rrbracket_{\rho'}, \text{ for the } \eta\text{-rule};$$

$$[\forall d \in X(A) \llbracket M \rrbracket_{\rho(x := d)} = \llbracket N \rrbracket_{\rho(x := d)}] \Rightarrow \llbracket \lambda x^A. M \rrbracket_\rho = \llbracket \lambda x^A. N \rrbracket_\rho, \text{ for the } \xi\text{-rule. } \blacksquare$$

Now we will give applications of the notion of type structure.

2D.20. PROPOSITION. Let $A \in \mathbb{T}^0$. Then there does not exist an $A \times A \rightarrow A$ 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 $A \times A \rightarrow A$ pairing, for otherwise f defined by

$$f(x, y) = \llbracket \text{pair} \rrbracket xy$$

would be an injection from $X(A) \times X(A)$ into $X(A)$, do exercise 2E.12. ■

2D.21. PROPOSITION. There is no term $\text{pred} \in \Lambda_{\rightarrow}^{\text{Ch}}(\text{Nat} \rightarrow \text{Nat})$ such that

$$\text{pred } \mathbf{c}_0 =_{\beta\eta} \mathbf{c}_0;$$

$$\text{pred } \mathbf{c}_{n+1} =_{\beta\eta} \mathbf{c}_n.$$

PROOF. As before for $X = \{0, 1\}$ the set $X(\text{Nat})$ is finite. Therefore

$$\mathcal{M}_X \models \mathbf{c}_n = \mathbf{c}_m,$$

for some $n \neq m$. If pred did exist, then it would follow easily that $\mathcal{M}_X \models \mathbf{c}_0 = \mathbf{c}_1$. But this implies that $X(0)$ has cardinality 1, since $\mathbf{c}_0(\mathsf{K}x)y = y$ but $\mathbf{c}_1(\mathsf{K}x)y = \mathsf{K}xy = x$, a contradiction. ■

Another application of semantics is that there are no fixed point combinators in $\Lambda_{\rightarrow}^{\text{Ch}}$.

2D.22. DEFINITION. A closed term Y is a *fixed point combinator* of type $A \in \mathbb{T}^0$ if

$$Y : \Lambda_{\rightarrow}^{\text{Ch}}((A \rightarrow A) \rightarrow A) \& Y =_{\beta\eta} \lambda f^{A \rightarrow A}. f(Yf).$$

2D.23. PROPOSITION. For no type A there exists in $\Lambda_{\rightarrow}^{\text{Ch}}$ 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 $f \in X(A \rightarrow A)$ without a fixed point:

$$\begin{aligned} f(z) &= x, & \text{if } z \neq x; \\ f(z) &= y, & \text{else.} \end{aligned}$$

If there is a fixed point combinator of type A , then $\llbracket Y \rrbracket f \in \mathcal{M}_X$ is a fixed point of f . Indeed, $Yx =_{\beta\eta} x(Yx)$ and taking $\llbracket \cdot \rrbracket_\rho$ with $\rho(x) = f$ the claim follows, a contradiction. ■

⁸Sometimes it is preferred to write $\llbracket \lambda x^A. M \rrbracket_\rho$ as $\lambda d \in X(A). \llbracket M[x^A := \underline{d}] \rrbracket_\rho$, 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.

Several results in this Section can easily be translated to $\lambda \xrightarrow{\mathbb{A}\infty}$ with arbitrarily many type variables, do exercise 2E.13.

2E. Exercises

- 2E.1. Find out which of the following terms are typable and determine for those that are the principal type.

$$\begin{aligned} &\lambda xyz.xz(yz); \\ &\lambda xyz.xy(xz); \\ &\lambda xyz.xy(zy). \end{aligned}$$

- 2E.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 B of M ? Is there a λl -term of type B ?
(ii) Find an expansion of M such that it has A as principal type.

- 2E.3. (Uniqueness of Type Assignments) Remember from B[1984] that

$$\Lambda_l \triangleq \{M \in \Lambda \mid \text{if } \lambda x.N \text{ is a subterm of } M, \text{ then } x \in \text{FV}(N)\}.$$

One has

$$M \in \Lambda_l, M \rightarrow_{\beta\eta} N \Rightarrow N \in \Lambda_l,$$

see e.g. B[1984], Lemma 9.1.2.

- (i) Show that for all $M_1, M_2 \in \Lambda_l^{\text{Ch}}(A)$ one has

$$|M_1| \equiv |M_2| \equiv M \in \Lambda_l^\varnothing \Rightarrow M_1 \equiv M_2.$$

[Hint. Use as induction loading towards open terms

$$|M_1| \equiv |M_2| \equiv M \in \Lambda_l \& \text{FV}(M_1) \equiv \text{FV}(M_2) \Rightarrow M_1 \equiv M_2.$$

This can be proved by induction on n , the length of the shortest β -reduction path to nf. For $n = 0$, see Propositions 1B.19(i) and 1B.24.]

- (ii) Show that in (i) the condition $M \in \Lambda_l^\varnothing$ cannot be weakened to

M has no K -redexes.

[Hint. Consider $M \equiv (\lambda x.xl)(\lambda z.l)$ and $A \equiv \alpha \rightarrow \alpha$.]

- 2E.4. Show that $\lambda \xrightarrow{\text{dB}}$ satisfies the Church-Rosser Theorem. [Hint. Use Proposition 1B.28 and translations between $\lambda \xrightarrow{\text{dB}}$ and $\lambda \xrightarrow{\text{Ch}}$.]

- 2E.5. (Hindley) Show that if $\vdash_{\lambda \xrightarrow{\text{cu}}} M : A$, then there is an M' such that

$$M' \rightarrow_{\beta\eta} M \& \text{pt}(M') = A.$$

[Hints. 1. First make an η -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 $(\alpha \rightarrow \beta) \rightarrow \gamma \rightarrow \delta$.)]

- 2E.6. Prove that $\Lambda_l^\varnothing(0) = \emptyset$ by applying the normalization and subject reduction theorems.

2E.7. Each type A of λ_{\rightarrow}^0 can be interpreted as an element $\llbracket A \rrbracket \in \mathbf{B}^{\mathbf{B}}$ as follows.

$$\llbracket A \rrbracket(i) = \llbracket A \rrbracket_{\rho_i},$$

where $\rho_i(0) = i$. There are four elements in $\mathbf{B}^{\mathbf{B}}$

$$\{\lambda x \in \mathbf{B}.0, \lambda x \in \mathbf{B}.1, \lambda x \in \mathbf{B}.x, \lambda x \in \mathbf{B}.1 - x\}.$$

Prove that $\llbracket A \rrbracket = \lambda x \in \mathbf{B}.1$ iff A is inhabited and $\llbracket A \rrbracket = \lambda x \in \mathbf{B}.x$ iff A is not inhabited.

- 2E.8. Show that Peirce's law $P = ((\alpha \rightarrow \beta) \rightarrow \alpha) \rightarrow \alpha$ is not forced in the Kripke model $\mathcal{K} = \langle K, \leq, 0, F \rangle$ with $K = \{0, 1\}$, $0 \leq 1$ and $F(0) = \emptyset, F(1) = \{\alpha\}$.
- 2E.9. Let X be a set and consider the typed λ -model \mathcal{M}_X . Notice that every permutation $\pi = \pi_0$ (bijection) of X can be lifted to all levels $X(A)$ by defining

$$\pi_{A \rightarrow B}(f) \triangleq \pi_B \circ f \circ \pi_A^{-1}.$$

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.]

- 2E.10. Prove that $\Lambda_{\rightarrow}^0(0) = \emptyset$ by applying models and the fact shown in the previous exercise that lambda definable elements are invariant under lifted permutations.
- 2E.11. (i) Show that $\mathcal{M}_X \models (\lambda x^A.x^A)y^A = y^A$.
(ii) Show that $\mathcal{M}_X \models (\lambda x^{A \rightarrow A}.x^{A \rightarrow A}) = (\lambda x^{A \rightarrow A}y^A.x^{A \rightarrow A}y^A)$.
(iii) Show that $\llbracket \mathbf{c}_2(\kappa x^0)y^0 \rrbracket_{\rho} = \rho(x)$.

- 2E.12. Let P, L, R be an $A \times B \rightarrow C$ pairing. Show that in every structure \mathcal{M}_X one has

$$\llbracket P \rrbracket xy = \llbracket P \rrbracket x'y' \Rightarrow x = x' \& y = y',$$

hence $\text{card}(A) \cdot \text{card}(B) \leq \text{card}(C)$.

- 2E.13. Show that Propositions 2D.20, 2D.21 and 2D.23 can be generalized to $\mathbb{A} = \mathbb{A}_{\infty}$ and the corresponding versions of $\lambda_{\rightarrow}^{\text{Cu}}$, by modifying the notion of type structure.
- 2E.14. Let $\sim A \equiv A \rightarrow 0$. Show that if 0 does not occur in A , then $\sim\sim(\sim\sim A \rightarrow A)$ is not inhabited. (One needs the *ex falso* rule to derive $\sim\sim(\sim\sim A \rightarrow A)$ as proposition.) Why is the condition about 0 necessary?
- 2E.15. We say that the structure of the rational numbers can be represented in $\lambda_{\rightarrow}^{\mathbb{A}}$ if there is a type $Q \in \mathbb{T}^{\mathbb{A}}$ and closed lambda terms:

$$\begin{aligned} 0, 1 : Q; \\ +, \cdot : Q \rightarrow Q \rightarrow Q; \\ -, ^{-1} : Q \rightarrow Q; \end{aligned}$$

such that $(Q, +, \cdot, -, ^{-1}, 0, 1)$ modulo $=_{\beta\eta}$ satisfies the axioms of a field of characteristic 0. Show that the rationals cannot be represented in $\lambda_{\rightarrow}^{\mathbb{A}}$. [Hint. Use a model theoretic argument.]

- 2E.16. Show that there is no closed term

$$P : \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$$

such that P is a bijection in the sense that

$$\forall M : \text{Nat} \exists ! N_1, N_2 : \text{Nat} \quad PN_1N_2 =_{\beta\eta} M.$$

- 2E.17. Show that every $M \in \Lambda^{\phi}((0 \rightarrow 0 \rightarrow 0) \rightarrow 0 \rightarrow 0)$ is $\beta\eta$ -convertible to $\lambda f^{0 \rightarrow 0 \rightarrow 0} x^0 . t$, with t given by the grammar

$$t := x \mid ft.$$

- 2E.18. [Hindley] Show that there is an ARS that is WCR but not CR. [Hint. An example of cardinality 4 exists.]

The next two exercises show that the minimal length of a reduction-path of a term to normal form is in the worst case non-elementary in the length of the term⁹. See Péter [1967] for the definition of the class of (Kalmár) 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 $2_n : \mathbb{N} \rightarrow \mathbb{N}$ as follows.

$$\begin{aligned} 2_0(x) &\triangleq x; \\ 2_{n+1}(x) &\triangleq 2^{2_n(x)}. \end{aligned}$$

Then every elementary function f is eventually bounded by some 2_n :

$$\exists n, m \forall x > m \quad f(x) \leq 2_n(x).$$

- 2E.19. (i) Define the function $\text{gk} : \mathbb{N} \rightarrow \mathbb{N}$ by

$$\begin{aligned} \text{gk}(m) &\triangleq \#F_{\text{GK}}(M), & \text{if } m = \#(M) \text{ for some untyped} \\ && \text{lambda term } M; \\ &\triangleq 0, & \text{else.} \end{aligned}$$

Here $\#M$ denotes the *Gödel-number* of the term M and F_{GK} is the Gross-Knuth reduction strategy defined by completely developing all present redexes in M , see B[1984]. Show that gk is Kalmár elementary.

- (ii) For a term $M \in \Lambda_{\rightarrow}^{\text{Ch}}$ define

$$D(M) \triangleq \max\{\text{dpt}(A \rightarrow B) \mid (\lambda x^A . P)^{A \rightarrow B} Q \text{ is a redex in } M\},$$

see Definition 1A.21(i). Show that if M is not a β -nf, then

$$F_{\text{GK}}(|M|) = |N| \Rightarrow D(M) > D(N),$$

where $|.| : \Lambda_{\rightarrow}^{\text{Ch}} \rightarrow \Lambda$ is the forgetful map. [Hint. Use Lévy's analysis of redex creation, see 2A.11(ii), or Lévy [1978], 1.8.4. lemme 3.3, for the proof.]

- (iii) If $M \in \Lambda$ is a term, then its *length*, notation $\text{lth}(M)$, is the number of symbols in M . Show that there is a constant c such that for typable lambda terms M one has for M sufficiently long

$$\text{dpth}(\text{pt}(M)) \leq c(\text{lth}(M)).$$

See the proof of Theorem 2C.14.

- (iv) Write $\sigma : M \rightarrow M^{\text{nf}}$ if σ is some reduction path of M to normal form M^{nf} . Let $\$ \sigma$ be the number of reduction steps in σ . Define

$$\$(M) \triangleq \min\{\$ \sigma \mid \sigma : M \rightarrow M^{\text{nf}}\}.$$

⁹In Gandy [1980b] 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.

Show that $\$M \leq g(\text{1th}(M))$, for some function $g \in \mathcal{E}_4$. [Hint. Take $g(m) = gk^m(m)$.]

- 2E.20. (i) Define $\mathbf{2}_1 \triangleq \lambda f^1 x^0. f(fx)$ and $\mathbf{2}_{n+1} \triangleq (\mathbf{2}_n[0:=1])\mathbf{2}$. Then for all $n \in \mathbb{N}$ one has $\mathbf{2}_n : 1 \rightarrow 0 \rightarrow 0$. Show that this type is the principal type of the Curry version $|\mathbf{2}_n|$ of $\mathbf{2}_n$.
- (ii) [Church] Show $(\mathbf{c}_n[0:=1])\mathbf{c}_m =_{\beta} \mathbf{c}_{m^n}$.
- (iii) Show $\mathbf{2}_n =_{\beta} \mathbf{c}_{2_n(1)}$, the notation is explained just above Exercise 2E.19.
- (iv) Let $M, N \in \Lambda$ be untyped terms. Show that if $M \rightarrow_{\beta} N$, then

$$\text{1th}(N) \leq \text{1th}(M)^2.$$

- (v) Conclude that $\$(M)$, see Exercise 2E.19, is in the worst case non-elementary in the length of M . That is, show that there is no elementary function f such that for all $M \in \Lambda_{\rightarrow}^{\text{Ch}}$

$$\$(M) \leq f(\text{1th}(M)).$$

- 2E.21. (i) Show that in the worst case the length of the principal type of a typable term is at least exponential in the length of the term, i.e. defining

$$f(m) = \max\{\text{1th}(\text{pt}(M)) \mid \text{1th}(M) \leq m\},$$

one has $f(n) \geq c^n$, for some real number $c > 1$ and sufficiently large n . [Hint. Define

$$M_n \triangleq \lambda x_n \cdots x_1. x_n(x_n x_{n-1})(x_{n-1}(x_{n-1} x_{n-2})) \cdots (x_2(x_2 x_1)).$$

Show that the principal type of M_n has length $> 2^n$.]

- (ii) Show that the length of the principal type of a term M is also at most exponential in the length of M . [Hint. First show that the depth of the principal type of a typable term M is linear in the length of M .]

- 2E.22. (Statman) We want to show that $\mathcal{M}_n \hookrightarrow \mathcal{M}_{\mathbb{N}}$, for $n \geq 1$, by an isomorphic embedding.

- (i) (Church's δ) For $A \in \mathbb{T}^0$ define $\delta_A \in \mathcal{M}_n(A^2 \rightarrow 0^2 \rightarrow 0)$ by

$$\begin{aligned} \delta_A x y u v &\triangleq u && \text{if } x = y; \\ &\triangleq v && \text{else.} \end{aligned}$$

- (ii) We add to the language $\Lambda_{\rightarrow}^{\text{Ch}}$ constants $\underline{k} : 0$ for $1 \leq k \leq n$ and a constant $\underline{\delta} : 0^4 \rightarrow 0$. The intended interpretation of $\underline{\delta}$ is the map δ_0 . We define the notion of reduction δ by the contraction rules

$$\begin{aligned} \underline{\delta} \underline{i} \underline{j} \underline{k} \underline{l} &\rightarrow_{\delta} \underline{k} && \text{if } i = j; \\ &\rightarrow_{\delta} \underline{l}, && \text{if } i \neq j. \end{aligned}$$

The resulting language of terms is called Λ_{δ} and on this we consider the notion of reduction $\rightarrow_{\beta\eta\delta}$.

- (iii) Show that every $M \in \Lambda_{\delta}$ satisfies $\text{SN}_{\beta\eta\delta}(M)$.
- (iv) Show that $\rightarrow_{\beta\eta\delta}$ is Church-Rosser.
- (v) Let $M \in \Lambda_{\delta}^0(0)$ be a closed term of type 0. Show that the normal form of M is one of the constants $\underline{1}, \dots, \underline{n}$.

- (vi) (Church's theorem.) Show that every element $\Phi \in \mathcal{M}_n$ can be defined by a closed term $M_\Phi \in \Lambda_\delta$, i.e. $\Phi = \llbracket M_\Phi \rrbracket^{\mathcal{M}_n}$. [Hint. For each $A \in \mathbb{T}$ define simultaneously the map $\Phi \mapsto M_\Phi : \mathcal{M}_n(A) \rightarrow \Lambda_\delta(A)$ and $\underline{\delta}_A \in \Lambda_\delta(A^2 \rightarrow 0^2 \rightarrow 0)$ such that $\llbracket \underline{\delta}_A \rrbracket = \delta_A$ and $\Phi = \llbracket M_\Phi \rrbracket^{\mathcal{M}_n}$. For $A = 0$ take $M_i = \underline{i}$ and $\underline{\delta}_0 = \underline{\delta}$. For $A = B \rightarrow C$, let $\mathcal{M}_n(B) = \{\Phi_1, \dots, \Phi_t\}$ and $C = C_1 \rightarrow \dots \rightarrow C_c \rightarrow 0$. Define

$$\begin{aligned} \underline{\delta}_A &\triangleq \lambda xyuv. (\underline{\delta}_C(xM_{\Phi_1})(yM_{\Phi_1}) \\ &\quad (\underline{\delta}_C(xM_{\Phi_2})(yM_{\Phi_2}) \\ &\quad (\dots \\ &\quad (\underline{\delta}_C(xM_{\Phi_{t-1}})(yM_{\Phi_{t-1}}) \\ &\quad (\underline{\delta}_C(xM_{\Phi_t})(yM_{\Phi_t})uv)v)v)v)v). \\ M_\Phi &\triangleq \lambda xy_1 \dots y_c. (\underline{\delta}_BxM_{\Phi_1}(M_{\Phi_1}\vec{y}) \\ &\quad (\underline{\delta}_BxM_{\Phi_2}(M_{\Phi_2}\vec{y}) \\ &\quad (\dots \\ &\quad (\underline{\delta}_BxM_{\Phi_{t-1}}(M_{\Phi_{t-1}}\vec{y})) \\ &\quad (\underline{\delta}_BxM_{\Phi_t}(M_{\Phi_t}\vec{y})0)\dots)).] \end{aligned}$$

- (vii) Show that $\Phi \mapsto \llbracket M_\Phi \rrbracket^{\mathcal{M}_n} : \mathcal{M}_n \hookrightarrow \mathcal{M}_n$ is the required embedding.
(viii) (To be used later.) Let $\pi_i^n \equiv (\lambda x_1 \dots x_n. x_i) : (0^n \rightarrow 0)$. Define

$$\begin{aligned} \Delta^n &\triangleq \lambda abuv\vec{x}. a (b(u\vec{x})(v\vec{x}) \dots (v\vec{x})(v\vec{x})) \\ &\quad (b(v\vec{x})(u\vec{x}) \dots (v\vec{x})(v\vec{x})) \\ &\quad \dots \\ &\quad (b(v\vec{x})(v\vec{x}) \dots (u\vec{x})(v\vec{x})) \\ &\quad (b(v\vec{x})(v\vec{x}) \dots (v\vec{x})(u\vec{x})). \end{aligned}$$

Then

$$\begin{aligned} \Delta^n \pi_i^n \pi_j^n \pi_k^n \pi_l^n &=_{\beta\eta\delta} \pi_k^n, \quad \text{if } i = j; \\ &=_{\beta\eta\delta} \pi_l^n, \quad \text{else.} \end{aligned}$$

Show that for $i \in \{1, \dots, n\}$ one has for all $M : 0$

$$\begin{aligned} M &=_{\beta\eta\delta} \underline{i} \Rightarrow \\ M[0 := 0^n \rightarrow 0][\delta := \Delta^n][\underline{1} := \pi_1^n] \dots [\underline{n} := \pi_n^n] &=_{\beta\eta} \pi_i^n. \end{aligned}$$

2E.23. (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 $\delta : \Sigma \times Q \rightarrow Q$ 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 typed λ -model over Q . Show that

$$w \in L^r(M) \Leftrightarrow \llbracket w \rrbracket^{\mathcal{M}_{\delta_{a_1} \dots \delta_{a_n} q_0}} \in F,$$

where $\delta_a(q) = \delta(a, q)$ and w is defined in 1D.8.

- (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 $\top_n = (0^2 \rightarrow 0)^n \rightarrow 0 \rightarrow 0$.

CHAPTER 3

TOOLS

3A. Semantics of λ_{\rightarrow}

So far the systems $\lambda_{\rightarrow}^{Cu}$ and $\lambda_{\rightarrow}^{Ch}$ (and also its variant $\lambda_{\rightarrow}^{dB}$) had closely related properties. In this chapter we will give two rather different semantics to $\lambda_{\rightarrow}^{Ch}$ and to $\lambda_{\rightarrow}^{Cu}$, respectively. This will appear in the intention one has while giving a semantics for these systems. For the Church systems $\lambda_{\rightarrow}^{Ch}$, in which every λ -term comes with its unique type, there is a semantics consisting of disjoint layers, each of these corresponding with a given type. Terms of type A will be interpreted as elements of the layer corresponding to A . The Curry systems $\lambda_{\rightarrow}^{Cu}$ are essentially treated as untyped λ -calculi, where one assigns to a term a set (that sometimes can be empty) of possible types. This then results in an untyped λ -model with overlapping subsets indexed by the types. This happens in such a way that if type A is assigned to term M , then the interpretation of M is an element of the subset with index A . The notion of semantics has been inspired by [Henkin \[1950\]](#), dealing with the completeness in the theory of types.

Semantics for type assignment *à la* Church

In this subsection we work with the Church variant of λ_{\rightarrow}^0 having one atomic type 0 , rather than with $\lambda_{\rightarrow}^{\mathbb{A}}$, having an arbitrary set of atomic types. We will write $\mathbb{T} = \mathbb{T}^0$. The reader is encouraged to investigate which results do generalize to $\mathbb{T}^{\mathbb{A}}$.

3A.1. DEFINITION. Let $\mathcal{M} = \{\mathcal{M}(A)\}_{A \in \mathbb{T}}$ be a family of non-empty sets indexed by types $A \in \mathbb{T}$.

(i) \mathcal{M} is called a *type structure* for λ_{\rightarrow}^0 if

$$\mathcal{M}(A \rightarrow B) \subseteq \mathcal{M}(B)^{\mathcal{M}(A)}.$$

Here \mathcal{X}^Y denotes the collection of set-theoretic functions

$$\{f \mid f : Y \rightarrow X\}.$$

(ii) Let X be a set. The *full type structure* \mathcal{M} over the ground set X defined in 2D.17 was specified by

$$\begin{aligned} \mathcal{M}(0) &\triangleq X \\ \mathcal{M}(A \rightarrow B) &\triangleq \mathcal{M}(B)^{\mathcal{M}(A)}, \quad \text{for all } A, B \in \mathbb{T}. \end{aligned}$$

(iii) Let \mathcal{M} be provided with *application operators*

$$(\mathcal{M}, \cdot) = (\{\mathcal{M}(A)\}_{A \in \mathbb{T}}, \{\cdot_{A,B}\}_{A,B \in \mathbb{T}})$$

$$\cdot_{A,B} : \mathcal{M}(A \rightarrow B) \times \mathcal{M}(A) \rightarrow \mathcal{M}(B).$$

A *typed applicative structure* is such an (\mathcal{M}, \cdot) satisfying *extensionality*:

$$\forall f, g \in \mathcal{M}(A \rightarrow B) [[\forall a \in \mathcal{M}(A) f \cdot_{A,B} a = g \cdot_{A,B} a] \Rightarrow f = g].$$

(iv) \mathcal{M} is called *trivial* if $\mathcal{M}(0)$ is a singleton. Then $\mathcal{M}(A)$ is a singleton for all $A \in \mathbb{T}$.

3A.2. NOTATION. For typed applicative structures we use the infix notation $f \cdot_{A,B} x$ or $f \cdot x$ for $\cdot_{A,B}(f, x)$. Often we will be even more brief, extensionality becoming

$$\forall f, g \in \mathcal{M}(A \rightarrow B) [[\forall a \in \mathcal{M}_A fa = ga] \Rightarrow f = g]$$

or simply,

$$\forall f, g \in \mathcal{M} [[\forall a fa = ga] \Rightarrow f = g],$$

where f, g range over the same type $A \rightarrow B$ and a ranges over \mathcal{M}_A .

3A.3. PROPOSITION. *The notions of type structure and typed applicative structure are equivalent.*

PROOF. In a type structure \mathcal{M} define $f \cdot a \triangleq f(a)$; extensionality is obvious. Conversely, let (\mathcal{M}, \cdot) be a typed applicative structure. Define the type structure \mathcal{M}' and $\Phi_A : \mathcal{M}(A) \rightarrow \mathcal{M}'(A)$ as follows.

$$\begin{aligned} \mathcal{M}'(0) &\triangleq \mathcal{M}(0); \\ \Phi_0(a) &\triangleq a; \\ \mathcal{M}'(A \rightarrow B) &\triangleq \{\Phi_{A \rightarrow B}(f) \in \mathcal{M}'(B)^{\mathcal{M}'(A)} \mid f \in \mathcal{M}(A \rightarrow B)\}; \\ \Phi_{A \rightarrow B}(f)(\Phi_A(a)) &\triangleq \Phi_B(f \cdot a). \end{aligned}$$

By definition Φ is surjective. By extensionality of the typed applicative structure it is also injective. Hence $\Phi_{A \rightarrow B}(f)$ is well defined. Clearly one has $\mathcal{M}'(A \rightarrow B) \subseteq \mathcal{M}'(B)^{\mathcal{M}'(A)}$. ■

3A.4. DEFINITION. Let \mathcal{M}, \mathcal{N} be two typed applicative structures. A *morphism* is a type indexed family $F = \{F_A\}_{A \in \mathbb{T}}$ such that for each $A, B \in \mathbb{T}$ one has

$$\begin{aligned} F_A &: \mathcal{M}(A) \rightarrow \mathcal{N}(A); \\ F_{A \rightarrow B}(f) \cdot F_A(a) &= F_B(f \cdot a). \end{aligned}$$

From now on we will not make a distinction between the notions ‘type structure’ and ‘typed applicative structure’.

3A.5. PROPOSITION. *Let \mathcal{M} be a type structure. Then*

$$\mathcal{M} \text{ is trivial} \Leftrightarrow \forall A \in \mathbb{T}. \mathcal{M}(A) \text{ is a singleton.}$$

PROOF. (\Leftarrow) By definition. (\Rightarrow) We will show this for $A = 1 = 0 \rightarrow 0$. If $\mathcal{M}(0)$ is a singleton, then for all $f, g \in \mathcal{M}(1)$ one has $\forall x: \mathcal{M}(0). (fx) = (gx)$, hence $f = g$, by extensionality. Therefore $\mathcal{M}(1)$ is a singleton. ■

3A.6. EXAMPLE. The full type structure $\mathcal{M}_X = \{X(A)\}_{A \in \mathbb{T}}$ over a non-empty set X , see definition 2D.17, is a typed applicative structure.

3A.7. DEFINITION. (i) Let (X, \leq) be a non-empty partially ordered set. Let $D(0) = X$ and $D(A \rightarrow B)$ consist of the monotone elements of $D(B)^{D(A)}$, where we order this set pointwise: for $f, g \in D(A \rightarrow B)$ define

$$f \leq g \iff \forall a \in D(A). fa \leq ga.$$

The elements of the typed applicative structure $D_X = \{D(A)\}_{A \in \mathbb{T}}$ are called the *hereditarily monotone functions*. See Howard in Troelstra [1973] as well as Bezem [1989] for several closely related type structures.

(ii) Let \mathcal{M} be a typed applicative structure. A *layered non-empty subfamily* of \mathcal{M} is a family $\Delta = \{\Delta(A)\}_{A \in \mathbb{T}}$ of sets, such that the following holds

$$\forall A \in \mathbb{T}. \emptyset \neq \Delta(A) \subseteq \mathcal{M}(A).$$

Δ is called *closed under application* if

$$f \in \Delta(A \rightarrow B), g \in \Delta(A) \Rightarrow fg \in \Delta(B).$$

Δ is called *extensional* if

$$\forall A, B \in \mathbb{T} \forall f, g \in \Delta(A \rightarrow B). [[\forall a \in \Delta(A). fa = ga] \Rightarrow f = g].$$

If Δ satisfies all these conditions, then $\mathcal{M} \upharpoonright \Delta = (\Delta, \cdot \upharpoonright \Delta)$ is a typed applicative structure.

3A.8. DEFINITION (Environments). (i) Let \mathcal{D} be a set and V the set of variables of the untyped lambda calculus. A *(term) environment in \mathcal{D}* is a total map

$$\rho : V \rightarrow \mathcal{D}.$$

The set of environments in \mathcal{D} is denoted by $\text{Env}_{\mathcal{D}}$.

(ii) If $\rho \in \text{Env}_{\mathcal{D}}$ and $d \in \mathcal{D}$, then $\rho[x := d]$ is the $\rho' \in \text{Env}_{\mathcal{D}}$ defined by

$$\rho'(y) \triangleq \begin{cases} d & \text{if } y = x, \\ \rho(y) & \text{otherwise.} \end{cases}$$

3A.9. DEFINITION. (i) Let \mathcal{M} be a typed applicative structure. Then a *(partial) valuation in \mathcal{M}* is a family of (partial) maps $\rho = \{\rho_A\}_{A \in \mathbb{T}}$ such that $\rho_A : \text{Var}(A) \uparrow \mathcal{M}(A)$.

(ii) Given a typed applicative structure \mathcal{M} and a partial valuation ρ in \mathcal{M} one defines the *partial semantics* $\llbracket \cdot \rrbracket_{\rho} : \Lambda_{\rightarrow}(A) \uparrow \mathcal{M}(A)$ as follows. Let Γ be a context and ρ a valuation. For $M \in \Lambda_{\rightarrow}^{\Gamma}(A)$ its semantics under ρ , notation $\llbracket M \rrbracket_{\rho}^{\mathcal{M}} \in \mathcal{M}(A)$, is

$$\begin{aligned} \llbracket x^A \rrbracket_{\rho}^{\mathcal{M}} &\triangleq \rho_A(x); \\ \llbracket PQ \rrbracket_{\rho}^{\mathcal{M}} &\triangleq \llbracket P \rrbracket_{\rho}^{\mathcal{M}} \llbracket Q \rrbracket_{\rho}^{\mathcal{M}}; \\ \llbracket \lambda x^A. P \rrbracket_{\rho}^{\mathcal{M}} &\triangleq \lambda d \in \mathcal{M}(A). \llbracket P \rrbracket_{\rho[x:=d]}^{\mathcal{M}}. \end{aligned}$$

We often write $\llbracket M \rrbracket_{\rho}$ for $\llbracket M \rrbracket_{\rho}^{\mathcal{M}}$, if there is little danger of confusion. The expression $\llbracket M \rrbracket_{\rho}$ may not always be defined, even if ρ is total. The problem arises with $\llbracket \lambda x. P \rrbracket_{\rho}$. Although the function

$$\lambda d \in \mathcal{M}(A). \llbracket P \rrbracket_{\rho[x:=d]} \in \mathcal{M}(B)^{\mathcal{M}(A)}$$

is uniquely determined by $\llbracket \lambda x.P \rrbracket_\rho d = \llbracket P \rrbracket_{\rho[x:=d]}$, it may fail to be an element of $\mathcal{M}(A \rightarrow B)$ which is only a subset of $\mathcal{M}(B)^{\mathcal{M}(A)}$. If $\llbracket M \rrbracket_\rho$ is defined, we write $\llbracket M \rrbracket_\rho \downarrow$, otherwise, if $\llbracket M \rrbracket_\rho$ is undefined, we write $\llbracket M \rrbracket_\rho \uparrow$.

3A.10. DEFINITION. (i) A type structure \mathcal{M} is called a λ^0_{\rightarrow} -model or a typed λ -model if for every partial valuation $\rho = \{\rho_A\}_A$ and every $A \in \mathbb{T}$ and $M \in \Lambda_{\rightarrow}^{\Gamma}(A)$ such that $\text{FV}(M) \subseteq \text{dom}(\rho)$ one has $\llbracket M \rrbracket_\rho \downarrow$.

(ii) Let \mathcal{M} be a typed λ -model and ρ a partial valuation. Then \mathcal{M}, ρ satisfies $M = N$, assuming implicitly that M and N have the same type, notation

$$\mathcal{M}, \rho \models M = N$$

if $\llbracket M \rrbracket_\rho^{\mathcal{M}} = \llbracket N \rrbracket_\rho^{\mathcal{M}}$.

(iii) Let \mathcal{M} be a typed λ -model. Then \mathcal{M} satisfies $M = N$, notation

$$\mathcal{M} \models M = N$$

if for all partial ρ with $\text{FV}(MN) \subseteq \text{dom}(\rho)$ one has $\mathcal{M}, \rho \models M = N$.

(iv) Let \mathcal{M} be a typed λ -model. The theory of \mathcal{M} is defined as

$$\text{Th}(\mathcal{M}) \triangleq \{M = N \mid M, N \in \Lambda_{\rightarrow}^{\phi} \& \mathcal{M} \models M = N\}.$$

3A.11. NOTATION. Let E_1, E_2 be partial (i.e. possibly undefined) expressions.

- (i) Write $E_1 \succsim E_2$ for $E_1 \downarrow \Rightarrow [E_2 \downarrow \& E_1 = E_2]$.
- (ii) Write $E_1 \simeq E_2$ for $E_1 \succsim E_2 \& E_2 \succsim E_1$.

3A.12. LEMMA. (i) Let $M \in \Lambda_0(A)$ and N be a subterm of M . Then

$$\llbracket M \rrbracket_\rho \downarrow \Rightarrow \llbracket N \rrbracket_\rho \downarrow.$$

(ii) Let $M \in \Lambda_0(A)$. Then

$$\llbracket M \rrbracket_\rho \simeq \llbracket M \rrbracket_{\rho \upharpoonright \text{FV}(M)}.$$

(iii) Let $M \in \Lambda_0(A)$ and ρ_1, ρ_2 be such that $\rho_1 \upharpoonright \text{FV}(M) = \rho_2 \upharpoonright \text{FV}(M)$. Then

$$\llbracket M \rrbracket_{\rho_1} \simeq \llbracket M \rrbracket_{\rho_2}.$$

PROOF. (i) By induction on the structure of M .

- (ii) Similarly.
- (iii) By (ii). ■

3A.13. LEMMA. Let \mathcal{M} be a typed applicative structure. Then

- (i) For $M \in \Lambda_0(A)$, $x, N \in \Lambda_0(B)$ one has

$$\llbracket M[x:=N] \rrbracket_\rho^{\mathcal{M}} \simeq \llbracket M \rrbracket_{\rho[x:=\llbracket N \rrbracket_\rho^{\mathcal{M}}]}^{\mathcal{M}}.$$

- (ii) For $M, N \in \Lambda_0(A)$ one has

$$M \rightarrowtail_{\beta\eta} N \Rightarrow \llbracket M \rrbracket_\rho^{\mathcal{M}} \succsim \llbracket N \rrbracket_\rho^{\mathcal{M}}.$$

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 $y \notin \text{FV}(N)$. We have

$$\begin{aligned} \llbracket (\lambda y.P)^{\bullet} \rrbracket_{\rho} &\simeq \llbracket \lambda y.P^{\bullet} \rrbracket_{\rho} \\ &\simeq \lambda d. \llbracket P^{\bullet} \rrbracket_{\rho[y:=d]} \\ &\simeq \lambda d. \llbracket P \rrbracket_{\rho[y:=d][x:=\llbracket N \rrbracket_{\rho[y:=d]}]}, \quad \text{by the IH,} \\ &\simeq \lambda d. \llbracket P \rrbracket_{\rho[y:=d][x:=\llbracket N \rrbracket_{\rho}]}, \quad \text{by Lemma 3A.12,} \\ &\simeq \lambda d. \llbracket P \rrbracket_{\rho[x:=\llbracket N \rrbracket_{\rho}][y:=d]} \\ &\simeq \llbracket \lambda y.P \rrbracket_{\rho[x:=\llbracket N \rrbracket_{\rho}]} \end{aligned}$$

(ii) By induction on the generation of $M \rightarrow_{\beta\eta} N$.

Case $M \equiv (\lambda x.P)Q$ and $N \equiv P[x := Q]$. Then

$$\begin{aligned} \llbracket (\lambda x.P)Q \rrbracket_{\rho} &\succsim (\lambda d. \llbracket P \rrbracket_{\rho[x:=d]})(\llbracket Q \rrbracket_{\rho}) \\ &\succsim \llbracket P \rrbracket_{\rho[x:=\llbracket Q \rrbracket_{\rho}]} \\ &\simeq \llbracket P[x := Q] \rrbracket_{\rho}, \quad \text{by (i).} \end{aligned}$$

Case $M \equiv \lambda x.Nx$, with $x \notin \text{FV}(N)$. Then

$$\begin{aligned} \llbracket \lambda x.Nx \rrbracket_{\rho} &\succsim \lambda d. \llbracket N \rrbracket_{\rho}(d) \\ &\simeq \llbracket N \rrbracket_{\rho}. \end{aligned}$$

Cases $M \rightarrow_{\beta\eta} N$ is $PZ \rightarrow_{\beta\eta} QZ$, $ZP \rightarrow_{\beta\eta} ZQ$ or $\lambda x.P \rightarrow_{\beta\eta} \lambda x.Q$, and follows directly from $P \rightarrow_{\beta\eta} Q$. Then the result follows from the IH.

The cases where $M \rightarrow_{\beta\eta} N$ follows via reflexivity or transitivity are easy to treat. ■

3A.14. DEFINITION. Let \mathcal{M}, \mathcal{N} be typed λ -models and let $A \in \Pi$.

(i) \mathcal{M} and \mathcal{N} are *elementary equivalent at A*, notation $\mathcal{M} \equiv_A \mathcal{N}$, iff

$$\forall M, N \in \Lambda_{\rightarrow}^{\emptyset}(A). [\mathcal{M} \models M = N \Leftrightarrow \mathcal{N} \models M = N].$$

(ii) \mathcal{M} and \mathcal{N} are *elementary equivalent*, notation $\mathcal{M} \equiv \mathcal{N}$, iff

$$\forall A \in \Pi. \mathcal{M} \equiv_A \mathcal{N}.$$

3A.15. PROPOSITION. Let \mathcal{M} be a typed λ -model. Then

$$\mathcal{M} \text{ is non-trivial} \Leftrightarrow \forall A \in \Pi. \mathcal{M}(A) \text{ is not a singleton.}$$

PROOF. (\Leftarrow) By definition. (\Rightarrow) We will show this for $A = 1 = 0 \rightarrow 0$. Let c_1, c_2 be distinct elements of $\mathcal{M}(0)$. Consider $M \equiv \lambda x^0.y^0 \in \Lambda_{\rightarrow}^{\emptyset}(1)$. Let ρ_i be the partial valuation with $\rho_i(y^0) = c_i$. Then $\llbracket M \rrbracket_{\rho_i} \downarrow$ and $\llbracket M \rrbracket_{\rho_1}c_1 = c_1, \llbracket M \rrbracket_{\rho_2}c_1 = c_2$. Therefore $\llbracket M \rrbracket_{\rho_1}, \llbracket M \rrbracket_{\rho_2}$ are different elements of $\mathcal{M}(1)$. ■

Thus with Proposition 3A.5 one has for a typed λ -model \mathcal{M}

$$\begin{aligned} \mathcal{M}(0) \text{ is a singleton} &\Leftrightarrow \forall A \in \Pi. \mathcal{M}(A) \text{ is a singleton} \\ &\Leftrightarrow \exists A \in \Pi. \mathcal{M}(A) \text{ is a singleton.} \end{aligned}$$

3A.16. PROPOSITION. Let \mathcal{M}, \mathcal{N} be typed λ -models and $F: \mathcal{M} \rightarrow \mathcal{N}$ a surjective morphism. Then the following hold.

- (i) $F(\llbracket M \rrbracket_{\rho}^{\mathcal{M}}) = \llbracket M \rrbracket_{F \circ \rho}^{\mathcal{N}}$, for all $M \in \Lambda_{\rightarrow}(A)$.
- (ii) $F(\llbracket M \rrbracket^{\mathcal{M}}) = \llbracket M \rrbracket^{\mathcal{N}}$, for all $M \in \Lambda_{\rightarrow}^{\circ}(A)$.

PROOF. (i) By induction on the structure of M .

Case $M \equiv x$. Then $F(\llbracket x \rrbracket_{\rho}^{\mathcal{M}}) = F(\rho(x)) = \llbracket x \rrbracket_{F \circ \rho}^{\mathcal{N}}$.

Case $M = PQ$. Then

$$\begin{aligned} F(\llbracket PQ \rrbracket_{\rho}^{\mathcal{M}}) &= F(\llbracket P \rrbracket_{\rho}^{\mathcal{M}}) \cdot_{\mathcal{N}} F(\llbracket Q \rrbracket_{\rho}^{\mathcal{M}}) \\ &= \llbracket P \rrbracket_{F \circ \rho}^{\mathcal{N}} \cdot_{\mathcal{N}} \llbracket Q \rrbracket_{F \circ \rho}^{\mathcal{N}}, \quad \text{by the IH,} \\ &= \llbracket PQ \rrbracket_{F \circ \rho}^{\mathcal{N}}. \end{aligned}$$

Case $M = \lambda x.P$. Then we must show

$$F(\lambda d \in \mathcal{M}. \llbracket P \rrbracket_{\rho[x:=d]}^{\mathcal{M}}) = \lambda e \in \mathcal{N}. \llbracket P \rrbracket_{(F \circ \rho)[x:=e]}^{\mathcal{M}}.$$

By extensionality it suffices to show for all $e \in \mathcal{N}$

$$F(\lambda d \in \mathcal{M}. \llbracket P \rrbracket_{\rho[x:=d]}^{\mathcal{M}}) \cdot_{\mathcal{N}} e = \llbracket P \rrbracket_{(F \circ \rho)[x:=e]}^{\mathcal{M}}.$$

By surjectivity of F it suffices to show this for $e = F(d)$. Indeed,

$$\begin{aligned} F(\llbracket P \rrbracket_{\rho[x:=d]}^{\mathcal{M}}) \cdot_{\mathcal{N}} F(d) &= F(\llbracket P \rrbracket_{\rho[x:=d]}^{\mathcal{N}}) \\ &= \llbracket P \rrbracket_{F \circ (\rho[x:=d])}^{\mathcal{N}}, \quad \text{by the IH,} \\ &= \llbracket P \rrbracket_{(F \circ \rho)[x:=F(d)]}^{\mathcal{N}}. \end{aligned}$$

(ii) By (i). ■

3A.17. PROPOSITION. Let \mathcal{M} be a typed λ -model.

- (i) $\mathcal{M} \models (\lambda x.M)N = M[x := N]$.
- (ii) $\mathcal{M} \models \lambda x.Mx = M$, if $x \notin \text{FV}(\mathcal{M})$.

PROOF. (i) $\llbracket (\lambda x.M)N \rrbracket_{\rho} = \llbracket \lambda x.M \rrbracket_{\rho} \llbracket N \rrbracket_{\rho}$

$$\begin{aligned} &= \llbracket M \rrbracket_{\rho[x := \llbracket N \rrbracket_{\rho}]}, \\ &= \llbracket M[x := N] \rrbracket_{\rho}, \quad \text{by Lemma 3A.13.} \end{aligned}$$

(ii) $\llbracket \lambda x.Mx \rrbracket_{\rho} d = \llbracket Mx \rrbracket_{\rho[x:=d]}$

$$\begin{aligned} &= \llbracket M \rrbracket_{\rho[x:=d]} d \\ &= \llbracket M \rrbracket_{\rho} d, \quad \text{as } x \notin \text{FV}(M). \end{aligned}$$

Therefore by extensionality $\llbracket \lambda x.Mx \rrbracket_{\rho} = \llbracket M \rrbracket_{\rho}$. ■

3A.18. LEMMA. Let \mathcal{M} be a typed λ -model. Then

$$\mathcal{M} \models M = N \Leftrightarrow \mathcal{M} \models \lambda x.M = \lambda x.N.$$

PROOF. $\mathcal{M} \models M = N \Leftrightarrow \forall \rho. \llbracket M \rrbracket_{\rho} = \llbracket N \rrbracket_{\rho}$ ■

$$\begin{aligned} &\Leftrightarrow \forall \rho, d. \llbracket M \rrbracket_{\rho[x:=d]} = \llbracket N \rrbracket_{\rho[x:=d]} \\ &\Leftrightarrow \forall \rho, d. \llbracket \lambda x.M \rrbracket_{\rho} d = \llbracket \lambda x.N \rrbracket_{\rho} d \\ &\Leftrightarrow \forall \rho. \llbracket \lambda x.M \rrbracket_{\rho} = \llbracket \lambda x.N \rrbracket_{\rho} \\ &\Leftrightarrow \mathcal{M} \models \lambda x.M = \lambda x.N. \end{aligned}$$

3A.19. PROPOSITION. (i) For every non-empty set X the type structure \mathcal{M}_X is a λ_{\rightarrow}^0 -model.

- (ii) Let X be a poset. Then \mathcal{D}_X is a λ_{\rightarrow}^0 -model.
- (iii) Let \mathcal{M} be a typed applicative structure. Assume that $\llbracket K_{A,B} \rrbracket^{\mathcal{M}} \downarrow$ and $\llbracket S_{A,B,C} \rrbracket^{\mathcal{M}} \downarrow$. Then \mathcal{M} is a λ_{\rightarrow}^0 -model.
- (iv) Let Δ be a layered non-empty subfamily of a typed applicative structure \mathcal{M} that is extensional and closed under application. Suppose $\llbracket K_{A,B} \rrbracket, \llbracket S_{A,B,C} \rrbracket$ are defined and in Δ . Then $\mathcal{M} \upharpoonright \Delta$, see Definition 3A.7(ii), is a λ_{\rightarrow}^0 -model.

PROOF. (i) Since \mathcal{M}_X is the full type structure, $\llbracket M \rrbracket_\rho$ always exists.

(ii) By induction on M one can show that $\lambda d. \llbracket M \rrbracket_{\rho(x:=d)}$ is monotonic. It then follows by induction on M that $\llbracket M \rrbracket_\rho \in \mathcal{D}_X$.

(iii) For every λ -term M there exists a typed applicative expression P consisting only of K s and S s such that $P \rightarrow_{\beta\eta} M$. Now apply Lemma 3A.13.

(iv) By (iii). ■

Operations on typed λ -models

Now we will introduce two operations on λ -models: $\mathcal{M}, \mathcal{N} \mapsto \mathcal{M} \times \mathcal{N}$, the Cartesian product, and $\mathcal{M} \mapsto \mathcal{M}^*$, the polynomial λ -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

3A.20. DEFINITION. If \mathcal{M}, \mathcal{N} are typed applicative structures, then the *Cartesian product* of \mathcal{M}, \mathcal{N} , notation $\mathcal{M} \times \mathcal{N}$, is the structure defined by

$$\begin{aligned} (\mathcal{M} \times \mathcal{N})(A) &\triangleq \mathcal{M}(A) \times \mathcal{N}(A) \\ (M_1, N_1) \cdot (M_2, N_2) &\triangleq (M_1 \cdot M_2, N_1 \cdot N_2). \end{aligned}$$

3A.21. PROPOSITION. Let \mathcal{M}, \mathcal{N} be typed λ -models. For a partial valuation ρ in $\mathcal{M} \times \mathcal{N}$ write $\rho(x) \triangleq (\rho_1(x), \rho_2(x))$. Then

- (i) $\llbracket M \rrbracket_{\rho}^{\mathcal{M} \times \mathcal{N}} = (\llbracket M \rrbracket_{\rho_1}^{\mathcal{M}}, \llbracket M \rrbracket_{\rho_2}^{\mathcal{N}})$.
- (ii) $\mathcal{M} \times \mathcal{N}$ is a λ -model.
- (iii) $\text{Th}(\mathcal{M} \times \mathcal{N}) = \text{Th}(\mathcal{M}) \cap \text{Th}(\mathcal{N})$.

PROOF. (i) By induction on M .

(ii) By (i).

$$\begin{aligned} \text{(iii)} \quad \mathcal{M} \times \mathcal{N}, \rho \models M = N &\Leftrightarrow \llbracket M \rrbracket_\rho = \llbracket N \rrbracket_\rho \\ &\Leftrightarrow (\llbracket M \rrbracket_{\rho_1}^{\mathcal{M}}, \llbracket M \rrbracket_{\rho_2}^{\mathcal{N}}) = (\llbracket N \rrbracket_{\rho_1}^{\mathcal{M}}, \llbracket N \rrbracket_{\rho_2}^{\mathcal{N}}) \\ &\Leftrightarrow \llbracket M \rrbracket_{\rho_1}^{\mathcal{M}} = \llbracket N \rrbracket_{\rho_1}^{\mathcal{M}} \& \llbracket M \rrbracket_{\rho_2}^{\mathcal{N}} = \llbracket N \rrbracket_{\rho_2}^{\mathcal{N}} \\ &\Leftrightarrow \mathcal{M}, \rho_1 \models M = N \& \mathcal{N}, \rho_2 \models M = N. \end{aligned}$$

Hence for closed terms M, N

$$\mathcal{M} \times \mathcal{N} \models M = N \Leftrightarrow \mathcal{M} \models M = N \& \mathcal{N} \models M = N. \blacksquare$$

Polynomial models

3A.22. DEFINITION. (i) We introduce for each $m \in \mathcal{M}(A)$ a new constant $\underline{m} : A$, for each type A we choose a set of variables

$$x_0^A, x_1^A, x_2^A, \dots,$$

and let $\underline{\mathcal{M}}$ be the set of all correctly typed applicative combinations of these typed constants and variables.

(ii) For a valuation $\rho : \text{Var} \rightarrow \mathcal{M}$ define the map $((-))_\rho = ((-))_{\rho}^{\underline{\mathcal{M}}} : \underline{\mathcal{M}} \rightarrow \mathcal{M}$ by

$$\begin{aligned} ((x))_\rho &\triangleq \rho(x); \\ ((\underline{m}))_\rho &\triangleq m; \\ ((PQ))_\rho &\triangleq ((P))_\rho ((Q))_\rho. \end{aligned}$$

(iii) Define

$$P \sim_{\mathcal{M}} Q \iff \forall \rho ((P))_\rho = ((Q))_\rho,$$

where ρ ranges over valuations in \mathcal{M} .

3A.23. LEMMA. (i) $\sim_{\mathcal{M}}$ is an equivalence relation satisfying $\underline{d}\underline{e} \sim_{\mathcal{M}} \underline{d}\underline{e}$.

(ii) For all $P, Q \in \underline{\mathcal{M}}$ one has

$$P_1 \sim_{\mathcal{M}} P_2 \Leftrightarrow \forall Q_1, Q_2 \in \underline{\mathcal{M}} [Q_1 \sim_{\mathcal{M}} Q_2 \Rightarrow P_1 Q_1 \sim_{\mathcal{M}} P_2 Q_2].$$

PROOF. Note that P, Q can take all values in $\mathcal{M}(A)$ and apply extensionality. ■

3A.24. DEFINITION. Let \mathcal{M} be a typed applicative structure. The *polynomial structure* over \mathcal{M} is $\mathcal{M}^* = (|\mathcal{M}^*|, \text{app})$ defined by

$$\begin{aligned} |\mathcal{M}^*| &\triangleq \underline{\mathcal{M}} / \sim_{\mathcal{M}} \equiv \{[P]_{\sim_{\mathcal{M}}} \mid P \in \underline{\mathcal{M}}\}, \\ \text{app } [P]_{\sim_{\mathcal{M}}} [Q]_{\sim_{\mathcal{M}}} &\triangleq [PQ]_{\sim_{\mathcal{M}}}. \end{aligned}$$

By Lemma 3A.23(ii) this is well defined.

Working with \mathcal{M}^* it is often convenient to use as elements those of $\underline{\mathcal{M}}$ and reason about them modulo $\sim_{\mathcal{M}}$.

3A.25. PROPOSITION. (i) $\mathcal{M} \subseteq \mathcal{M}^*$ by the embedding morphism $i \triangleq \lambda d.[d] : \mathcal{M} \rightarrow \mathcal{M}^*$.

(ii) The embedding i can be extended to an embedding $i : \underline{\mathcal{M}} \rightarrow \mathcal{M}^*$.

(iii) There exists an isomorphism $G : \mathcal{M}^* \cong \mathcal{M}^{**}$.

PROOF. (i) It is easy to show that i is injective and satisfies

$$i(de) = i(d) \cdot_{\mathcal{M}^*} i(e).$$

(ii) Define

$$\begin{aligned} i'(x) &\triangleq x \\ i'(\underline{m}) &\triangleq [\underline{m}] \\ i'(d_1 d_2) &\triangleq i'(d_1) i'(d_2). \end{aligned}$$

We write again i for i' .

(iii) By definition $\underline{\mathcal{M}}$ is the set of all typed applicative combinations of typed variables x^A and constants \underline{m}^A and $\underline{\mathcal{M}}^*$ is the set of all typed applicative combinations of typed variables y^A and constants $(\underline{m}^*)^A$. Define a map $\underline{\mathcal{M}} \rightarrow \underline{\mathcal{M}}^*$ also denoted by G as follows.

$$\begin{aligned} G(\underline{m}) &\triangleq [\underline{m}] \\ G(x_{2i}) &\triangleq [x_i] \\ G(x_{2i+1}) &\triangleq y_i. \end{aligned}$$

Then we have

- (1) $P \sim_{\mathcal{M}} Q \Rightarrow G(P) \sim_{\mathcal{M}^*} G(Q)$.
- (2) $G(P) \sim_{\mathcal{M}^*} G(Q) \Rightarrow P \sim_{\mathcal{M}} Q$.
- (3) $\forall Q \in \underline{\mathcal{M}}^* \exists P \in \underline{\mathcal{M}} [G(P) \sim Q]$.

Therefore G induces the required isomorphism on the equivalence classes. ■

3A.26. DEFINITION. Let $P \in \underline{\mathcal{M}}$ and let x be a variable. We say that

P 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 $((P))_{\rho_1} = ((P))_{\rho_2}$.

3A.27. LEMMA. If P does not depend on x , then $P \sim_{\mathcal{M}} P[x:=Q]$ for all $Q \in \underline{\mathcal{M}}$.

PROOF. First show that $((P[x := Q]))_{\rho} = ((P))_{\rho[x := ((Q))_{\rho}]}$, in analogy to Lemma 3A.13(i). Now suppose P does not depend on x . Then

$$\begin{aligned} ((P[x := Q]))_{\rho} &= ((P))_{\rho[x := ((Q))_{\rho}]} \\ &= ((P))_{\rho}, \quad \text{as } P \text{ does not depend on } x. \blacksquare \end{aligned}$$

3A.28. PROPOSITION. Let \mathcal{M} be a typed applicative structure. Then

- (i) \mathcal{M} is a typed λ -model \Leftrightarrow for each $P \in \mathcal{M}^*$ and variable x of $\underline{\mathcal{M}}$ there exists an $F \in \mathcal{M}^*$ not depending on x such that $F[x] = P$.
- (ii) \mathcal{M} is a typed λ -model $\Rightarrow \mathcal{M}^*$ is a typed λ -model.

PROOF. (i) Choosing representatives for $P, F \in \mathcal{M}^*$ we show

\mathcal{M} is a typed λ -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 $Fx \sim_{\mathcal{M}} P$.

(\Rightarrow) Let \mathcal{M} be a typed λ -model and let P be given. We treat an illustrative example, e.g. $P \equiv fx^0y^0$, with $f \in \mathcal{M}(1_2)$. We take $F \equiv \llbracket \lambda yz_fx.z_fxy \rrbracket yf$. Then

$$((Fx))_{\rho} = \llbracket \lambda yz_fx.z_fxy \rrbracket \rho(y)f\rho(x) = f\rho(x)\rho(y) = ((fxy))_{\rho},$$

hence indeed $Fx \sim_{\mathcal{M}} fxy$. In general for each constant \underline{d} in P we take a variable z_d and define $F \equiv \llbracket \lambda \vec{y} \vec{z}_d \vec{x}. P \rrbracket \vec{y} \vec{f}$.

(\Leftarrow) We show $\forall M \in \Lambda_{\rightarrow}(A) \exists P_M \in \underline{\mathcal{M}}(A) \forall \rho. \llbracket M \rrbracket_{\rho} = ((P_M))_{\rho}$, by induction on $M : A$. For M being a variable or application this is trivial. For $M = \lambda x.N$, we know by the induction hypothesis that $\llbracket N \rrbracket_{\rho} = ((P_N))_{\rho}$ for all ρ . By assumption there is an F not depending on x such that $Fx \sim_{\mathcal{M}} P_N$. Then

$$((F))_{\rho} d = ((Fx))_{\rho[x := d]} = ((P_N))_{\rho[x := d]} =_{\text{IH}} \llbracket N \rrbracket_{\rho[x := d]}.$$

Hence $\llbracket \lambda x.N \rrbracket_{\rho} = ((F))_{\rho}$. So indeed $\llbracket M \rrbracket_{\rho} \downarrow$ for every ρ such that $\text{FV}(M) \subseteq \text{dom}(\rho)$. Hence \mathcal{M} is a typed λ -model.

(ii) By (i) \mathcal{M}^* is a λ -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 λ -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 $Ny \sim_{\mathcal{M}^*} M$. 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 λ -model: $M' = Nx$, so $Ny = M$. ■

3A.29. PROPOSITION. *If \mathcal{M} is a typed λ -model, then $\text{Th}(\mathcal{M}^*) = \text{Th}(\mathcal{M})$.*

PROOF. Do exercise 3F.5. ■

3A.30. REMARK. In general for type structures $\mathcal{M}^* \times \mathcal{N}^* \not\cong (\mathcal{M} \times \mathcal{N})^*$, but the isomorphism holds in case \mathcal{M}, \mathcal{N} are typed λ -models.

Semantics for type assignment à la Curry

Now we will employ models of untyped λ -calculus in order to give a semantics for $\lambda_{\rightarrow}^{\text{Cu}}$. The idea, due to [Scott \[1975a\]](#), is to interpret a type $A \in \mathbb{T}^{\mathbb{A}}$ as a subset of an untyped λ -model in such a way that it contains all the interpretations of the untyped λ -terms $M \in \Lambda(A)$. As usual one has to pay attention to $\text{FV}(M)$.

3A.31. DEFINITION. (i) An *applicative structure* is a pair $\langle \mathcal{D}, \cdot \rangle$, consisting of a set \mathcal{D} together with a binary operation $\cdot : \mathcal{D} \times \mathcal{D} \rightarrow \mathcal{D}$ on it.

(ii) An *(untyped) λ -model* for the untyped λ -calculus is of the form

$$\mathcal{D} = \langle \mathcal{D}, \cdot, [\]^{\mathcal{D}} \rangle,$$

where $\langle \mathcal{D}, \cdot \rangle$ is an applicative structure and $[\]^{\mathcal{D}} : \Lambda \times \text{Env}_{\mathcal{D}} \rightarrow \mathcal{D}$ satisfies the following.

- (1) $[\![x]\!]_{\rho}^{\mathcal{D}} = \rho(x);$
- (2) $[\![MN]\!]_{\rho}^{\mathcal{D}} = [\![M]\!]_{\rho}^{\mathcal{D}} \cdot [\![N]\!]_{\rho}^{\mathcal{D}};$
- (3) $[\![\lambda x. M]\!]_{\rho}^{\mathcal{D}} = [\![\lambda y. M[x := y]]\!]_{\rho}^{\mathcal{D}}, \quad (\alpha)$
provided $y \notin \text{FV}(M)$;
- (4) $\forall d \in \mathcal{D}. [\![M]\!]_{\rho[x:=d]}^{\mathcal{D}} = [\![N]\!]_{\rho[x:=d]}^{\mathcal{D}} \Rightarrow [\![\lambda x. M]\!]_{\rho}^{\mathcal{D}} = [\![\lambda x. N]\!]_{\rho}^{\mathcal{D}}; \quad (\xi)$
- (5) $\rho \upharpoonright \text{FV}(M) = \rho' \upharpoonright \text{FV}(M) \Rightarrow [\![M]\!]_{\rho}^{\mathcal{D}} = [\![M]\!]_{\rho'}^{\mathcal{D}};$
- (6) $[\![\lambda x. M]\!]_{\rho}^{\mathcal{D}} \cdot d = [\![M]\!]_{\rho[x:=d]}^{\mathcal{D}}. \quad (\beta)$

We will write $[\]_{\rho}$ for $[\]_{\rho}^{\mathcal{D}}$ if there is little danger of confusion.

Note that by (5) for closed terms the interpretation does not depend on the ρ .

3A.32. DEFINITION. Let \mathcal{D} be a λ -model and let $\rho \in \text{Env}_{\mathcal{D}}$ be an environment in \mathcal{D} . Let $M, N \in \Lambda$ be untyped λ -terms and let T be a set of equations between λ -terms.

(i) We say that *\mathcal{D} with environment ρ satisfies the equation $M = N$* , notation

$$\mathcal{D}, \rho \models M = N,$$

if $[\![M]\!]_{\rho}^{\mathcal{D}} = [\![N]\!]_{\rho}^{\mathcal{D}}$.

(ii) We say that *\mathcal{D} with environment ρ satisfies T* , notation

$$\mathcal{D}, \rho \models T,$$

if $\mathcal{D}, \rho \models M = N$, for all $(M = N) \in T$.

(iii) We define \mathcal{D} satisfies T , notation

$$\mathcal{D} \models T$$

if for all ρ one has $\mathcal{D}, \rho \models T$. If the set T consists of equations between closed terms, then the ρ is irrelevant.

(iv) Define that T satisfies equation $M = N$, notation

$$T \models M = N$$

if for all \mathcal{D} and $\rho \in \text{Env}_{\mathcal{D}}$ one has

$$\mathcal{D}, \rho \models T \Rightarrow \mathcal{D}, \rho \models M = N.$$

3A.33. THEOREM (Completeness theorem). *Let $M, N \in \Lambda$ be arbitrary and let T be a set of equations. Then*

$$T \vdash_{\lambda\beta\eta} M = N \Leftrightarrow T \models M = N.$$

PROOF. (\Rightarrow) ('Soundness') By induction on the derivation of $T \vdash M = N$.

(\Leftarrow) ('Completeness' proper) By taking the (extensional open) term model of T , see B[1984], 4.1.17. ■

Following Scott [1975a] a λ -model gives rise to a unified interpretation of λ -terms $M \in \Lambda$ and types $A \in \mathbb{T}^{\mathbb{A}}$. The terms will be interpreted as elements of \mathcal{D} and the types as subsets of \mathcal{D} .

3A.34. DEFINITION. Let \mathcal{D} be a λ -model. On the powerset $\mathcal{P}(D)$ one can define for $X, Y \in \mathcal{P}(D)$ the element $(X \Rightarrow Y) \in \mathcal{P}(\mathcal{D})$ as follows.

$$(X \Rightarrow Y) \triangleq \{d \in D \mid d.X \subseteq Y\} \triangleq \{d \in \mathcal{D} \mid \forall x \in X. (d \cdot x) \in Y\}.$$

3A.35. DEFINITION. Let \mathcal{D} be a λ -model. Given a type environment $\xi : \mathbb{A} \rightarrow \mathcal{P}(\mathcal{D})$, the *interpretation* of an $A \in \mathbb{T}^{\mathbb{A}}$ into $\mathcal{P}(\mathcal{D})$, notation $\llbracket A \rrbracket_{\xi}$, is defined as follows.

$$\begin{aligned} \llbracket \alpha \rrbracket_{\xi} &\triangleq \xi(\alpha), & \text{for } \alpha \in \mathbb{A}; \\ \llbracket A \rightarrow B \rrbracket_{\xi} &\triangleq \llbracket A \rrbracket_{\xi} \Rightarrow \llbracket B \rrbracket_{\xi}. \end{aligned}$$

3A.36. DEFINITION. Let \mathcal{D} be a λ -model and let $M \in \Lambda, A \in \mathbb{T}^{\mathbb{A}}$. Let ρ, ξ range over term and type environments, respectively.

(i) We say that \mathcal{D} with ρ, ξ satisfies the type assignment $M : A$, notation

$$\mathcal{D}, \rho, \xi \models M : A$$

if $\llbracket M \rrbracket_{\rho} \in \llbracket A \rrbracket_{\xi}$.

(ii) Let Γ be a type assignment basis. Then

$$\mathcal{D}, \rho, \xi \models \Gamma \triangleleft \text{ for all } (x:A) \in \Gamma \text{ one has } \mathcal{D}, \rho, \xi \models x : A.$$

(iii) $\Gamma \models M : A \Leftrightarrow \forall \mathcal{D}, \rho, \xi [\mathcal{D}, \rho, \xi \models \Gamma \Rightarrow \mathcal{D}, \rho, \xi \models M : A]$.

3A.37. PROPOSITION. *Let Γ, M, A respectively range over bases, untyped terms and types in $\mathbb{T}^{\mathbb{A}}$. Then*

$$\Gamma \vdash_{\lambda_{\mathbb{A}}}^{C_u} M : A \Leftrightarrow \Gamma \models M : A.$$

PROOF. (\Rightarrow) By induction on the length of proof.

(\Leftarrow) This has been proved independently in Hindley [1983] and Barendregt, Coppo, and Dezani-Ciancaglini [1983]. See Corollary 17A.11. ■

3B. Lambda theories and term models

In this Section we treat consistent sets of equations between terms of the same type and their term models.

3B.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, \dots we write constants as c, d, e, \dots , or being explicit as c^A, d^A, e^A, \dots . The letters $\mathcal{C}, \mathcal{D}, \dots$ range over sets of constants (of varying types).

(ii) Let \mathcal{D} be a set of constants with types in \mathbb{T}^0 . Write $\Lambda_{\rightarrow}[\mathcal{D}](A)$ for the set of open terms of type A , possibly containing constants in \mathcal{D} . Moreover

$$\Lambda_{\rightarrow}[\vec{\mathcal{D}}] \triangleq \cup_{A \in \mathbb{T}} \Lambda_{\rightarrow}[\vec{\mathcal{D}}](A).$$

(iii) Similarly $\Lambda_{\rightarrow}^o[\mathcal{D}](A)$ and $\Lambda_{\rightarrow}^o[\mathcal{D}]$ consist of closed terms possibly containing the constants in \mathcal{D} .

(iv) An *equation over \mathcal{D}* (i.e. between closed λ -terms with constants from \mathcal{D}) is of the form $M = N$ with $M, N \in \Lambda_{\rightarrow}^o[\mathcal{D}]$ of the same type.

(v) A term $M \in \Lambda_{\rightarrow}[\mathcal{D}]$ is *pure* if it does not contain constants from \mathcal{D} , i.e. if $M \in \Lambda_{\rightarrow}$. In this subsection we will consider sets of equations over \mathcal{D} . When writing $M = N$, we implicitly assume that M, N have the same type.

3B.2. DEFINITION. Let \mathcal{E} be a set of equations over \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)
$\lambda x.Mx = M$, if $x \notin \text{FV}(M)$	(η)
$\frac{}{M = N}$, if $(M = N) \in \mathcal{E}$	(ε)
$M = M$	(reflexivity)
$\frac{M = N}{N = M}$	(symmetry)
$\frac{M = N \quad N = L}{M = L}$	(transitivity)
$\frac{M = N}{MZ = NZ}$	(R-congruence)
$\frac{MZ = NZ}{M = N}$	(L-congruence)
$\frac{MZ = NZ \quad M = N}{\lambda x.M = \lambda x.N}$	(ξ)

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.

3B.3. REMARK. A typed lambda theory always is a $\lambda\beta\eta$ -theory.

3B.4. NOTATION. (i) $\mathcal{E}^+ \triangleq \{M = N \mid \mathcal{E} \vdash M = N\}$.

(ii) For $A \in \mathbb{T}^0$ write $\mathcal{E}(A) \triangleq \{M = N \mid (M = N) \in \mathcal{E} \& M, N \in \Lambda_{\rightarrow}[\mathcal{D}](A)\}$.

(iii) $\mathcal{E}_{\beta\eta} \triangleq \emptyset^+$.

3B.5. PROPOSITION. If $Mx =_{\mathcal{E}} Nx$, with $x \notin \text{FV}(M) \cup \text{FV}(N)$, then $M =_{\mathcal{E}} N$.

PROOF. Use (ξ) and (η) . ■

3B.6. DEFINITION. Let \mathcal{M} be a typed λ -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

$$\forall \mathcal{M}. [\mathcal{M} \models \mathcal{E} \Rightarrow \mathcal{M} \models M = N].$$

3B.7. 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 3A.17. The rules reflexivity, symmetry, transitivity and L,R-congruence are trivial to treat. The case falling under the rule (ξ) follows from Lemma 3A.18. ■

From non-trivial models one can obtain typed lambda theories.

3B.8. PROPOSITION. Let \mathcal{M} be a non-trivial typed λ -model.

(i) $\mathcal{M} \models \mathcal{E} \Rightarrow \mathcal{E}$ is consistent.

(ii) $\text{Th}(\mathcal{M})$ 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 $d = (\lambda xy.x)d = (\lambda xy.y)d = e$ for arbitrary d, e . Hence \mathcal{M} is trivial.

(ii) Clearly $\mathcal{M} \models \text{Th}(\mathcal{M})$. Hence by (i) $\text{Th}(\mathcal{M})$ is consistent. If $\text{Th}(\mathcal{M}) \vdash M = N$, then by soundness $\mathcal{M} \models M = N$, and therefore $(M = N) \in \text{Th}(\mathcal{M})$. ■

The full type structure over a finite set yields an interesting λ -theory.

Term models

3B.9. DEFINITION. Let \mathcal{D} be a set of constants of various types in \mathbb{T}^0 and let \mathcal{E} be a set of equations over \mathcal{D} . Define the type structure $\mathcal{M}_{\mathcal{E}}$ by

$$\mathcal{M}_{\mathcal{E}}(A) \triangleq \{[M]_{\mathcal{E}} \mid M \in \Lambda_{\rightarrow}[\mathcal{D}](A)\},$$

where $[M]_{\mathcal{E}}$ is the equivalence class modulo the congruence relation $=_{\mathcal{E}}$. Define the binary operator \cdot as follows.

$$[M]_{\mathcal{E}} \cdot [N]_{\mathcal{E}} \triangleq [MN]_{\mathcal{E}}.$$

This is well-defined, because $=_{\mathcal{E}}$ is a congruence. We often will suppress \cdot .

3B.10. PROPOSITION. (i) $(\mathcal{M}_{\mathcal{E}}, \cdot)$ is a typed applicative 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 $\{\vec{x}\} = \text{FV}(M)$ and the \vec{N} are determined by $\rho(x_i) = [N_i]_{\mathcal{E}}$.

(iii) $\mathcal{M}_\mathcal{E}$ is a typed model, called the open term model of \mathcal{E} .

PROOF. (i) We need to verify extensionality.

$$\begin{aligned} \forall d \in \mathcal{M}_\mathcal{E}. [M]d = [N]d &\Rightarrow [M][x] = [N][x], \quad \text{for a fresh } x, \\ &\Rightarrow [Mx] = [Nx] \\ &\Rightarrow Mx =_\mathcal{E} Nx \\ &\Rightarrow M =_\mathcal{E} N, \quad \text{by } (\xi), (\eta) \text{ and (transitivity),} \\ &\Rightarrow [M] = [N]. \end{aligned}$$

(ii) We show that $\llbracket M \rrbracket_\rho$ defined as $[M[x := \vec{N}]]_\mathcal{E}$ satisfies the conditions in Definition 3A.9(ii).

$$\begin{aligned} \llbracket x \rrbracket_\rho &= [x[x := N]]_\mathcal{E}, \quad \text{with } \rho(x) = [N]_\mathcal{E}, \\ &= [N]_\mathcal{E} \\ &= \rho(x); \\ \llbracket PQ \rrbracket_\rho &= [(PQ)[\vec{x} := \vec{N}]]_\mathcal{E} \\ &= [P[\vec{x} := \vec{N}]Q[\vec{x} := \vec{N}]]_\mathcal{E} \\ &= [P[\vec{x} := \vec{N}]]_\mathcal{E}[[Q[\vec{x} := \vec{N}]]_\mathcal{E}] \\ &= \llbracket P \rrbracket_\rho \llbracket Q \rrbracket_\rho; \\ \llbracket \lambda y.P \rrbracket_\rho [Q]_\mathcal{E} &= [(\lambda y.P)[\vec{x} := \vec{N}]]_\mathcal{E} [Q]_\mathcal{E} \\ &= [\lambda y.P[\vec{x} := \vec{N}]]_\mathcal{E} [Q]_\mathcal{E} \\ &= [P[\vec{x} := \vec{N}][y := Q]]_\mathcal{E} \\ &= [P[\vec{x}, y := \vec{N}, Q]]_\mathcal{E}, \quad \text{because } y \notin \text{FV}(\vec{N}) \text{ by the} \\ &\quad \text{variable convention and } y \notin \{\vec{x}\}, \\ &= \llbracket P \rrbracket_{\rho[y := [Q]_\mathcal{E}]} \end{aligned}$$

(iii) As $\llbracket M \rrbracket_\rho$ is always defined by (ii). ■

3B.11. COROLLARY. (i) $\mathcal{M}_\mathcal{E} \models M = N \Leftrightarrow M =_\mathcal{E} N$.

(ii) $\mathcal{M}_\mathcal{E} \models \mathcal{E}$.

PROOF. (i) (\Rightarrow) Suppose $\mathcal{M}_\mathcal{E} \models M = N$. Then $\llbracket M \rrbracket_\rho = \llbracket N \rrbracket_\rho$ for all ρ . Choosing $\rho(x) = [x]_\mathcal{E}$ one obtains $\llbracket M \rrbracket_\rho = [M[\vec{x} := \vec{x}]]_\mathcal{E} = [M]_\mathcal{E}$, and similarly for N , hence $[M]_\mathcal{E} = [N]_\mathcal{E}$ and therefore $M =_\mathcal{E} N$.

$$\begin{aligned} (\Leftarrow) \quad M =_\mathcal{E} N &\Rightarrow M[\vec{x} := \vec{P}] =_\mathcal{E} N[\vec{x} := \vec{P}] \\ &\Rightarrow [M[\vec{x} := \vec{P}]]_\mathcal{E} = [N[\vec{x} := \vec{P}]]_\mathcal{E} \\ &\Rightarrow \llbracket M \rrbracket_\rho = \llbracket N \rrbracket_\rho \\ &\Rightarrow \mathcal{M}_\mathcal{E} \models M = N. \end{aligned}$$

(ii) If $M = N \in \mathcal{E}$, then $M =_\mathcal{E} N$, hence $\mathcal{M}_\mathcal{E} \models M = N$, by (i). ■

Using this Corollary we obtain completeness in a simple way.

3B.12. THEOREM (Completeness). $\mathcal{E} \vdash M = N \Leftrightarrow \mathcal{M}_\mathcal{E} \models M = N$.

PROOF. (\Rightarrow) By soundness, Proposition 3B.7.

$$\begin{aligned}
 (\Leftarrow) \quad \mathcal{E} \models M = N &\Rightarrow \mathcal{M}_{\mathcal{E}} \models M = N, \text{ as } \mathcal{M}_{\mathcal{E}} \models \mathcal{E}, \\
 &\Rightarrow M =_{\mathcal{E}} N \\
 &\Rightarrow \mathcal{E} \vdash M = N. \blacksquare
 \end{aligned}$$

3B.13. COROLLARY. Let \mathcal{E} be a set of equations. Then

$$\mathcal{E} \text{ has a non-trivial model} \Leftrightarrow \mathcal{E} \text{ is consistent.}$$

PROOF. (\Rightarrow) By Proposition 3B.8. (\Leftarrow) Suppose that $\mathcal{E} \not\models x^0 = y^0$. Then by the Theorem one has $\mathcal{E} \not\models x^0 = y^0$. 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. \blacksquare

If \mathcal{D} contains enough constants, then one can similarly define the applicative structure $\mathcal{M}_{\mathcal{E}[\mathcal{D}]}^\emptyset$ by restricting $\mathcal{M}_{\mathcal{E}}$ to closed terms. See section 3.3.

Constructing Theories

The following result is due to [Jacopini \[1975\]](#).

3B.14. PROPOSITION. Let \mathcal{E} be a set of equations between closed terms in $\Lambda_{\rightarrow}^\emptyset[\mathcal{D}]$. Then $\mathcal{E} \vdash M = N$ if for some $n \in \mathbb{N}$, $F_1, \dots, F_n \in \Lambda_{\rightarrow}[\mathcal{D}]$ and $P_1 = Q_1, \dots, P_n = Q_n \in \mathcal{E}$ one has $\text{FV}(F_i) \subseteq \text{FV}(M) \cup \text{FV}(N)$ and

$$\left. \begin{array}{lll} M &=_{\beta\eta} & F_1 P_1 Q_1 \\ F_1 Q_1 P_1 &=_{\beta\eta} & F_2 P_2 Q_2 \\ &\dots& \\ F_{n-1} Q_{n-1} P_{n-1} &=_{\beta\eta} & F_n P_n Q_n \\ F_n Q_n P_n &=_{\beta\eta} & N. \end{array} \right\} \quad (1)$$

This scheme (1) is called a [Jacopini tableau](#) and the sequence F_1, \dots, 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 $\beta\eta$ -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 as list of witnesses just 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, \dots, F_n , then the list for $M = N$ is F'_1, \dots, F'_n with $F'_i \equiv \lambda ab.F_i abZ$ (respectively $F'_i \equiv \lambda ab.Z(F_i ab)$). 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, \dots, F_n , then the new list is F'_1, \dots, F'_n with $F'_i \equiv \lambda p q x.F_i p q$. Here we use that the equations in \mathcal{E} are between closed terms. \blacksquare

Remember that $\text{true} \equiv \lambda xy.x$, $\text{false} \equiv \lambda xy.y$ both having type $1_2 = 0 \rightarrow 0 \rightarrow 0$.

3B.15. LEMMA. Let \mathcal{E} be a set of equations over \mathcal{D} . Then

$$\mathcal{E} \text{ is consistent} \Leftrightarrow \mathcal{E} \not\models \text{true} = \text{false}.$$

PROOF. (\Leftarrow) By definition. (\Rightarrow) Suppose $\mathcal{E} \vdash \lambda xy.x = \lambda xy.y$. Then $\mathcal{E} \vdash P = Q$ for arbitrary $P, Q \in \Lambda_{\rightarrow}(0)$. But then for arbitrary terms M, N of the same type $A = A_1 \rightarrow \dots \rightarrow A_n \rightarrow 0$ one has $\mathcal{E} \vdash M\vec{z} = N\vec{z}$ for fresh $\vec{z} = z_1, \dots, z_n$ of the right type, hence $\mathcal{E} \vdash M = N$, by Proposition 3B.5. \blacksquare

3B.16. DEFINITION. Let $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A)$ be closed terms of type A .

(i) M is *inconsistent* with N , notation $M \# N$, if

$$\{M = N\} \vdash \text{true} = \text{false}.$$

(ii) M is *separable from* N , notation $M \perp N$, iff for some $F \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A \rightarrow 1_2)$

$$FM = \text{true} \& FN = \text{false}.$$

The following result, stating that inconsistency implies separability, is not true for the untyped lambda calculus: the equation $K = YK$ is inconsistent, but K and YK are not separable, as follows from the Genericity Lemma, see B[1984] Proposition 14.3.24.

3B.17. PROPOSITION. Let $M, N \in \Lambda_{\rightarrow}^{\emptyset}(A)$ be closed pure terms of type A . Then

$$M \# N \Leftrightarrow M \perp N.$$

PROOF. (\Leftarrow) Trivially separability implies inconsistency.

(\Rightarrow) Suppose $\{M = N\} \vdash \text{true} = \text{false}$. Then also $\{M = N\} \vdash x = y$. Hence by Proposition 3B.14 one has

$$\begin{aligned} x &=_{\beta\eta} F_1 MN \\ F_1 NM &=_{\beta\eta} F_2 MN \\ &\dots \\ F_n NM &=_{\beta\eta} y. \end{aligned}$$

Let n be minimal for which this is possible. We can assume that the F_i are all pure terms with $\text{FV}(F_i) \subseteq \{x, y\}$ at most. The nf of $F_1 NM$ 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 the nf of $F_1 MM$. It must be either x or y .

Case 1: $F_1 MM =_{\beta\eta} x$. Then set $F \equiv \lambda a x y. F_1 a M$ and we have $FM =_{\beta\eta} \text{true}$ and $FN =_{\beta\eta} \text{false}$.

Case 2: $F_1 MM =_{\beta\eta} y$. Then set $F \equiv \lambda a x y. F_1 M a$ and we have $FM =_{\beta\eta} \text{false}$ and $FN =_{\beta\eta} \text{true}$. ■

This Proposition does not hold for $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$, see Exercise 3F.2.

3B.18. COROLLARY. Let \mathcal{E} be a set of equations over $\mathcal{D} = \emptyset$. 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. ■

In the untyped theory λ the set $\mathcal{H} = \{M = N \mid M, N \text{ are closed unsolvable}\}$ is consistent and has a unique maximal consistent extension \mathcal{H}^* , see B[1984]. The following result is similar for λ_{\rightarrow} , as there are no unsolvable terms.

3B.19. THEOREM. Let

$$\mathcal{E}_{\max} \triangleq \{M=N \mid M, N \in \Lambda_{\rightarrow}^{\emptyset} \text{ and } M, N \text{ are not separable}\}.$$

Then this is the unique maximally consistent set of equations.

PROOF. By the corollary this set is consistent. By Proposition 3B.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.

3C. Syntactic and semantic logical relations

In this section we work in $\lambda_{\rightarrow}^{0,\text{Ch}}$. We 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

3C.1. DEFINITION. Let n be a fixed natural number and let $\vec{\mathcal{D}} = \mathcal{D}_1, \dots, \mathcal{D}_n$ be sets of constants of various given types.

(i) R is called an (n -ary) *family of (syntactic) relations* (or sometimes just a *(syntactic) relation*) on $\Lambda_{\rightarrow}[\vec{\mathcal{D}}]$, if $R = \{R_A\}_{A \in \mathbb{T}}$ and for $A \in \mathbb{T}$

$$R_A \subseteq \Lambda_{\rightarrow}[\mathcal{D}_1](A) \times \dots \times \Lambda_{\rightarrow}[\mathcal{D}_n](A).$$

If we want to make the sets of constants explicit, we say that R is a relation *on terms from* $\mathcal{D}_1, \dots, \mathcal{D}_n$.

(ii) Such an R is called a *(syntactic) logical relation* if

$$\forall A, B \in \mathbb{T} \forall M_1 \in \Lambda_{\rightarrow}[\mathcal{D}_1](A \rightarrow B), \dots, M_n \in \Lambda_{\rightarrow}[\mathcal{D}_n](A \rightarrow B).$$

$$R_{A \rightarrow B}(M_1, \dots, M_n) \Leftrightarrow \forall N_1 \in \Lambda_{\rightarrow}[\mathcal{D}_1](A) \dots N_n \in \Lambda_{\rightarrow}[\mathcal{D}_n](A) [R_A(N_1, \dots, N_n) \Rightarrow R_B(M_1 N_1, \dots, M_n N_n)].$$

(iii) R is called empty if $R_0 = \emptyset$.

Given $\vec{\mathcal{D}}$, a logical family $\{R_A\}$ is completely determined by R_0 . For $A \neq 0$ the R_A do depend on the choice of the $\vec{\mathcal{D}}$.

3C.2. LEMMA. *If R is a non-empty logical relation, then $\forall A \in \mathbb{T}^0. R_A \neq \emptyset$.*

PROOF. (For R unary.) By induction on A . Case $A = 0$. By assumption. Case $A = B \rightarrow C$. Then $R_{B \rightarrow C}(M) \Leftrightarrow \forall P \in \Lambda_{\rightarrow}(B). [R_B(P) \Rightarrow R_C(MP)]$. By the induction hypothesis one has $R_C(N)$, for some N . Then $M \equiv \lambda p.N \in \Lambda_{\rightarrow}(B \rightarrow C)$ is in R_A . ■

Even the empty logical relation is interesting.

3C.3. PROPOSITION. *Let R be the n -ary logical relation on $\Lambda_{\rightarrow}[\vec{\mathcal{D}}]$ determined by $R_0 = \emptyset$. Then*

$$\begin{aligned} R_A &= \Lambda_{\rightarrow}[\mathcal{D}_1](A) \times \dots \times \Lambda_{\rightarrow}[\mathcal{D}_n](A), && \text{if } \Lambda_{\rightarrow}^{\emptyset}(A) \neq \emptyset; \\ &= \emptyset, && \text{if } \Lambda_{\rightarrow}^{\emptyset}(A) = \emptyset. \end{aligned}$$

PROOF. For notational simplicity we take $n = 1$. By induction on A . If $A = 0$, then we are done, as $R_0 = \emptyset$ and $\Lambda_{\rightarrow}^{\emptyset}(0) = \emptyset$. If $A = A_1 \rightarrow \dots \rightarrow A_m \rightarrow 0$, then

$$\begin{aligned} R_A(M) &\Leftrightarrow \forall P_i \in R_{A_i}. R_0(M \vec{P}) \\ &\Leftrightarrow \forall P_i \in R_{A_i}. \perp, \end{aligned}$$

seeing R both as a relation and as a set, and ‘ \perp ’ stands for the false proposition. This last statement either is always the case, namely if

$$\begin{aligned} \exists i. R_{A_i} = \emptyset &\Leftrightarrow \exists i. \Lambda_{\rightarrow}^{\emptyset}(A_i) = \emptyset, && \text{by the induction hypothesis,} \\ &\Leftrightarrow \Lambda_{\rightarrow}^{\emptyset}(A) \neq \emptyset, && \text{by Proposition 2D.4.} \end{aligned}$$

Or else, namely if $\Lambda_{\rightarrow}^{\emptyset}(A) = \emptyset$, it is never the case, by the same reasoning. ■

3. TOOLS

3C.4. EXAMPLE. Let $n = 2$ and set $R_0(M, N) \Leftrightarrow M =_{\beta\eta} N$. Let R be the logical relation determined by R_0 . Then it is easily seen that for all A and $M, N \in \Lambda_{\rightarrow}[\vec{\mathcal{D}}](A)$ one has $R_A(M, N) \Leftrightarrow M =_{\beta\eta} N$.

3C.5. DEFINITION. (i) Let M, N be lambda terms. Then M is a *weak head expansion* of N , notation $M \rightarrow_{wh} N$, if $M \equiv (\lambda x.P)Q\vec{R}$ and $N \equiv P[x := Q]\vec{R}$.

(ii) A family R on $\Lambda_{\rightarrow}[\vec{\mathcal{D}}]$ is called *expansive* if R_0 is closed under coordinatewise weak head expansion, i.e. if $M'_i \rightarrow_{wh} M_i$ for $1 \leq i \leq n$, then

$$R_0(M_1, \dots, M_n) \Rightarrow R_0(M'_1, \dots, M'_n).$$

3C.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

$$M' \rightarrow_{wh} M \Rightarrow M'N \rightarrow_{wh} MN. \blacksquare$$

3C.7. EXAMPLE. This example prepares an alternative proof of the Church-Rosser property using logical relations.

(i) Let $M \in \Lambda_{\rightarrow}$. We say that $\beta\eta$ is *confluent* from M , notation $\downarrow_{\beta\eta} M$, if whenever $N_1 \beta\eta \leftarrow M \rightarrow_{\beta\eta} N_2$, then there exists a term L such that $N_1 \rightarrow_{\beta\eta} L \beta\eta \leftarrow N_2$. Define R_0 on $\Lambda_{\rightarrow}(0)$ by

$$R_0(M) \Leftrightarrow \beta\eta \text{ is confluent from } M.$$

Then R_0 determines a logical R which is expansive by the permutability of head contractions with internal ones.

(ii) Let R be the logical relation on Λ_{\rightarrow} generated from

$$R_0(M) \Leftrightarrow \downarrow_{\beta\eta} M.$$

Then for an arbitrary type $A \in \mathbb{T}$ one has

$$R_A(M) \Rightarrow \downarrow_{\beta\eta} M.$$

[Hint. Write $M \downarrow_{\beta\eta} N$ if $\exists Z [M \rightarrow_{\beta\eta} Z \beta\eta \leftarrow N]$. First show that for an arbitrary variable x 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

$$N_1x \downarrow_{\beta\eta} N_2x \Rightarrow N_1 \downarrow_{\beta\eta} N_2.$$

Then use induction on A .]

3C.8. DEFINITION. (i) Let $R \subseteq \Lambda_{\rightarrow}[\mathcal{D}_1](A) \times \dots \times \Lambda_{\rightarrow}[\mathcal{D}_n](A)$ and $*_1, \dots, *_n$

$$*_i : \text{Var}(A) \rightarrow \Lambda_{\rightarrow}[\mathcal{D}_i](A)$$

be substitutors, each $*$ applicable to all variables of all types. Write $R(*_1, \dots, *_n)$ if $R_A(x^{*_1}, \dots, x^{*_n})$ for each variable x of type A .

(ii) Define $R^* \subseteq \Lambda_{\rightarrow}[\mathcal{D}_1](A) \times \dots \times \Lambda_{\rightarrow}[\mathcal{D}_n](A)$ by

$$R_A^*(M_1, \dots, M_n) \stackrel{\Delta}{\Leftrightarrow} \forall *_1 \dots *_n [R(*_1, \dots, *_n) \Rightarrow R_A(M_1^{*_1}, \dots, M_n^{*_n})].$$

(iii) R is called *substitutive* if $R = R^*$, i.e.

$$R_A(M_1, \dots, M_n) \Leftrightarrow \forall *_1 \dots *_n [R(*_1, \dots, *_n) \Rightarrow R_A(M_1^{*_1}, \dots, M_n^{*_n})].$$

3C.9. LEMMA. *Let R be logical.*

(i) *Suppose that $R_0 \neq \emptyset$. Then for closed terms $M_1 \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}_1], \dots, M_n \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}_n]$*

$$R_A(M_1, \dots, M_n) \Leftrightarrow R_A^*(M_1, \dots, M_n).$$

(ii) For pure closed terms $M_1 \in \Lambda_{\rightarrow}^{\emptyset}, \dots, M_n \in \Lambda_{\rightarrow}^{\emptyset}$

$$R_A(M_1, \dots, M_n) \Leftrightarrow R_A^*(M_1, \dots, M_n).$$

(iii) For a substitutive R one has for arbitrary open $M_1, \dots, M_n, N_1, \dots, N_n$

$$R_A(M_1, \dots, M_n) \& R_B(N_1, \dots, N_n) \Rightarrow R_A(M_1[x^B := N_1], \dots, M_n[x^B := N_n]).$$

PROOF. (i) Clearly $R_A(\vec{M})$ implies $R_A^*(\vec{M})$, as the \vec{M} are closed. For the converse assume $R_A^*(\vec{M})$, that is $R_A(\vec{M}^*)$, for all substitutors $\vec{*}$ satisfying $R(\vec{*})$. As $R_0 \neq \emptyset$, we have $R_B \neq \emptyset$, for all $B \in \mathbb{T}^0$, by Lemma 3C.2. So we can take $\vec{*}_i$ such that $R_B(\vec{x}^{*_i})$, for all $x = x^B$. But then $R(*)$ and hence $R(\vec{M}^*)$, which is $R(\vec{M})$.

(ii) If $\Lambda_{\rightarrow}^{\emptyset}(A) = \emptyset$, then this set does not contain closed pure terms and we are done. If $\Lambda_{\rightarrow}^{\emptyset}(A) \neq \emptyset$, then by Lemma 3C.3 we have $R_A = (\Lambda_{\rightarrow}^{\emptyset}(A))^n$ and we are also done.

(iii) Since R is substitutive we have $R^*(\vec{M})$. Let $*_i = [x := N_i]$. Then $R(*_1, \dots, *_n)$ and hence $R(M_1[x := N_1], \dots, M_n[x := N_n])$. ■

Part (i) of this Lemma does not hold for $R_0 = \emptyset$ and $D_1 \neq \emptyset$. Take for example $D_1 = \{c^0\}$. Then vacuously $R_0^*(c^0)$, but not $R_0(c^0)$.

3C.10. EXERCISE. (CR for $\beta\eta$ via logical relations.) Let R be the logical relation on Λ_{\rightarrow} generated by $R_0(M)$ iff $\downarrow_{\beta\eta} M$. 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 $\downarrow_{\beta\eta} M$. The same holds for arbitrary open terms N : let $\{\vec{x}\} = \text{FV}(M)$, then

$$\begin{aligned} \lambda \vec{x}.N \text{ is closed} &\Rightarrow R(\lambda \vec{x}.N) \\ &\Rightarrow R((\lambda \vec{x}.N)\vec{x}), \quad \text{since } R(x_i), \\ &\Rightarrow R(N), \quad \text{since } R \text{ is closed under } \rightarrow_{\beta}, \\ &\Rightarrow \downarrow_{\beta\eta} N. \end{aligned}$$

Thus the Church-Rosser property holds for $\rightarrow_{\beta\eta}$.

3C.11. PROPOSITION. Let R be an arbitrary n -ary family on $\Lambda_{\rightarrow}[\vec{D}]$. 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 $R(x^*)$. Therefore $R^*(x)$.
- (ii) We have to prove

$$R^*(M) \Leftrightarrow \forall N \in \Lambda_{\rightarrow}[\vec{D}][R^*(N) \Rightarrow R^*(MN)].$$

(\Rightarrow) Assume $R^*(M) \& R^*(N)$ in order to show $R^*(MN)$. Let $*$ be a substitutor such that $R(*)$. Then

$$\begin{aligned} R^*(M) \& R^*(N) \Rightarrow R(M^*) \& R(N^*) \\ &\Rightarrow R(M^*N^*) \equiv R((MN)^*) \\ &\Rightarrow R^*(MN). \end{aligned}$$

(\Leftarrow) By the assumption and (i) we have

$$R^*(Mx), \quad (1)$$

where we choose x to be fresh. In order to prove $R^*(M)$ we have to show $R(M^*)$, whenever $R(*)$. Because R is logical 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 $M \rightarrow_{wh} M^w$ be a weak head reduction step. Then

$$\begin{aligned} R^*(M^w) &\Rightarrow R(M^{w*}) \equiv R(M^{*w}) \\ &\Rightarrow R(M^*) \\ &\Rightarrow R^*(M). \end{aligned}$$

(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 therefore we have $R^*(M^{*_1}) \equiv R^*(M)$. Conversely, assume $R^*(M)$, i.e.

$$\forall * [R(*) \Rightarrow R(M^*)], \quad (2)$$

in order to show $\forall *_1 [R^*(*_1) \Rightarrow R^*(M^{*_1})]$. Now

$$\begin{aligned} R^*(*_1) &\Leftrightarrow \forall *_2 [R(*_2) \Rightarrow R(*_1*_2)], \\ R^*(M^{*_1}) &\Leftrightarrow \forall *_2 [R(*_2) \Rightarrow R(M^{*_1*_2})]. \end{aligned}$$

Therefore by (2) applied to $*_1*_2$ we are done.

(v) Let R be logical and expansive. Assume $R^*(M)$. Then

$$\begin{aligned} R^*(N) &\Rightarrow R^*(M[x:=N]), \quad \text{since } R^* \text{ is substitutive,} \\ &\Rightarrow R^*((\lambda x.M)N), \quad \text{since } R^* \text{ is expansive.} \end{aligned}$$

Therefore $R^*(\lambda x.M)$ since R^* is logical. ■

3C.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 $M \in \Lambda_{\rightarrow}(A)$ one has*

$$R_A(M, \dots, M).$$

PROOF. By induction on M we show that $R_A(M, \dots, M)$.

Case $M \equiv x$. Then the statement follows from the assumption $R = R^*$ (substitutivity) and Proposition 3C.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 3C.11(v). ■

3C.13. COROLLARY. *Let R be an n -ary expansive logical relation. Then for all closed $M \in \Lambda_{\rightarrow}^{\emptyset}$ one has $R(M, \dots, M)$.*

PROOF. By Proposition 3C.11(ii), (iii), (iv) it follows that R^* is expansive, substitutive, and logical. Hence the theorem applied to R^* yields $R^*(M, \dots, M)$. Then we have $R(\vec{M})$, by Lemma 3C.9(ii). ■

The proof in Exercise 3C.10 was in fact an application of this Corollary. In the following Example we present the proof of weak normalization in Prawitz [1965].

3C.14. EXAMPLE. Let R be the logical relation determined by

$$R_0(M) \Leftrightarrow M \text{ is normalizable.}$$

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 3C.13 that each closed term is normalizable. Hence all terms are normalizable by taking closures. For strong normalization a similar proof breaks down. The corresponding R is not expansive.

3C.15. EXAMPLE. Now we ‘relativize’ the theory of logical relations to closed terms. A family of relations $S_A \subseteq \Lambda_{\rightarrow}^{\circ}[\mathcal{D}_1](A) \times \dots \times \Lambda_{\rightarrow}^{\circ}[\mathcal{D}_n](A)$ which satisfies

$$\begin{aligned} S_{A \rightarrow B}(M_1, \dots, M_n) &\Leftrightarrow \forall N_1 \in \Lambda_{\rightarrow}^{\circ}[\mathcal{D}_1](A) \dots N_n \in \Lambda_{\rightarrow}^{\circ}[\mathcal{D}_n](A) \\ [S_A(N_1, \dots, N_n)] &\Rightarrow S_B(M_1N_1, \dots, M_nN_n) \end{aligned}$$

can be lifted to a substitutive logical relation S^* on $\Lambda_{\rightarrow}[\mathcal{D}_1] \times \dots \times \Lambda_{\rightarrow}[\mathcal{D}_n]$ as follows. Define for substitutors $*_i : \text{Var}(A) \rightarrow \Lambda_{\rightarrow}^{\circ}[\mathcal{D}_i](A)$

$$S_A(*_1, \dots, *_n) \Leftrightarrow \forall x^A S_A(x^{*_1}, \dots, x^{*_n}).$$

Now define S^* as follows: for $M_i \in \Lambda_{\rightarrow}[\mathcal{D}_i](A)$

$$S_A^*(M_1, \dots, M_n) \Leftrightarrow \forall *_1 \dots *_n [S_A(*_1, \dots, *_n) \Rightarrow S_A(M_1^{*_1}, \dots, M_n^{*_n})].$$

Show that if S is closed under coordinatewise weak head expansions, then S^* is expansive.

The following definition is needed in order to relate the notions of logical relation and semantic logical relation, to be defined in 3C.21.

3C.16. DEFINITION. Let R be an $n + 1$ -ary family. The *projection* of R , notation $\exists R$, is the n -ary family defined by

$$\exists R(M_1, \dots, M_n) \Leftrightarrow \exists M_{n+1} \in \Lambda_{\rightarrow}[\mathcal{D}_{n+1}] R(M_1, \dots, M_{n+1}).$$

3C.17. PROPOSITION. (i) *The universal n-ary relation R^U is defined by*

$$R_A^U \triangleq \Lambda_{\rightarrow}[\mathcal{D}_1](A) \times \dots \times \Lambda_{\rightarrow}[\mathcal{D}_n](A).$$

This relation is logical, expansive and substitutive.

(ii) *Let $R = \{R_A\}_{A \in \text{PT}^0}, S = \{S_A\}_{A \in \text{PT}^0}$ with $R_A \subseteq \Lambda_{\rightarrow}[\mathcal{D}_1](A) \times \dots \times \Lambda_{\rightarrow}[\mathcal{D}_m](A)$ and $S_A \subseteq \Lambda_{\rightarrow}[\mathcal{E}_1](A) \times \dots \times \Lambda_{\rightarrow}[\mathcal{E}_n](A)$ be non-empty logical relations. Define*

$$(R \times S)_A \subseteq \Lambda_{\rightarrow}[\mathcal{D}_1](A) \times \dots \times \Lambda_{\rightarrow}[\mathcal{D}_m](A) \times \Lambda_{\rightarrow}[\mathcal{E}_1](A) \times \dots \times \Lambda_{\rightarrow}[\mathcal{E}_n](A)$$

by

$$(R \times S)_A(M_1, \dots, M_m, N_1, \dots, N_n) \stackrel{\Delta}{\Leftrightarrow} R_A(M_1, \dots, M_m) \& S_A(N_1, \dots, N_n).$$

Then $R \times S$ is a non-empty logical relation. If moreover R and S are both substitutive, then so is $R \times S$.

(iii) If R is an n -ary family and π is a permutation of $\{1, \dots, n\}$, then R^π defined by

$$R^\pi(M_1, \dots, M_n) \xrightarrow{\Delta} R(M_{\pi(1)}, \dots, M_{\pi(n)})$$

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 substitutive logical relation on terms from $\mathcal{D}_1, \dots, \mathcal{D}_n$ and let $\mathcal{D} \subseteq \cap_i \mathcal{D}_i$. Then the diagonal of R , notation R^Δ , defined by

$$R^\Delta(M) \xrightarrow{\Delta} R(M, \dots, M)$$

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.

$$\begin{aligned} (R \times S)_{A \rightarrow B}(M, N) &\Leftrightarrow R_{A \rightarrow B}(M) \& S_{A \rightarrow B}(N) \\ &\Leftrightarrow [\forall P. R_A(P) \Rightarrow R_B(MP)] \& \\ &\quad [\forall Q. R_A(Q) \Rightarrow R_B(NQ)] \\ &\Leftrightarrow \forall (P, Q). (R \times S)_A(P, Q) \Rightarrow (R \times S)_B(MP, NQ). \end{aligned}$$

For the last (\Leftarrow) one needs that the R, S are non-empty, and Lemma 3C.2. If both R, S are substitutive, then trivially so is $R \times S$.

(iii) Trivial.

(iv) We show for $n = 2$ that R^Δ is logical. We have

$$\begin{aligned} R^\Delta(M) &\Leftrightarrow R(M, M) \\ &\Leftrightarrow \forall N_1, N_2. R(N_1, N_2) \Rightarrow R(MN_1, MN_2) \\ &\Leftrightarrow \forall N. R(N, N) \Rightarrow R(MN, MN), \end{aligned} \tag{1}$$

where validity of the last equivalence is argued as follows. Direction (\Rightarrow) is trivial. As to (\Leftarrow), suppose (1) and $R(N_1, N_2)$, in order to show $R(MN_1, MN_2)$. By Proposition 3C.11(i) one has $R(x, x)$, for fresh 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

$$\forall P. [\exists R(P) \Rightarrow \exists R(MP)] \tag{2}$$

implies $\exists N \forall P, Q. [R(P, Q) \Rightarrow R(MP, NQ)]$. Again we have $R(x, x)$. Therefore by (2)

$$\exists N_1. R(Mx, N_1).$$

Choosing $N \equiv \lambda x. N_1$, we get $R^*(Mx, Nx)$, because R is substitutive. Then $R(P, Q)$ implies $R(MP, NQ)$, as in (iv). ■

The following property R states that an M essentially does not contain the constants from \mathcal{D} . Remember that a term $M \in \Lambda_\rightarrow[\mathcal{D}]$ is called *pure* iff $M \in \Lambda_\rightarrow$. The property $R(M)$ states that M is convertible to a pure term.

3C.18. PROPOSITION. Define for $M \in \Lambda_{\rightarrow}[\mathcal{D}](A)$

$$R_A^{\beta\eta}(M) \iff \exists N \in \Lambda_{\rightarrow}(A) M =_{\beta\eta} N.$$

Then

- (i) $R^{\beta\eta}$ is logical.
- (ii) $R^{\beta\eta}$ is expansive.
- (iii) $R^{\beta\eta}$ is substitutive.

PROOF. (i) If $R^{\beta\eta}(M)$ and $R^{\beta\eta}(N)$, then clearly $R^{\beta\eta}(MN)$. Conversely, suppose $\forall N [R^{\beta\eta}(N) \Rightarrow R^{\beta\eta}(MN)]$. Since obviously $R^{\beta\eta}(x)$ it follows that $R^{\beta\eta}(Mx)$ for fresh x . Hence there exists a pure $L =_{\beta\eta} Mx$. But then $\lambda x. L =_{\beta\eta} M$, hence $R^{\beta\eta}(M)$.

(ii) Trivial as $P \rightarrow_{\text{wh}} Q \Rightarrow P =_{\beta\eta} Q$.

(iii) We must show $R^{\beta\eta} = R^{\beta\eta*}$. Suppose $R^{\beta\eta}(M)$ and $R^{\beta\eta}(*)$. Then $M = N$, with N pure and hence $M^* = N^*$ is pure, so $R^{\beta\eta*}(M)$. Conversely, suppose $R^{\beta\eta*}(M)$. Then for $*$ with $x^* = x$ one has $R^{\beta\eta}(*)$. Hence $R^{\beta\eta}(M^*)$. But this is $R^{\beta\eta}(M)$. ■

3C.19. PROPOSITION. Let R be an n -ary logical, expansive and substitutive relation on terms from $\mathcal{D}_1, \dots, \mathcal{D}_n$. Define the restriction to pure terms $R \upharpoonright \Lambda$, again a relation on terms from $\mathcal{D}_1, \dots, \mathcal{D}_n$, by

$$(R \upharpoonright \Lambda)_A(M_1, \dots, M_n) \iff R^{\beta\eta}(M_1) \& \dots \& R^{\beta\eta}(M_n) \& R_A(M_1, \dots, M_n),$$

where $R^{\beta\eta}$ is as in Proposition 3C.18. Then $R \upharpoonright \Lambda$ is logical, expansive and substitutive.

PROOF. Intersection of relations preserves the notion logical, expansive and substitutive. ■

3C.20. PROPOSITION. Given a set of equations \mathcal{E} between closed terms of the same type, define $R_{\mathcal{E}}$ by

$$R_{\mathcal{E}}(M, N) \iff \mathcal{E} \vdash M = N.$$

Then

- (i) $R_{\mathcal{E}}$ is logical.
- (ii) $R_{\mathcal{E}}$ is expansive.
- (iii) $R_{\mathcal{E}}$ is substitutive.
- (iv) $R_{\mathcal{E}}$ is a congruence relation.

PROOF. (i) We must show

$$\mathcal{E} \vdash M_1 = M_2 \Leftrightarrow \forall N_1, N_2 [\mathcal{E} \vdash N_1 = N_2 \Rightarrow \mathcal{E} \vdash M_1 N_1 = M_2 N_2].$$

(\Rightarrow) Let $\mathcal{E} \vdash M_1 = M_2$ and $\mathcal{E} \vdash N_1 = N_2$. Then $\mathcal{E} \vdash M_1 N_1 = M_2 N_2$ follows by (R -congruence), (L -congruence) and (transitivity).

(\Leftarrow) For all x one has $\mathcal{E} \vdash x = x$, so $\mathcal{E} \vdash M_1 x = M_2 x$. Choose x fresh. Then $M_1 = M_2$ follows by (ξ -rule), (η) and (transitivity).

(ii) Obvious, since provability from \mathcal{E} is closed under β -conversion, hence *a fortiori* under weak head expansion.

(iii) Assume that $R_{\mathcal{E}}(M, N)$ in order to show $R_{\mathcal{E}}^*(M, N)$. So suppose $R_{\mathcal{E}}(x^{*1}, x^{*2})$. We must show $R_{\mathcal{E}}(M^{*1}, N^{*2})$. Now going back to the definition of $R_{\mathcal{E}}$ 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 $\text{FV}(MN) \subseteq \{\vec{x}\}$, then

$$\begin{aligned} M^{*1} &=_{\beta} (\lambda \vec{x}.M)\vec{x}^{*1} \\ &=_{\mathcal{E}} (\lambda \vec{x}.N)\vec{x}^{*2} \\ &=_{\beta} N^{*2}. \end{aligned}$$

(iv) Obvious. ■

Semantic logical relations

3C.21. DEFINITION. Let $\mathcal{M}_1, \dots, \mathcal{M}_n$ be typed applicative structures.

(i) S is an n -ary *family of (semantic) relations* or just a *(semantic) relation* on $\mathcal{M}_1 \times \dots \times \mathcal{M}_n$ iff $S = \{S_A\}_{A \in \mathbb{T}}$ and for all A

$$S_A \subseteq \mathcal{M}_1(A) \times \dots \times \mathcal{M}_n(A).$$

(ii) S is a *(semantic) logical relation* if

$$S_{A \rightarrow B}(d_1, \dots, d_n) \Leftrightarrow \forall e_1 \in \mathcal{M}_1(A) \dots e_n \in \mathcal{M}_n(A) [S_A(e_1, \dots, e_n) \Rightarrow S_B(d_1e_1, \dots, d_ne_n)].$$

for all A, B and all $d_1 \in \mathcal{M}_1(A \rightarrow B), \dots, d_n \in \mathcal{M}_n(A \rightarrow B)$.

(iii) The relation S is called *non-empty* if S_0 is non-empty.

Note that S is an n -ary relation on $\mathcal{M}_1 \times \dots \times \mathcal{M}_n$ iff S is a unary relation on the single structure $\mathcal{M}_1 \times \dots \times \mathcal{M}_n$.

3C.22. EXAMPLE. Define S on $\mathcal{M} \times \mathcal{M}$ by $S(d_1, d_2) \stackrel{\Delta}{\iff} d_1 = d_2$. Then S is logical.

3C.23. EXAMPLE. Let \mathcal{M} be a model and let $\pi = \pi_0$ be a permutation of $\mathcal{M}(0)$ which happens to be an element of $\mathcal{M}(0 \rightarrow 0)$. Then π can be lifted to higher types by defining

$$\pi_{A \rightarrow B}(d) \triangleq \lambda e \in \mathcal{M}(A). \pi_B(d(\pi_A^{-1}(e))).$$

Now define S_π (the graph of π)

$$S_\pi(d_1, d_2) \stackrel{\Delta}{\iff} \pi(d_1) = d_2.$$

Then S_π is logical.

3C.24. EXAMPLE. (Friedman [1975]) Let \mathcal{M}, \mathcal{N} be typed structures. A *partial surjective homomorphism* is a family $h = \{h_A\}_{A \in \mathbb{T}}$ of partial maps

$$h_A : \mathcal{M}(A) \nrightarrow \mathcal{N}(A)$$

such that

$$\begin{aligned} h_{A \rightarrow B}(d) = e &\Leftrightarrow e \in \mathcal{N}(A \rightarrow B) \text{ is the unique element (if it exists)} \\ &\text{such that } \forall f \in \text{dom}(h_A) [e(h_A(f)) = h_B(df)]. \end{aligned}$$

This implies that, if all elements involved exist, then

$$h_{A \rightarrow B}(d)h_A(f) = h_B(df).$$

Note that $h(d)$ can fail to be defined if one of the following conditions holds

1. for some $f \in \text{dom}(h_A)$ one has $df \notin \text{dom}(h_B)$;
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 $\mathcal{N}_{A \rightarrow B}$.

Of course, 3 is the basic reason for partialness, whereas 1 and 2 are derived reasons. A partial surjective homomorphism h is completely determined by its h_0 . If we take $\mathcal{M} = \mathcal{M}_X$ and h_0 is any surjection $X \rightarrow \mathcal{N}_0$, then h_A is, although partial, indeed surjective for all A . Define $S_A(d, e) \Leftrightarrow h_A(d) = e$, the graph of h_A . Then S is logical. Conversely, if S_0 is the graph of a surjective partial map $h_0 : \mathcal{M}(0) \rightarrow \mathcal{N}(0)$, and the logical relation S on $\mathcal{M} \times \mathcal{N}$ induced by this S_0 satisfies

$$\forall e \in \mathcal{N}(A) \exists d \in \mathcal{M}(A) S_A(d, e),$$

then S is the graph of a partial surjective homomorphism from \mathcal{M} to \mathcal{N} .

Kreisel's Hereditarily Recursive Operations are one of the first appearances of logical relations, see Bezem [1985a] for a detailed account of extensionality in this context.

3C.25. PROPOSITION. *Let $R \subseteq \mathcal{M}_1 \times \cdots \times \mathcal{M}_n$ be the n -ary semantic logical relation determined by $R_0 = \emptyset$. Then*

$$\begin{aligned} R_A &= \mathcal{M}_1(A) \times \cdots \times \mathcal{M}_n(A), && \text{if } \Lambda_{\rightarrow}^{\emptyset}(A) \neq \emptyset; \\ &= \emptyset, && \text{if } \Lambda_{\rightarrow}^{\emptyset}(A) = \emptyset. \end{aligned}$$

PROOF. Analogous to the proof of Proposition 3C.3 for semantic logical relations, using that for all \mathcal{M}_i and all types A one has $\mathcal{M}_i(A) \neq \emptyset$, by Definition 3A.1. ■

3C.26. THEOREM (*Fundamental theorem for semantic logical relations*).

Let $\mathcal{M}_1, \dots, \mathcal{M}_n$ be typed λ -models and let S be logical on $\mathcal{M}_1 \times \cdots \times \mathcal{M}_n$. Then for each term $M \in \Lambda_{\rightarrow}^{\emptyset}$ one has

$$S(\llbracket M \rrbracket^{\mathcal{M}_1}, \dots, \llbracket M \rrbracket^{\mathcal{M}_n}).$$

PROOF. We treat the case $n = 1$. Let $S \subseteq \mathcal{M}$ be logical. We claim that for all $M \in \Lambda_{\rightarrow}$ and all partial valuations ρ such that $\text{FV}(M) \subseteq \text{dom}(\rho)$ one has

$$S(\rho) \Rightarrow S(\llbracket M \rrbracket_{\rho}).$$

This follows by an easy induction on M . In case $M \equiv \lambda x.N$ one should show $S(\llbracket \lambda x.N \rrbracket_{\rho})$, assuming $S(\rho)$. This means that for all d of the right type with $S(d)$ one has $S(\llbracket \lambda x.N \rrbracket_{\rho}^d)$. This is the same as $S(\llbracket N \rrbracket_{\rho[x:=d]})$, which holds by the induction hypothesis.

The statement now follows immediately from the claim, by taking as ρ the empty function. ■

We give two applications.

3C.27. EXAMPLE. Let S be the graph of a partial surjective homomorphism $h : \mathcal{M} \rightarrow \mathcal{N}$. 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 that paper that for infinite X one has

$$\mathcal{M}_X \models M = N \Leftrightarrow M =_{\beta\eta} N.$$

We have derived this in another way.

3C.28. EXAMPLE. Let \mathcal{M} be a typed applicative structure. Let $\Delta \subseteq \mathcal{M}$. Write $\Delta(A) = \Delta \cap \mathcal{M}(A)$. Assume that $\Delta(A) \neq \emptyset$ for all $A \in \mathbb{T}$ and

$$d \in \Delta(A \rightarrow B), e \in \Delta(A) \Rightarrow de \in \Delta(B).$$

Then Δ may fail to be a typed applicative structure because it is not extensional. Equality as a binary relation E_0 on $\Delta(0) \times \Delta(0)$ 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 typed applicative structure. In particular, if \mathcal{M} is a typed λ -model, then write

$$\begin{aligned}\Delta^+ &\triangleq \{\llbracket M \rrbracket \vec{d} \mid M \in \Lambda_{\rightarrow}^o, \vec{d} \in \Delta\} \\ &= \{d \in \mathcal{M} \mid \exists M \in \Lambda_{\rightarrow}^o, \exists d_1 \dots d_n \in \Delta \quad \llbracket M \rrbracket d_1 \dots d_n = d\}.\end{aligned}$$

for the applicative closure of Δ . The *Gandy-hull* of Δ in \mathcal{M} is the set Δ^{+E} . From the fundamental theorem for semantic logical relations it can be derived that

$$\mathcal{G}_{\Delta}(\mathcal{M}) = \Delta^{+E}/E$$

is a typed λ -model. This model will be also called the Gandy-hull of Δ in \mathcal{M} . Do Exercise 3F.34 to get acquainted with the notion of the Gandy hull.

3C.29. DEFINITION. Let $\mathcal{M}_1, \dots, \mathcal{M}_n$ be type structures.

(i) Let S be an n -ary relation on $\mathcal{M}_1 \times \dots \times \mathcal{M}_n$. For valuations ρ_1, \dots, ρ_n with $\rho_i : \text{Var} \rightarrow \mathcal{M}_i$ we define

$$S(\rho_1, \dots, \rho_n) \Leftrightarrow S(\rho_1(x), \dots, \rho_n(x)), \text{ for all variables } x \text{ satisfying } \forall i. \rho_i(x) \downarrow.$$

(ii) Let S be an n -ary relation on $\mathcal{M}_1 \times \dots \times \mathcal{M}_n$. The *lifting of S* to $\mathcal{M}_1^* \times \dots \times \mathcal{M}_n^*$, notation S^* , is defined for $d_1 \in \mathcal{M}_1^*, \dots, d_n \in \mathcal{M}_n^*$ as follows.

$$\begin{aligned}S^*(d_1, \dots, d_n) &\stackrel{\Delta}{\Leftrightarrow} \forall \rho_1 : \text{V} \rightarrow \mathcal{M}_1, \dots, \rho_n : \text{V} \rightarrow \mathcal{M}_n \\ [S(\rho_1, \dots, \rho_n)] &\Rightarrow S((d_1)_{\rho_1}^{\mathcal{M}_1}, \dots, (d_n)_{\rho_n}^{\mathcal{M}_n}).\end{aligned}$$

The interpretation $((-))_{\rho} : \mathcal{M}^* \rightarrow \mathcal{M}$ was defined in Definition 3A.22(ii).

(iii) For $\rho : \text{V} \rightarrow \mathcal{M}^*$ define the ‘substitution’ $(-)^\rho : \mathcal{M}^* \rightarrow \mathcal{M}^*$ as follows.

$$\begin{aligned}x^\rho &\triangleq \rho(x); \\ \underline{m}^\rho &\triangleq \underline{m}; \\ (d_1 d_2)^\rho &\triangleq d_1^\rho d_2^\rho\end{aligned}$$

(iv) Let now S be an n -ary relation on $\mathcal{M}_1^* \times \dots \times \mathcal{M}_n^*$. Then S is called *substitutive* if for all $d_1 \in \mathcal{M}_1^*, \dots, d_n \in \mathcal{M}_n^*$ one has

$$\begin{aligned}S(d_1, \dots, d_n) &\Leftrightarrow \forall \rho_1 : \text{V} \rightarrow \mathcal{M}_1^*, \dots, \rho_n : \text{V} \rightarrow \mathcal{M}_n^* \\ [S(\rho_1, \dots, \rho_n)] &\Rightarrow S(d_1^{\rho_1}, \dots, d_n^{\rho_n}).\end{aligned}$$

3C.30. REMARK. If $S \subseteq \mathcal{M}_1^* \times \dots \times \mathcal{M}_n^*$ is substitutive, then for every variable x one has $S(x, \dots, x)$.

3C.31. EXAMPLE. (i) Let S be the equality relation on $\mathcal{M} \times \mathcal{M}$. Then S^* is the equality relation on $\mathcal{M}^* \times \mathcal{M}^*$.

(ii) If S is the graph of a surjective homomorphism, then S^* is the graph of a partial surjective homomorphism whose restriction (in the literal sense, not the analogue of 3C.19) to \mathcal{M} is S and which fixes each indeterminate x .

3C.32. LEMMA. Let $S \subseteq \mathcal{M}_1 \times \dots \times \mathcal{M}_n$ be a semantic logical relation.

- (i) Let $\vec{d} \in \mathcal{M}_1 \times \dots \times \mathcal{M}_n$. Then $S(\vec{d}) \Rightarrow S^*(\vec{d})$.
- (ii) Suppose S is non-empty and that the \mathcal{M}_i are λ -models. Then for $\vec{d} \in \mathcal{M}_1 \times \dots \times \mathcal{M}_n$ one has $S^*(\vec{d}) \Rightarrow S(\vec{d})$.

PROOF. For notational simplicity, take $n = 1$.

- (i) Suppose that $S(d)$. Then $S^*(\underline{d})$, as $((\underline{d}))_\rho = d$, hence $S((\underline{d}))_\rho$, for all ρ .

(ii) Suppose $S^*(\underline{d})$. Then for all $\rho : V \rightarrow M$ one has

$$\begin{aligned} S(\rho) &\Rightarrow S((\underline{d}))_{\rho}^{M^*} \\ &\Rightarrow S(d). \end{aligned}$$

Since S_0 is non-empty, say $d \in S_0$, also S_A is non-empty for all $A \in \mathbb{T}^0$: the constant function $\lambda \vec{x}. d \in S_A$. Hence there exists a ρ such that $S(\rho)$ and therefore $S(d)$. ■

3C.33. PROPOSITION. *Let $S \subseteq M_1 \times \dots \times M_n$ be a semantic logical relation. Then $S^* \subseteq M_1^* \times \dots \times M_n^*$ and one has the following.*

- (i) $S^*(x, \dots, x)$ for all variables.
- (ii) S^* is a semantic logical relation.
- (iii) S^* is substitutive.
- (iv) If S is substitutive and each M_i is a typed λ -model, then

$$S^*(d_1, \dots, d_n) \Leftrightarrow S(\lambda \vec{x}. d_1, \dots, \lambda \vec{x}. d_n),$$

where the variables on which the \vec{d} depend are included in the list \vec{x} .

PROOF. Take $n=1$ for notational simplicity.

- (i) If $S(\rho)$, then by definition one has $S((x))_{\rho}$ for all variables x . Therefore $S^*(x)$.
- (ii) We have to show

$$S_{A \rightarrow B}^*(d) \Leftrightarrow \forall e \in M^*(A). [S_A^*(e) \Rightarrow S_B^*(de)].$$

(\Rightarrow) Suppose $S_{A \rightarrow B}^*(d)$, $S_A^*(e)$, in order to show $S_B^*(de)$. So assume $S(\rho)$ towards $S((de))_{\rho}$. By the assumption we have $S((d))_{\rho}$, $S((e))_{\rho}$, hence indeed $S((de))_{\rho}$, as S is logical.

(\Leftarrow) Assume the RHS in order to show $S^*(d)$. To this end suppose $S(\rho)$ towards $S((d))_{\rho}$. Since S is logical it suffices to show $S(e) \Rightarrow S((d))_{\rho}e$ for all $e \in M$. Taking $e \in M$, we have

$$\begin{aligned} S(e) &\Rightarrow S^*(\underline{e}), && \text{by Lemma 3C.32(i),} \\ &\Rightarrow S^*(de), && \text{by the RHS,} \\ &\Rightarrow S((d))_{\rho}e, && \text{as } e = ((\underline{e}))_{\rho} \text{ and } S(\rho). \end{aligned}$$

(iii) For $d \in M^*$ we show that $S^*(d) \Leftrightarrow \forall \rho : V \rightarrow M^*. [S^*(\rho) \Rightarrow S^*(d^{\rho})]$, i.e.

$$\forall \rho : V \rightarrow M. [S(\rho) \Rightarrow S((d))_{\rho}^{M^*}] \Leftrightarrow \forall \rho' : V \rightarrow M^*. [S^*(\rho') \Rightarrow S^*(d^{\rho'})].$$

As to (\Rightarrow). Let $d \in M^*$ and suppose

$$\forall \rho : V \rightarrow M. [S(\rho) \Rightarrow S((d))_{\rho}^{M^*}], \quad (1)$$

and

$$S^*(\rho'), \text{ for a given } \rho' : V \rightarrow M^*, \quad (2)$$

in order to show $S^*(d^{\rho'})$. To this end we assume

$$S(\rho'') \text{ with } \rho'' : V \rightarrow M \quad (3)$$

in order to show

$$S((d^{\rho'}))_{\rho''}^{M^*}. \quad (4)$$

Now define

$$\rho'''(x) \triangleq ((\rho'(x)))_{\rho''}^{M^*}.$$

Then $\rho'':\mathbb{V} \rightarrow \mathcal{M}$ and by (2), (3) one has $S(\rho''(x))$ (being $S(((\rho'(x)))_{\rho''}^{\mathcal{M}^*})$), hence

$$S(((d))_{\rho''}). \quad (5)$$

By induction on the structure of $d \in \mathcal{M}^*$ (considered as $\underline{\mathcal{M}}$ modulo $\sim_{\mathcal{M}}$) it follows that

$$((d))_{\rho''}^{\mathcal{M}} = ((d^{\rho'}))_{\rho''}^{\mathcal{M}}.$$

Therefore (5) yields (4).

As to (\Leftarrow). Assume the RHS. Taking $\rho'(x) = x \in \mathcal{M}^*$ one has $S^*\rho'$ by (i), hence $S^*(d_{\rho'}^{\mathcal{M}^*})$. Now one easily shows by induction on $d \in \underline{\mathcal{M}}$ that $d_{\rho'}^{\mathcal{M}^*} = d$, so one has $S^*(d)$.

(iv) W.l.o.g. we assume that d depends only on y and that $\vec{x} = y$. As \mathcal{M} is a typed λ -model, there is a unique $F \in \mathcal{M}$ such that for all $y \in \mathcal{M}$ one has $Fy = d$. This F is denoted as $\lambda y.d$.

$$\begin{aligned} S(d) &\Leftrightarrow S(Fy) \\ &\Leftrightarrow \forall \rho: \mathbb{V} \rightarrow \mathcal{M}^* [S(\rho) \Rightarrow S(((i(Fy)))_{\rho})], \quad \text{as } S \text{ is substitutive,} \\ &\Leftrightarrow \forall \rho: \mathbb{V} \rightarrow \mathcal{M}^* [S(\rho) \Rightarrow S(((i(F)))_{\rho}((i(y)))_{\rho})], \\ &\Leftrightarrow \forall e \in \mathcal{M}^*. [S(e) \Rightarrow S(Fe)], \quad \text{taking } \rho(x) = e, \\ &\Leftrightarrow S(F), \quad \text{as } S \text{ is logical,} \\ &\Leftrightarrow S(\lambda y.d). \blacksquare \end{aligned}$$

3C.34. PROPOSITION. Let $S \subseteq \mathcal{M}_1 \times \cdots \times \mathcal{M}_m$ and $S' \subseteq \mathcal{N}_1 \times \cdots \times \mathcal{N}_n$ be non-empty logical relations. Define $S \times S'$ on $\mathcal{M}_1 \times \cdots \times \mathcal{M}_m \times \mathcal{N}_1 \times \cdots \times \mathcal{N}_n$ by

$$(S \times S')(d_1, \dots, d_m, e_1, \dots, e_n) \stackrel{\Delta}{\iff} S(d_1, \dots, d_m) \& S'(e_1, \dots, e_n).$$

Then $S \times S' \subseteq \mathcal{M}_1 \times \cdots \times \mathcal{M}_m \times \mathcal{N}_1 \times \cdots \times \mathcal{N}_n$ is a non-empty logical relation. If moreover both S and S' are substitutive, then so is $S \times S'$.

PROOF. As for syntactic logical relations. \blacksquare

3C.35. PROPOSITION. (i) The universal relation S^U defined by $S^U \triangleq \mathcal{M}_1^* \times \cdots \times \mathcal{M}_n^*$ is substitutive and logical on $\mathcal{M}_1^* \times \cdots \times \mathcal{M}_n^*$.

(ii) Let S be an n -ary logical relation on $\mathcal{M}^* \times \cdots \times \mathcal{M}^*$ (n -copies of \mathcal{M}^*). Let π be a permutation of $\{1, \dots, n\}$. Define S^π on $\mathcal{M}^* \times \cdots \times \mathcal{M}^*$ by

$$S^\pi(d_1, \dots, d_n) \stackrel{\Delta}{\iff} S(d_{\pi(1)}, \dots, d_{\pi(n)}).$$

Then S^π is a logical relation. If moreover S is substitutive, then so is S^π .

(iii) If S is an n -ary substitutive logical relation on $\mathcal{M}^* \times \cdots \times \mathcal{M}^*$, then the diagonal S^Δ defined by

$$S^\Delta(d) \stackrel{\Delta}{\iff} S(d, \dots, d)$$

is a unary substitutive logical relation on \mathcal{M}^* .

(iv) If \mathcal{S} is a class of n -ary substitutive logical relations on $\mathcal{M}_1^* \times \cdots \times \mathcal{M}_n^*$, then the relation $\cap \mathcal{S} \subseteq \mathcal{M}_1^* \times \cdots \times \mathcal{M}_n^*$ is a substitutive logical relation.

(v) If S is an $(n+1)$ -ary substitutive logical relation on $\mathcal{M}_1^* \times \cdots \times \mathcal{M}_{n+1}^*$ and \mathcal{M}_{n+1}^* is a typed λ -model, then $\exists S$ defined by

$$\exists S(d_1, \dots, d_n) \stackrel{\Delta}{\iff} \exists d_{n+1}. S(d_1, \dots, d_{n+1})$$

is an n -ary substitutive logical relation.

PROOF. For convenience we take $n = 1$. We treat (v), leaving the rest to the reader.

(v) Let $S \subseteq \mathcal{M}_1^* \times \mathcal{M}_2^*$ be substitutive and logical. Define $R(d_1) \Leftrightarrow \exists d_2 \in \mathcal{M}_2^*. S(d_1.d_2)$, towards

$$\forall d_1 \in \mathcal{M}_1^*. [R(d_1) \Leftrightarrow \forall e_1 \in \mathcal{M}_1^*. [R(e_1) \Rightarrow R(d_1e_1)]].$$

(\Rightarrow) Suppose $R(d_1), R(e_1)$ in order to show $R(d_1e_1)$. Then there are $d_2, e_2 \in \mathcal{M}_2^*$ such that $S(d_1, d_2), S(e_1, e_2)$. Then $S(d_1e_1, d_2, e_2)$, as S is logical. Therefore $R(d_1e_1)$ indeed.

(\Leftarrow) Suppose $\forall e_1 \in \mathcal{M}_1^*. [R(e_1) \Rightarrow R(d_1e_1)]$, towards $R(d_1)$. By the assumption

$$\forall e_1 [\exists e_2. S(e_1, e_2) \Rightarrow \exists e'_2. S(d_1e_1, e'_2)].$$

Hence

$$\forall e_1, e_2 \exists e'_2. [S(e_1, e_2) \Rightarrow S(d_1e_1, e'_2)]. \quad (1)$$

As S is substitutive, we have $S(x, x)$, by Remark 3C.30. We continue as follows

$$\begin{aligned} S(x, x) &\Rightarrow S(d_1x, e'_2[x]), && \text{for some } e'_2 = e'_2[x] \text{ by (1),} \\ &\Rightarrow S(d_1x, d_2x), && \text{where } d_2 = \lambda x. e'_2[x] \text{ using that } \mathcal{M}_2^* \\ &\Rightarrow S(e_1, e_2) \Rightarrow S(d_1e_1, d_2, e_2), && \text{is a typed } \lambda\text{-model,} \\ &\Rightarrow S(d_1, d_2), && \text{by substitutivity of } S, \\ &\Rightarrow R(d_1). && \text{since } S \text{ is logical,} \end{aligned}$$

This establishes that $\exists S = R$ is logical.

Now assume that S is substitutive, in order to show that so is R . I.e. we must show

$$R(d_1) \Leftrightarrow \forall \rho_1. [\forall x \in V. R(\rho_1(x))] \Rightarrow R((d_1)^{\rho_1}). \quad (1)$$

(\Rightarrow) Assuming $R(d_1), R(\rho_1(x))$ we get $S(d_1, d_2), S(\rho_1(x), d_2^x)$, for some d_2, d_2^x . Defining ρ_2 by $\rho_2(x) = d_2^x$, for the free variables in d_2 , we get $S(\rho_1(x), \rho_2(x))$, hence by the substitutivity of S it follows that $S((d_1)^{\rho_1}, (d_2)^{\rho_2})$ and therefore $R((d_1)^{\rho_1})$.

(\Leftarrow) By the substitutivity of S one has for all variables x that $S(x, x)$, by Remark 3C.30, hence also $R(x)$. Now take in the RHS of (1) the identity valuation $\rho_1(x) = x$, for all x . Then one obtains $R((d_1)^{\rho_1})$, which is $R(d_1)$. ■

3C.36. EXAMPLE. Consider $\mathcal{M}_{\mathbb{N}}$ and define

$$S_0(n, m) \Leftrightarrow n \leq m,$$

where \leq is the usual ordering on \mathbb{N} . Then $\{d \in S^* \mid d =^* d\}/=^*$ is the set of hereditarily monotone functionals. Similarly $\exists(S^*)$ induces the set of hereditarily majorizable functionals, see the section by Howard in [Troelstra \[1973\]](#).

Relating syntactic and semantic logical relations

One may wonder whether the Fundamental Theorem for semantic logical relations follows from the syntactic version (but not vice versa; e.g. the usual semantic logical relations are automatically closed under $\beta\eta$ -conversion). This indeed is the case. The ‘hinge’ is that a logical relation $R \subseteq \Lambda_{\rightarrow}[\mathcal{M}^*]$ can be seen as a semantic logical relation (as $\Lambda_{\rightarrow}[\mathcal{M}^*]$ is a typed applicative structure) and at the same time as a syntactic one (as $\Lambda_{\rightarrow}[\mathcal{M}^*]$ consists of terms from some set of constants). We also need this dual vision for the notion of substitutivity. For this we have to merge the syntactic and the semantic version of these notions. Let \mathcal{M} be a typed applicative structure, containing at each

type A variables of type A . A valuation is a map $\rho: V \rightarrow \mathcal{M}$ such that $\rho(x^A) \in \mathcal{M}(A)$. This ρ can be extended to a substitution $(-)^\rho: \mathcal{M} \rightarrow \mathcal{M}$. A unary relation $R \subseteq \mathcal{M}$ is substitutive if for all $M \in \mathcal{M}$ one has

$$R(M) \Leftrightarrow [\forall x: V. [R(\rho(x)) \Rightarrow R((M)^\rho)]].$$

The notion substitutivity is analogous for relations $R \subseteq \Lambda_{\rightarrow}[\mathcal{D}]$, using Definition 3C.8(iii), as for relations $R \subseteq \mathcal{M}^*$, using Definition 3C.29(iv).

3C.37. NOTATION. Let \mathcal{M} be a typed applicative structure. Write

$$\begin{aligned}\Lambda_{\rightarrow}[\mathcal{M}] &\triangleq \Lambda_{\rightarrow}[\{\underline{d} \mid d \in \mathcal{M}\}]; \\ \Lambda_{\rightarrow}(\mathcal{M}) &\triangleq \Lambda_{\rightarrow}[\mathcal{M}]/=_\beta.\end{aligned}$$

Then $\Lambda_{\rightarrow}[\mathcal{M}]$ is typed applicative structure and $\Lambda_{\rightarrow}(\mathcal{M})$ is a typed λ -model.

3C.38. DEFINITION. Let \mathcal{M} , and hence also \mathcal{M}^* , be a typed λ -model. For $\rho: V \rightarrow \mathcal{M}^*$ extend $\llbracket - \rrbracket_\rho: \Lambda_{\rightarrow} \rightarrow \mathcal{M}^*$ to $\llbracket - \rrbracket_\rho^{\mathcal{M}^*}: \Lambda_{\rightarrow}[\mathcal{M}^*] \rightarrow \mathcal{M}^*$ as follows.

$$\begin{aligned}\llbracket x \rrbracket_\rho &\triangleq \rho(x) \\ \llbracket m \rrbracket_\rho &\triangleq m, \quad \text{with } m \in \mathcal{M}^*, \\ \llbracket PQ \rrbracket_\rho &\triangleq \llbracket P \rrbracket_\rho \llbracket Q \rrbracket_\rho \\ \llbracket \lambda x. P \rrbracket_\rho &\triangleq d, \quad \text{the unique } d \in \mathcal{M}^* \text{ with } \forall e. de = \llbracket P \rrbracket_{\rho[x:=e]}.\end{aligned}$$

Remember the definition 3C.29 of $(-)^\rho: \mathcal{M}^* \rightarrow \mathcal{M}^*$.

$$\begin{aligned}(x)^\rho &\triangleq \rho(x) \\ (\underline{m})^\rho &\triangleq m, \quad \text{with } m \in \mathcal{M}^*, \\ (PQ)^\rho &\triangleq (P)^\rho(Q)^\rho.\end{aligned}$$

Now define the predicate $D \subseteq \Lambda_{\rightarrow}[\mathcal{M}^*] \times \mathcal{M}^*$ as follows.

$$D(M, d) \stackrel{\Delta}{\Leftrightarrow} \forall \rho: V \rightarrow \mathcal{M}^*. \llbracket M \rrbracket_\rho^{\mathcal{M}^*} = (d)^\rho.$$

3C.39. LEMMA. D is a substitutive semantic logical relation.

PROOF. First we show that D is logical. We must show for $M \in \Lambda_{\rightarrow}[\mathcal{M}^*], d \in \mathcal{M}^*$ that

$$D(M, d) \Leftrightarrow \forall N \in \Lambda_{\rightarrow}[\mathcal{M}^*] \forall e \in \mathcal{M}^*. [D(N, e) \Rightarrow D(MN, de)].$$

(\Rightarrow) Suppose $D(M, d), D(N, e)$, towards $D(MN, de)$. Then for all $\rho: V \rightarrow \mathcal{M}^*$ by definition $\llbracket M \rrbracket_\rho^{\mathcal{M}^*} = (d)^\rho$ and $\llbracket N \rrbracket_\rho^{\mathcal{M}^*} = (e)^\rho$. But then $\llbracket MN \rrbracket_\rho^{\mathcal{M}^*} = (de)^\rho$, and therefore $D(MN, de)$.

(\Leftarrow) Now suppose $\forall N \in \Lambda_{\rightarrow}[\mathcal{M}^*] \forall e \in \mathcal{M}^*. [D(N, e) \Rightarrow D(MN, de)]$, towards $D(M, d)$. Let x be a fresh variable, i.e. not in M or d . Note that $x \in \Lambda_{\rightarrow}[\mathcal{M}^*]$, $x \in \mathcal{M}^*$, and $D(x, x)$.

Hence by assumption

$$\begin{aligned}
 D(x, x) &\Rightarrow \forall \rho \llbracket Mx \rrbracket_\rho = (dx)^\rho \\
 &\Rightarrow \forall \rho \llbracket M \rrbracket_\rho \llbracket x \rrbracket_\rho = (d)^\rho(x)^\rho \\
 &\Rightarrow \forall \rho \llbracket M \rrbracket_{\rho'} \llbracket x \rrbracket_{\rho'} = (d)^{\rho'}(x)^{\rho'}, \quad \text{where } \rho' = \rho[x := e], \\
 &\Rightarrow \forall \rho \forall e \in \mathcal{M}^*. \llbracket M \rrbracket_\rho e = (d)^\rho e, \quad \text{by the freshness of } x, \\
 &\Rightarrow \forall \rho \llbracket M \rrbracket_\rho = (d)^\rho, \quad \text{by extensionality,} \\
 &\Rightarrow D(M, d).
 \end{aligned}$$

Secondly we show that D is substitutive. We must show for $M \in \Lambda \rightarrow [\mathcal{M}^*]$, $d \in \mathcal{M}^*$

$$\begin{aligned}
 D(M, d) &\Leftrightarrow \forall \rho_1: \mathbb{V} \rightarrow \Lambda \rightarrow [\mathcal{M}^*], \rho_2: \mathbb{V} \rightarrow \mathcal{M}^*. \\
 &\quad [\forall x \in \mathbb{V}. D(\rho_1(x), \rho_2(x)) \Rightarrow D((M)^{\rho_1}, (d)^{\rho_2})].
 \end{aligned}$$

(\Rightarrow) Suppose $D(M, d)$ and $\forall x \in \mathbb{V}. D(\rho_1(x), \rho_2(x))$ towards $D((M)^{\rho_1}, (d)^{\rho_2})$. Then for all $\rho: \mathbb{V} \rightarrow \mathcal{M}^*$ one has

$$\llbracket M \rrbracket_\rho = (d)^\rho \quad (1)$$

$$\forall x \in \mathbb{V}. \llbracket \rho_1(x) \rrbracket_\rho = (\rho_2(x))^\rho. \quad (2)$$

Let $\rho'_1(x) = \llbracket \rho_1(x) \rrbracket_\rho^{\mathcal{M}^*}$ and $\rho'_2(x) = (\rho_2(x))^\rho$. By induction on M and d one can show analogous to Lemma 3A.13(i) that

$$\llbracket M^{\rho_1} \rrbracket_\rho = \llbracket M \rrbracket_{\rho'_1} \quad (3)$$

$$((d)^{\rho_2})^\rho = (d)^{\rho'_2}. \quad (4)$$

It follows by (2) that $\rho'_1 = \rho'_2$ and hence by (3), (4), and (1) that $\llbracket (M)^{\rho_1} \rrbracket_\rho = ((d)^{\rho_2})^\rho$, for all ρ . Therefore $D((M)^{\rho_1}, (d)^{\rho_2})$.

(\Leftarrow) Assume the RHS. Define $\rho_1(x) = x \in \Lambda \rightarrow [\mathcal{M}^*]$, $\rho_2(x) = x \in \mathcal{M}^*$. Then we have $D(\rho_1, \rho_2)$, hence by the assumption $D((M)^{\rho_1}, (d)^{\rho_2})$. By the choice of ρ_1, ρ_2 this is $D(M, d)$. ■

3C.40. LEMMA. Let $M \in \Lambda \rightarrow$. Then $\llbracket M \rrbracket^{\mathcal{M}^*} = \llbracket \llbracket M \rrbracket \rrbracket^{\mathcal{M}^*} \in \mathcal{M}^*$.

PROOF. Let $i: \mathcal{M} \rightarrow \mathcal{M}^*$ be the canonical inbedding defined by $i(d) = \underline{d}$. Then for all $M \in \Lambda \rightarrow$ and all $\rho: \mathbb{V} \rightarrow \mathcal{M}$ one has

$$i(\llbracket M \rrbracket_\rho^{\mathcal{M}}) = \llbracket M \rrbracket_{i \circ \rho}^{\mathcal{M}^*}.$$

Hence for closed terms M it follows that $\llbracket M \rrbracket^{\mathcal{M}^*} = \llbracket M \rrbracket_{i \circ \rho}^{\mathcal{M}^*} = i(\llbracket M \rrbracket_\rho^{\mathcal{M}}) = \llbracket \llbracket M \rrbracket \rrbracket^{\mathcal{M}}$. ■

3C.41. DEFINITION. Let $R \subseteq \Lambda \rightarrow [\mathcal{M}_1^*] \times \cdots \times \Lambda \rightarrow [\mathcal{M}_n^*]$. Then R is called *invariant* if for all $M_1, N_1 \in \Lambda \rightarrow [\mathcal{M}_1^*], \dots, M_n, N_n \in \Lambda \rightarrow [\mathcal{M}_n^*]$ one has

$$\left. \begin{array}{l} R(M_1, \dots, M_n) \\ \mathcal{M}_1^* \models M_1 = N_1 \& \dots \& \mathcal{M}_n^* \models M_n = N_n \end{array} \right\} \Rightarrow R(N_1, \dots, N_n).$$

3C.42. DEFINITION. Let $\mathcal{M}_1, \dots, \mathcal{M}_n$ be typed applicative structures.

(i) Let $S \subseteq \mathcal{M}_1^* \times \cdots \times \mathcal{M}_n^*$. Define the relation $S^\wedge \subseteq \Lambda \rightarrow [\mathcal{M}_1^*] \times \cdots \times \Lambda \rightarrow [\mathcal{M}_n^*]$ by

$$\begin{aligned}
 S^\wedge(M_1, \dots, M_n) &\triangleleft \exists d_1 \in \mathcal{M}_1^* \cdots \exists d_n \in \mathcal{M}_n^*. [S(d_1, \dots, d_n) \& \\
 &\quad D(M_1, d_1) \& \dots \& D(M_n, d_n)].
 \end{aligned}$$

(ii) Let $R \subseteq \Lambda_{\rightarrow}[\mathcal{M}_1^*] \times \cdots \times \Lambda_{\rightarrow}[\mathcal{M}_n^*]$. Define $R^{\vee} \subseteq \mathcal{M}_1^* \times \cdots \times \mathcal{M}_n^*$ by

$$R^{\vee}(d_1, \dots, d_n) \stackrel{\Delta}{\iff} \exists M_1 \in \Lambda_{\rightarrow}[\mathcal{M}_1^*], \dots, M_n \in \Lambda_{\rightarrow}[\mathcal{M}_n^*]. [R(M_1, \dots, M_n) \ \& \ D(M_1, d_1) \ \& \ \dots \ \& \ D(M_n, d_n)].$$

3C.43. DEFINITION. Let $\iota : V \rightarrow \mathcal{M}^*$ be the ‘identity’ valuation, that is $\iota(x) \triangleq [x]$.

3C.44. LEMMA. (i) Let $S \subseteq \mathcal{M}_1^* \times \cdots \times \mathcal{M}_n^*$. Then S^{\wedge} is invariant.

(ii) Let $R \subseteq \Lambda_{\rightarrow}[\mathcal{M}_1^*] \times \cdots \times \Lambda_{\rightarrow}[\mathcal{M}_n^*]$ be invariant. Then for all $M_1 \in \Lambda_{\rightarrow}^{\sigma}[\mathcal{M}_1^*], \dots, M_n \in \Lambda_{\rightarrow}^{\sigma}[\mathcal{M}_n^*]$ one has

$$R(M_1, \dots, M_n) \Rightarrow R^{\vee}([\![M_1]\!]_{\iota}^{\mathcal{M}_1^*}, \dots, [\![M_n]\!]_{\iota}^{\mathcal{M}_n^*}).$$

PROOF. For notational convenience we take $n = 1$.

$$\begin{aligned} (i) \quad S^{\wedge}(M) \ \& \ \mathcal{M}^* \models M = N & \Rightarrow & \exists d \in \mathcal{M}^*. [S(d) \ \& \ D(M, d)] \ \& \ \mathcal{M}^* \models M = N \\ & \Rightarrow & \exists d \in \mathcal{M}^*. [S(d) \ \& \ \\ & & \forall \rho. [\![M]\!]_{\rho} = (d)^{\rho} \ \& \ [\![M]\!]_{\rho} = [\![N]\!]_{\rho}] \\ & \Rightarrow & \exists d. [S(d) \ \& \ D(N, d)] \\ & \Rightarrow & S^{\wedge}(N). \end{aligned}$$

(ii) Suppose $R(M)$. Let $M' = [\![M]\!]_{\iota} \in \Lambda_{\rightarrow}[\mathcal{M}^*]$. Then $[\![M']\!]_{\rho} = [\![M]\!]_{\iota} = [\![M]\!]_{\rho}$, since M is closed. Hence $R(M')$ by the invariance of R and $D(M', [\![M]\!]_{\iota})$. Therefore $R^{\vee}([\![M]\!]_{\iota})$. ■

3C.45. PROPOSITION. Let $\mathcal{M}_1, \dots, \mathcal{M}_n$ be typed λ -models.

(i) Let $S \subseteq \mathcal{M}_1^* \times \cdots \times \mathcal{M}_n^*$ be a substitutive semantic logical relation. Then S^{\wedge} is an invariant and substitutive syntactic logical relation.

(ii) Let $R \subseteq \Lambda_{\rightarrow}[\mathcal{M}_1^*] \times \cdots \times \Lambda_{\rightarrow}[\mathcal{M}_n^*]$ be a substitutive syntactic logical relation. Then R^{\vee} is a substitutive semantic logical relation.

PROOF. Again we take $n = 1$.

(i) By Lemma 3C.44(i) S^{\wedge} is invariant. Moreover, one has for $M \in \Lambda_{\rightarrow}[\mathcal{M}^*]$

$$S^{\wedge}(M) \Leftrightarrow \exists d \in \mathcal{M}^*. [S(d) \ \& \ D(M, d)].$$

By assumption S is a substitutive logical relation and also D , by Proposition 3C.39. By Proposition 3C.35(iv) and (v) so is their conjunction and its \exists -projection S^{\wedge} .

(ii) One has for $d \in \mathcal{M}^*$

$$R^{\vee}(d) \Leftrightarrow \exists M \in \Lambda_{\rightarrow}[\mathcal{M}^*]. [D(M, d) \ \& \ R(M)].$$

We conclude similarly. ■

3C.46. PROPOSITION. Let $\mathcal{M}_1, \dots, \mathcal{M}_n$ be typed λ -models. Let $S \subseteq \mathcal{M}_1^* \times \cdots \times \mathcal{M}_n^*$ be a substitutive logical relation. Then $S^{\wedge\vee} = S$.

PROOF. For notational convenience take $n = 1$. Write $T = S^{\wedge}$. Then for $d \in \mathcal{M}^*$

$$\begin{aligned} T^{\vee}(d) &\Leftrightarrow \exists M \in \Lambda_{\rightarrow}[\mathcal{M}^*]. [T(M) \ \& \ D(M, d)], \\ &\Leftrightarrow \exists M \in \Lambda_{\rightarrow}[\mathcal{M}^*] \exists d' \in \mathcal{M}^*. [S(d') \ \& \ D(M, d') \ \& \ D(M, d)], \\ &\quad \text{which implies } d' = d, \text{ as } \mathcal{M}^* = \underline{\mathcal{M}} / \sim_{\mathcal{M}}, \\ &\Leftrightarrow S(d), \end{aligned}$$

where the last \Leftarrow follows by taking $M = \underline{d}$, $d' = d$. Therefore $S^{\wedge\vee} = S$. ■

Using this result, the Fundamental Theorem for semantic logical relations can be derived from the syntactic version.

3C.47. PROPOSITION. *The Fundamental Theorem for syntactic logical relations implies the one for semantic logical relations. That is, let $\mathcal{M}_1, \dots, \mathcal{M}_n$ be λ -models, then for the following two statements one has (i) \Rightarrow (ii).*

(i) *Let R on $\Lambda_{\rightarrow}[\vec{\mathcal{M}}]$ be an expansive and substitutive syntactic logical relation. Then for all $A \in \mathbb{T}$ and all pure terms $M \in \Lambda_{\rightarrow}(A)$ one has*

$$R_A(M, \dots, M).$$

(ii) *Let S on $\mathcal{M}_1 \times \dots \times \mathcal{M}_n$ be a semantic logical relation. Then for each term $M \in \Lambda_{\rightarrow}^{\emptyset}(A)$ one has*

$$S_A(\llbracket M \rrbracket^{\mathcal{M}_1}, \dots, \llbracket M \rrbracket^{\mathcal{M}_n}).$$

PROOF. We show (ii) assuming (i). For notational simplicity we take $n = 1$. Therefore let $S \subseteq \mathcal{M}$ be logical and $M \in \Lambda_{\rightarrow}^{\emptyset}$, in order to show $S(\llbracket M \rrbracket)$. First we assume that S is non-empty. Then $S^* \subseteq \mathcal{M}^*$ is a substitutive semantic logical relation, by Propositions 3C.33(iii) and (ii). Writing $R = S^{*\wedge} \subseteq \Lambda_{\rightarrow}(\mathcal{M}^*)$ we have that R is an invariant (hence expansive) and substitutive logical relation, by Proposition 3C.45(i). For $M \in \Lambda_{\rightarrow}^{\emptyset}(A)$ we have $R_A(M)$, by (i), and proceed as follows.

$$\begin{aligned} R_A(M) &\Rightarrow R^{\vee}(\llbracket M \rrbracket^{\mathcal{M}^*}), && \text{by Lemma 3C.44(ii), as } M \text{ is closed,} \\ &\Rightarrow S_A^{*\wedge\vee}(\llbracket M \rrbracket^{\mathcal{M}^*}), && \text{as } R = S^{*\wedge}, \\ &\Rightarrow S_A^*(\llbracket M \rrbracket^{\mathcal{M}^*}), && \text{by Proposition 3C.46(i),} \\ &\Rightarrow S_A^*(\llbracket \llbracket M \rrbracket^{\mathcal{M}} \rrbracket), && \text{by Lemma 3C.40,} \\ &\Rightarrow S_A(\llbracket M \rrbracket^{\mathcal{M}}), && \text{by Lemma 3C.32(ii) and the assumption.} \end{aligned}$$

In case S is empty, then we also have $S_A(\llbracket M \rrbracket^{\mathcal{M}})$, by Proposition 3C.25. ■

3D. Type reducibility

In this Section we study in the context of $\lambda_{\rightarrow}^{\text{dB}}$ over \mathbb{T}^0 how equality of terms of a certain type A can be reduced to equality of terms of another type. This is the case if there is a definable injection of $\Lambda_{\rightarrow}^{\emptyset}(A)$ into $\Lambda_{\rightarrow}^{\emptyset}(B)$. The resulting poset of ‘reducibility degrees’ will turn out to be the ordinal $\omega + 4 = \{0, 1, 2, 3, \dots, \omega, \omega + 1, \omega + 2, \omega + 3\}$.

3D.1. DEFINITION. Let A, B be types of $\lambda_{\rightarrow}^{\mathbb{A}}$.

(i) We say that there is a *type reduction* from A to B (A is $\beta\eta$ *reducible* to B), notation $A \leq_{\beta\eta} B$, if for some closed term $\Phi: A \rightarrow B$ one has for all closed $M_1, M_2: A$

$$M_1 =_{\beta\eta} M_2 \Leftrightarrow \Phi M_1 =_{\beta\eta} \Phi M_2,$$

i.e. equalities between terms of type A can be uniformly translated to those of type B .

(ii) Write $A \sim_{\beta\eta} B$ iff $A \leq_{\beta\eta} B \& B \leq_{\beta\eta} A$.

(iii) Write $A <_{\beta\eta} B$ for $A \leq_{\beta\eta} B \& B \not\leq_{\beta\eta} A$.

An easy result is the following.

3D.2. LEMMA. $A = A_1 \rightarrow \dots \rightarrow A_a \rightarrow 0$ and $B = A_{\pi(1)} \rightarrow \dots \rightarrow A_{\pi(a)} \rightarrow 0$, where π is a permutation of the set $\{1, \dots, a\}$. We say that A and B are equal up to permutation of arguments. Then

(i) $B \leq_{\beta\eta} A$

$$(ii) A \sim_{\beta\eta} B.$$

PROOF. (i) We have $B \leq_{\beta\eta} A$ via

$$\Phi \equiv \lambda m:B \lambda x_1 \cdots x_a.m x_{\pi(1)} \cdots x_{\pi(a)}.$$

(ii) By (i) applied to π^{-1} . ■

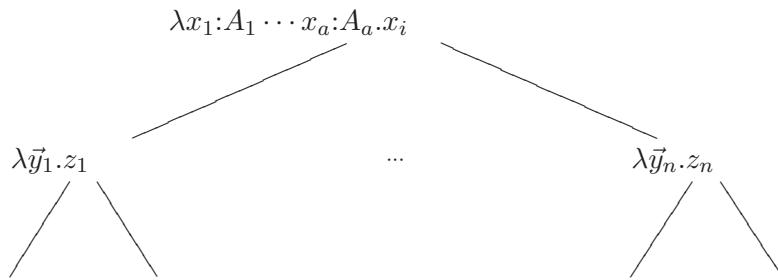
The reducibility theorem, Statman [1980a], states that there is one type to which all types of \mathbb{T}^0 can be reduced. At first this may seem impossible. Indeed, in a full type structure \mathcal{M} the cardinality of the sets of higher type increases arbitrarily. So one cannot always have an injection $\mathcal{M}_A \rightarrow \mathcal{M}_B$. But reducibility means that one restricts oneself to definable elements (modulo $=_{\beta\eta}$) and then the injections are possible. The proof will occupy¹⁰ 3D.3-3D.8. 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 will follow. 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 $\omega + 4$. 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 3F.7.

3D.3. PROPOSITION. *Every type can be reduced to a type of rank ≤ 3 , see Definition 1A.21(ii). I.e.*

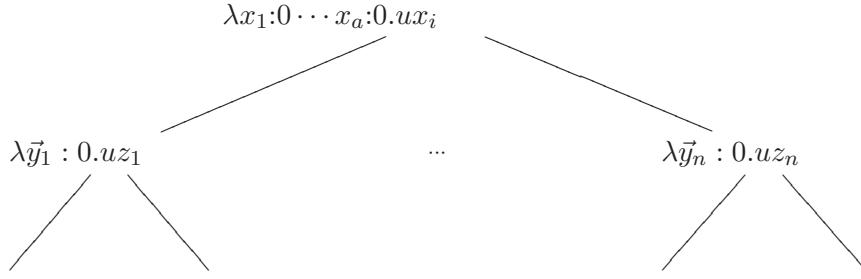
$$\forall A \in \mathbb{T}^0 \exists B \in \mathbb{T}^0. [A \leq_{\beta\eta} B \ \& \ \text{rk}(B) \leq 3].$$

PROOF. [The intuition behind the construction of the term Φ responsible for the reducibility is as follows. If M is a term with Böhm tree (see B[1984])



¹⁰A simpler alternative route discovered later by Joly is described in the exercises 3F.15 and 3F.17, needing also exercise 3F.16.

then let UM be a term with “Böhm tree” of the form



where all the typed variables are pushed down to type 0 and the variables u (each occurrence possibly different) take 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^\flat and A^\sharp .

$$\begin{aligned} 0^\flat &\triangleq 0; \\ 0^\sharp &\triangleq 0; \\ (A_1 \rightarrow \cdots \rightarrow A_a \rightarrow 0)^\flat &\triangleq (0^a \rightarrow 0); \\ (A_1 \rightarrow \cdots \rightarrow A_a \rightarrow 0)^\sharp &\triangleq 0 \rightarrow A_1^\flat \rightarrow \cdots \rightarrow A_a^\flat \rightarrow 0. \end{aligned}$$

Notice that $\text{rk}(A^\sharp) \leq 2$.

In the infinite context

$$\{u_A : A^\sharp \mid A \in \mathbb{T}\}$$

define inductively for any type A terms $V_A : 0 \rightarrow A$, $U_A : A \rightarrow A^\flat$.

$$\begin{aligned} U_0 &\triangleq \lambda x : 0. x; \\ V_0 &\triangleq \lambda x : 0. x; \\ U_{A_1 \rightarrow \cdots \rightarrow A_a \rightarrow 0} &\triangleq \lambda z : A \lambda x_1 \cdots x_a : 0. z(V_{A_1} x_1) \cdots (V_{A_a} x_a); \\ V_{A_1 \rightarrow \cdots \rightarrow A_a \rightarrow 0} &\triangleq \lambda x : 0 \lambda y_1 : A_1 \cdots y_a : A_a. u_A x(U_{A_1} y_1) \cdots (U_{A_a} y_a), \end{aligned}$$

where $A = A_1 \rightarrow \cdots \rightarrow A_a \rightarrow 0$.

Remark that for $C = A_1 \rightarrow \cdots \rightarrow A_a \rightarrow B$ one has

$$U_C = \lambda z : C \lambda x_1 \cdots x_a : 0. U_B(z(V_{A_1} x_1) \cdots (V_{A_a} x_a)). \quad (1)$$

Indeed, both sides are equal to

$$\lambda z : C \lambda x_1 \cdots x_a y_1 \cdots y_b : 0. z(V_{A_1} x_1) \cdots (V_{A_a} x_a) (V_{B_1} y_1) \cdots (V_{B_b} y_b),$$

with $B = B_1 \rightarrow \cdots \rightarrow B_b \rightarrow 0$.

Notice that for a closed term M of type $A = A_1 \rightarrow \cdots \rightarrow A_a \rightarrow 0$ one can write

$$M =_\beta \lambda y_1 : A_1 \cdots y_a : A_a. y_i(M_1 y_1 \cdots y_a) \cdots (M_n y_1 \cdots y_a),$$

with the M_1, \dots, M_n closed. Write $A_i = A_{i1} \rightarrow \cdots \rightarrow A_{in} \rightarrow 0$.

Now verify that

$$\begin{aligned} U_A M &= \lambda x_1 \cdots x_a : 0.M(V_{A_1} x_1) \cdots (V_{A_a} x_a) \\ &= \lambda \vec{x}.(V_{A_i} x_i)(M_1(V_{A_1} x_1) \cdots (V_{A_a} x_a)) \cdots (M_n(V_{A_1} x_1) \cdots (V_{A_a} x_a)) \\ &= \lambda \vec{x}.u_{A_i} x_i(U_{A_{i1}}(M_1(V_{A_1} x_1) \cdots (V_{A_a} x_a))) \cdots (U_{A_{in}}(M_n(V_{A_1} x_1) \cdots (V_{A_a} x_a))) \\ &= \lambda \vec{x}.u_{A_i} x_i(U_{B_1} M_1 \vec{x}) \cdots (U_{B_n} M_n \vec{x}), \end{aligned}$$

using (1), where $B_j = A_1 \rightarrow \cdots \rightarrow A_a \rightarrow A_{ij}$ for $1 \leq j \leq n$ is the type of M_j . Hence we have that if $U_A M =_{\beta\eta} U_A N$, then for $1 \leq j \leq n$

$$U_{B_j} M_j =_{\beta\eta} U_{B_j} N_j.$$

Therefore it follows by induction on the complexity of the β -nf of M that if $U_A M =_{\beta\eta} U_A N$, then $M =_{\beta\eta} N$.

Now take as term for the reducibility $\Phi \equiv \lambda m : A \lambda u_{B_1} \cdots u_{B_k}. U_A m$, where the \vec{u} are all the ones occurring in the construction of U_A . It follows that

$$A \leq_{\beta\eta} B_1^\sharp \rightarrow \cdots \rightarrow B_k^\sharp \rightarrow A^\flat.$$

Since $\text{rk}(B_1^\sharp \rightarrow \cdots \rightarrow B_k^\sharp \rightarrow A^\flat) \leq 3$, we are done. ■

For an alternative proof, see Exercise 3F.15.

In the following proposition it will be proved that we can further reduce types to one particular type of rank 3. First do exercise 3F.8 to get some intuition. We need the following notation.

3D.4. NOTATION. (i) Remember that for $k \geq 0$ one has

$$1_k \triangleq 0^k \rightarrow 0,$$

where in general $A^0 \rightarrow 0 \triangleq 0$ and $A^{k+1} \rightarrow 0 \triangleq A \rightarrow (A^k \rightarrow 0)$.

(ii) For $k_1, \dots, k_n \geq 0$ write

$$(k_1, \dots, k_n) \triangleq 1_{k_1} \rightarrow \cdots \rightarrow 1_{k_n} \rightarrow 0.$$

(iii) For $k_{11}, \dots, k_{1n_1}, \dots, k_{m1}, \dots, k_{mn_m} \geq 0$ write

$$\begin{pmatrix} k_{11} & \cdots & k_{1n_1} \\ \vdots & & \vdots \\ k_{m1} & \cdots & k_{mn_m} \end{pmatrix} \triangleq (k_{11}, \dots, k_{1n_1}) \rightarrow \cdots \rightarrow (k_{m1}, \dots, k_{mn_m}) \rightarrow 0.$$

Note the ‘‘matrix’’ has a dented right side (the n_i are in general unequal).

3D.5. PROPOSITION. *Every type A of rank ≤ 3 is reducible to*

$$1_2 \rightarrow 1 \rightarrow 1 \rightarrow 2 \rightarrow 0.$$

PROOF. Let A be a type of rank ≤ 3 . It is not difficult to see that A is of the form

$$A = \begin{pmatrix} k_{11} & \cdots & k_{1n_1} \\ \vdots & & \vdots \\ k_{m1} & \cdots & k_{mn_m} \end{pmatrix}$$

We will first ‘reduce’ A to type $3 = 2 \rightarrow 0$ using an open term Ψ , 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

$$p_i(pM_1M_2) \rightarrow_p M_i.$$

[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. [Extension of the notion of reduction by adding

$$p(p_1M)(p_2M) \rightarrow_s M$$

preserves the CR property, see 5B.10. In the untyped calculus this is not the case, see [Klop \[1980\]](#) or [B\[1984\]](#), ch. 14.] Goto ♠.

♠ Given the pairing p, p_1, p_2 one can extend it as follows. Write

$$\begin{aligned} p^1 &\triangleq \lambda x:0.x; \\ p^{k+1} &\triangleq \lambda x_1 \cdots x_k x_{k+1}:0.p(p^k x_1 \cdots x_k) x_{k+1}; \\ p_1^1 &\triangleq \lambda x:0.x; \\ p_{k+1}^{k+1} &\triangleq p_2; \\ p_i^{k+1} &\triangleq \lambda z:0.p_i^k(p_1 z), && \text{for } i \leq k; \\ P^k &\triangleq \lambda f_1 \cdots f_k:1 \lambda z:0.p^k(f_1 z) \cdots (f_k z); \\ P_i^k &\triangleq \lambda g:1 \lambda z:0.p_i^k(g z), && \text{for } i \leq k. \end{aligned}$$

Then $p^k : 0^k \rightarrow 0, p_i^k : 0 \rightarrow 0, P^k : 1^k \rightarrow 1, P_i^k : 1 \rightarrow 1$. We have that p^k acts as a coding for k -tuples of elements of type 0 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

$$\begin{aligned} f^{k \rightarrow 1} &\triangleq \lambda z:0.f(p_1^k z) \cdots (p_k^k z); \\ g^{1 \rightarrow k} &\triangleq \lambda z_1 \cdots z_k:0.g(p^k z_1 \cdots z_k). \end{aligned}$$

Then $f^{k \rightarrow 1}$ is f moved to type 1 and $g^{1 \rightarrow k}$ is g moved to type 1_k .

Using $\beta\eta p$ -convertibility one can show

$$\begin{aligned} p_i^k(p^k z_1 \cdots z_k) &= z_i; \\ P_i^k(P^k f_1 \cdots f_k) &= f_i; \\ (f^{k \rightarrow 1})^{1 \rightarrow k} &= f. \end{aligned}$$

For $(g^{1 \rightarrow k})^{k \rightarrow 1} = g$ one needs \rightarrow_s , the surjectivity of the pairing.

In order to define the term required for the reducibility start with a term $\Psi:A \rightarrow 3$ (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”.

$$\begin{aligned}\Psi &\equiv \lambda M:A \lambda F:2.M \\ &\quad [\lambda f_{11}:1_{k_{11}} \cdots f_{1n_1}:1_{k_{1n_1}}.p_1(F(P^{n_1}f_{11}^{k_{11} \rightarrow 1} \cdots f_{1n_1}^{k_{1n_1} \rightarrow 1}))] \cdots \\ &\quad [\lambda f_{m1}:1_{k_{m1}} \cdots f_{mn_m}:1_{k_{mn_m}}.p_m(F(P^{n_m}f_{m1}^{k_{m1} \rightarrow 1} \cdots f_{mn_m}^{k_{mn_m} \rightarrow 1}))]; \\ \Psi^{-1} &\equiv \lambda N:(2 \rightarrow 0) \lambda K_1:(k_{11}, \dots, k_{1n_1}) \cdots \lambda K_m:(k_{m1}, \dots, k_{mn_m}). \\ &\quad N(\lambda f:1.p^m[K_1(P_1^{n_1}f)^{1 \rightarrow k_{11}} \cdots (P_{n_1}^{n_1}f)^{1 \rightarrow k_{1n_1}}] \cdots \\ &\quad [K_m(P_1^{n_m}f)^{1 \rightarrow k_{m1}} \cdots (P_{n_m}^{n_m}f)^{1 \rightarrow k_{mn_m}}]).\end{aligned}$$

Claim. For closed terms M_1, M_2 of type A we have

$$M_1 =_{\beta\eta} M_2 \Leftrightarrow \Psi M_1 =_{\beta\eta} \Psi M_2.$$

It then follows that for the reduction $A \leq_{\beta\eta} 1_2 \rightarrow 1 \rightarrow 1 \rightarrow 3$ 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

$$\Psi^{-1}(\Psi M) =_{\beta\eta p} M. \tag{1}$$

We will write down the computation for the “matrix”

$$\begin{pmatrix} k_{11} & \\ k_{21} & k_{22} \end{pmatrix}$$

which is perfectly general.

$$\begin{aligned}\Psi M &=_{\beta} \lambda F:2.M[\lambda f_{11}:1_{k_{11}}.p_1(F(P^1f_{11}^{k_{11} \rightarrow 1}))] \\ &\quad [\lambda f_{21}:1_{k_{21}} \lambda f_{22}:1_{k_{22}}.p_2(F(P^2f_{21}^{k_{21} \rightarrow 1}f_{22}^{k_{22} \rightarrow 1}))]; \\ \Psi^{-1}(\Psi M) &=_{\beta} \lambda K_1:(k_{11}) \lambda K_2:(k_{21}, k_{22}). \\ &\quad \Psi M(\lambda f:1.p^1[K_1(P_1^1f)^{1 \rightarrow k_{11}}][K_2(P_1^2f)^{1 \rightarrow k_{21}}(P_2^2f)^{1 \rightarrow k_{22}}]) \\ &\equiv \lambda K_1:(k_{11}) \lambda K_2:(k_{21}, k_{22}).\Psi M \underline{H}, \text{ say,} \\ &=_{\beta} \lambda K_1 K_2.M[\lambda f_{11}.p_1(H(P^1f_{11}^{k_{11} \rightarrow 1}))] \\ &\quad [\lambda f_{21} \lambda f_{22}.p_2(H(P^2f_{21}^{k_{21} \rightarrow 1}f_{22}^{k_{22} \rightarrow 1}))]; \\ &=_{\beta p} \lambda K_1 K_2.M[\lambda f_{11}.p_1(p^2[K_1 f_{11}][..‘irrelevant’..])] \\ &\quad [\lambda f_{21} \lambda f_{22}.p_2(p^2[..‘irrelevant’..][K_2 f_{21} f_{22}])]; \\ &=_{p} \lambda K_1 K_2.M(\lambda f_{11}.K_1 f_{11})(\lambda f_{21} f_{22}.K_2 f_{21} f_{22}) \\ &=_{\eta} \lambda K_1 K_2.M K_1 K_2 \\ &=_{\eta} M,\end{aligned}$$

since

$$\begin{aligned}H(P^1f_{11}) &=_{\beta p} p^2[K_1 f_{11}][..‘irrelevant’..] \\ H(P^2f_{21}^{k_{21} \rightarrow 1}f_{22}^{k_{22} \rightarrow 1}) &=_{\beta p} p^2[..‘irrelevant’..][K_2 f_{21} f_{22}].\end{aligned}$$

The argument now can be finished in a model theoretic or syntactic way.

♣ If $\Psi M_1 =_{\beta\eta} \Psi M_2$, then $\Psi^{-1}(\Psi M_1) =_{\beta\eta} \Psi^{-1}(\Psi M_2)$. But then by (1) $M_1 =_{\beta\eta p} M_2$. It follows from the Church-Rosser theorem for $\beta\eta p$ that $M_1 =_{\beta\eta} M_2$, since these terms do not contain p . Goto ■.

♠ If $\Psi M_1 =_{\beta\eta} \Psi M_2$, then

$$\lambda p:1_2\lambda p_1p_2:1.\Psi^{-1}(\Psi M_1) =_{\beta\eta} \lambda p:1_2\lambda p_1p_2:1.\Psi^{-1}(\Psi M_2).$$

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 $\beta\eta p$ conversion it follows from (1) that

$$\mathcal{M}(\omega) \models M_1 = M_2.$$

But then $M_1 =_{\beta\eta} M_2$, by a result of Friedman [1975]. ■

We will see below, Corollary 3D.32(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 3F.10.

3D.6. PROPOSITION. *Let A be a type of rank ≤ 2 . Then*

$$2 \rightarrow A \leq_{\beta\eta} 1 \rightarrow 1 \rightarrow 0 \rightarrow A.$$

PROOF. Let $A \equiv (k_1, \dots, k_n) = 1_{k_1} \rightarrow \dots 1_{k_n} \rightarrow 0$. The term that will perform the reduction is relatively simple

$$\Phi \triangleq \lambda M:(2 \rightarrow A)\lambda f, g:1\lambda z:0.M(\lambda h:1.f(h(g(hz)))).$$

In order to show that for all $M_1, M_2:2 \rightarrow A$ one has

$$\Phi M_1 =_{\beta\eta} \Phi M_2 \Rightarrow M_1 =_{\beta\eta} M_2,$$

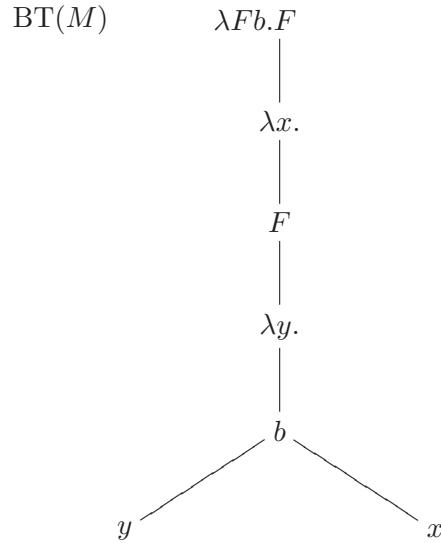
we may assume w.l.o.g. that $A = 1_2 \rightarrow 0$. A typical element of $2 \rightarrow 1_2 \rightarrow 0$ is

$$M \equiv \lambda F:2\lambda b:1_2.F(\lambda x.F(\lambda y.byx)).$$

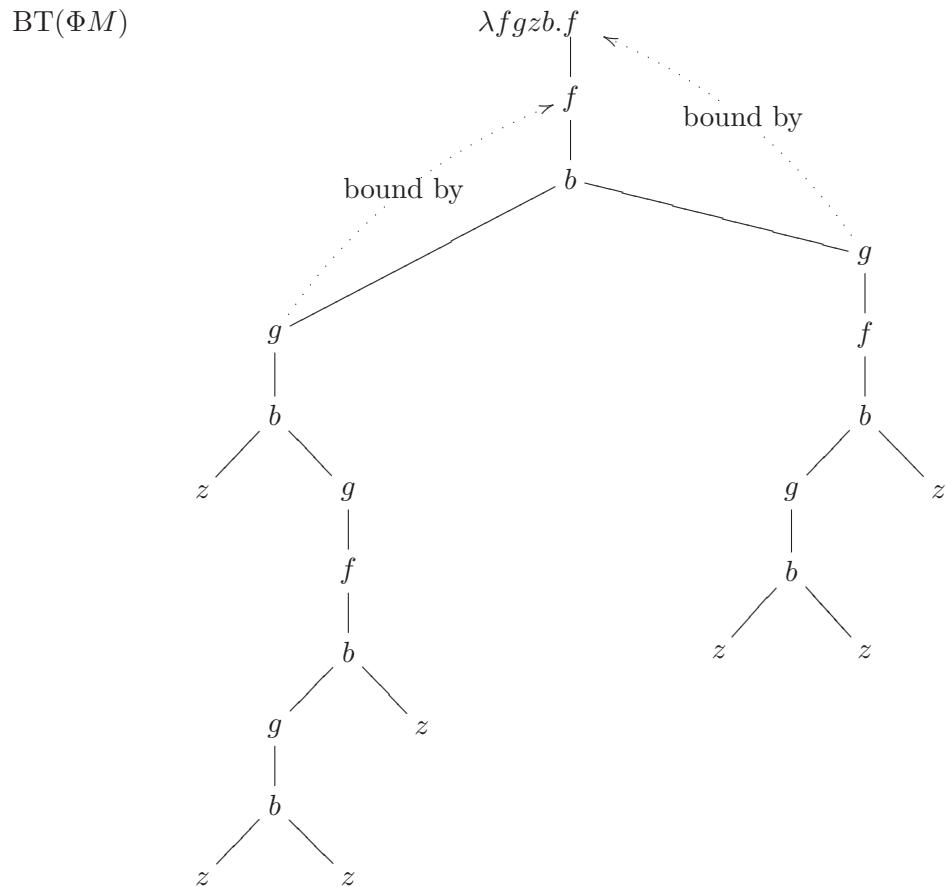
Note that its translation has the following long $\beta\eta$ -nf

$$\begin{aligned} \Phi M &= \lambda f, g:1\lambda z:0\lambda b:1_2.f(N_x[x:=g(N_x[x:=z])]), \\ &\quad \text{where } N_x \equiv f(b(g(bzx))x), \\ &\equiv \lambda f, g:1\lambda z:0\lambda b:1_2.f(f(b(g(bz[g(f(b(g(bzz))z)])))[g(f(b(g(bzz))z)]))). \end{aligned}$$

This term M and its translation have the following trees.



and



Note that if we can ‘read back’ M from its translation ΦM , then we are done. Let $\text{Cut}_{g \rightarrow z}$ be a syntactic operation on terms that replaces maximal subterms of the form gP by z . For example (omitting the abstraction prefix)

$$\text{Cut}_{g \rightarrow z}(\Phi M) = f(f(bzz)).$$

Note that this gives us back the ‘skeleton’ of the term M , by reading $f \dots$ as $F(\lambda \odot \dots)$. 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 realizing that the occurrence z coming from $g(P)$ should be bound at the position f just above where $\text{Cut}_{g \rightarrow z}(P)$ matches in $\text{Cut}_{g \rightarrow z}(\Phi M)$ above that z . For a precise inductive argument for this fact, see [Statman \[1980a\]](#), Lemma 5, or do exercise 3F.16. ■

The following simple proposition brings almost to an end the chain of reducibility of types.

3D.7. PROPOSITION.

$$1^4 \rightarrow 1_2 \rightarrow 0 \rightarrow 0 \leq_{\beta\eta} 1_2 \rightarrow 0 \rightarrow 0.$$

PROOF. As it is equally simple, let us prove instead

$$1 \rightarrow 1_2 \rightarrow 0 \rightarrow 0 \leq_{\beta\eta} 1_2 \rightarrow 0 \rightarrow 0.$$

Define $\Phi : (1 \rightarrow 1_2 \rightarrow 0 \rightarrow 0) \rightarrow 1_2 \rightarrow 0 \rightarrow 0$ by

$$\Phi \triangleq \lambda M : (1 \rightarrow 1_2 \rightarrow 0 \rightarrow 0) \lambda b : 1_2 \lambda c : 0. M(f^+)(b^+)c,$$

where

$$\begin{aligned} f^+ &\triangleq \lambda t : 0. b(\#f)t; \\ b^+ &\triangleq \lambda t_1 t_2 : 0. b(\#b)(bt_1 t_2); \\ \#f &\triangleq bcc; \\ \#b &\triangleq bc(bcc). \end{aligned}$$

The terms $\#f, \#b$ serve as ‘tags’. Notice that M of type $1 \rightarrow 1_2 \rightarrow 0 \rightarrow 0$ has a closed long $\beta\eta$ -nf of the form

$$M^{\text{nf}} \equiv \lambda f : 1 \lambda b : 1_2 \lambda c : 0. t$$

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 $\Phi M =_{\beta\eta} \Phi(M^{\text{nf}}) \equiv M^+$ with

$$M^+ \equiv \lambda f : 1 \lambda b : 1_2 \lambda c : 0. t^+,$$

where t^+ is inductively defined by

$$\begin{aligned} c^+ &\triangleq c; \\ (ft)^+ &\triangleq b(\#f)t^+; \\ (bt_1 t_2)^+ &\triangleq b(\#b)(bt_1^+ t_2^+). \end{aligned}$$

It is clear that M^{nf} can be constructed back from M^+ . Therefore

$$\begin{aligned}\Phi M_1 =_{\beta\eta} \Phi M_2 &\Rightarrow M_1^+ =_{\beta\eta} M_2^+ \\ &\Rightarrow M_1^+ \equiv M_2^+ \\ &\Rightarrow M_1^{\text{nf}} \equiv M_2^{\text{nf}} \\ &\Rightarrow M_1 =_{\beta\eta} M_2. \blacksquare\end{aligned}$$

Similarly one can show that any type of rank ≤ 2 is reducible to \top^2 , do exercise 3F.19
Combining Propositions 3D.3-3D.7 we obtain the reducibility theorem.

3D.8. THEOREM (Reducibility Theorem, Statman [1980a]). *Let*

$$\top^2 \triangleq 1_2 \rightarrow 0 \rightarrow 0.$$

Then

$$\forall A \in \mathbb{T}^0 \quad A \leq_{\beta\eta} \top^2.$$

PROOF. Let A be any type. Harvesting the results we obtain

$$\begin{aligned}A &\leq_{\beta\eta} B, && \text{with } \text{rk}(B) \leq 3, \text{ by 3D.3,} \\ &\leq_{\beta\eta} 1_2 \rightarrow 1^2 \rightarrow 2 \rightarrow 0, && \text{by 3D.5,} \\ &\leq_{\beta\eta} 2 \rightarrow 1_2 \rightarrow 1^2 \rightarrow 0, && \text{by simply permuting arguments,} \\ &\leq_{\beta\eta} 1^2 \rightarrow 0 \rightarrow 1_2 \rightarrow 1^2 \rightarrow 0, && \text{by 3D.6,} \\ &\leq_{\beta\eta} 1_2 \rightarrow 0 \rightarrow 0, && \text{by an other permutation and 3D.7} \blacksquare\end{aligned}$$

Now we turn attention to the type hierarchy, Statman [1980a].

3D.9. DEFINITION. For the ordinals $\alpha \leq \omega + 3$ define the type $A_\alpha \in \mathbb{T}^0$ as follows.

$$\begin{aligned}A_0 &\triangleq 0; \\ A_1 &\triangleq 0 \rightarrow 0; \\ &\dots \\ A_k &\triangleq 0^k \rightarrow 0; \\ &\dots \\ A_\omega &\triangleq 1 \rightarrow 0 \rightarrow 0; \\ A_{\omega+1} &\triangleq 1 \rightarrow 1 \rightarrow 0 \rightarrow 0; \\ A_{\omega+2} &\triangleq 3 \rightarrow 0 \rightarrow 0; \\ A_{\omega+3} &\triangleq 1_2 \rightarrow 0 \rightarrow 0.\end{aligned}$$

3D.10. 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

$$\Phi_{k,k+1} \triangleq \lambda m : A_k \lambda z x_1 \cdots x_k : 0. m x_1 \cdots x_k =_{\beta\eta} \lambda m : A_k. K m.$$

Moreover, $A_k \leq_{\beta\eta} A_\omega$ via

$$\Phi_{k,\omega} \triangleq \lambda m : A_k \lambda f : 1 \lambda x : 0. m(\mathbf{c}_1 f x) \cdots (\mathbf{c}_k f x).$$

Then $A_\omega \leq_{\beta\eta} A_{\omega+1}$ via

$$\Phi_{\omega,\omega+1} \triangleq \lambda m : A_\omega \lambda f, g : 1 \lambda x : 0. mfx.$$

Now $A_{\omega+1} \leq_{\beta\eta} A_{\omega+2}$ via

$$\Phi_{\omega+1,\omega+2} \triangleq \lambda m : A_{\omega+1} \lambda H : 3 \lambda x : 0. H(\lambda f : 1. H(\lambda g : 1. mfgx)).$$

Finally, $A_{\omega+2} \leq_{\beta\eta} A_{\omega+3} = \top^2$ by the reducibility Theorem 3D.8. Do Exercise 3F.18 that asks for a concrete term $\Phi_{\omega+2,\omega+3}$. ■

3D.11. PROPOSITION. *For $\alpha, \beta \leq \omega + 3$ one has*

$$\alpha \leq \beta \Leftrightarrow A_\alpha \leq_{\beta\eta} A_\beta.$$

PROOF. This will be proved in 3E.52. ■

3D.12. COROLLARY. *For $\alpha, \beta \leq \omega + 3$ one has*

$$A_\alpha \leq_{\beta\eta} A_\beta \Leftrightarrow \alpha \leq \beta. \blacksquare$$

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.

3D.13. DEFINITION. A type $A \in \mathbb{T}$ is called *large* if it has a negative subterm occurrence, see Definition 9C.1, of the form $B_1 \rightarrow \dots \rightarrow B_n \rightarrow 0$, with $n \geq 2$; A is *small* otherwise.

3D.14. EXAMPLE. $1_2 \rightarrow 0 \rightarrow 0$ and $((1_2 \rightarrow 0) \rightarrow 0) \rightarrow 0$ are large; $(1_2 \rightarrow 0) \rightarrow 0$ and $3 \rightarrow 0 \rightarrow 0$ are small.

Now we will partition the types $\mathbb{T} = \mathbb{T}^0$ in the following classes.

3D.15. DEFINITION (Type Hierarchy). Define the following sets of types.

$$\begin{aligned} \mathbb{T}_{-1} &\triangleq \{A \mid A \text{ is not inhabited}\}; \\ \mathbb{T}_0 &\triangleq \{A \mid A \text{ is inhabited, small, } \text{rk}(A) = 1 \text{ and} \\ &\quad A \text{ has exactly one component of rank 0}\}; \\ \mathbb{T}_1 &\triangleq \{A \mid A \text{ is inhabited, small, } \text{rk}(A) = 1 \text{ and} \\ &\quad A \text{ has at least two components of rank 0}\}; \\ \mathbb{T}_2 &\triangleq \{A \mid A \text{ is inhabited, small, } \text{rk}(A) \in \{2, 3\} \text{ and} \\ &\quad A \text{ has exactly one component of rank } \geq 1\}; \\ \mathbb{T}_3 &\triangleq \{A \mid A \text{ is inhabited, small, } \text{rk}(A) \in \{2, 3\} \text{ and} \\ &\quad A \text{ has at least two components of rank } \geq 1\}; \\ \mathbb{T}_4 &\triangleq \{A \mid A \text{ is inhabited, small and } \text{rk}(A) > 3\}; \\ \mathbb{T}_5 &\triangleq \{A \mid A \text{ is inhabited and large}\}. \end{aligned}$$

Typical elements of \mathbb{T}_{-1} are $0, 2, 4, \dots$. This class we will not consider much. The types in $\mathbb{T}_0, \dots, \mathbb{T}_5$ are all inhabited. The unique element of \mathbb{T}_0 is $1 = 0 \rightarrow 0$ and the elements of \mathbb{T}_1 are 1_p , with $p \geq 2$, see the next Lemma. Typical elements of \mathbb{T}_2 are $1 \rightarrow 0 \rightarrow 0, 2 \rightarrow 0$ and also $0 \rightarrow 1 \rightarrow 0 \rightarrow 0, 0 \rightarrow (1_3 \rightarrow 0) \rightarrow 0 \rightarrow 0$. The types in $\mathbb{T}_1, \dots, \mathbb{T}_4$ are all small. Types in $\mathbb{T}_0 \cup \mathbb{T}_1$ all have rank 1; types in $\mathbb{T}_2 \cup \dots \cup \mathbb{T}_5$ all have rank ≥ 2 .

Examples of types of rank 2 not in \mathbb{T}_2 are $(1 \rightarrow 1 \rightarrow 0 \rightarrow 0) \in \mathbb{T}_3$ and $(1_2 \rightarrow 0 \rightarrow 0) \in \mathbb{T}_5$. Examples of types of rank 3 not in \mathbb{T}_2 are $((1_2 \rightarrow 0) \rightarrow 1 \rightarrow 0) \in \mathbb{T}_3$ and $((1 \rightarrow 1 \rightarrow 0) \rightarrow 0 \rightarrow 0) \in \mathbb{T}_5$.

3D.16. LEMMA. *Let $A \in \mathbb{T}$. Then*

- (i) $A \in \mathbb{T}_0$ iff $A = (0 \rightarrow 0)$.
- (ii) $A \in \mathbb{T}_1$ iff $A = (0^p \rightarrow 0)$, for $p \geq 2$.

(iii) $A \in \mathbb{T}_2$ iff up to permutation of components

$$A \in \{(1_p \rightarrow 0) \rightarrow 0^q \rightarrow 0 \mid p \geq 1, q \geq 0\} \cup \{1 \rightarrow 0^q \rightarrow 0 \mid q \geq 1\}$$

PROOF. (i), (ii) If $\text{rk}(A) = 1$, then $A = 0^p \rightarrow 0$, $p \geq 1$. If $A \in \mathbb{T}_0$, then $p = 1$; if $A \in \mathbb{T}_1$, then $p \geq 2$. The converse implications are obvious.

(iii) Clearly the displayed types all belong to \mathbb{T}_2 . Conversely, let $A \in \mathbb{T}_2$. Then A is inhabited and small with rank in $\{2, 3\}$ and only one component of maximal rank.

Case $\text{rk}(A) = 2$. Then $A = A_1 \rightarrow \dots \rightarrow A_a \rightarrow 0$, with $\text{rk}(A_i) \leq 1$ and exactly one A_j has rank 1. Then up to permutation $A = (0^p \rightarrow 0) \rightarrow 0^q \rightarrow 0$. Since A is small $p = 1$; since A is inhabited $q \geq 1$; therefore $A = 1 \rightarrow 0^q \rightarrow 0$, in this case.

Case $\text{rk}(A) = 3$. Then it follows similarly that $A = A_1 \rightarrow 0^q \rightarrow 0$, with $A_1 = B \rightarrow 0$ and $\text{rk}(B) = 1$. Then $B = 1_p$ with $p \geq 1$. Therefore $A = (1_p \rightarrow 0) \rightarrow 0^q \rightarrow 0$, where now $q = 0$ is possible, since $(1_p \rightarrow 0) \rightarrow 0$ is already inhabited by $\lambda m.m(\lambda x_1 \dots x_p.x_1)$. ■

3D.17. PROPOSITION. *The \mathbb{T}_i form a partition of \mathbb{T}^0 .*

PROOF. The classes are disjoint by definition.

Any type of rank ≤ 1 belongs to $\mathbb{T}_{-1} \cup \mathbb{T}_0 \cup \mathbb{T}_1$. Any type of rank ≥ 2 is either not inhabited and then belongs to \mathbb{T}_{-1} , or belongs to $\mathbb{T}_2 \cup \mathbb{T}_3 \cup \mathbb{T}_4 \cup \mathbb{T}_5$. ■

3D.18. THEOREM (Hierarchy Theorem, Statman [1980a]). (i) *The set of types \mathbb{T}^0 over the unique groundtype 0 is partitioned in the classes $\mathbb{T}_{-1}, \mathbb{T}_0, \mathbb{T}_1, \mathbb{T}_2, \mathbb{T}_3, \mathbb{T}_4, \mathbb{T}_5$.*

$$\begin{aligned} \text{(ii) Moreover, } A \in \mathbb{T}_5 &\Leftrightarrow A \sim_{\beta\eta} 1_2 \rightarrow 0 \rightarrow 0; \\ A \in \mathbb{T}_4 &\Leftrightarrow A \sim_{\beta\eta} 3 \rightarrow 0 \rightarrow 0; \\ A \in \mathbb{T}_3 &\Leftrightarrow A \sim_{\beta\eta} 1 \rightarrow 1 \rightarrow 0 \rightarrow 0; \\ A \in \mathbb{T}_2 &\Leftrightarrow A \sim_{\beta\eta} 1 \rightarrow 0 \rightarrow 0; \\ A \in \mathbb{T}_1 &\Leftrightarrow A \sim_{\beta\eta} 0^k \rightarrow 0, \quad \text{for some } k > 1; \\ A \in \mathbb{T}_0 &\Leftrightarrow A \sim_{\beta\eta} 0 \rightarrow 0; \\ A \in \mathbb{T}_{-1} &\Leftrightarrow A \sim_{\beta\eta} 0. \end{aligned}$$

$$\begin{aligned} \text{(iii)} \quad 0 &\quad \begin{array}{l} \sim_{\beta\eta} 0 \rightarrow 0 \\ \sim_{\beta\eta} 0^2 \rightarrow 0 \\ \sim_{\beta\eta} \dots \\ \sim_{\beta\eta} 0^k \rightarrow 0 \\ \sim_{\beta\eta} \dots \\ \sim_{\beta\eta} 1 \rightarrow 0 \rightarrow 0 \\ \sim_{\beta\eta} 1 \rightarrow 1 \rightarrow 0 \rightarrow 0 \\ \sim_{\beta\eta} 3 \rightarrow 0 \rightarrow 0 \\ \sim_{\beta\eta} 1_2 \rightarrow 0 \rightarrow 0 \end{array} \quad \begin{array}{l} \in \mathbb{T}_0 \\ \left. \begin{array}{c} \sim_{\beta\eta} \\ \dots \end{array} \right\} \in \mathbb{T}_1 \\ \in \mathbb{T}_2 \\ \in \mathbb{T}_3 \\ \in \mathbb{T}_4 \\ \in \mathbb{T}_5. \end{array} \end{aligned}$$

PROOF. (i) By Proposition 3D.17.

(ii) By (i) and Corollary 3D.12 it suffices to show just the \Rightarrow 's.

As to \mathbb{T}_5 , it is enough to show that $1_2 \rightarrow 0 \rightarrow 0 \leq_{\beta\eta} A$, for every **inhabited** large type A , since we know already the converse. For this, see Statman [1980a], Lemma 7. As a warm-up exercise do 3F.26.

As to \mathbb{T}_4 , it is shown in Statman [1980a], Proposition 2, that if A is small, then $A \leq_{\beta\eta} 3 \rightarrow 0 \rightarrow 0$. It remains to show that for any small inhabited type A of rank > 3 one has $3 \rightarrow 0 \rightarrow 0 \leq_{\beta\eta} A$. Do exercise 3F.30.

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 3F.28(ii).

As to \mathbb{T}_i , with $i = 1, 0, -1$, notice that $\Lambda^\phi(0^k \rightarrow 0)$ contains exactly k closed terms for $k \geq 0$. This is sufficient.

(iii) By Corollary 3D.12. ■

3D.19. DEFINITION. Let $A \in \mathbb{T}^0$. The *class* of A , notation $\text{class}(A)$, is the unique i with $i \in \{-1, 0, 1, 2, 3, 4, 5\}$ such that $A \in \mathbb{T}_i$.

3D.20. REMARK. (i) Note that by the Hierarchy theorem one has for all $A, B \in \mathbb{T}^0$

$$A \leq_{\beta\eta} B \Rightarrow \text{class}(A) \leq \text{class}(B).$$

(ii) As $B \leq_{\beta\eta} A \rightarrow B$ via the map $\Phi = \lambda x^B y^A.x$, this implies

$$\text{class}(B) \leq \text{class}(A \rightarrow B).$$

3D.21. REMARK. Let $C_{-1} \triangleq 0$,

$$\begin{aligned} C_0 &\triangleq 0 \rightarrow 0, \\ C_{1,k} &\triangleq 0^k \rightarrow 0, \quad \text{with } k > 1, \\ C_1 &\triangleq 0^2 \rightarrow 0, \\ C_2 &\triangleq 1 \rightarrow 0 \rightarrow 0, \\ C_3 &\triangleq 1 \rightarrow 1 \rightarrow 0 \rightarrow 0, \\ C_4 &\triangleq 3 \rightarrow 0 \rightarrow 0, \\ C_5 &\triangleq 1_2 \rightarrow 0 \rightarrow 0. \end{aligned}$$

Then for $A \in \mathbb{T}^0$ one has

(i) If $i \neq 1$, then

$$\text{class}(A) = i \Leftrightarrow A \sim_{\beta\eta} C_i.$$

$$\begin{aligned} \text{(ii)} \quad \text{class}(A) = 1 &\Leftrightarrow \exists k. A \sim_{\beta\eta} C_{1,k}. \\ &\Leftrightarrow \exists k. A \equiv C_{1,k}. \end{aligned}$$

This follows from the Hierarchy Theorem.

For an application in the next section we need a variant of the hierarchy theorem.

3D.22. DEFINITION. Let $A \equiv A_1 \rightarrow \dots \rightarrow A_a \rightarrow 0$, $B \equiv B_1 \rightarrow \dots \rightarrow B_b \rightarrow 0$ be types.

(i) A is *head-reducible* to B , notation $A \leq_h B$, iff for some term $\Phi \in \Lambda^\phi(A \rightarrow B)$ one has

$$\forall M_1, M_2 \in \Lambda^\phi(A) [M_1 =_{\beta\eta} M_2 \Leftrightarrow \Phi M_1 =_{\beta\eta} \Phi M_2],$$

and moreover Φ is of the form

$$\Phi = \lambda m : A \lambda x_1 : B_0 \cdots x_b : B_b. m P_1 \cdots P_a, \tag{1}$$

with $\text{FV}(P_1, \dots, P_a) \subseteq \{x_1, \dots, x_b\}$ and $m \notin \{x_1 \cdots x_b\}$.

(ii) A is *multi head-reducible* to B , notation $A \leq_{h^+} B$, iff there are closed terms $\Phi_1, \dots, \Phi_m \in \Lambda^\phi(A \rightarrow B)$ each of the form (1) such that

$$\forall M_1, M_2 \in \Lambda^\phi(A) [M_1 =_{\beta\eta} M_2 \Leftrightarrow \Phi_1 M_1 =_{\beta\eta} \Phi_1 M_2 \And \dots \And \Phi_m M_1 =_{\beta\eta} \Phi_m M_2].$$

(iii) Write $A \sim_h B$ iff $A \leq_h B \leq_h A$ and similarly
 $A \sim_{h^+} B$ iff $A \leq_{h^+} B \leq_{h^+} A$.

Clearly $A \leq_h B \Rightarrow A \leq_{h^+} B$. Moreover, both \leq_h and \leq_{h^+} are transitive, do Exercise 3F.14. We will formulate in Corollary 3D.27 a variant of the hierarchy theorem.

3D.23. LEMMA. $0 \leq_h 1 \leq_h 0^2 \rightarrow 0 \leq_h 1 \rightarrow 0 \rightarrow 0 \leq_h 1 \rightarrow 1 \rightarrow 0 \rightarrow 0$.

PROOF. By inspecting the proof of Proposition 3D.10. ■

3D.24. LEMMA. (i) $1 \rightarrow 0 \rightarrow 0 \not\leq_{h^+} 0^k \rightarrow 0$, for $k \geq 0$.

(ii) If $A \leq_{h^+} 1 \rightarrow 0 \rightarrow 0$, then $A \leq_{\beta\eta} 1 \rightarrow 0 \rightarrow 0$.

(iii) $1_2 \rightarrow 0 \rightarrow 0 \not\leq_{h^+} 1 \rightarrow 0 \rightarrow 0$, $3 \rightarrow 0 \rightarrow 0 \not\leq_{h^+} 1 \rightarrow 0 \rightarrow 0$, and $1 \rightarrow 1 \rightarrow 0 \rightarrow 0 \not\leq_{h^+} 1 \rightarrow 0 \rightarrow 0$.

(iv) $0^2 \rightarrow 0 \not\leq_{h^+} 0 \rightarrow 0$.

(v) Let $A, B \in \mathbb{T}^0$. If $\Lambda_{\rightarrow}^{\phi}(A)$ is infinite and $\Lambda_{\rightarrow}^{\phi}(B)$ finite, then $A \not\leq_{h^+} B$.

PROOF. (i) By a cardinality argument: $\Lambda_{\rightarrow}^{\phi}(1 \rightarrow 0 \rightarrow 0)$ contains infinitely many different elements. These cannot be mapped injectively into the finite $\Lambda_{\rightarrow}^{\phi}(0^k \rightarrow 0)$, not even in the way of \leq_{h^+} .

(ii) Suppose $A \leq_{h^+} 1 \rightarrow 0 \rightarrow 0$ via Φ_1, \dots, Φ_k . Then each element M of $\Lambda_{\rightarrow}^{\phi}(A)$ is mapped to a k -tuple of Church numerals $\langle \Phi_1(M), \dots, \Phi_k(M) \rangle$. This k -tuple can be coded as a single numeral by iterating the Cantorian pairing function on the natural numbers, which is polynomially definable and hence λ -definable.

(iii) By (ii) and the Hierarchy Theorem.

(iv) Type $0^2 \rightarrow 0$ contains two closed terms. These cannot be mapped injectively into the singleton $\Lambda_{\rightarrow}^{\phi}(0 \rightarrow 0)$, even not by the multiple maps.

(v) Suppose $A \leq_{h^+} B$ via Φ_1, \dots, Φ_k . Then the sequences $\langle \Phi_1(M), \dots, \Phi_k(M) \rangle$ are all different for $M \in \Lambda_{\rightarrow}^{\phi}(A)$. As B is finite (with say m elements), there are only finitely many sequences of length k (in fact m^k). This is impossible as $\Lambda_{\rightarrow}^{\phi}(A)$ is infinite. ■

3D.25. PROPOSITION. Let $A, B \in \mathbb{T}_i^0$. Then

(i) If $i \notin \{1, 2\}$, then $A \sim_h B$.

(ii) If $i \in \{1, 2\}$, then $A \sim_{h^+} B$.

PROOF. (i) Since $A, B \in \mathbb{T}_i$ and $i \neq 1$ one has by Theorem 3D.18 $A \sim_{\beta\eta} B$. By inspection of the proof of that theorem in all cases except for $A \in \mathbb{T}_2$ one obtains $A \sim_h B$. Do exercise 3F.29.

(ii) Case $i = 1$. We must show that $1_2 \sim_{h^+} 1_k$ for all $k \geq 2$. It is easy to show that $1_2 \leq_h 1_p$, for $p \geq 2$. It remains to verify that $1_k \leq_{h^+} 1_2$ for $k \geq 2$. W.l.o.g. take $k = 3$. Then $M \in \Lambda_{\rightarrow}^{\phi}(1_3)$ is of the form $M \equiv \lambda x_1 x_2 x_3. x_i$. Hence for $M, N \in \Lambda_{\rightarrow}^{\phi}(1_3)$ with $M \neq_{\beta\eta} N$ either

$$\lambda y_1 y_2. M y_1 y_1 y_2 \neq_{\beta\eta} \lambda y_1 y_2. N y_1 y_1 y_2 \text{ or } \lambda y_1 y_2. M y_1 y_2 y_2 \neq_{\beta\eta} \lambda y_1 y_2. N y_1 y_2 y_2.$$

Hence $1_3 \leq_{h^+} 1_2$.

Case $i = 2$. Do Exercise 3F.28. ■

3D.26. COROLLARY. Let $A, B \in \mathbb{T}^0$, with $A = A_1 \rightarrow \dots \rightarrow A_a \rightarrow 0$, $B = B_1 \rightarrow \dots \rightarrow B_b \rightarrow 0$.

(i) $A \sim_h B \Rightarrow A \sim_{\beta\eta} B$.

(ii) $A \sim_{\beta\eta} B \Rightarrow A \sim_{h^+} B$.

(iii) Suppose $A \leq_{h^+} B$. Then for $M, N \in \Lambda^{\phi}(A)$

$$M \neq_{\beta\eta} N (: A) \Rightarrow \lambda \vec{x}. M R_1 \cdots R_a \neq_{\beta\eta} \lambda \vec{x}. N R_1 \cdots R_a (: B),$$

for some fixed R_1, \dots, R_a with $\text{FV}(\vec{R}) \subseteq \{\vec{x}\} = \{x_1^{B_1}, \dots, x_b^{B_b}\}$.

PROOF. (i) Trivially one has $A \leq_h B \Rightarrow A \leq_{\beta\eta} B$. The result follows.

- (ii) By the Proposition and the hierarchy theorem.
- (iii) By the definition of \leq_{h^+} . ■

3D.27. COROLLARY (Hierarchy Theorem Revisited, Statman [1980b]).

$$\begin{aligned}
 A \in \mathbb{T}_5 &\Leftrightarrow A \sim_h 1_2 \rightarrow 0 \rightarrow 0; \\
 A \in \mathbb{T}_4 &\Leftrightarrow A \sim_h 3 \rightarrow 0 \rightarrow 0; \\
 A \in \mathbb{T}_3 &\Leftrightarrow A \sim_h 1 \rightarrow 1 \rightarrow 0 \rightarrow 0; \\
 A \in \mathbb{T}_2 &\Leftrightarrow A \sim_{h^+} 1 \rightarrow 0 \rightarrow 0; \\
 A \in \mathbb{T}_1 &\Leftrightarrow A \sim_{h^+} 0^2 \rightarrow 0; \\
 A \in \mathbb{T}_0 &\Leftrightarrow A \sim_h 0 \rightarrow 0; \\
 A \in \mathbb{T}_{-1} &\Leftrightarrow A \sim_h 0.
 \end{aligned}$$

PROOF. The Hierarchy Theorem 3D.18 and Proposition 3D.25 establish the \Rightarrow implications. As \sim_h implies $\sim_{\beta\eta}$, the \Leftarrow we only have to prove for $A \sim_{h^+} 1 \rightarrow 0 \rightarrow 0$ and $A \sim_{h^+} 0^2 \rightarrow 0$. Suppose $A \sim_{h^+} 1 \rightarrow 0 \rightarrow 0$, but $A \notin \mathbb{T}_2$. Again by the Hierarchy Theorem one has $A \in \mathbb{T}_3 \cup \mathbb{T}_4 \cup \mathbb{T}_5$ or $A \in \mathbb{T}_{-1} \cup \mathbb{T}_0 \cup \mathbb{T}_1$. If $A \in \mathbb{T}_3$, then $A \sim_{\beta\eta} 1 \rightarrow 1 \rightarrow 0 \rightarrow 0$, hence $A \sim_{h^+} 1 \rightarrow 1 \rightarrow 0 \rightarrow 0$. Then $1 \rightarrow 0 \rightarrow 0 \sim_{h^+} 1 \rightarrow 1 \rightarrow 0 \rightarrow 0$, contradicting Lemma 3D.24(ii). If $A \in \mathbb{T}_4$ or $A \in \mathbb{T}_5$, then a contradiction can be obtained similarly.

In the second case A is either empty or $A \equiv 0^k \rightarrow 0$, for some $k > 0$; moreover $1 \rightarrow 0 \rightarrow 0 \leq_{h^+} A$. The subcase that A is empty cannot occur, since $1 \rightarrow 0 \rightarrow 0$ is inhabited. The subcase $A \equiv 0^k \rightarrow 0$, contradicts Lemma 3D.24(i).

Finally, suppose $A \sim_{h^+} 0^2 \rightarrow 0$ and $A \notin \mathbb{T}_1$. If $A \in \mathbb{T}_{-1} \cup \mathbb{T}_0$, then $\Lambda_{\rightarrow}^{\phi}(A)$ has at most one element. This contradicts $0^2 \rightarrow 0 \leq_{h^+} A$, as $0^2 \rightarrow 0$ has two distinct elements. If $A \in \mathbb{T}_2 \cup \mathbb{T}_3 \cup \mathbb{T}_4 \cup \mathbb{T}_5$, then $1 \rightarrow 0 \rightarrow 0 \leq_{\beta\eta} A \leq_{h^+} 0^2 \rightarrow 0$, giving A infinitely many closed inhabitants, contradicting Lemma 3D.24(v). ■

Applications of the reducibility theorem

The reducibility theorem has several consequences.

3D.28. DEFINITION. Let \mathcal{C} be a class of $\lambda_{\rightarrow}^{\text{Ch}}$ models. \mathcal{C} is called *complete* if

$$\forall M, N \in \Lambda^{\phi}[\mathcal{C} \models M = N \Leftrightarrow M =_{\beta\eta} N].$$

3D.29. DEFINITION. (i) $\mathcal{T} = \mathcal{T}_{b,c}$ is the algebraic structure of trees inductively defined as follows.

$$\mathcal{T} ::= c \mid b \mid \mathcal{T} \mathcal{T}$$

(ii) For a typed λ -model \mathcal{M} we say that \mathcal{T} *can be embedded into* \mathcal{M} , notation $\mathcal{T} \hookrightarrow \mathcal{M}$, if there exist $b_0 \in \mathcal{M}(0 \rightarrow 0 \rightarrow 0)$, $c_0 \in \mathcal{M}(0)$ such that

$$\forall t, s \in \mathcal{T}[t \neq s \Rightarrow \mathcal{M} \models t^{\text{cl}} b_0 c_0 \neq s^{\text{cl}} b_0 c_0],$$

where $u^{\text{cl}} = \lambda b:0 \rightarrow 0 \rightarrow 0 \lambda c:0. u$, 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 embedding using b_0, c_0 implies for example that $b_0 c_0 (b_0 c_0 c_0), b_0 c_0 c_0$ and c_0 are mutually different in \mathcal{M} .

Note that $\mathcal{T} \not\hookrightarrow \mathcal{M}_2 (= \mathcal{M}_{\{1,2\}})$. To see this, write $gx = bxx$. One has $g^2(c) \neq g^4(c)$, but $\mathcal{M}_2 \models \forall g:0 \rightarrow 0 \forall c:0. g^2(c) = g^4(c)$, do exercise 3F.20.

Remember that $\top^2 = 1_2 \rightarrow 0 \rightarrow 0$, the type of binary trees, see Definition 1D.12.

3D.30. LEMMA. (i) $\Pi_{i \in I} \mathcal{M}_i \models M = N \Leftrightarrow \forall i \in I. \mathcal{M}_i \models M = N$.
(ii) $M \in \Lambda^\varnothing(\top^2) \Leftrightarrow \exists s \in \mathcal{T}. M =_{\beta\eta} s^{\text{cl}}$.

PROOF. (i) Since $\llbracket M \rrbracket^{\Pi_{i \in I} \mathcal{M}_i} = \lambda i \in I. \llbracket M \rrbracket^{\mathcal{M}_i}$.

(ii) By an analysis of the possible shapes of the normal forms of terms of type \top^2 . ■

3D.31. 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 structures 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

$$\begin{aligned} t \neq s &\Rightarrow t^{\text{cl}} \neq_{\beta\eta} s^{\text{cl}} \\ &\Rightarrow \mathcal{C} \not\models t^{\text{cl}} = s^{\text{cl}}, && \text{by completeness,} \\ &\Rightarrow \mathcal{M}_{ts} \models t^{\text{cl}} \neq s^{\text{cl}}, && \text{for some } \mathcal{M}_{ts} \in \mathcal{C}, \\ &\Rightarrow \mathcal{M}_{ts} \models t^{\text{cl}} b_{ts} c_{ts} \neq s^{\text{cl}} b_{ts} c_{ts}, \end{aligned}$$

for some $b_{ts} \in \mathcal{M}(0 \rightarrow 0 \rightarrow 0)$, $c_{ts} \in \mathcal{M}(0)$ by extensionality. Note that in the third implication the axiom of (countable) choice is used.

It now follows by Lemma 3D.30(i) that we can take as countable product $\Pi_{t' \neq s'} \mathcal{M}_{t's'}$

$$\Pi_{t' \neq s'} \mathcal{M}_{t's'} \models t^{\text{cl}} \neq s^{\text{cl}},$$

since they differ on the pair $b_0 c_0$ 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

$$M =_{\beta\eta} N \Rightarrow \mathcal{C} \models M = N.$$

For the converse, let by the reducibility theorem $F : A \rightarrow \top^2$ be such that

$$M =_{\beta\eta} N \Leftrightarrow FM =_{\beta\eta} FN,$$

for all $M, N \in \Lambda_\rightarrow^\varnothing$. Then

$$\begin{aligned} \mathcal{C} \models M = N &\Rightarrow \Pi_{i \in I} \mathcal{M}_i \models M = N, && \text{by the lemma,} \\ &\Rightarrow \Pi_{i \in I} \mathcal{M}_i \models FM = FN, \\ &\Rightarrow \Pi_{i \in I} \mathcal{M}_i \models t^{\text{cl}} = s^{\text{cl}}, \end{aligned}$$

where t, s are such that

$$FM =_{\beta\eta} t^{\text{cl}}, FN =_{\beta\eta} s^{\text{cl}}, \tag{1}$$

as by Lemma 2A.18 every closed term of type \top^2 is $\beta\eta$ -convertible to some u^{cl} with $u \in \mathcal{T}$. Now the chain of arguments continues as follows

$$\begin{aligned} &\Rightarrow t \equiv s, && \text{by the embedding property,} \\ &\Rightarrow FM =_{\beta\eta} FN, && \text{by (1),} \\ &\Rightarrow M =_{\beta\eta} N, && \text{by reducibility. ■} \end{aligned}$$

3D.32. COROLLARY. (i) [[Friedman \[1975\]](#)] $\{\mathcal{M}_\mathbb{N}\}$ is complete.

(ii) [[Plotkin \[1980\]](#)] $\{\mathcal{M}_n \mid n \in \mathbb{N}\}$ is complete.

(iii) $\{\mathcal{M}_{\mathbb{N}_\perp}\}$ is complete.

(iv) $\{\mathcal{M}_D \mid D \text{ a finite cpo}\}$, 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 .

3D.33. PROPOSITION (Finite completeness theorem, Statman [1982]). *For every type A in \mathbb{T}^0 and every $M \in \Lambda^\circ(A)$ there is a number $n = n_M$ such that for all $N \in \Lambda^\circ(A)$*

$$\mathcal{M}_n \models M = N \Leftrightarrow M =_{\beta\eta} N.$$

PROOF. By the reduction Theorem 3D.8 it suffices to show this for $A = \top^2$. Let M a closed term of type \top^2 be given. Each closed term N of type \top^2 has as long $\beta\eta$ -nf

$$N = \lambda b:1_2 \lambda c:0.s_N,$$

where $s_N \in \mathcal{T}$. Let $\mathbf{p} : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$ be an injective pairing on the integers such that $\mathbf{p}(k_1, k_2) > k_i$. Take

$$n_M = (\llbracket M \rrbracket^{\mathcal{M}_\omega} \mathbf{p} 0) + 1.$$

Define $\mathbf{p}' : X_{n+1}^2 \rightarrow X_{n+1}$, where $X_{n+1} = \{0, \dots, n+1\}$, by

$$\begin{aligned} \mathbf{p}'(k_1, k_2) &= \mathbf{p}(k_1, k_2), && \text{if } k_1, k_2 \leq n \text{ \& } \mathbf{p}(k_1, k_2) \leq n; \\ &= n+1 && \text{else.} \end{aligned}$$

Suppose $\mathcal{M}_n \models M = N$. Then $\llbracket M \rrbracket^{\mathcal{M}_n} \mathbf{p}' 0 = \llbracket N \rrbracket^{\mathcal{M}_n} \mathbf{p}' 0$. By the choice of n it follows that $\llbracket M \rrbracket^{\mathcal{M}_n} \mathbf{p} 0 = \llbracket N \rrbracket^{\mathcal{M}_n} \mathbf{p} 0$ and hence $s_M = s_N$. Therefore $M =_{\beta\eta} N$. ■

3E. The five canonical term-models

We work with $\lambda_\rightarrow^{\text{Ch}}$ based on \mathbb{T}^0 . We often will use for a term like $\lambda x^A.x^A$ its de Bruijn notation $\lambda x:A.x$, since it takes less space. Another advantage of this notation is that we can write $\lambda f:1 x:0.f^2x \equiv \lambda f:1 x:0.f(fx)$, which is $\lambda f^1 x^0.f^1(f^1x^0)$ in Church's notation.

The open terms of $\lambda_\rightarrow^{\text{Ch}}$ form an extensional model, the term-model $\mathcal{M}_{\Lambda_\rightarrow}$. 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 0. 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 $\Lambda_\rightarrow^\circ[\mathcal{D}]$, 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 0 coincides with $=_{\beta\eta}$, can be obtained in this way.

3E.1. DEFINITION. Let \mathcal{D} be a set of constants, each with its own type in \mathbb{T}^0 . Then \mathcal{D} is *sufficient* if for every $A \in \mathbb{T}^0$ there is a closed term $M \in \Lambda_\rightarrow^\circ[\mathcal{D}](A)$.

For example $\{x^0\}, \{F^2, f^1\}$ are sufficient. But $\{f^1\}, \{\Psi^3, f^1\}$ are not. Note that

$$\mathcal{D} \text{ is sufficient} \Leftrightarrow \Lambda_\rightarrow^\circ[\mathcal{D}](0) \neq \emptyset.$$

3E.2. DEFINITION. Let $M, N \in \Lambda_\rightarrow^\circ[\mathcal{D}](A)$ with $A = A_1 \rightarrow \dots \rightarrow A_a \rightarrow 0$.

(i) M is \mathcal{D} -extensionally equivalent with N , notation $M \approx_{\mathcal{D}}^{\text{ext}} N$, iff

$$\forall t_1 \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A_1) \cdots t_a \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A_a). M\vec{t} =_{\beta\eta} N\vec{t}.$$

[If $a = 0$, then $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](0)$; in this case $M \approx_{\mathcal{D}}^{\text{ext}} N \Leftrightarrow M =_{\beta\eta} N$.]

(ii) M is \mathcal{D} -observationally equivalent with N , notation $M \approx_{\mathcal{D}}^{\text{obs}} N$, iff

$$\forall F \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A \rightarrow 0) FM =_{\beta\eta} FN. \blacksquare$$

3E.3. REMARK. (i) Let $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A)$ and $F \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A \rightarrow B)$. Then

$$M \approx_{\mathcal{D}}^{\text{obs}} N \Rightarrow FM \approx_{\mathcal{D}}^{\text{obs}} FN.$$

(ii) Let $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A \rightarrow B)$. Then

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Leftrightarrow \forall Z \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A). MZ \approx_{\mathcal{D}}^{\text{ext}} NZ.$$

(iii) Let $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A)$. Then

$$M \approx_{\mathcal{D}}^{\text{obs}} N \Rightarrow M \approx_{\mathcal{D}}^{\text{ext}} N,$$

by taking $F \equiv \lambda m.m\vec{t}$.

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 $\beta\eta$ -convertibility: M and N may act differently on different variables, even if they act the same on all those closed terms. The relation $\approx_{\mathcal{D}}^{\text{ext}}$ is related to what is called in the untyped calculus the ω -rule, see B[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 0. But one can do ‘experiments’ with M and N , the outcome of which is observational, i.e. of type 0, by putting these terms in a context $C[-]$ resulting in two terms of type 0. For closed terms it amounts to the same to consider just FM and FN for all $F \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A \rightarrow 0)$.

The main result in this section is Theorem 3E.34, it states that for all \mathcal{D} and for all $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$ of the same type one has

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Leftrightarrow M \approx_{\mathcal{D}}^{\text{obs}} N. \quad (1)$$

After this has been proved, we can write simply $M \approx_{\mathcal{D}} N$. The equivalence (1) will first be established in Corollary 3E.18 for some ‘canonical’ sets of constants. The general result will follow, Theorem 3E.34, using the theory of type reducibility.

The following obvious result is often used.

3E.4. REMARK. Let $M \equiv M[\vec{d}], N \equiv N[\vec{d}] \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A)$, where all occurrences of \vec{d} are displayed. Then

$$M[\vec{d}] =_{\beta\eta} N[\vec{d}] \Leftrightarrow \lambda \vec{x}. M[\vec{x}] =_{\beta\eta} \lambda \vec{x}. N[\vec{x}].$$

The reason is that new constants and fresh variables are used in the same way and that the latter can be bound.

3E.5. PROPOSITION. Suppose that $\approx_{\mathcal{D}}^{\text{ext}}$ is logical on $\Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$. Then

$$\forall M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}] [M \approx_{\mathcal{D}}^{\text{ext}} N \Leftrightarrow M \approx_{\mathcal{D}}^{\text{obs}} N].$$

PROOF. By Remark 3E.3(iii) we only have to show (\Rightarrow). So assume $M \approx_{\mathcal{D}}^{\text{ext}} N$. Let $F \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A \rightarrow 0)$. Then trivially

$$\begin{aligned} & F \approx_{\mathcal{D}}^{\text{ext}} F. \\ \Rightarrow & FM \approx_{\mathcal{D}}^{\text{ext}} FN, \quad \text{as by assumption } \approx_{\mathcal{D}}^{\text{ext}} \text{ is logical,} \\ \Rightarrow & FM =_{\beta\eta} FN, \quad \text{because the type is 0.} \end{aligned}$$

Therefore $M \approx_{\mathcal{D}}^{\text{obs}} N$. ■

The converse of Proposition 3E.5 is a good warm-up exercise. That is, if

$$\forall M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}] [M \approx_{\mathcal{D}}^{\text{ext}} N \Leftrightarrow M \approx_{\mathcal{D}}^{\text{obs}} N],$$

then $\approx_{\mathcal{D}}^{\text{ext}}$ is the logical relation on $\Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$ determined by $\beta\eta$ -equality on $\Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](0)$.

3E.6. DEFINITION. $\text{BetaEta}^{\mathcal{D}} = \{\text{BetaEta}_A^{\mathcal{D}}\}_{A \in \mathbb{T}^0}$ is the logical relation on $\Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$ determined by

$$\text{BetaEta}_0^{\mathcal{D}}(M, N) \Leftrightarrow M =_{\beta\eta} N,$$

for $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](0)$.

3E.7. LEMMA. Let $\mathbf{d} = \mathbf{d}^{A \rightarrow 0} \in \mathcal{D}$, with $A = A_1 \rightarrow \dots \rightarrow A_a \rightarrow 0$. Suppose

- (i) $\forall F, G \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A) [F \approx_{\mathcal{D}}^{\text{ext}} G \Rightarrow F =_{\beta\eta} G]$;
- (ii) $\forall t_i \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A_i) \text{BetaEta}^{\mathcal{D}}(t_i, t_i)$, $1 \leq i \leq a$.

Then $\text{BetaEta}_{A \rightarrow 0}^{\mathcal{D}}(\mathbf{d}, \mathbf{d})$.

PROOF. Write $S = \text{BetaEta}^{\mathcal{D}}$. Let \mathbf{d} be given. Then

$$\begin{aligned} S(F, G) &\Rightarrow F \vec{t} =_{\beta\eta} G \vec{t}, \quad \text{since } \forall \vec{t} \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}] S(t_i, t_i) \text{ by assumption (ii),} \\ &\Rightarrow F \approx_{\mathcal{D}}^{\text{ext}} G, \\ &\Rightarrow F =_{\beta\eta} G, \quad \text{by assumption (i),} \\ &\Rightarrow dF =_{\beta\eta} dG. \end{aligned}$$

Therefore we have by definition $S(\mathbf{d}, \mathbf{d})$. ■

3E.8. LEMMA. Let S be a syntactic n -ary logical relation on $\Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$, that is closed under $=_{\beta\eta}$. Suppose $S(\mathbf{d}, \dots, \mathbf{d})$ holds for all $\mathbf{d} \in \mathcal{D}$. Then for all $M \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$ one has

$$S(M, \dots, M).$$

PROOF. Let $\mathcal{D} = \{\mathbf{d}_1^{A_1}, \dots, \mathbf{d}_n^{A_n}\}$. M can be written as

$$M \equiv M[\vec{d}] =_{\beta\eta} (\lambda \vec{x}. M[\vec{x}]) \vec{d} \equiv M^+ \vec{d},$$

with M^+ a closed and pure term (i.e. without free variables or constants). Then

$$\begin{aligned} & S(M^+, \dots, M^+), \quad \text{by the fundamental theorem} \\ & \quad \text{for syntactic logical relations} \\ \Rightarrow & S(M^+ \vec{d}, \dots, M^+ \vec{d}), \quad \text{since } S \text{ is logical and } \forall \mathbf{d} \in \mathcal{D}. S(\vec{d}), \\ \Rightarrow & S(M, \dots, M), \quad \text{since } S \text{ is } =_{\beta\eta} \text{ closed. ■} \end{aligned}$$

3E.9. LEMMA. Suppose that for all $\mathbf{d} \in \mathcal{D}$ one has $\text{BetaEta}^{\mathcal{D}}(\mathbf{d}, \mathbf{d})$. Then $\approx_{\mathcal{D}}^{\text{ext}}$ is $\text{BetaEta}^{\mathcal{D}}$ and hence logical.

PROOF. Write $S = \text{BetaEta}^{\mathcal{D}}$. By the assumption and the fact that S is $=_{\beta\eta}$ closed (since S_0 is), Lemma 3E.8 implies that

$$S(M, M) \tag{0}$$

for all $M \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$. It now follows that S is an equivalence relation on $\Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$. Claim

$$S_A(F, G) \Leftrightarrow F \approx_{\mathcal{D}}^{\text{ext}} G,$$

for all $F, G \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A)$. This is proved by induction on the structure of A . If $A = 0$, then this follows by definition. If $A = B \rightarrow C$, then we proceed as follows.

$$\begin{aligned} (\Rightarrow) \quad S_{B \rightarrow C}(F, G) &\Rightarrow S_C(Ft, Gt), \quad \text{for all } t \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](B), \\ &\quad \text{since } t \approx_{\mathcal{D}}^{\text{ext}} t \text{ and hence, by the IH, } S_B(t, t), \\ &\Rightarrow Ft \approx_{\mathcal{D}}^{\text{ext}} Gt, \quad \text{for all } t \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}], \text{ by the IH,} \\ &\Rightarrow F \approx_{\mathcal{D}}^{\text{ext}} G, \quad \text{by definition.} \end{aligned}$$

$$\begin{aligned} (\Leftarrow) \quad F \approx_{\mathcal{D}}^{\text{ext}} G &\Rightarrow Ft \approx_{\mathcal{D}}^{\text{ext}} Gt, \quad \text{for all } t \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}], \\ &\Rightarrow S_C(Ft, Gt) \end{aligned} \tag{1}$$

by the induction hypothesis. In order to prove $S_{B \rightarrow C}(F, G)$, assume $S_B(t, s)$ towards $S_C(Ft, Gs)$. 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 (which on this type is the same as $\approx_{\mathcal{D}}^{\text{ext}}$ by the IH) that $S_C(Ft, Gs)$ indeed.

By the claim $\approx_{\mathcal{D}}^{\text{ext}}$ is S and therefore $\approx_{\mathcal{D}}^{\text{ext}}$ is logical. ■

3E.10. DEFINITION. Let $\mathcal{D} = \{c_1^{A_1}, \dots, c_k^{A_k}\}$ be a finite set of typed constants.

- (i) The *characteristic type* of \mathcal{D} , notation $\nabla(\mathcal{D})$, is $A_1 \rightarrow \dots \rightarrow A_k \rightarrow 0$.
- (ii) We say that a type $A = A_1 \rightarrow \dots \rightarrow A_a \rightarrow 0$ is *represented in \mathcal{D}* if there are distinct constants $d_1^{A_1}, \dots, d_a^{A_a} \in \mathcal{D}$.

In other words, $\nabla(\mathcal{D})$ is intuitively the type of $\lambda \vec{d}_i. d^0$, where $\mathcal{D} = \{\vec{d}_i\}$ (the order of the abstractions is immaterial, as the resulting types are all $\sim_{\beta\eta}$ equivalent). Note that $\nabla(\mathcal{D})$ is represented in \mathcal{D} .

3E.11. DEFINITION. Let \mathcal{D} be a set of constants.

- (i) If \mathcal{D} is finite, then the *class* of \mathcal{D} is the class of the type $\nabla(\mathcal{D})$, i.e. the unique i such that $\nabla(\mathcal{D}) \in \mathbb{T}_i$.
- (ii) In general the class of \mathcal{D} is

$$\max\{\text{class}(A) \mid A \text{ represented in } \mathcal{D}\}.$$

- (iii) A *characteristic type* of \mathcal{D} , notation $\nabla(\mathcal{D})$ is any A represented in \mathcal{D} such that $\text{class}(\mathcal{D}) = \text{class}(A)$. That is, $\nabla(\mathcal{D})$ is any type represented in \mathcal{D} of highest class.

It is not hard to see that for finite \mathcal{D} the two definitions of $\text{class}(\mathcal{D})$ coincide.

3E.12. REMARK. Note that it follows by Remark 3D.20 that

$$\mathcal{D}_1 \subseteq \mathcal{D}_2 \Rightarrow \text{class}(\mathcal{D}_1) \leq \text{class}(\mathcal{D}_2).$$

In order to show that for arbitrary \mathcal{D} extensional equivalence is the same as observational equivalence this will be done first for the following ‘canonical’ sets of constants.

3E.13. DEFINITION. The following sets of constants will play a crucial role in this section.

$$\begin{aligned}\mathcal{C}_{-1} &\triangleq \emptyset; \\ \mathcal{C}_0 &\triangleq \{\mathbf{c}^0\}; \\ \mathcal{C}_1 &\triangleq \{\mathbf{c}^0, \mathbf{d}^0\}; \\ \mathcal{C}_2 &\triangleq \{\mathbf{f}^1, \mathbf{c}^0\}; \\ \mathcal{C}_3 &\triangleq \{\mathbf{f}^1, \mathbf{g}^1, \mathbf{c}^0\}; \\ \mathcal{C}_4 &\triangleq \{\Phi^3, \mathbf{c}^0\}; \\ \mathcal{C}_5 &\triangleq \{\mathbf{b}^{12}, \mathbf{c}^0\}. \blacksquare\end{aligned}$$

3E.14. REMARK. The actual names of the constants is irrelevant, for example \mathcal{C}_2 and $\mathcal{C}'_2 = \{\mathbf{g}^1, \mathbf{c}^0\}$ will give rise to isomorphic term models. Therefore we may assume that a set of constants \mathcal{D} of class i is disjoint with \mathcal{C}_i .

From now on in this section \mathcal{C} ranges over the canonical sets of constants $\{\mathcal{C}_{-1}, \dots, \mathcal{C}_5\}$ and \mathcal{D} over arbitrary sets of constants.

3E.15. REMARK. Let \mathcal{C} be one of the canonical sets of constants. The characteristic types of these \mathcal{C} are as follows.

$$\begin{aligned}\nabla(\mathcal{C}_{-1}) &= 0; \\ \nabla(\mathcal{C}_0) &= 0 \rightarrow 0; \\ \nabla(\mathcal{C}_1) &= 1_2 = 0 \rightarrow 0 \rightarrow 0; \\ \nabla(\mathcal{C}_2) &= 1 \rightarrow 0 \rightarrow 0; \\ \nabla(\mathcal{C}_3) &= 1 \rightarrow 1 \rightarrow 0 \rightarrow 0; \\ \nabla(\mathcal{C}_4) &= 3 \rightarrow 0 \rightarrow 0; \\ \nabla(\mathcal{C}_5) &= 1_2 \rightarrow 0 \rightarrow 0.\end{aligned}$$

So $\nabla(\mathcal{C}_i) = C_i$, where the type C_i is as in Remark 3D.21. Also one has

$$i \leq j \Leftrightarrow \nabla(\mathcal{C}_i) \leq_{\beta\eta} \nabla(\mathcal{C}_j),$$

as follows from the theory of type reducibility.

We will need the following combinatorial lemma about $\approx_{\mathcal{C}_4}^{\text{ext}}$.

3E.16. LEMMA. *For every $F, G \in \Lambda[\mathcal{C}_4](2)$ one has*

$$F \approx_{\mathcal{C}_4}^{\text{ext}} G \Rightarrow F =_{\beta\eta} G.$$

PROOF. We must show

$$[\forall h \in \Lambda[\mathcal{C}_4](1). Fh =_{\beta\eta} Gh] \Rightarrow F =_{\beta\eta} G. \quad (1)$$

In order to do this, a classification has to be given for the elements of $\Lambda[\mathcal{C}_4](2)$. Define for $A \in \mathbb{T}^0$ and context Δ

$$A_\Delta = \{M \in \Lambda[\mathcal{C}_4](A) \mid \Delta \vdash M : A \ \& \ M \text{ in } \beta\eta\text{-nf}\}.$$

It is easy to show that 0_Δ and 2_Δ are generated by the following ‘two-level’ grammar, see van Wijngaarden [1981].

$$\begin{aligned}2_\Delta &::= \lambda f:1.0_{\Delta,f:1} \\ 0_\Delta &::= \mathbf{c} \mid \Phi 2_\Delta \mid \Delta.1 0_\Delta,\end{aligned}$$

where $\Delta.A$ consists of $\{v \mid v^A \in \Delta\}$.

It follows that a typical element of 2_\emptyset is

$$\lambda f_1:1.\Phi(\lambda f_2:1.f_1(f_2(\Phi(\lambda f_3:1.f_3(f_2(f_1(f_3 \mathbf{c}))))))).$$

Hence a general element can be represented by a list of words

$$\langle w_1, \dots, w_n \rangle,$$

with $w_i \in \Sigma_i^*$ and $\Sigma_i = \{f_1, \dots, f_i\}$, the representation of the typical element above being $\langle \epsilon, f_1 f_2, f_3 f_2 f_1 f_3 \rangle$. The inhabitation machines in Section 1C were inspired by this example.

Let $h_m = \lambda z:0.\Phi(\lambda g:1.g^m(z))$; then $h_m \in 1_\emptyset$. We claim that

$$\forall F, G \in \Lambda^\phi_{\rightarrow}[\mathcal{C}_4](2) \exists m \in \mathbb{N}. [F h_m =_{\beta\eta} G h_m \Rightarrow F =_{\beta\eta} G].$$

For a given $F \in \Lambda[\mathcal{C}_4](2)$ and $m \in \mathbb{N}$ one can find a representation of the $\beta\eta$ -nf of $F h_m$ from the representation of the $\beta\eta$ -nf $F^{\text{nf}} \in 2_\emptyset$ of F . It will turn out that if m is large enough, then F^{nf} can be determined ('read back') from the $\beta\eta$ -nf of $F h_m$.

In order to see this, let F^{nf} be represented by the list of words $\langle w_1, \dots, w_n \rangle$, as above. The occurrences of f_1 can be made explicit and we write

$$w_i = w_{i0} f_1 w_{i1} f_1 w_{i2} \cdots f_1 w_{ik_i}.$$

Some of the w_{ij} will be empty (in any case the w_{1j}) and $w_{ij} \in \Sigma_i^{-*}$ with $\Sigma_i^- = \{f_2, \dots, f_i\}$. Then F^{nf} can be written as (using for application—contrary to the usual convention—association to the right)

$$\begin{aligned} F^{\text{nf}} &\equiv \lambda f_1.w_{10} f_1 w_{11} \cdots f_1 w_{1k_1} \\ &\quad \Phi(\lambda f_2.w_{20} f_1 w_{21} \cdots f_1 w_{2k_2} \\ &\quad \cdots \\ &\quad \Phi(\lambda f_n.w_{n0} f_1 w_{n1} \cdots f_1 w_{nk_n} \\ &\quad \mathbf{c})..). \end{aligned}$$

Now we have

$$\begin{aligned}
(Fh_m)^{\text{nf}} &\equiv w_{10} \\
&\Phi(\lambda g.g^m w_{11}) \\
&\dots \\
&\Phi(\lambda g.g^m w_{1k_1}) \\
&\Phi(\lambda f_2.w_{20}) \\
&\Phi(\lambda g.g^m w_{21}) \\
&\dots \\
&\Phi(\lambda g.g^m w_{2k_2}) \\
&\Phi(\lambda f_3.w_{30}) \\
&\Phi(\lambda g.g^m w_{31}) \\
&\dots \\
&\Phi(\lambda g.g^m w_{3k_3}) \\
&\dots \\
&\dots \\
&\Phi(\lambda f_n.w_{n0}) \\
&\Phi(\lambda g.g^m w_{n1}) \\
&\dots \\
&\Phi(\lambda g.g^m w_{nk_n}) \\
&(\mathbf{c}(\dots))(\dots)(\dots)(\dots)(\dots)(\dots).
\end{aligned}$$

So if $m > \max_{ij}\{\text{length}(w_{ij})\}$ we can read back the w_{ij} and hence F^{nf} from $(Fh_m)^{\text{nf}}$. Therefore using an m large enough (1) can be shown as follows:

$$\begin{aligned}
\forall h \in \Lambda[\mathcal{C}_4](1).Fh =_{\beta\eta} Gh &\Rightarrow Fh_m =_{\beta\eta} Gh_m \\
&\Rightarrow (Fh_m)^{\text{nf}} \equiv (Gh_m)^{\text{nf}} \\
&\Rightarrow F^{\text{nf}} \equiv G^{\text{nf}} \\
&\Rightarrow F =_{\beta\eta} F^{\text{nf}} \equiv G^{\text{nf}} =_{\beta\eta} G. \blacksquare
\end{aligned}$$

3E.17. PROPOSITION. *For all $i \in \{-1, 0, 1, 2, 3, 4, 5\}$ the relations $\approx_{\mathcal{C}_i}^{\text{ext}}$ are logical.*

PROOF. Write $\mathcal{C} = \mathcal{C}_i$. For $i = -1$ the relation $\approx_{\mathcal{C}}^{\text{ext}}$ is universally valid by the empty implication, as there are never terms \vec{t} making $M\vec{t}, N\vec{t}$ of type 0. Therefore, the result is trivially valid.

Let S be the logical relation on $\Lambda_{\rightarrow}^{\emptyset}[\mathcal{C}]$ determined by $=_{\beta\eta}$ on the ground level $\Lambda_{\rightarrow}^{\emptyset}[\mathcal{C}](0)$. By Lemma 3E.9 we have to check $S(\mathbf{c}, \mathbf{c})$ for all constants \mathbf{c} in \mathcal{C}_i . For $i \neq 4$ this is easy (trivial for constants of type 0 and almost trivial for the ones of type 1 and $1_2 = (0^2 \rightarrow 0)$; in fact for all terms $h \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{C}]$ of these types one has $S(h, h)$).

For $i = 4$ we reason as follows. Write $S = \text{BetaEta}^{\mathcal{C}_4}$. It suffices by Lemma 3E.9 to show that $S(\Phi^3, \Phi^3)$. By Lemma 3E.7 it suffices to show

$$F \approx_{\mathcal{C}_4} G \Rightarrow F =_{\beta\eta} G$$

for all $F, G \in \Lambda_{\rightarrow}^{\emptyset}[C_4](2)$, which has been verified in Lemma 3E.16, and $S(t, t)$ for all $t \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{C}_4](1)$, which follows directly from the definition of S , since $=_{\beta\eta}$ is a congruence:

$$\forall M, N \in \Lambda_{\rightarrow}^{\emptyset}[0]. [M =_{\beta\eta} N \Rightarrow tM =_{\beta\eta} tN]. \blacksquare$$

3E.18. COROLLARY. *Let \mathcal{C} be one of the canonical classes of constants. Then*

$$\forall M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{C}]. [M \approx_{\mathcal{C}}^{\text{obs}} N \Leftrightarrow M \approx_{\mathcal{C}}^{\text{ext}} N].$$

PROOF. By the Proposition and Proposition 3E.5. \blacksquare

Arbitrary finite sets of constants \mathcal{D}

Now we pay attention to arbitrary finite sets of constants \mathcal{D} .

3E.19. REMARK. Before starting the proof of the next results it is good to realize the following. For $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D} \cup \{\mathbf{c}^A\}] \setminus \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$ it makes sense to state $M \approx_{\mathcal{D}}^{\text{ext}} N$, but in general we do not have

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Rightarrow M \approx_{\mathcal{D} \cup \{\mathbf{c}^A\}}^{\text{ext}} N. \quad (+)$$

Indeed, taking $\mathcal{D} = \{\mathbf{d}^0\}$ this is the case for $M \equiv \lambda x^0 b^{12}.b\mathbf{c}^0 x$, $N \equiv \lambda x^0 b^{12}.b\mathbf{c}^0 \mathbf{d}^0$. The implication (+) does hold if $\text{class}(\mathcal{D}) = \text{class}(\mathcal{D} \cup \{\mathbf{c}^A\})$, as we will see later.

We first need to show the following proposition.

PROPOSITION (Lemma P_i , with $i \in \{3, 4, 5\}$). Let \mathcal{D} be a finite set of constants of class $i > 2$ and $\mathcal{C} = \mathcal{C}_i$. Then for $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$ of the same type we have

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Rightarrow M \approx_{\mathcal{D} \cup \mathcal{C}}^{\text{ext}} N.$$

We will assume that $\mathcal{D} \cap \mathcal{C} = \emptyset$, see Remark 3E.14. This assumption is not yet essential since if \mathcal{D}, \mathcal{C} overlap, then the statement $M \approx_{\mathcal{D} \cup \mathcal{C}}^{\text{ext}} N$ is easier to prove. The proof occupies 3E.20–3E.27.

NOTATION. Let $A = A_1 \rightarrow \dots \rightarrow A_a \rightarrow 0$ and $d \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](0)$. Define $\mathsf{K}^A d \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A)$ by

$$\mathsf{K}^A d \triangleq (\lambda x_1 : A_1 \dots \lambda x_a : A_a. d).$$

3E.20. LEMMA. *Let \mathcal{D} be a finite set of constants of class $i > 1$. Then for all $A \in \mathbb{T}^0$ the set $\Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A)$ contains infinitely many distinct lnf-s.*

PROOF. Because $i > -1$ there is a term in $\Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](\nabla(\mathcal{D}))$. Hence \mathcal{D} is sufficient and there exists a $d^0 \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](0)$ in lnf. Since $i > 1$ there is a constant $\mathbf{d}^B \in \mathcal{D}$ with $B = B_1 \rightarrow \dots \rightarrow B_b \rightarrow 0$, and $b > 0$. Define the sequence of elements in $\Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](0)$:

$$\begin{aligned} d_0 &\triangleq d^0; \\ d_{k+1} &\triangleq \mathbf{d}^B (\mathsf{K}^{B_1} d_k) \dots (\mathsf{K}^{B_b} d_k). \end{aligned}$$

As d_k is a lnf and $|d_{k+1}| > |d_k|$, the $\{\mathsf{K}^A d_0, \mathsf{K}^A d_1, \dots\}$ are distinct lnf-s in $\Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A)$. \blacksquare

3E.21. REMARK. We want to show that for $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$ of the same type one has

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Rightarrow M \approx_{\mathcal{D} \cup \{\mathbf{c}^0\}}^{\text{ext}} N. \quad (0)$$

The strategy will be to show that for all $P, Q \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D} \cup \{\mathbf{c}^0\}](0)$ in lnf one can find a term $T_c \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A)$ such that

$$P \not\equiv Q \Rightarrow P[\mathbf{c}^0 := T_c] \not\equiv Q[\mathbf{c}^0 := T_c]. \quad (1)$$

Then (0) can be proved via the contrapositive

$$\begin{aligned}
 M \not\approx_{\mathcal{D} \cup \{\mathbf{c}^0\}}^{\text{ext}} N &\Rightarrow M\vec{t} \neq_{\beta\eta} N\vec{t} (: 0), & \text{for some } \vec{t} \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D} \cup \{\mathbf{c}^0\}] \\
 &\Rightarrow P \not\equiv Q, & \text{by taking lnf-s,} \\
 &\Rightarrow P[\mathbf{c} := T_c] \not\equiv Q[\mathbf{c} := T_c], & \text{by (1),} \\
 &\Rightarrow M\vec{s} \neq_{\beta\eta} N\vec{s}, & \text{with } \vec{s} = \vec{t}\mathbf{c} := T_c, \\
 &\Rightarrow M \not\approx_{\mathcal{D}} N.
 \end{aligned}$$

3E.22. LEMMA. Let \mathcal{D} be of class $i \geq 1$ and let \mathbf{c}^0 be an arbitrary constant of type 0. Then for $M, N \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D}]$ of the same type

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Rightarrow M \approx_{\mathcal{D} \cup \{\mathbf{c}^0\}}^{\text{ext}} N.$$

PROOF. Using Remark 3E.21 let $P, Q \in \Lambda_{\rightarrow}[\mathcal{D} \cup \{\mathbf{c}^0\}](0)$ and assume $P \not\equiv Q$.

Case $i > 1$. Consider the difference in the Böhm trees of P, Q at a node with smallest length. If at that node in neither trees there is a \mathbf{c} , then we can take $T_c = d^0$ for any $d^0 \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D}]$. If at that node in exactly one of the trees there is \mathbf{c} and in the other a different $s \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D} \cup \{\mathbf{c}^0\}]$, then we must take d^0 sufficiently large, which is possible by Lemma 3E.20, in order to preserve the difference; these are all cases.

Case $i = 1$. Then $\mathcal{D} = \{\mathbf{d}_1^0, \dots, \mathbf{d}_k^0\}$, with $k \geq 2$. So one has $P, Q \in \{\mathbf{d}_1^0, \dots, \mathbf{d}_k^0, \mathbf{c}^0\}$. If $\mathbf{c} \notin \{P, Q\}$, then take any $T_c = \mathbf{d}_i$. Otherwise one has $P \equiv \mathbf{c}$, $Q \equiv \mathbf{d}_i$, say. Then take $T_c \equiv \mathbf{d}_j$, for some $j \neq i$. ■

3E.23. REMARK. Let $\mathcal{D} = \{\mathbf{d}^0\}$ be of class $i = 0$. Then Lemma 3E.22 is false. Take for example $\lambda x^0.x \approx_{\mathcal{D}}^{\text{ext}} \lambda x^0.\mathbf{d}$, as \mathbf{d} is the only element of $\Lambda_{\rightarrow}^{\phi}[\mathcal{D}](0)$. But $\lambda x^0.x \not\approx_{\{\mathbf{d}^0, \mathbf{c}^0\}}^{\text{ext}} \lambda x^0.\mathbf{d}$.

3E.24. LEMMA (P_5). Let \mathcal{D} be a finite set of class $i = 5$ and $\mathcal{C} = \mathcal{C}_5 = \{\mathbf{c}^0, \mathbf{b}^{12}\}$. Then for $M, N \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D}]$ of the same type one has

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Rightarrow M \approx_{\mathcal{D} \cup \mathcal{C}}^{\text{ext}} N.$$

PROOF. By Lemma 3E.22 it suffices to show for $M, N \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D}]$ of the same type

$$M \approx_{\mathcal{D} \cup \{\mathbf{c}^0\}}^{\text{ext}} N \Rightarrow M \approx_{\mathcal{D} \cup \{\mathbf{c}^0, \mathbf{b}^{12}\}}^{\text{ext}} N.$$

By Remark 3E.21 it suffices to find for distinct lnf-s $P, Q \in \Lambda_{\rightarrow}[\mathcal{D} \cup \{\mathbf{c}^0, \mathbf{b}^{12}\}](0)$ a term $T_b \in \Lambda_{\rightarrow}[\mathcal{D} \cup \{\mathbf{c}^0\}](1_2)$ such that

$$P[\mathbf{b} := T_b] \not\equiv Q[\mathbf{b} := T_b]. \quad (1).$$

We look for such a term that is in any case injective: for all $R, R', S, S' \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D} \cup \{\mathbf{c}^0\}](0)$

$$T_b R S =_{\beta\eta} T_b R' S' \Rightarrow R =_{\beta\eta} R' \& S =_{\beta\eta} S'.$$

Now let $\mathcal{D} = \{\mathbf{d}_1 : A_1, \dots, \mathbf{d}_b : A_b\}$. Since \mathcal{D} is of class 5 the type $\nabla(\mathcal{D}) = A_1 \rightarrow \dots \rightarrow A_b \rightarrow 0$ is inhabited and large. Let $T \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D}](0)$.

Remember that a type $A = A_1 \rightarrow \dots \rightarrow A_b \rightarrow 0$ is large if it has a negative occurrence of a subtype with more than one component. So one has one of the following two cases.

Case 1. For some $i \leq b$ one has $A_i = B_1 \rightarrow \dots \rightarrow B_b \rightarrow 0$ with $b \geq 2$.

Case 2. Each $A_i = A'_i \rightarrow 0$ and some A'_i is large, $1 \leq i \leq b$.

Now we define for a type A that is large the term $T_A \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](1_2)$ by induction on the structure of A , following the mentioned cases.

$$\begin{aligned} T_A &= \lambda x^0 y^0 . \mathbf{d}_i(\mathsf{K}^{B_1} x)(\mathsf{K}^{B_2} y)(\mathsf{K}^{B_3} T) \cdots (\mathsf{K}^{B_b} T), && \text{if } i \leq b \text{ is the least such that} \\ &&& A_i = B_1 \rightarrow \cdots \rightarrow B_b \rightarrow 0 \text{ with } b \geq 2, \\ &= \lambda x^0 y^0 . \mathbf{d}_i(\mathsf{K}^{A'_i}(T_{A'_i} xy)), && \text{if each } A_j = A'_j \rightarrow 0 \text{ and } i \leq a \text{ is} \\ &&& \text{the least such that } A'_i \text{ is large.} \end{aligned}$$

By induction on the structure of the large type A one easily shows using the Church-Rosser theorem that T_A is injective in the sense above.

Let $A = \nabla(\mathcal{D})$, which is large. We cannot yet take $T_b \equiv T_A$. For example the difference $\mathbf{bcc} \neq_{\beta\eta} T_A \mathbf{ccc}$ gets lost. By Lemma 3E.20 there exists a $T^+ \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](0)$ with

$$|T^+| > \max\{|P|, |Q|\}.$$

Define

$$T_b = (\lambda x y. T_A(T_A x T^+) y) \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](1_2).$$

Then also this T_b is injective. The T^+ acts as a ‘tag’ to remember where T_b is inserted. Therefore this T_b satisfies (1). ■

3E.25. LEMMA (P_4). *Let \mathcal{D} be a finite set of class $i = 4$ and $\mathcal{C} = \mathcal{C}_4 = \{\mathbf{c}^0, \Phi^3\}$. Then for $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$ of the same type one has*

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Rightarrow M \approx_{\mathcal{D} \cup \mathcal{C}}^{\text{ext}} N.$$

PROOF. By Remark 3E.21 and Lemma 3E.22 it suffices to show that for all distinct lnf-s $P, Q \in \Lambda_{\rightarrow}[\mathcal{D} \cup \{\mathbf{c}^0, \Phi^3\}](0)$ there exists a term $T_{\Phi} \in \Lambda_{\rightarrow}[\mathcal{D} \cup \{\mathbf{c}^0\}](3)$ such that

$$P[\Phi := T_{\Phi}] \neq Q[\Phi := T_{\Phi}]. \quad (1)$$

Let $A = A_1 \rightarrow \cdots \rightarrow A_a \rightarrow 0$ be a small type of rank $k \geq 2$. Wlog we assume that $\text{rk}(A_1) = \text{rk}(A) - 1$. As A is small one has $A_1 = B \rightarrow 0$, with B small of rank $k - 2$.

Let H be a term variable of type 2. We construct a term

$$M_A \equiv M_A[H] \in \Lambda_{\rightarrow}^{\{H:2\}}(A).$$

The term M_A is defined directly if $k \in \{2, 3\}$; else via M_B , with $\text{rk}(M_B) = \text{rk}(M_A) - 2$.

$$\begin{aligned} M_A &\triangleq \lambda x_1 : A_1 \cdots \lambda x_a : A_a . H x_1, && \text{if } \text{rk}(A) = 2, \\ &\triangleq \lambda x_1 : A_1 \cdots \lambda x_a : A_a . H(\lambda z : 0 . x_1(\mathsf{K}^B z)), && \text{if } \text{rk}(A) = 3, \\ &\triangleq \lambda x_1 : A_1 \cdots \lambda x_a : A_a . x_1 M_B, && \text{if } \text{rk}(A) \geq 4. \end{aligned}$$

Let $A = \nabla(\mathcal{D})$ which is small and has rank $k \geq 4$. Then wlog $A_1 = B \rightarrow 0$ has rank ≥ 3 . Then $B = B_1 \rightarrow \cdots \rightarrow B_b \rightarrow 0$ has rank ≥ 2 . Let

$$T = (\lambda H : 2 . \mathbf{d}_1^{A_1}(M_B[H])) \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](3).$$

Although T is injective, we cannot use it to replace Φ^3 , as the difference in (1) may get lost in translation. Again we need a ‘tag’ to keep the difference between P, Q . Let $n > \max\{|P|, |Q|\}$. Let B_i be the ‘first’ with $\text{rk}(B_i) = k - 3$. As B_i is small, we have $B_i = C_i \rightarrow 0$. We modify the term T :

$$T_{\Phi} \triangleq (\lambda H : 2 . \mathbf{d}_1^{A_1}(\lambda y_1 : B_1 \cdots \lambda y_b : B_b . (y_i \circ \mathsf{K}^{C_i})^n(M_B[H] \vec{y}))) \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](3).$$

This term satisfies (1). ■

3E.26. LEMMA (P_3). Let \mathcal{D} be a finite set of class $i = 3$ and $\mathcal{C} = \mathcal{C}_3 = \{\mathbf{c}^0, \mathbf{f}^1, \mathbf{g}^1\}$. Then for $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$ of the same type one has

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Rightarrow M \approx_{\mathcal{D} \cup \mathcal{C}}^{\text{ext}} N.$$

PROOF. Again it suffices that for all distinct lnf-s $P, Q \in \Lambda_{\rightarrow}[\mathcal{D} \cup \{\mathbf{c}^0, \mathbf{f}^1, \mathbf{g}^1\}](0)$ there exist terms $T_f, T_g \in \Lambda_{\rightarrow}[\mathcal{D} \cup \{\mathbf{c}^0\}](1)$ such that

$$P[\mathbf{f}, \mathbf{g} := T_f, T_g] \not\equiv Q[\mathbf{f}, \mathbf{g} := T_f, T_g]. \quad (1)$$

Writing $\mathcal{D} = \{\mathbf{d}_1:A_1, \dots, \mathbf{d}_a:A_a\}$, for all $1 \leq i \leq a$ one has $A_i = 0$ or $A_i = B_i \rightarrow 0$ with $\text{rk}(B_i) \leq 1$, since $\nabla(\mathcal{D}) \in \mathbb{T}_3$. This implies that all constants in \mathcal{D} can have at most one argument. Moreover there are at least two constants, say w.l.o.g. $\mathbf{d}_1, \mathbf{d}_2$, with types $B_1 \rightarrow 0, B_2 \rightarrow 0$, respectively, that is having one argument. As \mathcal{D} is sufficient there is a $d \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](0)$. Define

$$\begin{aligned} T_1 &\triangleq \lambda x:0. \mathbf{d}_1(\mathsf{K}^{B_1}x) && \text{in } \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](1), \\ T_2 &\triangleq \lambda x:0. \mathbf{d}_2(\mathsf{K}^{B_2}x) && \text{in } \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](1). \end{aligned}$$

As P, Q are different lnf-s, we have

$$\begin{aligned} P &\equiv P_1(\lambda \vec{x}_1.P_2(\lambda \vec{x}_2. \dots P_p(\lambda \vec{x}_p.X) ..)), \\ Q &\equiv Q_1(\lambda \vec{y}_1.Q_2(\lambda \vec{y}_2. \dots Q_q(\lambda \vec{y}_q.Y) ..)), \end{aligned}$$

where the $P_i, Q_j \in (\mathcal{D} \cup \mathcal{C}_3)$, the \vec{x}_i, \vec{y}_j are possibly empty strings of variables of type 0, and X, Y are variables or constants of type 0. Let (U, V) be the first pair of symbols among the (P_i, Q_i) that are different. Distinguishing cases we define T_f, T_g such that (1). As a shorthand for the choices we write (m, n) , $m, n \in \{1, 2\}$, for the choice $T_f = T_m, T_g = T_n$.

Case 1. One of U, V , say U , is a variable or in $\mathcal{D}/\{\mathbf{d}_1, \mathbf{d}_2\}$. This U will not be changed by the substitution. If V is changed, after reducing we get $U \not\equiv \mathbf{d}_i$. Otherwise nothing happens with U, V and the difference is preserved. Therefore we can take any pair (m, n) .

Case 2. One of U, V is \mathbf{d}_i .

Subcase 2.1. The other is in $\{\mathbf{f}, \mathbf{g}\}$. Then take (j, j) , where $j = 3 - i$.

Subcase 2.2. The other one is \mathbf{d}_{3-j} . Then neither is replaced; take any pair.

Case 3. $\{U, V\} = \{\mathbf{f}, \mathbf{g}\}$. Then both are replaced and we can take $(1, 2)$.

After deciphering what is meant the verification that the difference is kept is trivial. ■

3E.27. PROPOSITION. Let \mathcal{D} be a finite set of class $i > 2$ and let $\mathcal{C} = \mathcal{C}_i$. Then for all $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$ of the same type one has

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Leftrightarrow M \approx_{\mathcal{D} \cup \mathcal{C}}^{\text{ext}} N.$$

PROOF. (\Rightarrow) By Lemmas 3E.24, 3E.25, and 3E.26. (\Leftarrow) Trivial. ■

3E.28. REMARK. (i) Proposition 3E.27 fails for $i = 0$ or $i = 2$. For $i = 0$, take $\mathcal{D} = \{\mathbf{d}^0\}$, $\mathcal{C} = \mathcal{C}_0 = \{\mathbf{c}^0\}$. Then for $P \equiv \mathsf{K}d, Q \equiv \mathsf{I}$ one has $P\mathbf{c} =_{\beta\eta} d \neq_{\beta\eta} \mathbf{c} =_{\beta\eta} Q\mathbf{c}$. But the only $u[\mathbf{d}] \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](0)$ is \mathbf{d} , loosing the difference: $Pd =_{\beta\eta} d =_{\beta\eta} Qd$. For $i = 2$, take $\mathcal{D} = \{\mathbf{g}:1, \mathbf{d}:0\}$, $\mathcal{C} = \mathcal{C}_2 = \{\mathbf{f}:1, \mathbf{c}:0\}$. Then for $P \equiv \lambda h:1.h(h(\mathbf{g}\mathbf{d}))$, $Q \equiv \lambda h:1.h(\mathbf{g}(\mathbf{h}\mathbf{d}))$ one has $P\mathbf{f} \neq_{\beta\eta} Q\mathbf{f}$, but the only $u[\mathbf{g}, \mathbf{d}] \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](0)$ are $\lambda x.\mathbf{g}^n x$ and $\lambda x.\mathbf{g}^n \mathbf{d}$, yielding $Pu =_{\beta\eta} \mathbf{g}^{2n+1} \mathbf{d} = Qu$, respectively $Pu =_{\beta\eta} \mathbf{g}^n \mathbf{d} =_{\beta\eta} Qu$.

(ii) Proposition 3E.27 clearly also holds for class $i = 1$.

3E.29. LEMMA. For $A = A_1 \rightarrow \dots \rightarrow A_a \rightarrow 0$. write $\mathcal{D}_A = \{\mathbf{c}_1^{A_1}, \dots, \mathbf{c}_a^{A_a}\}$. Let $M, N \in \Lambda_{\rightarrow}^{\emptyset}$ be pure closed terms of the same type.

(i) Suppose $A \leq_{h^+} B$. Then

$$M \approx_{\mathcal{D}_B}^{\text{ext}} N \Rightarrow M \approx_{\mathcal{D}_A}^{\text{ext}} N.$$

(ii) Suppose $A \sim_{h^+} B$. Then

$$M \approx_{\mathcal{D}_A}^{\text{ext}} N \Leftrightarrow M \approx_{\mathcal{D}_B}^{\text{ext}} N.$$

PROOF. (i) We show the contrapositive.

$$\begin{aligned} M \not\approx_{\mathcal{D}_A}^{\text{ext}} N &\Rightarrow \exists \vec{t} \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}_A]. M \vec{t}[\mathbf{a}_1, \dots, \mathbf{a}_a] \neq_{\beta\eta} N \vec{t}[\mathbf{a}_1, \dots, \mathbf{a}_a] (: 0) \\ &\Rightarrow \exists \vec{t} \lambda \vec{a}. M \vec{t}[\vec{a}] \neq_{\beta\eta} \lambda \vec{a}. N \vec{t}[\vec{a}] (: A), \text{ by Remark 3E.4,} \\ &\Rightarrow \exists \vec{t} \lambda \vec{b}. (\lambda \vec{a}. M \vec{t}[\vec{a}]) \vec{R}[\vec{b}] \neq_{\beta\eta} \lambda \vec{b}. (\lambda \vec{a}. N \vec{t}[\vec{a}]) \vec{R}[\vec{b}] (: B), \\ &\quad \text{by 3D.26(iii), as } A \leq_{h^+} B, \\ &\Rightarrow \exists \vec{t} \lambda \vec{b}. M \vec{t}[\vec{R}[\vec{b}]] \neq_{\beta\eta} \lambda \vec{b}. N \vec{t}[\vec{R}[\vec{b}]] (: B) \\ &\Rightarrow \exists \vec{t} M \vec{t}[\vec{R}[\mathbf{b}_1, \dots, \mathbf{b}_b]] \neq_{\beta\eta} N \vec{t}[\vec{R}[\mathbf{b}_1, \dots, \mathbf{b}_b]] (: 0), \text{ by Remark 3E.4,} \\ &\Rightarrow M \not\approx_{\mathcal{D}_B}^{\text{ext}} N. \end{aligned}$$

(ii) By (i). ■

3E.30. PROPOSITION. Let $\mathcal{D} = \{\mathbf{d}_1^{B_1}, \dots, \mathbf{d}_k^{B_k}\}$ be of class $i > 2$ and $\mathcal{C} = \mathcal{C}_i$, with $\mathcal{D} \cap \mathcal{C} = \emptyset$. Let $A \in \mathbb{T}^0$. Then we have the following.

(i) For $P[\vec{d}], Q[\vec{d}] \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A)$, such that $\lambda \vec{x}. P[\vec{x}], \lambda \vec{x}. Q[\vec{x}] \in \Lambda_{\rightarrow}^{\emptyset}(B_1 \rightarrow \dots \rightarrow B_k \rightarrow 0)$ the following are equivalent.

- (1) $P[\vec{d}] \approx_{\mathcal{D}}^{\text{ext}} Q[\vec{d}]$.
- (2) $\lambda \vec{x}. P[\vec{x}] \approx_{\mathcal{C}} \lambda \vec{x}. Q[\vec{x}]$.
- (3) $\lambda \vec{x}. P[\vec{x}] \approx_{\mathcal{D}}^{\text{ext}} \lambda \vec{x}. Q[\vec{x}]$.

(ii) In particular, for pure closed terms $P, Q \in \Lambda_{\rightarrow}^{\emptyset}(A)$ one has

$$P \approx_{\mathcal{D}}^{\text{ext}} Q \Leftrightarrow P \approx_{\mathcal{C}} Q.$$

PROOF. (i) We show (1) \Rightarrow (2) \Rightarrow (3) \Rightarrow (1).

(1) \Rightarrow (2). Assume $P[\vec{d}] \approx_{\mathcal{D}}^{\text{ext}} Q[\vec{d}]$. Then

$$\begin{aligned} &\Rightarrow P[\vec{d}] \approx_{\mathcal{D} \cup \mathcal{C}}^{\text{ext}} Q[\vec{d}], \quad \text{by Proposition 3E.27,} \\ &\Rightarrow P[\vec{d}] \approx_{\mathcal{C}}^{\text{ext}} Q[\vec{d}], \\ &\Rightarrow P[\vec{d}] \vec{t} =_{\beta\eta} Q[\vec{d}] \vec{t}, \quad \text{for all } \vec{t} \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{C}], \\ &\Rightarrow P[\vec{s}] \vec{t} =_{\beta\eta} Q[\vec{s}] \vec{t}, \quad \text{for all } \vec{t}, \vec{s} \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{C}] \text{ as } \mathcal{D} \cap \mathcal{C} = \emptyset, \\ &\Rightarrow \lambda \vec{x}. P[\vec{x}] \approx_{\mathcal{C}}^{\text{ext}} \lambda \vec{x}. Q[\vec{x}]. \end{aligned}$$

(2) \Rightarrow (3). By assumption $\nabla(\mathcal{D}) \sim_{h^+} \nabla(\mathcal{C})$. As $\mathcal{D} = \mathcal{D}_{\nabla(\mathcal{D})}$ and $\mathcal{C} = \mathcal{D}_{\nabla(\mathcal{C})}$ one has

$$\lambda \vec{x}. P[\vec{x}] \approx_{\mathcal{D}}^{\text{ext}} \lambda \vec{x}. Q[\vec{x}] \Leftrightarrow \lambda \vec{x}. P[\vec{x}] \approx_{\mathcal{C}}^{\text{ext}} \lambda \vec{x}. Q[\vec{x}],$$

by Lemma 3E.29.

(3) \Rightarrow (1). Assume $\lambda \vec{x}.P[\vec{x}] \approx_{\mathcal{D}}^{\text{ext}} \lambda \vec{x}.Q[\vec{x}]$. Then

$$\begin{aligned} \Rightarrow & \quad (\lambda \vec{x}.P(\vec{x})\vec{R}\vec{S}) =_{\beta\eta} (\lambda \vec{x}.Q(\vec{x})\vec{R}\vec{S}), \quad \text{for all } \vec{R}, \vec{S} \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D}], \\ \Rightarrow & \quad P(\vec{R})\vec{S} =_{\beta\eta} Q(\vec{R})\vec{S}, \quad \text{for all } \vec{R}, \vec{S} \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D}], \\ \Rightarrow & \quad P(\vec{d})\vec{S} =_{\beta\eta} Q(\vec{d})\vec{S}, \quad \text{for all } \vec{S} \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D}], \\ \Rightarrow & \quad P(\vec{d}) \approx_{\mathcal{D}}^{\text{ext}} Q(\vec{d}). \end{aligned}$$

(ii) By (i). ■

The proposition does not hold for class $i = 2$. Take $\mathcal{D} = \mathcal{C}_2 = \{\mathbf{f}^1, \mathbf{c}^0\}$ and

$$P[\mathbf{f}, \mathbf{c}] \equiv \lambda h:0.h(h(\mathbf{f}\mathbf{c})), \quad Q \equiv \lambda h:0.h(h(\mathbf{c})).$$

Then $P[\mathbf{f}, \mathbf{c}] \approx_{\mathcal{D}}^{\text{ext}} Q[\mathbf{f}, \mathbf{c}]$, but $\lambda f c.P[f, c] \not\approx_{\mathcal{D}}^{\text{ext}} \lambda f c.Q[f, c]$.

3E.31. PROPOSITION. Let \mathcal{D} be set of constants of class $i \neq 2$. Then

- (i) The relation $\approx_{\mathcal{D}}^{\text{ext}}$ on $\Lambda_{\rightarrow}^{\phi}[\mathcal{D}]$ is logical.
- (ii) The relations $\approx_{\mathcal{D}}^{\text{ext}}$ and $\approx_{\mathcal{D}}^{\text{obs}}$ on $\Lambda_{\rightarrow}^{\phi}[\mathcal{D}]$ coincide.

PROOF. (i) In case \mathcal{D} is of class -1 , then $M \approx_{\mathcal{D}}^{\text{ext}} N$ is universally valid by the empty implication. Therefore, the result is trivially valid.

In case \mathcal{D} is of class 0 or 1 , then $\nabla(\mathcal{D}) \in \mathbb{T}_0 \cup \mathbb{T}_1$. Hence $\nabla(\mathcal{D}) = 0^k \rightarrow 0$ for some $k \geq 1$. Then $\mathcal{D} = \{c_1^0, \dots, c_k^0\}$. Now trivially BetaEta $^{\mathcal{D}}(c, c)$ for $c \in \mathcal{D}$ of type 0 . Therefore $\approx_{\mathcal{D}}^{\text{ext}}$ is logical, by Lemma 3E.9.

For \mathcal{D} of class $i > 2$ we reason as follows. Write $\mathcal{C} = \mathcal{C}_i$. We may assume that $\mathcal{C} \cap \mathcal{D} = \emptyset$, see Remark 3E.14.

We must show that for all $M, N \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D}](A \rightarrow B)$ one has

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Leftrightarrow \forall P, Q \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D}](A)[P \approx_{\mathcal{D}}^{\text{ext}} Q \Rightarrow MP \approx_{\mathcal{D}}^{\text{ext}} NQ]. \quad (1)$$

(\Rightarrow) Assume $M[\vec{d}] \approx_{\mathcal{D}}^{\text{ext}} N[\vec{d}]$ and $P[\vec{d}] \approx_{\mathcal{D}}^{\text{ext}} Q[\vec{d}]$, with $M, N \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D}](A \rightarrow B)$ and $P, Q \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D}](B)$, in order to show $M[\vec{d}]P[\vec{d}] \approx_{\mathcal{D}}^{\text{ext}} N[\vec{d}]Q[\vec{d}]$. Then $\lambda \vec{x}.M[\vec{x}] \approx_{\mathcal{C}} \lambda \vec{x}.N[\vec{x}]$ and $\lambda \vec{x}.P[\vec{x}] \approx_{\mathcal{C}} \lambda \vec{x}.Q[\vec{x}]$, by Proposition 3E.30(i). Consider the pure closed term

$$H \equiv \lambda f:(\vec{E} \rightarrow A \rightarrow B)\lambda m:(\vec{E} \rightarrow A)\lambda \vec{x}:\vec{E}.f\vec{x}(m\vec{x}).$$

As $\approx_{\mathcal{C}}$ is logical, one has $H \approx_{\mathcal{C}} H$, $\lambda \vec{x}.M[\vec{x}] \approx_{\mathcal{C}} \lambda \vec{x}.N[\vec{x}]$, and $\lambda \vec{x}.P[\vec{x}] \approx_{\mathcal{C}} \lambda \vec{x}.Q[\vec{x}]$. So

$$\begin{aligned} \lambda \vec{x}.M[\vec{x}]P[\vec{x}] &=_{\beta\eta} H(\lambda \vec{x}.M[\vec{x}])(\lambda \vec{x}.P[\vec{x}]) \\ &\approx_{\mathcal{C}} H(\lambda \vec{x}.N[\vec{x}])(\lambda \vec{x}.Q[\vec{x}]), \\ &=_{\beta\eta} \lambda \vec{x}.N[\vec{x}]Q[\vec{x}]. \end{aligned}$$

But then again by the proposition

$$M[\vec{d}]P[\vec{d}] \approx_{\mathcal{D}}^{\text{ext}} N[\vec{d}]Q[\vec{d}].$$

(\Leftarrow) Assume the RHS of (1) in order to show $M \approx_{\mathcal{D}}^{\text{ext}} N$. That is, one has to show

$$MP_1 \cdots P_k =_{\beta\eta} NP_1 \cdots P_k, \quad (2)$$

for all $\vec{P} \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D}]$. As $P_1 \approx_{\mathcal{D}}^{\text{ext}} P_1$, by assumption it follows that $MP_1 \approx_{\mathcal{D}}^{\text{ext}} NP_1$. Hence one has (2) by definition.

(ii) That $\approx_{\mathcal{D}}^{\text{ext}}$ is $\approx_{\mathcal{D}}^{\text{obs}}$ on $\Lambda_{\rightarrow}^{\phi}[\mathcal{D}]$ follows by (i) and Proposition 3E.5. ■

3E.32. LEMMA. Let \mathcal{D} be a finite set of constants. Then \mathcal{D} is of class 2 iff one of the following cases holds.

$$\begin{aligned}\mathcal{D} &= \{\mathbf{F}:(1_{p+1} \rightarrow 0), \mathbf{c}_1, \dots, \mathbf{c}_q:0\}, p, q \geq 0; \\ \mathcal{D} &= \{\mathbf{f}:1, \mathbf{c}_1, \dots, \mathbf{c}_{q+1}:0\}, q \geq 0.\end{aligned}$$

PROOF. By Lemma 3D.16. ■

3E.33. PROPOSITION. Let \mathcal{D} be of class 2. Then the following hold.

- (i) The relation $\approx_{\mathcal{D}}^{\text{ext}}$ on $\Lambda_{\rightarrow}^{\varnothing}[\mathcal{D}]$ is logical.
- (ii) The relations $\approx_{\mathcal{D}}^{\text{ext}}$ and $\approx_{\mathcal{D}}^{\text{obs}}$ on $\Lambda_{\rightarrow}^{\varnothing}[\mathcal{D}]$ coincide.

PROOF. (i) Assume that $\mathcal{D} = \{\mathbf{F}, \mathbf{c}_1, \dots, \mathbf{c}_q\}$ (the other possibility according Lemma 3E.32 is more easy). By Proposition 3E.9 (i) it suffices to show that for $\mathbf{d} \in \mathcal{D}$ one has $S(\mathbf{d}, \mathbf{d})$. This is easy for the ones of type 0. For $\mathbf{F} : (1_{p+1} \rightarrow 0)$ assume for notational simplicity that $k = 0$, i.e. $\mathbf{F} : 2$. By Lemma 3E.7 it suffices to show $f \approx_{\mathcal{D}}^{\text{ext}} g \Rightarrow f =_{\beta\eta} g$ for $f, g \in \Lambda_{\rightarrow}^{\varnothing}[\mathcal{D}](1)$. Now elements of $\Lambda_{\rightarrow}^{\varnothing}[\mathcal{D}](1)$ are of the form

$$\lambda x_1. \mathbf{F}(\lambda x_2. \mathbf{F}(\dots(\lambda x_{m-1}. \mathbf{F}(\lambda x_m. c))\dots)),$$

where $c \equiv x_i$ or $c \equiv \mathbf{c}_j$. Therefore if $f \neq_{\beta\eta} g$, then inspecting the various possibilities (e.g. one has

$$\begin{aligned}f &\equiv \lambda x_1. \mathbf{F}(\lambda x_2. \mathbf{F}(\dots(\lambda x_{m-1}. \mathbf{F}(\lambda x_m. x_n))\dots)) \equiv KA \\ g &\equiv \lambda x_1. \mathbf{F}(\lambda x_2. \mathbf{F}(\dots(\lambda x_{m-1}. \mathbf{F}(\lambda x_m. x_1))\dots)),\end{aligned}$$

do Exercise 3F.25), one has $f(\mathbf{F}f) \neq_{\beta\eta} g(\mathbf{F}f)$ or $f(\mathbf{F}g) \neq_{\beta\eta} g(\mathbf{F}g)$, hence $f \not\approx_{\mathcal{D}}^{\text{ext}} g$.

- (ii) By (i) and Proposition 3E.5. ■

Harvesting the results we obtain the following main theorem.

3E.34. THEOREM (Statman [1980b]). Let \mathcal{D} be a finite set of typed constants of class i and $\mathcal{C} = \mathcal{C}_i$. Then

- (i) $\approx_{\mathcal{D}}^{\text{ext}}$ is logical.
- (ii) For closed terms $M, N \in \Lambda_{\rightarrow}^{\varnothing}[\mathcal{D}]$ of the same type one has

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Leftrightarrow M \approx_{\mathcal{D}}^{\text{obs}} N.$$

- (iii) For pure closed terms $M, N \in \Lambda_{\rightarrow}^{\varnothing}$ of the same type one has

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Leftrightarrow M \approx_{\mathcal{C}}^{\text{ext}} N.$$

PROOF. (i) By Propositions 3E.31 and 3E.33.

- (ii) Similarly.

(iii) Let $\mathcal{D} = \{\mathbf{d}_1^{A_1}, \dots, \mathbf{d}_k^{A_k}\}$. Then $\nabla(\mathcal{D}) = A_1 \rightarrow \dots \rightarrow A_k \rightarrow 0$ and in the notation of Lemma 3E.29 one has $\mathcal{D}_{\nabla(\mathcal{D})} = \mathcal{D}$, up to renaming constants. One has $\nabla(\mathcal{D}) \in \mathbb{T}_i$, hence by the hierarchy theorem revisited $\nabla(\mathcal{D}) \sim_{h+} \mathcal{C}_i$. Thus $\approx_{\mathcal{D}_{\nabla(\mathcal{D})}}$ is equivalent with $\approx_{\mathcal{D}_{\mathcal{C}_i}}$ on pure closed terms, by Lemma 3E.29. As $\mathcal{D}_{\nabla(\mathcal{D})} = \mathcal{D}$ and $\mathcal{D}_{\mathcal{C}_i} = \mathcal{C}_i$, we are done. ■

From now on we can write $\approx_{\mathcal{D}}$ for $\approx_{\mathcal{D}}^{\text{ext}}$ and $\approx_{\mathcal{D}}^{\text{obs}}$.

Infinite sets of constants

Remember that for \mathcal{D} a possibly infinite set of typed constants we defined

$$\text{class}(\mathcal{D}) = \max\{\text{class}(\mathcal{D}_f) \mid \mathcal{D}_f \subseteq \mathcal{D} \ \& \ \mathcal{D}_f \text{ is finite}\}.$$

The notion of class is well defined and one has $\text{class}(\mathcal{D}) \in \{-1, 0, 1, 2, 3, 4, 5\}$.

3E.35. PROPOSITION. *Let \mathcal{D} be a possibly infinite set of constants of class i . Let $A \in \mathbb{T}^0$ and $M \equiv M[\vec{d}], N \equiv N[\vec{d}] \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A)$. Then the following are equivalent.*

$$(i) \quad M \approx_{\mathcal{D}}^{\text{ext}} N.$$

$$(ii) \quad \text{For all finite } \mathcal{D}_f \subseteq \mathcal{D} \text{ containing the } \vec{d} \text{ such that } \text{class}(\mathcal{D}_f) = \text{class}(\mathcal{D}) \text{ one has}$$

$$M \approx_{\mathcal{D}_f}^{\text{ext}} N.$$

$$(iii) \quad \text{There exists a finite } \mathcal{D}_f \subseteq \mathcal{D} \text{ containing the } \vec{d} \text{ such that } \text{class}(\mathcal{D}_f) = \text{class}(\mathcal{D}) \text{ and}$$

$$M \approx_{\mathcal{D}_f}^{\text{ext}} N.$$

PROOF. (i) \Rightarrow (ii). Trivial as there are less equations to be satisfied in $M \approx_{\mathcal{D}_f}^{\text{ext}} N$.

(ii) \Rightarrow (iii). Let $\mathcal{D}_f \subseteq \mathcal{D}$ be finite with $\text{class}(\mathcal{D}_f) = \text{class}(\mathcal{D})$. Let $\mathcal{D}_{f'} = \mathcal{D}_f \cup \{\vec{d}\}$. Then $i = \text{class}(\mathcal{D}_f) \leq \text{class}(\mathcal{D}_{f'}) \leq i$, by Remark 3E.12. Therefore $\mathcal{D}_{f'}$ satisfies the conditions of (ii) and one has $M \approx_{\mathcal{D}_{f'}}^{\text{ext}} N$.

(iii) \Rightarrow (i). Suppose towards a contradiction that $M \approx_{\mathcal{D}_f}^{\text{ext}} N$ but $M \not\approx_{\mathcal{D}}^{\text{ext}} N$. Then for some finite $\mathcal{D}_{f'} \subseteq \mathcal{D}$ of class i containing \vec{d} one has $M \not\approx_{\mathcal{D}_{f'}}^{\text{ext}} N$. We distinguish cases.

Case $\text{class}(\mathcal{D}) > 2$. Since $\text{class}(\mathcal{D}_f) = \text{class}(\mathcal{D}_{f'}) = i$, Proposition 3E.30(i) implies that

$$\lambda \vec{x}. M[\vec{x}] \approx_{\mathcal{C}_i}^{\text{ext}} \lambda \vec{x}. N[\vec{x}] \ \& \ \lambda \vec{x}. M[\vec{x}] \not\approx_{\mathcal{C}_i}^{\text{ext}} \lambda \vec{x}. N[\vec{x}],$$

a contradiction.

Case $\text{class}(\mathcal{D}) = 2$. Then by Lemma 3E.32 the set \mathcal{D} consists either of a constant f^1 or $\mathbf{F}^{1_{p+1} \rightarrow 0}$ and furthermore only type 0 constants c^0 . So $\mathcal{D}_f \cup \mathcal{D}_{f'} = \mathcal{D}_f \cup \{c_1^0, \dots, c_k^0\}$. As $M \approx_{\mathcal{D}_f}^{\text{ext}} N$ by Lemma 3E.22 one has $M \approx_{\mathcal{D}_f \cup \mathcal{D}_{f'}}^{\text{ext}} N$. But then *a fortiori* $M \approx_{\mathcal{D}_{f'}}^{\text{ext}} N$, a contradiction.

Case $\text{class}(\mathcal{D}) = 1$. Then \mathcal{D} consists of only type 0 constants and we can reason similarly, again using Lemma 3E.22.

Case $\text{class}(\mathcal{D}) = 0$. Then $\mathcal{D} = \{0\}$. Hence the only subset of \mathcal{D} having the same class is \mathcal{D} itself. Therefore $\mathcal{D}_f = \mathcal{D}_{f'}$, a contradiction.

Case $\text{class}(\mathcal{D}) = -1$. We say that a type $A \in \mathbb{T}^0$ is \mathcal{D} -inhabited if $P \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A)$ for some term P . Using Proposition 2D.4 one can show

$$A \text{ is inhabited} \Leftrightarrow A \text{ is } \mathcal{D}\text{-inhabited}.$$

From this one can show for all \mathcal{D} of class -1 that

$$A \text{ inhabited} \Rightarrow \forall M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A). M \approx_{\mathcal{D}}^{\text{ext}} N.$$

In fact the assumption is not necessary, as for non-inhabited types the conclusion holds vacuously. This is a contradiction with $M \not\approx_{\mathcal{D}}^{\text{ext}} N$. ■

As a consequence of this Proposition we now show that the main theorem also holds for possibly infinite sets \mathcal{D} of typed constants.

3E.36. THEOREM. *Let \mathcal{D} be a set of typed constants of class i and $\mathcal{C} = \mathcal{C}_i$. Then*

- (i) $\approx_{\mathcal{D}}^{\text{ext}}$ is logical.
- (ii) For closed terms $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$ of the same type one has

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Leftrightarrow M \approx_{\mathcal{D}}^{\text{obs}} N.$$

- (iii) For pure closed terms $M, N \in \Lambda_{\rightarrow}^{\emptyset}$ of the same type one has

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Leftrightarrow M \approx_{\mathcal{C}}^{\text{ext}} N.$$

PROOF. (i) Let $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A \rightarrow B)$. We must show

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Leftrightarrow \forall P, Q \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A). [P \approx_{\mathcal{D}}^{\text{ext}} Q \Rightarrow MP \approx_{\mathcal{D}}^{\text{ext}} NQ].$$

(\Rightarrow) Suppose $M \approx_{\mathcal{D}}^{\text{ext}} N$ and $P \approx_{\mathcal{D}}^{\text{ext}} Q$. Let $\mathcal{D}_f \subseteq \mathcal{D}$ be a finite subset of class i containing the constants in M, N, P, Q . Then $M \approx_{\mathcal{D}_f}^{\text{ext}} N$ and $P \approx_{\mathcal{D}_f}^{\text{ext}} Q$. Since $\approx_{\mathcal{D}_f}^{\text{ext}}$ is logical by Theorem 3E.34 one has $MP \approx_{\mathcal{D}_f}^{\text{ext}} NQ$. But then $MP \approx_{\mathcal{D}}^{\text{ext}} NQ$.

(\Leftarrow) Assume the RHS. Let \mathcal{D}_f be a finite subset of \mathcal{D} of the same class containing all the constants of M, N, P, Q . One has

$$\begin{aligned} P \approx_{\mathcal{D}_f}^{\text{ext}} Q &\Rightarrow P \approx_{\mathcal{D}}^{\text{ext}} Q, && \text{by Proposition 3E.35,} \\ &\Rightarrow MP \approx_{\mathcal{D}}^{\text{ext}} NQ, && \text{by assumption,} \\ &\Rightarrow MP \approx_{\mathcal{D}_f}^{\text{ext}} NQ, && \text{by Proposition 3E.35.} \end{aligned}$$

Therefore $M \approx_{\mathcal{D}_f}^{\text{ext}} N$. Then by Proposition 3E.35 again we have $M \approx_{\mathcal{D}}^{\text{ext}} N$.

(ii) By (i) and Proposition 3E.5.

(iii) Let \mathcal{D}_f be a finite subset of \mathcal{D} of the same class. Then by Proposition 3E.35 and Theorem 3E.34

$$M \approx_{\mathcal{D}}^{\text{ext}} N \Leftrightarrow M \approx_{\mathcal{D}_f}^{\text{ext}} N \Leftrightarrow M \approx_{\mathcal{C}}^{\text{ext}} N. \blacksquare$$

Term models

In this subsection we assume that \mathcal{D} is a finite sufficient set of constants, that is, every type $A \in \mathbb{T}^0$ is inhabited by some $M \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$. This is the same as saying $\text{class}(\mathcal{D}) \geq 0$.

3E.37. DEFINITION. Define

$$\mathcal{M}[\mathcal{D}] \triangleq \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}] / \approx_{\mathcal{D}},$$

with application defined by

$$[F]_{\mathcal{D}}[M]_{\mathcal{D}} \triangleq [FM]_{\mathcal{D}}.$$

Here $[-]_{\mathcal{D}}$ denotes an equivalence class modulo $\approx_{\mathcal{D}}$.

3E.38. THEOREM. Let \mathcal{D} be sufficient. Then

- (i) Application in $\mathcal{M}[\mathcal{D}]$ is well-defined.
- (ii) For all $M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]$ one has

$$[M]^{\mathcal{M}[\mathcal{D}]} = [M]_{\approx_{\mathcal{D}}}.$$

- (iii) $\mathcal{M}[\mathcal{D}] \models M = N \Leftrightarrow M \approx_{\mathcal{D}} N$.

- (iv) $\mathcal{M}[\mathcal{D}]$ is an extensional term-model.

PROOF. (i) As the relation $\approx_{\mathcal{D}}$ is logical, application is independent of the choice of representative:

$$F \approx_{\mathcal{D}} F' \& M \approx_{\mathcal{D}} M' \Rightarrow FM \approx_{\mathcal{D}} F'M'.$$

(ii) By induction on open terms $M \in \Lambda_{\rightarrow}[\mathcal{D}]$ it follows that

$$\llbracket M \rrbracket_{\rho} = [M[\vec{x} := \rho(x_1), \dots, \rho(x_n)]]_{\mathcal{D}}.$$

Hence (ii) follows by taking $\rho(x) = [x]_{\mathcal{D}}$.

(iii) By (ii).

(iv) Use (ii) and Remark 3E.3(ii). ■

3E.39. LEMMA. Let A be represented in \mathcal{D} . Then for all $M, N \in \Lambda_{\rightarrow}^{\emptyset}(A)$, pure closed terms of type A , one has

$$M \approx_{\mathcal{D}} N \Leftrightarrow M =_{\beta\eta} N.$$

PROOF. The (\Leftarrow) direction is trivial. As to (\Rightarrow)

$$\begin{aligned} M \approx_{\mathcal{D}} N &\Leftrightarrow \forall \vec{T} \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}]. M\vec{T} =_{\beta\eta} N\vec{T} \\ &\Rightarrow M\vec{d} =_{\beta\eta} N\vec{d}, && \text{for some } \vec{d} \in \mathcal{D} \text{ since} \\ &&& A \text{ is represented in } \mathcal{D}, \\ &\Rightarrow M\vec{x} =_{\beta\eta} N\vec{x}, && \text{by Remark 3E.4 as} \\ &&& M, N \text{ are pure,} \\ &\Rightarrow M =_{\eta} \lambda\vec{x}. M\vec{x} =_{\beta\eta} \lambda\vec{x}. N\vec{x} =_{\eta} N. && \blacksquare \end{aligned}$$

3E.40. DEFINITION. (i) If \mathcal{M} is a model of $\Lambda_{\rightarrow}^{\text{Ch}}[\mathcal{D}]$, then for a type A its *A-section* is simply $\mathcal{M}(A)$.

(ii) We say that \mathcal{M} is *A-complete* (*A-complete for pure terms*) if for all closed terms (pure closed terms, respectively) M, N of type A one has

$$\mathcal{M} \models M = N \Leftrightarrow M =_{\beta\eta} N.$$

(iii) \mathcal{M} is *complete* (for pure terms) if for all types $A \in \mathbb{T}^0$ it is *A-complete* (for pure terms).

(iv) A model \mathcal{M} is called *fully abstract* if

$$\forall A \in \mathbb{T}^0 \forall x, y \in \mathcal{M}(A) [\forall f \in \mathcal{M}(A \rightarrow 0). fx = fy \Rightarrow x = y].$$

3E.41. COROLLARY. Let \mathcal{D} be sufficient. Then $\mathcal{M}[\mathcal{D}]$ has the following properties.

(i) $\mathcal{M}[\mathcal{D}]$ is an extensional term-model.

(ii) $\mathcal{M}[\mathcal{D}]$ is fully abstract.

(iii) Let A be represented in \mathcal{D} . Then $\mathcal{M}[\mathcal{D}]$ is *A-complete for pure closed terms*.

(iv) In particular, $\mathcal{M}[\mathcal{D}]$ is $\nabla(\mathcal{D})$ -complete and 0-complete for pure closed terms.

PROOF. (i) By Theorem 3E.38 the definition of application is well-defined. That extensibility holds follows from the definition of $\approx_{\mathcal{D}}$. As all combinators $[\mathsf{K}_{AB}]_{\mathcal{D}}, [\mathsf{S}_{ABC}]_{\mathcal{D}}$ are in $\mathcal{M}[\mathcal{D}]$, the structure is a model.

(ii) By Theorem 3E.38(ii). Let $x, y \in \mathcal{M}(A)$ be $[X]_{\mathcal{D}}, [Y]_{\mathcal{D}}$ respectively. Then

$$\begin{aligned} \forall f \in \mathcal{M}(A \rightarrow 0). fx = fy &\Rightarrow \forall F \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A \rightarrow 0). [FX]_{\mathcal{D}} = [FY]_{\mathcal{D}} \\ &\Rightarrow \forall F \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A \rightarrow 0). FX \approx_{\mathcal{D}} FY (: 0) \\ &\Rightarrow \forall F \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{D}](A \rightarrow 0). FX =_{\beta\eta} FY \\ &\Rightarrow X \approx_{\mathcal{D}} Y \\ &\Rightarrow [X]_{\mathcal{D}} = [Y]_{\mathcal{D}} \\ &\Rightarrow x = y. \end{aligned}$$

(iii) By Lemma 3E.39.

(iv) By (iii) and the fact that $\nabla(\mathcal{D})$ is represented in \mathcal{D} . For 0 the result is trivial. ■

3E.42. PROPOSITION. (i) Let $0 \leq i \leq j \leq 5$. Then for pure closed terms $M, N \in \Lambda_{\rightarrow}^{\phi}$

$$\mathcal{M}[\mathcal{C}_j] \models M = N \Rightarrow \mathcal{M}[\mathcal{C}_i] \models M = N.$$

(ii) $\text{Th}(\mathcal{M}[\mathcal{C}_5]) \subseteq \dots \subseteq \text{Th}(\mathcal{M}[\mathcal{C}_1])$, see Definition 3A.10(iv). All inclusions are proper.

PROOF. (i) Let $M, N \in \Lambda_{\rightarrow}^{\phi}$ be of the same type. Then

$$\begin{aligned} \mathcal{M}[\mathcal{C}_i] \not\models M = N &\Rightarrow M \not\approx_{\mathcal{C}_i} N \\ &\Rightarrow M(\vec{t}[\vec{c}]) \neq_{\beta\eta} N(\vec{t}[\vec{c}]) : 0, \text{ for some } (\vec{t}[\vec{c}]) \in \Lambda_{\rightarrow}^{\phi}[\mathcal{C}], \\ &\Rightarrow \lambda\vec{c}.M(\vec{t}[\vec{c}]) \neq_{\beta\eta} \lambda\vec{c}.N(\vec{t}[\vec{c}]) : \nabla(\mathcal{C}_i), \text{ by Remark 3E.4}, \\ &\Rightarrow \Psi(\lambda\vec{c}.M(\vec{t}[\vec{c}])) \neq_{\beta\eta} \Psi(\lambda\vec{c}.N(\vec{t}[\vec{c}])) : \nabla(\mathcal{C}_j), \\ &\quad \text{since } \nabla(\mathcal{C}_i) \leq_{\beta\eta} \nabla(\mathcal{C}_j) \text{ via some injective } \Psi, \\ &\Rightarrow \Psi(\lambda\vec{c}.M(\vec{t}[\vec{c}])) \not\approx_{\mathcal{C}_j} \Psi(\lambda\vec{c}.N(\vec{t}[\vec{c}])), \text{ since by 3E.41(iv)} \\ &\quad \text{the model } \mathcal{M}[\mathcal{C}_j] \text{ is } \nabla(\mathcal{C}_j)\text{-complete for pure terms}, \\ &\Rightarrow \mathcal{M}[\mathcal{C}_j] \not\models \Psi(\lambda\vec{c}.M(\vec{t}[\vec{c}])) = \Psi(\lambda\vec{c}.N(\vec{t}[\vec{c}])) \\ &\Rightarrow \mathcal{M}[\mathcal{C}_j] \not\models M = N, \text{ since } \mathcal{M}[\mathcal{C}_j] \text{ is a model.} \end{aligned}$$

(ii) By (i) the inclusions hold; they are proper by Exercise 3F.31. ■

3E.43. 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 one has for $M, N \in \Lambda_{\rightarrow}^{\phi}(A)$

$$\mathcal{M}[\mathcal{D}] \models M = N \iff M =_{\beta\eta} N$$

$$\Downarrow \qquad \qquad \Updownarrow$$

$$\mathcal{M}[\mathcal{D}] \models \Phi M = \Phi N \Rightarrow \Phi M =_{\beta\eta} \Phi N$$

by the definition of reducibility. ■

3E.44. COROLLARY. Let $\approx_{\mathcal{D}}^{\text{ext}}$ be logical. If $\mathcal{M}[\mathcal{D}]$ is A -complete but not B -complete for pure closed terms, then $A \not\leq_{\beta\eta} B$. ■

3E.45. COROLLARY. $\mathcal{M}[\mathcal{C}_5]$ is complete for pure terms, i.e. for all A and $M, N \in \Lambda_{\rightarrow}^{\phi}(A)$

$$\mathcal{M}[\mathcal{C}_5] \models M = N \Leftrightarrow M =_{\beta\eta} N.$$

PROOF. $\mathcal{M}[\mathcal{C}_5]$ is $\nabla(\mathcal{C}_5)$ -complete for pure terms, by Corollary 3E.41(iii). Since for every type A one has $A \leq_{\beta\eta} \top = \nabla(\mathcal{C}_5)$, by the reducibility Theorem 3D.8, it follows by Lemma 3E.43 that this model is also A -complete. ■

So $\text{Th}(\mathcal{M}[\mathcal{C}_5])$, the smallest theory, is actually just $\beta\eta$ -convertibility, which is decidable. At the other end of the hierarchy a dual property holds.

3E.46. DEFINITION. $\mathcal{M}_{\min} = \mathcal{M}[\mathcal{C}_1]$ is called the *minimal model* of $\lambda_{\rightarrow}^{\mathbb{A}}$ since it equates most terms. $\text{Th}_{\max} = \text{Th}(\mathcal{M}[\mathcal{C}_1])$ is called the *maximal theory*. The names will be justified below.

3E.47. PROPOSITION. Let $A \equiv A_1 \rightarrow \dots \rightarrow A_a \rightarrow 0 \in \mathbb{T}^0$. Let $M, N \in \Lambda_{\rightarrow}^{\emptyset}(A)$ be pure closed terms. Then the following statements are equivalent.

1. $M = N$ is inconsistent.
2. For all models \mathcal{M} of $\lambda_{\rightarrow}^{\mathbb{A}}$ one has $\mathcal{M} \not\models M = N$.
3. $\mathcal{M}_{\min} \not\models M = N$.
4. $\exists P_1 \in \Lambda^{x,y:0}(A_1) \dots P_a \in \Lambda^{x,y:0}(A_a). M \vec{P} = x \ \& \ N \vec{P} = y$.
5. $\exists F \in \Lambda^{x,y:0}(A \rightarrow 0). FM = x \ \& \ FN = y$.
6. $\exists G \in \Lambda^{\emptyset}(A \rightarrow 0^2 \rightarrow 0). FM = \lambda xy.x \ \& \ FN = \lambda xy.y$.

PROOF. (1) \Rightarrow (2) By soundness. (2) \Rightarrow (3) Trivial. (3) \Rightarrow (4) Since \mathcal{M}_{\min} consists of $\Lambda^{x,y:0} / \approx_{C_1}$. (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. ■

3E.48. COROLLARY. $\text{Th}(\mathcal{M}_{\min})$ is the unique maximally consistent extension of λ_{\rightarrow}^0 .

PROOF. By taking in the proposition the negations one has $M = N$ is consistent iff $\mathcal{M}_{\min} \models M = N$. Hence $\text{Th}(\mathcal{M}_{\min})$ contains all consistent equations. Moreover this theory is consistent. Therefore the statement follows. ■

We already did encounter $\text{Th}(\mathcal{M}_{\min})$ as \mathcal{E}_{\max} in Definition 3B.19 before. In Section 4D it will be proved that it is decidable. $\mathcal{M}[\mathcal{C}_0]$ is the degenerate model consisting of one element at each type, since

$$\forall M, N \in \Lambda_{\rightarrow}^{\emptyset}[\mathcal{C}_0](0) \ M = x = N.$$

Therefore its theory is inconsistent and hence decidable.

3E.49. REMARK. For the theories, $\text{Th}(\mathcal{M}[\mathcal{C}_2])$, $\text{Th}(\mathcal{M}[\mathcal{C}_3])$ and $\text{Th}(\mathcal{M}[\mathcal{C}_4])$ it is not known whether they are decidable.

3E.50. THEOREM. Let \mathcal{D} be a sufficient set of constants of class $i \geq 0$. Then

- (i) $\forall M, N \in \Lambda_{\rightarrow}^{\emptyset}[M \approx_{\mathcal{D}} N \Leftrightarrow M \approx_{C_i} N]$.
- (ii) $\mathcal{M}[\mathcal{D}]$ is $\nabla(C_i)$ -complete for pure terms.

PROOF. (i) By Proposition 3E.30(ii). (ii) By (i) and Corollary 3E.41(iv). ■

3E.51. REMARK. So there are exactly five canonical term-models that are not elementary equivalent (plus the degenerate term-model equating everything).

Proof of Proposition 3D.11

In the previous section the types A_{α} were introduced. The following proposition was needed to prove that these form a hierarchy.

3E.52. PROPOSITION. For $\alpha, \beta \leq \omega + 3$ one has

$$\alpha \leq \beta \Leftrightarrow A_{\alpha} \leq_{\beta} A_{\beta}.$$

PROOF. Notice that for $\alpha \leq \omega$ the cardinality of $\Lambda_{\rightarrow}^{\emptyset}(A_{\alpha})$ equals α : For example $\Lambda_{\rightarrow}^{\emptyset}(A_2) = \{\lambda xy:0.x, \lambda xy:0.y\}$ and $\Lambda_{\rightarrow}^{\emptyset}(A_{\omega}) = \{\lambda f:1\lambda x:0.f^k x \mid k \in \mathbb{N}\}$. Therefore for $\alpha, \alpha' \leq \omega$ one has $A_{\alpha} \leq_{\beta} A_{\alpha'} \Rightarrow \alpha = \alpha'$.

It remains to show that $A_{\omega+1} \not\leq_{\beta} A_{\omega}, A_{\omega+2} \not\leq_{\beta} A_{\omega+1}, A_{\omega+3} \not\leq_{\beta} A_{\omega+2}$.

As to $A_{\omega+1} \not\leq_{\beta} A_{\omega}$, consider

$$M \equiv \lambda f, g:1\lambda x:0.f(g(f(gx))),$$

$$N \equiv \lambda f, g:1\lambda x:0.f(g(g(fx))).$$

Then $M, N \in \Lambda_{\rightarrow}^{\phi}(A_{\omega+1})$, and $M \neq_{\beta\eta} N$. By Corollary 3E.41(iii) we know that $\mathcal{M}[\mathcal{C}_2]$ is A_{ω} -complete. It is not difficult to show that $\mathcal{M}[\mathcal{C}_2] \models M = N$, by analyzing the elements of $\Lambda_{\rightarrow}^{\phi}[\mathcal{C}_2](1)$. Therefore, by Corollary 3E.44, the conclusion follows.

As to $A_{\omega+2} \not\leq_{\beta\eta} A_{\omega+1}$, this is proved in [Dekkers \[1988\]](#) as follows. Consider

$$\begin{aligned} M &\equiv \lambda F:3\lambda x:0.F(\lambda f_1:1.f_1(F(\lambda f_2:1.f_2(f_1x)))) \\ N &\equiv \lambda F:3\lambda x:0.F(\lambda f_1:1.f_1(F(\lambda f_2:1.f_2(f_2x)))) . \end{aligned}$$

Then $M, N \in \Lambda_{\rightarrow}^{\phi}(A_{\omega+2})$ and $M \neq_{\beta\eta} N$. In Proposition 12 of mentioned paper it is proved that $\Phi M =_{\beta\eta} \Phi N$ for each $\Phi \in \Lambda_{\rightarrow}^{\phi}(A_{\omega+2} \rightarrow A_{\omega+1})$.

As to $A_{\omega+3} \not\leq_{\beta\eta} A_{\omega+2}$, consider

$$\begin{aligned} M &\equiv \lambda h:1_2\lambda x:0.h(hx(hxx))(hxx), \\ N &\equiv \lambda h:1_2\lambda x:0.h(hxx)(h(hxx)x) . \end{aligned}$$

Then $M, N \in \Lambda_{\rightarrow}^{\phi}(A_{\omega+3})$, and $M \neq_{\beta\eta} N$. Again $\mathcal{M}[\mathcal{C}_4]$ is $A_{\omega+2}$ -complete. It is not difficult to show that $\mathcal{M}[\mathcal{C}_4] \models M = N$, by analyzing the elements of $\Lambda_{\rightarrow}^{\phi}[\mathcal{C}_4](1_2)$. Therefore, by Corollary 3E.44, the conclusion follows. ■

3F. Exercises

- 3F.1. Convince yourself of the validity of Proposition 3C.3 for $n = 2$.
- 3F.2. Show that there are $M, N \in \Lambda_{\rightarrow}^{\phi}[\{d^0\}]((1_2 \rightarrow 1_2 \rightarrow 0) \rightarrow 0)$ such that $M \# N$, but not $M \perp N$. [Hint. Take $M \equiv [\lambda xy.x, \lambda xy.d^0] \equiv \lambda z^{1_2 \rightarrow 1_2 \rightarrow 0}.z(\lambda xy.x)(\lambda xy.d^0)$, $N \equiv [\lambda xy.d^0, \lambda xy.y]$. The $[P, Q]$ notation for pairs is from [B\[1984\]](#).]
- 3F.3. Remember $\mathcal{M}_n = \mathcal{M}_{\{1, \dots, n\}}$ and $\mathbf{c}_i = (\lambda fx.f^i x) \in \Lambda_{\rightarrow}^{\phi}(1 \rightarrow 0 \rightarrow 0)$.
 - (i) Show that for $i, j \in \mathbb{N}$ one has

$$\mathcal{M}_n \models \mathbf{c}_i = \mathbf{c}_j \Leftrightarrow i = j \vee [i, j \geq n-1 \& \forall k_1 \leq k \leq n. i \equiv j \pmod{k}] .$$

[Hint. For $a \in \mathcal{M}_n(0)$, $f \in \mathcal{M}_n(1)$ 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 $1 \leq m \leq n$, where

$$\begin{aligned} f_n(k) &= k+1, & \text{if } k < n, & \text{and } g_m(k) &= k+1, & \text{if } k < m, \\ &= n, & \text{if } k = n, & &= 1, & \text{if } k = m, \\ && & &= k, & \text{else.} \end{aligned}$$

Conclude that e.g. $\mathcal{M}_5 \models \mathbf{c}_4 = \mathbf{c}_{64}$, $\mathcal{M}_6 \not\models \mathbf{c}_4 = \mathbf{c}_{64}$ and $\mathcal{M}_6 \models \mathbf{c}_5 = \mathbf{c}_{65}$.

- (ii) Conclude that $\mathcal{M}_n \equiv_{1 \rightarrow 0 \rightarrow 0} \mathcal{M}_m \Leftrightarrow n = m$, see Definitions 3A.14 and 3B.4.
- (iii) Show directly that $\bigcap_n \text{Th}(\mathcal{M}_n)(1) = \mathcal{E}_{\beta\eta}(1)$.
- (iv) Show, using results in Section 3D, that $\bigcap_n \text{Th}(\mathcal{M}_n) = \text{Th}(\mathcal{M}_{\mathbb{N}}) = \mathcal{E}_{\beta\eta}$.

- 3F.4. 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 2E.19. Define $s(A)$ to be the number of occurrences of atoms in the type $A \in \mathbb{T}^0$, i.e.

$$\begin{aligned}s(0) &\triangleq 1 \\ s(A \rightarrow B) &\triangleq s(A) + s(B).\end{aligned}$$

Write $\#X$ for the cardinality of the set X . Show the following.

- (i) $2_n \leq 2_{n+p}$.
- (ii) $2_{n+2}^{2p+1} \leq 2_{n+p+3}$.
- (iii) $2_n^{2p} \leq 2_{n+p}$.
- (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)?

3F.5. Show that if \mathcal{M} is a type model, then for the corresponding polynomial type model \mathcal{M}^* one has $\text{Th}(\mathcal{M}^*) = \text{Th}(\mathcal{M})$.

3F.6. Show that

$$A_1 \rightarrow \cdots \rightarrow A_n \rightarrow 0 \leq_{\beta\eta} A_{\pi 1} \rightarrow \cdots \rightarrow A_{\pi n} \rightarrow 0,$$

for any permutation $\pi \in S_n$

3F.7. Let $A = (2 \rightarrow 2 \rightarrow 0) \rightarrow 2 \rightarrow 0$ and
 $B = (0 \rightarrow 1^2 \rightarrow 0) \rightarrow 1_2 \rightarrow (0 \rightarrow 1 \rightarrow 0) \rightarrow 0^2 \rightarrow 0$. Show that

$$A \leq_{\beta\eta} B.$$

[Hint. Use the term $\lambda z : A \lambda u_1 : (0 \rightarrow 1^2 \rightarrow 0) \lambda u_2 : 1_2 \lambda u_3 : (0 \rightarrow 2) \lambda x_1 x_2 : 0. z[y_1, y_2 : 2. u_1 x_1 (\lambda w : 0. y_1(u_2 w)) (\lambda w : 0. y_2(u_2 w))] [u_3 x_2].$]

3F.8. Let $A = (1^2 \rightarrow 0) \rightarrow 0$. Show that

$$A \leq_{\beta\eta} 1_2 \rightarrow 2 \rightarrow 0.$$

[Hint. Use the term $\lambda M : A \lambda p : 1_2 \lambda F : 2. M(\lambda f, g : 1. F(\lambda z : 0. p(fz)(gz)))$.]

3F.9. (i) Show that

$$\begin{pmatrix} 2 \\ 3 & 4 \end{pmatrix} \leq_{\beta\eta} 1 \rightarrow 1 \rightarrow \begin{pmatrix} 2 \\ 3 & 3 \end{pmatrix}.$$

(ii) Show that

$$\begin{pmatrix} 2 \\ 3 & 3 \end{pmatrix} \leq_{\beta\eta} 1 \rightarrow 1 \rightarrow \begin{pmatrix} 2 \\ 3 \end{pmatrix}.$$

(iii) * Show that

$$\begin{pmatrix} 2 & 2 \\ 3 & 2 \end{pmatrix} \leq_{\beta\eta} 1_2 \rightarrow \begin{pmatrix} 2 \\ 3 & 2 \end{pmatrix}.$$

[Hint. Use $\Phi = \lambda M \lambda p : 1_2 \lambda H'_1 H_2. M$
 $[\lambda f_{11}, f_{12} : 1_2. H'_1(\lambda xy : 0. p(f_{12}xy, H_2 f_{11}))$
 $[\lambda f_{21} : 1_3 \lambda f_{22} : 1_2. H_2 f_{21} f_{22}].]$]

3F.10. Show directly that $3 \rightarrow 0 \leq_{\beta\eta} 1 \rightarrow 1 \rightarrow 0 \rightarrow 0$. [Hint. Use

$$\Phi \equiv \lambda M : 3 \lambda f, g : 1 \lambda z : 0. M(\lambda h : 1. f(h(g(hz)))).$$

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 $\beta\eta$) on these.]

- 3F.11. Give example of $F, G \in \Lambda[C_4]$ such that $Fh_2 =_{\beta\eta} Gh_2$, but $F \neq_{\beta\eta} G$, where $h_2 \equiv \lambda z:0.\Phi(\lambda g:1.g(gz))$.
- 3F.12. Suppose $(\vec{A} \rightarrow 0), (\vec{B} \rightarrow 0) \in \mathbb{T}_i$, with $i > 2$. Then
- (i) $(\vec{A} \rightarrow \vec{B} \rightarrow 0) \in \mathbb{T}_i$.
 - (ii) $(\vec{A} \rightarrow \vec{B} \rightarrow 0) \sim_h \vec{A} \rightarrow 0$.
- 3F.13. (i) Suppose that $\text{class}(A) \geq 0$. Then

$$\begin{aligned} A \leq_{\beta\eta} B &\Rightarrow (C \rightarrow A) \leq_{\beta\eta} (C \rightarrow B). \\ A \sim_{\beta\eta} B &\Rightarrow (C \rightarrow A) \sim_{\beta\eta} (C \rightarrow B). \end{aligned}$$

[Hint. Distinguish cases for the class of A .]

- (ii) Show that in (i) the condition on A cannot be dropped.
 [Hint. Take $A \equiv 1_2 \rightarrow 0$, $B \equiv C \equiv 0$.]

- 3F.14. Show that the relations \leq_h and \leq_{h+} are transitive.

- 3F.15. ([Joly \[2001a\]](#), Lemma 2, p. 981, based on an idea of Dana Scott) Show that any type A is reducible to

$$1_2 \rightarrow 2 \rightarrow 0 = (0 \rightarrow (0 \rightarrow 0)) \rightarrow ((0 \rightarrow 0) \rightarrow 0) \rightarrow 0.$$

[Hint. We regard each closed term of type A as an untyped lambda term and then we retype all the variables as type 0 replacing applications XY by fXY ($\triangleq X \bullet Y$) and abstractions $\lambda x.X$ by $g(\lambda x.X)$ ($\triangleq \lambda^{\bullet} x.X$) where $f : 1_2$, $g : 2$. Scott thinks of f and g as a retract pair satisfying $g \circ f = I$ (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 $U_A : A \rightarrow 0$ and $V_A : 0 \rightarrow A$ as follows.]

$$\begin{aligned} U_0 &\triangleq \lambda x:0.x; \quad V_0 \triangleq \lambda x:0.x; \\ U_{A \rightarrow B} &\triangleq \lambda u.g(\lambda x:0.U_B(\textcolor{red}{u}(V_Ax))); \\ V_{A \rightarrow B} &\triangleq \lambda v\lambda y.V_B(fv(U_Ay)). \end{aligned}$$

Let $A = A_1 \rightarrow \dots \rightarrow A_a \rightarrow 0$, $A_i = A_{i_1} \rightarrow \dots \rightarrow A_{i_r} \rightarrow 0$ and write for a closed $M : A$

$$M = \lambda y_1 \dots y_a.y_i(M_1 y_1 \dots y_{\textcolor{teal}{a}}) \dots (\textcolor{teal}{M}_{r_i} y_1 \dots y_{\textcolor{teal}{a}}),$$

with the M_i closed (this is the “ Φ -nf” if the M_i are written similarly). Then

$$U_A M \rightarrow \lambda^{\bullet} \vec{x}.x_i(U_{B_1}(\textcolor{teal}{M}_1 \vec{x})) \bullet \dots \bullet (U_{B_n}(\textcolor{teal}{M}_n \vec{x})),$$

where $B_j = A_1 \rightarrow \dots \rightarrow A_a \rightarrow A_{ij}$, 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

$$U_A M =_{\beta\eta} U_A N \Rightarrow M =_{\beta\eta} N.$$

Conclude that $A \leq_{\beta\eta} 1_2 \rightarrow 2 \rightarrow 0$ via $\Phi \equiv \lambda bfg.U_A b.$]

- 3F.16. In this exercise the combinatorics of the argument needed in the proof of 3D.6 is analyzed. Let $(\lambda F:2.M) : 3$. Define M^+ to be the long $\beta\eta$ nf of $M[F := H]$, where

$$H = (\lambda h:1.f(h(g(hz)))) \in \Lambda_{\rightarrow}^{\{f,g:1,z:0\}}(2).$$

Write $\text{cut}_{g \rightarrow z}(P) = P[g := Kz]$.

- (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 $\text{cut}_{g \rightarrow z}(P)$ is a proper subterm of $\text{cut}_{g \rightarrow z}(M^+)$.
- (ii) Let $M \equiv F(\lambda x:0.N)$. Then we know

$$M^+ =_{\beta\eta} f(N^+[x := g(N^+[x := z])]).$$

Show that if $g(P) \subseteq M^+$ is maximal and

$$\text{length}(\text{cut}_{g \rightarrow z}(P)) + 1 = \text{length}(\text{cut}_{g \rightarrow z}(M^+)),$$

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 $\text{length}(\text{cut}_{g \rightarrow z}(P)) + 1 = \text{length}(\text{cut}_{g \rightarrow z}(M^+))$ 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^+ .

3F.17. Show directly that

$$2 \rightarrow 1_2 \rightarrow 0 \leq_{\beta\eta} 1^2 \rightarrow 1_2 \rightarrow 0 \rightarrow 0,$$

via $\Phi \equiv \lambda M:2 \rightarrow 1_2 \rightarrow 0 \lambda f g:1 \lambda b:1_2 \lambda x:0.M(\lambda h.f(h(g(hx))))b$.

Finish the alternative proof that $\top = 1_2 \rightarrow 0 \rightarrow 0$ satisfies $\forall A \in \mathbb{T}(\lambda^0_\rightarrow).A \leq_{\beta\eta} \top$, by showing in the style of the proof of Proposition 3D.7 the easy

$$1^2 \rightarrow 1_2 \rightarrow 0 \rightarrow 0 \leq_{\beta\eta} 1_2 \rightarrow 0 \rightarrow 0.$$

3F.18. Show directly that (without the reducibility theorem)

$$3 \rightarrow 0 \rightarrow 0 \leq_{\beta\eta} 1_2 \rightarrow 0 \rightarrow 0 = \top.$$

3F.19. Show directly the following.

- (i) $1_3 \rightarrow 1_2 \rightarrow 0 \leq_{\beta\eta} \top$.
- (ii) For any type A of rank ≤ 2 one has $A \leq_{\beta\eta} \top$.

3F.20. Show that all elements $g \in \mathcal{M}_2(0 \rightarrow 0)$ satisfy $g^2 = g^4$. Conclude that $\mathcal{T} \not\vdash \mathcal{M}_2$.

3F.21. Let \mathcal{D} have enough constants. Show that the class of \mathcal{D} is not

$$\min\{i \mid \forall D.[D \text{ represented in } \mathcal{D} \Rightarrow D \leq_{\beta\eta} \nabla(\mathcal{C}_i)]\}.$$

[Hint. Consider $\mathcal{D} = \{\mathbf{c}^0, \mathbf{d}^0, \mathbf{e}^0\}$.]

3F.22. 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.

3F.23. Let $\mathcal{M} = \mathcal{M}_{\min}$.

- (i) Determine in $\mathcal{M}(1 \rightarrow 0 \rightarrow 0)$ 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 $\mathcal{M}(1_2 \rightarrow 0 \rightarrow 0)$.

3F.24. Let \mathcal{M} be a model and let $|\mathcal{M}_0| \leq \kappa$. By Example 3C.24 there exists a partial surjective homomorphism $h : \mathcal{M}_\kappa \dashv \mathcal{M}$.

- (i) Show that $h^{-1}(\mathcal{M}) \subseteq \mathcal{M}_\kappa$ is closed under λ -definability. [Hint. Use Example 3C.27.]
- (ii) Show that as in Example 3C.28 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.

3F.25. Let $\mathcal{D} = \{\mathbf{F}^2, \mathbf{c}_1^0, \dots, \mathbf{c}_n^0\}$.

(i) Give a characterization of the elements of $\Lambda_{\rightarrow}^{\phi}[\mathcal{D}](1)$.

(ii) For $f, g \in \Lambda_{\rightarrow}^{\phi}[\mathcal{D}](1)$ show that $f \neq_{\beta\eta} g \Rightarrow f \not\sim_{\mathcal{D}} g$ by applying both f, g to $\mathbf{F}f$ or $\mathbf{F}g$.

3F.26. Prove the following.

$$\begin{aligned} 1_2 \rightarrow 0 \rightarrow 0 &\leq_{\beta\eta} ((1_2 \rightarrow 0) \rightarrow 0) \rightarrow 0 \rightarrow 0, \text{ via} \\ &\lambda m \lambda F : ((1_2 \rightarrow 0) \rightarrow 0) \lambda x : 0.F(\lambda h : 1_2.mhx) \text{ or via} \\ &\lambda m \lambda F : ((1_2 \rightarrow 0) \rightarrow 0) \lambda x : 0.m(\lambda p q : 0.F(\lambda h : 1_2.hpq))x. \\ 1_2 \rightarrow 0 \rightarrow 0 &\leq_{\beta\eta} (1 \rightarrow 1 \rightarrow 0) \rightarrow 0 \rightarrow 0 \\ &\text{via } \lambda m H x.m(\lambda a b.H(\mathbf{K}a)(\mathbf{K}b))x. \end{aligned}$$

3F.27. Show that $\mathbb{T}_2 = \{(1_p \rightarrow 0) \rightarrow 0^q \rightarrow 0 \mid p \cdot q > 0\}$.

3F.28. In this Exercises we show that $A \sim_{\beta\eta} B \& A \sim_{h^+} B$, for all $A, B \in \mathbb{T}_2$.

(i) First we establish for $p \geq 1$

$$1 \rightarrow 0 \rightarrow 0 \sim_{\beta\eta} 1 \rightarrow 0^p \rightarrow 0 \& 1 \rightarrow 0 \rightarrow 0 \sim_{h^+} 1 \rightarrow 0^p \rightarrow 0.$$

(a) Show $1 \rightarrow 0 \rightarrow 0 \leq_h 1 \rightarrow 0^p \rightarrow 0$. Therefore

$$1 \rightarrow 0 \rightarrow 0 \leq_{\beta\eta} 1 \rightarrow 0^p \rightarrow 0 \& 1 \rightarrow 0 \rightarrow 0 \leq_{h^+} 1 \rightarrow 0^p \rightarrow 0.$$

(b) Show $1 \rightarrow 0^p \rightarrow 0 \leq_{h^+} 1 \rightarrow 0 \rightarrow 0$. [Hint. Using inhabitation machines one sees that the long normal forms of terms in $\Lambda_{\rightarrow}^{\phi}(1 \rightarrow 0^p \rightarrow 0)$ are of the form $L_i^n \equiv \lambda f : 1 \lambda x_1 \cdots x_p : 0.f^n x_i$, with $n \geq 0$ and $1 \leq i \leq p$. Define $\Phi_i : (1 \rightarrow 0^p \rightarrow 0) \rightarrow (1 \rightarrow 0 \rightarrow 0)$, with $i = 1, 2$, as follows.

$$\begin{aligned} \Phi_1 L &\triangleq \lambda f : 1 \lambda x : 0.Lfx^{\sim p}; \\ \Phi_2 L &\triangleq \lambda f : 1 \lambda x : 0.L(f^1 x) \cdots (f^p x). \end{aligned}$$

Then $\Phi_1 L_i^n =_{\beta\eta} \mathbf{c}_n$ and $\Phi_2 L_i^n =_{\beta\eta} \mathbf{c}_i$. Hence for $M, N \in \Lambda_{\rightarrow}^{\phi}(1 \rightarrow 0_q \rightarrow 0)$

$$M \neq_{\beta\eta} N \Rightarrow \Phi_1 M \neq_{\beta\eta} \Phi_1 N \text{ or } \Phi_2 M \neq_{\beta\eta} \Phi_2 N.]$$

(c) Conclude that also $1 \rightarrow 0^p \rightarrow 0 \leq_{\beta\eta} 1 \rightarrow 0 \rightarrow 0$, by taking as reducing term

$$\Phi \equiv \lambda m f x.P_2(\Phi_1 m)(\Phi_2 m),$$

where P_2 λ -defines a polynomial injection $p_2 : \mathbb{N}^2 \rightarrow \mathbb{N}$.

(ii) Now we establish for $p \geq 1, q \geq 0$ that

$$1 \rightarrow 0 \rightarrow 0 \sim_{\beta\eta} (1_p \rightarrow 0) \rightarrow 0^q \rightarrow 0 \& 1 \rightarrow 0 \rightarrow 0 \sim_{h^+} 1_p \rightarrow 0^q \rightarrow 0.$$

(a) Show $1 \rightarrow 0 \rightarrow 0 \leq_h (1_p \rightarrow 0) \rightarrow 0^q \rightarrow 0$ using

$$\Phi \equiv \lambda m F x_1 \cdots x_q.m(\lambda z.F(\lambda y_1 \cdots y_p.z)).$$

(b) Show $(1_p \rightarrow 0) \rightarrow 0^q \rightarrow 0 \leq_{h^+} 1 \rightarrow 0 \rightarrow 0$. [Hint. For $L \in \Lambda_{\rightarrow}^{\phi}((1_p \rightarrow 0) \rightarrow 0^q \rightarrow 0)$ its lnf is of one of the following forms.

$$\begin{aligned} L^{n,k,r} &= \lambda F : (1_p \rightarrow 0) \lambda y_1 \cdots y_q : 0.F(\lambda \vec{z}_1 \cdots F(\lambda \vec{z}_n.z_{kr})..) \\ M^{n,s} &= \lambda F : (1_p \rightarrow 0) \lambda y_1 \cdots y_q : 0.F(\lambda \vec{z}_1 \cdots F(\lambda \vec{z}_n.y_s)..), \end{aligned}$$

where $\vec{z}_k = z_{k1} \cdots z_{kp}$, $1 \leq k \leq n$, $1 \leq r \leq p$, and $1 \leq s \leq q$, in case $q > 0$ (otherwise the $M^{n,s}$ does not exist). Define three terms $O_1, O_2, O_3 \in \Lambda^{\varnothing}_{\rightarrow}(1 \rightarrow 0 \rightarrow 1_p \rightarrow 0)$ as follows.

$$\begin{aligned} O_1 &\triangleq \lambda f x g. g(f^1 x) \cdots (f^p x) \\ O_2 &\triangleq \lambda f x g. f(g x^{\sim p}) \\ O_3 &\triangleq \lambda f x g. f(g(f(g x^{\sim p}))^{\sim p}). \end{aligned}$$

Define terms $\Phi_i \in \Lambda^{\varnothing}_{\rightarrow}(((1_p \rightarrow 0) \rightarrow 0^q \rightarrow 0) \rightarrow 1 \rightarrow 0 \rightarrow 0)$ for $1 \leq i \leq 3$ by

$$\begin{aligned} \Phi_1 L &\triangleq \lambda f x. L(O_1 f x)(f^{p+1} x) \cdots (f^{p+q} x); \\ \Phi_i L &\triangleq \lambda f x. L(O_i f x)x^{\sim q}, \quad \text{for } i \in \{2, 3\}. \end{aligned}$$

Verify that

$$\begin{aligned} \Phi_1 L^{n,k,r} &= \mathbf{c}_r \\ \Phi_1 M^{n,s} &= \mathbf{c}_{p+s} \\ \Phi_2 L^{n,k,r} &= \mathbf{c}_n \\ \Phi_2 M^{n,s} &= \mathbf{c}_n \\ \Phi_3 L^{n,k,r} &= \mathbf{c}_{2n+1-k} \\ \Phi_3 M^{n,s} &= \mathbf{c}_n. \end{aligned}$$

Therefore if $M \neq_{\beta\eta} N$ are terms in $\Lambda^{\varnothing}_{\rightarrow}(1_p \rightarrow 0^q \rightarrow 0)$, then for at least one $i \in \{1, 2, 3\}$ one has $\Phi_i(M) \neq_{\beta\eta} \Phi_i(N)$.

(c) Show $1_p \rightarrow 0^q \rightarrow 0 \leq_{\beta\eta} 1 \rightarrow 0 \rightarrow 0$, using a polynomial injection $p_3 : \mathbb{N}^3 \rightarrow \mathbb{N}$.

3F.29. Show that for all $A, B \notin \mathbb{T}_1 \cup \mathbb{T}_2$ one has $A \sim_{\beta\eta} B \Rightarrow A \sim_h B$.

3F.30. Let A be an inhabited small type of rank > 3 . Show that

$$3 \rightarrow 0 \rightarrow 0 \leq_m A.$$

[Hint. For small B of rank ≥ 2 one has $B \equiv B_1 \rightarrow \cdots B_b \rightarrow 0$ with $B_i \equiv B_{i_1} \rightarrow 0$ for all i and $\text{rank}(B_{i_0}) = \text{rank}(B) - 2$ for some i_0 . Define for such B the term

$$X^B \in \Lambda^{\varnothing}[F^2](B),$$

where F^2 is a variable of type 2.

$$\begin{aligned} X^B &\triangleq \lambda x_1 \cdots x_b. F^2 x_{i_0}, && \text{if } \text{rank}(B) = 2; \\ &\triangleq \lambda x_1 \cdots x_b. F^2(\lambda y:0.x_{i_0}(\lambda y_1 \cdots y_k.y)), && \text{if } \text{rank}(B) = 3 \text{ and} \\ &&& \text{where } B_{i_0} \text{ having} \\ &&& \text{rank 1 is } 0^k \rightarrow 0; \\ &\triangleq \lambda x_1 \cdots x_b. x_{i_0} X^{B_{i_0}}, && \text{if } \text{rank}(B) > 3. \end{aligned}$$

(Here $X^{B_{i_0}}$ is well-defined since B_{i_0} is also small.) As A is inhabited, take $\lambda x_1 \cdots x_b. N \in \Lambda^{\varnothing}(A)$. Define $\Psi : (3 \rightarrow 0 \rightarrow 0) \rightarrow A$ by

$$\Psi(M) \triangleq \lambda x_1 \cdots x_b. M(\lambda F^2.x_i X^{A_{i_1}})N,$$

where i is such that A_{i_1} has rank ≥ 2 . Show that Ψ works.]

3F.31. Consider the following equations.

1. $\lambda f:1\lambda x:0.fx = \lambda f:1\lambda x:0.f(fx)$;
2. $\lambda f,g:1\lambda x:0.f(g(g(fx))) = \lambda f,g:1\lambda x:0.f(g(f(gx)))$;

3. $\lambda F:3\lambda x:0.F(\lambda f_1:1.f_1(F(\lambda f_2:1.f_2(f_1x)))) = \lambda F:3\lambda x:0.F(\lambda f_1:1.f_1(F(\lambda f_2:1.f_2(f_2x)))).$
4. $\lambda h:1_2\lambda x:0.h(hx(hxx))(hxx) = \lambda h:1_2\lambda x:0.h(hxx)(h(hxx)x).$
- (i) Show that 1 holds in \mathcal{M}_{C_1} , but not in \mathcal{M}_{C_2} .
- (ii) Show that 2 holds in \mathcal{M}_{C_2} , but not in \mathcal{M}_{C_3} .
- (iii) Show that 3 holds in \mathcal{M}_{C_3} , but not in \mathcal{M}_{C_4} .
[Hint. Use Lemmas 7a and 11 in Dekkers [1988].]
- (iv) Show that 4 holds in \mathcal{M}_{C_4} , but not in \mathcal{M}_{C_5} .
- 3F.32. Construct six pure closed terms of the same type in order to show that the five canonical theories are maximally different. I.e. we want terms M_1, \dots, M_6 such that in $\text{Th}(\mathcal{M}_{C_5})$ the M_1, \dots, M_6 are mutually different; also $M_6 = M_5$ in $\text{Th}(\mathcal{M}_{C_4})$, but different from M_1, \dots, M_4 ; also $M_5 = M_4$ in $\text{Th}(\mathcal{M}_{C_3})$, but different from M_1, \dots, M_3 ; also $M_4 = M_3$ in $\text{Th}(\mathcal{M}_{C_2})$, but different from M_1, M_2 ; also $M_3 = M_2$ in $\text{Th}(\mathcal{M}_{C_1})$, but different from M_1 ; finally $M_2 = M_1$ in $\text{Th}(\mathcal{M}_{C_0})$.
[Hint. Use the previous exercise and a polynomially defined pairing operator.]
- 3F.33. Let \mathcal{M} be a typed lambda model. Let S be the logical relation determined by $S_0 = \emptyset$. Show that $S_0^* \neq \emptyset$.
- 3F.34. We work with $\lambda_{\rightarrow}^{\text{Ch}}$ over \mathbb{T}^0 . Consider the full type structure $\mathcal{M}_1 = \mathcal{M}_{\mathbb{N}}$ over the natural numbers, the open term model $\mathcal{M}_2 = \mathcal{M}(\beta\eta)$, and the closed term model $\mathcal{M}_3 = \mathcal{M}^o[\{\mathbf{h}^1, \mathbf{c}^0\}](\beta\eta)$. For these models consider three times the Gandy-Hull

$$\begin{aligned}\mathcal{G}_1 &= \mathcal{G}_{\{\mathbf{S}:1,0:0\}}(\mathcal{M}_1) \\ \mathcal{G}_2 &= \mathcal{G}_{\{[f:1],[x:0]\}}(\mathcal{M}_2) \\ \mathcal{G}_3 &= \mathcal{G}_{\{\mathbf{h}:1,[\mathbf{c}:0]\}}(\mathcal{M}_3),\end{aligned}$$

where \mathbf{S} is the successor function and $0 \in \mathbb{N}$, f, x are variables and \mathbf{h}, \mathbf{c} are constants, of type 1, 0 respectively. Prove

$$\mathcal{G}_1 \cong \mathcal{G}_2 \cong \mathcal{G}_3.$$

[Hint. Consider the logical relation R on $\mathcal{M}_3 \times \mathcal{M}_2 \times \mathcal{M}_1$ determined by

$$R_0 = \{ \langle [\mathbf{h}^k(\mathbf{c})], [f^k(x)], k \rangle \mid k \in \mathbb{N} \}.$$

Apply the Fundamental Theorem for logical relations.]

- 3F.35. A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is *slantwise λ -definable*, see also Fortune, Leivant, and O'Donnell [1983] and Leivant [1990] if there is a substitution operator $+$ for types and a closed term $F \in \Lambda^o(\mathbb{N}^+ \rightarrow \mathbb{N})$ such that

$$F \mathbf{c}_k^+ =_{\beta\eta} \mathbf{c}_{f(k)}.$$

This can be generalized to functions of k -arguments, allowing for each argument a different substitution operator.

- (i) Show that $f(x, y) = x^y$ is slantwise λ -definable.
- (ii) Show that the predecessor function is slantwise λ -definable.
- (iii) Show that subtraction is not slantwise λ -definable. [Hint. Suppose towards a contradiction that a term $m : \text{Nat}_{\tau} \rightarrow \text{Nat}_{\rho} \rightarrow \text{Nat}_{\sigma}$ defines subtraction. Use the Finite Completeness Theorem, Proposition 3D.33, for $A = \text{Nat}_{\sigma}$ and $M = \mathbf{c}_0$.]

- 3F.36. (Finite generation, Joly [2002]) Let $A \in \mathbb{T}$. Then A is said to be *finitely generated* if there exist types A_1, \dots, A_t and terms $M_1 : A_1, \dots, A_t : M_t$ such that for any $M : A$, M is $\beta\eta$ convertible to an applicative combination of M_1, \dots, M_t .

Example. $\text{Nat} = 1 \rightarrow 0 \rightarrow 0$ is finitely generated by $c_0 \equiv (\lambda f x. x) : \text{Nat}$ and $S \equiv (\lambda n f x. f(fx)) : (\text{Nat} \rightarrow \text{Nat})$.

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 $F@M =_{\beta\eta} N$ (F is *surjective*).

A type A is said to be *poor* if there is a finite sequence of variables \vec{x} , such that every $M \in \Lambda_{\rightarrow}^{\phi}(A)$ in $\beta\eta$ -nf has $\text{FV}(M) \subseteq \vec{x}$. Otherwise A is said to be *rich*.

Example. Let $A = (1 \rightarrow 0) \rightarrow 0 \rightarrow 0$ is poor. A typical $\beta\eta$ -nf of type A has the shape $\lambda F \lambda x(F(\lambda x(\dots(F(\lambda y(F(\lambda y \dots x \dots))) \dots)))$). One allows the term to violate the variable convention (that asks different occurrences of bound variables to be different). 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 $(3) \Rightarrow (4)$.) 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 $v_1 : A_1, \dots, v_m : A_m$ both free and bound. We shall transform X using a fixed set of new variables. First we assume the set of A_i is closed under subtype. (a) Show that we can assume that X is fully expanded. For example, if X has the form

$$\lambda x_1 \dots x_t. (\lambda x. X_0) X_1 \dots X_s$$

then $(\lambda x. X_0) X_1 \dots X_s$ has one of the A_i as a type (just normalize and consider the type of the head variable). Thus we can eta expand

$$\lambda x_1 \dots x_t. (\lambda x. X_0) X_1 \dots X_s$$

and repeat recursively. We need only double the set of variables to do this. We do this keeping the same notation. (b) Thus given

$$X = \lambda x_1 \dots x_t. (\lambda x. X_0) X_1 \dots X_s$$

we have $X_0 = \lambda y_1 \cdots y_r.Y$, where $Y : 0$. Now if $r > m$, each multiple occurrence of v_i in the prefix $\lambda y_1 \cdots y_r$ is dummy and those that occur in the initial segment $\lambda y_1 \cdots y_s$ can be removed with the corresponding X_j . The remaining variables will be labelled z_1, \dots, z_k . The remaining X_j will be labelled Z_1, \dots, Z_l . Note that $r - s + t < m + 1$. Thus

$$X = \lambda x_1 \cdots x_t.(\lambda z_1 \cdots z_k Y)Z_1 \cdots Z_l,$$

where $k < 2m + 1$. We can now repeat this analysis recursively on Y , and Z_1, \dots, 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) \Rightarrow (2). We have already shown that the lambda definability problem for the monster M is undecidable. Suppose (1) and $\neg(2)$ 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 $M, N : C$

$$P(B(n)) \models M = N \Rightarrow P(n) \models [0 := B]M = [0 := B]N.$$

Conclude from this that lambda definability for M is decidable, which is not the case.

CHAPTER 4

DEFINABILITY, UNIFICATION AND MATCHING

4A. 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 2D.17 as follows.

$$\begin{aligned} X(0) &= X, \\ X(A \rightarrow B) &= X(B)^{X(A)}; \\ \mathcal{M}_X &= \{X(A)\}_{A \in \Pi}. \end{aligned}$$

Note that if X is finite then all the $X(A)$ are finite. In that case we can represent each element 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 $X(0 \rightarrow 0)$ as follows. If 0 is followed by 0 to the right this means that 0 is mapped onto 0, etcetera.

$\begin{array}{ c c } \hline 0 & 0 \\ \hline 1 & 0 \\ \hline \end{array}$	$\begin{array}{ c c } \hline 0 & 1 \\ \hline 1 & 1 \\ \hline \end{array}$	$\begin{array}{ c c } \hline 0 & 0 \\ \hline 1 & 1 \\ \hline \end{array}$	$\begin{array}{ c c } \hline 0 & 1 \\ \hline 1 & 0 \\ \hline \end{array}$
---	---	---	---

Any element of the model can be expressed in a similar way, for instance the following table represents an element of $X((0 \rightarrow 0) \rightarrow 0)$.

$\begin{array}{ c c } \hline 0 & 0 \\ \hline 1 & 0 \\ \hline \end{array}$	0
$\begin{array}{ c c } \hline 0 & 1 \\ \hline 1 & 1 \\ \hline \end{array}$	0
$\begin{array}{ c c } \hline 0 & 0 \\ \hline 1 & 1 \\ \hline \end{array}$	0
$\begin{array}{ c c } \hline 0 & 1 \\ \hline 1 & 0 \\ \hline \end{array}$	1

We know that $I \equiv \lambda x.x$ is the only closed $\beta\eta$ -nf of type $0 \rightarrow 0$. As $\llbracket I \rrbracket = \mathbf{1}_X$, the identity on X is the only function of $X(0 \rightarrow 0)$ that is denoted by a closed term.

4A.1. DEFINITION. Let $\mathcal{M} = \mathcal{M}_X$ be a type structure over a finite set X and let $d \in \mathcal{M}(A)$. Then d is called **λ -definable** if $d = \llbracket M \rrbracket^{\mathcal{M}}$, for some $M \in \Lambda^o(A)$.

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 method of showing that decision problems are undecidable proceeds via reducing them to well-known undecidable problems (and eventually to the undecidable Halting problem).

4A.2. DEFINITION. (i) A *decision problem* is a subset $P \subseteq \mathbb{N}$. This P is called *decidable* if its characteristic function $K_P : \mathbb{N} \rightarrow \{0, 1\}$ is computable. An *instance of a problem* is the question “ $n \in P?$ ”. Often problems are subsets of syntactic objects, like terms or descriptions of automata, that are considered as subsets of \mathbb{N} via some coding.

(ii) Let $P, Q \subseteq \mathbb{N}$ be problems. Then P is (*many-one*) *reducible* to problem Q , notation $P \leq_m Q$, if there is a computable function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that

$$n \in P \Leftrightarrow f(n) \in Q.$$

(iii) More generally, a problem P is *Turing reducible* to a problem Q , notation $P \leq_T Q$, if the characteristic function K_P is computable in K_Q , see e.g. Rogers Jr. [1967].

The following is well-known.

4A.3. PROPOSITION. *Let P, Q be problems.*

- (i) *If $P \leq_m Q$, then $P \leq_T Q$.*
- (ii) *If $P \leq_T Q$, then the undecidability of P implies that of Q .*

PROOF. (i) Suppose that $P \leq_m Q$. Then there is a computable function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that $\forall n \in \mathbb{N}. [n \in P \Leftrightarrow f(n) \in Q]$. Therefore $K_P(n) = K_Q(f(n))$. Hence $P \leq_T Q$.

(ii) Suppose that $P \leq_T Q$ and that Q is decidable, in order to show that P is decidable. Then K_Q is computable and so is K_P , as it is computable in K_Q . ■

The proof of Loader’s result proceeds by reducing the two-letter word rewriting problem, which is well-known to be undecidable, to the λ -definability problem in \mathcal{M}_X . By Proposition 4A.3 the undecidability of the λ -definability follows.

4A.4. DEFINITION (Word rewriting problem). Let $\Sigma = \{A, B\}$, a two letter alphabet.

- (i) A *word* (over Σ) is a finite sequence of letters $w_1 \cdots w_n$ with $w_i \in \Sigma$. The set of words over Σ is denoted by Σ^* .
- (ii) If $w = w_1 \cdots w_n$, then $\text{1th}(w) = n$ is called the length of w . If $\text{1th}(w) = 0$, 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* from u 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 *\mathcal{R} -derivable* from u , notation $u \vdash_{\mathcal{R}} s$, if it has a derivation.

4A.5. EXAMPLE. Consider the word AB and the rule $AB \hookrightarrow AABB$. Then $AB \vdash AAABB$, but $AB \not\vdash AAB$.

We will need the following well-known result, see e.g. Post [1947].

4A.6. THEOREM. *There is a word $u_0 \in \Sigma^*$ and a finite set of rewrite rules R such that $\{u \in \Sigma^* \mid u_0 \vdash_R u\}$ is undecidable.*

4A.7. DEFINITION. Given the alphabet $\Sigma = \{A, B\}$, define the set

$$X = X_\Sigma \triangleq \{A, B, *, L, R, Y, N\}.$$

The objects L and R are suggested to be read *left* and *right* and Y and N *yes* and *no*. In 4A.8-4A.21 we write \mathcal{M} for the full type structure \mathcal{M}_X built over the set X .

4A.8. DEFINITION. [Word encoding] Let $n > 0$ and $1_n = 0^n \rightarrow 0$ and $MN^{\sim n} \equiv MN \cdots N$, with n times the same term N . Let $w = w_1 \cdots w_n$ be a word of length n .

(i) The word w is *encoded* as the object $\underline{w} \in \mathcal{M}(1_n)$ defined as follows.

$$\begin{aligned}\underline{w}(*^{\sim(i-1)}, w_i, *^{\sim(n-i)}) &\triangleq Y; \\ \underline{w}(*^{\sim(i-1)}, L, R, *^{\sim(n-i-1)}) &\triangleq Y; \\ \underline{w}(x_1, \dots, x_n) &\triangleq N, \quad \text{otherwise.}\end{aligned}$$

(ii) The word w is *weakly encoded* by an object $h \in \mathcal{M}(1_n)$ if

$$\begin{aligned}h(*^{\sim(i-1)}, w_i, *^{\sim(n-i)}) &= Y; \\ h(*^{\sim(i-1)}, L, R, *^{\sim(n-i-1)}) &= Y.\end{aligned}$$

4A.9. DEFINITION. (Encoding of a rule) In order to define the encoding of a rule we use the notation $(a_1 \cdots a_k \mapsto Y)$ to denote the element $h \in \mathcal{M}(1_k)$ defined by

$$\begin{aligned}ha_1 \cdots a_k &\triangleq Y; \\ hx_1 \cdots x_k &\triangleq N, \quad \text{otherwise.}\end{aligned}$$

Now a rule $v \hookrightarrow w$ where $\text{1th}(v) = m$ and $\text{1th}(w) = n$ is encoded as the object $\underline{v \hookrightarrow w} \in \mathcal{M}(1_m \rightarrow 1_n)$ defined as follows.

$$\begin{aligned}\underline{v \hookrightarrow w}(v) &\triangleq \underline{w}; \\ \underline{v \hookrightarrow w}(*^{\sim m} \mapsto Y) &\triangleq (*^{\sim n} \mapsto Y); \\ \underline{v \hookrightarrow w}(R *^{\sim(m-1)} \mapsto Y) &\triangleq (R *^{\sim(n-1)} \mapsto Y); \\ \underline{v \hookrightarrow w}(*^{\sim(m-1)} L \mapsto Y) &\triangleq (*^{\sim(n-1)} L \mapsto Y); \\ \underline{v \hookrightarrow w}(h) &\triangleq \lambda x_1 \cdots x_n. N, \quad \text{otherwise.}\end{aligned}$$

As usual we identify a term $M \in \Lambda(A)$ with its denotation $\llbracket M \rrbracket \in X(A)$.

4A.10. LEMMA. Let s, u be two words over Σ and let $v \hookrightarrow w$ be a rule. Let the lengths of the words s, u, v, w be p, q, m, n , respectively. Then $svu \vdash swu$ and

$$swu \vec{s} \vec{w} \vec{u} = (\underline{v \hookrightarrow w}(\lambda \vec{v}. \underline{svu} \vec{s} \vec{v} \vec{u})) \vec{w}, \tag{1}$$

where $\vec{s}, \vec{u}, \vec{v}, \vec{w}$ are sequences of elements in X with lengths p, q, m, n , respectively.

PROOF. The RHS of (1) is obviously either Y or N . Now $\text{RHS} = Y$

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 $\vec{w} = *^{\sim(i-1)} L R *^{\sim(n-i-1)}$
- $\lambda \vec{v}. \underline{svu} \vec{s} \vec{v} \vec{u} = (*^{\sim m} \mapsto Y)$ 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 $\vec{w} = *^{\sim(n-1)} L$
- $\vec{s} = *^{\sim(p-1)} L, \vec{u} = *^{\sim q}$ 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
- $\vec{s} \vec{w} \vec{u} = * \dots * LR * \dots *$

iff $\underline{swu} \vec{s} \vec{w} \vec{u} = Y$. ■

4A.11. PROPOSITION. Let $\mathcal{R} = \{R_1, \dots, R_r\}$ be a set of rules. Then

$$u \vdash_{\mathcal{R}} s \Rightarrow \exists F \in \Lambda^{\emptyset} \underline{s} = Fu \underline{R}_1 \dots \underline{R}_r.$$

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 s , 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.

4A.12. CONVENTION. For the rest of this subsection we consider a fixed word W and set of rewrite rules $\mathcal{R} = \{R_1, \dots, R_k\}$ with $R_i = V_i \hookrightarrow W_i$. Moreover we let w, r_1, \dots, r_k be variables of the types of $\underline{W}, \underline{R}_1, \dots, \underline{R}_k$ respectively. Finally ρ is a valuation such that $\rho(w) = \underline{W}$, $\rho(r_i) = \underline{R}_i$ and $\rho(x^0) = *$ for all variables of type 0.

The first lemma classifies the terms M in lnf that denote a weak encoding of a word.

4A.13. LEMMA. Let M be a long normal form with $\text{FV}(M) \subseteq \{w, r_1, \dots, r_k\}$. Suppose $\llbracket M \rrbracket_{\rho} = \underline{V}$, 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 $\vec{x}, \vec{x}_1, \vec{y}:0$ are variables and the \vec{x}_1 are distinct elements of the \vec{x} .

PROOF. Since $\llbracket M \rrbracket_{\rho}$ 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 0. 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 weak word 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 weak word 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 $\lambda \vec{y}. N$,
- $\llbracket \lambda \vec{y}. N \rrbracket_\rho = \underline{v}_i$.
- and none of the variables \vec{x}_1 is the variable x_n .

4A.14. LEMMA. *Let M with $\text{FV}(M) \subseteq \{w, r_1, \dots, r_k, x_1, \dots, x_p\}$, with $\vec{x}:0$ be a lnf of type 0 that is not a variable. If $x_1 \in \text{FV}(M)$ and there is a valuation φ such that $\varphi(x_1) = A$ or $\varphi(x_1) = B$ and $\llbracket M \rrbracket_\varphi = Y$, then $\varphi(y) = *$, for all other variables $y:0$ in $\text{FV}(M)$.*

PROOF. By induction on the structure of M .

Case $M \equiv w P_1 \cdots P_n$. Then the terms P_1, \dots, P_n must all be variables. Otherwise, some P_j would have as head variable one of w, r_1, \dots, r_k , and $\llbracket P_j \rrbracket_\varphi$ would be Y or N . Then $\llbracket M \rrbracket_\varphi$ would be N , quod non. 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 .

Case $M = r_i(\lambda \vec{w}. Q) \vec{P}$. As above, the terms \vec{P} must all be variables. If some P_j is equal to x_1 , then $\llbracket \lambda \vec{w}. Q \rrbracket_\varphi$ is the word v_i . So Q is not a variable and all the other variables in \vec{P} denote $*$. Let l be the first letter of v_i . We have $\llbracket \lambda \vec{w}. Q \rrbracket_\varphi l * \cdots * = Y$ and hence

$$\llbracket Q \rrbracket_{\varphi \cup \{\langle w_1, l \rangle, \langle w_2, * \rangle, \dots, \langle w_m, * \rangle\}} = Y.$$

By induction hypothesis it follows that $\varphi \cup \{\langle w_1, l \rangle, \langle w_2, * \rangle, \dots, \langle w_m, * \rangle\}$ takes the value $*$ on all free variables of Q , except for w_1 . Hence φ takes the value $*$ on all free variables of $\lambda \vec{w}. Q$. Therefore φ takes the value $*$ on all free variables of M , except for x_1 .

If none of the \vec{P} is x_1 , then $x_1 \in \text{FV}(\lambda \vec{w}. Q)$. Since $\llbracket r_i(\lambda \vec{w}. Q) \vec{P} \rrbracket_\varphi = Y$, it follows that $\llbracket \lambda \vec{w}. Q \rrbracket_\varphi$ is not the constant function equal to N . Hence there are objects a_1, \dots, a_m such that $\llbracket \lambda \vec{w}. Q \rrbracket_\varphi(a_1) \cdots (a_m) = Y$. Therefore

$$\llbracket Q \rrbracket_{\varphi \cup \{\langle w_1, a_1 \rangle, \dots, \langle w_m, a_m \rangle\}} = Y.$$

By the induction hypothesis $\varphi \cup \{\langle w_1, a_1 \rangle, \dots, \langle w_m, a_m \rangle\}$ takes the value $*$ on all the variables free in Q , except for x_1 . So φ takes the value $*$ on all the variables free in $\lambda \vec{w}. Q$, except for x_1 . Moreover $a_1 = \cdots = a_m = *$, and thus $\llbracket \lambda \vec{w}. Q \rrbracket_\varphi * \cdots * = Y$. Therefore the function $\llbracket \lambda \vec{w}. Q \rrbracket_\varphi$ can only be the function mapping $* \cdots *$ to Y and the other values to N . Hence $\llbracket r_i(\lambda \vec{w}. Q) \rrbracket_\varphi$ is the function mapping $* \cdots *$ to Y and the other values to N and φ takes the value $*$ on \vec{P} . Therefore φ takes the value $*$ on all free variables of M except for x_1 . ■

4A.15. LEMMA. *If the term $M = \lambda \vec{x}. (r_i(\lambda \vec{w}. Q) \vec{y})$ denotes a weak word encoding, then the variables \vec{y} do not occur free in $\lambda \vec{w}. Q$ and $\llbracket \lambda \vec{w}. Q \rrbracket_{\varphi_0}$ is the encoding of the word v_i .*

PROOF. Consider a variable y_j . This variable, say, x_h . Let l be the h^{th} letter of the word w' , we have

$$\llbracket M \rrbracket *^{\sim(h-1)} l *^{\sim(k-h)} = Y$$

Let $\varphi = \varphi_0 \cup \{(x_h, l)\}$. We have

$$r_i(\llbracket \lambda \vec{w}.Q \rrbracket_\varphi) *^{\sim(j-1)} l *^{\sim(m-j)} = Y$$

Hence $\llbracket \lambda \vec{w}.Q \rrbracket_\varphi$ is the encoding of the word v_i . Let l' be the first letter of this word, we have

$$\llbracket \lambda \vec{w}.Q \rrbracket_\varphi(l') * \cdots * = Y$$

and hence

$$\llbracket Q \rrbracket_{\varphi \cup \{(w_1, l'), (w_2, *), \dots, (w_m, *)\}} = Y$$

By Lemma 4A.14, $\varphi \cup \{(w_1, l'), (w_2, *), \dots, (w_m, *)\}$ takes the value $*$ on all variables free in Q except w_1 . Hence y_j is not free in Q nor in $\lambda \vec{w}.Q$.

At last $\llbracket \lambda \vec{w}.Q \rrbracket_\varphi$ is the encoding of v_i and y_j does not occur in it. Thus $\llbracket \lambda \vec{w}.Q \rrbracket_{\varphi_0}$ is the encoding of v_i . ■

4A.16. LEMMA. *Let M be a term of type 0 with $\text{FV}(M) \subseteq \{w, r_1, \dots, r_r, x_1, \dots, x_n\}$ and $\vec{x}:0$ that is not a variable. Then there is a variable x such that*

either $\varphi(z) = L \Rightarrow \llbracket M \rrbracket_\varphi = N$, for all valuations φ ,
or $\varphi(z) \in \{A, B\} \Rightarrow \llbracket M \rrbracket_\varphi = N$, for all valuations φ .

PROOF. By induction on the structure of M .

Case $M \equiv w \vec{P}$. Then the terms $\vec{P} = t_1, \dots, t_n$ must be variables. Take $z = P_n$. Then $\varphi(z) = L$ implies $\llbracket M \rrbracket_\varphi = N$.

Case $M \equiv r_i(\lambda \vec{w}.Q) \vec{P}$. By induction hypothesis, there is a variable z' free in Q , such that

$$\forall \varphi [\varphi(z') = L \Rightarrow \llbracket M \rrbracket_\varphi = N]$$

or

$$\forall \varphi [\varphi(z') = A \vee \varphi(z') = B \Rightarrow \llbracket M \rrbracket_\varphi = N].$$

If the variable z' is not among w_1, \dots, w_n we take $z = z'$. Either for all valuations such that $\varphi(z) = L$, $\llbracket \lambda \vec{w}.Q \rrbracket_\varphi$ is the constant function equal to N and thus $\llbracket M \rrbracket_\varphi = N$, or for all valuations such that $\varphi(z) = A$ or $\varphi(z) = B$, $\llbracket \lambda \vec{w}.Q \rrbracket_\varphi$ is the constant function equal to N and thus $\llbracket M \rrbracket_\varphi = N$.

If the variable $z' = w_j$ ($j \leq m-1$), then for all valuations $\llbracket \lambda \vec{w}.Q \rrbracket_\varphi$ 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, $\llbracket \lambda \vec{w}.Q \rrbracket_\varphi$ is not the encoding of the word v_i and hence $\llbracket r_i(\lambda \vec{w}.Q) \rrbracket_\varphi$ is either the function mapping $* \cdots *$ to Y and other arguments to N , the function mapping $R * \cdots *$ to Y and other arguments to N , the function mapping $* \cdots * L$ to Y and other arguments to N or the function mapping all arguments to N . We take $z = P_n$ and for all valuations such that $\varphi(z) = A$ or $\varphi(z) = B$ we have $\llbracket M \rrbracket_\varphi = N$.

At last if $z' = w_m$, then for all valuations $\llbracket \lambda \vec{w}.Q \rrbracket_\varphi$ is a function taking the value N when applied to any sequence of arguments whose m^{th} element is L or for all valuations $\llbracket \lambda \vec{w}.Q \rrbracket_\varphi$ 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, $\llbracket \lambda \vec{w}.Q \rrbracket_\varphi$ is not the function mapping $* \cdots * L$ to Y and other arguments to N . Hence $\llbracket r_i(\lambda \vec{w}.Q) \rrbracket_\varphi$ is either w_i or the function mapping $* \cdots *$ to Y and other arguments to N the function mapping $R * \cdots *$ to Y and other arguments to N or the function mapping all arguments to N . We take $z = P_n$ and for all valuations such that $\varphi(z) = A$ or $\varphi(z) = B$ we have $\llbracket M \rrbracket_\varphi = N$.

In the second case, for all valuations, $\llbracket \lambda \vec{w}.Q \rrbracket_\varphi$ is not the encoding of the word v_i . Hence $\llbracket r_i(\lambda \vec{w}.Q) \rrbracket_\varphi$ is either the function mapping $* \cdots *$ to Y and other arguments to N the function mapping $R * \cdots *$ to Y and other arguments to N , the function mapping $* \cdots * L$ to Y and other arguments to N or the function mapping all arguments to N . We take $z = P_n$ and for all valuations such that $\varphi(z) = L$ we have $\llbracket M \rrbracket_\varphi = N$. ■

4A.17. LEMMA. *If the term $M = \lambda \vec{x}.r_i(\lambda \vec{w}.Q)\vec{y}$ denotes a weak word encoding, then none of the variables \vec{y} is the variable x_n , where $\vec{x} = x_1, \dots, x_n$.*

PROOF. By the Lemma 4A.16, we know that there is a variable z such that either for all valuations satisfying $\varphi(z) = L$ we have

$$\llbracket r_i(\lambda \vec{w}.Q)\vec{y} \rrbracket_\varphi = N,$$

or for all valuations satisfying $\varphi(z) = A$ or $\varphi(z) = B$ we have

$$\llbracket r_i(\lambda \vec{w}.Q)\vec{y} \rrbracket_\varphi = N.$$

Since M denotes a weak word encoding, the only possibility is that $z = x_n$ and for all valuations such that $\varphi(x_n) = L$ we have

$$\llbracket r_i(\lambda \vec{w}.Q)\vec{y} \rrbracket_\varphi = N.$$

Now, if y_j were equal to x_n and y_{j+1} to some x_h , then the object

$$\llbracket r_i(\lambda \vec{w}.Q)\vec{y} \rrbracket_{\varphi_0 \cup \{(x_n, L), (x_h, R)\}}$$

would be equal to $r_i(\llbracket \lambda \vec{w}.Q \rrbracket_{\varphi_0}) * \cdots * LR * \cdots *$ and, as $\llbracket \lambda \vec{w}.Q \rrbracket_{\varphi_0}$ is the encoding of the word v_i , also to Y . This is a contradiction. ■

We are now ready to conclude the proof.

4A.18. PROPOSITION. *If M is a lnf, with $\text{FV}(M) \subseteq \{w, r_1, \dots, r_k\}$, that denotes a weak word encoding w' , then w' is derivable.*

PROOF. Case $M = \lambda \vec{x}.w\vec{y}$. Then, as M denotes a weak word encoding, it depends on all its arguments and thus all the variables x_1, \dots, x_n are among \vec{y} . Since the \vec{y} are distinct, \vec{y} is a permutation of x_1, \dots, x_n . As M denotes a weak word encoding, one has $\llbracket M \rrbracket * \cdots * LR * \cdots * = Y$. Hence this permutation is the identity and

$$M = \lambda \vec{x}.(w\vec{x}).$$

The word w' is the word w and hence it is derivable.

Case $M = \lambda \vec{x}.r_i(\lambda \vec{w}.Q)\vec{y}$. We know that $\llbracket \lambda \vec{w}.Q \rrbracket_{\varphi_0}$ is the encoding of the word v_i and thus $\llbracket r_i(\lambda \vec{w}.Q) \rrbracket_{\varphi_0}$ is the encoding of the word w_i . Since M denotes a weak word encoding, one has $\llbracket M \rrbracket * \cdots * LR * \cdots * = Y$. If some y_j ($j \leq n - 1$) is, say, x_h then, by Lemma 4A.17, $h \neq k$ and thus $\llbracket M \rrbracket *^{\sim(h-1)} LR *^{\sim(k-h-1)} = Y$ and $y_{i+1} = x_{h+1}$. Hence $\vec{y} = x_{p+1}, \dots, x_{p+l}$. Rename the variables x_1, \dots, x_p as \vec{x}' and x_{p+l+1}, \dots, x_l as $\vec{z} = z_1, \dots, z_q$. Then

$$M = \lambda \vec{x}' \vec{y} \vec{z}.r_i(\lambda \vec{w}.Q)\vec{y}.$$

Write $w' = u_1 w u_2$, where u_1 has length p , w length l and u_2 length q .

The variables \vec{y} are not free in $\lambda \vec{w}.Q$, hence the term $\lambda \vec{x}' \vec{w} \vec{z}.Q$ 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

$$\llbracket \lambda \vec{x}' \vec{y} \vec{z}. r_i(\lambda \vec{w}. Q) \vec{y} \rrbracket *^{\sim(j-1)} l *^{\sim(p-j+l+q)} = Y.$$

Let $\varphi = \varphi_0 \cup \{\langle x_j, l \rangle\}$. The function $\llbracket r_i(\lambda \vec{w}. Q) \rrbracket_\varphi$ maps $* \dots *$ to Y . Hence, the function $\llbracket \lambda \vec{w}. Q \rrbracket_\varphi$ maps $* \dots *$ to Y and other arguments to N . Hence

$$\llbracket \lambda \vec{x}' \vec{w} \vec{z}. Q \rrbracket *^{\sim(j-1)} l *^{\sim(p-j+m+q)} = Y.$$

- We know that $\llbracket \lambda \vec{w}. Q \rrbracket_{\varphi_0}$ is the encoding of the word v_i . Hence if l is the j^{th} letter of the word v_i , then

$$\llbracket \lambda \vec{x}' \vec{w} \vec{z}. Q \rrbracket *^{\sim(p+j-1)} l *^{\sim(l-j+q)} = Y.$$

- In a way similar to the first case, we prove that if l is the j^{th} letter of u_2 . We have

$$\llbracket \lambda \vec{x}' \vec{w} \vec{z}. Q \rrbracket *^{\sim(p+m+j-1)} l *^{\sim(q-j)} = Y.$$

- Second clause.

- If $j \leq p-1$, we have

$$\llbracket \lambda \vec{x}' \vec{y} \vec{z}. r_i(\lambda \vec{w}. Q) \vec{y} \rrbracket *^{\sim(j-1)} LR *^{\sim(p-j-1+m+q)} = Y.$$

Let φ be φ_0 but x_j to L and x_{j+1} to R . The function $\llbracket r_i(\lambda \vec{w}. Q) \rrbracket_\varphi$ maps $* \dots *$ to Y . Hence, the function $\llbracket \lambda \vec{w}. Q \rrbracket_\varphi$ maps $* \dots *$ to Y and other arguments to N and

$$\llbracket \lambda \vec{x}' \vec{w} \vec{z}. Q \rrbracket *^{\sim(j-1)} LR *^{\sim(p-j-1+m+q)} = Y.$$

- We have

$$\llbracket \lambda \vec{x}' \vec{y} \vec{z}. (r_i(\lambda \vec{w}. Q) \vec{y}) \rrbracket *^{\sim(p-1)} LR *^{\sim(l-1+q)} = Y.$$

Let φ be φ_0 but x_p to L . The function $\llbracket r_i(\lambda \vec{w}. Q) \rrbracket_\varphi$ maps $R * \dots *$ to Y . Hence, the function $\llbracket \lambda \vec{w}. Q \rrbracket_\varphi$ maps $R * \dots *$ to Y and other arguments to N and

$$\llbracket \lambda \vec{x}' \vec{w} \vec{z}. Q \rrbracket *^{\sim(p-1)} LR *^{\sim(m-1+q)} = Y.$$

- We know that $\llbracket \lambda \vec{w}. Q \rrbracket_{\varphi_0}$ is the encoding of the word v_i . Hence if $j \leq m-1$ then

$$\llbracket \lambda \vec{x}' \vec{w} \vec{z}. Q \rrbracket *^{\sim(p+j-1)} LR *^{\sim(m-j-1+q)} = Y.$$

- In a way similar to the second, we prove that

$$\llbracket \lambda \vec{x}' \vec{w} \vec{z}. Q \rrbracket *^{\sim(p+m-1)} LR *^{\sim(q-1)} = Y.$$

- In a way similar to the first, we prove that if $j \leq q-1$, we have

$$\llbracket \lambda \vec{x}' \vec{w} \vec{z}. Q \rrbracket *^{\sim(p+m+j-1)} LR *^{\sim(q-j-1)} = Y.$$

Hence the term $\lambda \vec{x}' \vec{w} \vec{z}. Q$ denotes a weak encoding of the word $u_1 v_i u_2$. By induction hypothesis, the word $u_1 v_i u_2$ is derivable and hence $u_1 w_i u_2$ is derivable.

At last we prove that $w = w_i$, i.e. that $w' = u_1 w_i u_2$. We know that $\llbracket r_i(\lambda \vec{w}. Q) \rrbracket_{\varphi_0}$ is the encoding of the word w_i . Hence

$$\llbracket \lambda \vec{x}' \vec{y} \vec{z}. r_i(\lambda \vec{w}. Q) \vec{y} \rrbracket *^{\sim(p+j-1)} l *^{\sim(l-j+q)} = Y$$

iff l is the j^{th} letter of the word w_i .

Since $\llbracket \lambda \vec{x}' \vec{y} \vec{z}. r_i(\lambda \vec{w}. Q) \vec{y} \rrbracket$ is a weak encoding of the word $u_1 w_i u_2$, if l is the j^{th} letter of the word w , we have

$$\llbracket \lambda \vec{x}' \vec{y} \vec{z}. r_i(\lambda \vec{w}. Q) \vec{y} \rrbracket *^{\sim(p+j-1)} l *^{\sim(l-j+q)} = Y$$

and l is the j^{th} letter of the word w_i . Hence $w = w_i$ and $w' = u_1 w_i u_2$ is derivable. ■

From Proposition 4A.11 and 4A.18, we conclude.

4A.19. PROPOSITION. *The word w' is derivable iff there is a term whose free variables are among w, r_1, \dots, r_k that denotes the encoding of w' .*

4A.20. COROLLARY. *Let w and w' be two words and $v_1 \hookrightarrow w_1, \dots, v_r \hookrightarrow w_r$ be rewrite rules. Let h be the encoding of w , h' be the encoding of w' , r_1 be the encoding of $v_1 \hookrightarrow w_1, \dots$, and r_k be the encoding of $v_r \hookrightarrow w_r$.*

Then the word w' is derivable from w with the rules $v_1 \hookrightarrow w_1, \dots, v_r \hookrightarrow w_r$ iff there is a definable function that maps h, r_1, \dots, r_k to h' .

The following result was proved by Ralph Loader 1993 and published in [Loader \[2001b\]](#).

4A.21. THEOREM (Loader). *λ -definability is undecidable, 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, r_1, \dots, r_k to h' and hence if w' is derivable from w with the rules $v_1 \hookrightarrow w_1, \dots, v_r \hookrightarrow w_r$ contradicting the undecidability of the word rewriting problem. ■

Joly has extended Loader's result in two directions as follows. Let $\mathcal{M}_n = \mathcal{M}_{\{0, \dots, n-1\}}$. Define for $n \in \mathbb{N}, A \in \mathbb{T}, d \in \mathcal{M}_n(A)$

$$D(n, A, d) \stackrel{\Delta}{\iff} d \text{ is } \lambda\text{-definable in } \mathcal{M}_n.$$

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 following.

4A.22. PROPOSITION. *Undecidability of λ -definability is monotonic in the following sense.*

$$\forall Ad. D(n_0, A, d) \text{ undecidable} \& n_0 \leq n_1 \Rightarrow \forall Ad. D(n_1, A, d) \text{ undecidable.}$$

PROOF. Use Exercise 3F.24(i). ■

Loader's proof above shows in fact that $\forall Ad. D(7, A, d)$ is undecidable. It was sharpened in [Loader \[2001a\]](#) showing that $\forall Ad. D(3, A, d)$ is undecidable. The ultimate sharpening in this direction is proved in [Joly \[2005\]](#): $\forall Ad. D(2, A, d)$ is undecidable.

Going in a different direction one also has the following.

4A.23. THEOREM (Joly [2005]). *$\forall nd. D(n, 3 \rightarrow 0 \rightarrow 0, d)$ is undecidable.*

Loosely speaking one can say that λ -definability at the monster type $M = 3 \rightarrow 0 \rightarrow 0$ is undecidable. Moreover, Joly also has characterized those types A that are undecidable in this sense.

4A.24. DEFINITION. A type A is called *finitely generated* if there are closed terms M_1, \dots, M_n , not necessarily of type A such that every closed term of type A is an applicative product of the M_1, \dots, M_n .

4A.25. THEOREM (Joly [2002]). *Let $A \in \mathbb{T}$. Then $\lambda\text{nd}.D(n, A, d)$ is decidable iff the closed terms of type A can be finitely generated.*

For a sketch of the proof see Exercise 3F.36.

4A.26. COROLLARY. *The monster type $M = 3 \rightarrow 0 \rightarrow 0$ is not finitely generated.*

PROOF. By Theorems 4A.25 and 4A.23. ■

4B. Undecidability of unification

The notion of (higher-order¹¹) unification and matching problems were introduced by Huet [1975]. In that paper it was proved that unification in general is undecidable. Moreover the question was asked whether matching is (un)decidable.

4B.1. DEFINITION. (i) Let $M, N \in \Lambda^\circ(A \rightarrow B)$. A *pure unification problem* is of the form

$$\exists X:A.MX = NX,$$

where one searches for an $X \in \Lambda^\circ(A)$ (and the equality is $=_{\beta\eta}$). A is called the *search-type* and B the *output-type* of the problem.

(ii) Let $M \in \Lambda^\circ(A \rightarrow B), N \in \Lambda^\circ(B)$. A *pure matching problem* is of the form

$$\exists X:A.MX = N,$$

where one searches for an $X \in \Lambda^\circ(A)$. 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

$$MX = (\lambda x.N)X.$$

4B.2. DEFINITION. The *rank (order)* of a unification or matching problem is $\text{rk}(A)$ ($\text{ord}(A)$ respectively), where A is the search-type. Remember that $\text{ord}(A) = \text{rk}(A) + 1$.

¹¹By contrast to the situation in 2C.11 the present form of unification is ‘higher-order’, because it asks whether functions exist that satisfy certain equations.

The rank of the output-type is less relevant. Basically one may assume that it is $\top = 1_2 \rightarrow 0 \rightarrow 0$. Indeed, by the Reducibility Theorem 3D.8 one has $\Phi : B \leq_{\beta\eta} \top$, for some closed term Φ . Then

$$MX = NX : B \Leftrightarrow (\Phi \circ M)X = (\Phi \circ N)X : \top.$$

One has $\text{rk}(\top) = 2$. The unification and matching problems with an output type of rank < 2 are decidable, see Exercise 4E.6.

The main results of this Section are that unification in general is undecidable from a low level onward, [Goldfarb \[1981\]](#), and matching up to order 4 is decidable, [Padovani \[2000\]](#).

In [Stirling \[2009\]](#) it is shown that matching in general is decidable. The paper is too recent and complex to be included here.

As a spin-off of the study of matching problems it will be shown that the maximal theory is decidable.

4B.3. EXAMPLE. The following are two examples of pure unification problems.

- (i) $\exists X:(1 \rightarrow 0).\lambda f:1.f(Xf) = X$.
- (ii) $\exists X:(1 \rightarrow 0 \rightarrow 0).\lambda fa.X(Xf)a = \lambda fa.Xf(Xfa)$.

This is not in the format of the previous Definition, but we mean of course

$$\begin{aligned} (\lambda x:(1 \rightarrow 0)\lambda f:1.f(xf))X &= (\lambda x:(1 \rightarrow 0)\lambda f:1.xf)X; \\ (\lambda x:(1 \rightarrow 0 \rightarrow 0)\lambda f:1\lambda a:0.x(xf)a)X &= (\lambda x:(1 \rightarrow 0 \rightarrow 0)\lambda f:1\lambda a:0.xf(xfa))X. \end{aligned}$$

The most understandable form is as follows (provided we remember the types)

$$\begin{aligned} (i) \quad \lambda f.f(Xf) &= X; \\ (ii) \quad X(Xf)a &= Xf(Xfa). \end{aligned}$$

The first problem has no solution, because there is no fixed point combinator in λ^0_\rightarrow . The second one does $(\lambda fa.f(fa))$ and $(\lambda fa.a)$, because $n^2 = 2n$ for $n \in \{2, 4\}$.

4B.4. EXAMPLE. The following are two pure matching problems.

$$\begin{aligned} X(Xf)a &= f^{10}a & X:1 \rightarrow 0 \rightarrow 0; f:1, a:0; \\ f(X(Xf)a) &= f^{10}a & X:1 \rightarrow 0 \rightarrow 0; f:1, a:0. \end{aligned}$$

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.

4B.5. DEFINITION. (i) Let M, N be closed terms of the same type. A pure [*unification problem with several unknowns*](#)

$$M\vec{X} =_{\beta\eta} N\vec{X} \tag{1}$$

searches for closed terms \vec{X} of the right type satisfying (1). The rank of a problem with several unknowns \vec{X} is

$$\max\{\text{rk}(A_i) \mid 1 \leq i \leq n\},$$

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, \dots, M_n and N_1, \dots, N_n such that M_i, N_i are of the same type for $1 \leq i \leq n$. searching for closed terms $\vec{X}_1, \dots, \vec{X}_n$ all occuring among \vec{X} such that

$$\begin{array}{ll} M_1\vec{X}_1 & =_{\beta\eta} N_1\vec{X}_1 \\ & \dots \\ M_n\vec{X}_n & =_{\beta\eta} N_n\vec{X}_n \end{array}$$

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 Λ^\varnothing . 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.

4B.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 \[1974\]](#).

$$\begin{array}{l} P_1(Pxy) = x \\ P_2(Pxy) = y. \end{array}$$

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 \[1974\]](#).

4B.7. EXAMPLE. An example of a unification problem with constants from $\Gamma = \{a:1, b:1\}$ is the following. We search for unknowns $W, X, Y, Z \in \Lambda^\Gamma(1)$ such that

$$\begin{aligned} X &= Y \circ W \circ Y \\ b \circ W &= W \circ b \\ W \circ W &= b \circ W \circ b \\ a \circ Y &= Y \circ a \\ X \circ X &= Z \circ b \circ b \circ a \circ a \circ b \circ b \circ Z, \end{aligned}$$

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 4E.5.

4B.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 unknowns can be reduced to a problem with one unknown with rank $\text{rk}(A) = \max\{k, 2\}$.*

(ii) *Systems of problems can be reduced to one problem, without altering the rank. The rank of the output type will be $\max\{\text{rk}(B_i), 2\}$, 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*

$$\max\{\text{rk}(\Gamma), k\}.$$

PROOF. We give the proof for unification.

(i) Following Notation 1D.23 we have

$$\begin{aligned} \exists \vec{X}. M \vec{X} &= N \vec{X} & (1) \\ \Leftrightarrow \exists X. (\lambda x. M(x \cdot 1) \cdots (x \cdot n)) X &= (\lambda x. N(x \cdot 1) \cdots (x \cdot n)) X. & (2) \end{aligned}$$

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 $\vec{X} \equiv X \cdot 1, \dots, X \cdot n$ work for (1). By Proposition 1D.22 we have $A = A_1 \times \cdots \times A_n$ is the type of X and $\text{rk}(A) = \max\{\text{rk}(A_1), \dots, \text{rk}(A_n), 2\}$.

(ii) Similarly for $\vec{X}_1, \dots, \vec{X}_n$ being subsequences of \vec{X} one has

$$\begin{aligned} \exists \vec{X} \quad M_1 \vec{X}_1 &= N_1 \vec{X}_1 & (1) \\ &\dots \\ M_n \vec{X}_n &= N_n \vec{X}_n & (2) \\ \Leftrightarrow \exists \vec{X} \quad (\lambda \vec{x}. \langle M_1 \vec{x}_1, \dots, M_n \vec{x}_n \rangle) \vec{X} &= (\lambda \vec{x}. \langle N_1 \vec{x}_1, \dots, N_n \vec{x}_n \rangle) \vec{X}. \end{aligned}$$

(iii) Write a non-pure problem with $M, N \in \Lambda^\Gamma(A \rightarrow B)$, and $\text{dom}(\Gamma) = \{\vec{y}\}$ 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

$$\exists X: (\bigwedge \Gamma \rightarrow A). (\lambda xy. M[\vec{y}](xy)) X = (\lambda xy. N[\vec{y}](xy)) X. \blacksquare$$

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

4B.9. PROPOSITION. *Unification with unknowns of type 1 and constants of types 0, 1 is decidable.*

PROOF. The essential work to be done is the solvability of Markov’s problem by Makanin. See Exercise 4E.5 for the connection and a reference. ■

In Statman [1981] it is shown that the set of (bit strings encoding) decidable unification problems is itself polynomial time decidable

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 Matijasevič makes things more easy.

4B.10. THEOREM (Matijasevič). (i) *There are two polynomials p_1, p_2 over \mathbb{N} (of degree 7 with 13 variables¹²) 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 [Matiyasevič \[1972\]](#), [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

$$n \in \mathbb{N} \Leftrightarrow \exists a, b, c, d \in \mathbb{N}. n = a^2 + b^2 + c^2 + d^2.$$

Finally write $\exists x \in \mathbb{N}. p(x, \dots) = 0$ as $\exists a, b, c, d \in \mathbb{Z}. p(a^2 + b^2 + c^2 + d^2, \dots) = 0$. ■

4B.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 1D.7. Let $U_{\vec{n}} = \{M_{p_1}[\vec{n}] \vec{x} = M_{p_2}[\vec{n}] \vec{x}\}$. Using that every $X \in \Lambda^\phi(\text{Nat})$ is a numeral, Proposition 2A.16, it follows that this unification problem is solvable iff $\vec{n} \in D$. ■

The construction of Matijasevič 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 it needs to use unification at variable types. There is a way out. In [Davis, Robinson, and Putnam \[1961\]](#) it is proved that every RE predicate is of the form $\exists \vec{x} \forall y_1 < t_1 \dots \forall y_n < t_n. p_1 = p_2$. Using this result and higher types (Nat_A , for some non-atomic A) one can get rid of the bounded quantifiers. The analogon of Proposition 2A.16 ($X:\text{Nat} \Rightarrow X$ a numeral) does not hold but one can filter out the ‘numerals’ by a unification (with $f:A \rightarrow A$):

$$f \circ (Xf) = (Xf) \circ f.$$

This yields without Matijasevič's theorem the undecidability of unification with the unknown of a fixed type.

4B.12. THEOREM. *Unification of order 2 (rank 1) with constants is undecidable.*

PROOF. See Exercise 4E.4. ■

This implies that pure unification of order 3 is undecidable, something we already saw in Corollary 4B.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 4B.11.

Another proof of the undecidability unification of order 2 with constants, not using Matijasevič's theorem, is in [Schubert \[1998\]](#).

¹²This can be pushed to polynomials of degree 4 and 58 variables or of degree $1.6 \cdot 10^{45}$ and 9 variables, see [Jones \[1982\]](#).

4C. 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 $=_{\beta\eta}$, which is the intended case, has been established in Stirling [2009], but will not be included here.

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.

4C.1. PROPOSITION. (i) *Matching with unknowns of rank 1 is NP-complete.*

(ii) *Pure matching of rank 2 is NP-complete.*

PROOF. (i) Consider $A = 0^2 \rightarrow 0 = \text{Bool}_0$. Using Theorem 2A.13, Proposition 1C.3 and Example 1C.8 it is easy to show that if $M \in \Lambda^\varnothing(A)$, then $M \in {}_{\beta\eta}\{\text{true}, \text{false}\}$. By Proposition 1D.2 a Boolean function $p(X_1, \dots, X_n)$ in the variables X_1, \dots, X_n is λ -definable by a term $M_p \in \Lambda^\varnothing(A^n \rightarrow A)$. Therefore

$$p \text{ is satisfiable} \Leftrightarrow M_p X_1 \cdots X_n = \text{true} \text{ is solvable.}$$

This is a matching problem of rank 1.

(ii) By (i) and Proposition 4B.8. ■

Following an idea of Statman [1982], the decidability of the matching problem can be reduced to the existence for every term N of a logical relation \parallel_N on terms $\lambda^0 \rightarrow$ such that

- \parallel_N is an equivalence relation;
- for all types A the quotient $\mathcal{T}_A / \parallel_N$ is finite;
- there is an algorithm that enumerates $\mathcal{T}_A / \parallel_N$, i.e. that takes in argument a type A 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 $\mathcal{T}_A / \parallel_N$ can be enumerated up to rank n .

The finite completeness theorem, Theorem 3D.33, yields the existence of a standard model \mathcal{M} such that the relation $\mathcal{M} \models M = N$ 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 0 that are $\beta\eta$ -equivalent and also all terms of type 0 that are not subterms of N . 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 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 in Comon and Jurski [1998].

Rank 3 interpolation problems

4C.2. DEFINITION. (i) An *interpolation equation* is a particular matching problem

$$X \vec{M} = N,$$

where M_1, \dots, M_n and N are closed terms. That is, the unknown X occurs at the head. A solution of such an equation is a term P such that

$$P \vec{M} =_{\beta\eta} N.$$

(ii) An *interpolation problem* is a conjunction of such equations with the same unknown. A solution of such a problem is a term P that is a solution for all the equations simultaneously.

(iii) 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 Inf . Hence, without loss of generality, we can restrict the search to such terms.

To prove the decidability of the rank 3 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 rank 3 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 $(0 \rightarrow 1 \rightarrow 0) \rightarrow 0$ has all the solutions

$$\lambda f.f(a(\lambda z_1.f(a(\lambda z_2.f(a \dots (\lambda z_n.f(z_1(\mathsf{K}(fz_2(\mathsf{K}(fz_3 \dots (fz_n(\mathsf{K}a)) \dots)) \dots)) \dots))).$$

Moreover since each z_i has z_1, \dots, 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

$$\lambda f.(f a \square)$$

where \square is a new constant of type $0 \rightarrow 0$. Moreover all the solutions above can be retrieved from this one by replacing the constant \square by an appropriate term (allowing captures in this replacement).

4C.3. DEFINITION. For each simple type A , we consider a constant \square_A . Let M be a term solution of an interpolation equation. A subterm occurrence of M of type A is *irrelevant* if replacing it by the constant \square_A yields a solution. A *relevant* solution is a closed solution where all irrelevant subterm occurrences are the constant \square_A .

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 $(1 \rightarrow 0 \rightarrow 0) \rightarrow 0$, the Church numeral $\mathbf{c}_1 \equiv \lambda f x. f x$ and a and h are constants of type 0 and 1_2 . A relevant solution of this equation substitutes X by the term $\lambda f. P$ where P is a relevant solution of the equation $P[f := \mathbf{c}_1] = ha$.

Let \mathcal{Q}_{ha} be the set of the relevant solutions P of the equation $P[f := \mathbf{c}_1] = ha$. More generally, let \mathcal{Q}_W be the set of relevant solutions P of the equation $P[f := \mathbf{c}_1] = W$.

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 \square_A . 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 (hP_1) and P_1 is in \mathcal{Q}_a or the form (fP_1P_2) and $(P_1[f := \mathbf{c}_1]P_2[f := \mathbf{c}_1]) = ha$. This means that there are terms P'_1 and P'_2 such that $P_1[f := \mathbf{c}_1] = P'_1$, $P_2[f := \mathbf{c}_1] = P'_2$ and $(P'_1P'_2) = ha$, in other words there are terms P'_1 and P'_2 such that P_1 is in $\mathcal{Q}_{P'_1}$, P_2 is in $\mathcal{Q}_{P'_2}$ and $(P'_1P'_2) = ha$. As $(P'_1P'_2) = ha$ there are three possibilities for P'_1 and P'_2 : $P'_1 = \mathbb{I}$ and $P'_2 = ha$, $P'_1 = \lambda z. hz$ and $P'_2 = a$ and $P'_1 = \lambda z. ha$ and $P'_2 = \square_o$. Hence (fP_1P_2) is in \mathcal{Q}_{ha} if either P_1 is in $\mathcal{Q}_\mathbb{I}$ and P_2 in \mathcal{Q}_{ha} or P_1 is in $\mathcal{Q}_{\lambda z. hz}$ and P_2 in \mathcal{Q}_a or P_1 is in $\mathcal{Q}_{\lambda z. ha}$ and $P_2 = \square_o$.

Hence, we have to analyze membership to \mathcal{Q}_a , $\mathcal{Q}_\mathbb{I}$, $\mathcal{Q}_{\lambda z. hz}$, $\mathcal{Q}_{\lambda z. ha}$.

- *analysis of membership to \mathcal{Q}_a*

A term is in \mathcal{Q}_a if it has either the form a or the form (fP_1P_2) and P_1 is in $\mathcal{Q}_\mathbb{I}$ and P_2 is in \mathcal{Q}_a or P_1 in $\mathcal{Q}_{\lambda z. a}$ and $P_2 = \square_o$.

Hence, we have to analyze membership to $\mathcal{Q}_{\lambda z. a}$,

- *analysis of membership to $\mathcal{Q}_\mathbb{I}$*

A term is in $\mathcal{Q}_\mathbb{I}$ if it has the form $\lambda z. P_1$ and P_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 $\lambda z. P_1$ and P_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 $\lambda z. P_1$ and P_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 $\lambda z. P_1$ and P_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 (fP_1P_2) and either P_1 is in \mathcal{Q}_l and P_2 is in \mathcal{Q}_z or P_1 is in $\mathcal{Q}_{\lambda z'.z}$ and $P_2 = \square_o$.

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 (hP_1) and P_1 is in \mathcal{Q}_z or the form (fP_1P_2) and either P_1 is in \mathcal{Q}_l and P_2 is in \mathcal{Q}_{hz} or P_1 is in $\mathcal{Q}_{\lambda z.hz}$ and P_2 is in \mathcal{Q}_z or P_1 is in $\mathcal{Q}_{\lambda z'.hz}$ and $P_2 = \square_o$.

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 $\lambda z'.P_1$ and P_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 $\lambda z'.P_1$ and P_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 .

$$\begin{aligned}
 & (hq_a) \rightarrow q_{ha} \\
 & (fq_l q_{ha}) \rightarrow q_{ha} \\
 & (fq_{\lambda z.hz} q_a) \rightarrow q_{ha} \\
 & (fq_{\lambda z.ha} q_{\square_o}) \rightarrow q_{ha} \\
 & \quad a \rightarrow q_a \\
 & (fq_l q_a) \rightarrow q_a \\
 & (fq_{\lambda z.a} q_{\square_o}) \rightarrow q_a \\
 & \quad \lambda z.q_z \rightarrow q_l \\
 & \quad \lambda z.q_{hz} \rightarrow q_{\lambda z.hz} \\
 & \quad \lambda z.q_{ha} \rightarrow q_{\lambda z.ha} \\
 & \quad \lambda z.q_a \rightarrow q_{\lambda z.a} \\
 & \quad z \rightarrow q_z \\
 & (fq_l q_z) \rightarrow q_z \\
 & (fq_{\lambda z'.z} q_{\square_o}) \rightarrow q_z \\
 & (hq_z) \rightarrow q_{hz} \\
 & (fq_l q_{hz}) \rightarrow q_{hz} \\
 & (fq_{\lambda z.hz} q_z) \rightarrow q_{hz} \\
 & (fq_{\lambda z'.hz} q_{\square_o}) \rightarrow q_{hz} \\
 & \quad \lambda z'.q_z \rightarrow q_{\lambda z'.z} \\
 & \quad \lambda z'.q_{hz} \rightarrow q_{\lambda z'.hz}
 \end{aligned}$$

Then we need a rule that permits to recognize \square_o in the state q_{\square_o} .

$$\square_o \rightarrow q_{\square_o}$$

and at last a rule that permits to recognize in q_0 the relevant solution of the equation $(X \mathbf{c}_1) = ha$

$$\lambda f.q_{ha} \rightarrow q_0$$

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 , \mathbf{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

$$N = \lambda y_1 \cdots y_p.P$$

where P is a pattern (see Definition 4C.4) of a subterm of ha and the free variables of P are in the set $\{z, z'\}$.

Tree automata for relevant solutions

The proof given here is for λ_{\rightarrow}^0 , but can easily be generalized to the full $\lambda_{\rightarrow}^{\mathbb{A}}$.

4C.4. DEFINITION. Let M be a normal term and \mathcal{V} be a set of k variables of type 0 not occurring in M where k is the size of M . A *pattern* of M is a term P such that there exists a substitution σ mapping the variables of \mathcal{V} to terms of type 0 such that $\sigma P = M$.

Consider an equation

$$X \vec{M} = N$$

where $\vec{M} = M_1, \dots, M_n$ and X is a variable of rank 3 type at most. Consider a finite number of constants \square_A for each type A subtype of a type of X . Let k be the size of N . Consider a fixed set \mathcal{V} of k variables of type 0. Let \mathcal{N} be the finite set of terms of the form $\lambda y_1 \cdots y_p.P$, where the y_i are of type 0, the term P is a pattern of a subterm of N and the free variables of P are in \mathcal{V} . Also the p should be bounded as follows: if $M_i : A_1^i \dots A_{n_i}^i \rightarrow 0$, then $p <$ the maximal arity of all A_j^i . It is easy to check that in the special case that P is not of ground type (that is, starts with a λ which, intuitively, binds a variable in N introduced directly or hereditarily by a constant of N of higher-order type) then one can take $p = 0$.

We define a tree automaton with the states q_W for W in \mathcal{N} and q_{\square_A} for each constant \square_A , and the transitions

- $(f_i q_{W_1} \cdots q_{W_n}) \rightarrow q_W$, if $(M_i \vec{W}) = W$ and replacing a W_i different from \square_A by a \square_A does not yield a solution,
- $(h q_{N_1} \cdots q_{N_n}) \rightarrow q_{(h N_1 \dots N_n)}$, for N_1, \dots, N_n and $(h N_1 \dots N_n)$ in \mathcal{N} ,
- $\square_A \rightarrow q_{\square_A}$
- $\lambda z. q_t \rightarrow q_{\lambda z. t}$
- $\lambda f_1 \cdots f_n. q_N \rightarrow q_0$.

4C.5. PROPOSITION. Let U and W be two elements of \mathcal{N} and X_1, \dots, X_n be variables of order at most two. Let σ be a relevant solution of the second-order matching problem

$$(U X_1 \cdots X_n) = W$$

then for each i , either σX_i is in \mathcal{N} (modulo alpha-conversion) or is equal to \square_A .

PROOF. Let U' be the normal form of $(U \sigma X_1 \cdots \sigma X_{i-1} X_i \sigma X_{i+1} \cdots \sigma X_n)$. If X_i has no occurrence in U' then as σ is relevant $\sigma X_i = \square_A$.

Otherwise consider the higher occurrence at position l of a subterm of type 0 of U' that has the form $(X_i V_1 \cdots V_p)$. The terms V_1, \dots, V_p have type 0. Let W_0 be the subterm of W at the same position l . The term W_0 has type 0, it is a pattern of a subterm of N .

Let V'_i be the normal form of $V_i[\sigma X_i/X_i]$. We have $(\sigma X_i V'_1 \cdots V'_p) = W_0$. Consider p variables y_1, \dots, y_p of \mathcal{V} that are not free in W_0 . We have $\sigma X_i = \lambda y_1 \cdots y_p.P$ and

$$P[V'_1/y_1, \dots, V'_p/y_p] = W_0.$$

Hence P is a pattern of a subterm of N and $\sigma X_i = \lambda y_1 \cdots y_p.P$ is an element of \mathcal{N} . ■

4C.6. REMARK. As a corollary of Proposition 4C.5, we get an alternative proof of the decidability of second-order matching.

4C.7. PROPOSITION. *Let*

$$X \vec{M} = N$$

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 $V \vec{M} = N$. It is sufficient to prove that V is recognized in the state q_N if and only if it is a relevant solution of the equation $V[f_1 := M_1, \dots, f_n := M_n] = N$. We prove, more generally, that for any term W of \mathcal{N} , V is recognized in q_W if and only if $V[f_1 := M_1, \dots, f_n := M_n] = W$.

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 $V[f_1 := M_1, \dots, f_n := M_n] = W$. If $V = (f_i V_1 \cdots V_p)$ then the term V_i is recognized in a state q_{W_i} , where W_i is either a term of \mathcal{N} or \square_A and $(M_i \vec{W}) = W$. In the first case, by induction hypothesis V_i is a relevant solution of the equation $V_i[f_1 := M_1, \dots, f_n := M_n] = M_i$ and in the second $V_i = \square_A$. Thus $(M_i V_1[f_1 := M_1, \dots, f_n := M_n] \cdots V_p[f_1 := M_1, \dots, f_n := M_n]) = N$, i.e. $V[f_1 := M_1, \dots, f_n := M_n] = N$, and moreover V is relevant. If $V = (h V_1 \cdots V_p)$, then the V_i are recognized in states q_{W_i} with W_i in \mathcal{N} . By induction hypothesis V_i are relevant solutions of $V_i[f_1 := M_1, \dots, f_n := M_n] = M_i$. Hence $V[f_1 := M_1, \dots, f_n := M_n] = N$ and moreover V is relevant. The case where V is an abstraction is similar.

Conversely, assume that V is a relevant solution of the problem

$$V[f_1 := M_1, \dots, f_n := M_n] = W.$$

We prove, by induction over the structure of V , that V is recognized in q_W .

If $V \equiv (f_i V_1 \cdots V_p)$ then

$$(M_i V_1[f_1 := M_1, \dots, f_n := M_n] \cdots V_p[f_1 := M_1, \dots, f_n := M_n]) = N.$$

Let $V'_i = V_i[f_1 := M_1, \dots, f_n := M_n]$. The V'_i are relevant solutions of the second-order matching problem $(M_i V'_1 \cdots V'_p) = N$. Now, by Proposition 4C.5, each V'_i is either an element of \mathcal{N} or the constant \square_A . In both cases V_i is a relevant solution of the equation $V_i[f_1 := M_1, \dots, f_n := M_n] = V'_i$ and by induction hypothesis V_i is recognized in q_{W_i} . Thus V is recognized in q_W .

If $V = (h V_1 \cdots V_p)$ then

$$(h V_1[f_1 := M_1, \dots, f_n := M_n] \cdots V_p[f_1 := M_1, \dots, f_n := M_n]) = W.$$

Let $W_i = V_i[f_1 := M_1, \dots, f_n := M_n]$. We have $(h \vec{W}) = W$ and V_i is a relevant solution of the equation $V_i[f_1 := M_1, \dots, f_n := M_n] = W_i$. By induction hypothesis V_i is recognized in q_{W_i} . Thus V is recognized in q_W . The case where V is an abstraction is similar. ■

4C.8. PROPOSITION. *Rank 3 dual interpolation is decidable.*

PROOF. Consider a system of equations and inequalities 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 0. 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 \square_A recognizes all terms of type A 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 rank 3 matching where problems are formulated in a language without any constant and the solutions also must not contain any constant.

Consider a problem $M = N$. The term N contains no constant. Hence, by the reducibility theorem, Theorem 3D.8, there are closed terms R_1, \dots, R_κ of type $A \rightarrow 0$, whose constants have order at most two (i.e. level at most one), such that for each term M of type A

$$M =_{\beta\eta} N \Leftrightarrow \forall \ell. (R_\ell M) =_{\beta\eta} (R_\ell N).$$

The normal forms of $(R_\ell N) \in \Lambda^0(0)$ are closed terms whose constants have order at most two, thus it contains no bound variables. Let \mathcal{U} be the set of all subterms of type 0 of the normal forms of $R_\ell N$. 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 0 that identifies all terms that are not in \mathcal{U} . This congruence has $\text{card}(\mathcal{U}) + 1$ equivalence classes.

4C.9. DEFINITION. $M =_{\beta\eta N} M' \Leftrightarrow \forall U \in \mathcal{U} [M =_{\beta\eta} U \Leftrightarrow M' =_{\beta\eta} U]$.

Notice that if $M, M' \in \Lambda^0(0)$ one has the following

$$\begin{aligned} M =_{\beta\eta N} M' &\Leftrightarrow M =_{\beta\eta} M' \text{ or } \forall U \in \mathcal{U} (M \neq_{\beta\eta} U \& M \neq_{\beta\eta} U) \\ &\Leftrightarrow [M =_{\beta\eta} M' \\ &\quad \text{or neither the normal form of } M \text{ nor that of } M' \text{ is in } \mathcal{U}] \end{aligned}$$

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 3C.28. However, we choose to do it explicitly so as to prepare for Definition 4C.18.

4C.10. DEFINITION. Let \parallel_N be the logical relation lifted from $=_{\beta\eta N}$ on closed terms.

4C.11. LEMMA. (i) \parallel_N is head-expansive.

- (ii) For each constant F of type of rank ≤ 1 one has $F \parallel_N F$.
- (iii) For any $X \in \Lambda(A)$ one has $X \parallel_N X$.
- (iv) \parallel_N is an equivalence relation.
- (v) $P \parallel_N Q \Leftrightarrow \forall S_1, \dots, S_k. P\vec{S} \parallel_N Q\vec{S}$.

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}_A 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 $\mathcal{T}_A / \parallel_N$ can be enumerated up to rank n . We first prove the following Proposition.

4C.12. PROPOSITION (Substitution lemma). *Let M be a normal term of type 0, whose free variables are x_1, \dots, x_n . Let $V_1, \dots, V_n, V'_1, \dots, V'_n$ be closed terms such that $V_1 \parallel_N V'_1, \dots, V_n \parallel_N V'_n$. Let $\sigma = V_1/x_1, \dots, V_n/x_n$ and $\sigma' = V'_1/x_1, \dots, V'_n/x_n$. Then*

$$\sigma M =_{\beta\eta N} \sigma' M$$

PROOF. By induction on the pair formed with the length of the longest reduction in σM and the size of M . The term M is normal and has type 0, thus it has the form $(f W_1 \cdots W_k)$.

If f is a constant, then let us write $W_i = \bar{\lambda} S_i$ with S_i of type 0. We have $\sigma M = (f \bar{\lambda} \sigma S_1 \cdots \bar{\lambda} \sigma S_k)$ and $\sigma' M = (f \bar{\lambda} \sigma' S_1 \cdots \bar{\lambda} \sigma' S_k)$. By induction hypothesis (as the S_i 's are subterms of M) we have $\sigma S_1 =_{\beta\eta N} \sigma' S_1, \dots, \sigma S_k =_{\beta\eta N} \sigma' S_k$, thus either for all i , $\sigma S_i =_{\beta\eta} \sigma' S_i$ and in this case $\sigma M =_{\beta\eta} \sigma' M$ or for some i , neither the normal forms of σS_i nor that of $\sigma' S_i$ is an element of \mathcal{U} . In this case neither the normal form of σM nor that of $\sigma' M$ is in \mathcal{U} and $\sigma M =_{\beta\eta N} \sigma' M$.

If f is a variable x_i and $k = 0$ then $M = x_i$, $\sigma M = V_i$ and $\sigma' M = V'_i$ and V_i and V'_i have type 0. Thus $\sigma M =_{\beta\eta N} \sigma' M$.

Otherwise, f is a variable x_i and $k \neq 0$. The term V_i has the form $\lambda z_1 \cdots \lambda z_k S$ and the term V'_i has the form $\lambda z_1 \cdots \lambda z_k S'$. We have

$$\sigma M = (V_i \sigma W_1 \cdots \sigma W_k) =_{\beta\eta} S[\sigma W_1/z_1, \dots, \sigma W_k/z_k]$$

and $\sigma' M = (V'_i \sigma' W_1 \cdots \sigma' W_k)$. As $V_i \parallel_N V'_i$, we get

$$\sigma' M =_{\beta\eta N} (V_i \sigma' W_1 \cdots \sigma' W_k) =_{\beta\eta N} S[\sigma' W_1/z_1, \dots, \sigma' W_k/z_k]$$

It is routine to check that for all i , $(\sigma W_i) \parallel_N (\sigma' W_i)$. Indeed, if the term W_i has the form $\lambda y_1 \cdots \lambda y_p O$, then for all closed terms $Q_1 \cdots Q_p$, we have

$$\begin{aligned} \sigma W_i Q_1 \cdots Q_p &= ((Q_1/y_1, \dots, Q_p/y_p) \circ \sigma) O \\ \sigma' W_i Q_1 \cdots Q_p &= ((Q_1/y_1, \dots, Q_p/y_p) \circ \sigma') O. \end{aligned}$$

Applying the induction hypothesis to O that is a subterm of M , we get

$$(\sigma W_i) Q_1 \cdots Q_p =_{\beta\eta N} (\sigma' W_i) Q_1 \cdots Q_p$$

and thus $(\sigma W_i) \parallel_N (\sigma' W_i)$.

As $(\sigma W_i) \parallel_N (\sigma' W_i)$ we can apply the induction hypothesis again, because

$$\sigma M \twoheadrightarrow s[\sigma W_1/z_1, \dots, \sigma W_k/z_k],$$

and get

$$S[\sigma W_1/z_1, \dots, \sigma W_k/z_k] =_{\beta\eta N} S[\sigma' W_1/z_1, \dots, \sigma' W_k/z_k]$$

Thus $\sigma M =_{\beta\eta N} \sigma' M$. ■

The next proposition is a direct corollary.

4C.13. PROPOSITION (Application lemma). *If $V_1 \parallel_N V'_1, \dots, V_n \parallel_N V'_n$, then for all term M of type $A_1 \rightarrow \dots \rightarrow A_n \rightarrow 0$,*

$$(M V_1 \cdots V_n) =_{\beta\eta N} (M V'_1 \cdots V'_n).$$

PROOF. Applying Proposition 4C.12 to the term $(M x_1 \cdots x_n)$. ■

We then prove the following lemma that justifies the use of the relations $=_{\beta\eta N}$ and \parallel_N .

4C.14. PROPOSITION (Discrimination lemma). *Let M be a term. Then*

$$M \parallel_N N \Rightarrow M =_{\beta\eta} N.$$

PROOF. As $M \parallel_N N$, by Proposition 4C.13, we have for all ℓ , $(R_\ell M) =_{\beta\eta N} (R_\ell N)$. Hence, as the normal form of $(R_\ell N)$ is in \mathcal{U} , $(R_\ell M) =_{\beta\eta} (R_\ell N)$. Thus $M =_{\beta\eta} N$. ■

Let us discuss now how we can decide and enumerate the relation \parallel_N . If M and M' are of type $A_1 \rightarrow \dots \rightarrow A_n \rightarrow 0$, then, by definition, $M \parallel_N M'$ if and only if

$$\forall W_1 \in \mathcal{T}_{A_1} \cdots \forall W_n \in \mathcal{T}_{A_n} (M \vec{W} =_{\beta\eta N} M' \vec{W})$$

The fact that $M \vec{W} =_{\beta\eta N} M' \vec{W}$ can be reformulated

$$\forall U \in \mathcal{U} (M \vec{W} =_{\beta\eta} U \text{ if and only if } M' \vec{W} =_{\beta\eta} U)$$

Thus $M \parallel_N M'$ if and only if

$$\forall W_1 \in \mathcal{T}_{A_1} \cdots \forall W_n \in \mathcal{T}_{A_n} \forall U \in \mathcal{U} (M \vec{W} =_{\beta\eta} U \text{ if and only if } M' \vec{W} =_{\beta\eta} M)$$

Thus to decide if $M \parallel_N M'$, we should list all the sequences U, W_1, \dots, W_n where U is an element of \mathcal{U} and W_1, \dots, W_n are closed terms of type A_1, \dots, A_n , and check that the set of sequences such that $M \vec{W} =_{\beta\eta} U$ is the same as the set of sequences such that $M' \vec{W} =_{\beta\eta} U$.

Of course, the problem is that there is an infinite number of such sequences. But by Proposition 4C.13 the fact that $M \vec{W} =_{\beta\eta N} M' \vec{W}$ is not affected if we replace the terms W_i by \parallel_N -equivalent terms. Hence, if we can enumerate the sets $\mathcal{T}_{A_1}/\parallel_N, \dots, \mathcal{T}_{A_n}/\parallel_N$ by sets $\mathcal{E}_{A_1}, \dots, \mathcal{E}_{A_n}$, then we can decide the relation \parallel_N for terms of type $A_1 \rightarrow \dots \rightarrow A_n \rightarrow 0$ by enumerating the sequences in $\mathcal{U} \times \mathcal{E}_{A_1} \times \dots \times \mathcal{E}_{A_n}$, and checking that the set of sequences such that $M \vec{W} =_{\beta\eta} U$ is the same as the set of sequences such that $M' \vec{W} =_{\beta\eta} U$.

As class of a term M for the relation \parallel_N is completely determined, by the set of sequences U, W_1, \dots, W_n such that $M \vec{W} =_{\beta\eta} U$ and there are a finite number of subsets of the set $\mathcal{E} = \mathcal{U} \times \mathcal{E}_{A_1} \times \dots \times \mathcal{E}_{A_n}$, we get this way that the set $\mathcal{T}_A/\parallel_N$ is finite.

To obtain an enumeration \mathcal{E}_A of the set $\mathcal{T}_A/\parallel_N$ we need to be able to select the subsets \mathcal{A} of $\mathcal{U} \times \mathcal{E}_{A_1} \times \dots \times \mathcal{E}_{A_n}$, such that there is a term M such that $M \vec{W} =_{\beta\eta} U$ if and only if the sequence U, \vec{W} is in \mathcal{A} . This condition is exactly the decidability of the dual interpolation problem. This leads to the following proposition.

4C.15. PROPOSITION (Enumeration lemma). *If dual interpolation of rank n is decidable, then the sets $\mathcal{T}_A/\parallel_N$ can be enumerated up to rank n .*

PROOF. By induction on the order of $A = A_1 \rightarrow \dots \rightarrow A_n \rightarrow 0$. By the induction hypothesis, the sets $\mathcal{T}_{A_1}/\parallel_N, \dots, \mathcal{T}_{A_n}/\parallel_N$ can be enumerated by sets $\mathcal{E}_{A_1}, \dots, \mathcal{E}_{A_n}$.

Let x be a variable of type A . For each subset \mathcal{A} of $\mathcal{E} = \mathcal{U} \times \mathcal{E}_{A_1} \times \dots \times \mathcal{E}_{A_n}$ we define the dual interpolation problem containing the equation $x\vec{W} = U$ for $U, W_1, \dots, W_p \in \mathcal{A}$ and the negated equation $x\vec{W} \neq U$ for $U, W_1, \dots, W_p \notin \mathcal{A}$. 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}_A .

We prove that this set is an enumeration of $\mathcal{T}_A/\parallel_N$, i.e. that for every term M of type A there is a term M' in \mathcal{E}_A such that $M' \parallel_N M$. Let \mathcal{A} be the set of sequences U, W_1, \dots, W_p such that $(M \vec{W}) =_{\beta\eta} U$. The dual interpolation problem corresponding to \mathcal{A} has a solution (for instance M). Thus one of its solutions M' is in \mathcal{E}_A . We have

$$\forall W_1 \in \mathcal{E}_{A_1} \dots \forall W_n \in \mathcal{E}_{A_n} \forall U \in \mathcal{U} ((M \vec{W}) =_{\beta\eta} U \Leftrightarrow (M' \vec{W}) =_{\beta\eta} U).$$

Thus

$$\forall W_1 \in \mathcal{E}_{A_1} \dots \forall W_n \in \mathcal{E}_{A_n} (M \vec{W}) =_{\beta\eta N} (M' \vec{W});$$

hence by Proposition 4C.13

$$\forall W_1 \in \mathcal{T}_{A_1} \dots \forall W_n \in \mathcal{T}_{A_n} (M \vec{W}) =_{\beta\eta N} (M' \vec{W}).$$

Therefore $M \parallel_N M'$. ■

Then, we prove that if the sets $\mathcal{T}_A/\parallel_N$ 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}_A .

4C.16. PROPOSITION (Matching lemma). *If the sets $\mathcal{T}_A/\parallel_N$ can be enumerated up to order n , then matching problems of rank n whose right hand side is N can be decided.*

PROOF. Let $\vec{X} = X_1, \dots, X_m$. We prove that if a matching problem $M\vec{X} = N$ has a solution \vec{V} , then it has also a solution \vec{V} , such that $\underline{V}_i \in \mathcal{E}_{A_i}$ for each i , where A_i is the type of X_i .

As \vec{V} is a solution of the problem $M = N$, we have $M\vec{V} =_{\beta\eta} N$.

For all i , let \underline{V}_i be a representative in \mathcal{E}_{A_i} of the class of V_i . We have

$$\underline{V}_1 \parallel_N V_1, \dots, \underline{V}_m \parallel_N V_m.$$

Thus by Proposition 4C.12

$$M\vec{V} =_{\beta\eta N} M\vec{V},$$

hence

$$M\vec{V} =_{\beta\eta N} N,$$

and therefore by Proposition 4C.14

$$M\vec{V} =_{\beta\eta} N.$$

Thus for checking whether a problem has a solution it suffices to check whether it has a solution \vec{V} , with each \underline{V}_i in \mathcal{E}_A ; such substitutions can be enumerated. ■

Note that the proposition can be generalized: the enumeration allows to solve every matching *inequality* of right member N , and more generally, every dual matching problem.

4C.17. THEOREM. *Rank 3 matching problems whose right hand side contain no constants can be decided.*

PROOF. Dual interpolation of order 4 is decidable, hence, by proposition 4C.15, if N is a closed term containing no constants, then the sets $\mathcal{T}_A / \|_N$ can be enumerated up to order 4, hence, by Proposition 4C.16, we can decide if a problem of the form $M = N$ has a solution. ■

The general case

We consider now terms formed in a language 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 \mathcal{U} as the set of subterms of N of type 0. The novelty here is that the bound variables of U may now appear free in the terms of \mathcal{U} . It is important here to chose the names x_1, \dots, x_n of these variables, once for all.

We define the congruence $M =_{\beta\eta N} M'$ on terms of type 0 that identifies all terms that are not in \mathcal{U} .

4C.18. DEFINITION. (i) Let $M, M' \in \Lambda(0)$ (not necessarily closed). Define

$$M =_{\beta\eta N} M' \Leftrightarrow \forall U \in \mathcal{U}. [M =_{\beta\eta} U \Leftrightarrow M' =_{\beta\eta} U].$$

(ii) Define the logical relation $\|_N$ by lifting $=_{\beta\eta N}$ to all open terms at higher types.

4C.19. LEMMA. (i) $\|_N$ is head-expansive.

- (ii) For any variable x of arbitrary type A one has $x \|_N x$.
- (iii) For each constant $F \in \Lambda(A)$ one has $F \|_N F$.
- (iv) For any $X \in \Lambda(A)$ one has $X \|_N X$.
- (v) $\|_N$ is an equivalence relation at all types.
- (vi) $P \|_N Q \Leftrightarrow \forall S_1, \dots, S_k. P \vec{S} \|_N Q \vec{S}$.

PROOF. (i) By definition the relation is closed under arbitrary $\beta\eta$ 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 enumeration Lemma, Proposition 4C.15. Due to the presence of the free variables, the proof of this lemma introduces several novelties. Given a subset \mathcal{A} of $\mathcal{E} = \mathcal{U} \times \mathcal{E}_{A_1} \times \dots \times \mathcal{E}_{A_n}$ we cannot define the dual interpolation problem containing the equation $(x \vec{W}) = U$ for $U, W_1, \dots, W_p \in \mathcal{A}$ and the negated equation $(x \vec{W}) \neq U$ for $U, W_1, \dots, W_p \notin \mathcal{A}$, because the right hand side of these equations may contain free variables. Thus, we shall replace these variables by fresh constants c_1, \dots, c_n . Let θ be the substitution $c_1/x_1, \dots, c_n/x_n$. To each set of sequences, we associate the dual interpolation problem containing the equation $(x \vec{W}) = \theta U$ or its negation.

This introduces two difficulties: first the term θU is not a subterm of N , thus, besides the relation $\|_N$, we shall need to consider also the relation $\|_{\theta U}$, and one of its enumerations, for each term U in \mathcal{U} . Then, the solutions of such interpolation problems could contain the constants c_1, \dots, c_n , and we may have difficulties proving that they

represent their \parallel_N -equivalence class. To solve this problem we need to duplicate the constants c_1, \dots, c_n with constants d_1, \dots, d_n . This idea goes back to Goldfarb [1981].

Let us consider a fixed set of constants $c_1, \dots, c_n, d_1, \dots, d_n$ that do not occur in N , and if M is a term containing constants c_1, \dots, c_n , but not the constants d_1, \dots, d_n , we write \tilde{M} for the term M where each constant c_i is replaced by the constant d_i .

Let $A = A_1 \rightarrow \dots \rightarrow A_n \rightarrow 0$ be a type. We assume that for any closed term U of type 0, the sets $\mathcal{T}_{A_i} / \parallel_U$ can be enumerated up to rank n by sets $\mathcal{E}_{A_i}^U$.

4C.20. DEFINITION. We define the set of sequences \mathcal{E} containing for each term U in \mathcal{U} and sequence W_1, \dots, W_n in $\mathcal{E}_{A_1}^{\theta U} \times \dots \times \mathcal{E}_{A_n}^{\theta U}$, the sequence $\theta U, W_1, \dots, W_n$. Notice that the terms in these sequences may contain the constants c_1, \dots, c_n but not the constants d_1, \dots, d_n .

To each subset of \mathcal{A} of \mathcal{E} we associate a dual interpolation problem containing the equations $x \vec{W} = U$ and $x \tilde{W}_1 \dots \tilde{W}_n = \tilde{U}$ for $U, W_1, \dots, W_n \in \mathcal{A}$ and the inequalities $x \vec{W} \neq U$ and $x \tilde{W}_1 \dots \tilde{W}_n \neq \tilde{U}$ for $U, W_1, \dots, W_n \notin \mathcal{A}$.

The first lemma justifies the use of constants duplication.

4C.21. PROPOSITION. *If an interpolation problem of Definition 4C.20 has a solution M , then it also has a solution M' that does not contain the constants $c_1, \dots, c_n, d_1, \dots, d_n$.*

PROOF. Assume that the term M contains a constant, say c_1 . Then by replacing this constant c_1 by a fresh constant e , we obtain a term M' . As the constant e is fresh, all the inequalities that M verify are still verified by M' . If M verifies the equations $x \vec{W} = U$ and $x \tilde{W}_1 \dots \tilde{W}_n = \tilde{U}$, then the constant e does not occur in the normal form of $M' \vec{W}$. Otherwise the constant c_1 would occur in the normal form of $M \tilde{W}_1 \dots \tilde{W}_n$, i.e. in the normal form of \tilde{U} which is not the case. Thus M' also verifies the equations $x \vec{W} = U$ and $x \tilde{W}_1 \dots \tilde{W}_n = \tilde{U}$.

We can replace this way all the constants $c_1, \dots, c_n, d_1, \dots, d_n$ by fresh constants, obtaining a solution where these constants do not occur. ■

Then, we prove that the interpolation problems of Definition 4C.20 characterize the equivalence classes of the relation \parallel_N .

4C.22. PROPOSITION. *Every term M of type A not containing the constants $c_1, \dots, c_n, d_1, \dots, d_n$ is the solution of a unique problem of Definition 4C.20.*

PROOF. Consider the subset \mathcal{A} of \mathcal{E} formed with sequences U, W_1, \dots, W_n such that $M \vec{W} = U$. The term M is the solution of the interpolation problem associated to \mathcal{A} and \mathcal{A} is the only subset of \mathcal{E} such that M is a solution to the interpolation problem associated to. ■

4C.23. PROPOSITION. *Let M and M' be two terms of type A not containing the constants $c_1, \dots, c_n, d_1, \dots, d_n$. Then M and M' are solutions of the same unique problem of Definition 4C.20 iff $M \parallel_N M'$.*

PROOF. By definition if $M \parallel_N M'$ then for all W_1, \dots, W_n and for all U in \mathcal{U} : $M \vec{W} =_{\beta\eta} U \Leftrightarrow M' \vec{W} =_{\beta\eta} U$. Thus for any U, \vec{W} in \mathcal{E} , $\theta^{-1}U$ is in \mathcal{U} and $M \theta^{-1}W_1 \dots \theta^{-1}W_n =_{\beta\eta} \theta^{-1}U \Leftrightarrow M' \theta^{-1}W_1 \dots \theta^{-1}W_n =_{\beta\eta} \theta^{-1}U$. Then, as the constants $c_1, \dots, c_n, d_1, \dots, d_n$ do not appear in M and M' , we have $M \vec{W} =_{\beta\eta} U \Leftrightarrow M' \vec{W} =_{\beta\eta} U$ and $M \tilde{W}_1 \dots \tilde{W}_n =_{\beta\eta} \tilde{U} \Leftrightarrow M' \tilde{W}_1 \dots \tilde{W}_n =_{\beta\eta} \tilde{U}$. Thus M and M' are the solutions of the same problem.

Conversely, assume that $M \not\parallel_N M'$. Then there exists terms W_1, \dots, W_n and a term U in \mathcal{U} such that $M \vec{W} =_{\beta\eta} U$ and $M' \vec{W} \neq_{\beta\eta} U$. Hence $M \theta W_1 \dots \theta W_n =_{\beta\eta} \theta U$ and $M' \theta W_1 \dots \theta W_n \neq_{\beta\eta} \theta U$. As the sets $\mathcal{E}_{A_i}^{\theta U}$ are enumeration of the sets $\mathcal{T}_{A_i}/ \parallel_{\theta U}$ there exists terms \vec{S} such that the $S_i \parallel_{\theta U} \theta W_i$ and $\theta U, \vec{S} \in \mathcal{E}$. Using Proposition 4C.13 we have $M \vec{S} =_{\beta\eta\theta U} M \theta W_1 \dots \theta W_n =_{\beta\eta} \theta U$, hence $M \vec{S} =_{\beta\eta\theta U} \theta U$ i.e. $M \vec{S} =_{\beta\eta} \theta U$. Similarly, we have $M' \vec{S} =_{\beta\eta\theta U} M' \theta W_1 \dots \theta W_n \neq_{\beta\eta} \theta U$ hence $M' \vec{S} \neq_{\beta\eta\theta U} \theta U$ i.e. $M' \vec{S} \neq_{\beta\eta} \theta U$. Hence M and M' are not the solutions of the same problem. ■

Finally, we can prove the enumeration lemma.

4C.24. PROPOSITION (Enumeration lemma). *If dual interpolation of rank n is decidable, then, for any closed term N of type 0, the sets $\mathcal{T}_A/\parallel_N$ can be enumerated up to rank n .*

PROOF. By induction on the order of A . Let $A = A_1 \rightarrow \dots \rightarrow A_n \rightarrow 0$. By the induction hypothesis, for any closed term U of type 0, the sets $\mathcal{T}_{A_i}/\parallel_U$ can be enumerated by sets $\mathcal{E}_{A_i}^U$.

We consider all the interpolation problems of Definition 4C.20. Using the decidability of dual interpolation of rank n , we select those of such problems that have a solution. By Proposition 4C.21, we can construct for each such problem a solution not containing the constants $c_1, \dots, c_n, d_1, \dots, d_n$ and by Proposition 4C.22 and 4C.23, these terms form an enumeration of $\mathcal{T}_A/\parallel_N$. ■

To conclude, we prove the matching lemma (Proposition 4C.16) exactly as in the particular case and then the theorem.

4C.25. THEOREM (Padovani). *Rank 3 matching problems can be decided.*

PROOF. Dual interpolation of order 4 is decidable, hence, by Proposition 4C.15, if N is a closed term, then the sets $\mathcal{T}_A/\parallel_N$ can be enumerated up to order 4, hence, by Proposition 4C.16, we can decide if a problem of the form $M = N$ has a solution. ■

4D. Decidability of the maximal theory

We prove now that the maximal theory is decidable. The original proof of this result is due to [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, see Definition 3E.46, is

$$\mathcal{T}_{\max}\{M = N \mid M, N \in \Lambda_0^\emptyset(A), A \in \mathbb{T}^0 \text{ & } \mathcal{M}_{\min}^{\vec{c}} \models M = N\},$$

where

$$\mathcal{M}_{\min}^{\vec{c}} = \Lambda_0^\emptyset[\vec{c}] / \approx_{\vec{c}}^{\text{ext}}$$

consists of all terms having the $\vec{c} = c_1, \dots, c_n$, with $n > 1$, of type 0 as distinct constants and $M \approx_{\vec{c}}^{\text{ext}} N$ on type $A = A_1 \rightarrow \dots \rightarrow A_a \rightarrow 0$ is defined by

$$M \approx_{\vec{c}}^{\text{ext}} N \Leftrightarrow \forall P_1 \in \Lambda_0^\emptyset[\vec{c}](A_1) \dots P_a \in \Lambda_0^\emptyset[\vec{c}](A_a). M \vec{P} =_{\beta\eta} N \vec{P}.$$

Theorem 3E.34 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 .

4D.1. DEFINITION. Let $A \in \mathbb{T}^{\mathbb{A}}$. The *degree* of A , notation $\|A\|$, is defined as follows.

$$\begin{aligned}\|0\| &= 2, \\ \|A \rightarrow B\| &= \|A\|! \|B\|, \quad \text{i.e. } \|A\| \text{ factorial times } \|B\|.\end{aligned}$$

4D.2. PROPOSITION. (i) $\|A_1 \rightarrow \cdots \rightarrow A_n \rightarrow 0\| = 2\|A_1\|! \cdots \|A_n\|!$.

$$\text{(ii)} \quad \|A_i\| < \|A_1 \rightarrow \cdots \rightarrow A_n \rightarrow 0\|.$$

$$\text{(iii)} \quad n < \|A_1 \rightarrow \cdots \rightarrow A_n \rightarrow 0\|.$$

(iv) If $p < \|A_i\|$, $\|B_1\| < \|A_i\|$, ..., $\|B_p\| < \|A_i\|$ then

$$\begin{aligned}\|A_1 \rightarrow \cdots \rightarrow A_{i-1} \rightarrow B_1 \rightarrow \cdots \rightarrow B_p \rightarrow A_{i+1} \rightarrow \cdots \rightarrow A_n \rightarrow 0\| &< \\ &< \|A_1 \rightarrow \cdots \rightarrow A_n \rightarrow 0\|.\end{aligned}$$

4D.3. DEFINITION. Let $M \in \Lambda_0^{\vec{c}}(A_1 \rightarrow \cdots \rightarrow A_n \rightarrow 0)$ be a lnf. Then either $M \equiv \lambda x_1 \cdots x_n.y$ or $M \equiv \lambda x_1 \cdots x_n.x_i M_1 \cdots M_p$. In the first case, M is called *constant*, in the second it *has index i*.

The following proposition states that for every type A , the terms $M \in \Lambda_0^{\vec{c}}(A)$ with a given index can be enumerated by a term $E : \vec{C} \rightarrow A$, where the \vec{C} have degrees lower than A .

4D.4. PROPOSITION. Let \approx be the equality in the minimal model (the maximal theory). Then for each type A and each natural number i , there exists a natural number $k < \|A\|$, types C_1, \dots, C_k such that $\|C_1\| < \|A\|, \dots, \|C_k\| < \|A\|$, a term E of type $C_1 \rightarrow \cdots \rightarrow C_k \rightarrow A$ and terms P_1 of type $A \rightarrow C_1, \dots, P_k$ of type $A \rightarrow C_k$ such that if M has index i then

$$M \approx E(P_1 M) \cdots (P_k M).$$

PROOF. By induction on $\|A\|$. Let us write $A = A_1 \rightarrow \cdots \rightarrow A_n \rightarrow 0$ and $A_i = B_1 \rightarrow \cdots \rightarrow B_m \rightarrow 0$. By induction hypothesis, for each j in $\{1, \dots, m\}$ there are types $D_{j,1}, \dots, D_{j,l_j}$, terms $E_j, P_{j,1}, \dots, P_{j,l_j}$ such that $l_j < \|A_i\|$, $\|D_{j,1}\| < \|A_i\|, \dots, \|D_{j,l_j}\| < \|A_i\|$ and if $N \in \Lambda_0^{\vec{c}}(A_i)$ has index j then

$$N \approx E_j(P_{j,1} N) \cdots (P_{j,l_j} N).$$

We take $k = m$, and define

$$C_1 \triangleq A_1 \rightarrow \cdots \rightarrow A_{i-1} \rightarrow D_{1,1} \rightarrow \cdots \rightarrow D_{1,l_1} \rightarrow A_{i+1} \rightarrow \cdots \rightarrow A_n \rightarrow 0,$$

...

$$C_k \triangleq A_1 \rightarrow \cdots \rightarrow A_{i-1} \rightarrow D_{k,1} \rightarrow \cdots \rightarrow D_{k,l_k} \rightarrow A_{i+1} \rightarrow \cdots \rightarrow A_n \rightarrow 0,$$

$$E \triangleq \lambda f_1 \cdots f_k x_1 \cdots x_n.$$

$$x_i(\lambda \vec{c}. f_1 x_1 \cdots x_{i-1} (P_{1,1} x_i) \cdots (P_{1,l_1} x_i) x_{i+1} \cdots x_n)$$

...

$$(\lambda \vec{c}. f_k x_1 \cdots x_{i-1} (P_{k,1} x_i) \cdots (P_{k,l_k} x_i) x_{i+1} \cdots x_n),$$

$$P_1 \triangleq \lambda g x_1 \cdots x_{i-1} \vec{z}_1 x_{i+1} \cdots x_n. g x_1 \cdots x_{i-1} (E_1 \vec{z}_1) x_{i+1} \cdots x_n,$$

...

$$P_k \triangleq \lambda g x_1 \cdots x_{i-1} \vec{z}_k x_{i+1} \cdots x_n. g x_1 \cdots x_{i-1} (E_k \vec{z}_k) x_{i+1} \cdots x_n,$$

where $\vec{z}_i = z_1, \dots, z_{l_i}$ for $1 \leq i \leq k$. We have $k < \|A_i\| < \|A\|, \|C_i\| < \|A\|$ for

$1 \leq i \leq k$ and for any $M \in \Lambda_0^\varnothing[\vec{c}](A)$

$$\begin{aligned} E(P_1 M) \cdots (P_k M) &= \lambda x_1 \cdots x_n.x_i \\ &\quad (\lambda \vec{c}.tx_1 \cdots x_{i-1}(E_1(P_{1,1}x_i) \cdots (P_{1,l_1}x_i))x_{i+1} \cdots x_n) \\ &\quad \dots \\ &\quad (\lambda \vec{c}.tx_1 \cdots x_{i-1}(E_k(P_{k,1}x_i) \cdots (P_{k,l_k}x_i))x_{i+1} \cdots x_n) \end{aligned}$$

We want to prove that if M has index i then this term is equal to M . Consider terms $\vec{Q} \in \Lambda_0^\varnothing[\vec{c}]$. We want to prove that for the term

$$\begin{aligned} Q &= Q_i(\lambda \vec{c}.tQ_1 \cdots Q_{i-1}(E_1(P_{1,1}Q_i) \cdots (P_{1,l_1}Q_i))Q_{i+1} \cdots Q_n) \\ &\quad \dots \\ &\quad (\lambda \vec{c}.tQ_1 \cdots Q_{i-1}(E_k(P_{k,1}Q_i) \cdots (P_{k,l_k}Q_i))Q_{i+1} \cdots Q_n) \end{aligned}$$

one has $Q \approx (MQ_1 \cdots Q_n)$. If Q_i is constant then this is obvious. Otherwise, it has an index j , say, and Q reduces to

$$Q' = MQ_1 \cdots Q_{i-1}(E_j(P_{j,1}Q_i) \cdots (P_{j,l_j}Q_i))Q_{i+1} \cdots Q_n.$$

By the induction hypothesis the term $(E_j(P_{j,1}Q_i) \cdots (P_{j,l_j}Q_i)) \approx Q_i$ and hence, by Theorem 3E.34 one has $Q = Q' \approx (MQ_1 \cdots Q_n)$. ■

4D.5. THEOREM. *Let \mathcal{M} be the minimal model built over $\vec{c}:0$, i.e.*

$$\mathcal{M} = \mathcal{M}_{\min} = \Lambda_0^\varnothing[\vec{c}]/\approx.$$

For each type A , we can compute a finite set $\mathcal{R}_A \subseteq \Lambda_0^\varnothing[\vec{c}](A)$ that enumerates $\mathcal{M}(A)$, i.e. such that

$$\forall M \in \mathcal{M}(A) \exists N \in \mathcal{R}_A. M \approx N.$$

PROOF. By induction on $\|A\|$. If $A = 0$, then we can take $\mathcal{R}_A = \{\vec{c}\}$. Otherwise write $A = A_1 \rightarrow \cdots \rightarrow A_n \rightarrow 0$. By Proposition 4D.4 for each $i \in \{1, \dots, n\}$, there exists a $k_i \in \mathbb{N}$, types $C_{i,1}, \dots, C_{i,k_i}$ smaller than A , a term E_i of type $C_{i,1} \rightarrow \cdots \rightarrow C_{i,k_i} \rightarrow A$ such that for each term M of index i , there exists terms P_1, \dots, P_{k_i} such that

$$M \approx (E_i P_1 \cdots P_{k_i}).$$

By the induction hypothesis, for each type $C_{i,j}$ we can compute a finite set $\mathcal{R}_{C_{i,j}}$ that enumerates $\mathcal{M}(C_{i,j})$. We take for \mathcal{R}_A all the terms of the form $(E_i Q_1 \cdots Q_{k_i})$ with Q_1 in $\mathcal{R}_{C_{i,1}}, \dots, Q_{k_i}$ in $\mathcal{R}_{C_{i,k_i}}$. ■

4D.6. COROLLARY (Padovani). *The maximal theory is decidable.*

PROOF. Check equivalence in any minimal model $\mathcal{M}_{\min}^{\vec{c}}$. At type $A = A_1 \rightarrow \cdots \rightarrow A_a \rightarrow 0$ we have

$$M \approx N \Leftrightarrow \forall P_1 \in \Lambda_0^\varnothing[\vec{c}](A_1) \cdots P_a \in \Lambda_0^\varnothing[\vec{c}](A_a). M \vec{P} =_{\beta\eta} N \vec{P},$$

where we can now restrict the \vec{P} to the \mathcal{R}_{A_j} . ■

4D.7. COROLLARY (Decidability of unification in \mathcal{T}_{\max}). *For terms*

$$M, N \in \Lambda_0^\varnothing[\vec{c}](A \rightarrow B),$$

of the same type, the following unification problem is decidable

$$\exists X \in \Lambda^\varnothing[\vec{c}](A). MX \approx NX.$$

PROOF. Working in $\mathcal{M}_{\min}^{\vec{c}}$, check the finitely many enumerating terms as candidates. ■

4D.8. COROLLARY (Decidability of atomic higher-order matching). (i) *For*

$$M_1 \in \Lambda_0^\varnothing[\vec{c}](A_1 \rightarrow 0), \dots, M_n \in \Lambda_0^\varnothing[\vec{c}](A_n \rightarrow 0),$$

with $1 \leq i \leq n$, the following problem is decidable

$$\begin{aligned} \exists X_1 \in \Lambda_0^\varnothing[\vec{c}](A_1), \dots, X_n \in \Lambda_0^\varnothing[\vec{c}](A_n). [M_1 X_1 =_{\beta\eta} c_1 \\ \dots \\ M_n X_n =_{\beta\eta} c_n]. \end{aligned}$$

(ii) *For $M, N \in \Lambda_0^\varnothing[\vec{c}](A \rightarrow 0)$ the following problem is decidable.*

$$\exists X \in \Lambda_0^\varnothing[\vec{c}](A). MX =_{\beta\eta} NX.$$

PROOF. (i) Since $\beta\eta$ -convertibility at type 0 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 4C.24 is not redundant. We follow the given construction, but actually the proof does not depend on it, see Exercise 4E.2. We first prove a converse to Proposition 4D.4.

4D.9. PROPOSITION. *Let E, P_1, \dots, P_k be the terms constructed in Proposition 4D.4. Then for any sequence of terms M_1, \dots, M_k , we have*

$$(P_j(EM_1 \cdots M_k)) \approx M_j.$$

PROOF. By induction on $\|A\|$ where A is the type of $(EM_1 \cdots M_k)$. The term

$$N \equiv P_j(EM_1 \cdots M_k)$$

reduces to

$$\begin{aligned} & \lambda x_1 \cdots x_{i-1} \vec{z}_j x_{i+1} \cdots x_n. E_j \vec{z}_j \\ & (\lambda \vec{c}. M_1 x_1 \cdots x_{i-1} (P_{1,1}(E_j \vec{z}_j)) \cdots (P_{1,l_1}(E_j \vec{z}_j)) x_{i+1} \cdots x_n) \\ & \dots \\ & (\lambda \vec{c}. M_k x_1 \cdots x_{i-1} (P_{k,1}(E_j \vec{z}_j)) \cdots (P_{k,l_k}(E_j \vec{z}_j)) x_{i+1} \cdots x_n) \end{aligned}$$

Then, since E_j is a term of index $l_j + j$, the term N continues to reduce to

$$\lambda x_1 \cdots x_{i-1} \vec{z}_j x_{i+1} \cdots x_n. M_j x_1 \cdots x_{i-1} (P_{j,1}(E_j \vec{z}_j)) \cdots (P_{j,l_j}(E_j \vec{z}_j)) x_{i+1} \cdots x_n.$$

We want to prove that this term is equal to M_j . Consider terms

$$N_1, \dots, N_{i-1}, \vec{L}_j, N_{i+1}, \dots, N_n \in \Lambda_0^\varnothing[\vec{c}].$$

It suffices to show that

$$\begin{aligned} & M_j N_1 \cdots N_{i-1} (P_{j,1}(E_j \vec{L}_j)) \cdots (P_{j,l_j}(E_j \vec{L}_j)) N_{i+1} \cdots N_n \approx \\ & M_j N_1 \cdots N_{i-1} \vec{L}_j N_{i+1} \cdots N_n. \end{aligned}$$

By the induction hypothesis we have

$$\begin{aligned} & (P_{j,1}(E_j \vec{L}_j)) \approx L_1, \\ & \dots \\ & (P_{j,l_j}(E_j \vec{L}_j)) \approx L_{l_j}. \end{aligned}$$

Hence by Theorem 3E.34 we are done. ■

4D.10. PROPOSITION. *The enumeration in Theorem 4D.5 is non-redundant, i.e.*

$$\forall A \in \mathbb{T}^0 \forall M, N \in \mathcal{R}_A. M \approx_{\mathcal{C}} N \Rightarrow M \equiv N.$$

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 $M = (E_i M'_1 \cdots M'_k)$ and $N = (E_i N'_1 \cdots N'_k)$. For all j , we have

$$M'_j \approx (P_j M) \approx (P_j N) \approx N'_j.$$

Hence, by induction hypothesis $M'_j = N'_j$ and therefore $M = N$. ■

4E. Exercises

4E.1. Let $\mathcal{M} = \mathcal{M}[\mathcal{C}_1]$ be the minimal model. Let $c_n = \text{card}(\mathcal{M}(1^n \rightarrow 0))$.

(i) Show that

$$\begin{aligned} c_0 &= 2; \\ c_{n+1} &= 2 + (n+1)c_n. \end{aligned}$$

(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” form a well-known sequence in combinatorics. See, for instance, [Flajolet and Sedgewick \[1993\]](#).

(iii) Can the cardinality of $\mathcal{M}(A)$ be bounded by a function of the form $k^{|A|}$ where $|A|$ is the size of $A \in \mathbb{T}^0$ and k a constant?

4E.2. Let $\mathcal{C} = \{c^0, d^0\}$. Let \mathcal{E} be a computable function that assigns to each type $A \in \mathbb{T}^0$ 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 4D one can effectively make \mathcal{E} non-redundant, i.e. such that

$$\forall A \in \mathbb{T}^0 \forall M, N \in \mathcal{E}_A. M \approx_{\mathcal{C}} N \Rightarrow M \equiv N.$$

4E.3. (Herbrand's Problem) Consider sets S of universally quantified equations

$$\forall x_1 \cdots x_n. [T_1 = T_2]$$

between first order terms involving constants f, g, h, \dots of various arities. 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. $a, b : 0$. We define the set \mathcal{S}^+ of simply typed equations by

$$\mathcal{S}^+ = \{ (\lambda \vec{x}. T_1 = \lambda \vec{x}. T_2) \mid (\forall \vec{x}[T_1 = T_2]) \in \mathcal{S} \}.$$

Show that the following are equivalent

- (i) $\mathcal{S} \not\models R = S$.
- (ii) $\mathcal{S}^+ \cup \{\lambda x. dxxab = \lambda x.a, dRSab = b\}$ is consistent.

Conclude that the consistency problem for finite sets of equations with constants is Π_1^0 -complete (in contrast to the decidability of finite sets of pure equations).

- 4E.4. (Undecidability of second-order unification) Consider the unification problem

$$Fx_1 \cdots x_n = Gx_1 \cdots x_n,$$

where each x_i has a type of rank < 2 . By the theory of reducibility we can assume that $Fx_1 \cdots x_n$ has type $(0 \rightarrow (0 \rightarrow 0)) \rightarrow (0 \rightarrow 0)$ and so by introducing new constants of types 0, and $0 \rightarrow (0 \rightarrow 0)$ we can assume $Fx_1 \cdots x_n$ has type 0. 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

$$\begin{aligned} a, b &: 0 \\ s &: 0 \rightarrow 0 \\ e &: 0 \rightarrow (0 \rightarrow (0 \rightarrow 0)) \end{aligned}$$

The nth numeral is $s^n a$.

- (i) Let $F: 0 \rightarrow 0$. F is said to be affine if $F = \lambda x. s^n x$. N is a numeral if there exists an affine F such that $Fa = N$. Show that F is affine $\Leftrightarrow F(sa) = s(Fa)$.
- (ii) Next show that $L = N + M$ iff there exist affine F and G such that $N = Fa$, $M = Ga$, and $L = F(Ga)$.
- (iii) We can encode a computation of $n * m$ by

$$e(n * m)m(e(n * (m - 1))(m - 1)(\dots(e(n * 1)11)\dots)).$$

Finally show that $L = N * M \Leftrightarrow \exists C, D, U, V \text{ affine and } \exists F, W$

$$\begin{aligned} Fab &= e(Ua)(Va)(Wab) \\ F(Ca)(sa)(e(Ca)(sa)b) &= e(U(Ca))(V(sa))(Fabl) \\ L &= Ua \\ N &= Ca \\ M &= Va \\ &= Da. \end{aligned}$$

- 4E.5. Consider $\Gamma_{n,m} = \{c_1:0, \dots, c_m:0, f_1:1, \dots, f_n:0\}$. 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 $\Sigma = \{a_1, \dots, a_n\}$ consider equations between words over $\Sigma \cup \{X_1, \dots, X_p\}$. The aim is to find for the unknowns \vec{X} words $w_1, \dots, w_p \in \Sigma^*$ such that the equations become syntactic identities. In [Makanin \[1977\]](#) it is proved that this problem is decidable (uniformly in n, p).

- 4E.6. (Decidability of unification of second-order terms) Consider the unification problem $F\vec{x} = G\vec{x}$ of type A with $\text{rk}(A) = 1$. Here we are interested in the case of pure unifiers of any types. Then $A = 1_m = 0^m \rightarrow 0$ for some natural number m . Consider for $i = 1, \dots, m$ 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 2E.9 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

In this Chapter several extensions of $\lambda_{\rightarrow}^{\text{Ch}}$ based on \mathbb{T}^0 are studied. In Section 5A the systems are embedded into classical predicate logic by essentially adding constants δ_A (for each type A) that determine whether for $M, N \in \Lambda_{\rightarrow}^0(A)$ one has $M = N$ or $M \neq N$. In Section 5B a triple of terms π, π_1, π_2 is added, that forms a surjective pairing. In both cases the resulting system becomes undecidable. In Section 5C the set of elements of ground type 0 is denoted by \mathbb{N} and is thought of as consisting of the natural numbers. One does not work with Church numerals but with new constants $0 : \mathbb{N}, S^+ : \mathbb{N} \rightarrow \mathbb{N}$, and $R_A : A \rightarrow (A \rightarrow \mathbb{N} \rightarrow A) \rightarrow \mathbb{N} \rightarrow A$, for all types $A \in \mathbb{T}^0$, denoting respectively zero, successor and the operator for describing primitive recursive functionals. In Section 5D Spector's bar recursive terms are studied. Finally in Section 5E fixed point combinators are added to the base system. This system is closely related to the system known as 'Edinburgh PCF'.

5A. Lambda delta

In this section λ_{\rightarrow}^0 in the form of $\lambda_{\rightarrow}^{\text{Ch}}$ based on \mathbb{T}^0 will be extended by constants δ ($= \delta_{A,B}$), for arbitrary A, B . [Church \[1940\]](#) used this extension to introduce a logical system called "the simple theory of types", based on classical logic. (The system is also referred 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 $a, a' : A, b, b' : B$

$$\begin{aligned}\delta aa'bb' &= b && \text{if } a = a'; \\ &= b' && \text{if } a \neq a'.\end{aligned}$$

Here $M \neq N$ is defined as $\neg(M = N)$, which is $(M = N) \supset K = K_*$. The type of the new constants is as follows

$$\delta_{A,B} : A \rightarrow A \rightarrow B \rightarrow B \rightarrow B.$$

The classical variant of the theory in which each term and variable carries its unique type will be considered only, but we 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

5A.1. DEFINITION. We will define a formal system called *higher order logic*, notation Δ . Terms are elements of $\Lambda_{\rightarrow}^{\text{Ch}}(\delta)$, the set of open typed terms with types from \mathbb{T}^0 , possibly containing constants δ . Formulas are built up from equations between terms of the same type using implication (\supset) and typed quantification ($\forall x^A.\varphi$). Absurdity is defined by $\perp \triangleq (K = K_*)$, where $K \triangleq \lambda x^0y^0.x$, $K_* \triangleq \lambda x^0y^0.y$. and negation by $\neg\varphi \triangleq \varphi \supset \perp$. 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. By contrast to other sections in this book Γ stands for a set of formulas. In Fig. 9 the axioms and rules of Δ are given. There Γ is a set of formulas, and $\text{FV}(\Gamma) = \{x \mid x \in \text{FV}(\varphi), \varphi \in \Gamma\}$. M, N, L, P, Q are terms.

Provability in this system will be denoted by $\Gamma \vdash_{\Delta} \varphi$, or simply by $\Gamma \vdash \varphi$.

5A.2. DEFINITION. The other logical connectives of Δ are introduced in the usual classical manner.

$$\begin{aligned}\varphi \vee \psi &\triangleq \neg\varphi \supset \psi; \\ \varphi \& \psi &\triangleq \neg(\neg\varphi \vee \neg\psi); \\ \exists x^A.\varphi &\triangleq \neg\forall x^A.\neg\varphi.\end{aligned}$$

5A.3. LEMMA. *For all formulas of Δ one has*

$$\perp \vdash \varphi.$$

PROOF. By induction on the structure of φ . If $\varphi \equiv (M = N)$, then observe that by (eta)

$$\begin{aligned}M &= \lambda \vec{x}.M\vec{x} = \lambda \vec{x}.K(M\vec{x})(N\vec{x}), \\ N &= \lambda \vec{x}.N\vec{x} = \lambda \vec{x}.K_*(M\vec{x})(N\vec{x}),\end{aligned}$$

where the \vec{x} are such that the type of $M\vec{x}$ is 0. Hence $\perp \vdash M = N$, since $\perp \equiv (K = K_*)$. If $\varphi \equiv (\psi \supset \chi)$ or $\varphi \equiv \forall x^A.\psi$, then the result follows immediately from the induction hypothesis. ■

5A.4. PROPOSITION. $\delta_{A,B}$ can be defined from $\delta_{A,0}$.

PROOF. Indeed, if we only have $\delta_{A,0}$ (with their properties) and define

$$\delta_{A,B} = \lambda mnpq\vec{x}.\delta_{A,0}mn(p\vec{x})(q\vec{x}),$$

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 0 and the same theory Δ will be obtained.

$\Gamma \vdash (\lambda x.M)N = M[x := N]$	(beta)
$\Gamma \vdash \lambda x.Mx = M, x \notin \text{FV}(M)$	(eta)
$\Gamma \vdash M = M$	(reflexivity)
$\frac{\Gamma \vdash M = N}{\Gamma \vdash N = M}$	(symmetry)
$\frac{\Gamma \vdash M = N, \Gamma \vdash N = L}{\Gamma \vdash M = L}$	(trans)
$\frac{\Gamma \vdash M = N, \Gamma \vdash P = Q}{\Gamma \vdash MP = NQ}$	(cong-app)
$\frac{\Gamma \vdash M = N \quad x \notin \text{FV}(\Gamma)}{\Gamma \vdash \lambda x.M = \lambda x.N}$	(cong-abs)
$\frac{\varphi \in \Gamma}{\Gamma \vdash \varphi}$	(axiom)
$\frac{\Gamma \vdash \varphi \supset \psi \quad \Gamma \vdash \varphi}{\Gamma \vdash \psi}$	(\supset -elim)
$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \supset \psi}$	(\supset -intr)
$\frac{\Gamma \vdash \forall x^A.\varphi \quad M \in \Lambda(A)}{\Gamma \vdash \varphi[x := M]}$	(\forall -elim)
$\frac{\Gamma \vdash \varphi \quad x^A \notin \text{FV}(\Gamma)}{\Gamma \vdash \forall x^A.\varphi}$	(\forall -intr)
$\frac{\Gamma, M \neq N \vdash \perp}{\Gamma \vdash M = N}$	(classical)
$\Gamma \vdash M = N \supset \delta MNPQ = P$	(delta _L)
$\Gamma \vdash M \neq N \supset \delta MNPQ = Q$	(delta _R)

FIGURE 9. Δ : Higher Order Logic

5A.5. PROPOSITION. Suppose that in the formulation of Δ one requires

$$\Gamma, \neg(M = N) \vdash_{\Delta} \perp \Rightarrow \Gamma \vdash_{\Delta} M = N \quad (1)$$

only for terms x, y of type 0. Then (1) holds for terms of all types.

PROOF. By (1) we have $\neg\neg M = N \supset M = N$ for terms of type 0. 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 0. 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 $M = N$. ■

5A.6. 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

$$\frac{\begin{array}{c} [\psi \supset \chi]^1 \quad [\psi]^3 \\ \hline \chi \end{array}}{\frac{\begin{array}{c} \perp \\ \hline \neg(\psi \supset \chi) \end{array}}{\frac{\begin{array}{c} \perp \\ \hline \neg\neg(\psi \supset \chi) \end{array}}{\frac{\begin{array}{c} \perp \\ \hline \neg\neg\chi \end{array}}{\frac{\begin{array}{c} \perp \\ \hline \neg\neg\chi \supset \chi \end{array}}{\frac{\psi \supset \chi}{\neg\neg(\psi \supset \chi) \supset \psi \supset \chi}}}}}}^4 \quad \vdots \quad 3$$

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

$$\frac{\begin{array}{c} [\forall x.\psi(x)]^1 \\ \hline \psi(x) \end{array}}{\frac{\begin{array}{c} \perp \\ \hline \neg\forall x.\psi(x) \end{array}}{\frac{\begin{array}{c} \perp \\ \hline \neg\neg\forall x.\psi(x) \end{array}}{\frac{\begin{array}{c} \perp \\ \hline \neg\neg\psi(x) \end{array}}{\frac{\begin{array}{c} \psi(x) \\ \hline \forall x.\psi(x) \end{array}}{\frac{\psi(x)}{\neg\neg\forall x.\psi(x) \supset \forall x.\psi(x)}}}}}}^3 \quad \vdots$$

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 .

5A.7. PROPOSITION. *The following equations hold universally (for those terms such that the equations make sense).*

$$\begin{aligned}
 \delta MMPQ &= P && (\delta\text{-identity}); \\
 \delta MNPP &= P && (\delta\text{-reflexivity}); \\
 \delta MNMN &= N && (\delta\text{-hypothesis}); \\
 \delta MNPQ &= \delta NMPQ && (\delta\text{-symmetry}); \\
 F(\delta MNPQ) &= \delta MN(FP)(FQ) && (\delta\text{-monotonicity}); \\
 \delta MN(P(\delta MN))(Q(\delta MN)) &= \delta MN(PK)(QK_*) && (\delta\text{-transitivity}).
 \end{aligned}$$

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 $\delta MNPP \neq P \vdash \perp$, but then $\vdash \delta MNPP = P$, by the classical rule. ■

5A.8. DEFINITION. The *equational version* of higher order logic, notation δ , consists of equations between terms of $\Lambda_{\rightarrow}^{\text{Ch}}(\delta)$ of the same type, axiomatized as in Fig. 10. As usual the axioms and rules are assumed to hold universally, i.e. the free variables may be replaced by arbitrary terms. \mathcal{E} denotes a set of equations between terms of the same type. The system δ may be given more conventionally by leaving out all occurrences of $\mathcal{E} \vdash_{\delta}$ and replacing in the rule (cong-abs) the proviso “ $x \notin \text{FV}(\mathcal{E})$ ” 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 .

5A.9. DEFINITION. (i) For an equation $E \equiv (M = N)$ in Δ , write $\mathbf{E.L} \triangleq M$ and $\mathbf{E.R} \triangleq N$.
(ii) Define for a formula φ of Δ the corresponding equation φ^+ as follows.

$$\begin{aligned}
 (M = N)^+ &\triangleq M = N; \\
 (\psi \supset \chi)^+ &\triangleq (\delta(\psi^+.L)(\psi^+.R)(\chi^+.L)(\chi^+.R) = \chi^+.R); \\
 (\forall x.\psi)^+ &\triangleq (\lambda x.\psi^+.L = \lambda x.\psi^+.R).
 \end{aligned}$$

(iii) If Γ is a set of formulas, then $\Gamma^+ \triangleq \{\varphi^+ \mid \varphi \in \Gamma\}$.

5A.10. REMARK. So, if $\psi^+ \equiv (M = N)$ and $\chi^+ \equiv (P = Q)$, then

$$\begin{aligned}
 (\psi \supset \chi)^+ &= (\delta MNPQ = Q); \\
 (\neg\psi)^+ &= (\delta MNKK_* = K_*); \\
 (\forall x.\psi)^+ &= (\lambda x.M = \lambda x.N).
 \end{aligned}$$

5A.11. THEOREM. *For every formula φ one has*

$$\vdash_{\Delta} (\varphi \leftrightarrow \varphi^+).$$

$\mathcal{E} \vdash (\lambda x.M)N = M[x := N]$	(β)
$\mathcal{E} \vdash \lambda x.Mx = M, x \notin \text{FV}(M)$	(η)
$\mathcal{E} \vdash M = N, \text{ if } (M = N) \in \mathcal{E}$	(axiom)
$\mathcal{E} \vdash M = M$	(reflexivity)
$\mathcal{E} \vdash M = N$	
$\frac{\mathcal{E} \vdash M = M}{\mathcal{E} \vdash N = M}$	(symmetry)
$\frac{\mathcal{E} \vdash M = N, \mathcal{E} \vdash N = L}{\mathcal{E} \vdash M = L}$	(trans)
$\frac{\mathcal{E} \vdash M = N, \mathcal{E} \vdash P = Q}{\mathcal{E} \vdash MP = NQ}$	(cong-app)
$\frac{\mathcal{E} \vdash M = N}{\mathcal{E} \vdash \lambda x.M = \lambda x.N} \quad x \notin \text{FV}(\mathcal{E})$	(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)
$\mathcal{E} \vdash \delta MN(P(\delta MN))(Q(\delta MN)) = \delta MN(PK)(QK_*)$	(δ -transitivity)

FIGURE 10. δ : Equational version of Δ

PROOF. Note that $(\varphi^+)^+ = \varphi^+$, $(\psi \supset \chi)^+ = (\psi^+ \supset \chi^+)^+$, and $(\forall x.\psi)^+ = (\forall x.\psi^+)^+$. The proof of the theorem is 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 5A.12-5A.18

5A.12. LEMMA. (i) $\vdash_{\delta} \delta MNPQz = \delta MN(Pz)(Qz)$.

(ii) $\vdash_{\delta} \delta MNPQ = \lambda z.\delta MN(Pz)(Qz)$, where z is fresh.

(iii) $\vdash_{\delta} \lambda z.\delta MNPQ = \delta MN(\lambda z.P)(\lambda z.Q)$, where $z \notin \text{FV}(MN)$.

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

- 5A.13. LEMMA. (i) $\delta MNPQ = Q \vdash_{\delta} \delta MNQP = P$.
(ii) $\delta MNPQ = Q, \delta MNQR = R \vdash_{\delta} \delta MNPR = R$.
(iii) $\delta MNPQ = Q, \delta MNUV = V \vdash_{\delta} \delta MN(PU)(QV) = QV$.

PROOF. (i) $P = \delta MNPP$
 $= \delta MN(\mathbf{KPQ})(\mathbf{K}_*QP)$
 $= \delta MN(\delta MNPQ)(\delta MNQP), \text{ by } (\delta\text{-transitivity}),$
 $= \delta MNQ(\delta MNQP), \text{ by assumption,}$
 $= \delta MN(\delta MNQQ)(\delta MNQP), \text{ by } \delta\text{-reflexivity,}$
 $= \delta MN(\mathbf{KQQ})(\mathbf{K}_*QP), \text{ by } (\delta\text{-transitivity}),$
 $= \delta MNQP.$

(ii) $R = \delta MNQR, \text{ by assumption,}$
 $= \delta MN(\delta MNPQ)(\delta MNQR), \text{ by assumption,}$
 $= \delta MN(\mathbf{KPQ})(\mathbf{K}_*QR), \text{ by } (\delta\text{-transitivity}),$
 $= \delta MNPR.$

(iii) Assuming $\delta MNPQ = Q$ and $\delta MNUV = V$ we obtain by (δ -monotonicity) applied twice that

$$\begin{aligned} \delta MN(PU)(QU) &= \delta MNPQU &= QU \\ \delta MN(QU)(QV) &= Q(\delta MNPUV) &= QV. \end{aligned}$$

Hence the result $\delta MN(PU)(QV) = QV$ follows by (ii). ■

5A.14. 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 $\mathcal{E}, M = N \vdash_{\delta} Q = P$, by (symmetry). Hence by the induction hypothesis one has $\mathcal{E} \vdash_{\delta} \delta MNQP = P$. But then by lemma 5A.13(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 5A.13(ii), (iii) or Lemma 5A.12(iii) respectively. ■

5A.15. LEMMA. (i) $\vdash_{\delta} \delta MN(\delta MNPQ)P = P$.

(ii) $\vdash_{\delta} \delta MNQ(\delta MNPQ) = Q$.

PROOF. (i) By (δ -transitivity) one has

$$\delta MN(\delta MNPQ)P = \delta MN(\mathbf{KPQ})P = \delta MNPP = P.$$

(ii) Similarly. ■

- 5A.16. LEMMA. (i) $\vdash_{\delta} \delta KK_* = K_*$;
(ii) $\vdash_{\delta} \delta MNKK_* = \delta MN$;
(iii) $\vdash_{\delta} \delta(\delta MN)K_*PQ = \delta MNQP$;
(iv) $\vdash_{\delta} \delta(\delta MNKK_*)K_*(\delta MNPQ)Q = Q$.

PROOF. (i) $K_* = \delta KK_*KK_*, \text{ by } (\delta\text{-hypothesis}),$
 $= \lambda ab.\delta KK_*(Kab)(K_*ab), \text{ by } (\eta) \text{ and Lemma 5A.12(ii)},$
 $= \lambda ab.\delta KK_*ab$
 $= \delta KK_*, \text{ by } (\eta).$

- (ii) $\delta MNKK_* = \delta MN(\delta MN)(\delta MN)$, by (δ -transitivity),
 $= \delta MN$, by (δ -reflexivity).
- (iii) $\delta MNQP = \delta MN(\delta KK_*PQ)(\delta K_*K_*PQ)$, by (i), (δ -identity),
 $= \delta MN(\delta(\delta MN)K_*PQ)(\delta(\delta MN)K_*PQ)$, by (δ -transitivity),
 $= \delta(\delta MN)K_*PQ$, by (δ -reflexivity).

(iv) By (ii) and (iii) we have

$$\delta(\delta MNKK_*)(K_*(\delta MNPQ))Q = \delta(\delta MN)K_*(\delta MNPQ)Q = \delta MNQ(\delta MNPQ).$$

Therefore we are done by lemma 5A.15(ii). ■

- 5A.17. LEMMA.
- (i) $\delta MN = K \vdash_{\delta} M = N$;
 - (ii) $\delta MNK_*K = K_* \vdash_{\delta} M = N$.
 - (iii) $\delta(\delta MNKK_*)K_*KK_* = K_* \vdash_{\delta} M = N$.

PROOF. (i) $M = KMN = \delta MNM = N$, by assumption and (δ -hypothesis).

(ii) Suppose $\delta MNK_*K = K_*$. Then by Lemma 5A.12(ii) and (δ -hypothesis)

$$M = K_*NM = \delta MNK_*KNM = \delta MN(K_*NM)(KNM) = \delta MNM = N.$$

(iii) By Lemma 5A.16(ii) and (iii)

$$\delta(\delta MNKK_*)K_*KK_* = \delta(\delta MN)K_*KK_* = \delta MNK_*K.$$

Hence by (ii) we are done. ■

Now we are able to prove the conservativity of Δ over δ .

5A.18. THEOREM. *For equations \mathcal{E} , E and formulas Γ, φ of Δ one has the following.*

- (i) $\Gamma \vdash_{\Delta} \varphi \Leftrightarrow \Gamma^+ \vdash_{\delta} \varphi^+$.
- (ii) $\mathcal{E} \vdash_{\Delta} E \Leftrightarrow \mathcal{E} \vdash_{\delta} E$.

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 5A.14.

Cases 6, 7. φ is introduced by a (\forall -elim) or (\forall -intro). Then the result follows easily from the induction hypothesis and axiom (β) or the rule (cong-abs). One needs that $FV(\Gamma) = FV(\Gamma^+)$.

Case 8. $\varphi \equiv (M = N)$ and follows from $\Gamma, M \neq N \vdash_{\Delta} \perp$ using the rule (classical). By the induction hypothesis $\Gamma^+, (M \neq N)^+ \vdash_{\delta} K = K_*$. By the deduction Theorem it follows that $\Gamma^+ \vdash_{\delta} \delta(\delta MNKK_*)K_*KK_* = K_*$. Hence we are done by Lemma 5A.17(iii).

Case 9. φ is the axiom ($M = N \supset \delta MNPQ = P$). Then φ^+ is provable in δ by Lemma 5A.15(i).

Case 10. φ is the axiom ($M \neq N \supset \delta MNPQ = Q$). Then φ^+ is provable in δ by Lemma 5A.16(iv).

(\Leftarrow) By the fact that δ is a subtheory of Δ and theorem 5A.11.

(ii) By (i) and the fact that $E^+ \equiv E$. ■

Logic of order n

In this subsection some results will be sketched but not (completely) proved.

5A.19. 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 $\text{rank}(A) \leq n$.
- (iii) Similarly $\delta(n)$ is the theory δ in which only terms $\delta_{A,B}$ are used with $\text{rank}(A) \leq n$.
- (iv) The *rank* of a formula φ is $\text{rank}(\varphi) = \max\{\text{rank}(\delta) \mid \delta \text{ occurs in } \varphi\}$.

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 such formulas can be eliminated.

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 deduction is not normal, then one can subject it to reduction as follows. This idea is from Prawitz [1965].

$$\frac{\begin{array}{c} \vdots \Sigma \\ \varphi \\ \hline \forall x.\varphi \end{array}}{\varphi[x := M]} \Rightarrow \frac{\begin{array}{c} \vdots \Sigma[x := M] \\ \varphi[x := M] \end{array}}{\varphi[x := M]}$$

$$\frac{\begin{array}{c} \vdots \Sigma_1 \\ [\varphi] \\ \vdots \Sigma_2 \\ \varphi \\ \hline \varphi \supset \psi \end{array}}{\psi} \Rightarrow \frac{\begin{array}{c} \vdots \Sigma_2 \\ [\varphi] \\ \vdots \Sigma_1 \\ \psi \end{array}}{\psi}$$

5A.20. THEOREM. Δ -reduction on deductions is SN. Moreover, each deduction has 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

$$\begin{aligned}\delta mmpq &= p, \\ \delta mnpp &= p, \\ \delta mnmn &= n, \\ \delta mnnpq &= \delta nmpq, \\ f(\delta mnnpq) &= \delta mn(fp)(fq), \\ \delta mm(p(\delta mn))(q(\delta mn)) &= \delta mn(p\mathbf{K})(q\mathbf{K}_*).\end{aligned}$$

(ii) Write $\Gamma_{\delta(n)} \triangleq \{\varphi \in \Gamma_\delta \mid \text{rank}(\varphi) \leq n\}$.

5A.21. PROPOSITION (Deduction theorem II). *Let \mathcal{S} be a set of equations or negations of equations in Δ , such that for $(U = V) \in \mathcal{S}$ or $(U \neq V) \in \mathcal{S}$ one has for the type A of U, V that $\text{rank}(A) \leq n$. Then*

- (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 style as the proof of Proposition 5A.14, but now using the normalization Theorem 5A.20. ■

5A.22. LEMMA. *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 $\delta MN\mathbf{KK}_* = \mathbf{K}_*$. 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. ■

5A.23. THEOREM. $\mathcal{E} \vdash_{\Delta(n)} E \Leftrightarrow \mathcal{E} \vdash_{\delta(n)} E$.

PROOF. (\Rightarrow) By taking $\mathcal{S} = \mathcal{E}$ and $M \equiv N \equiv x$ in Lemma 5A.22(i) one obtains $\mathcal{E} \vdash_{\delta(n)} \delta xxPQ = Q$. Hence $\mathcal{E} \vdash_{\delta(n)} P = Q$, by (δ -identity). (\Leftarrow) Trivial. ■

5A.24. THEOREM. (i) Let $\text{rank}(\mathcal{E}, M = N) \leq 1$. Then

$$\mathcal{E} \vdash_{\Delta} M = N \Leftrightarrow \mathcal{E} \vdash_{\delta(1)} M = N.$$

(ii) Let Γ, A be first-order sentences. Then

$$\Gamma \vdash_{\Delta} A \Leftrightarrow \Gamma \vdash_{\delta(1)} A^+.$$

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 $\Delta(n+1)$.

5B. Surjective pairing

5B.1. DEFINITION. A *pairing* on a set X consists of three maps π, π_1, π_2 such that

$$\pi : X \rightarrow X \rightarrow X$$

$$\pi_i : X \rightarrow X$$

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 one can encode data-structures.

5B.2. REMARK. From a (surjective) pairing one can define $\pi^n : X^n \rightarrow X$, $\pi_i^n : X \rightarrow X$, $1 \leq i \leq n$ such that

$$\begin{aligned}\pi_i^n(\pi^n x_1 \cdots x_n) &= x_i, \quad 1 \leq i \leq n, \\ \pi^n(\pi_1^n x) \cdots (\pi_n^n x) &= x, \quad \text{in case of surjectivity.}\end{aligned}$$

Moreover $\pi = \pi^2$ and $\pi_i = \pi_i^2$, for $1 \leq i \leq 2$.

PROOF. Define

$$\begin{aligned}\pi^1(x) &= x \\ \pi^{n+1}x_1 \cdots x_{n+1} &= \pi(\pi^n x_1 \cdots x_n)x_{n+1} \\ \pi_1^1(x) &= x \\ \pi_i^{n+1}(x) &= \pi_i^n(\pi_1(x)), \quad \text{if } i \leq n, \\ &= \pi_2(x), \quad \text{if } i = n + 1. \blacksquare\end{aligned}$$

Surjective pairing is not typable in untyped λ -calculus and therefore also not in λ_\rightarrow , see Barendregt [1974]. In spite of this in de Vrijer [1989], and later also in Størvring [2006] for the extensional case, it is shown that adding surjective pairing to untyped λ -calculus yields a conservative extension. Moreover normal forms remain unique, see de Vrijer [1987] and Klop and de Vrijer [1989]. By contrast the main results in this section are the following. 1. After adding a surjective pairing to λ_\rightarrow^0 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

$$M \in \mathcal{X} \Leftrightarrow \exists N FMN = GMN.$$

Both results will be proved by introducing Cartesian monoids and studying freely generated ones.

The system λ_{SP}

Inspired by the notion of a surjective pairing we define λ_{SP} as an extension of the simply typed lambda calculus λ_\rightarrow^0 .

5B.3. DEFINITION. (i) The set of *types of λ_{SP}* is simply \mathbb{T}^0 .

(ii) The *terms of λ_{SP}* , notation Λ_{SP} (or $\Lambda_{SP}(A)$ for terms of a certain type A or $\Lambda^0, \Lambda_{SP}^0(A)$ for closed terms), are obtained from λ_\rightarrow^0 by adding to the formation of terms the constants $\pi : 1_2 = 0^2 \rightarrow 0, \pi_1 : 1, \pi_2 : 1$.

(iii) Equality for λ_{SP} is axiomatized by β , η and the following scheme. For all $M, M_1, M_2 : 0$

$$\begin{aligned}\pi_i(\pi M_1 M_2) &= M_i; \\ \pi(\pi_1 M)(\pi_2 M) &= M.\end{aligned}$$

(iv) A notion of reduction SP is introduced on λ_{SP} -terms by the following contraction rules: for all $M, M_1, M_2 : 0$

$$\begin{aligned}\pi_i(\pi M_1 M_2) &\rightarrow M_i; \\ \pi(\pi_1 M)(\pi_2 M) &\rightarrow M.\end{aligned}$$

Usually we will consider SP in combination with $\beta\eta$, obtaining $\beta\eta SP$.

According to a well-known result in [Klop \[1980\]](#) reduction coming from surjective pairing in untyped lambda calculus is not confluent (i.e. does not satisfy the Church-Rosser property). This gave rise to the notion of *left-linearity* in term rewriting, see [Terese \[2003\]](#). We will see below, Proposition 5B.10, that in the present typed case the situation is different.

5B.4. THEOREM. *The conversion relation $=_{\beta\eta SP}$, generated by the notion of reduction $\beta\eta SP$, coincides with that of the theory λ_{SP} .*

PROOF. As usual. ■

For objects of higher type pairing can be defined in terms of π, π_1, π_2 as follows.

5B.5. DEFINITION. For every type $A \in \mathbb{T}$ we define $\pi^A : A \rightarrow A \rightarrow A$, $\pi_i : A \rightarrow A$ as follows, cf. the construction in Proposition 1D.21.

$$\begin{aligned}\pi^0 &\triangleq \pi; \\ \pi_i^0 &\triangleq \pi_i; \\ \pi^{A \rightarrow B} &\triangleq \lambda xy:(A \rightarrow B) \lambda z:A. \pi^B(xz)(yz); \\ \pi_i^{A \rightarrow B} &\triangleq \lambda x:(A \rightarrow B) \lambda z:A. \pi_i^B(xz).\end{aligned}$$

Sometimes we may suppress type annotations in π^A, π_1^A, π_2^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.

5B.6. PROPOSITION. *Let $\pi = \pi^A, \pi_i = \pi_i^A$. Then for $M, M_1, M_2 \in \Lambda_{SP}(A)$*

$$\begin{aligned}\pi(\pi_1 M)(\pi_2 M) &\rightarrow_{\beta\eta SP} M; \\ \pi_i(\pi M_1 M_2) &\rightarrow_{\beta\eta SP} M_i, \quad (i = 1, 2).\end{aligned}$$

PROOF. By induction on the type A . ■

Note that the above reductions may involve more than one step, typically additional $\beta\eta$ -steps.

Inspired by Remark 5B.2 one can show the following.

5B.7. PROPOSITION. Let $A \in \mathbb{T}^0$. Then there exist $\pi^{A,n} : \Lambda_{SP}^\emptyset(A^n \rightarrow A)$, and $\pi_i^{A,n} : \Lambda_{SP}^\emptyset(A \rightarrow A)$, $1 \leq i \leq n$, such that

$$\begin{aligned}\pi_i^{A,n}(\pi^{A,n}M_1 \cdots M_n) &\rightarrow_{\beta\eta SP} M_i, \quad 1 \leq i \leq n, \\ \pi^{A,n}(\pi_1^{A,n}M) \cdots (\pi_n^{A,n}M) &\rightarrow_{\beta\eta SP} M.\end{aligned}$$

The original π, π_1, π_2 can be called $\pi^{0,2}, \pi_1^{0,2}, \pi_2^{0,2}$.

Now we will show that the notion of reduction $\beta\eta SP$ is confluent.

5B.8. LEMMA. *The notion of reduction $\beta\eta SP$ 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 that already did exist, and hence cannot create any redexes. ■

5B.9. LEMMA. (i) *The notion of reduction SP is SN.*

- (ii) *If $M \rightarrow_{\beta\eta SP} N$, then there exists P such that $M \rightarrow_{\beta\eta} P \rightarrow_{SP} N$.*
- (iii) *The notion of reduction $\beta\eta SP$ is SN.*

PROOF. (i) Since SP-reductions are strictly decreasing.

(ii) Show $M \rightarrow_{SP} L \rightarrow_{\beta\eta} N \Rightarrow \exists L' M \rightarrow_{\beta\eta} L' \rightarrow_{\beta\eta SP} N$. Then (ii) follows by a staircase diagram chase.

(iii) By (i), the fact that $\beta\eta$ is SN and a staircase diagram chase, possible by (ii). ■

Now we show that the notion of reduction $\beta\eta SP$ is confluent, in spite of being not left-linear.

5B.10. PROPOSITION. *$\beta\eta SP$ is confluent.*

PROOF. By lemma 5B.9(iii) and Newman's Lemma 5C.8. ■

5B.11. DEFINITION. (i) An *SP-retraction pair from A to B* is a pair of terms $M:A \rightarrow B$ and $N:B \rightarrow A$ such that $N \circ M =_{\beta\eta SP} !_A$.

(ii) A is a *SP-retract* of B , notation $A \triangleleft_{SP} B$, if there is an SP-retraction pair from A to B .

The proof of the following result is left as an exercise to the reader.

5B.12. PROPOSITION. Define types N_n as follows. $N_0 \triangleq 0$ and $N_{n+1} \triangleq N_n \rightarrow N_n$. Then for every type A , one has $A \triangleleft_{SP} N_{\text{rank}(A)}$.

Cartesian monoids

We start with the definition of a Cartesian monoid, introduced in [Scott \[1980\]](#) and, independently, in [Lambek \[1980\]](#).

5B.13. DEFINITION. (i) A *Cartesian monoid* is a structure

$$\mathcal{C} \triangleq \langle \mathcal{M}, *, I, L, R, \langle \cdot, \cdot \rangle \rangle$$

such that $(\mathcal{M}, *, I)$ is a monoid ($*$ is associative and I is a two sided unit), $L, R \in \mathcal{M}$ and $\langle \cdot, \cdot \rangle : \mathcal{M}^2 \rightarrow \mathcal{M}$ and satisfy for all $x, y, z \in \mathcal{M}$

$$\begin{aligned} L * \langle x, y \rangle &= x \\ R * \langle x, y \rangle &= y \\ \langle x, y \rangle * z &= \langle x * z, y * z \rangle \\ \langle L, R \rangle &= I \end{aligned}$$

- (ii) \mathcal{M} is called *trivial* if $L = R$.
- (iii) A map $f : \mathcal{M} \rightarrow \mathcal{M}'$ is a *morphism* if

$$\begin{aligned} f(m * n) &= f(m) * f(n); \\ f(\langle m, n \rangle) &= \langle f(m), f(n) \rangle, \\ f(L) &= L', \\ f(R) &= R'. \end{aligned}$$

Then automatically one has $f(I) = I'$.

Note that if \mathcal{M} is trivial, then it consists of only one element: for all $x, y \in \mathcal{M}$

$$x = L * \langle x, y \rangle = R * \langle x, y \rangle = y.$$

5B.14. 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 $\langle L, R \rangle = \langle L * I, R * I \rangle = I$. ■

5B.15. 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 carrier set the closed $\beta\eta SP$ -terms of type $1 = 0 \rightarrow 0$.

5B.16. DEFINITION. Write for $M, N \in \Lambda_{SP}^\circ(1)$

$$\begin{aligned} \langle M, N \rangle &\triangleq \pi^1 MN; \\ M \circ N &\triangleq \lambda x:0.M(Nx); \\ \mathsf{I} &\triangleq \lambda x:0.x; \\ \mathsf{L} &\triangleq \pi_1^0; \\ \mathsf{R} &\triangleq \pi_2^0. \end{aligned}$$

Define

$$\mathcal{C}^0 = \langle \Lambda_{SP}^\circ(1) / =_{\beta\eta SP}, \circ, \mathsf{I}, \mathsf{L}, \mathsf{R}, \langle \cdot, \cdot \rangle \rangle.$$

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 $1^n \rightarrow 1$.

5B.17. PROPOSITION. \mathcal{C}^0 is a non-trivial Cartesian monoid.

PROOF. For $x, y, z : 1$ the following equations are valid in λ_{SP} .

$$\begin{aligned} \mathsf{I} \circ x &= x; \\ x \circ \mathsf{I} &= x; \\ \mathsf{L} \circ \langle x, y \rangle &= x; \\ \mathsf{R} \circ \langle x, y \rangle &= y; \\ \langle x, y \rangle \circ z &= \langle x \circ z, y \circ z \rangle; \\ \langle \mathsf{L}, \mathsf{R} \rangle &= \mathsf{I}. \blacksquare \end{aligned}$$

The third equation is intuitively right, if we remember that the pairing on type 1 is lifted pointwise from a pairing on type 0; that is, $\langle f, g \rangle = \lambda x. \pi(fx)(gx)$.

5B.18. EXAMPLE. Let $[\cdot, \cdot]$ be any surjective pairing of natural numbers, with left and right projections $l, r : \mathbb{N} \rightarrow \mathbb{N}$. For example, we can take Cantor's well-known bijection¹³ 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

$$\mathcal{N}^1 \triangleq \langle \mathbb{N} \rightarrow \mathbb{N}, I, \circ, l, r, \langle \cdot, \cdot \rangle \rangle.$$

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.

5B.19. DEFINITION. (i) Let T_{CM} be the terms in the signature of Cartesian monoids, i.e. built up from constants $\{I, L, R\}$ and variables, using the binary constructors $\langle \cdot, \cdot \rangle$ and $*$.

(ii) Sometimes we need to be explicit which variables we use and set T_{CM}^n equal to the terms generated from $\{I, L, R\}$ and variables x_1, \dots, x_n , using $\langle \cdot, \cdot \rangle$ and $*$. In particular T_{CM}^0 consists of the closed such terms, without variables.

(iii) Consider the notion of reduction CM on T_{CM} , giving rise to the reduction relations \rightarrow_{CM} and its transitive reflexive closure \rightarrowtail_{CM} , introduced by the contraction rules

$$\begin{aligned} L * \langle M, N \rangle &\rightarrow M \\ R * \langle M, N \rangle &\rightarrow N \\ \langle M, N \rangle * T &\rightarrow \langle M * T, N * T \rangle \\ \langle L, R \rangle &\rightarrow I \\ \langle L * M, R * M \rangle &\rightarrow M \\ I * M &\rightarrow M \\ M * I &\rightarrow M \end{aligned}$$

modulo the associativity axioms (i.e. the terms $M * (N * L)$ and $(M * N) * L$ are considered to be the same), see Terese [2003]. The following result is mentioned in Curien [1993].

5B.20. PROPOSITION. (i) CM is WCR.

- (ii) CM is SN.
- (iii) CM is CR.

¹³A variant of this function is used in Section 5C as a non-surjective pairing function $[x, y] + 1$, such that, deliberately, 0 does not encode a pair. This variant is specified in detail and explained in Figure 12.

PROOF. (i) Examine all critical pairs. Modulo associativity there are many such pairs, but they all converge. Consider, as an example, the following reductions:

$$x * z \leftarrow (L * \langle x, y \rangle) * z = L * (\langle x, y \rangle * z) \rightarrow L * \langle x * z, y * z \rangle \rightarrow x * z.$$

(ii) Interpret CM as integers by putting

$$\begin{aligned} \llbracket x \rrbracket &= 2; \\ \llbracket e \rrbracket &= 2, && \text{if } e \text{ is } L, R \text{ or } I; \\ \llbracket e_1 * e_2 \rrbracket &= \llbracket e_1 \rrbracket \cdot \llbracket e_2 \rrbracket; \\ \llbracket \langle e_1, e_2 \rangle \rrbracket &= \llbracket e_1 \rrbracket + \llbracket e_2 \rrbracket + 1. \end{aligned}$$

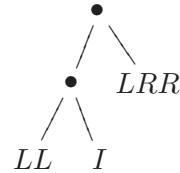
Then $\llbracket \cdot \rrbracket$ preserves associativity and

$$e \rightarrow_{CM} e' \Rightarrow \llbracket e \rrbracket > \llbracket e' \rrbracket.$$

Therefore CM is SN.

(iii) By (i), (ii) and Newman's lemma 5C.8. ■

Closed terms in CM -nf can be represented as binary trees with strings of L, R (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 $\mathcal{F}[x_1, \dots, x_n]$

5B.21. DEFINITION. (i) The closed term model of the theory of Cartesian monoids consists of T_{CM}^0 modulo $=_{CM}$ and is denoted by \mathcal{F} . It is the *free Cartesian monoid* with no generators.

(ii) The *free Cartesian monoid* over the generators $\vec{x} = x_1, \dots, x_n$, notation $\mathcal{F}[\vec{x}]$, is T_{CM}^n modulo $=_{\mathcal{M}}$.

5B.22. 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 $g : \mathcal{F} \rightarrow \mathcal{M}$ to a non-trivial Cartesian monoid \mathcal{M} is injective.

PROOF. (i) We can assume that a, b are in normal form. Seen as trees (not looking at the words over $\{L, R\}$ at the leaves) the a, b can be made congruent by expansions of the form $x \leftarrow \langle L * x, R * x \rangle$. These 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 $\mathcal{F}[x_1, \dots, x_n]$

Now we will show that $\mathcal{F}[x_1, \dots, x_n]$ is finitely generated as a monoid, i.e. from finitely many of its elements using the operation $*$ only.

5B.23. 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} .

$$\begin{aligned} \langle\langle x \rangle \triangleq x; \\ \langle\langle x_1, \dots, x_{n+1} \rangle \triangleq \langle\langle x_1, \dots, x_n \rangle, x_{n+1} \rangle, \quad n > 0; \\ \langle x \rangle \triangleq x; \\ \langle x_1, \dots, x_{n+1} \rangle \triangleq \langle x_1, \langle x_2, \dots, x_{n+1} \rangle \rangle, \quad n > 0. \end{aligned}$$

5B.24. DEFINITION. (i) For $\mathcal{H} \subseteq \mathcal{F}$ let $[\mathcal{H}]$ be the submonoid of \mathcal{F} generated by \mathcal{H} using the operation $*$.

(ii) Define the finite subset $\mathcal{G} \subseteq \mathcal{F}$ as follows.

$$\mathcal{G} \triangleq \{\langle X * L, Y * L * R, Z * R * R \rangle \mid X, Y, Z \in \{L, R, I\}\} \cup \{\langle I, I, I \rangle\}.$$

We will show that $[\mathcal{G}] = \mathcal{F}$.

5B.25. LEMMA. Define a **string** to be an expression of the form $X_1 * \dots * X_n$, with $X_i \in \{L, R, I\}$. Then for all strings s, s_1, s_2, s_3 one has the following.

- (i) $\langle s_1, s_2, s_3 \rangle \in [\mathcal{G}]$.
- (ii) $s \in [\mathcal{G}]$.

PROOF. (i) Note that

$$\langle X * L, Y * L * R, Z * R * R \rangle * \langle s_1, s_2, s_3 \rangle = \langle X * s_1, Y * s_2, Z * s_3 \rangle.$$

Hence, starting from $\langle I, I, I \rangle \in \mathcal{G}$ every triple of strings can be generated because the X, Y, Z range over $\{L, R, I\}$.

(ii) Notice that

$$\begin{aligned} s &= \langle L, R \rangle * s \\ &= \langle L * s, R * s \rangle \\ &= \langle L * s, \langle L, R \rangle * R * s \rangle \\ &= \langle L * s, L * R * s, R * R * s \rangle, \end{aligned}$$

which is in $[\mathcal{G}]$ by (i). ■

5B.26. LEMMA. Let $e_1, \dots, e_n \in \mathcal{F}$. Suppose $\langle\langle e_1, \dots, e_n \rangle\rangle \in [\mathcal{G}]$. Then

- (i) $e_i \in [\mathcal{G}]$, for $1 \leq i \leq n$.
- (ii) $\langle\langle e_1, \dots, e_n, \langle e_i, e_j \rangle \rangle \in [\mathcal{G}]$ for $0 \leq i, j \leq n$.
- (iii) $\langle\langle e_1, \dots, e_n, X * e_i \rangle\rangle \in [\mathcal{G}]$ for $X \in \{L, R, I\}$.

PROOF. (i) By Lemma 5B.25(ii) one has $F_1 \equiv L^{(n-1)} \in [\mathcal{G}]$ and $F_i \equiv R * L^{(n-i)} \in [\mathcal{G}]$. Hence

$$\begin{aligned} e_1 &= F_1 * \langle\langle e_1, \dots, e_n \rangle\rangle \in [\mathcal{G}]; \\ e_i &= F_i * \langle\langle e_1, \dots, e_n \rangle\rangle \in [\mathcal{G}], \quad \text{for } i = 2, \dots, n. \end{aligned}$$

(ii) By Lemma 5B.25(i) one has $\langle I, \langle F_i, F_j \rangle \rangle = \langle I, F_i, F_j \rangle \in [\mathcal{G}]$. Hence

$$\langle\langle e_1, \dots, e_n, \langle e_i, e_j \rangle \rangle = \langle I, \langle F_i, F_j \rangle \rangle * \langle\langle e_1, \dots, e_n \rangle\rangle \in [\mathcal{G}].$$

(iii) Similarly $\langle\langle e_1, \dots, e_n, X * e_i \rangle\rangle = \langle I, X * F_i \rangle * \langle\langle e_1, \dots, e_n \rangle\rangle \in [\mathcal{G}]$. ■

5B.27. THEOREM. As a monoid, \mathcal{F} is finitely generated. In fact $\mathcal{F} = [\mathcal{G}]$.

PROOF. We have $e \in \mathcal{F}$ iff there is a sequence $e_1 \equiv L, e_2 \equiv R, e_3 \equiv I, \dots, e_n \equiv e$ 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 $X \in \{L, R, I\}$.

By Lemma 5B.25(i) we have $\langle\langle e_1, e_2, e_3 \rangle\rangle \in [\mathcal{G}]$. By Lemma 5B.26(ii), (iii) it follows that

$$\langle\langle e_1, e_2, e_3, \dots, e_n \rangle\rangle \in [\mathcal{G}].$$

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 B[1984].

5B.28. COROLLARY. (i) Let \mathcal{M} be a finitely generated Cartesian monoid. Then \mathcal{M} is generated by two of its elements.

(ii) $\mathcal{F}[x_1, \dots, x_n]$ is generated by two elements.

PROOF. (i) Let $\mathcal{G} = \{g_1, \dots, g_n\}$ be the set of generators of \mathcal{M} . Then \mathcal{G} and hence \mathcal{M} is generated by R and $\langle\langle g_1, \dots, g_n, L \rangle\rangle$.

(ii) $\mathcal{F}[\vec{x}]$ is generated by \mathcal{G} and the \vec{x} , hence by (i) by two elements. ■

Invertibility in \mathcal{F}

5B.29. DEFINITION. (i) Let \mathcal{L} (\mathcal{R}) be the submonoid of the right (left) invertible elements of \mathcal{F}

$$\begin{aligned} \mathcal{L} &\triangleq \{a \in \mathcal{F} \mid \exists b \in \mathcal{F} \ b * a = I\}; \\ \mathcal{R} &\triangleq \{a \in \mathcal{F} \mid \exists b \in \mathcal{F} \ a * b = I\}. \end{aligned}$$

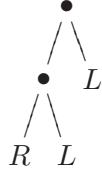
(ii) Let \mathcal{I} be the subgroup of \mathcal{F} consisting of invertible elements

$$\mathcal{I} \triangleq \{a \in \mathcal{F} \mid \exists b \in \mathcal{F} \ a * b = b * a = I\}.$$

It is easy to see that $\mathcal{I} = \mathcal{L} \cap \mathcal{R}$. Indeed, if $a \in \mathcal{L} \cap \mathcal{R}$, then there are $b, b' \in \mathcal{F}$ such that $b * a = I = a * b'$. But then $b = b * a * b' = b'$, so $a \in \mathcal{I}$. The converse is trivial.

5B.30. EXAMPLES. (i) $L, R \in \mathcal{R}$, since both have the right inverse $\langle I, I \rangle$.

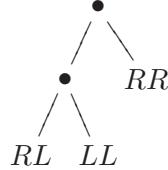
(ii) The element $a = \langle \langle R, L \rangle, L \rangle$ having as ‘tree’



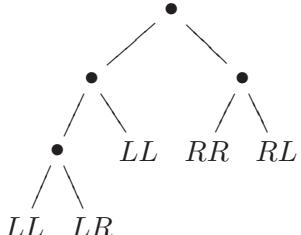
has as left inverse $b = \langle R, LL \rangle$, where we do not write the $*$ in strings.

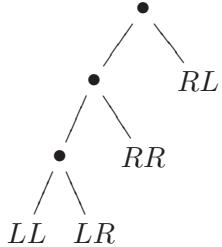
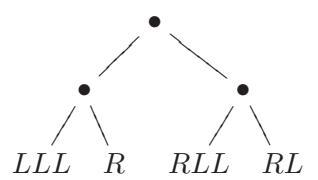
(iii) The element  has no left inverse, since “ R cannot be obtained”.

(iv) The element $a = \langle \langle RL, LL \rangle, RR \rangle$ having the following tree



has the following right inverse $b = \langle \langle RLb, LLb \rangle, RRb \rangle = \langle \langle LL, RL \rangle, R \rangle = \langle L, R \rangle = I$.

(v) The element  has no right inverse, as “ LL occurs twice”.

(vi) The element  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.

5B.31. 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} .

5B.32. DEFINITION. $B_n \triangleq \langle LR^0, \dots, LR^{n-1}, LLR^n, RLR^n, RR^n \rangle;$
 $C_0 \triangleq \langle R, L \rangle,$
 $C_{n+1} \triangleq \langle LR^0, \dots, LR^{n-1}, LRR^n, LR^n, RRR^n \rangle.$

5B.33. PROPOSITION. (i) \mathcal{I} is the subgroup of \mathcal{F} generated (using $*$ and $^{-1}$) by

$$\{B_n \mid n \in \mathbb{N}\} \cup \{C_n \mid n \in \mathbb{N}\}.$$

(ii) $\mathcal{R} = [\{L\} \cup \mathcal{I}] = [\{R\} \cup \mathcal{I}],$ where $[\cdot]$ is defined in Definition 5B.24.

PROOF. (i) In fact $\mathcal{I} = [\{B_0, B_0^{-1}, B_1, B_1^{-1}, C_0, C_1\}]$. Here $[\mathcal{H}]$ is the subset generated from \mathcal{H} using only $*$. Do Exercise 5F.15.

(ii) By Proposition 5B.31. ■

5B.34. REMARK. (i) The B_n alone generate the so-called *Thompson-Freyd-Heller group*, see exercise 5F.14(iv).

(ii) A related group consisting of λ -terms is $G(\boldsymbol{\lambda}\boldsymbol{\eta})$ consisting of invertible closed untyped lambda terms modulo $\beta\eta$ -conversion, see B84, Section 21.3.

5B.35. 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 , with $1 \leq i \leq n$, set

$$h_i = \langle \langle R^m, \dots, R^m, I \rangle, R^m \rangle$$

where the right-associative $\langle R^m, \dots, R^m, I \rangle$ -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.

$$\begin{aligned} d_r(\vec{w}) &\triangleq 0, & \text{if } \vec{w} \text{ is a string of } L, R\text{'s;} \\ d_r(\langle t, s \rangle) &\triangleq d_r(s) + 1. \end{aligned}$$

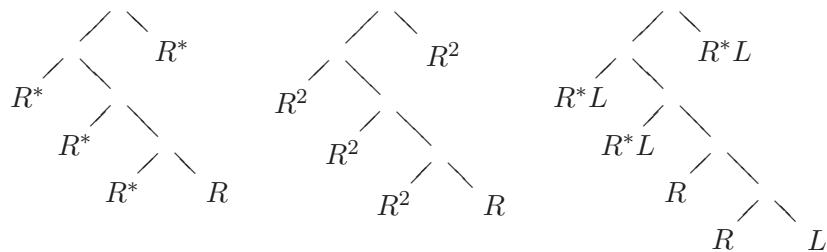
Note that if t is a normal member of \mathcal{F} and $d_r(t) < m$, then

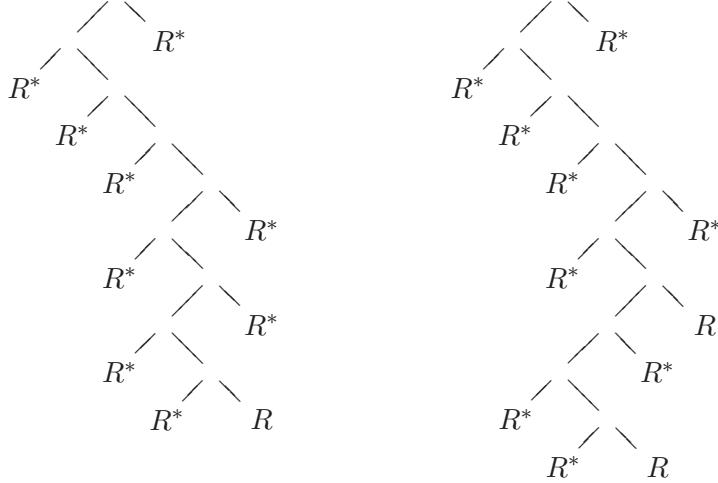
$$h_i * t =_{\text{CM}} \langle \langle t', \dots, t', t \rangle \rangle, t' \rangle,$$

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.

$$\begin{aligned} f_1(\vec{x}) &= x_3 R; \\ f_2(\vec{x}) &= \langle \langle R^2, R^2, R^2, R \rangle \rangle, R^2 \rangle; \\ f_3(\vec{x}) &= x_2 \langle R, R, L \rangle; \\ f_4(\vec{x}) &= x_3 x_1 x_2 R; \\ f_5(\vec{x}) &= x_3 x_1 \langle x_2 R, R \rangle. \end{aligned}$$

Then $f_1(\vec{h}), \dots, f_5(\vec{h})$ 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 $\mathcal{C}^0 = \langle \Lambda_{SP}^\circ(1) / =_{\beta\eta SP}, \circ, I, L, R, \langle \cdot, \cdot \rangle \rangle$.

5B.36. PROPOSITION. *There is a surjective homomorphism $h : \mathcal{F} \rightarrow \mathcal{C}^0$.*

PROOF. If $M : 1$ is a closed term and in long $\beta\eta SP$ 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 $M \equiv I$, $M = \langle \lambda a.X_1, \lambda a.X_2 \rangle$, $M = L \circ (\lambda a.X)$ or $M = R \circ (\lambda a.X)$, 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 $I, L, R, \langle \cdot, \cdot \rangle, \circ$. Thus the elements of \mathcal{C}^0 are generated from $\{I, \circ, L, R, \langle \cdot, \cdot \rangle\}$ in an algebraic way. Now define

$$\begin{aligned} h(I) &= I; \\ h(L) &= L; \\ h(R) &= R; \\ h(\langle a, b \rangle) &= \langle h(a), h(b) \rangle; \\ h(a * b) &= h(a) \circ h(b). \end{aligned}$$

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.

5B.37. THEOREM. $\mathcal{F} \cong \mathcal{C}^0$.

PROOF 1. We will show that the homomorphism h in Proposition 5B.36 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

$$a \neq b \Rightarrow a^{\text{nf}} \not\equiv b^{\text{nf}} \Rightarrow h(a^{\text{nf}}) \neq h(b^{\text{nf}}) \Rightarrow h(a) \neq h(b),$$

so h is injective. ■₁

PROOF 2. By Proposition 5B.22. ■₂

The structure \mathcal{C}^0 will be generalized as follows.

5B.38. DEFINITION. Consider the type $1^n \rightarrow 1 = (0 \rightarrow 0)^n \rightarrow 0 \rightarrow 0$. Define

$$\mathcal{C}^n \triangleq \langle \Lambda_{SP}^\circ(1^n \rightarrow 1) / =_{\beta\eta SP}, I_n, L_n, R_n, \circ_n, \langle -, - \rangle_n \rangle,$$

where writing $\vec{x} = x_1, \dots, x_n : 1$

$$\begin{aligned} \langle M, N \rangle_n &\triangleq \lambda \vec{x}. \langle M \vec{x}, N \vec{x} \rangle; \\ M \circ_n N &\triangleq \lambda \vec{x}. (M \vec{x}) \circ (N \vec{x}); \\ I_n &\triangleq \lambda \vec{x}. I; \\ L_n &\triangleq \lambda \vec{x}. L; \\ R_n &\triangleq \lambda \vec{x}. R. \end{aligned}$$

5B.39. PROPOSITION. \mathcal{C}^n is a non-trivial Cartesian monoid.

PROOF. Easy. ■

5B.40. PROPOSITION. $\mathcal{C}^n \cong \mathcal{F}[x_1, \dots, x_n]$.

PROOF. As before, let $h_n : \mathcal{F}[\vec{x}] \rightarrow \mathcal{C}^n$ be induced by

$$\begin{aligned} h_n(x_i) &= \lambda \vec{x} \lambda z : 0. x_i z &= \lambda \vec{x}. x_i; \\ h_n(I) &= \lambda \vec{x} \lambda z : 0. z &= I_n; \\ h_n(L) &= \lambda \vec{x} \lambda z : 0. \pi_1 z &= L_n; \\ h_n(R) &= \lambda \vec{x} \lambda z : 0. \pi_2 z &= R_n; \\ h_n(\langle s, t \rangle) &= \lambda \vec{x} \lambda z : 0. \pi(s \vec{x} z)(t \vec{x} z) &= \langle h_n(s), h_n(t) \rangle_n. \end{aligned}$$

As before one can show that this is an isomorphism. ■

In the sequel an important case is $n = 1$, i.e. $\mathcal{C}^{1 \rightarrow 1} \cong \mathcal{F}[x]$.

Hilbert-Post completeness of $\lambda_{\rightarrow} SP$

The claim that an equation $M = N$ is either a $\beta\eta SP$ convertibility or inconsistent is proved in two steps. First it is proved for the type $1 \rightarrow 1$ 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.

5B.41. PROPOSITION. (i) Let $M, N \in \Lambda_{SP}^\circ(1)$. Then

$$M \neq_{\beta\eta SP} N \Rightarrow M \#_{\beta\eta SP} N.$$

(ii) The same holds for $M, N \in \Lambda_{SP}^\circ(1 \rightarrow 1)$.

PROOF. (i) Since $\mathcal{F} \cong \mathcal{C}^0 = \Lambda_{SP}^\circ(1)$, by Theorem 5B.37, this follows from Proposition 5B.22(i).

(ii) If $M, N \in \Lambda_{SP}^\circ(1 \rightarrow 1)$, then

$$\begin{aligned} M \neq N &\Rightarrow \lambda f:1.Mf \neq \lambda f:1.Nf \\ &\Rightarrow Mf \neq Nf \\ &\Rightarrow MF \neq NF, && \text{for some } F \in \Lambda_{SP}^\circ(1), \text{ by 5B.35,} \\ &\Rightarrow MF \# NF, && \text{by (i) as } MF, NF \in \Lambda_{SP}^\circ(1), \\ &\Rightarrow M \# N. \blacksquare \end{aligned}$$

We now want to generalize this last result for all types by using type reducibility in the context of λ_{SP} .

5B.42. DEFINITION. Let $A, B \in \mathbb{T}$. We say that A is $\beta\eta SP$ -reducible to B , notation

$$A \leq_{\beta\eta SP} B,$$

if there exists $\Phi : A \rightarrow B$ such that for any closed $N_1, N_2 : A$

$$N_1 = N_2 \Leftrightarrow \Phi N_1 = \Phi N_2.$$

5B.43. PROPOSITION. For each type A one has $A \leq_{\beta\eta SP} 1 \rightarrow 1$.

PROOF. We can copy the proof of 3D.8 to obtain $A \leq_{\beta\eta SP} 1_2 \rightarrow 0 \rightarrow 0$. Moreover, by

$$\lambda u x a. u(\lambda z_1 z_2. x(\pi(xz_1)(xz_2)))a$$

one has $1_2 \rightarrow 0 \rightarrow 0 \leq_{\beta\eta SP} 1 \rightarrow 1$. \blacksquare

5B.44. COROLLARY. Let $A \in \mathbb{T}$ and $M, N \in \Lambda_{SP}^\circ$. Then

$$M \neq_{\beta\eta SP} N \Rightarrow M \#_{\beta\eta SP} N.$$

PROOF. Let $A \leq_{\beta\eta SP} 1 \rightarrow 1$ using Φ . Then

$$\begin{aligned} M \neq N &\Rightarrow \Phi M \neq \Phi N \\ &\Rightarrow \Phi M \# \Phi N, \text{ by corollary 5B.41(ii),} \\ &\Rightarrow M \# N. \blacksquare \end{aligned}$$

We obtain the following Hilbert-Post completeness theorem.

5B.45. THEOREM. Let \mathcal{M} be a model of λ_{SP} . For any type A and closed terms $M, N \in \Lambda^\circ(A)$ the following are equivalent.

- (i) $M =_{\beta\eta SP} N$;
- (ii) $\mathcal{M} \models M = N$;
- (iii) $\lambda_{SP} \cup \{M = N\}$ is consistent.

PROOF. ((i) \Rightarrow (ii)) By soundness. ((ii) \Rightarrow (iii)) Since truth implies consistency. ((iii) \Rightarrow (i)) By corollary 5B.44. \blacksquare

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

5B.46. DEFINITION. Let $R \subseteq \Lambda_{SP}^\varnothing(A_1) \times \cdots \times \Lambda_{SP}^\varnothing(A_n)$ be an n-ary relation.

(i) R is called *equational* if

$$\exists B \in \mathbb{T}^0 \exists M, N \in \Lambda_{SP}^\varnothing(A_1 \rightarrow \cdots \rightarrow A_n \rightarrow B) \forall \vec{F}$$

$$R(F_1, \dots, F_n) \Leftrightarrow MF_1 \cdots F_n = NF_1 \cdots F_n. \quad (1)$$

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

$$R(\vec{F}) \Leftrightarrow \exists \vec{G} S(\vec{F}, \vec{G})$$

(iii) R is called *Diophantine* if it is the projection of an equational relation.

Note that equational 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 recursively enumerable. Our main result will be the converse. The proof occupies 5B.47-5B.57.

5B.47. PROPOSITION. (i) *Equational relations are closed under substitution of lambda definable functions. This means that if R is equational and R' is defined by*

$$R'(\vec{F}) \stackrel{\Delta}{\Leftrightarrow} R(H_1 \vec{F}, \dots, H_n \vec{F}),$$

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. E.g.

$$\begin{aligned} M_1 \vec{F} = N_1 \vec{F} \& M_2 \vec{F} = N_2 \vec{F} \Leftrightarrow \pi(M_1 \vec{F})(M_2 \vec{F}) = \pi(N_1 \vec{F})(N_2 \vec{F}) \\ \Leftrightarrow M \vec{F} = N \vec{F}, \end{aligned}$$

with $M \equiv \lambda \vec{f}. \pi(M_1 \vec{f})(M_2 \vec{f})$ and N is similarly defined.

(iii) By dummy projections.

(iv) By some easy logical manipulations. E.g. let

$$R_i(\vec{F}) \Leftrightarrow \exists \vec{G}_i. M_i \vec{G}_i \vec{F} = N_i \vec{G}_i \vec{F}.$$

Then

$$R_1(\vec{F}) \& R_2(\vec{F}) \Leftrightarrow \exists \vec{G}_1 \vec{G}_2. [M_1 \vec{G}_1 \vec{F} = N_1 \vec{G}_1 \vec{F} \& M_2 \vec{G}_2 \vec{F} = N_2 \vec{G}_2 \vec{F}]$$

and we can use (i). ■

5B.48. LEMMA. Let $\Phi_i : A_i \leq_{SP} (1 \rightarrow 1)$ and let $R \subseteq \prod_{i=1}^n \Lambda_{SP}^\varnothing(A_i)$ be $=$ -closed coordinatewise. Define $R^\Phi \subseteq \Lambda_{SP}^\varnothing(1 \rightarrow 1)^n$ by

$$R^\Phi(G_1, \dots, G_n) \Leftrightarrow \exists F_1 \cdots F_n [\Phi_1 F_1 = G_1 \& \cdots \Phi_n F_n = G_n \& R(F_1, \dots, F_n)].$$

We have the following.

(i) If R^Φ is Diophantine, then R is Diophantine.

(ii) If R^Φ is re, then R is re.

PROOF. (i) By Proposition 5B.47(iv), noting that

$$R(F_1, \dots, F_n) \Leftrightarrow R^\Phi(\Phi_1 F_1, \dots, \Phi_n F_n).$$

(ii) Similarly. ■

From Proposition 5B.7 we can assume without loss of generality that $n = 1$ in Diophantine equations.

5B.49. LEMMA. Let $R \subseteq (\Lambda_{SP}^\emptyset(1 \rightarrow 1))^n$ closed under $=$. Define $R^\wedge \subseteq \Lambda_{SP}^\emptyset(1 \rightarrow 1)$ by

$$R^\wedge(F) \Leftrightarrow R(\pi_1^{1 \rightarrow 1,n}(F), \dots, \pi_n^{1 \rightarrow 1,n}(F)).$$

Then

- (i) R is Diophantine iff R^\wedge is Diophantine.
- (ii) R is re iff R^\wedge is re.

PROOF. By Proposition 5B.47(i) and the pairing functions $\pi^{1 \rightarrow 1,n}$. ■

Note that

$$R(F_1, \dots, F_n) \Leftrightarrow R^\wedge(\pi^{1 \rightarrow 1,n} F_1 \dots F_n).$$

5B.50. COROLLARY. In order to prove that every re relation $R \subseteq \prod_{i=1}^n \Lambda_{SP}^\emptyset(A_i)$ that is closed under $=_{\beta\eta SP}$ is Diophantine, it suffices to do this just for such $R \subseteq \Lambda_{SP}^\emptyset(1 \rightarrow 1)$.

PROOF. By the previous two lemmas. ■

So now we are interested in recursively enumerable subsets of $\Lambda_{SP}^\emptyset(1 \rightarrow 1)$ closed under $=_{\beta\eta SP}$. Since

$$(\mathcal{T}_{CM}^1 / =_{CM}) = \mathcal{F}[x] \cong \mathcal{C}^1 = (\Lambda_{SP}^\emptyset(1 \rightarrow 1) / =_{\beta\eta SP})$$

one can shift attention to relations on \mathcal{T}_{CM}^1 closed under $=_{CM}$. We say loosely 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 $\mathcal{C}^1 \cong \mathcal{F}[x]$).

5B.51. DEFINITION. A k -ary relation R on $\mathcal{F}[\vec{x}]$ is called *Diophantine* if there exist $s(u_1, \dots, u_k, \vec{v}), t(u_1, \dots, u_k, \vec{v}) \in \mathcal{F}[\vec{u}, \vec{v}]$ such that

$$R(f_1[\vec{x}], \dots, f_k[\vec{x}]) \Leftrightarrow \exists \vec{v} \in \mathcal{F}[\vec{x}] . s(f_1[\vec{x}], \dots, f_k[\vec{x}], \vec{v}) = t(f_1[\vec{x}], \dots, f_k[\vec{x}], \vec{v}).$$

The isomorphism $h_n : \mathcal{F}[\vec{x}] \rightarrow \mathcal{C}^n$ given by Proposition 5B.38 induces an isomorphism

$$h_n^k : (\mathcal{F}[\vec{x}])^k \rightarrow (\mathcal{C}^n)^k.$$

Diophantine relations on \mathcal{F} are closed under conjunction as before.

5B.52. PROPOSITION (Transfer lemma). (i) Let $X \subseteq (\mathcal{F}[x_1, \dots, x_n])^k$ be equational (Diophantine). Then $h_n^k(X) \subseteq (\mathcal{C}^n)^k$ is equational (Diophantine), respectively.

(ii) Let $X \subseteq (\mathcal{C}^n)^k$ be re and closed under $=_{\beta\eta SP}$. Then $(h_n^k)^{-1}(X) \subseteq (\mathcal{F}[x_1, \dots, x_n])^k$ is re and closed under $=_{CM}$.

5B.53. COROLLARY. In order to prove that every re relation on \mathcal{C}^1 closed under $=_{\beta\eta SP}$ is Diophantine it suffices to show that every re relation on $\mathcal{F}[x]$ closed under $=_{CM}$ is Diophantine.

Before proving that every $=$ -closed 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 Matijasevič's solution to Hilbert's 10th 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.

5B.54. DEFINITION. Write $s_0 \triangleq I$, $s_{n+1} \triangleq R^{n+1}$, elements of \mathcal{F} . The set of *numerals* in \mathcal{F} is defined by

$$\mathcal{N} \triangleq \{s_n \mid n \in \mathbb{N}\}.$$

We have the following.

5B.55. PROPOSITION. $f \in \mathcal{N} \Leftrightarrow f * R = R * f$.

PROOF. This is because if f is normal and $f * R = R * f$, then the binary tree part of f must be trivial, i.e. f must be a string of L 's and R 's, therefore consists of only R 's. ■

5B.56. DEFINITION. A sequence of k -ary relations $R_n \subseteq \mathcal{F}$ is called *Diophantine uniformly in n* if there is a $k + 1$ -ary Diophantine relation $P \subseteq \mathcal{F}^{k+1}$ such that

$$R_n(\vec{u}) \Leftrightarrow P(s_n, \vec{u}).$$

Now we build up a toolkit of Diophantine relations on \mathcal{F} .

1. \mathcal{N} is equational (hence Diophantine).

PROOF. In 5B.55 it was proved that

$$f \in \mathcal{N} \Leftrightarrow f * R = R * f. \blacksquare$$

2. The sets $\mathcal{F} * L$, $\mathcal{F} * R \subseteq \mathcal{F}$ and $\{L, R\}$ are equational. In fact one has

$$\begin{aligned} \text{(i)} \quad f \in \mathcal{F} * L &\Leftrightarrow f * \langle L, L \rangle = f. \\ \text{(ii)} \quad f \in \mathcal{F} * R &\Leftrightarrow f * \langle R, R \rangle = f. \\ \text{(iii)} \quad f \in \{L, R\} &\Leftrightarrow f * \langle I, I \rangle = I. \end{aligned}$$

PROOF.

(i) 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 $f = f * \langle I, I \rangle * L \in \mathcal{F} * L$.

(ii) Similarly.

(iii) (\Leftarrow) By distinguishing the possible shapes of the nf of f . ■

3. Notation

$$\begin{aligned} [] &\triangleq R; \\ [f_0, \dots, f_{n-1}] &\triangleq \langle f_0 * L, \dots, f_{n-1} * L, R \rangle, \quad \text{if } n > 0. \end{aligned}$$

One easily sees that $[f_0, \dots, f_{n-1}] * [I, f_n] = [f_0, \dots, f_n]$. Write

$$\text{Aux}_n(f) \triangleq [f, f * R, \dots, f * R^{n-1}].$$

Then the relations $h = \text{Aux}_n(f)$ are Diophantine uniformly in n .

PROOF. Indeed,

$$h = \text{Aux}_n(f) \Leftrightarrow R^n * h = R \& h = R * h * \langle \langle L, L \rangle, \langle f * R^{n-1} * L, R \rangle \rangle.$$

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 $h_n = R$ and $h = R * h * \langle \langle L, L \rangle, \langle f * R^{n-1} * L, R \rangle \rangle = R * h * _$, 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 $h = \text{Aux}_n(f)$ ■.

4. Write $\text{Seq}_n(f) \stackrel{\Delta}{\iff} f = [f_0, \dots, f_{n-1}]$, for some f_0, \dots, f_{n-1} . Then Seq_n is Diophantine uniformly in n .

PROOF. One has $\text{Seq}_n(f)$ iff

$$R^n * f = R \& \text{Aux}_n(L) * \langle I, L \rangle * f = \text{Aux}_n(L) * \langle I, L \rangle * f * \langle L, L \rangle,$$

as can be proved similarly (use 2(i)). ■

5. Define

$$\text{Cp}_n(f) \stackrel{\Delta}{=} [f, \dots, f], \text{ (} n \text{ times } f \text{).}$$

(By default $\text{Cp}_0(f) \stackrel{\Delta}{=} [\] \stackrel{\Delta}{=} R$.) Then $\text{Cp}_n(f) = g$ is Diophantine uniformly in n .

PROOF. $\text{Cp}_n(f) = g$ iff

$$\text{Seq}_n(g) \& g = R * g * \langle L, f * L, R \rangle. \blacksquare$$

6. Let $\text{Pow}_n(f) \stackrel{\Delta}{=} f^n$. Then $\text{Pow}_n(f) = g$ is Diophantine uniformly in n .

PROOF. One has $\text{Pow}_n(f) = g$ iff

$$\exists h [\text{Seq}_n(h) \& h = R * h * \langle f * L, f * L, R \rangle \& L * h = g].$$

This can be proved in a similar way (it helps to realize that h has to be of the form $h = [f^n, \dots, f^1]$). ■

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) $P_+(s_n, s_m, s_k) \Leftrightarrow n + m = k$.
- (2) $P_\times(s_n, s_m, s_k) \Leftrightarrow n \cdot m = k$.

PROOF. (i) Define $P_+(x, y, z) \Leftrightarrow x * y = z$. This relation is Diophantine and works: $R^n * R^m = R^k \Leftrightarrow R^{n+m} = R^k \Leftrightarrow n + m = k$.

(ii) Let $\text{Pow}_n(f) = g \Leftrightarrow P(s_n, f, g)$, with P Diophantine. Then choose $P_\times = P$. ■

8. Let $X \subseteq \mathbb{N}$ be a recursively enumerable set of natural numbers. Then $\{s_n \mid n \in X\}$ is Diophantine.

PROOF. By 7 and the famous Theorem of Matiyasevič [1972]. ■

9. Define $\text{Seq}_n^\mathcal{N} \stackrel{\Delta}{=} \{[s_{m_0}, \dots, s_{m_{n-1}}] \mid m_0, \dots, m_{n-1} \in \mathbb{N}\}$. Then the relation $f \in \text{Seq}_n^\mathcal{N}$ is Diophantine uniformly in n .

PROOF. Indeed, $f \in \text{Seq}_n^\mathcal{N}$ iff

$$\text{Seq}_n(f) \& f * \langle R * L, R \rangle = \text{Aux}_n(R * L) * \langle I, R^n \rangle * f. \blacksquare$$

10. Let $f = [f_0, \dots, f_{n-1}]$ and $g = [g_0, \dots, g_{n-1}]$. We write

$$f \# g = [f_0 * g_0, \dots, f_{n-1} * g_{n-1}].$$

Then there exists a Diophantine relation P such that for arbitrary n and $f, g \in \text{Seq}_n$ one has

$$P(f, g, h) \Leftrightarrow h = f \# g.$$

PROOF. Let

$$\text{Cmp}_n(f) = [L * f, L * R * f * R, \dots, L * R^{n-1} * f * R^{n-1}].$$

Then $g = \text{Cmp}_n(f)$ is Diophantine uniformly in n .

This requires some work. One has by the by now familiar technique

$$\begin{aligned} \text{Cmp}_n(f) = g &\Leftrightarrow \\ \exists h_1, h_2, h_3 &[\\ \text{Seq}_n(h_1) \& \& f = h_1 * \langle I, R^n * f \rangle \\ \text{Seq}_{n^2}(h_2) \& \& h_2 = R^n * h_2 * \langle \langle L, L \rangle, h_1 * \langle R^{n-1} * L, R \rangle \rangle \\ \text{Seq}_n^N(h_3) \& \& h_3 = R * h_3 * \langle \langle I, I \rangle^{n+1} * L, \langle R^{n^2-1} * L, R \rangle \rangle \\ &\& g = \text{Aux}_n(L^2) * \langle h_3, R^n \rangle * \langle h_2, R \rangle \\] &. \end{aligned}$$

For understanding it helps to identify the h_1, h_2, h_3 . Suppose $f = \langle f_0, \dots, f_{n-1}, f_n \rangle$. Then

$$\begin{aligned} h_1 &= [f_0, f_1, \dots, f_{n-1}]; \\ h_2 &= [f_0, f_1, \dots, f_{n-1}, \\ &\quad f_0 * R, f_1 * R, \dots, f_{n-1} * R, \\ &\quad \dots, \\ &\quad f_0 * R^{n-1}, f_1 * R^{n-1}, \dots, f_{n-1} * R^{n-1}]; \\ h_3 &= [I, R^{n+1}, R^{2(n+1)}, \dots, R^{(n-1)(n+1)}]. \end{aligned}$$

Now define

$$P(f, g, h) \stackrel{\Delta}{\Leftrightarrow} \exists n [\text{Seq}_n(f) \& \text{Seq}_n(g) \& \text{Cmp}_n(f * L) * \langle I, R^n \rangle * g = h].$$

Then P is Diophantine and for arbitrary n and $f, g \in \text{Seq}_n$ one has

$$h = f \# g \Leftrightarrow P(f, g, h). \blacksquare$$

11. For $f = [f_0, \dots, f_{n-1}]$ define $\Pi(f) \triangleq f_0 * \dots * f_{n-1}$. Then there exists a Diophantine relation P such that for all $n \in \mathbb{N}$ and all $f \in \text{Seq}_n$ one has

$$P(f, g) \Leftrightarrow \Pi(f) = g.$$

PROOF. Define $P(f, g) \triangleleft\triangleright$

$$\begin{aligned} & \exists n, h [\\ & \quad \text{Seq}_n(f) \& \\ & \quad \text{Seq}_{n+1}(h) \& h = ((f * \langle I, R \rangle) \# (R * h)) * \langle L, I * L, R \rangle \\ & \quad \& g = L * h * \langle I, R \rangle \\ &]. \end{aligned}$$

Then P works as can be seen realizing h has to be

$$[f_0 * \dots * f_{n-1}, f_1 * \dots * f_{n-1}, \dots, f_{n-2} * f_{n-1}, f_{n-1}, I]. \blacksquare$$

12. Define $\text{Byte}_n(f) \triangleleft\triangleright f = [b_0, \dots, b_{n-1}]$, for some $b_i \in \{L, R\}$. Then Byte_n is Diophantine uniformly in n .

PROOF. Using 2 one has $\text{Byte}_n(f)$ iff

$$\text{Seq}_n(f) \& f * \langle \langle I, I \rangle, R \rangle = \text{Cp}_n(I). \blacksquare$$

13. Let $m \in \mathbb{N}$ and let $[m]_2$ be its binary notation of length n . Let $[m]_{\text{Byte}} \in \text{Seq}_n^{\mathcal{N}}$ 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 $[6]_{\text{Byte}} = [R, L, L]$. Then there exists a Diophantine relation Bin such that for all $m \in \mathbb{N}$

$$\text{Bin}(s_m, f) \Leftrightarrow f = [m]_{\text{Byte}}.$$

PROOF. We need two auxiliary maps.

$$\begin{aligned} \text{Pow2}(n) &\triangleq [R^{2^{n-1}}, \dots, R^{2^0}]; \\ \text{Pow2}I(n) &\triangleq [\langle R^{2^{n-1}}, I \rangle, \dots, \langle R^{2^0}, I \rangle]. \end{aligned}$$

These relations $\text{Pow2}(n) = g$ and $\text{Pow2}I(n) = g$ are Diophantine uniformly in n . Indeed, $\text{Pow2}(n) = g$ iff

$$\text{Seq}_n(g) \& g = ((R * g) \# (R * g)) * [I, R];$$

and $\text{Pow2}I(n) = g$ iff

$$\begin{aligned} \text{Seq}_n(g) \& \text{Cp}_n(L) \# g = \text{Pow2}(n); \\ \& \text{Cp}_n(R) \# g = \text{Cp}_n(I). \end{aligned}$$

It follows that Bin is Diophantine since $\text{Bin}(m, f)$ iff

$$m \in \mathcal{N} \& \exists n [\text{Byte}_n(f) \& \Pi(f \# \text{Pow2}I(n)) = m]. \blacksquare$$

14. We now define a surjection $\varphi : \mathbb{N} \rightarrow \mathcal{F}$. Remember that \mathcal{F} is generated by two elements $\{e_0, e_1\}$ using only $*$. One has $e_1 = L$. Define

$$\varphi(n) \triangleq e_{i_0} * \dots * e_{i_{m-1}},$$

where $[n]_2 \triangleq i_{m-1} \dots i_0$. 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. Code(n, f) defined by $\varphi(n) = f$ is Diophantine uniformly in n .

PROOF. Indeed, Code(n, f) iff

$$\exists g [\text{Bin}(n, g) \& \Pi(g * \langle \langle e_0, e_1 \rangle, R \rangle) = f]. \blacksquare$$

16. Every $=$ -closed 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

$$g \in \mathcal{X} \Leftrightarrow \exists f \ f \in \#\mathcal{X} \ \& \ \text{Code}(f, g). \blacksquare$$

17. Every $=$ -closed 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 5F.12 and the fact that such relations are closed under intersection. ■

5B.57. THEOREM. A relation R on closed Λ_{SP} terms is Diophantine if and only if R is closed coordinate wise under $=$ and recursively enumerable.

PROOF. By 17 and corollaries 5B.50 and 5B.53. ■

5C. Gödel's system \mathcal{T} : higher-order primitive recursion

5C.1. DEFINITION. 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. The primitive recursive functions were thought to consist of all computable functions. This was shown to be false in [Sudan \[1927\]](#) and [Ackermann \[1928\]](#), who independently gave examples of computable functions that are not primitive recursive. Ten 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 $< n$. 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 maps that are commonly called functionals, to emphasize that they are more general than number theoretic functions.

Consider the full typestructure $\mathcal{M}_{\mathbb{N}}$ over the natural numbers, see Definition 2D.17. 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 above mentioned interest in higher-order computations leads to the following schema of higher-order primitive recursion proposed in Gödel [1958]¹⁴.

$$\begin{aligned} RMN0 &= M \\ RMN(n+1) &= N(RMn)n \end{aligned}$$

Here M need not be a natural number, but can have any $A \in \mathbb{T}^0$ as type (see Section 1A). The corresponding type of N is $A \rightarrow \mathbb{N} \rightarrow A$, where \mathbb{N} is the type of the natural numbers. We make some further observations with respect to this schema. First, the dependence of F on G and H in the first-order schema is made explicit by defining RMN , which is to be compared to F . Second, the parameters \bar{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* $R_A : A \rightarrow (A \rightarrow \mathbb{N} \rightarrow A) \rightarrow \mathbb{N} \rightarrow A$ for every type A .

5C.2. DEFINITION. 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 λ^0_{\rightarrow} -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 λ^0_{\rightarrow} -terms. Now consider $P \equiv R_0 K^*$. Then we have $P0 = 0$ and $P(n+1) = R_0 K^*(n+1) = K^*(R_0 K^* n) = n$ for all $n \in \mathbb{N}$, so that we call P the predecessor function. Now consider $x \dashv y \equiv Rx(P * K)y$. Here $P * K$ is the composition of P and K , that is, $(P * K)xy = P(K(x, y)) = P(x)$. We have $x \dashv 0 = x$ and $x \dashv (y+1) = Rx(P * K)(y+1) = (P * K)(Rx(P * K)y)y = P(Rx(P * K)y) = P(x \dashv y)$. Thus we have defined cut-off subtraction \dashv as primitive recursive functional.

In the previous paragraph, we have only used $R_{\mathbb{N}}$ 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 $R_{\mathbb{N} \rightarrow \mathbb{N}}$ as well. Given 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 $R_{\mathbb{N} \rightarrow \mathbb{N}}$ 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. The function A is a variant, due to Péter [1967] of a function by Ackermann. The essential difficulty of the function A is the nested recursion in the third clause below.

¹⁴For the purpose of the so-called Dialectica interpretation, a translation of intuitionistic arithmetic into the quantifier free theory of primitive recursive functionals of finite type, yielding a consistency proof for arithmetic.

5C.3. DEFINITION (Ackermann function).

$$\begin{aligned} A(0, m) &\triangleq m + 1 \\ A(n + 1, 0) &\triangleq A(n, 1) \\ A(n + 1, m + 1) &\triangleq A(n, A(n + 1, m)) \end{aligned}$$

Write $A(n) \triangleq \lambda m. A(n, m)$. Then $A(0)$ is the successor function and $A(n + 1, m) = A(n)^{m+1}(1)$, by the last two equations. Therefore we can define $A = RSH$, 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 $A(2, m) = H(A(1), 1, m) = A(1)^{m+1}(1) = 2m + 3$.

Syntax of λ_T

In this section we formalize Gödel's \mathcal{T} as an extension of the simply typed lambda calculus $\lambda_{\rightarrow}^{\text{Ch}}$ over \mathbb{T}^0 , called λ_T . In this and the next two sections we write the type atom 0 as ' \mathbf{N} ', as it is intended as type of the natural numbers.

5C.4. DEFINITION. The theory *Gödel's \mathcal{T}* , notation λ_T , is defined as follows.

- (i) The set of *types* of λ_T is defined by $\mathbb{T}(\lambda_T) = \mathbb{T}^{\{\mathbf{N}\}}$, where the atomic type \mathbf{N} is called the *natural number type*.
- (ii) The *terms* of λ_T are obtained by adding to the term formation rules of λ_{\rightarrow}^0 the constants $0 : \mathbf{N}$, $S^+ : \mathbf{N} \rightarrow \mathbf{N}$ and $R_A : A \rightarrow (A \rightarrow \mathbf{N} \rightarrow A) \rightarrow \mathbf{N} \rightarrow A$ for all types A .
- (iii) We denote the set of (closed) terms of type A by $\Lambda_T(A)$ (respectively $\Lambda_T^{\emptyset}(A)$) and put $\Lambda_T = \bigcup_A \Lambda_T(A)$ ($\Lambda_T^{\emptyset} = \bigcup_A \Lambda_T^{\emptyset}(A)$).
- (iv) Terms constructed from 0 and S^+ only are called *numerals*, with 1 abbreviating $S^+(0)$, 2 abbreviating $S^+(S^+(0))$, and so on. An arbitrary numeral will be denoted by n .
- (v) We define inductively $n^{A \rightarrow B} \equiv \lambda x^A. n^B$, with $n^{\mathbf{N}} \equiv n$.
- (vi) The *formulas* of λ_T are equations between terms (of the same type).
- (vii) 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.
- (viii) The *notion of reduction T* on λ_T , notation \rightarrow_T , is defined by the following contraction rules (extending β -reduction):

$$\begin{aligned} (\lambda x.M)N &\rightarrow_T M[x := N] \\ R_AMN0 &\rightarrow_T M \\ R_AMN(S^+P) &\rightarrow_T N(R_AMNP)P \end{aligned}$$

This gives rise to reduction relations \rightarrow_T , $\rightarrow\!\!\rightarrow_T$. Gödel did not consider η -reduction.

5C.5. THEOREM. *The conversion relation $=_T$ coincides with equality provable in λ_T .*

PROOF. By an easy extension of the proof of this result in untyped lambda calculus, see B[1984] Proposition 3.2.1. ■

5C.6. LEMMA. *Every closed normal form of type \mathbf{N} is a numeral.*

PROOF. Consider the leftmost symbol of a closed normal form of type \mathbf{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 \mathbf{N} and a redex is not a normal form. If the leftmost symbol is 0, then the term is the numeral 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 $RMNP\vec{Q}$, 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 $RMNP$ with P a numeral can be excluded, as $RMNP$ should be a normal form. ■

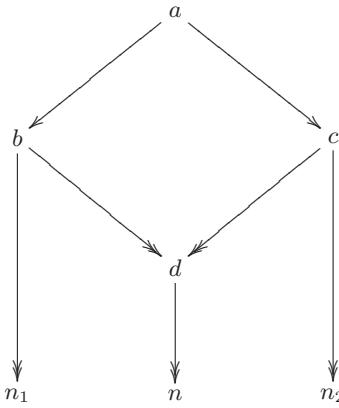
We now prove **SN** and **CR** for λ_T , two results that could be proved independently from each other. However, the proof of **CR** can be simplified by using **SN**, which we prove first by an extension of the proof of **SN** for λ_{\rightarrow}^0 , Theorem 2B.1.

5C.7. THEOREM. *Every $M \in \Lambda_T$ is **SN** with respect to \rightarrow_T .*

PROOF. Recall the notion of computability from the proof of Theorem 2B.1. We generalize it to terms of λ_T . We shall frequently use that computable terms are **SN**, see formula (2) in the proof of Theorem 2B.1. In view of the definition of computability it suffices to prove that the constants $0, S^+, R_A$ of λ_T are computable. The constant $0 : N$ is computable since it is **SN**. Consider S^+P with computable $P : N$, 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 R_AMNP is of type A . Since $P : N$ is computable, it is **SN**. Since \rightarrow_T is finitely branching, P has only finitely many normal forms, which are numerals by Lemma 5C.6. Let $\#P$ be the largest of those numerals. We shall prove by induction on $\#P$ that R_AMNP is computable. Let \vec{Q} be computable such that $R_AMNP\vec{Q}$ is of type N . We have to show that $R_AMNP\vec{Q}$ is **SN**. If $\#P = 0$, then every reduct of $R_AMNP\vec{Q}$ passes through a reduct of $M\vec{Q}$, and **SN** follows since $M\vec{Q}$ is computable. If $\#P = S^+n$, then every reduct of $R_AMNP\vec{Q}$ passes through a reduct of $N(R_AMNP')P'\vec{Q}$, where P' is such that S^+P' is a reduct of P . Then we have $\#P' = n$ and by induction it follows that R_AMNP' is computable. Now **SN** follows since all terms involved are computable. We have proved that R_AMNP is computable whenever M, N, P are, and hence R_A is computable. ■

5C.8. LEMMA (Newman's Lemma, localized). *Let S be a set and \rightarrow a binary relation on S that is WCR. For every $a \in S$ we have: if $a \in \mathbf{SN}$, then $a \in \mathbf{CR}$.*

PROOF. Call an element *ambiguous* if it reduces to two (or more) distinct normal forms. Assume $a \in \mathbf{SN}$, then a reduces to at least one normal form and all reducts of a are **SN**. It suffices for $a \in \mathbf{CR}$ 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 $a \rightarrow b \rightarrow \dots \rightarrow n_1$ and $a \rightarrow c \rightarrow \dots \rightarrow n_2$. Applying WCR to the diverging reduction steps yields a common reduct d such that $b \rightarrow d$ and $c \rightarrow d$. Since $d \in \mathbf{SN}$ 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 11.

FIGURE 11. Ambiguous a has ambiguous reduct b or c .

Hence a has a one-step reduct which is again ambiguous and SN. Iterating this argument yields an infinite reduction sequence contradicting $a \in \text{SN}$, so a cannot be ambiguous. ■

5C.9. THEOREM. *Every $M \in \Lambda_T$ is WCR with respect to \rightarrow_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. ■

5C.10. THEOREM. *Every $M \in \Lambda_T$ is CR with respect to \rightarrow_T .*

PROOF. By Newman's Lemma 5C.8, using Theorem 5C.7. ■

If one considers λ_T also with η -reduction, then the above results can also be obtained. For SN it simply suffices to strengthen the notion of computability for the base case to SN with also η -reductions included. WCR and hence CR are harder to obtain and require techniques like η -postponement, see B[1984], Section 15.1.6.

Semantics of λ_T

In this section we give a general model definition of λ_T building on that of λ_\rightarrow^0 .

5C.11. DEFINITION. A *model of λ_T* is a typed λ -model with interpretations of the constants 0 , S^+ and R_A for all A , such that the schema of higher-order primitive recursion is valid.

5C.12. EXAMPLE. Recall the full typestructure over the natural numbers, that is, sets $\mathcal{M}_N = \mathbb{N}$ and $\mathcal{M}_{A \rightarrow B} = \mathcal{M}_A \rightarrow \mathcal{M}_B$, with set-theoretic application. The full typestructure becomes the canonical model of λ_T by interpreting 0 as 0 , S^+ as the successor function, and the constants R_A as primitive recursors of the right type. The proof that $\llbracket R_A \rrbracket$ is well-defined goes by induction.

Other interpretations of Gödel's T can be found in Exercises 5F.28-5F.31.

Computational strength

As primitive recursion over higher types turns out to be equivalent with transfinite ordinal recursion, we give a brief review of the theory of ordinals.

The following are some ordinal numbers, simply called *ordinals*, in increasing order.

$$0, 1, 2, \dots, \omega, \omega + 1, \omega + 2, \dots, \omega + \omega = \omega \cdot 2, \dots, \omega \cdot \omega = \omega^2, \dots, \omega^\omega, \dots, \omega^{(\omega^\omega)}, \dots$$

Apart from ordinals, also some basic operations of ordinal arithmetic are visible, namely addition, multiplication and exponentiation, denoted in the same way as in high-school algebra. The dots \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 $\forall \gamma < \alpha. P(\gamma)$ for all α .

In fact the arithmetical operations are defined by means of two more primitive operations on ordinals, namely the *successor* operation $+1$ 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 n produced by iterating the successor operation on 0.

These primitive operations divide the ordinals in three classes: the *successor* ordinals of the form $\alpha + 1$, 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 0. (In fact 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 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: $1 + \omega = \omega \neq \omega + 1$ and $2 \cdot \omega = \omega \neq \omega \cdot 2$. Furthermore, multiplication is left distributive over addition, but not right distributive: $(1 + 1) \cdot \omega = \omega \neq 1 \cdot \omega + 1 \cdot \omega$. The sum $\alpha + \beta$ is weakly increasing in α and strictly increasing in β . Similarly for the product $\alpha \cdot \beta$ with $\alpha > 0$. The only exponentiations we shall use, 2^α and ω^α , are strictly increasing in α .

Addition	Multiplication	Exponentiation ($\alpha > 0$)
$\alpha + 0 \triangleq \alpha$	$\alpha \cdot 0 \triangleq 0$	$\alpha^0 \triangleq 1$
$\alpha + (\beta + 1) \triangleq (\alpha + \beta) + 1$	$\alpha \cdot (\beta + 1) \triangleq \alpha \cdot \beta + \alpha$	$\alpha^{\beta+1} \triangleq \alpha^\beta \cdot \alpha$
$\alpha + \lambda \triangleq \bigcup \{\alpha + \beta \mid \beta < \lambda\}$	$\alpha \cdot \lambda \triangleq \bigcup \{\alpha \cdot \beta \mid \beta < \lambda\}$	$\alpha^\lambda \triangleq \bigcup \{\alpha^\beta \mid \beta < \lambda\}$

TABLE 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.

5C.13. DEFINITION. Let f be an ordinal function. Define by induction $f^0(\alpha) \triangleq \alpha$, $f^{\beta+1}(\alpha) \triangleq f(f^\beta(\alpha))$ and $f^\lambda(\alpha) \triangleq \bigcup\{f^\beta(\alpha) \mid \beta < \lambda\}$ for every limit ordinal λ . We call f^β the β -th *transfinite iteration* of f .

5C.14. EXAMPLE. As examples we redefine the arithmetical operations above.

$$\begin{aligned}\alpha + \beta &= f^\beta(\alpha) \\ \alpha \cdot \beta &= g_\alpha^\beta(0) \\ \alpha^\beta &= h_\alpha^\beta(1),\end{aligned}$$

with f the successor function, $g_\alpha(\gamma) = \gamma + \alpha$, and $h_\alpha(\gamma) = \gamma \cdot \alpha$. Do Exercise 5F.33.

We proceed with the canonical construction for finding the least fixed point of a weakly increasing ordinal function if there exists one. The proof is in Exercise 5F.19.

5C.15. LEMMA. Let f be a weakly increasing ordinal function. Then:

- (i) $f^{\alpha+1}(0) \geq f^\alpha(0)$ for all α ;
- (ii) $f^\alpha(0)$ is weakly increasing in α ;
- (iii) $f^\alpha(0)$ does not surpass any fixed point of f ;
- (iv) $f^\alpha(0)$ is strictly increasing (and hence $f^\alpha(0) \geq \alpha$), until a fixed point of f is reached, after which $f^\alpha(0)$ becomes constant.

If a weakly increasing ordinal function f has a fixed point, then it has a smallest fixed point and Lemma 5C.15 above guarantees that this so-called *least fixed point* is of the form $f^\alpha(0)$, that is, can be obtained by transfinite iteration of f starting at 0. This justifies the following definition.

5C.16. DEFINITION. Let f be a weakly increasing ordinal function having a least fixed point which we denote by $\text{lfp}(f)$. The *closure ordinal* of f is the smallest ordinal α such that $f^\alpha(0) = \text{lfp}(f)$.

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 .

5C.17. 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 : $0, f(0), f(f(0))$ 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 $f(\lambda) = \bigcup\{f(\alpha) \mid \alpha < \lambda\} = f^\omega(0) = \lambda$, so the closure ordinal is ω . ■

For example, $f(\alpha) = 1 + \alpha$ has $\text{lfp}(f) = \omega$, and $f(\alpha) = (\omega + 1) \cdot \alpha$ has $\text{lfp}(f) = 0$. In contrast, $f(\alpha) = \alpha + 1$ has no fixed point (note that the latter f is weakly increasing, but the condition on limit ordinals is not satisfied). Finally, $f(\alpha) = 2^\alpha$ has $\text{lfp}(f) = \omega$, and the least fixed point of $f(\alpha) = \omega^\alpha$ is denoted by ϵ_0 , being the supremum of the sequence:

$$0, \omega^0 = 1, \omega^1 = \omega, \omega^\omega, \omega^{\omega^\omega}, \omega^{\omega^{\omega^\omega}}, \dots$$

In the following proposition we formulate some facts about ordinals that we need in the sequel.

5C.18. 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 $\alpha_1, \alpha_2, \dots, \alpha_n$ 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 $\alpha < \beta$.

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 $\omega^\alpha \cdot n$ 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 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 12, yields an encoding of pairs $\langle x, y \rangle$ of natural numbers x, y as given in Definition 5C.19.

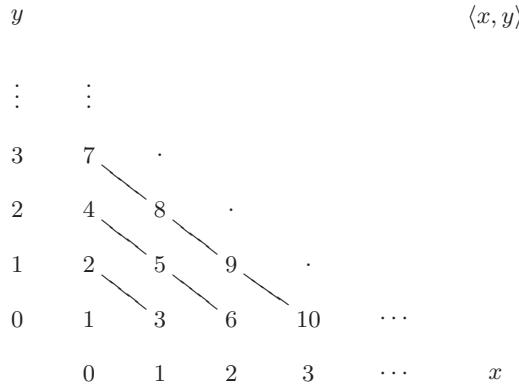


FIGURE 12. $\langle x, y \rangle$ -values for $x + y \leq 3$

Finite sequences $[x_1, \dots, x_k]$ 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.

5C.19. DEFINITION. Recall that $1+2+\cdots+n = \frac{1}{2}n(n+1)$ gives the number of grid points satisfying $x + y < n$. The function \dashv below is to be understood as cut-off subtraction,

that is, $x - y = 0$ whenever $y \geq x$. Define the following functions.

$$\begin{aligned}\langle x, y \rangle &\triangleq \frac{1}{2}(x+y)(x+y+1) + x + 1 \\ \text{sum}(p) &\triangleq \min\{n \mid p \leq \frac{1}{2}n(n+1)\} - 1 \\ x(p) &\triangleq p - \langle 0, \text{sum}(p) \rangle \\ y(p) &\triangleq \text{sum}(p) - x(p)\end{aligned}$$

Now let $[] \triangleq 0$ and, for $k > 0$, $[x_1, \dots, x_k] \triangleq \langle x_1, [x_2, \dots, x_k] \rangle$ encode lists. Define $\text{lth}(0) \triangleq 0$ and $\text{lth}(p) \triangleq 1 + \text{lth}(y(p))$ ($p > 0$) to compute the length of a list.

The following lemma is a straightforward consequence of the above definition.

5C.20. LEMMA. *For all $p > 0$ we have $p = \langle x(p), y(p) \rangle$. Moreover, $\langle x, y \rangle > x$, $\langle x, y \rangle > y$, $\text{lth}([x_1, \dots, x_k]) = k$ 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 $\bar{\alpha}$ for the natural number representing the ordinal α .

5C.21. DEFINITION. Let $\alpha < \epsilon_0$ have Cantor normal form $\omega^{\alpha_1} + \dots + \omega^{\alpha_k}$. We encode α by putting $\bar{\alpha} = [\bar{\alpha}_1, \bar{\alpha}_2, \dots, \bar{\alpha}_n]$. This representation is well-defined since every α_i ($1 \leq i \leq n$) is strictly smaller than α . The zero ordinal 0, having the empty sum as Cantor normal form, is thus represented by the empty list $[]$, so by the natural number 0.

Examples are $\bar{0} = []$, $\bar{1} = [[], []]$, $\bar{2} = [[[], []], \dots]$ and $\bar{\omega} = [[[[], []], []], \dots]$ and so on. Observe that $[[], [[]]]$ does not represent an ordinal as $\omega^0 + \omega^1$ is not a Cantor normal form. The following lemmas allow one to identify which natural numbers represent ordinals and to compare them.

5C.22. LEMMA. *Let \prec be the lexicographic ordering on lists. Then \prec is primitive recursive and $\bar{\alpha} \prec \bar{\beta} \Leftrightarrow \alpha < \beta$ for all $\alpha, \beta < \epsilon_0$.*

PROOF. Define $\langle x, y \rangle \prec \langle x', y' \rangle \Leftrightarrow (x \prec x') \vee (x = x' \wedge y \prec y')$ 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 $\dots \prec [0, 0, 1] \prec [0, 1], \prec [1]$ has no smallest element.) ■

5C.23. LEMMA. *For $x \in \mathbb{N}$, define the following notions.*

$$\begin{aligned}\text{Ord}(x) &\triangleq x = \bar{\alpha} \text{ for some ordinal } \alpha < \epsilon_0; \\ \text{Succ}(x) &\triangleq x = \bar{\alpha} \text{ for some successor ordinal } < \epsilon_0; \\ \text{Lim}(x) &\triangleq x = \bar{\alpha} \text{ for some limit ordinal } < \epsilon_0; \\ \text{Fin}(x) &\triangleq x = \bar{\alpha} \text{ for some ordinal } \alpha < \omega.\end{aligned}$$

Then Ord, Fin, Succ and Lim are primitive recursive predicates.

PROOF. By course of value recursion.

- (i) Put $\text{Ord}(0)$ and $\text{Ord}(\langle x, y \rangle) \Leftrightarrow (\text{Ord}(x) \wedge \text{Ord}(y) \wedge (y > 0 \Rightarrow x(y) \preceq x))$.
- (ii) Put $\neg\text{Succ}(0)$ and $\text{Succ}(\langle x, y \rangle) \Leftrightarrow (\text{Ord}(\langle x, y \rangle) \wedge (x > 0 \Rightarrow \text{Succ}(y)))$.
- (iii) Put $\text{Lim}(x) \Leftrightarrow (\text{Ord}(x) \wedge \neg\text{Succ}(s) \wedge x \neq [])$.
- (iv) Put $\text{Fin}(x) \Leftrightarrow (x = []) \vee (x = \langle 0, y \rangle \wedge \text{Fin}(y))$. ■

5C.24. LEMMA. *There exist primitive recursive functions \exp (base ω exponentiation), succ (successor), pred (predecessor), plus (addition), $\exp 2$ (base 2 exponentiation) such that for all α, β : $\exp(\bar{\alpha}) = \overline{\omega^\alpha}$, $\text{succ}(\bar{\alpha}) = \overline{\alpha + 1}$, $\text{pred}(\bar{0}) = \bar{0}$, $\text{pred}(\overline{\alpha + 1}) = \bar{\alpha}$, $\text{plus}(\bar{\alpha}, \bar{\beta}) = \overline{\alpha + \beta}$, $\exp 2(\bar{\alpha}) = \overline{2^\alpha}$.*

PROOF. Put $\exp(x) = [x]$. Put $\text{succ}(0) = \langle 0, 0 \rangle$ and $\text{succ}(\langle x, y \rangle) = \langle x, \text{succ}(y) \rangle$, then $\text{succ}([x_1, \dots, x_k]) = [x_1, \dots, x_k, 0]$. Put $\text{pred}(0) = 0$, $\text{pred}(\langle x, 0 \rangle) = x$ and $\text{pred}(\langle x, y \rangle) = \langle x, \text{pred}(y) \rangle$ for $y > 0$. For plus, use the absorption property in adding the Cantor normal forms of α and β . For $\exp 2$ we use $\omega^\beta = 2^{\omega \cdot \beta}$. Let α have Cantor normal form $\omega^{\alpha_1} + \dots + \omega^{\alpha_k}$. Then $\omega \cdot \alpha = \omega^{1+\alpha_1} + \dots + \omega^{1+\alpha_k}$. By absorption, $1 + \alpha_i = \alpha_i$ whenever $\alpha_i \geq \omega$. It follows that we have

$$\alpha = \omega \cdot (\omega^{\alpha_1} + \dots + \omega^{\alpha_i} + \omega^{n_1} + \dots + \omega^{n_p}) + n,$$

for suitable n_j, n with $\alpha_1 \geq \dots \geq \alpha_i \geq \omega$, $n_j + 1 = \alpha_{i+j} < \omega$ for $1 \leq j \leq p$ and $n = k - i - p$ with $\alpha_{k'} = 0$ for all $i + p < k' \leq k$. Using $\omega^\beta = 2^{\omega \cdot \beta}$ we can calculate $2^\alpha = \omega^\beta \cdot 2^n$ with $\beta = \omega^{\alpha_1} + \dots + \omega^{\alpha_i} + \omega^{n_1} + \dots + \omega^{n_p}$ and n as above. If $\bar{\alpha} = [x_1, \dots, x_i, \dots, x_j, \dots, 0, \dots, 0]$, then $\bar{\beta} = [x_1, \dots, x_i, \dots, \text{pred}(x_j), \dots]$ and we can obtain $\exp 2(\bar{\alpha}) = \overline{2^\alpha} = \overline{\omega^\beta \cdot 2^n}$ by doubling n times $\overline{\omega^\beta} = \exp(\bar{\beta})$ using plus. ■

5C.25. LEMMA. *There exist primitive recursive functions num , mun such that $\text{num}(n) = \bar{n}$ and $\text{mun}(\bar{n}) = n$ for all n . In particular we have $\text{mun}(\text{num}(n)) = n$ and $\text{num}(\text{mun}(\bar{n})) = \bar{n}$ for all n . In other words, num is the order isomorphism between $(\mathbb{N}, <)$ and $(\{\bar{n} \mid n \in \mathbb{N}\}, \prec)$ and mun is the inverse order isomorphism.*

PROOF. Put $\text{num}(0) = 0 = []$ and $\text{num}(n + 1) = \text{succ}(\text{num}(n))$ and $\text{mun}(0) = 0$ and $\text{mun}(\langle x, y \rangle) = \text{mun}(y) + 1$. ■

5C.26. LEMMA. *There exists a primitive recursive function p such that $p(\bar{\alpha}, \bar{\beta}, \bar{\gamma}) = \bar{\alpha}'$ with $\alpha' < \alpha$ and $\beta < \gamma + 2^{\alpha'}$, provided that α is a limit and $\beta < \gamma + 2^\alpha$.*

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 $\bar{\alpha}'$ can be computed from $\bar{\alpha}, \bar{\beta}, \bar{\gamma}$ in a primitive recursive way, as will become clear by the following argument. If $\beta \leq \gamma$, then we can simply take $\alpha' = 0$. Otherwise, let $\beta = \omega^{\beta_1} + \dots + \omega^{\beta_n}$ and $\gamma = \omega^{\gamma_1} + \dots + \omega^{\gamma_m}$ 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 $\alpha = \omega \cdot \xi$ for suitable ξ , and hence $2^\alpha = \omega^\xi$. Since $\beta < \gamma + 2^\alpha$ it follows by absorption that $\omega^{\beta_i} + \dots + \omega^{\beta_n} < \omega^\xi$. Hence $\beta_i + 1 \leq \xi$, so $\omega^{\beta_i} + \dots + \omega^{\beta_n} \leq \omega^{\beta_i} \cdot n < \omega^{\beta_i} \cdot 2^n = 2^{\omega \cdot \beta_i + n}$. Now take $\alpha' = \omega \cdot \beta_i + n < \omega \cdot (\beta_i + 1) \leq \omega \cdot \xi = \alpha$ and observe $\beta < \gamma + 2^{\alpha'}$. ■

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 $\text{Fin}(\bar{\alpha})$, $\alpha \prec \beta$ instead of $\bar{\alpha} \prec \bar{\beta}$, 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, $\forall x (\text{Ord}(x) \Rightarrow P(x))$, and $\exists x (\text{Ord}(x) \wedge P(x))$, 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.

5C.27. DEFINITION. Define

$$\text{Ind}(P) \stackrel{\Delta}{\iff} \forall \alpha ((\forall \beta < \alpha P(\beta)) \Rightarrow P(\alpha)).$$

Then the principle of *transfinite induction* up to α , notation **TI** $_{\alpha}$, states

$$\text{Ind}(P) \Rightarrow \forall \beta < \alpha P(\beta).$$

Here $\text{Ind}(P)$ expresses that P is *inductive*, that is, $\forall \beta < \alpha P(\beta)$ implies $P(\alpha)$ 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 $P(0)$ and $P(\alpha) \Rightarrow P(\alpha + 1)$ 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 Section.)

By Lemma 5C.25, 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** $_{\alpha}$ 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} .

5C.28. 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 5C.25. ■

5C.29. LEMMA. *The schema $\text{TI}_{\omega \cdot 2}$ is provable in Peano Arithmetic with the schema TI_{ω} .*

PROOF. Assume TI_{ω} and $\text{Ind}(P)$ for some P . In order to prove $\forall \alpha < \omega \cdot 2 P(\alpha)$ define $P'(\alpha) \equiv \forall \beta < \omega + \alpha P(\beta)$. By TI_{ω} we have $P'(0)$. Also $P'(\alpha) \Rightarrow P'(\alpha + 1)$, as $P'(\alpha)$ implies $P(\omega + \alpha)$ by $\text{Ind}(P)$. If $\text{Lim}(\alpha)$, 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 $\text{TI}_{\omega \cdot 2}$. ■

5C.30. LEMMA. *The schema $\text{TI}_{2^{\alpha}}$ is provable in Peano Arithmetic with the schema TI_{α} , for all $\alpha < \epsilon_0$.*

PROOF. Assume TI_{α} and $\text{Ind}(P)$ for some P . In order to prove $\forall \alpha' < 2^{\alpha} P(\alpha')$ define $P'(\alpha') \equiv \forall \beta (\forall \beta' < \beta P(\beta') \Rightarrow \forall \beta' < \beta + 2^{\alpha'} P(\beta'))$. The intuition behind $P'(\alpha')$ is: if P holds on an arbitrary initial segment, then we can prolong this segment with $2^{\alpha'}$. The goal will be to prove $P'(\alpha)$, since we can then prolong the empty initial segment on which P vacuously holds to one of length 2^{α} . 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 $P'(0)$ as P is inductive and $2^0 = 1$. The argument for

$P'(\alpha) \Rightarrow P'(\alpha+1)$ amounts to applying $P'(\alpha)$ twice, relying on $2^{\alpha+1} = 2^\alpha + 2^\alpha$. Assume $P'(\alpha)$ and $\forall \beta' < \beta P(\beta')$ for some β . By $P'(\alpha)$ we have $\forall \beta' < \beta + 2^\alpha P(\beta')$. Hence again by $P'(\alpha)$, but now with $\beta + 2^\alpha$ instead of β , we have $\forall \beta' < \beta + 2^\alpha + 2^\alpha P(\beta')$. We conclude $P'(\alpha+1)$. 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.

5C.31. LEMMA (Gentzen). *TI $_\alpha$ is provable in Peano Arithmetic for every ordinal $\alpha < \epsilon_0$.*

PROOF. Use $\omega^\beta = 2^{\omega \cdot \beta}$, so $2^{\omega \cdot 2} = \omega^2$ and $2^{\omega^2} = \omega^\omega$. From ω^ω on, iterating exponentiation with base 2 yields the same ordinals as with base ω . We start with Lemma 5C.28 to obtain TI_ω , continue with Lemma 5C.29 to obtain $\text{TI}_{\omega \cdot 2}$, and surpass TI_α for every ordinal $\alpha < \epsilon_0$ by iterating Lemma 5C.30 a sufficient number of times. ■

We now translate the Gentzen argument from transfinite induction to transfinite recursion, closely following the development of Terlouw [1982].

5C.32. DEFINITION. Given a functional F of type $0 \rightarrow A$ and ordinals α, β , define primitive recursively

$$[F]_\beta^\alpha(\beta') \triangleq \begin{cases} F(\beta') & \text{if } \beta' \prec \beta \preceq \alpha, \\ 0^A & \text{otherwise.} \end{cases}$$

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 $[F]_\alpha \triangleq [F]_\alpha^\alpha$, that is, the functional F restricted to an initial segment of ordinals smaller than α .

5C.33. 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:

$$F(\beta) \triangleq G([F]_\beta^\alpha, \beta).$$

Note that, by the above definition, $F(\beta) = G(0^{0 \rightarrow A}, \beta)$ 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 5C.29, with the primitive recursive functionals taking over the role of Peano Arithmetic.

5C.34. LEMMA. *Every functional definable by the schema TR_ω is \mathcal{T} -definable.*

PROOF. Let $F_0(\alpha) = G([F_0]_\alpha^\omega, \alpha)$ be defined by TR_ω . We have to show that F_0 is \mathcal{T} -definable. Define primitive recursively F_1 by $F_1(0) \triangleq 0^{0 \rightarrow A}$ and

$$F_1(n+1, \alpha) \triangleq \begin{cases} F_1(n, \alpha) & \text{if } \alpha < n \\ G([F_1(n)]_\alpha^\omega, \alpha) & \text{otherwise} \end{cases}$$

By induction one shows $[F_0]_n^\omega = [F_1(n)]_n^\omega$ for all n . Define primitive recursively F_2 by $F_2(n) \triangleq F_1(n+1, n)$ and $F_2(\alpha) \triangleq 0^A$ 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

$$F_0(\alpha) \triangleq \begin{cases} F_2(\alpha) & \text{if } \alpha < \omega \\ G(F_2, \omega) & \text{if } \alpha = \omega \\ G(0^{0 \rightarrow A}, \alpha) & \text{otherwise} \end{cases}$$

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.

5C.35. LEMMA. *Every functional definable by the schema $TR_{\omega \cdot 2}$ is definable by the schema TR_ω .*

PROOF. Put $\omega \cdot 2 = \alpha$ and let $F_0(\beta) \triangleq G([F_0]_\beta^\alpha, \beta)$ 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 $F_1(\beta) \triangleq G([F_1]_\beta^\omega, \beta)$ 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 $P'(0)$ in the proof of Lemma 5C.29. Now define H of type $0 \rightarrow (0 \rightarrow A) \rightarrow (0 \rightarrow A)$ by TR_ω as follows. The more complicated type of H as compared to the type $0 \rightarrow A$ of F is the counterpart of the more complicated formula P' as compared to P in the proof of Lemma 5C.29.

$$\begin{aligned} H(0, F) &\triangleq [F_1]_\omega \\ H(\beta + 1, F, \beta') &\triangleq \begin{cases} H(\beta, F, \beta') & \text{if } \beta' < \omega + \beta \\ G(H(\beta, F), \beta') & \text{if } \beta' = \omega + \beta \\ 0^A & \text{otherwise} \end{cases} \end{aligned}$$

This definition can easily be cast in the form $H(\beta) \triangleq G'([H]_\beta^\omega, \beta)$ for suitable G' , so that H is actually defined by TR_ω . We can prove $H(\beta, 0^{0 \rightarrow A}) = [F_0]_{\omega+\beta}^\alpha$ for all $\beta < \omega$ by TI_ω . Finally we define

$$F_2(\beta') \triangleq \begin{cases} F_1(\beta') & \text{if } \beta' < \omega \\ G(H(\beta, 0^{0 \rightarrow A}), \beta') & \text{if } \beta' = \omega + \beta < \alpha \\ G(0^{0 \rightarrow A}, \beta') & \text{otherwise} \end{cases}$$

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. ■

5C.36. LEMMA. *Every functional definable by the schema TR_{2^α} is definable by the schema TR_α , for all $\alpha < \epsilon_0$.*

PROOF. Let $F_0(\beta) \triangleq G([F_0]_\beta^{2^\alpha}, \beta)$ be defined by TR_{2^α} . 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 5C.30, namely that we can prolong initial segments with 2^α .

$$propH(\alpha') \iff \forall \beta, F ([F]_\beta^{2^\alpha} = [F_0]_\beta^{2^\alpha} \Rightarrow [H(\alpha', \beta, F)]_{\beta+2^{\alpha'}}^{2^\alpha} = [F_0]_{\beta+2^{\alpha'}}^{2^\alpha})$$

To make $\text{prop}H$ come true, define H of type $0 \rightarrow 0 \rightarrow (0 \rightarrow A) \rightarrow (0 \rightarrow A)$ as follows.

$$H(0, \beta, F, \beta') \triangleq \begin{cases} F(\beta') & \text{if } \beta' < \beta \leq 2^\alpha \\ G([F]_\beta^{2^\alpha}, \beta) & \text{if } \beta' = \beta \leq 2^\alpha \\ 0^A & \text{otherwise} \end{cases}$$

$$H(\alpha' + 1, \beta, F) \triangleq H(\alpha', \beta + 2^{\alpha'}, H(\alpha', \beta, F))$$

If α' is a limit ordinal, then we use the function p from Lemma 5C.26.

$$H(\alpha', \beta, F, \beta') \triangleq \begin{cases} H(p(\alpha', \beta', \beta), \beta, F, \beta') & \text{if } \beta' < \beta + 2^{\alpha'} \\ 0^A & \text{otherwise} \end{cases}$$

This definition can easily be cast in the form $H(\beta) \triangleq G'([H]_\beta^\alpha, \beta)$ for suitable G' , so that H is in fact defined by TR_α . We shall prove that $\text{prop}H(\alpha')$ is inductive, and conclude $\text{prop}H(\alpha')$ for all $\alpha' \leq \alpha$ by TI_α . This implies $[H(\alpha', 0, 0^{0 \rightarrow A})]_{2^{\alpha'}}^{2^\alpha} = [F_0]_{2^{\alpha'}}^{2^\alpha}$ for all $\alpha' \leq \alpha$, so that one could manufacture F_0 from H in the following way:

$$F_0(\beta) \triangleq \begin{cases} H(\alpha, 0, 0^{0 \rightarrow A}, \beta) & \text{if } \beta < 2^\alpha \\ G(H(\alpha, 0, 0^{0 \rightarrow A}), \beta) & \text{if } \beta = 2^\alpha \\ G(0^{0 \rightarrow A}, \beta) & \text{otherwise} \end{cases}$$

It remains to show that $\text{prop}H(\alpha')$ is inductive up to and including α . For the case $\alpha' = 0$ we observe that $H(0, \beta, F)$ follows F up to β , applies G to the initial segment of $[F]_\beta^{2^\alpha}$ in β , and zeroes after β . This entails $\text{prop}H(0)$, as $2^0 = 1$. Analogous to the successor case in the proof of Lemma 5C.30, we prove $\text{prop}H(\alpha + 1)$ by applying $\text{prop}H(\alpha)$ twice, once with β and once with $\beta + 2^\alpha$. Given β and F we infer:

$$\begin{aligned} [F]_\beta^{2^\alpha} = [F_0]_\beta^{2^\alpha} &\Rightarrow [H(\alpha', \beta, F)]_{\beta+2^{\alpha'}}^{2^\alpha} = [F_0]_{\beta+2^{\alpha'}}^{2^\alpha} \Rightarrow \\ &[H(\alpha', \beta + 2^{\alpha'}, H(\alpha', \beta, F))]_{\beta+2^{\alpha'+1}}^{2^\alpha} = [F_0]_{\beta+2^{\alpha'+1}}^{2^\alpha} \end{aligned}$$

For the limit case, assume $\alpha' \leq \alpha$ is a limit ordinal such that $\text{prop}H$ holds for all smaller ordinals. Recall that, according to Lemma 5C.26 and putting $\alpha'' = p(\alpha', \beta', \beta)$, $\alpha'' < \alpha'$ and $\beta' < \beta + 2^{\alpha''}$ whenever $\beta' < \beta + 2^{\alpha'}$. Now assume $[F]_\beta^{2^\alpha} = [F_0]_\beta^{2^\alpha}$ and $\beta' < \beta + 2^{\alpha'}$, then $[H(\alpha'', \beta, F)]_{\beta+2^{\alpha''}}^{2^\alpha} = [F_0]_{\beta+2^{\alpha''}}^{2^\alpha}$ by $\text{prop}H(\alpha'')$, so $H(\alpha'', \beta, F, \beta') = F_0(\beta')$. It follows that $[H(\alpha', \beta, F)]_{\beta+2^{\alpha'}}^{2^\alpha} = [F_0]_{\beta+2^{\alpha'}}^{2^\alpha}$. ■

5C.37. 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 5C.31. ■

Lemma 5C.37 shows that ϵ_0 is a lower bound for the computational strength of Gödel's system \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.

5D. Spector's system \mathcal{B} : bar recursion

Spector [1962] extends Gödel's \mathcal{T} with a definition schema called bar recursion.¹⁵ 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 conveniently be encoded by pairs consisting of a functional of type \mathbb{N} and one of type $\mathbb{N} \rightarrow A$. The intuition is that the pair $\langle x, C \rangle$ encodes the sequence of the first x values of C , that is, $C(0), \dots, C(x-1)$. We need auxiliary functionals to extend finite sequences of any type. A convenient choice is the primitive recursive functional $\text{Ext}_A : (\mathbb{N} \rightarrow A) \rightarrow \mathbb{N} \rightarrow A \rightarrow \mathbb{N} \rightarrow A$ defined by:

$$\text{Ext}_A(C, x, a, y) \triangleq \begin{cases} C(y) & \text{if } y < x, \\ a & \text{otherwise.} \end{cases}$$

We shall often omit the type subscript in Ext_A , and abbreviate $\text{Ext}(C, x, a)$ by $C *_x a$ and $\text{Ext}(C, x, 0^A)$ by $[C]_x$. We are now in a position to formulate the schema of *bar recursion*:¹⁶

$$\varphi(x, C) = \begin{cases} G(x, C) & \text{if } Y[C]_x < x, \\ H(\lambda a^A. \varphi(x+1, C *_x a), x, C) & \text{otherwise.} \end{cases}$$

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 in 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 $G(x, C) \triangleq 0$, $H(Z, x, C) \triangleq 1 + Z(1)$ and

$$Y(F) \triangleq \begin{cases} 0 & \text{if } F(m) = 1 \text{ for all } m, \\ n & \text{otherwise, where } n = \min\{m \mid F(m) \neq 1\}. \end{cases}$$

Let $1^{\mathbb{N} \rightarrow \mathbb{N}}$ be the constant 1 function. The crux of Y is that $Y[1^{\mathbb{N} \rightarrow \mathbb{N}}]_x = x$ for all x , so that the bar recursion is not well-founded. We calculate

$$\varphi(0, 1^{\mathbb{N} \rightarrow \mathbb{N}}) = 1 + \varphi(1, 1^{\mathbb{N} \rightarrow \mathbb{N}}) = \dots = n + \varphi(n, 1^{\mathbb{N} \rightarrow \mathbb{N}}) = \dots$$

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 .

¹⁵For the purpose of characterizing the provably recursive functions of analysis, yielding a consistency proof of analysis.

¹⁶Spector 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).

5D.1. DEFINITION. The theory *Spector's B*, notation λ_B is defined as follows. $\mathbb{T}(\lambda_B) = \mathbb{T}(\lambda_T)$. We use A^N as shorthand for the type $N \rightarrow A$. The *terms* of λ_B are obtained by adding constants for *bar recursion*

$$\begin{aligned}\mathbf{B}_{A,B} : (A^N \rightarrow N) \rightarrow (N \rightarrow A^N \rightarrow B) \rightarrow ((A \rightarrow B) \rightarrow N \rightarrow A^N \rightarrow B) \rightarrow N \rightarrow A^N \rightarrow B \\ \mathbf{B}^c_{A,B} : (A^N \rightarrow N) \rightarrow (N \rightarrow A^N \rightarrow B) \rightarrow ((A \rightarrow B) \rightarrow N \rightarrow A^N \rightarrow B) \rightarrow N \rightarrow A^N \rightarrow N \rightarrow B\end{aligned}$$

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 BYGH). The *reduction relation* \rightarrow_B of λ_B extends \rightarrow_T by adding the following (schematic) rules for the constants \mathbf{B}, \mathbf{B}^c (omitting type annotations A, B):

$$\begin{aligned}BYGHXC \rightarrow_B \mathbf{B}^c YGHXC(X \dashv Y[C]_X) \\ \mathbf{B}^c YGHXC(S^+ N) \rightarrow_B GXC \\ \mathbf{B}^c YGHXC 0 \rightarrow_B H(\lambda a. BYGH(S^+ X)(C *_X a)) XC\end{aligned}$$

The reduction rules for \mathbf{B}, \mathbf{B}^c require some explanation. First note that $x \dashv Y[C]_x = 0$ iff $Y[C]_x \geq x$, so that testing $x \dashv Y[C]_x = 0$ amounts to evaluating the (negation) of the bar condition. Consider a primitive recursive functional If_0 satisfying $If_0 0 M_1 M_0 = M_0$ and $If_0(S^+ P) M_1 M_0 = M_1$. A straightforward translation of the definition schema of bar recursion into a reduction rule:

$$BYGHXC \rightarrow If_0 (X \dashv [C]_X)(GXC)(H(\lambda x. BYGH(S^+ X)(C *_X x)) XC)$$

would lead to infinite reduction sequences (the innermost \mathbf{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 \mathbf{B} and \mathbf{B}^c .

Theorem 5C.5, Lemma 5C.6 and Theorem 5C.9 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 5C.8. The proof of SN for λ_B is considerably more difficult than for λ_T , which reflects the meta-mathematical 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 3C, 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 meta-mathematical 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 .

5D.2. DEFINITION. (i) For every set $\mathcal{X} \subseteq \Lambda_B$, let $nf(\mathcal{X})$ denote the set of B -normal forms of terms from \mathcal{X} . For all $\mathcal{X} \subseteq \Lambda_B(A \rightarrow B)$ and $\mathcal{Y} \subseteq \Lambda_B(A)$, let \mathcal{XY} denote the set of all applications of terms in \mathcal{X} to terms in \mathcal{Y} . Furthermore, if $M(x_1, \dots, x_k)$ is a term with free variables x_1, \dots, x_k , and $\mathcal{X}_1, \dots, \mathcal{X}_k$ 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 $M(\mathcal{X}_1, \dots, \mathcal{X}_k)$.

(ii) By induction on the type A we define that a set \mathcal{X} of closed terms of type A is *hereditarily finite*, notation $\mathcal{X} \in \mathcal{HF}_A$.

$$\begin{aligned}\mathcal{X} \in \mathcal{HF}_N &\iff \mathcal{X} \subseteq \Lambda_B^\varnothing(N) \cap \text{SN} \text{ and } nf(\mathcal{X}) \text{ is finite} \\ \mathcal{X} \in \mathcal{HF}_{A \rightarrow B} &\iff \mathcal{X} \subseteq \Lambda_B^\varnothing(A \rightarrow B) \text{ and } \mathcal{XY} \in \mathcal{HF}_B \text{ whenever } Y \in \mathcal{HF}_A\end{aligned}$$

(iii) A closed term M is called *hereditarily finite*, notation $M \in \mathbf{HF}^0$, if $\{M\} \in \mathcal{HF}$.

(iv) If $M(x_1, \dots, x_k)$ is a term all whose free variables occur among x_1, \dots, x_k , then $M(x_1, \dots, x_k)$ is *hereditarily finite*, notation $M(x_1, \dots, x_k) \in \mathbf{HF}$, if $M(\mathcal{X}_1, \dots, \mathcal{X}_k)$ is hereditarily finite for all $\mathcal{X}_i \in \mathcal{HF}$ of appropriate types ($1 \leq i \leq k$).

We will show in Theorem 5D.15 that every bar recursive term is hereditarily finite, and hence strongly normalizing.

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 $MN_1 \dots N_k$ and $\vec{\mathcal{XY}}$ stands for $\mathcal{XY}_1 \dots Y_k$. The first two lemmas are instrumental for proving hereditary finiteness.

5D.3. LEMMA. $\mathcal{X} \subseteq \Lambda_B^\varnothing(A_1 \rightarrow \dots \rightarrow A_n \rightarrow N)$ is hereditarily finite if and only if $\mathcal{X}\vec{Y} \in \mathcal{HF}_N$ for all $Y_1 \in \mathcal{HF}_{A_1}, \dots, Y_n \in \mathcal{HF}_{A_n}$.

PROOF. By induction on n , applying Definition 5D.2. ■

5D.4. DEFINITION. Given two sets of terms $\mathcal{X}, \mathcal{X}' \subseteq \Lambda_B^\varnothing$, 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 $A \equiv A_1 \rightarrow \dots \rightarrow A_n \rightarrow N$ with $n \geq 0$ and let $\mathcal{X}, \mathcal{X}' \subseteq \Lambda_B^\varnothing(A)$. We say that \mathcal{X} is *hereditarily adfluent* with \mathcal{X}' if $\mathcal{X}\vec{Y}$ is adfluent with $\mathcal{X}'\vec{Y}$, for all $Y_1 \in \mathcal{HF}_{A_1}, \dots, Y_n \in \mathcal{HF}_{A_n}$.

5D.5. LEMMA. Let $\mathcal{X}, \mathcal{X}' \subseteq \Lambda_B^\varnothing(A)$ 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 $A \equiv A_1 \rightarrow \dots \rightarrow A_n \rightarrow N$. Assume $\mathcal{X}' \in \mathcal{HF}_A$. Let $Y_1 \in \mathcal{HF}_{A_1}, \dots, Y_n \in \mathcal{HF}_{A_n}$, then $\mathcal{X}\vec{Y}$ is adfluent with $\mathcal{X}'\vec{Y}$. It follows that $\mathcal{X}\vec{Y} \subseteq \text{SN}$ since $\mathcal{X}'\vec{Y} \subseteq \text{SN}$ and $nf(\mathcal{X}\vec{Y}) \subseteq nf(\mathcal{X}'\vec{Y})$, so $nf(\mathcal{X}\vec{Y})$ is finite since $nf(\mathcal{X}'\vec{Y})$ is. Applying Lemma 5D.3 we obtain $\mathcal{X} \in \mathcal{HF}_A$. ■

Note that the above lemma holds in particular if $n = 0$, that is, if $A \equiv N$.

5D.6. LEMMA. Let A be a type of λ_B . Then

- (i) $\mathbf{HF}_A \subseteq \text{SN}$.
- (ii) $0^A \in \mathbf{HF}_A$.
- (iii) $\mathbf{HF}_A^0 \subseteq \text{SN}$.

PROOF. We prove (ii) and (iii) by simultaneous induction on A . Then (i) follows immediately. Obviously, $0 \in \mathbf{HF}_N$ and $\mathbf{HF}_N^0 \subseteq \text{SN}$. For the induction step $A \rightarrow B$, assume (ii) and (iii) hold for all smaller types. If $M \in \mathbf{HF}_{A \rightarrow B}^0$, then by the induction hypothesis (ii) $0^A \in \mathbf{HF}_A^0$, so $M0^A \in \mathbf{HF}_B^0$, so $M0^A$ is SN by the induction hypothesis (iii), and hence M is SN. Recall that $0^{A \rightarrow B} \equiv \lambda x^A.0^B$. Let $\mathcal{X} \in \mathcal{HF}_A$, then $\mathcal{X} \subseteq \text{SN}$ by the induction hypothesis. It follows that $0^{A \rightarrow B}\mathcal{X}$ is hereditarily adfluent with 0^B . By the induction hypothesis we have $0^B \in \mathbf{HF}_B$, so $0^{A \rightarrow B}\mathcal{X} \in \mathcal{HF}_B$ by Lemma 5D.5. Therefore $0^{A \rightarrow B} \in \mathbf{HF}_{A \rightarrow B}$. ■

The proofs of the following three lemmas are left to the reader.

5D.7. LEMMA. Every reduct of a hereditarily finite term is hereditarily finite.

5D.8. LEMMA. *Subsets of hereditarily finite sets of terms are hereditarily finite.*

In particular elements of a hereditarily finite set are hereditarily finite.

5D.9. LEMMA. *Finite unions of hereditarily finite sets are hereditarily finite.*

In this connection of course only unions of the same type make sense.

5D.10. LEMMA. *The hereditarily finite terms are closed under application.*

PROOF. Immediate from Definition 5D.2. ■

5D.11. LEMMA. *The hereditarily finite terms are closed under lambda abstraction.*

PROOF. Let $M(x, x_1, \dots, x_k) \in \mathcal{HF}$ be a term all whose free variables occur among x, x_1, \dots, x_k . We have to prove $\lambda x. M(x, x_1, \dots, x_k) \in \mathcal{HF}$, that is,

$$\lambda x. M(x, \vec{x}_1, \dots, \vec{x}_k) \in \mathcal{HF}$$

for given $\vec{x} = \vec{x}_1, \dots, \vec{x}_k \in \mathcal{HF}$ of appropriate types. Let $\mathcal{X} \in \mathcal{HF}$ be of the same type as the variable x , so $\mathcal{X} \subseteq \text{SN}$ by Lemma 5D.6. We also have $M(x, \vec{x}) \subseteq \text{SN}$ by the assumption on M and Lemma 5D.6. It follows that $(\lambda x. M(x, \vec{x}))\mathcal{X}$ is hereditarily adfluent with $M(\vec{x}, \vec{x})$. Again by the assumption on M we have that $M(\vec{x}, \vec{x}) \in \mathcal{HF}$, so that $(\lambda x. M(x, \vec{x}))\mathcal{X} \in \mathcal{HF}$ by Lemma 5D.5. We conclude that $\lambda x. M(x, \vec{x}) \in \mathcal{HF}$, so $\lambda x. M(x, x_1, \dots, x_k) \in \mathcal{HF}$. ■

5D.12. THEOREM. *Every term of Λ_T is hereditarily finite.*

PROOF. By Lemma 5D.10 and Lemma 5D.11, 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 0 are obviously hereditarily finite. Regarding S^+ , let $\mathcal{X} \in \mathcal{HF}_N$, then $S^+\mathcal{X} \subseteq \Lambda_B^\phi(N) \cap \text{SN}$ and $nf(S^+\mathcal{X})$ is finite since $nf(\mathcal{X})$ is finite. Hence $S^+\mathcal{X} \in \mathcal{HF}_N$, so S^+ is hereditarily finite. It remains to prove that the constants R_A are hereditarily finite. Let $M, N, \mathcal{X} \in \mathcal{HF}$ be of appropriate types and consider $R_A MN\mathcal{X}$. We have in particular $\mathcal{X} \in \mathcal{HF}_N$, so $nf(\mathcal{X})$ is finite, and the proof of $R_A MN\mathcal{X} \in \mathcal{HF}$ goes by induction on the largest numeral in $nf(\mathcal{X})$. If $nf(\mathcal{X}) = \{0\}$, then $R_A MN\mathcal{X}$ is hereditarily adfluent with M . Since $M \in \mathcal{HF}$ we can apply Lemma 5D.5 to obtain $R_A MN\mathcal{X} \in \mathcal{HF}$. For the induction step, assume $R_A MN\mathcal{X}' \in \mathcal{HF}$ 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 S^+n . Define

$$\mathcal{X}' \triangleq \{X \mid S^+X \text{ is a reduct of a term in } \mathcal{X}\}$$

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 $R_A MN\mathcal{X}' \in \mathcal{HF}$, so $N(R_A MN\mathcal{X}')\mathcal{X}' \in \mathcal{HF}$ and hence

$$N(R_A MN\mathcal{X}')\mathcal{X}' \cup M \in \mathcal{HF},$$

by Lemmas 5D.10, 5D.9. We have that $R_A MN\mathcal{X}$ is hereditarily adfluent with

$$N(R_A MN\mathcal{X}')\mathcal{X}' \cup M,$$

so $R_A MN\mathcal{X} \in \mathcal{HF}$ by Lemma 5D.5. This completes the induction step. ■

Before we can prove that B is hereditarily finite we need the following lemma.

5D.13. LEMMA. *Let $\mathcal{Y}, \mathcal{G}, \mathcal{H}, \mathcal{X}, \mathcal{C} \in \mathcal{HF}$ be of appropriate type. Then*

$$BYGHXC \in \mathcal{HF},$$

whenever $\text{BYGH}(S^+X)(C *_x A) \in \mathcal{H}\mathcal{F}$ for all $A \in \mathcal{H}\mathcal{F}$ of appropriate type.

PROOF. Let conditions be as above. Abbreviate BYGH by B and $B^c\text{YGH}$ by B^c . Assume $B(S^+X)(C *_x A) \in \mathcal{H}\mathcal{F}$ for all $A \in \mathcal{H}\mathcal{F}$. Below we will frequently and implicitly use that $\dashv, *, []$ 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 $BX\mathcal{C}$ is hereditarily adfluent with $B^cX\mathcal{C}(X \dashv Y[\mathcal{C}]_X)$, and hence with $\mathcal{G}X \cup \mathcal{H}(\lambda a.B(S^+X)(C *_x a))\mathcal{C}X$. It suffices to show that the latter set is in $\mathcal{H}\mathcal{F}$. We have $\mathcal{G}X \in \mathcal{H}\mathcal{F}$, so by Lemma 5D.9 the union is hereditarily finite if $\mathcal{H}(\lambda a.B(S^+X)(C *_x a))\mathcal{C}X$ is. It suffices that $\lambda a.B(S^+X)(C *_x a) \in \mathcal{H}\mathcal{F}$, and this will follow by the assumption above. We first observe that $\{0^A\} \in \mathcal{H}\mathcal{F}$ so $B(S^+X)(C *_x \{0^A\}) \in \mathcal{H}\mathcal{F}$ and hence $B(S^+X)(C *_x a) \subseteq \text{SN}$ by Lemma 5D.6. Let $A \in \mathcal{H}\mathcal{F}$. Since $B(S^+X)(C *_x a), A \subseteq \text{SN}$ we have that $(\lambda a.B(S^+X)(C *_x a))A$ is adfluent with $B(S^+X)(C *_x A) \in \mathcal{H}\mathcal{F}$ and hence hereditarily finite itself by Lemma 5D.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 Y, G, H, X and C such that $\text{BYGH}X\mathcal{C}$ is not hereditarily finite. We introduce the following abbreviations: B for BYGH and $X+n$ for $S^+(\dots(S^+X)\dots)$ (n times S^+). By Lemma 5D.13, there exists $U \in \mathcal{H}\mathcal{F}$ such that $B(X+1)(C *_x U)$ is not hereditarily finite. Hence again by Lemma 5D.13, there exists $V \in \mathcal{H}\mathcal{F}$ such that $B(X+2)((C *_x U) *_x V)$ is not hereditarily finite. Using dependent choice¹⁷, let

$$\mathcal{D} \triangleq C \cup (C *_x U) \cup ((C *_x U) *_x V) \cup \dots$$

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 5D.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 $\mathcal{D} \in \mathcal{H}\mathcal{F}$. Then we will arrive at a contradiction, since $Y\mathcal{D} \in \mathcal{H}\mathcal{F}$ implies that Y is bounded on \mathcal{D} , so that the bar condition is satisfied after finitely many steps, which conflicts with the construction process.

5D.14. LEMMA. *The set \mathcal{D} constructed above is hereditarily finite.*

PROOF. Let $N, Z \in \mathcal{H}\mathcal{F}$ be of appropriate type, that is, N of type N and Z such that $DN\vec{Z}$ is of type N . We have to show $DN\vec{Z} \in \mathcal{H}\mathcal{F}$. Since all elements of \mathcal{D} are hereditarily finite we have $DN\vec{Z} \subseteq \text{SN}$. By an easy generalization of Theorem 5C.9 we have WCR for λ_B , so by Newman's Lemma 5C.8 we have $DN\vec{Z} \subseteq \text{CR}$. Since $N \in \mathcal{H}\mathcal{F}$ it follows that $nf(N)$ is finite, say $nf(N) \subseteq \{0, \dots, n\}$ for n large enough. It remains to show that $nf(DN\vec{Z})$ is finite. Since all terms in $DN\vec{Z}$ are CR, their normal forms are unique. As a consequence we may apply a leftmost innermost reduction strategy to any term $DN\vec{Z} \in DN\vec{Z}$. At this point it might be helpful to remind the reader of the intended meaning of $*: C *_x A$

¹⁷The axiom of dependent choice DC states the following. Let $R \subseteq X^2$ be a binary relation on a set X such that $\forall x \in X \exists y \in X. R(x, y)$. Then $\forall x \in X \exists f : \text{Nat} \rightarrow X. [f(0) = x \ \& \ \forall n \in \text{Nat}. R(f(n), f(n + 1))]$. DC is an immediate consequence of the ordinary axiom of choice in set theory.

represents the finite sequence $C0, \dots, C(x-1), A$. More formally,

$$(C *_x A)y \triangleq \begin{cases} C(y) & \text{if } y < x, \\ A & \text{otherwise.} \end{cases}$$

With this in mind it is easily seen that $nf(\mathcal{D}\vec{\mathcal{N}}\vec{\mathcal{Z}})$ is a subset of $nf(\mathcal{D}_n\vec{\mathcal{N}}\vec{\mathcal{Z}})$, with

$$\mathcal{D}_n \triangleq \mathcal{C} \cup (\mathcal{C} *_{\mathcal{X}} \mathcal{U}) \cup ((\mathcal{C} *_{\mathcal{X}} \mathcal{U}) *_{\mathcal{X}+1} \mathcal{V}) \cup \dots \cup (\dots (\mathcal{C} *_{\mathcal{X}} \mathcal{U}) * \dots *_{\mathcal{X}+n} \mathcal{W})$$

a finite initial part of the infinite union \mathcal{D} . The set $nf(\mathcal{D}_n\vec{\mathcal{N}}\vec{\mathcal{Z}})$ is finite since the union is finite and all sets involved are in \mathcal{HF} . Hence \mathcal{D} is hereditarily finite by Lemma 5D.3. ■

Since \mathcal{D} is hereditarily finite, it follows that $nf(\mathcal{YD})$ is finite. Let k be larger than any numeral in $nf(\mathcal{YD})$. Consider

$$\mathcal{B}_k \triangleq \mathcal{B}(\mathcal{X}+k)(\dots (\mathcal{C} *_{\mathcal{X}} \mathcal{U}) * \dots *_{\mathcal{X}+k} \mathcal{W}')$$

as obtained in the construction above, iterating Lemma 5D.13, hence not hereditarily finite. Since k is a strict upper bound of $nf(\mathcal{YD})$ it follows that the set $nf((\mathcal{X}+k) \dashv \mathcal{YD})$ consists of numerals greater than 0, so that \mathcal{B}_k is hereditarily adfluent with $\mathcal{G}(\mathcal{X}+k)\mathcal{D}$. The latter set is hereditarily finite since it is an application of hereditarily finite sets (use Lemma 5D.14). Hence \mathcal{B}_k is hereditarily finite by Lemma 5D.5, which yields a plain contradiction.

By this contradiction, \mathcal{B} must be hereditarily finite, and so is \mathcal{B}^c , which follows by inspection of the reduction rules. As a consequence we obtain the main theorem of this section.

5D.15. THEOREM. *Every bar recursive term is hereditarily finite.*

5D.16. COROLLARY. *Every bar recursive term is strongly normalizable.*

5D.17. 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 \mathcal{B}^c and by enforcing every \mathcal{B} -redex to reduce via \mathcal{B}^c . Both Tait and Vogel use infinite terms. The proof above is based on [Bezem \[1985a\]](#) 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 η -reduction, then the above results can also be obtained in a similar way as for λ_T with η -reduction.

Semantics of λ_B

In this section we give some interpretations of Spector's \mathcal{B} .

5D.18. DEFINITION. A model of λ_B is a model of λ_T with interpretations of the constants $\mathcal{B}_{A,B}$ and $\mathcal{B}_{A,B}^c$ 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 $[\varphi] = [\mathbf{BYGH}]$.

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. For this *closed term model* the reader is asked in Exercise 5F.37 to prove that it is extensional. An important 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 a finite amount of information about their arguments. On the other hand continuity ensures that bar recursion is well-founded, since a continuous Y eventually gets the constant value YC on increasing initial segments $[C]_x$. In Exercise 5F.36 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 in Kleene [1959b] using the ‘S1-S9 recursive functionals’. Computability alone, without uniform continuity on all binary functions, does not yield a model of bar recursion, see Exercise 5F.32. 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. See Bezem [1985b] for a closely related model based on strongly majorizable functionals.

5D.19. DEFINITION (*Hereditarily finite functionals*). Recall the full type structure over the natural numbers: $\mathcal{M}_N \triangleq \mathbb{N}$ and $\mathcal{M}_{A \rightarrow B} \triangleq \mathcal{M}_A \rightarrow \mathcal{M}_B$. A set $X \subseteq \mathcal{M}_N$ is hereditarily finite if X is finite. A set $X \subseteq \mathcal{M}_{A \rightarrow B}$ is hereditarily finite if $X\mathcal{Y} \subseteq \mathcal{M}_B$ is hereditarily finite for every hereditarily finite $\mathcal{Y} \subseteq \mathcal{M}_A$. Here and below, $X\mathcal{Y}$ denotes the set of all results that can be obtained by applying functionals from 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 5D.13:

5D.20. LEMMA. *Let $\mathcal{Y}, \mathcal{G}, \mathcal{H}, \mathcal{X}, \mathcal{C}$ be hereditarily finite sets of appropriate type. Then $[\mathbb{B}]\mathcal{Y}\mathcal{G}\mathcal{H}\mathcal{X}\mathcal{C}$ is well defined and hereditarily finite whenever $[\mathbb{B}]\mathcal{Y}\mathcal{G}\mathcal{H}(\mathcal{X} + 1)(\mathcal{C} *_{\mathcal{X}} \mathcal{A})$ 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 5D.13. The set of longer and longer initial sequences with elements taken from hereditarily finite sets (cf. the set \mathcal{D} in Lemma 5D.14) 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.

Following Troelstra [1973], Section 2.4.5 and 2.7.2, we define the following notion of *hereditary extensional equality*.

5D.21. DEFINITION. We put \approx_N to be $=$, convertibility of closed terms in $\Lambda_B^\phi(\mathbb{N})$. For the type $A \equiv B \rightarrow B'$ we define $M \approx_A M'$ if and only if $M, M' \in \Lambda_B^\phi(A)$ 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 $M \in \Lambda_B^\phi(A)$ and A . This is proved in Exercise 5F.37.

5E. 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 $(A \rightarrow A) \rightarrow A$ such that YF is a fixed point of F , that is, $YF = F(YF)$, for every F of type $A \rightarrow A$. 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 5C. We can rephrase these equations as

$$RMNn = \text{If}_0 n (N(RMN(n-1))(n-1))M,$$

where $\text{If}_0 n M_1 M_0 = M_0$ if $n = 0$ and M_1 if $n > 0$. Hence we can write

$$\begin{aligned} RMN &= \lambda n. \text{If}_0 n (N(RMN(n-1))(n-1))M \\ &= (\lambda f n. \text{If}_0 n (N(f(n-1))(n-1))M)(RMN) \end{aligned}$$

This equation is of the form $YF = F(YF)$ with

$$F \triangleq \lambda f n. \text{If}_0 n (N(f(n-1))(n-1))M$$

and $YF = RMN$. It is easy to see that YF satisfies the recursion equation for RMN uniformly in M, N . This shows that, given functionals If_0 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 $G_F \triangleq \lambda n. \text{If}_0 F(n) G_F(n+1) n$. 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 $G_F(1), G_F(2), \dots$ 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 $G_F(0) = G_F(1) = G_F(2) = \dots$ 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 .

5E.1. DEFINITION. The theory [Platek's \$\mathcal{Y}\$](#) , notation λ_Y , is defined as follows. $\mathbb{T}(\lambda_Y) \triangleq \mathbb{T}(\lambda_T) = \mathbb{T}^{\{\mathbb{N}\}}$. The *terms* of λ_Y are obtained by adding constants

$$\textcolor{blue}{Y}_A : (A \rightarrow A) \rightarrow A$$

for all types A to the constants of λ_T . The set of (closed) terms of λ_Y (of type A) is denoted by $\Lambda_Y^\emptyset(A)$. 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 $\textcolor{blue}{Y}F = F(\textcolor{blue}{Y}F)$ for all appropriate types. The *reduction relation* \rightarrow_Y of λ_Y extends \rightarrow_T by adding the following rule for the constants $\textcolor{blue}{Y}$ (omitting type annotations A):

$$\textcolor{blue}{Y} \rightarrow_Y \lambda f.f(\textcolor{blue}{Y}f).$$

The reduction rule for $\textcolor{blue}{Y}$ requires some explanation, as the rule $\textcolor{blue}{Y}F \rightarrow F(\textcolor{blue}{Y}F)$ seems simpler. However, with the latter rule we would have diverging reductions $\lambda f.\textcolor{blue}{Y}f \rightarrow_\eta \textcolor{blue}{Y}$ and $\lambda f.\textcolor{blue}{Y}f \rightarrow_Y \lambda f.f(\textcolor{blue}{Y}f)$ that cannot be made to converge, so that we would lose CR of \rightarrow_Y in combination with η -reduction.

The SN property does not hold for λ_Y : the term $\textcolor{blue}{Y}$ does not have a $\textcolor{blue}{Y}$ -nf. However, the Church-Rosser property for λ_Y with β -reduction and with $\beta\eta$ -reduction can be proved by standard techniques from higher-order rewriting theory, for example, by using weak orthogonality, see [van Raamsdonk \[1996\]](#).

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. For example, λ_Y is inherently sequential: there is no 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. This subject is more thoroughly studied in domain theory, see e.g. [Gunter \[1992\]](#) or [Abramsky and Jung \[1994\]](#).

5E.2. DEFINITION. A *model of λ_Y* is a model of λ_T with interpretations of the constants $\textcolor{blue}{Y}_A$ 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 partialness. 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 have to

interpret terms like YS^+ . 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, see for example B[1984], Section 1.2). The next lines bear some similarity to the introductory treatment of ordinals in Section 5C. We call a set *directed* if it is not empty and 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 $\text{lfp}(f)$, 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.

5E.3. DEFINITION. Define \mathbb{N}_A^\perp by induction on A .

$$\begin{aligned}\mathbb{N}_\mathbb{N}^\perp &\triangleq \mathbb{N}^\perp, \\ \mathbb{N}_{A \rightarrow B}^\perp &\triangleq [\mathbb{N}_A^\perp \rightarrow \mathbb{N}_B^\perp], \text{ the set of all continuous maps.}\end{aligned}$$

Given the fact that cpo-s with continuous maps 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 $[\text{Y}] = \text{lfp}$ for all appropriate types.

5E.4. 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 maps on cpo-s. This option is elaborated in detail in van Draanen [1995].

5F. Exercises

5F.1. Prove in δ the following equations.

- (i) $\delta MN\mathbf{K}_*\mathbf{K} = \delta(\delta MN)\mathbf{K}_*$.
- (ii) $\delta(\lambda z.\delta(Mz)(Nz))(\lambda z.\mathbf{K}) = \delta MN$.

[Hint. Start observing that $\delta(Mz)(Nz)(Mz)(Nz) = Nz$.]

5F.2. Prove Proposition 5B.12: for all types A one has $A \triangleleft_{SP} N_{rk(A)}$.

5F.3. Let λ_P be λ_\rightarrow^0 extended with a simple (not surjective) pairing. Show that Theorem 5B.45 does not hold for this theory. [Hint show that in this theory the equation $\lambda x:0.\langle \pi_1 x, \pi_2 x \rangle = \lambda x:0.x$ does not hold by constructing a counter model, but is nevertheless consistent.]

5F.4. Does every model of λ_{SP} have the same first order theory?

- 5F.5. (i) Show that if a pairing function $\langle , \rangle : 0 \rightarrow (0 \rightarrow 0)$ and projections $L, R : 0 \rightarrow 0$ satisfying $L\langle x, y \rangle = x$ and $R\langle x, y \rangle = y$ are added to λ^0_\rightarrow , then for a non-trivial model \mathcal{M} one has (see 4.2)

$$\forall A \in \mathbb{T} \forall M, N \in \Lambda^\varnothing(A) [\mathcal{M} \models M = N \Rightarrow M =_{\beta\eta} N].$$

- (ii) (Schwichtenberg and Berger [1991]) Show that for \mathcal{M} a model of λ_T one has (see 4.3)

$$\forall A \in \mathbb{T} \forall M, N \in \Lambda^\varnothing(A) [\mathcal{M} \models M = N \Rightarrow M =_{\beta\eta} N].$$

- 5F.6. Show that $\mathcal{F}[x_1, \dots, x_n]$ for $n \geq 0$ does not have one generator. [Hint. Otherwise this monoid would be commutative, which is not the case.]

- 5F.7. Show that $R \subseteq \Lambda^\varnothing(A) \times \Lambda^\varnothing(B)$ is equational iff

$$\exists M, N \in \Lambda^\varnothing(A \rightarrow B \rightarrow 1 \rightarrow 1) \forall F [R(F) \Leftrightarrow MF = NF].$$

- 5F.8. Show that there is a Diophantine equation $1t \subseteq \mathcal{F}^2$ such that for all $n, m \in \mathbb{N}$

$$1t(R^n, R^m) \Leftrightarrow n < m.$$

- 5F.9. Define $\text{Seq}_n^{\mathcal{N}_k}(h)$ if $h = [R^{m_0}, \dots, R^{m_{n-1}}]$, for some $m_0, \dots, m_{n-1} < k$. Show that $\text{Seq}_n^{\mathcal{N}_k}$ is Diophantine uniformly in n .

- 5F.10. Let \mathcal{B} be some finite subset of \mathcal{F} . Define $\text{Seq}_n^{\mathcal{B}}(h)$ if $h = [g_0, \dots, g_{n-1}]$, with each $g_i \in \mathcal{B}$. Show that $\text{Seq}_n^{\mathcal{B}}$ is Diophantine uniformly in n .

- 5F.11. 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.

- 5F.12. Show that $\mathcal{F} \subseteq \mathcal{F}[x]$ is Diophantine.

- 5F.13. 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

$$f \in \{R^n \mid n \in \mathbb{N}\} \cup \{L\} \Leftrightarrow \exists g \in \mathcal{F} [t(f, g) = s(f, g)].$$

[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.]

- 5F.14. 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 $Z, O : \mathbf{C} \rightarrow \mathbf{C}$ ‘shift operators’ on Cantor space as follows.

$$\begin{aligned} Z(f)(0) &\triangleq 0; \\ Z(f)(n+1) &\triangleq f(n); \\ O(f)(0) &\triangleq 1; \\ O(f)(n+1) &\triangleq f(n). \end{aligned}$$

Write $0f = Z(f)$ and $1f = O(f)$. If $\mathcal{X} \subseteq \mathbf{C} \rightarrow \mathbf{C}$ 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 following.

$$\begin{aligned} I &\triangleq \lambda x \in \{Z, O\}^+. z; \\ L &\triangleq Z; \\ R &\triangleq O; \\ x * y &\triangleq y \circ x; \\ \langle x, y \rangle &\triangleq \begin{cases} x(f), & \text{if } f(0) = 0; \\ y(f), & \text{if } f(0) = 1. \end{cases} \end{aligned}$$

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 introduced in Definition 5B.32 generate this group.

5F.15. Let

$$\begin{aligned} B_0 &\triangleq \langle LL, RL, R \rangle & B_0^{-1} &\triangleq \langle \langle L, LR \rangle, LRR, RRR \rangle \rangle \\ B_1 &\triangleq \langle L, LLR, RLR, RR \rangle & B_1^{-1} &\triangleq \langle L, \langle LR, LRR \rangle, RRR \rangle \rangle \\ C_0 &\triangleq \langle R, L \rangle \\ C_1 &\triangleq \langle LR, L, RR \rangle \rangle. \end{aligned}$$

Show that for the invertible elements of the free Cartesian monoid \mathcal{F} one has

$$\mathcal{I} = [B_0, B_0^{-1}, B_1, B_1^{-1}, C_0, C_1].$$

[Hint. Show that

$$\begin{aligned} B_0 \langle A, B, C \rangle &= \langle A, B, C \rangle \\ B_1 \langle A, \langle B, C \rangle, D \rangle &= \langle A, B, C, D \rangle \\ C_0 \langle A, B \rangle &= \langle B, A \rangle \\ C_1 \langle A, B, C \rangle &= \langle B, A, C \rangle. \end{aligned}$$

Use this to transform any element $M \in \mathcal{I}$ into \mathbb{I} . By the inverse transformation we get M as the required product.]

5F.16. Show that the B_n in Definition 5B.32 satisfy

$$B_{n+2} = B_n B_{n+1} B_n^{-1}.$$

5F.17. Prove Proposition 5B.12: for all types A one has $A \triangleleft_{SP} N_{rank(A)}$.

5F.18. Does every model of λ_{SP} have the same first order theory?

5F.19. Prove the Lemma 5C.15. [Hint. Use the following procedure:

- (i) To be proved by induction on α ;
- (ii) Prove $\alpha \leq \beta \Rightarrow f^\alpha(0) \leq f^\beta(0)$ by induction on β ;
- (iii) Assume $f(\beta) = \beta$ and prove $f^\alpha(0) \leq \beta$ by induction on α ;

- (iv) Prove $\alpha < \beta \Rightarrow f^\alpha(0) < f^\beta(0)$ for all α, β such that $f^\alpha(0)$ is below any fixed point, by induction on β .]
- 5F.20. Justify the equation $f(\lambda) = \lambda$ in the proof of 5C.17.
- 5F.21. Let A be the Ackermann function. Calculate $A(3, m)$ and verify that $A(4, 0) = 13$ and $A(4, 1) = 65533$.
- 5F.22. With one occurrence hidden in H , the term RSH contains $R_{\mathbb{N} \rightarrow \mathbb{N}}$ twice. Define A using R_N and $R_{\mathbb{N} \rightarrow \mathbb{N}}$ only once. Is it possible to define A with R_N only, possibly with multiple occurrences?
- 5F.23. 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 .
- 5F.24. 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.
- 5F.25. [Simultaneous primitive recursion] Assume G_i, H_i ($i = 1, 2$) have been given and define F_i ($i = 1, 2$) as follows.

$$\begin{aligned} F_i(0, \vec{x}) &\triangleq G_i(\vec{x}); \\ F_i(n + 1, \vec{x}) &\triangleq H_i(F_1(n, \vec{x}), (F_2(n, \vec{x}), n, \vec{x})). \end{aligned}$$

Show that F_i ($i = 1, 2$) can be defined by first-order primitive recursion. [Hint. Use a pairing function such as in Figure 12.]

- 5F.26. [Nested recursion, Péter [1967]] Define

$$\begin{aligned} F(n, m) &\triangleq 0, && \text{if } m \cdot n = 0; \\ F(n + 1, m + 1) &\triangleq G(m, n, F(m, n, F(m, n, F(m + 1, n))), F(m + 1, n)). \end{aligned}$$

Show that F can be defined from G, H using higher-order primitive recursion.

- 5F.27. [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 arithmetical 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 $\varphi^D \triangleq \exists \vec{x} \forall \vec{y} \varphi_D(\vec{x}, \vec{y})$ 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 $\varphi^D \triangleq \varphi_D \triangleq \varphi$.

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. $(\varphi \wedge \psi)^D \triangleq \exists \vec{x}, \vec{u} \forall \vec{y}, \vec{v} (\varphi \wedge \psi)_D$, with
 $(\varphi \wedge \psi)_D \triangleq (\varphi_D(\vec{x}, \vec{y}) \wedge \psi_D(\vec{u}, \vec{v}))$.
3. $(\varphi \vee \psi)^D \triangleq \exists z, \vec{x}, \vec{u} \forall \vec{y}, \vec{v} (\varphi \vee \psi)_D$, with
 $(\varphi \vee \psi)_D \triangleq ((z = 0 \Rightarrow \varphi_D(\vec{x}, \vec{y})) \wedge (z \neq 0 \Rightarrow \psi_D(\vec{u}, \vec{v})))$.
4. $(\varphi \Rightarrow \psi)^D \triangleq \exists \vec{u}', \vec{y}' \forall \vec{x}, \vec{v} (\varphi \Rightarrow \psi)_D$, with
 $(\varphi \Rightarrow \psi)_D \triangleq (\varphi_D(\vec{x}, \vec{y}') \vec{x} \vec{v}) \Rightarrow \psi_D(\vec{u}' \vec{x}, \vec{v})$.

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 below, assume $\varphi^D(z) \equiv \exists \vec{x} \forall \vec{y} \varphi_D(\vec{x}, \vec{y}, z)$.

5. $(\exists z \varphi(z))^D \triangleq \exists z, \vec{x} \forall \vec{y} (\exists z \varphi(z))_D$, with $(\exists z \varphi(z))_D \triangleq \varphi_D(\vec{x}, \vec{y}, z)$.
6. $(\forall z \varphi(z))^D \triangleq \exists \vec{x}' \forall z, \vec{y} (\forall z \varphi(z))_D$, with $(\forall z \varphi(z))_D \triangleq \varphi_D(\vec{x}' z, \vec{y}, z)$.
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 in this way the D-interpretation of $(\varphi \Rightarrow (\varphi \vee \psi))^D$ is *validated* by higher-order primitive recursive functionals. Validate the D-interpretation of $(\varphi \Rightarrow (\varphi \wedge \psi))^D$. 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.

- 5F.28. 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} .
5F.29. 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 .

By induction we define a family of sets, the *hereditarily recursive operators* $HRO_B \subseteq \mathbb{N}$ for every type B , as follows.

$$\begin{aligned} HRO_{\mathbb{N}} &\triangleq \mathbb{N} \\ HRO_{B \rightarrow B'} &\triangleq \{x \in \mathbb{N} \mid x * y \in HRO_{B'} \text{ for all } y \in HRO_B\} \end{aligned}$$

Prove that HRO with Kleene application constitutes a model for Gödel's \mathcal{T} .

- 5F.30. By simultaneous induction we define a family of sets, the *hereditarily extensional operators* $HEO_B \subseteq \mathbb{N}$ for every type B , equipped with an equivalence relation $=_B$ as follows.

$$\begin{aligned} HEO_{\mathbb{N}} &\triangleq \mathbb{N} \\ x =_{\mathbb{N}} y &\iff x = y \\ HEO_{B \rightarrow B'} &\triangleq \{x \in \mathbb{N} \mid x * y \in HEO_{B'} \text{ for all } y \in HEO_B \text{ and} \\ &\quad x * y =_B x * y' \text{ for all } y, y' \in HEO_B \text{ with } y =_B y'\} \\ x =_{B \rightarrow B'} x' &\iff x, x' \in HEO_{B \rightarrow B'} \text{ and } x * y =_B x' * y \text{ for all } y \in HEO_B. \end{aligned}$$

Prove that HEO with Kleene application constitutes a model for Gödel's \mathcal{T} .

- 5F.31. Recall that extensionality essentially means that objects having the same applicative behavior can be identified. Which of the above models of λ_T , the full type structure, the closed term model, HRO and HEO, is extensional?
- 5F.32. 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 .
- 5F.33. Verify that the redefinition of the ordinal arithmetic in Example 5C.14 is correct.
- 5F.34. Prove Lemma 5C.15. More precisely:
- (i) To be proved by induction on α ;
 - (ii) Prove $\alpha \leq \beta \Rightarrow f^\alpha(0) \leq f^\beta(0)$ by induction on β ;
 - (iii) Assume $f(\beta) = \beta$ and prove $f^\alpha(0) \leq \beta$ by induction on α ;
 - (iv) Prove $\alpha < \beta \Rightarrow f^\alpha(0) < f^\beta(0)$ for all α, β such that $f^\alpha(0)$ is below any fixed point, by induction on β .
- 5F.35. Justify the equation $f(\lambda) = \lambda$ in the proof of Lemma 5C.17.
- 5F.36. This exercise introduces the *continuous functionals*, Kleene [1959a]. Define for $f, g \in \mathbb{N} \rightarrow \mathbb{N}$ the (partial) application of f to g by $f(g) = f(\overline{gn}) - 1$, where n is the smallest number such that $f(\overline{gn}) > 0$, provided there is such n . If there is no such n , then $f * g$ is undefined. The idea is that f uses only a finite amount of 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 $\mathbb{N} \rightarrow \mathbb{N}$ and elements of \mathcal{C}_A . For the base type we put $\mathcal{C}_{\mathbb{N}} = \mathbb{N}$ and let the constant functions be the associates of the corresponding natural numbers. For higher types we define that $f \in \mathbb{N} \rightarrow \mathbb{N}$ is an associate of $F \in \mathcal{C}_A \rightarrow \mathcal{C}_B$ 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 $\mathcal{C}_{A \rightarrow B}$ is defined as the subset of those $F \in \mathcal{C}_A \rightarrow \mathcal{C}_B$ that have an associate. Show that \mathcal{C} is a model for bar recursion.
- 5F.37. Show that for any closed term $M \in \Lambda_B^\varnothing$ one has $M \approx M$, see Definition 5D.21. [Hint. Type subscripts are omitted. Define a predicate $\text{Ext}(M(\vec{x}))$ for any open term M with free variables among $\vec{x} = x_1, \dots, x_n$ by

$$M(X_1, \dots, X_n) \approx M(X'_1, \dots, X'_n)$$

for all $X_1, \dots, X_n, X'_1, \dots, X'_n \in \Lambda_B^\varnothing$ with $X_1 \approx X'_1, \dots, X_n \approx X'_n$. Then prove by induction on terms that Ext holds for any open term, so in particular for closed

terms. For \mathbf{B} , prove first the following. Suppose

$$Y \approx Y', G \approx G', H \approx H', X \approx X', C \approx C',$$

and for all $A \approx A'$

$$\mathbf{B}YGH(S^+X)(C *_X A) \approx \mathbf{B}Y'G'H'(S^+X')(C' *_X' A')$$

then

$$\mathbf{B}YGHXC \approx \mathbf{B}Y'G'H'X'C'.]$$

- 5F.38. It is possible to define λ_Y as an extension of λ_\rightarrow^0 using the Church numerals $\mathbf{c}_n \triangleq \lambda x^{\mathbb{N}} f^{\mathbb{N} \rightarrow \mathbb{N}}. f^n x$. Show that every partial recursive function is also definable in this version of λ_Y .

CHAPTER 6

APPLICATIONS

6A. 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 λ -calculi exist in untyped and typed versions. There are also other differences in the various lambda calculi. The λ -calculus introduced in [Church \[1936\]](#) is the untyped λI -calculus in which an abstraction $\lambda x.M$ is only allowed if x occurs among the free variables of M . Nowadays, “ λ -calculus” refers to the λK -calculus developed under the influence of Curry, in which $\lambda x.M$ is allowed even if x does not occur in M . This book treats the typed versions of the lambda calculus. Of these, the most elementary are the versions of the simply typed λ -calculus $\lambda \rightarrow^{\mathbb{A}}$ introduced in Chapter 1.

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 λ -calculi.

Lambda definability was introduced for functions on the set of natural numbers \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, λ -calculus is strong enough to do this, as was emphasized in [Böhm \[1966\]](#) 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, Piperno, and Guerrini \[1994\]](#) or [Berarducci and Böhm \[1993\]](#). 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 simplified syntax.

$$\begin{aligned}\text{tree} &\triangleq \bullet \mid \text{leaf } \text{nat} \mid \text{tree} + \text{tree} \\ \text{nat} &\triangleq 0 \mid \text{succ } \text{nat}\end{aligned}$$

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 $(\text{leaf } 3) + ((\text{leaf } 5) + \bullet)$. This tree together with its mirror image look as follows ('leaf 3' is essentially 3, but we officially need to write the constructor to warrant unicity of types; in the examples below we do not write it).



Operations on such trees can be defined by recursion. For example the action of mirroring can be defined by

$$\begin{aligned}f_{\text{mir}}(\bullet) &\triangleq \bullet; \\ f_{\text{mir}}(\text{leaf } n) &\triangleq \text{leaf } n; \\ f_{\text{mir}}(t_1 + t_2) &\triangleq f_{\text{mir}}(t_2) + f_{\text{mir}}(t_1).\end{aligned}$$

Then one has for example that

$$f_{\text{mir}}((\text{leaf } 3) + ((\text{leaf } 5) + \bullet)) = ((\bullet + \text{leaf } 5) + \text{leaf } 3).$$

We will now show in two different ways how trees can be represented as lambda terms and how operations like f_{mir} on these objects become lambda definable. The first method is from [Böhm and Berarducci \[1985\]](#). The resulting data objects and functions can be represented by lambda terms typable in the second order lambda calculus $\lambda 2$, see [Girard, Lafont, and Taylor \[1989\]](#) or [Barendregt \[1992\]](#).

6A.1. DEFINITION. (i) Let b, l, p be variables (used as mnemonics for bud, leaf and plus). Define $\varphi = \varphi^{b, l, p} : \text{tree} \rightarrow \text{term}$, where term is the collection of untyped lambda terms, as follows.

$$\begin{aligned}\varphi(\bullet) &\triangleq b; \\ \varphi(\text{leaf } n) &\triangleq l^{\lceil n \rceil}; \\ \varphi(t_1 + t_2) &\triangleq p \varphi(t_1) \varphi(t_2).\end{aligned}$$

Here $\lceil n \rceil \equiv \lambda f x. f^n x$ is Church's numeral representing n as lambda term.

(ii) Define $\psi_1 : \text{tree} \rightarrow \text{term}$ as follows.

$$\psi_1(t) \triangleq \lambda b l p. \varphi(t).$$

6A.2. PROPOSITION. Define

$$\begin{aligned}B_1 &\triangleq \lambda b l p. b; \\ L_1 &\triangleq \lambda n b l p. l n; \\ P_1 &\triangleq \lambda t_1 t_2 b l p. p(t_1 b l p)(t_2 b l p).\end{aligned}$$

Then one has

- (i) $\psi_1(\bullet) = B_1$.
- (ii) $\psi_1(\text{leaf } n) = L_1 \lceil n \rceil$.
- (iii) $\psi_1(t_1 + t_2) = P_1 \psi_1(t_1) \psi_1(t_2)$.

PROOF. (i) Trivial.

(ii) We have

$$\begin{aligned}\psi_1(\text{leaf } n) &= \lambda b l p. \varphi(\text{leaf } n) \\ &= \lambda b l p. l \lceil n \rceil \\ &= (\lambda n b l p. l n) \lceil n \rceil \\ &= L_1 \lceil n \rceil.\end{aligned}$$

(iii) Similarly, using that $\psi_1(t) b l p = \varphi(t)$. ■

This Proposition states that the trees we considered are representable as lambda terms in such a way that the constructors $(\bullet, \text{leaf}$ and $+$) are lambda definable. In fact, the lambda terms involved can be typed in $\lambda 2$. A nice connection between these terms and proofs in second order logic is given in [Leivant \[1983b\]](#).

Now we will show that iterative functions over these trees, like f_{mir} , are lambda definable.

6A.3. PROPOSITION (Iteration). *Given lambda terms A_0, A_1, A_2 there exists a lambda term F such that (for variables n, t_1, t_2)*

$$\begin{aligned}F B_1 &= A_0; \\ F(L_1 n) &= A_1 n; \\ F(P_1 t_1 t_2) &= A_2(Ft_1)(Ft_2).\end{aligned}$$

PROOF. Take $F \triangleq \lambda w.w A_0 A_1 A_2$. ■

As is well known, primitive recursive functions can be obtained from iterative functions.

There is a way of coding a finite sequence of lambda terms M_1, \dots, M_k as one lambda term

$$\langle M_1, \dots, M_k \rangle \triangleq \lambda z.z M_1 \dots M_k$$

such that the components can be recovered. Indeed, take

$$U_k^i \triangleq \lambda x_1 \dots x_k. x_i,$$

then

$$\langle M_1, \dots, M_k \rangle U_k^i = M_i.$$

6A.4. COROLLARY (Primitive recursion). *Given lambda terms C_0, C_1, C_2 there exists a lambda term H such that*

$$\begin{aligned}H B_1 &= C_0; \\ H(L_1 n) &= C_1 n; \\ H(P_1 t_1 t_2) &= C_2 t_1 t_2 (Ht_1)(Ht_2).\end{aligned}$$

PROOF. Define the auxiliary function $F \triangleq \lambda t. \langle t, Ht \rangle$. Then by the Proposition F can be defined using iteration. Indeed,

$$F(P_1 t_1 t_2) = \langle P_1 t_1 t_2, H(P_1 t_1 t_2) \rangle = A_2(Ft_1)(Ft_2),$$

with

$$A_2 \triangleq \lambda t_1 t_2. \langle P(t_1 U_2^1)(t_2 U_2^1), C_2(t_1 U_2^1)(t_2 U_2^1)(t_1 U_2^2)(t_2 U_2^2) \rangle.$$

Now take $H = \lambda t. F t U_2^2$. [This was a trick Kleene found at the dentist treated under laughing-gas, see Kleene [1975].] ■

Now we will present the method of Böhm, Piperno, and Guerrini [1994] and Berarducci and Böhm [1993] to represent data types. Again we consider the example of labelled trees.

6A.5. DEFINITION. Define $\psi_2 : \text{tree} \rightarrow \text{term}$ as follows.

$$\begin{aligned}\psi_2(\bullet) &\triangleq \lambda e. e U_3^1 e; \\ \psi_2(\text{leaf } n) &\triangleq \lambda e. e U_3^2 \lceil n \rceil e; \\ \psi_2(t_1 + t_2) &\triangleq \lambda e. e U_3^3 \psi_2(t_1) \psi_2(t_2) e.\end{aligned}$$

Then the basic constructors for labeled trees are definable by

$$\begin{aligned}B_2 &\triangleq \lambda e. e U_3^1 e; \\ L_2 &\triangleq \lambda n \lambda e. e U_3^2 n e; \\ P_2 &\triangleq \lambda t_1 t_2 \lambda e. e U_3^3 t_1 t_2 e.\end{aligned}$$

6A.6. PROPOSITION. *Given lambda terms A_0, A_1, A_2 there exists a term F such that*

$$\begin{aligned}FB_2 &= A_0 F; \\ F(L_2 n) &= A_1 n F; \\ F(P_2 x y) &= A_2 x y F.\end{aligned}$$

PROOF. Try $F \triangleq \langle\langle X_0, X_1, X_2 \rangle\rangle$, the 1-tuple of a triple. Then we must have

$$\begin{aligned}FB_2 &= B_2 \langle X_0, X_1, X_2 \rangle \\ &= U_3^1 X_0 X_1 X_2 \langle X_0, X_1, X_2 \rangle \\ &= X_0 \langle X_0, X_1, X_2 \rangle \\ &= A_0 \langle\langle X_0, X_1, X_2 \rangle\rangle \\ &= A_0 F,\end{aligned}$$

provided $X_0 = \lambda x. A_0 \langle x \rangle$. Similarly one can find X_1, X_2 . ■

This second representation is essentially untypable, at least in typed λ -calculi in which all typable terms are normalizing. This follows from the following consequence of a result similar to Proposition 6A.6. Let $K = \lambda x y. x$, $K_* = \lambda x y. y$ represent true and false respectively. Then writing

if bool then X else Y fi

for

bool X Y,

the usual behavior of the conditional is obtained. Now if we represent the natural numbers as a data type in the style of the second representation, we immediately get that the lambda definable functions are closed under minimization. Indeed, let

$$\chi(x) = \mu y [g(x, y) = 0],$$

and suppose that g is lambda defined by G . Then there exists a lambda term H such that

$$Hxy = \text{if } \underline{\text{zero?}}(Gxy) \text{ then } y \text{ else } (Hx(\underline{\text{succ}}\ y)) \text{ fi.}$$

Indeed, we can write this as $Hx = AxH$ and apply Proposition 6A.6, but now formulated for the inductively defined type `num`. Then $F \triangleq \lambda x.Hx^{\lceil 0 \rceil}$ does represent χ . Here `succ` represents the successor function and `zero?` a test for zero; both are lambda definable, again by the analogon to Proposition 6A.6. Since minimization enables us to define all partial recursive functions, the terms involved cannot be typed in a normalizing system.

Self-interpretation

A lambda term M can be represented internally as a lambda term ${}^\lceil M \rceil$. This representation should be such that, for example, one has lambda terms P_1, P_2 satisfying $P_i^{\lceil X_1 X_2 \rceil} = X_i$. Kleene [1936] already showed that there is a ('meta-circular') self-interpreter E such that, for closed terms M one has $E^{\lceil M \rceil} = M$. The fact that data types can be represented directly in the λ -calculus was exploited by Mogensen [1992] to find a simpler representation for ${}^\lceil M \rceil$ and E .

The difficulty of representing lambda terms internally is that they do not form a first order algebraic data type due to the binding effect of the lambda. Mogensen [1992] solved this problem as follows. Consider the data type with signature

`const`, `app`, `abs`

where `const` and `abs` are unary, and `app` is a binary constructor. Let `const`, `app` and `abs` be a representation of these in λ -calculus (in the style of Definition 6A.5).

6A.7. PROPOSITION (Mogensen [1992]). Define

$$\begin{aligned} {}^\lceil x \rceil &\triangleq \underline{\text{const}}\ x; \\ {}^\lceil PQ \rceil &\triangleq \underline{\text{app}}\ {}^\lceil P \rceil {}^\lceil Q \rceil; \\ {}^\lceil \lambda x.P \rceil &\triangleq \underline{\text{abs}}(\lambda x.{}^\lceil P \rceil). \end{aligned}$$

Then there exists a self-interpreter E such that for all lambda terms M (possibly containing variables) one has

$$E^{\lceil M \rceil} = M.$$

PROOF. By an analogon to Proposition 6A.6 there exists a lambda term E such that

$$\begin{aligned} E(\underline{\text{const}}\ x) &= x; \\ E(\underline{\text{app}}\ p q) &= (Ep)(Eq); \\ E(\underline{\text{abs}}\ z) &= \lambda x.E(zx). \end{aligned}$$

Then by an easy induction one can show that $E^{\lceil M \rceil} = M$ for all terms M . ■

Following the construction of Proposition 6A.6 by Böhm, Piperno, and Guerrini [1994], this term E is given the following very simple form:

$$E \triangleq \langle \langle K, S, C \rangle \rangle,$$

where $S \triangleq \lambda xyz.xz(yz)$ and $C \triangleq \lambda xyz.x(zy)$. This is a good improvement over Kleene [1936] or B[1984]. See also Barendregt [1991], [1994], [1995] for more about self-interpreters.

Development of functional programming

In this subsection a short history is presented of how lambda calculi (untyped and typed) inspired (either consciously or unconsciously) the creation of *functional programming*.

Imperative versus functional programming

While Church had captured the notion of computability via the lambda calculus, Turing had done the same via his model of computation based on Turing machines. When in the second world war computational power was needed for military purposes, the first electronic devices were built basically as Turing machines with random access memory. Statements in the instruction set for these machines, like $x := x + 1$, are directly related to the instructions of a Turing machine. Such statements are much more easily interpreted by hardware than the act of substitution fundamental to the λ -calculus. In the beginning, the hardware of the early computers was modified each time a different computational job had to be done. Then von Neumann, who must have known¹⁸ Turing's concept of a universal Turing machine, suggested building one machine that could be programmed to do all possible computational jobs using *software*. In the resulting computer revolution, almost all machines are based on this so called von Neumann computer, consisting of a programmable universal machine. It would have been more appropriate to call it the Turing computer.

The model of computability introduced by Church (lambda definability)—although equivalent to that of Turing—was harder to interpret in hardware. Therefore the emergence of the paradigm of functional programming, that is based essentially on lambda definability, took much more time. Because functional programs are closer to the specification of computational problems than imperative ones, this paradigm is more convenient than the traditional imperative one. Another important feature of functional programs is that parallelism is much more naturally expressed in them, than in imperative programs. See [Turner \[1981\]](#) and [Hughes \[1989\]](#) for some evidence for the elegance of the functional paradigm. The implementation difficulties for functional programming have to do with memory usage, compilation time and actual run time of functional programs. In the contemporary state of the art of implementing functional languages, these problems have been solved satisfactorily.¹⁹

Classes of functional languages

Let us describe some languages that have been—and in some cases still are—influential in the expansion of functional programming. These languages come in several classes.

Lambda calculus by itself is not yet a complete model of computation, since an expression M may be evaluated by different so-called reduction strategies that indicate which sub-term of M is evaluated first (see [B\[1984\]](#), Ch. 12). By the Church-Rosser theorem this order of evaluation is not important for the final result: the normal form

¹⁸Church had invited Turing to the United States in the mid 1930's. After his first year it was von Neumann who invited Turing to stay for a second year. See [Hodges \[1983\]](#).

¹⁹Logical programming languages also have the mentioned advantages. But so far pure logical languages of industrial quality have not been developed. (Prolog is not pure and λ -Prolog, see [Nadathur and Miller \[1988\]](#), although pure, is presently a prototype.)

of a lambda term is unique if it exists. But the order of evaluation makes a difference for efficiency (both time and space) and also for the question whether or not a normal form is obtained at all.

So called ‘eager’ functional languages have a reduction strategy that evaluates an expression like FA by first evaluating F and A (in no particular order) to, say, $F' \equiv \lambda a. \dots a \dots a \dots$ and A' and then contracting $F'A'$ to $\dots A' \dots A' \dots$. This evaluation strategy has definite advantages for the efficiency of the implementation. The main reason for this is that if A is large, but its normal form A' is small, then it is advantageous both for time and space efficiency to perform the reduction in this order. Indeed, evaluating FA directly to

$$\dots A \dots A \dots$$

takes more space and if A is now evaluated twice, it also takes more time.

Eager evaluation, however, is not a normalizing reduction strategy in the sense of B[1984], CH. 12. For example, if $F \equiv \lambda x.I$ and A does not have a normal form, then evaluating FA eagerly diverges, while

$$FA \equiv (\lambda x.I)A = I,$$

if it is evaluated leftmost outermost (roughly ‘from left to right’). This kind of reduction is called ‘lazy evaluation’.

It turns out that eager languages are, nevertheless, computationally complete, as we will soon see. The implementation of these languages was the first milestone in the development of functional programming. The second milestone consisted of the efficient implementation of lazy languages.

In addition to the distinction between eager and lazy functional languages there is another one of equal importance. This is the difference between untyped and typed languages. The difference comes directly from the difference between the untyped λ -calculus and the various typed λ -calculi, see B[1984]. Typing is useful, because many programming bugs (errors) result in a typing error that can be detected automatically prior to running one’s program. On the other hand, typing is not too cumbersome, since in many cases the types need not be given explicitly. The reason for this is that, by the type reconstruction algorithm of Curry [1969] and Hindley [1969] (later rediscovered by Milner [1978]), one can automatically find the type (in a certain context) of an untyped but typable expression. Therefore, the typed versions of functional programming languages are often based on the implicitly typed lambda calculi à la Curry. Types also play an important role in making implementations of lazy languages more efficient, see below.

Besides the functional languages that will be treated below, the languages APL and FP have been important historically. The language APL, introduced in Iverson [1962], has been, and still is, relatively widespread. The language FP was designed by Backus, who gave, in his lecture (Backus [1978]) at the occasion of receiving his Turing award (for his work on imperative languages) a strong and influential plea for the use of functional languages. Both APL and FP programs consist of a set of basic functions that can be combined to define operations on data structures. The language APL has, for example, many functions for matrix operations. In both languages composition is the only way

to obtain new functions and, therefore, they are less complete than a full functional language in which user defined functions can be created. As a consequence, these two languages are essentially limited in their ease of expressing algorithms.

Eager functional languages

Let us first give the promised argument that eager functional languages are computationally complete. Every computable (recursive) function is lambda definable in the λ -calculus (see [Church \[1941\]](#) or [B\[1984\]](#), Theorem 9.2.16). In the λ -calculus a term having a normal form is strongly normalizing (see [Church and Rosser \[1936\]](#) or [B\[1984\]](#), Theorem 9.1.5). Therefore an eager evaluation strategy will find the required normal form.

The first functional language, LISP, was designed and implemented by [McCarthy, Abrahams, Edwards, Hart, and Levin \[1962\]](#). The evaluation of expressions in this language is eager. LISP had (and still has) considerable impact on the art of programming. Since it has a good programming environment, many skillful programmers were attracted to it and produced interesting programs (so called ‘artificial intelligence’). LISP is not a pure functional language for several reasons. Assignment is possible in it; there is a confusion between local and global variables²⁰ (‘dynamic binding’; some LISP users even like it); LISP uses the ‘Quote’, where $(\text{Quote } M)$ is like $\lceil M \rceil$. In later versions of LISP, Common LISP (see [Steele Jr. \[1984\]](#)) and Scheme (see [Abelson, Dybvig, Haynes, Rozas, IV, Friedman, Kohlbecker, Jr., Bartley, Halstead, \[1991\]](#)), dynamic binding is no longer present. The ‘Quote’ operator, however, is still present in these languages. Since $!a = a$ but $\lceil !a \rceil \neq \lceil a \rceil$ adding ‘Quote’ to the λ -calculus is inconsistent. As one may not reduce in LISP within the scope of a ‘Quote’, however, having a ‘Quote’ in LISP is not inconsistent. ‘Quote’ is not an available function but only a constructor. That is, if M is a well-formed expression, so is $(\text{Quote } M)$ ²¹. Also, LISP has a primitive fixed point operator ‘LABEL’ (implemented as a cycle) that is also found in later functional languages.

In the meantime, [Landin \[1964\]](#) developed an abstract machine—the SECD machine—for the implementation of reduction. Many implementations of eager functional languages, including some versions of LISP, have used, or are still using, this computational model. (The SECD machine also can be modelled for lazy functional languages, see [Henderson \[1980\]](#).) Another way of implementing functional languages is based on the

²⁰This means substitution of an expression with a free variable into a context in which that variable becomes bound. The originators of LISP were in good company: in [Hilbert and Ackermann \[1928\]](#) the same was done, as was noticed by von Neumann in his review of that book. Church may have known von Neumann’s review and avoided confusing local and global variables by introducing α -conversion.

²¹Using ‘Quote’ as a function would violate the Church-Rosser property. An example is

$$(\lambda x.x(!a)) \text{ Quote}$$

that then would reduce to both

$$\text{Quote} (!a) \rightarrow \lceil !a \rceil$$

and to

$$(\lambda x.xa) \text{ Quote} \rightarrow \text{Quote} a \rightarrow \lceil a \rceil$$

and there is no common reduct for these two expressions $\lceil !a \rceil$ and $\lceil a \rceil$.

so called CPS-translation. This was introduced in [Reynolds \[1972\]](#) and used in compilers by [Steele Jr. \[1978\]](#) and [Appel \[1992\]](#). See also [Plotkin \[1975\]](#) and [Reynolds \[1993\]](#).

The first important typed functional language with an eager evaluation strategy is Standard ML, see [Milner \[1978\]](#). This language is based on the Curry variant $\lambda_{\rightarrow}^{\text{Ch}}$, the simply typed λ -calculus with implicit typing. Expressions are type-free, but are only legal if a type can be derived for them. By the algorithm of Curry and Hindley cited above, it is decidable whether an expression does have a type and, moreover, its most general type can be computed. Milner added two features to $\lambda_{\rightarrow}^{\text{A}}$. The first is the addition of new primitives. One has the fixed point combinator Y as primitive, with essentially all types of the form $(A \rightarrow A) \rightarrow A$, with $A \equiv (B \rightarrow C)$, assigned to it. Indeed, if $f : A \rightarrow A$, then $\text{Y}f$ is of type A so that both sides of

$$f(\text{Y}f) = \text{Y}f$$

have type A . Primitives for basic arithmetic operations are also added. With these additions, ML becomes a universal programming language, while $\lambda_{\rightarrow}^{\text{A}}$ is not (since all its terms are normalizing). The second addition to ML is the ‘let’ construction

$$\underline{\text{let}} \ x \ \underline{\text{be}} \ N \ \underline{\text{in}} \ M \ \underline{\text{end}}. \quad (1)$$

This language construct has as its intended interpretation

$$M[x := N], \quad (2)$$

so that one may think that the let construction is not necessary. If, however, N is large, then this translation of (1) becomes space inefficient. Another interpretation of (1) is

$$(\lambda x.M)N. \quad (3)$$

But this interpretation has its limitations, as N has to be given one fixed type, whereas in (2) the various occurrences of N may have different types. The expression (1) is a way to make use of both the space reduction (‘sharing’) of the expression (3) and the ‘implicit polymorphism’ in which N can have more than one type of (2). An example of the let expression is

$$\underline{\text{let}} \ \text{id} \ \underline{\text{be}} \ \lambda x.x \ \underline{\text{in}} \ \lambda f x. (\text{id} f)(\text{id} x) \ \underline{\text{end}}.$$

This is typable by

$$(A \rightarrow A) \rightarrow (A \rightarrow A),$$

if the second occurrence of id gets type $(A \rightarrow A) \rightarrow (A \rightarrow A)$ and the third $(A \rightarrow A)$.

Because of its relatively efficient implementation and the possibility of type checking at compile time (for finding errors), the language ML has evolved into important industrial variants (like Standard ML of New Jersey).

Although not widely used in industry, a more efficient implementation of ML is based on the abstract machine CAML, see [Cousineau, Curien, and Mauny \[1987\]](#). CAML was inspired by the categorical foundations of the λ -calculus, see [Smyth and Plotkin \[1982\]](#), [Koymans \[1982\]](#) and [Curien \[1993\]](#). All of these papers have been inspired by the work on denotational semantics of Scott, see [Scott \[1972\]](#) and [Gunter and Scott \[1990\]](#).

Lazy functional languages

Although all computable functions can be represented in an eager functional programming language, not all reductions in the full $\lambda\mathbb{K}$ -calculus can be performed using eager evaluation. We already saw that if $F \equiv \lambda x. I$ and A does not have a normal form, then eager evaluation of FA does not terminate, while this term does have a normal form. In ‘lazy’ functional programming languages the reduction of FA to I is possible, because the reduction strategy for these languages is essentially leftmost outermost reduction which is normalizing.

One of the advantages of having lazy evaluation is that one can work with ‘infinite’ objects. For example there is a legal expression for the potentially infinite list of primes

$$[2, 3, 5, 7, 11, 13, 17 \dots],$$

of which one can take the n -th projection in order to get the n -th prime. See [Turner \[1981\]](#) and [Hughes \[1989\]](#) for interesting uses of the lazy programming style.

Above we explained why eager evaluation can be implemented more efficiently than lazy evaluation: copying large expressions is expensive because of space and time costs. In [Wadsworth \[1971\]](#) the idea of *graph reduction* was introduced in order to also do lazy evaluation efficiently. In this model of computation, an expression like $(\lambda x. \dots x \dots x \dots)A$ does not reduce to $\dots A \dots A \dots$ but to $\dots @ \dots @ \dots ; @ : A$, where the first two occurrences of $@$ are pointers referring to the A behind the third occurrence. In this way lambda expressions become dags (directed acyclic graphs).²²

Based on the idea of graph reduction, using carefully chosen combinators as primitives, the experimental language SASL, see [Turner \[1976\], \[1979\]](#), was one of the first implemented lazy functional languages. The notion of graph reduction was extended by Turner by implementing the fixed point combinator (one of the primitives) as a cyclic graph. (Cyclic graphs were already described in [Wadsworth \[1971\]](#) but were not used there.) Like LISP, the language SASL is untyped. It is fair to say that—unlike programs written in the eager languages such as LISP and Standard ML—the execution of SASL programs was orders of magnitude slower than that of imperative programs in spite of the use of graph reduction.

In the 1980s typed versions of lazy functional languages did emerge, as well as a considerable speed-up of their performance. A lazy version of ML, called Lazy ML (LML), was implemented efficiently by a group at Chalmers University, see [Johnsson \[1984\]](#). As underlying computational model they used the so called G-machine, that avoids building graphs whenever efficient. For example, if an expression is purely arithmetical (this can be seen from type information), then the evaluation can be done more efficiently than by using graphs. Another implementation feature of the LML is the compilation into super-combinators, see [Hughes \[1984\]](#), that do not form a fixed set, but are created on demand depending on the expression to be evaluated. Emerging from SASL, the first fully developed typed lazy functional language called MirandaTM was developed by

²²Robin Gandy mentioned at a meeting for the celebration of his seventieth birthday that already in the early 1950s Turing had told him that he wanted to evaluate lambda terms using graphs. In Turing’s description of the evaluation mechanism he made the common oversight of confusing free and bound variables. Gandy pointed this out to Turing, who then said: “Ah, this remark is worth 100 pounds a month!”

[Turner \[1985\]](#). Special mention should be made of its elegance and its functional I/O interface (see below).

Notably, the ideas in the G-machine made lazy functional programming much more efficient. In the late 1980s very efficient implementations of two typed lazy functional languages appeared that we will discuss below: Clean, see [van Eekelen and Plasmeijer \[1993\]](#), and Haskell, see [Peyton Jones and Wadler \[1993\]](#), [Hudak, Peyton Jones, Wadler, Boutel, Fairbairn, Fasel, Guzman, Hammond, Hughes, Johnsson, \[1992\]](#). These languages, with their implementations, execute functional programs in a way that is comparable to the speed of contemporary imperative languages such as C.

Interactive functional languages

The versions of functional programming that we have considered so far could be called ‘autistic’. A program consists of an expression M , its execution of the reduction of M and its output of the normal form M^{nf} (if it exists). Although this is quite useful for many purposes, no interaction with the outside world is made. Even just dealing with input and output (I/O) requires interaction.

We need the concept of a ‘process’ as opposed to a function. Intuitively a process is something that (in general) is geared towards continuation while a function is geared towards termination. Processes have an input channel on which an input stream (a potentially infinite sequence of tokens) is coming in and an output channel on which an output stream is coming out. A typical process is the control of a traffic light system: it is geared towards continuation, there is an input stream (coming from the push-buttons for pedestrians) and an output stream (regulating the traffic lights). Text editing is also a process. In fact, even the most simple form of I/O is already a process.

A primitive way to deal with I/O in a functional language is used in some versions of ML. There is an input stream and an output stream. Suppose one wants to perform the following process P :

read the first two numbers x, y of the input stream;
put their difference $x - y$ onto the output stream

Then one can write in ML the following program

```
write (read – read).
```

This is not very satisfactory, since it relies on a fixed order of evaluation of the expression ‘**read** – **read**’.

A more satisfactory way consists of so-called continuations, see [Gordon \[1994\]](#). To the λ -calculus one adds primitives **Read**, **Write** and **Stop**. The operational semantics of an expression is now as follows:

$$\begin{aligned}
 M &\Rightarrow M^{\text{hnf}}, & \text{where } M^{\text{hnf}} \text{ is the head normal form}^{23} \text{ of } M; \\
 \text{Read } M &\Rightarrow M a, & \text{where } a \text{ is taken off the input stream;} \\
 \text{Write } b M &\Rightarrow M, & \text{and } b \text{ is put into the output stream;} \\
 \text{Stop} &\Rightarrow & \text{i.e., do nothing.}
 \end{aligned}$$

²³A head nf in λ -calculus is of the form $\lambda \vec{x}.yM_1 \dots M_n$, with the $M_1 \dots M_n$ possibly not in nf.

Now the process P above can be written as

$$P = \text{Read}(\lambda x. \text{Read}(\lambda y. \text{Write}(x - y) \text{ Stop})).$$

If, instead, one wants a process Q that continuously takes two elements of the input stream and put the difference on the output stream, then one can write as a program the following extended lambda term

$$Q = \text{Read}(\lambda x. \text{Read}(\lambda y. \text{Write}(x - y) Q)),$$

which can be found using the fixed point combinator.

Now, every interactive program can be written in this way, provided that special commands written on the output stream are interpreted. For example one can imagine that writing

‘echo’ 7 or ‘print’ 7

on the output channel will put 7 on the screen or print it out respectively. The use of continuations is equivalent to that of monads in programming languages like Haskell, as shown in [Gordon \[1994\]](#). (The present version of Haskell I/O is more refined than this; we will not consider this issue.)

If A_0, A_1, A_2, \dots is an effective sequence of terms (i.e., $A_n = F[n]$ for some F), then this infinite list can be represented as a lambda term

$$\begin{aligned} [A_0, A_1, A_2, \dots] &\equiv [A_0, [A_1, [A_2, \dots]]] \\ &= H[0], \end{aligned}$$

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

$$H[n] = [F[n], H[n+1]].$$

This H can be defined using the fixed point combinator.

Now the operations **Read**, **Write** and **Stop** can be made explicitly lambda definable if we use

$$\begin{aligned} \text{In} &= [A_0, A_1, A_2, \dots], \\ \text{Out} &= [\dots, B_2, B_1, B_0], \end{aligned}$$

where **In** is a representation of the potentially infinite input stream given by ‘the world’ (i.e., the user and the external operating system) and **Out** of the potentially infinite output stream given by the machine running the interactive functional language. Every interactive program M should be acting on $[\text{In}, \text{Out}]$ as argument. So M in the continuation language becomes

$$M[\text{In}, \text{Out}].$$

The following definition then matches the operational semantics.

$$(1) \quad \left\{ \begin{array}{lcl} \text{Read } F [[A, \text{In}'], \text{Out}] & = & F A [\text{In}', \text{Out}]; \\ \text{Write } F B [\text{In}, \text{Out}] & = & F [\text{In}, [B, \text{Out}]] \\ \text{Stop } [\text{In}, \text{Out}] & = & [\text{In}, \text{Out}]. \end{array} \right.$$

In this way $[\text{In}, \text{Out}]$ acts as a dynamic state. An operating system should take care that the actions on $[\text{In}, \text{Out}]$ are actually performed to the I/O channels. Also we have to take

care that statements like ‘echo’ 7 are being interpreted. It is easy to find pure lambda terms `Read`, `Write` and `Stop` satisfying (1). This seems to be a good implementation of the continuations and therefore a good way to deal with interactive programs.

There is, however, a serious problem. Define

$$M \equiv \lambda p.[\text{Write } b_1 \text{ Stop } p, \text{Write } b_2 \text{ Stop } p].$$

Now consider the evaluation

$$\begin{aligned} M [\text{In}, \text{Out}] &= [\text{Write } b_1 \text{ Stop } [\text{In}, \text{Out}], \text{Write } b_2 \text{ Stop } [\text{In}, \text{Out}]] \\ &= [[\text{In}, [b_1, \text{Out}]], [\text{In}, [b_2, \text{Out}]]]. \end{aligned}$$

Now what will happen to the actual output channel: should b_1 be added to it, or perhaps b_2 ?

The dilemma is caused by the duplication of the I/O channels `[In,Out]`. One solution is not to explicitly mention the I/O channels, as in the λ -calculus with continuations. This is essentially what happens in the method of monads in the interactive functional programming language Haskell. If one writes something like

$$\text{Main } f_1 \circ \cdots \circ f_n$$

the intended interpretation is $(f_1 \circ \cdots \circ f_n)[\text{In}, \text{Out}]$.

The solution put forward in the functional language Clean is to use a typing system that guarantees that the I/O channels are never duplicated. For this purpose a so-called ‘uniqueness’ typing system is designed, see [Barendsen and Smetsers \[1993\], \[1996\]](#), that is related to linear logic (see [Girard \[1995\]](#)). Once this is done, one can improve the way in which parts of the world are used explicitly. A representation of all aspects of the world can be incorporated in λ -calculus. Instead of having just `[In,Out]`, the world can now be extended to include (a representation of) the screen, the printer, the mouse, the keyboard and whatever gadgets one would like to add to the computer periphery (e.g., other computers to form a network). So interpreting

$$\text{'print' } 7$$

now becomes simply something like

$$\text{put 7 printer.}$$

This has the advantage that if one wants to echo a 7 and to print a 3, but the order in which this happens is immaterial, then one is not forced to make an over-specification, like sending first ‘print’ 3 and then ‘echo’ 7 to the output channel:

$$[\dots, \text{'echo' } 7, \text{'print' } 3]$$

By representing inside the λ -calculus with uniqueness types as many gadgets of the world as one would like, one can write something like

$$\begin{aligned} F [\text{keyboard}, \text{mouse}, \text{screen}, \text{printer}] &= \\ &= [\text{keyboard}, \text{mouse}, \text{put 3 screen}, \text{put 7 printer}]. \end{aligned}$$

What happens first depends on the operating system and parameters, that we do not know (for example on how long the printing queue is). But we are not interested in this. The system satisfies the Church-Rosser theorem and the eventual result (7 is printed and 3 is echoed) is unambiguous. This makes Clean somewhat more natural than Haskell

(also in its present version) and definitely more appropriate for an implementation on parallel hardware.

Both Clean and Haskell are state of the art functional programming languages producing efficient code; as to compiling time Clean belongs to the class of fast compilers (including those for imperative languages). Many serious applications are written in these languages. The interactive aspect of both languages is made possible by lazy evaluation and the use of higher type²⁴ functions, two themes that are at the core of the λ -calculus (λK -calculus, that is). It is to be expected that they will have a significant impact on the production of modern (interactive window based) software.

Other aspects of functional programming

In several of the following viable applications there is a price to pay. Types can no longer be derived by the Hindley-Milner algorithm, but need to be deduced by an assignment system more complex than that of the simply typed λ -calculus $\lambda \rightarrow$.

Type classes

Certain types come with standard functions or relations. For example on the natural numbers and integers one has the successor function, the equality and the order relation. A type class is like a signature in computer science or a similarity type in logic: it states to which operations, constants, and relations the data type is coupled. In this way one can write programs not for one type but for a class of types.

If the operators on classes are not only first order but higher order, one obtains ‘type constructor classes’, that are much more powerful. See [Jones \[1993\]](#), where the idea was introduced and [Voigtländer \[2009\]](#) for recent results.

Generic programming

The idea of type classes can be pushed further. Even if data types are different, in the sense that they have different constructors, one can share code. For

$$[a_0, a_1, a_2, \dots]$$

a stream, there is the higher type function ‘ map_s ’ that acts like

$$\text{map}_s f[a_0, a_1, a_2, \dots] \rightarrow [fa_0, fa_1, fa_2, \dots].$$

But there is also a ‘ map_t ’ that distributes a function over all data present at nodes of the tree.

Generic programming makes it possible to write one program ‘ map ’ that acts both for streams and trees. What happens here is that this ‘ map ’ works on the code for data types and recognizes its structure. Then ‘ map ’ transforms itself, when requested, into the right version to do the intended work. See [Hinze, Jeuring, and Löh \[2007\]](#) for an elaboration of this idea. In [Plasmeijer, Achten, and Koopman \[2007\]](#) generic programming is exploited for efficient programming of web-interfaces for work flow systems.

²⁴In the functional programming community these are called ‘higher *order* functions’. We prefer to use the more logically correct expression ‘higher *type*’, since ‘higher order’ refers to quantification over types, like in the system $\lambda 2$ (system F) of Girard, see [Girard, Lafont, and Taylor \[1989\]](#).

Dependent types

These types come from the language Automath, see next Section, intended to express mathematical properties as a type depending on a term. This breaks the independence of types from terms, but is quite useful in proof-checking. A typical dependent type is an n -dimensional vector space F^n , that depends on the element n of another type. In functional programming dependent types have been used to be able to type more functions. See [Augustson \[1999\]](#).

Dynamic types

The underlying computational model for functional programming consists of reducing λ -terms. From the λ -calculus point of view, one can pause a reduction of a term towards some kind of normal form, in order to continue work later with the intermediate expression. In many efficient compilers of functional programming languages one does not reduce any term, but translates it into some machine code and works on it until there is (the code of) the normal form. There are no intermediate expressions, in particular the type information is lost during (partial) execution. The mechanism of ‘dynamic types’ makes it possible to store the intermediate values in such a way that a reducing computer can be switched off and work is continued the next day. Even more exciting applications of this idea to distributed or even parallel computing is to exchange partially evaluated expressions and continue the computation process elsewhere.

In applications like web-browsers one may want to ask for ‘plug-ins’, that employ functions involving types that are not yet known to the designer of the application. This becomes possible using dynamic types. See [Pil \[1999\]](#).

Generalized Algebraic Data types

These form another powerful extension of the simple types for functional languages. See [Peyton Jones, Vytiniotis, Weirich, and Washburn \[2006\]](#).

Major applications of functional programming

Among the many functional programs for an impressive range of applications, two major ones stand out. The first consists of the proof-assistants, to be discussed in the next Section. The second consists of design languages for hardware, see [Sheeran \[2005\]](#) and [Nikhil, R. S. \[2008\]](#).

6B. 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 theory λ_L and a map translating formulas A of L into types $[A]$ of λ_L such that

$$\vdash_L A \Leftrightarrow \Gamma_A \vdash_{\lambda_L} M : [A], \text{ for some } M,$$

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

$$\vdash_L A, \text{ with proof } D \Leftrightarrow \Gamma_A \vdash_{\lambda_L} [D] : [A], \quad (1)$$

where the map $[]$ is extended to cover also derivations. For deductions from a set of assumptions one has

$$\Delta \vdash_L A, \text{ with proof } D \Leftrightarrow \Gamma_A, [\Delta] \vdash_{\lambda_L} [D] : [A].$$

Curry did not observe the correspondence in this precise form. He noted that inhabited types in λ_{\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 (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 (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 (1) was efficiently verifiable 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 modern versions of proof-assistants family, like Mizar, COQ ([Bertot and Castéran \[2004\]](#)), HOL, and Isabelle ([Nipkow, Paulson, and Wenzel \[2002b\]](#)), 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.²⁵ In Kahn [1995] methods are developed in order to translate proof objects automatically into natural language. It is hoped that

²⁵For example, proving Gödel’s incompleteness theorem contains the following technical point. 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 involved 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. The incompleteness of this stronger theory is stronger than that of arithmetic. But then the usually resulting *essential* incompleteness result states incompleteness for all extensions of an arithmetical theory with inductive types, which is a weaker result than the essential incompleteness of just arithmetic.

in the near future new proof checkers will emerge in which formalizing is not much more difficult than, say, writing an article in TeX.

Computer Mathematics

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 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.²⁶ All the systems of CM have been inspired by the Automath project of

²⁶This way of doing mathematics, the axiomatic method, was also described by Aristotle. It was Euclid of Alexandria [300] who first used this method very successfully in his Elements.

de Bruijn (see [de Bruijn \[1970\]](#), [\[1994\]](#) and [Nederpelt, Geuvers, and de Vrijer \[1994\]](#)) 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 $A_0, A_1 \dots, A_n$ such that $A = A_n$ 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 [Barendregt \[1992\]](#), 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 Isabelle/HOL, Nipkow, Paulson, and Wenzel [2002b], HOL-light and Coq, (see Bertot and Castéran [2004]), all 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 criterion²⁷.

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 van Benthem 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.

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) 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 \rightarrow_ℓ for primitive recursion on the natural numbers. Operations like $f(n) = [\sqrt{n}]$ are primitive recursive and hence are lambda definable (using $\rightarrow_{\beta\ell}$) by a term, say F , in the

²⁷The reductions may sometimes cause the proof-checking to be of an unacceptable time complexity. We have that p is a proof of A iff $\text{type}(p) =_{\beta\delta} A$. 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.

lambda calculus extended by an operation for primitive recursion R satisfying

$$\begin{aligned} R A B \text{zero} &\rightarrow_{\iota} A \\ R A B (\text{succ } x) &\rightarrow_{\iota} B x (R A B x). \end{aligned}$$

Then, writing $\lceil 0 \rceil = \text{zero}$, $\lceil 1 \rceil = \text{succ zero}$, \dots , as

$$\lceil 6 \rceil = \lceil 6 \rceil$$

is formally derivable, it follows from the Poincaré principle that the same is true for

$$F \lceil 45 \rceil = \lceil 6 \rceil$$

(with the same proof-object), since $F \lceil 45 \rceil \twoheadrightarrow_{\beta\iota} \lceil 6 \rceil$. 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

$$F \lceil n \rceil \twoheadrightarrow_{\beta\iota} \lceil f(n) \rceil$$

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_{Prime} for the characteristic function of the predicate `Prime`. This term should satisfy the following statement

$$\begin{aligned} \forall n [(\text{Prime } n \leftrightarrow K_{\text{Prime}} n = \lceil 1 \rceil) \& \\ (K_{\text{Prime}} n = \lceil 0 \rceil \vee K_{\text{Prime}} n = \lceil 1 \rceil)]. \end{aligned}$$

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 $x + 1 : \text{Nat}$ (in context $x:\text{Nat}$), the expression on a syntactic level represented internally satisfies ' $x + 1$ ' : `term(Nat)`, for the suitably defined inductive type `term(Nat)`. After introducing a reduction relation $\twoheadrightarrow_{\iota}$ for primitive recursion over this data type, one can use techniques similar to those of Section 6A to lambda define g , say by G , so that

$$G ' (x + 1)(x + 1) ' \twoheadrightarrow_{\beta\iota} 'x^2 + 2x + 1'.$$

Now in order to finish the proof of (3), one needs to construct a self-interpreter E , such that for all expressions $p : \text{Nat}$ one has

$$\mathsf{E} 'p' \twoheadrightarrow_{\beta\iota} p$$

and prove the proof obligation for G which is

$$\forall t:\text{term}(\text{Nat}) \mathsf{E}(G t) = \mathsf{E} t.$$

It follows that

$$\mathsf{E}(G ' (x + 1)(x + 1) ') = \mathsf{E} ' (x + 1)(x + 1) ';$$

now since

$$\begin{aligned} \mathsf{E}(G\,(x+1)(x+1)) &\rightarrow_{\beta\iota} \mathsf{E}\,x^2 + 2x + 1 \\ &\rightarrow_{\beta\iota} x^2 + 2x + 1 \\ \mathsf{E}\,(x+1)(x+1) &\rightarrow_{\beta\iota} (x+1)(x+1), \end{aligned}$$

we have by the Poincaré principle

$$(x+1)(x+1) = x^2 + 2x + 1.$$

The use of inductive types like `Nat` and `term(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 $P = \text{Prime}$ 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.

Foundations for existing proof-assistants

Early indications of the possibility to relate logic and types are [Church \[1940\]](#) and a remark in [Curry and Feys \[1958\]](#). The former is worked out in [Andrews \[2002\]](#). The latter has lead to the Curry-Howard correspondence between formulas and types ([Howard \[1980\]](#) written in 1969, [Martin-Löf \[1984\]](#), [Barendregt \[1992\]](#), [de Groote \[1995\]](#), and [Sørensen and Urzyczyn \[2006\]](#)).

Higher order logic as foundations has given rise to the mathematical assistants HOL ([Gordon and Melham \[1993\]](#), [</hol.sourceforge.net>](http://hol.sourceforge.net)), HOL Light ([Harrison \[2009a\]](#), www.cl.cam.ac.uk/~jrh13/hol-light/), and Isabelle²⁸, ([Nipkow, Paulson, and Wenzel \[2002a\]](#), www.cl.cam.ac.uk/research/hvg/isabelle). The type theory as foundations gave rise to the systems Coq (based on constructive logic, but with the possibility of impredicativity; [Bertot and Castéran \[2004\]](#), [<coq.inria.fr>](http://coq.inria.fr)) and Agda (based on Martin-Löf's type theory: intuitionistic and predicative; [Bove, Dybjer, and Norell \[2009\]](#)). We also mention the proof assistant Mizar ([Muzalewski \[1993\]](#), [<mizar.org>](http://mizar.org)) that is based on an extension of ZFC set theory. On the other end of the spectrum there is ACL₂ ([Kaufmann, Manolios, and Moore \[2000\]](#)), that is based on primitive recursive arithmetic.

²⁸Isabelle is actually a ‘[logical framework](#)’ in which a proof assistant proper can be defined. The main version is Isabelle/HOL, which representing higher order logic.

All these systems give (usually interactive) support for the fully formal proof of a mathematical theorem, derived from user specified axioms. For an insightful comparison of these and many more existing proof assistants see [Wiedijk \[2006\]](#), in which the irrationality of $\sqrt{2}$ has been formalized using seventeen different assistants.

Highlights

By the end of the twentieth century the technology of formalizing mathematical proofs was there, but impressive examples were missing. The situation changed dramatically during the first decade of the twenty-first century. The full formalization and computer verification of the Four Color Theorem in was achieved in Coq by [Gonthier \[2008\]](#) (formalizing the proof in [Robertson, Sanders, Seymour, and Thomas \[1997\]](#)); the Prime Number Theorem in Isabelle by [Avigad, Donnelly, Gray, and Raff \[2007\]](#) (elementary proof by Selberg) and in HOL Light by [Harrison \[2009b\]](#) (the classical proof by Hadamard and de la Vallée Poussin using complex function theory). Building upon the formalization of the Four Color Theorem the Jordan Curve Theorem has been formalized by Tom Hales, who did this as one of the ingredients needed for the full formalization of his proof of the Kepler Conjecture, [Hales \[2005\]](#).

Certifying software, and hardware

This development of high quality mathematical proof assistants was accelerated by the industrial need for reliable software and hardware. The method to certify industrial products is to fully formalize both their specification and their design and then to provide a proof that the design meets the specification²⁹. This reliance on so called ‘Formal Methods’ had been proposed since the 1970s, but lacked to be convincing. Proofs of correctness were much more complex than the mere correctness itself. So if a human had to judge the long proofs of certification, then nothing was gained. The situation changed dramatically after the proof assistants came of age. The ARM6 processor—predecessor of the ARM7 embedded in the large majority of mobile phones, personal organizers and MP3 players—was certified, [Fox \[2003\]](#), by mentioned method. The seL4 operating system has been fully specified and certified, [Klein, Elphinstone, Heiser, Andronick, Cock, Derrin, Elkaduwe, Engelhardt, Kolanski, Norrish, \[2009\]](#). The same holds for a realistic kernel of an optimizing compiler for the C programming language, [Leroy \[2009\]](#).

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; the expression ‘illative’ comes from ‘*illatum*’ past

²⁹This presupposes that the distance between the desired behaviour and the specification on the one hand, and that of the disign and realization on the other is short enough to be bridged properly.

participle of 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 PTS³⁰, 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 is as follows. The interpretation of predicate logic in ILC is such that

$$\begin{aligned}\vdash_{\text{logic}} A \text{ with proof } p &\Leftrightarrow \forall r \vdash_{\text{ILC}} [A]_r[p] \\ &\Leftrightarrow \vdash_{\text{ILC}} [A]_I[p] \\ &\Leftrightarrow \vdash_{\text{ILC}} [A]_K[p] = K[A]'_I[p] = [A]'_I,\end{aligned}$$

where r ranges over untyped lambda terms. Now if $r = I$, then this translation is the propositions-as-types interpretation; if, on the other hand, one has $r = K$, then the interpretation becomes an isomorphic version of first order logic denoted by $[A]'_I$. See [Barendregt, Bunder, and Dekkers \[1993\]](#) and [Dekkers, Bunder, and Barendregt \[1998\]](#) for these results. A short introduction to ILC (in its combinatory version) can be found in [B\[1984\]](#), Appendix B.

6C. Proof theory

Lambda terms for natural deduction, sequent calculus and cut elimination

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. 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 typable lambda terms in normal form. The cut elimination theorem becomes a simple consequence of the fact that typed lambda terms possess a normal form. This Section is based on [Barendregt and Ghilezan \[2000\]](#).

³⁰For 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.

Introduction

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. [In this section the mismatch will not be avoided, and we obtain](#) 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.

[\[Added in print.\] The relation between natural deduction and sequent calculus formulations of intuitionistic logic has been explored in several ways in Espírito Santo \[2000\], von Plato \[2001a\], von Plato \[2001b\], and Joachimski and Matthes \[2003\]. Several sequent lambda calculi Espírito Santo \[2007\], Espírito Santo, Ghilezan, and Ivetić \[2008\] have been developed for encoding proofs in sequent intuitionistic logic and addressing normalisation and cut-elimination proofs. In von Plato \[2008\] an unpublished manuscript of Gentzen is described, showing that Gentzen knew reduction and normalisation for natural deduction derivations. The manuscript is published as Gentzen \[2008\]. Finally, there is a vivid line of investigation on the computational interpretations of classical logic. We will not discuss these in this section.](#)

Next to the well-known system λ_{\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 Section is a subsystem of one in [Barbanera, Dezani-Ciancaglini, and de'Liguoro \[1995\]](#) and also implicitly present in [Mints \[1996\]](#).

For simplicity the results are presented only for the essential kernel of intuitionistic propositional logic, i.e. for the minimal implicational fragment. The method probably can be extended to the full first-order intuitionistic logic, using the terms as in [Mints \[1996\]](#).

The logical systems N , L and L^{cf}

6C.1. DEFINITION. The set **form** of formulas (of minimal implicational propositional logic) is defined by the following simplified syntax.

form ::= atom form → form
atom ::= p atom'

Note that the set of formulas is $\mathbb{T}^{\mathbb{A}}$ with $\mathbb{A} = \{p, p', p'', \dots\}$, i.e. a notational variant of \mathbb{T}^{∞} . The intention is *a priori* different: the formulas are intended to denote propositions, with the \rightarrow -operation denoting implication; the types denote collections of lambda terms, with the \rightarrow denoting the functionality of these.

We write p, q, r, \dots for arbitrary atoms and A, B, C, \dots for arbitrary formulas. Sets of formulas are denoted by Γ, Δ, \dots . The set Γ, A stands for $\Gamma \cup \{A\}$. [Because of the use of sets for](#)

assumptions in derivability, the structural rules are only implicitly present. In particular $\Gamma, A \vdash A$ covers weakening and $\Gamma \vdash A, \Gamma, B \vdash C \Rightarrow \Gamma, A \rightarrow B \vdash C$ contraction.

6C.2. DEFINITION. (i) A formula A is *derivable* in the system N from the set Γ ³¹, notation $\Gamma \vdash_N A$, if $\Gamma \vdash A$ can be generated by the following axiom and rules.

N
$\frac{A \in \Gamma}{\Gamma \vdash A}$ axiom
$\frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \rightarrow \text{elim}$
$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \rightarrow \text{intr}$

(ii) A formula A is *derivable* from a set of 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
$\frac{A \in \Gamma}{\Gamma \vdash A}$ axiom
$\frac{\Gamma \vdash A \quad \Gamma, B \vdash C}{\Gamma, A \rightarrow B \vdash C} \rightarrow \text{left}$
$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \rightarrow \text{right}$
$\frac{\Gamma \vdash A \quad \Gamma, A \vdash B}{\Gamma \vdash B} \text{cut}$

(iii) The system L^{cf} is obtained from the system L by omitting the rule (cut).

L^{cf}
$\frac{A \in \Gamma}{\Gamma \vdash A}$ axiom
$\frac{\Gamma \vdash A \quad \Gamma, B \vdash C}{\Gamma, A \rightarrow B \vdash C} \rightarrow \text{left}$
$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \rightarrow \text{right}$

6C.3. LEMMA. Suppose $\Gamma \subseteq \Gamma'$. Then

$$\Gamma \vdash A \Rightarrow \Gamma' \vdash A$$

³¹By contrast to the situation for bases, Definition 1A.14(iii), the set Γ is arbitrary

in all systems.

PROOF. By a trivial induction on derivations. ■

6C.4. PROPOSITION. *For all Γ and A we have*

$$\Gamma \vdash_N A \Leftrightarrow \Gamma \vdash_L A.$$

PROOF. (\Rightarrow) By induction on derivations in N . For the rule (\rightarrow elim) we need the rule (cut).

$$\frac{\Gamma \vdash_L A \quad \frac{\Gamma, B \vdash_L B}{\Gamma, A \rightarrow B \vdash_L B} \text{ (axiom)}}{\Gamma \vdash_L B} \text{ (cut)}$$

(\Leftarrow) By induction on derivations in L . The rule (\rightarrow left) is treated as follows.

$$\frac{\frac{\Gamma \vdash_N A}{\Gamma, A \rightarrow B \vdash_N A} \text{ (6C.3)} \quad \frac{\Gamma, A \rightarrow B \vdash_N A \rightarrow B}{\Gamma, A \rightarrow B \vdash_N B} \text{ (axiom)}}{\Gamma, A \rightarrow B \vdash_N C} \frac{\Gamma, B \vdash_N C}{\Gamma \vdash_N B \rightarrow C} \text{ (}\rightarrow\text{ intr)} \quad \frac{\Gamma \vdash_N B \rightarrow C}{\Gamma, A \rightarrow B \vdash_N C} \text{ (}\rightarrow\text{ elim)}$$

The rule (cut) is treated as follows.

$$\frac{\Gamma \vdash_N A \quad \frac{\Gamma, A \vdash_N B}{\Gamma \vdash_N A \rightarrow B} \text{ (}\rightarrow\text{ intr)}}{\Gamma \vdash_N B} \text{ (}\rightarrow\text{ elim). ■}$$

6C.5. DEFINITION. Consider the following rule as alternative to the rule (cut).

$$\frac{\Gamma, A \rightarrow A \vdash B}{\Gamma \vdash B} \text{ (cut')}$$

The system L' is defined by replacing the rule (cut) by (cut').

6C.6. PROPOSITION. *For all Γ and A*

$$\Gamma \vdash_L A \Leftrightarrow \Gamma \vdash_{L'} A.$$

PROOF. (\Rightarrow) The rule (cut) is treated as follows.

$$\frac{\Gamma \vdash_{L'} A \quad \frac{\Gamma, A \vdash_{L'} B}{\Gamma, A \rightarrow A \vdash_{L'} B} \text{ (}\rightarrow\text{ left)}}{\Gamma \vdash_{L'} B} \text{ (cut')}$$

(\Leftarrow) The rule (cut') is treated as follows.

$$\frac{\frac{\Gamma, A \vdash_L A}{\Gamma, A \rightarrow A \vdash_L B} \text{ (axiom)}}{\Gamma \vdash_L B} \frac{\Gamma \vdash_L A \rightarrow A}{\Gamma \vdash_L B} \text{ (}\rightarrow\text{ right)} \quad \frac{\Gamma \vdash_L B}{\Gamma \vdash_L B} \text{ (cut). ■}$$

Note that we have not yet investigated the role of L^{cf} .

The type assignment systems λN , λL and λL^{cf}

6C.7. DEFINITION. (i) A *type assignment* is an expression of the form

$$P : A,$$

where $P \in L$ is an untyped 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 .

In the following definition, the system λ_{\rightarrow} over \mathbb{T}^∞ is called λN . The formulas of N are isomorphic to types in \mathbb{T}^∞ and the derivations in N of a formula A are isomorphic to the closed terms M of A considered as type. If the derivation is from a set of assumptions $\Gamma = \{A_1, \dots, A_n\}$, then the derivation corresponds to an open term M under the basis $\{x_1:A_1, \dots, x_n:A_n\}$. This correspondence is called the *Curry-Howard isomorphism* or the *formulas-as-types—terms-as-proofs* interpretation. One can consider a proposition as the type of its proofs. Under this correspondence the collection of proofs of $A \rightarrow B$ consists of functions mapping the collection of proofs of A into those of B . See [Howard \[1980\]](#), [Martin-Löf \[1984\]](#), [de Groote \[1995\]](#), and [Sørensen and Urzyczyn \[2006\]](#) and the references therein for more on this topic.

6C.8. DEFINITION. (i) A type assignment $P : A$ is *derivable* from the context Γ in the system λN , notation

$$\Gamma \vdash_{\lambda N} P : A,$$

if $\Gamma \vdash P : A$ can be generated by the following axiom and rules.

λN	
$(x:A) \in \Gamma$	axiom
$\frac{}{\Gamma \vdash x : A}$	
$\frac{\Gamma \vdash P : (A \rightarrow B) \quad \Gamma \vdash Q : A}{\Gamma \vdash (PQ) : B}$	\rightarrow elim
$\frac{\Gamma, x:A \vdash P : B}{\Gamma \vdash (\lambda x.P) : (A \rightarrow B)}$	\rightarrow intr

(ii) A type assignment $P : A$ is *derivable* from 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
$\frac{(x:A) \in \Gamma}{\Gamma \vdash x:A}$
$\frac{\Gamma \vdash Q : A \quad \Gamma, x:B \vdash P : C}{\Gamma, y : A \rightarrow B \vdash P[x:=yQ] : C} \rightarrow \text{left}$
$\frac{\Gamma, x:A \vdash P : B}{\Gamma \vdash (\lambda x.P) : (A \rightarrow B)} \rightarrow \text{right}$
$\frac{\Gamma \vdash Q : A \quad \Gamma, x:A \vdash P : B}{\Gamma \vdash P[x:=Q] : B} \text{ cut}$

In the rule (\rightarrow left) it is required that $\Gamma, y:A \rightarrow B$ is a context. This is the case if y is fresh or if $\Gamma = \Gamma, y:A \rightarrow B$, i.e. $y:A \rightarrow B$ already occurs in Γ .

(iii) The system λL^{cf} is obtained from the system λL by omitting the rule (cut).

λL^{cf}
$\frac{(x:A) \in \Gamma}{\Gamma \vdash x:A}$
$\frac{\Gamma \vdash Q : A \quad \Gamma, x:B \vdash P : C}{\Gamma, y : A \rightarrow B \vdash P[x:=yQ] : C} \rightarrow \text{left}$
$\frac{\Gamma, x:A \vdash P : B}{\Gamma \vdash (\lambda x.P) : (A \rightarrow B)} \rightarrow \text{right}$

6C.9. 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 $\lambda L'$
$\frac{\Gamma, x:A \rightarrow A \vdash P : B}{\Gamma \vdash P[x:=l] : B} \text{ cut'}$

NOTATION. Let $\Gamma = \{A_1, \dots, A_n\}$ and $\vec{x} = \{x_1, \dots, x_n\}$. Write

$$\Gamma_{\vec{x}} = \{x_1:A_1, \dots, x_n:A_n\}$$

and

$$\Lambda^\circ(\vec{x}) = \{P \in \text{term} \mid FV(P) \subseteq \vec{x}\},$$

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 .)

6C.10. 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*

$$\Gamma \vdash_S A \Leftrightarrow \exists \vec{x} \exists P \in \Lambda^\circ(\vec{x}) \quad \Gamma_{\vec{x}} \vdash_{\lambda S} P : A.$$

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 λ_\rightarrow , the simply typed lambda calculus, we know the following results from previous Chapters: Theorem 2B.1 and Propositions 1B.6 and 1B.3. From corollary 6C.14 it follows that the results also hold for λL .

6C.11. PROPOSITION. (i) (Normalization theorem for λN).

$$\Gamma \vdash_{\lambda N} P : A \Rightarrow P \text{ is strongly normalizing.}$$

(ii) (Subject reduction theorem for λN).

$$\Gamma \vdash_{\lambda N} P : A \& P \rightarrow_\beta P' \Rightarrow \Gamma \vdash_{\lambda N} P' : A.$$

(iii) (Inversion Lemma for λN). *Type assignment for terms of a certain syntactic form can only be 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) $\Gamma \vdash_{\lambda N} \lambda x.P : C \Rightarrow \Gamma, x:A \vdash_{\lambda N} P : B \& C \equiv A \rightarrow B,$
for some types A, B .

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 .

6C.12. PROPOSITION. $\Gamma \vdash_{\lambda N} P : A \Rightarrow \Gamma \vdash_{\lambda L} P : A.$

PROOF. By inductions on derivations in λN . Modus ponens (\rightarrow elim) is treated as follows.

$$\frac{\Gamma \vdash_{\lambda L} Q : A \quad \Gamma, x:B \vdash_{\lambda L} x:B}{\Gamma, y:A \rightarrow B \vdash_{\lambda L} yQ : B} (\rightarrow \text{left})$$

$$\frac{\Gamma \vdash_{\lambda L} P : A \rightarrow B \quad \Gamma, y:A \rightarrow B \vdash_{\lambda L} yQ : B}{\Gamma \vdash_{\lambda L} PQ : B} (\text{cut}). \blacksquare$$

6C.13. PROPOSITION. (i) $\Gamma \vdash_{\lambda L} P : A \Rightarrow \Gamma \vdash_{\lambda N} P' : A$, for some $P' \rightarrow_\beta P$.

(ii) $\Gamma \vdash_{\lambda L} P : A \Rightarrow \Gamma \vdash_{\lambda N} P : A.$

PROOF. (i) By induction on derivations in λL . The rule (\rightarrow left) is treated as follows (the justifications are left out, but they are as in the proof of 6C.4).

$$\frac{\Gamma \vdash_{\lambda N} Q : A}{\Gamma, y:A \rightarrow B \vdash_{\lambda N} Q : A} \quad \frac{\Gamma, y:A \rightarrow B \vdash_{\lambda N} y:A \rightarrow B}{\Gamma, y:A \rightarrow B \vdash_{\lambda N} yQ : B} \quad \frac{\Gamma, x:B \vdash_{\lambda N} P : C}{\Gamma \vdash_{\lambda N} (\lambda x.P) : B \rightarrow C}$$

$$\frac{}{\Gamma, y:A \rightarrow B \vdash_{\lambda N} (\lambda x.P)(yQ) : C}$$

Now $(\lambda x.P)(yQ) \rightarrow_{\beta} P[x:=yQ]$ as required. The rule (cut) is treated as follows.

$$\frac{\Gamma, x:A \vdash_{\lambda N} P : B \quad \Gamma \vdash_{\lambda N} Q : A \quad \Gamma \vdash_{\lambda N} (\lambda x.P) : A \rightarrow B}{\Gamma \vdash_{\lambda N} (\lambda x.P)Q : B} \begin{array}{l} (\rightarrow \text{intr}) \\ (\rightarrow \text{elim}) \end{array}$$

Now $(\lambda x.P)Q \rightarrow_{\beta} P[x:=Q]$ as required.

(ii) By (i) and the subject reduction theorem for λN (6C.11(ii)). ■

6C.14. COROLLARY. $\Gamma \vdash_{\lambda L} P : A \Leftrightarrow \Gamma \vdash_{\lambda N} P : A$.

PROOF. By Propositions 6C.12 and 6C.13(ii). ■

Now we will investigate the role of the cut-free system.

6C.15. PROPOSITION.

$$\Gamma \vdash_{\lambda L^{\text{cf}}} P : A \Rightarrow P \text{ is in } \beta\text{-nf.}$$

PROOF. By an easy induction on derivations. ■

6C.16. LEMMA. *Suppose*

$$\Gamma \vdash_{\lambda L^{\text{cf}}} P_1 : A_1, \dots, \Gamma \vdash_{\lambda L^{\text{cf}}} P_n : A_n.$$

Then

$$\Gamma, x:A_1 \rightarrow \dots \rightarrow A_n \rightarrow B \vdash_{\lambda L^{\text{cf}}} xP_1 \dots P_n : B$$

for those variables x such that $\Gamma, x:A_1 \rightarrow \dots \rightarrow A_n \rightarrow B$ is a context.

PROOF. We treat the case $n = 2$, which is perfectly general. We abbreviate $\vdash_{\lambda L^{\text{cf}}}$ as \vdash .

$$\frac{\Gamma \vdash P_1 : A_1 \quad \frac{\Gamma \vdash P_2 : A_2 \quad \frac{\Gamma, z:B \vdash z : B}{\Gamma, y:A_2 \rightarrow B \vdash yP_2 \equiv z[z:=yP_2] : B} \text{ (axiom)}}{\Gamma, x:A_1 \rightarrow A_2 \rightarrow B \vdash xP_1 P_2 \equiv (yP_2)[y:=xP_1] : B} \text{ (}\rightarrow\text{ left)} \quad \text{ (}\rightarrow\text{ left)}$$

Note that x may occur in some of the P_i . ■

6C.17. PROPOSITION. *Suppose that P is a β -nf. Then*

$$\Gamma \vdash_{\lambda N} P : A \Rightarrow \Gamma \vdash_{\lambda L^{\text{cf}}} P : A.$$

PROOF. By induction on the following generation of normal forms.

$$\text{nf} = \text{var nf}^* \mid \lambda \text{var.nf}$$

Here var nf^* stands for var followed by 0 or more occurrences of nf . The case $P \equiv \lambda x.P_1$ is easy. The case $P \equiv xP_1 \dots P_n$ follows from the previous lemma, using the generation lemma for λN , Proposition 6C.11(iii). ■

Now we get as bonus the *Hauptsatz* of [Gentzen \[1936\]](#) for minimal implicational sequent calculus.

6C.18. THEOREM (Cut elimination).

$$\Gamma \vdash_L A \Rightarrow \Gamma \vdash_{L^{\text{cf}}} A.$$

PROOF. $\Gamma \vdash_L A \Rightarrow \Gamma_{\vec{x}} \vdash_{\lambda L} P : A$, for some $P \in \Lambda^\circ(\vec{x})$, by 6C.10,
 $\Rightarrow \Gamma_{\vec{x}} \vdash_{\lambda N} P : A$, by 6C.13(ii),
 $\Rightarrow \Gamma_{\vec{x}} \vdash_{\lambda N} P^{\text{nf}} : A$, by 6C.11(i),(ii),
 $\Rightarrow \Gamma_{\vec{x}} \vdash_{\lambda L^{\text{cf}}} P^{\text{nf}} : A$, by 6C.17,
 $\Rightarrow \Gamma \vdash_{L^{\text{cf}}} A$, by 6C.10. ■

As it is clear that the proof implies that cut-elimination can be used to normalize terms typable in $\lambda N = \lambda \rightarrow$, [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 class citizens, whereas in λL the provable formulas and the lambda terms are.

In λN typable terms are built up as usual (following the grammar of lambda terms). In λL^{cf} only normal terms are typable. 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 $C \rightarrow B$ (in the context $z:A, y:A \rightarrow B$) via two different cut-free derivations:

$$\frac{x:C, z:A \vdash z : A \quad x:C, z:A, u:B \vdash u : B}{x:C, z:A, y:A \rightarrow B \vdash yz : B} (\rightarrow \text{left})$$

$$\frac{x:C, z:A, y:A \rightarrow B \vdash yz : B}{z:A, y:A \rightarrow B \vdash \lambda x.yz : C \rightarrow B} (\rightarrow \text{right})$$

and

$$\frac{z:A \vdash z:A \quad \frac{x:C, z:A, u:B \vdash u : B}{z:A, u:B \vdash \lambda x.u : C \rightarrow B} (\rightarrow \text{right})}{z:A, y:A \rightarrow B \vdash \lambda x.yz : C \rightarrow B} (\rightarrow \text{left})$$

These correspond, respectively, to the following two formations of terms

$$\begin{array}{rcl} u & \longmapsto & yz \quad \longmapsto \lambda x.yz, \\ u & \longmapsto & \lambda x.u \quad \longmapsto \lambda x.yz. \end{array}$$

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 \[1999\]](#), [Schwichtenberg \[1999\]](#) and [Troelstra \[1999\]](#).

In [Herbelin \[1995\]](#) the mismatch between L-derivations and lambda terms is repaired by translating these into terms with explicit substitution:

$$\begin{aligned} \lambda x.(u < u:=yz >), \\ (\lambda x.u) < u:=yz >. \end{aligned}$$

In this Section 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

$$P \longmapsto P[x:=!].$$

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 $!$.

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.

6D. 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 approach 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 book, 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 $B \rightarrow A$. 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:

$$A \rightarrow C/B \Leftrightarrow A \bullet B \rightarrow C \Leftrightarrow B \rightarrow A \setminus C \quad (\text{Res})$$

In fact, we have the *pure* logic of residuation here: (Res), together with Reflexivity ($A \rightarrow A$) and Transitivity (from $A \rightarrow B$ and $B \rightarrow C$, conclude $A \rightarrow C$), fully characterizes the derivability relation, as the following completeness result shows.

COMPLETENESS $A \rightarrow B$ 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 $np \setminus s$, transitive $(np \setminus s)/np$, ditransitive $((np \setminus s)/np)/np$, 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 $s/(np \setminus s)$, direct object $((np \setminus s)/np) \setminus (np \setminus s)$, prepositional object $(pp/np) \setminus pp$, 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 $A/(C/B)$, $A/(B \setminus C)$, 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 \[1958\]](#), [Lambek \[1961\]](#) proves that Cut is a redundant rule in this presentation. Top-down backward-chaining proof search in the cut-free system respects the subformula property and yields a decision procedure.

$$\begin{array}{c}
 \frac{}{A \Rightarrow A} \text{Ax} \quad \frac{\Delta \Rightarrow A \quad \Gamma[A] \Rightarrow B}{\Gamma[\Delta] \Rightarrow B} \text{Cut} \\
 \frac{\Gamma \Rightarrow A \quad \Delta \Rightarrow B}{(\Gamma, \Delta) \Rightarrow A \bullet B} (\bullet R) \quad \frac{\Gamma[(A, B)] \Rightarrow C}{\Gamma[A \bullet B] \Rightarrow C} (\bullet L) \\
 \frac{\Delta \Rightarrow B \quad \Gamma[A] \Rightarrow C}{\Gamma[(\Delta, B \setminus A)] \Rightarrow C} (\setminus L) \quad \frac{(B, \Gamma) \Rightarrow A}{\Gamma \Rightarrow B \setminus A} (\setminus R) \\
 \frac{\Delta \Rightarrow B \quad \Gamma[A] \Rightarrow C}{\Gamma[(A/B, \Delta)] \Rightarrow C} (/L) \quad \frac{(\Gamma, B) \Rightarrow A}{\Gamma \Rightarrow A/B} (/R)
 \end{array}$$

To specify a grammar for a particular language it is enough now to give its lexicon. $\text{Lex} \subseteq \Sigma \times \mathcal{F}$ 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 , $w_1 \cdots w_n \in L(\text{Lex}, B)$ iff for $1 \leq i \leq n$, $(w_i, A_i) \in \text{Lex}$, and $\Gamma \Rightarrow B$ where Γ is a tree with ‘yield’ at its endpoints A_1, \dots, A_n . [Buszkowski and Penn \[1990\]](#) model the acquisition of lexical type assignments as a process of solving type equations. Their unification-based algorithms take function-argument structures as input (binary trees with a distinguished 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 model-theoretic 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 map 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. $D_{A \rightarrow B}$, the semantic domain for a functional

type $A \rightarrow B$, is the set of functions from D_A to D_B . The map from syntactic to semantic types $(\cdot)'$ could now stipulate for basic syntactic types that $np' = e$, $s' = t$, and $n' = e \rightarrow t$. 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 $(A/B)' = (B \setminus A)' = B' \rightarrow A'$. 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	$(s/(np \setminus s))/n$	$(e \rightarrow t) \rightarrow (e \rightarrow t) \rightarrow t$
intransitive verb	$np \setminus s$	$e \rightarrow t$
transitive verb	$(np \setminus s)/np$	$e \rightarrow e \rightarrow t$
reflexive pronoun	$((np \setminus s)/np) \setminus (np \setminus s)$	$(e \rightarrow e \rightarrow t) \rightarrow e \rightarrow t$
relative pronoun	$(n \setminus n)/(np \setminus s)$	$(e \rightarrow t) \rightarrow (e \rightarrow t) \rightarrow e \rightarrow t$

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 $x_1 : A_1, \dots, x_n : A_n$. 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, \dots, x_n , representing a program which, given inputs $k_1 \in D_{A'_1}, \dots, k_n \in D_{A'_n}$, 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 $(A/B)' = (B \setminus A)'$. 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.

$$\frac{(\Gamma, x : B) \vdash M : A}{\Gamma \vdash \lambda x. M : A/B} I/ \quad \frac{(x : B, \Gamma) \vdash M : A}{\Gamma \vdash \lambda x. M : B \setminus A} I\setminus$$

$$\frac{\Gamma \vdash M : A/B \quad \Delta \vdash N : B}{(\Gamma, \Delta) \vdash MN : A} E/ \quad \frac{\Gamma \vdash N : B \quad \Delta \vdash M : B \setminus A}{(\Gamma, \Delta) \vdash MN : A} E\setminus$$

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.

$$\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B} C \quad \frac{\Gamma \vdash B}{\Gamma, A \vdash B} W$$

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 absence 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 labor 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 ($e \rightarrow t$) arguments (noun and verb phrase), and to test the intersection for non-emptiness in the case of 'some'.

a, some (determiner)	$(e \rightarrow t) \rightarrow (e \rightarrow t) \rightarrow t$	$\lambda P \lambda Q. (\exists \lambda x. ((P x) \wedge (Q x)))$
himself (reflexive pronoun)	$(e \rightarrow e \rightarrow t) \rightarrow e \rightarrow t$	$\lambda R \lambda x. ((R x) x)$
that (relative pronoun)	$(e \rightarrow t) \rightarrow (e \rightarrow t) \rightarrow e \rightarrow t$	$\lambda P \lambda Q \lambda x. ((P x) \wedge (Q x)))$

The interplay between lexical and derivational aspects of meaning assembly is illustrated with the natural deduction below. Using variables x_1, \dots, 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 $\text{boy}^{e \rightarrow t}$ and $\text{hurt}^{e \rightarrow e \rightarrow t}$, we obtain, after β conversion, $(\exists \lambda y. ((\text{boy } y) \wedge ((\text{hurt } y) y)))$. 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.

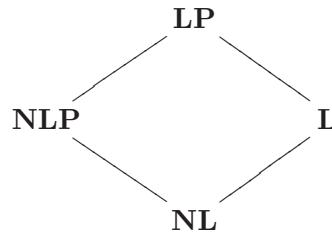


FIGURE 13. Various Lambek calculi

$$\frac{\frac{a}{(s/(np\backslash s))/n} \quad \frac{\text{boy}}{n} \quad (/E)}{(a, \text{boy}) \vdash s/(np\backslash s)} \quad \frac{\frac{\text{hurt}}{(np\backslash s)/np} \quad \frac{\text{himself}}{((np\backslash s)/np)\backslash(np\backslash s)} \quad (\backslash E)}{(\text{hurt}, \text{himself}) \vdash np\backslash s} \quad (/E) \\
 ((a, \text{boy}), (\text{hurt}, \text{himself})) \vdash s$$

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 $(A \rightarrow B) \rightarrow C$. 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 $(M^{(A \rightarrow B) \rightarrow C} \lambda x^A. N^B)^C$, 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 well-formedness 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.

$$\mathcal{F} ::= \mathcal{A} \mid \Diamond \mathcal{F} \mid \Box \mathcal{F} \mid \mathcal{F} \setminus \mathcal{F} \mid \mathcal{F} \bullet \mathcal{F} \mid \mathcal{F} / \mathcal{F}$$

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 $\Diamond A \rightarrow B$ iff $A \rightarrow \Box B$ (Res).

$$x \in V(\Diamond A) \text{ iff } \exists y. R_\diamond xy \text{ and } y \in V(A) \quad x \in V(\Box A) \text{ iff } \forall y. R_\diamond yx \text{ implies } y \in V(A)$$

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 ($A \rightarrow B$ implies $\Diamond A \rightarrow \Diamond B$ and $\Box A \rightarrow \Box B$), and that their compositions satisfy $\Diamond \Box A \rightarrow A \rightarrow \Box \Diamond A$. 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.

$$\frac{\Gamma[(A)] \Rightarrow B}{\Gamma[\Diamond A] \Rightarrow B} \Diamond L \quad \frac{\Gamma \Rightarrow A}{(\Gamma) \Rightarrow \Diamond A} \Diamond R$$

$$\frac{\Gamma[A] \Rightarrow B}{\Gamma[(\Box A)] \Rightarrow B} \Box L \quad \frac{(\Gamma) \Rightarrow A}{\Gamma \Rightarrow \Box A} \Box R$$

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 re-bracketing of $(np, ((np \setminus s)/np, np))$ under associativity. The (c) example shows that apart from re-bracketing also reordering would be required to access a non-peripheral

gap.

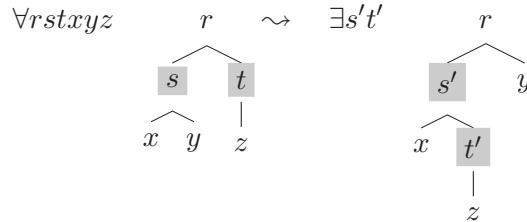
- (a) the paper that appeared today $(n \setminus n)/(np \setminus s)$
- (b) the paper that John wrote $(n \setminus n)/(s/np) + \text{Ass}$
- (c) the paper that John wrote today $(n \setminus n)/(s/np) + \text{Ass,Com}$

The controlled structural rules below allow the required restructuring and reordering only for \Diamond marked resources. In combination with a type assignment $(n \setminus n)/(s/\Diamond\Box np)$ 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 $\Diamond\Box np \rightarrow np$.

$$(P1) \quad (A \bullet B) \bullet \Diamond C \longrightarrow A \bullet (B \bullet \Diamond C) \quad (P2) \quad (A \bullet B) \bullet \Diamond C \longrightarrow (A \bullet \Diamond C) \bullet B$$

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 , proof-theoretically 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 sub-structural communication in terms of \Diamond, \Box is worked out in [Kurtonina and Moortgat \[1997\]](#). Let \mathcal{L} and \mathcal{L}' be neighbors in the landscape of Fig. 13.

We have translations \cdot^\natural from $\mathcal{F}(/, \bullet, \backslash)$ of \mathcal{L} to $\mathcal{F}(\diamond, \square, /, \bullet, \backslash)$ of \mathcal{L}' such that

$$\mathcal{L} \vdash A \rightarrow B \text{ iff } \mathcal{L}' \vdash A^\natural \rightarrow B^\natural$$

The \cdot^\natural translation decorates formulas of the source logic \mathcal{L} with the control operators \diamond, \square . 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 neighbor **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 $(\text{Ass}_\diamond) \diamond(A \bullet \diamond(B \bullet C)) \longleftrightarrow \diamond(\diamond(A \bullet B) \bullet C)$, the image of (Ass) under \cdot^\natural .

$$\begin{aligned} p^\natural &= p \quad (p \in \mathcal{A}) \\ (A \bullet B)^\natural &= \diamond(A^\natural \bullet B^\natural) \\ (A/B)^\natural &= \square A^\natural / B^\natural \\ (B \setminus A)^\natural &= B^\natural \setminus \square A^\natural \end{aligned}$$

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 \[1993\], \[2006\]](#): 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 \diamond controlled structural rules is studied in [Moot \[2002\]](#), who establishes a PSPACE complexity ceiling for linear (for \bullet), non-expanding (for \diamond) 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 than weak capacity. [Tiede \[2001\], \[2002\]](#)

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 neighbor **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

In the Handbook of Logic and Language, [van Benthem and ter Meulen \[1997\]](#), the material discussed in this section is covered in greater depth in the chapters of Moortgat and Buszkowski. The monograph [van Benthem \[1995\]](#) is indispensable 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 grammar development environment GRAIL of [Moot \[2002\]](#). The kernel is a general type-logical theorem prover based on proof nets and structural graph rewriting. [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 [Baldridge \[2002\]](#) 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 map 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 and Pogodalla \[2004\]](#). 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 theory of structural control, see [Kurtonina and Moortgat \[1997\]](#) shows that the two viewpoints are feasible.

Part 2

RECURSIVE TYPES $\lambda^A_=_$

The simple types of λ_{\rightarrow} of Part I are *freely* generated from the type atoms \mathbb{A} . This means that there are no identifications like $\alpha = \alpha \rightarrow \beta$ or $0 \rightarrow 0 = (0 \rightarrow 0) \rightarrow 0$.

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. Another way of obtaining type identifications is to add the ‘fixed-point operator’ μ for types 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 $X = A[X]$. Following a suggestion of Dana Scott [1975b], both approaches (types modulo a set of equations \mathcal{E} or using the operator μ) 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. $a = a \rightarrow b$). 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. Recursive type specifications are used in programming languages. One can, for example, define the type of lists of elements of type A by the equation

$$\text{list} = 1 + (A \times \text{list}).$$

For this we need 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. Recursive types have been used in several programming languages since ALGOL-68, see van Wijngaarden [1981] and Pierce [2002].

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 as follows.

$$\frac{\begin{array}{c} x:C \vdash x : C \\ \hline x:C \vdash x:C \rightarrow A & x:C \vdash x : C \end{array}}{\frac{x:C \vdash xx : A}{\frac{\vdash \lambda x.xx : C \rightarrow A}{\vdash \lambda x.xx : C}} C \rightarrow A = C}$$

Another example is the fixed-point operator $\mathbf{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ that now will have as type $(A \rightarrow A) \rightarrow A$ for all types A such that there exists 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 regained.

Finally, we also consider type structures in which type algebras are enriched with a partial order, so that now one can have $a \leq a \rightarrow b$. Subtyping could be pursued much further, looking at systems of inequalities as generalized simultaneous recursions. Here we limit our treatment to a few basic properties: type systems featuring subtyping will be dealt with thoroughly in the next Part III.

CHAPTER 7

THE SYSTEMS $\lambda^A_=_$

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 [1975b] this can be described by considering type algebras consisting of a set with a binary operation \rightarrow on it. In such structures one can have for example $a = a \rightarrow b$. The notion of type algebra was anticipated in Breazu-Tannen and Meyer [1985] expanding on a remark of Scott [1975b]; it was taken up in Statman [1994] as an alternative to the presentation of recursive types via the μ -operator. It will be used as a unifying theme throughout this Part.

7A. Type-algebras and type assignment

Type algebras

7A.1. DEFINITION. (i) A *type algebra* is a structure

$$\mathcal{A} = \langle |\mathcal{A}|, \rightarrow_{\mathcal{A}} \rangle,$$

where $\rightarrow_{\mathcal{A}}$ is a binary operation on $|\mathcal{A}|$.

(ii) The type-algebra $\langle \mathbb{T}^{\mathbb{A}}, \rightarrow \rangle$, consisting of the simple types under the operation \rightarrow , is called the *free type algebra* over \mathbb{A} . This terminology will be justified in 7B.1 below.

NOTATION. (i) 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 $|\mathcal{A}|$ and \rightarrow for $\rightarrow_{\mathcal{A}}$.

(ii) We will use α, β, \dots to denote arbitrary elements of \mathbb{A} and A, B, C, \dots to range over $\mathbb{T}^{\mathbb{A}}$. On the other hand a, b, c, \dots range over a type algebra \mathcal{A} .

Type assignment à la Curry

We now introduce formal systems for assigning elements of a type algebra 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 formal rules to assign types to λ -terms are defined as in Section 1A, 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.

$$\Gamma \vdash M : a,$$

where one has $a \in \mathcal{A}$ and Γ , called a *basis* over \mathcal{A} , 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 $\Gamma = \{x_1:a_1, \dots, x_n:a_n\}$ should be distinct, i.e. $x_i = x_j \Rightarrow i = j$.

7A.2. DEFINITION. Let \mathcal{A} be a Type Algebra, $a, b \in \mathcal{A}$, and let $M \in \Lambda$. Then the *Curry* system of type assignment $\lambda_{\equiv}^{A,Cu}$, or simply λ_{\equiv}^A , is defined by the following rules.

$\text{(axiom)} \quad \Gamma \vdash x : a \quad \text{if } (x:a) \in \Gamma$ $\text{(\rightarrow E)} \quad \frac{\Gamma \vdash M : a \rightarrow b \quad \Gamma \vdash N : a}{\Gamma \vdash (MN) : b}$ $\text{(\rightarrow I)} \quad \frac{\Gamma, x:a \vdash M : b}{\Gamma \vdash (\lambda x.M) : (a \rightarrow b)}$
--

FIGURE 14. The system λ_{\equiv}^A .

In rule $(\rightarrow I)$ it is assumed that $\Gamma, x:a$ is a basis.

We write $\Gamma \vdash_{\lambda_{\equiv}^A} M : a$, or simply $\Gamma \vdash_{\lambda_{\equiv}^A} M : a$, in case $\Gamma \vdash M : a$ can be derived in λ_{\equiv}^A .

We could denote this system by λ_{\rightarrow}^A , but we write λ_{\equiv}^A to emphasize the difference with the system $\lambda_{\rightarrow}^{\mathbb{A}}$, which is λ_{\equiv}^A over the free type algebra $\mathcal{A} = \mathbb{T}^{\mathbb{A}}$. In a general \mathcal{A} we can have identifications, for example $b = b \rightarrow a$ and then of course we have

$$\Gamma \vdash_{\mathcal{A}} M : b \Rightarrow \Gamma \vdash_{\mathcal{A}} M : (b \rightarrow a).$$

This makes a dramatic difference. There are examples of type assignment in λ_{\equiv}^A to terms which have no type in the simple type assignment system $\lambda_{\rightarrow}^{\mathbb{A}}$.

7A.3. EXAMPLE. Let \mathcal{A} be a type algebra and let $a, b \in \mathcal{A}$ with $b = (b \rightarrow a)$. Then

- (i) $\vdash_{\mathcal{A}} (\lambda x.xx) : b$.
- (ii) $\vdash_{\mathcal{A}} \Omega : a$, where $\Omega \triangleq (\lambda x.xx)(\lambda x.xx)$.
- (iii) $\vdash_{\mathcal{A}} Y : (a \rightarrow a) \rightarrow a$,

where $Y \triangleq \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ is the fixed point combinator.

PROOF. (i) The following is a deduction of $\vdash_{\mathcal{A}} (\lambda x.xx) : b$.

$$\frac{\frac{x:b \vdash x : b \quad x:b \vdash x : b}{x:b \vdash xx : a} (\rightarrow E), \quad b = (b \rightarrow a)}{\vdash_{\mathcal{A}} (\lambda x.xx) : (b \rightarrow a) = b}$$

(ii) As $\vdash_{\mathcal{A}} (\lambda x.xx) : b$, we also have $\vdash_{\mathcal{A}} (\lambda x.xx) : (b \rightarrow a)$, since $b = b \rightarrow a$. Therefore $\vdash_{\mathcal{A}} (\lambda x.xx)(\lambda x.xx) : a$.

(iii) We can prove $\vdash_{\mathcal{A}} Y : (a \rightarrow a) \rightarrow a$ in λ_{\equiv}^A in the following way. First modify the deduction constructed in (i) to obtain $f:a \rightarrow a \vdash_{\mathcal{A}} \lambda x.f(xx) : b$. Since $b = b \rightarrow a$ we have as in (ii) by rule $(\rightarrow E)$

$$f : a \rightarrow a \vdash_{\mathcal{A}} (\lambda x.f(xx))(\lambda x.f(xx)) : a$$

from which we get

$$\vdash_{\mathcal{A}} \lambda f.(\lambda x.f(xx))(\lambda x.f(xx)) : (a \rightarrow a) \rightarrow a. \blacksquare$$

7A.4. PROPOSITION. Suppose that $\Gamma \subseteq \Gamma'$. Then

$$\Gamma \vdash_{\mathcal{A}} M : a \Rightarrow \Gamma' \vdash_{\mathcal{A}} M : A.$$

We say that the rule ‘weakening’ is admissible.

PROOF. By induction on derivations. \blacksquare

Quotients and syntactic type-algebras and morphisms

A ‘recursive type’ b satisfying $b = (b \rightarrow a)$ can be easily obtained by working modulo the right equivalence relations.

7A.5. DEFINITION. (i) A *congruence* on a type algebra $\mathcal{A} = \langle \mathcal{A}, \rightarrow \rangle$ is an equivalence relation \approx on \mathcal{A} such that for all $a, b, a', b' \in \mathcal{A}$ one has

$$a \approx a' \& b \approx b' \Rightarrow (a \rightarrow b) \approx (a' \rightarrow b').$$

(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

$$\langle \mathcal{A}/\approx, \rightarrow_{\approx} \rangle,$$

where

$$\begin{aligned} \mathcal{A}/\approx &\triangleq \{[a]_{\approx} \mid a \in \mathcal{A}\} \\ [a]_{\approx} \rightarrow_{\approx} [b]_{\approx} &\triangleq [a \rightarrow b]_{\approx}. \end{aligned}$$

Since \approx is a congruence, the operation \rightarrow_{\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 7A.16 we shall see that every type algebra has this form, up to isomorphism.

7A.6. DEFINITION. Let $\mathbb{T} = \mathbb{T}^{\mathbb{A}}$.

(i) A *syntactic* type-algebra over \mathbb{A} is of the form

$$\mathcal{A} = \langle \mathbb{T}/\approx, \rightarrow_{\approx} \rangle,$$

where \approx is a congruence on $\langle \mathbb{T}, \rightarrow \rangle$.

(ii) We usually write \mathbb{T}/\approx for the syntactic type-algebra $\langle \mathbb{T}/\approx, \rightarrow_{\approx} \rangle$, as no confusion can arise since \rightarrow_{\approx} is determined by \approx .

7A.7. REMARK. (i) 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 .

(ii) The free type-algebra over \mathbb{A} is also syntactic, in fact it is the same as $\mathbb{T}^{\mathbb{A}}/=_\mathbb{A}$, where $=$ is the ordinary equality relation on $\mathbb{T}^{\mathbb{A}}$. This algebra will henceforth be denoted simply by $\mathbb{T}^{\mathbb{A}}$.

7A.8. 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} , notation¹ $h : \mathcal{A} \rightarrow \mathcal{B}$, iff for all $a, b \in \mathcal{A}$ one has

$$h(a \rightarrow_{\mathcal{A}} b) = h(a) \rightarrow_{\mathcal{B}} h(b).$$

(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 $i : \mathcal{A} \hookrightarrow \mathcal{B}$.

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).

7A.9. DEFINITION. Let \mathcal{A} be a type algebra.

- (i) An *equation over* \mathcal{A} is of the form $(\underline{a} = \underline{b})$ with $a, b \in \mathcal{A}$.
- (ii) \mathcal{A} *satisfies* such an equation $\underline{a} = \underline{b}$ (or $\underline{a} = \underline{b}$ *holds* in \mathcal{A}), notation

$$\mathcal{A} \models \underline{a} = \underline{b},$$

if $a = b$.

- (iii) \mathcal{A} *satisfies a set \mathcal{E} of equations over* \mathcal{A} , notation

$$\mathcal{A} \models \mathcal{E},$$

if every equation $\underline{a} = \underline{b} \in \mathcal{E}$ holds in \mathcal{A} .

Here \underline{a} is the corresponding constant for an element $a \in \mathcal{A}$. But usually we will write for $\underline{a} = \underline{b}$ simply $a = b$.

7A.10. DEFINITION. Let \mathcal{A} be a type-algebra and let \mathcal{E} be a set of equations over \mathcal{A} .

- (i) The *least congruence relation on \mathcal{A} extending \mathcal{E}* is introduced via an equality defined by the following axioms and rules, where a, a', b, b', c range over \mathcal{A} . The system of *equational logic* extended by the statements in \mathcal{E} , notation (\mathcal{E}) , is defined as follows.

¹This is an overloading of the symbol “ \rightarrow ” with little danger of confusion.

(axiom)	$\mathcal{E} \vdash a = b$	if $(a = b) \in \mathcal{E}$
(refl)	$\mathcal{E} \vdash a = a$	
(symm)	$\frac{\mathcal{E} \vdash a = b}{\mathcal{E} \vdash b = a}$	
(trans)	$\frac{\mathcal{E} \vdash a = b \quad \mathcal{E} \vdash b = c}{\mathcal{E} \vdash a = c}$	
(\rightarrow -cong)	$\frac{\mathcal{E} \vdash a = a' \quad \mathcal{E} \vdash b = b'}{\mathcal{E} \vdash a \rightarrow b = a' \rightarrow b'}$	

FIGURE 15. The system of equational logic (\mathcal{E}) .

If \mathcal{E}' is another set of equations over \mathcal{A} we write

$$\mathcal{E} \vdash \mathcal{E}'$$

if $\mathcal{E} \vdash a = b$ for all $a = b \in \mathcal{E}'$.

(ii) Write $=_{\mathcal{E}} \triangleq \{(a, b) \mid a, b \in \mathcal{A} \text{ & } \mathcal{E} \vdash a = b\}$. This is the least congruence relation extending \mathcal{E} .

(iii) The *quotient type-algebra \mathcal{A} modulo \mathcal{E}* , notation \mathcal{A}/\mathcal{E} is defined as

$$\mathcal{A}/\mathcal{E} \triangleq (\mathcal{A}/=_{\mathcal{E}}).$$

If we want to construct recursive types a, b such that $b = b \rightarrow a$, then we simply work modulo $=_{\mathcal{E}}$, with $\mathcal{E} = \{b = b \rightarrow a\}$.

7A.11. DEFINITION. Let $h : \mathcal{A} \rightarrow \mathcal{B}$ be a morphism between type algebras.

- (i) For $a_1, a_2 \in \mathcal{A}$ define $h(a_1 = a_2) \triangleq (h(a_1) = h(a_2))$.
- (ii) $h(\mathcal{E}) \triangleq \{h(a_1 = a_2) \mid a_1 = a_2 \in \mathcal{E}\}$.

7A.12. LEMMA. Let \mathcal{E} be a set of equations over \mathcal{A} and let $a, b \in \mathcal{A}$.

- (i) $\mathcal{A} \models \mathcal{E} \& \mathcal{E} \vdash a = b \Rightarrow \mathcal{A} \models a = b$.

Let moreover $h : \mathcal{A} \rightarrow \mathcal{B}$ be a morphism. Then

- (ii) $\mathcal{A} \models a_1 = a_2 \Rightarrow \mathcal{B} \models h(a_1 = a_2)$.
- (iii) $\mathcal{A} \models \mathcal{E} \Rightarrow \mathcal{B} \models h(\mathcal{E})$.

PROOF. (i) By induction on the proof of $\mathcal{E} \vdash a = b$.

- (ii) Since $h(a_1 = a_2) = (h(a_1) = h(a_2))$.
- (iii) By (ii). ■

7A.13. REMARK. (i) Slightly *misusing language* we simply state that $a = b$, instead of $[a] = [b]$, holds in \mathcal{A}/\mathcal{E} . This is comparable to saying that $1+2=0$ holds in $\mathbb{Z}/(3)$, rather than saying that $[1]_{(3)} + [2]_{(3)} = [0]_{(3)}$ holds.

- (ii) Similarly we write sometimes $h(a) = b$ instead of $h([a]) = [b]$.

7A.14. LEMMA. Let \mathcal{E} be a set of equations over \mathcal{A} and let $a, b \in \mathcal{A}$. Then

- (i) $\mathcal{A}/\mathcal{E} \models a = b \Leftrightarrow \mathcal{E} \vdash a = b$.
- (ii) $\mathcal{A}/\mathcal{E} \models \mathcal{E}$.

PROOF. (i) By the definition of \mathcal{A}/\mathcal{E} .

- (ii) By (i). ■

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 7A.10(iii), 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} .

7A.15. EXAMPLE. (i) Let $\mathbb{T}^0 = \mathbb{T}^{\{0\}}$, $\mathcal{E}_1 = \{0 = 0 \rightarrow 0\}$. Then all elements of \mathbb{T}^0 are equated in $\mathbb{T}^0/\mathcal{E}_1$. As a type algebra, $\mathbb{T}^0/\mathcal{E}_1$ contains therefore only one element $[0]_{\mathcal{E}_1}$ (that will be identified with 0 itself by Remark 7A.7(i)). For instance we have

$$\mathbb{T}^0/\mathcal{E}_1 \models 0 = 0 \rightarrow 0 \rightarrow 0.$$

Moreover we have that 0 is a solution for $X = X \rightarrow 0$ in $\mathbb{T}^0/\mathcal{E}_1$.

At the semantic level an equation like $0 = 0 \rightarrow 0$ 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 7G.12).

(ii) Let $\mathbb{T}^\infty = \mathbb{T}^{\mathbb{A}_\infty}$ be a set of types with $\infty \in \mathbb{A}_\infty$. Define \mathcal{E}_∞ as the set of equations

$$\infty = T \rightarrow \infty, \quad \infty = \infty \rightarrow T,$$

where T ranges over \mathbb{T}^∞ . 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}^∞ , where $A(X)$ is any type expression over \mathbb{T}^∞ with at least one free occurrence of X . Note that in $\mathbb{T}^\infty/\mathcal{E}_\infty$ one does not have that $a \rightarrow b = a' \rightarrow b' \Rightarrow a = a' \& b = b'$.

We now show that every type-algebra can be considered as a syntactic one.

7A.16. PROPOSITION. *Every type-algebra is isomorphic to a syntactic one.*

PROOF. Given $\mathcal{A} = \langle \mathcal{A}, \rightarrow \rangle$, take $\mathbb{A} = \{\underline{a} \mid a \in \mathcal{A}\}$ and

$$\mathcal{E} = \{\underline{a} \rightarrow \underline{b} = \underline{a} \rightarrow \underline{b} \mid a, b \in \mathbb{A}\}.$$

Then \mathcal{A} is isomorphic to $\mathbb{T}^{\mathbb{A}}/\mathcal{E}$ via the isomorphism $a \mapsto [\underline{a}]_{\mathcal{E}}$. ■

7A.17. DEFINITION. Let \mathcal{E} be a set of equations over \mathcal{A} and let \mathcal{B} be a type algebra.

(i) \mathcal{B} *justifies* \mathcal{E} if for some $h: \mathcal{A} \rightarrow \mathcal{B}$

$$\mathcal{B} \models h(\mathcal{E}).$$

(ii) \mathcal{E}' over \mathcal{B} *justifies* \mathcal{E} if \mathcal{B}/\mathcal{E}' justifies \mathcal{E} .

The intention is that h interprets the constants of \mathcal{E} in \mathcal{B} in such a way that the equations as seen in \mathcal{B} become valid. We will see in Proposition 7B.7 that

$$\mathcal{B} \text{ justifies } \mathcal{E} \Leftrightarrow \text{there exists a morphism } h: \mathcal{A}/\mathcal{E} \rightarrow \mathcal{B}.$$

Type assignment in a syntactic type algebra

7A.18. NOTATION. If $\mathcal{A} = \mathbb{T}/\approx$ is a syntactic type algebra, then we write

$$x_1:A_1, \dots, x_n:A_n \vdash_{\mathbb{T}/\approx} M : A$$

for

$$x_1:[A_1]_{\approx}, \dots, x_n:[A_n]_{\approx} \vdash_{\mathbb{T}/\approx} M : [A]_{\approx}.$$

We will present systems often in the following form.

7A.19. PROPOSITION. *The system of type assignment $\lambda_{\equiv}^{\mathbb{T}/\approx}$ can be axiomatized by the following axioms and rules.*

(axiom)	$\Gamma \vdash x : A \quad \text{if } (x:A) \in \Gamma$
(\rightarrow E)	$\frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash (MN) : B}$
(\rightarrow I)	$\frac{\Gamma, x:A \vdash M : B}{\Gamma \vdash (\lambda x.M) : (A \rightarrow B)}$
(equal)	$\frac{\Gamma \vdash M : A \quad A \approx B}{\Gamma \vdash M : B}$

FIGURE 16. The system $\lambda_{\equiv}^{\mathbb{T}/\approx}$.

where now A, B range over \mathbb{T} and Γ is of the form $\{x_1:A_1, \dots, x_n:A_n\}$, $\vec{A} \in \mathbb{T}$.

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.

7A.20. LEMMA. *Let $h : \mathcal{A} \rightarrow \mathcal{B}$ be a type algebra morphism. Then for $\Gamma = \{x_1:A_1, \dots, x_n:A_n\}$*

$$\Gamma \vdash_{\mathcal{A}} M : A \Rightarrow h(\Gamma) \vdash_{\mathcal{B}} M : h(A),$$

where $h(\Gamma) \triangleq \{x_1:h(A_1), \dots, x_n:h(A_n)\}$.

In Chapter 9 we will prove the following properties of type assignment.

1. A type assignment system $\lambda_{\equiv}^{\mathcal{A}}$ has the subject reduction property for β -reduction iff \mathcal{A} is invertible: $a \rightarrow b = a' \rightarrow b' \Rightarrow a = a' \& b = b'$, 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 λ_{\rightarrow} . 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

Explicitly typed versions of λ -calculus with recursive types can also be defined as for the simply 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, see Definition 1C.4.

The Church version

7A.21. DEFINITION. Let $\mathcal{A} = \mathbb{T}/\approx$ be a syntactic type algebra and $A, B \in \mathcal{A}$. We introduce a *Church version of λ_{\equiv}^A* , notation $\lambda_{\equiv}^{A,\text{Ch}}$. The set of *typed terms of the system $\lambda_{\equiv}^{A,\text{Ch}}$* , notation $\Lambda_{\equiv}^{A,\text{Ch}}(A)$ for each type A , is defined by the following term formation rules.

$$\boxed{\begin{array}{l} x^A \in \Lambda_{\equiv}^{A,\text{Ch}}(A); \\ M \in \Lambda_{\equiv}^{A,\text{Ch}}(A \rightarrow B), N \in \Lambda_{\equiv}^{A,\text{Ch}}(A) \Rightarrow (MN) \in \Lambda_{\equiv}^{A,\text{Ch}}(B); \\ M \in \Lambda_{\equiv}^{A,\text{Ch}}(B) \Rightarrow (\lambda x^A.M) \in \Lambda_{\equiv}^{A,\text{Ch}}(A \rightarrow B); \\ M \in \Lambda_{\equiv}^{A,\text{Ch}}(A) \text{ and } A \approx B \Rightarrow M \in \Lambda_{\equiv}^{A,\text{Ch}}(B). \end{array}}$$

FIGURE 17. The family $\Lambda_{\equiv}^{A,\text{Ch}}$ of typed terms.

This is not a type assignment system but a disjoint family of typed terms.

The de Bruijn version

A formulation of the system in the “de Bruijn” style is 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 Chapter 10. Given a syntactic type algebra $\mathcal{A} = \mathbb{T}/\approx$ the formulation of the system $\lambda_{\equiv}^{A,\text{dB}}$ in the de Bruijn style is given by the rules in Fig. 18.

$$\boxed{\begin{array}{ll} (\text{axiom}) & \Gamma \vdash x : A \quad \text{if } (x:A) \in \Gamma \\ (\rightarrow\text{E}) & \frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B} \\ (\rightarrow\text{I}) & \frac{\Gamma, x:A \vdash M : B}{\Gamma \vdash (\lambda x:A.M) : A \rightarrow B} \\ (\text{equiv}) & \frac{\Gamma \vdash M : A \quad A \approx B}{\Gamma \vdash M : B} \end{array}}$$

FIGURE 18. The system $\lambda_{\equiv}^{A,\text{dB}}$.

Theorems 1B.19, 1B.32, 1B.35, and 1B.36, relating the systems $\lambda_{\rightarrow}^{\text{Cu}}$, $\lambda_{\rightarrow}^{\text{Ch}}$, and $\lambda_{\rightarrow}^{\text{dB}}$, also hold after a change of notations, for example $\lambda_{\rightarrow}^{\text{Ch}}$ must be changed into $\lambda_{\equiv}^{A,\text{Ch}}$, for the systems of recursive types $\lambda_{\equiv}^{A,\text{Cu}}$, $\lambda_{\equiv}^{A,\text{Ch}}$, and $\lambda_{\equiv}^{A,\text{dB}}$. The proofs are equally simple.

The Church version with coercions

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 $\lambda_{\equiv}^{\mathcal{A}, \text{Ch}}$ 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. To this aim we need to introduce new constants representing, in a syntactic type algebra, the equality axioms between types. The most interesting case is when these equalities are of the form $\alpha = A$ with α an atomic type. Equations of this form will be extensively studied and motivated in Section 7C).

7A.22. DEFINITION. Let $\mathcal{A} = \mathbb{T}/=_\mathcal{E}$, where \mathcal{E} is a set of type equations of the form $\alpha = A$ with α an atomic type. We introduce a system $\lambda_{\equiv}^{\mathcal{A}, \text{Cho}}$.

(i) The set of *typed terms of the system* $\lambda_{\equiv}^{\mathcal{A}, \text{Cho}}$, notation $\Lambda_{\equiv}^{\mathcal{A}, \text{Cho}}(A)$ for each type A , is defined as follows

$$\boxed{\begin{aligned} x^A &\in \Lambda_{\equiv}^{\mathcal{A}, \text{Cho}}(A); \\ \alpha = A \in \mathcal{E} &\Rightarrow \text{fold}_\alpha \in \Lambda_{\equiv}^{\mathcal{A}, \text{Cho}}(A \rightarrow \alpha); \\ \alpha = A \in \mathcal{E} &\Rightarrow \text{unfold}_\alpha \in \Lambda_{\equiv}^{\mathcal{A}, \text{Cho}}(\alpha \rightarrow A); \\ M \in \Lambda_{\equiv}^{\mathcal{A}, \text{Cho}}(A \rightarrow B), N \in \Lambda_{\equiv}^{\mathcal{A}, \text{Cho}}(A) &\Rightarrow (MN) \in \Lambda_{\equiv}^{\mathcal{A}, \text{Cho}}(B); \\ M \in \Lambda_{\equiv}^{\mathcal{A}, \text{Cho}}(B) &\Rightarrow (\lambda x^A. M) \in \Lambda_{\equiv}^{\mathcal{A}, \text{Cho}}(A \rightarrow B). \end{aligned}}$$

FIGURE 19. The family $\Lambda_{\equiv}^{\mathcal{A}, \text{Cho}}$ of typed terms.

The terms $\text{fold}_\alpha, \text{unfold}_\alpha$ are called *coercions* and represent the two ways in which the equation $\alpha = A$ can be applied. This will be exploited in Section 7C.

(ii) Add for each equation $\alpha = A \in \mathcal{E}$ the following reduction rules.

$$\boxed{\begin{aligned} (R_{\mathcal{E}}^{\text{uf}}) \quad \text{unfold}_\alpha(\text{fold}_\alpha M^A) &\rightarrow M^A, \quad \text{if } \alpha = A \in \mathcal{E}; \\ (R_{\mathcal{E}}^{\text{fu}}) \quad \text{fold}_\alpha(\text{unfold}_\alpha M^\alpha) &\rightarrow M^\alpha, \quad \text{if } \alpha = A \in \mathcal{E}. \end{aligned}}$$

FIGURE 20. The reduction rules on typed terms in $\Lambda_{\equiv}^{\mathcal{A}, \text{Cho}}$.

The rules $(R_{\mathcal{E}}^{\text{uf}})$ and $(R_{\mathcal{E}}^{\text{fu}})$ represent the isomorphism between α and A expressed by the equation $\alpha = A$.

7A.23. EXAMPLE. Let $\mathcal{E} \triangleq \{\alpha = \alpha \rightarrow \beta\}$. The following term is the version of $\lambda x.xx$ in the system $\lambda_{\equiv}^{\mathcal{A}, \text{Cho}}$ above.

$$\text{fold}_\alpha(\lambda x^\alpha. (\text{unfold}_\alpha x^\alpha) x^\alpha) \in \Lambda_{\mathcal{A}}^{\text{Cho}}(\alpha)$$

The system $\lambda_{\equiv}^{\mathcal{A}, \text{Cho}}$ in which all type equivalences are expressed via coercions is equivalent to the system $\lambda_{\equiv}^{\mathcal{A}, \text{Ch}}$, in the sense that for each term $M \in \Lambda_{\equiv}^{\mathcal{A}, \text{Ch}}(A)$ there is a term $M' \in \Lambda_{\equiv}^{\mathcal{A}, \text{Cho}}(A)$ obtained from an η -expansion of M by adding some coercions. Conversely for each term $M' \in \Lambda_{\equiv}^{\mathcal{A}, \text{Cho}}(A)$ there is a term $M \in \Lambda_{\equiv}^{\mathcal{A}, \text{Ch}}(A)$ which is η -equivalent to a term $M'' \in \Lambda_{\equiv}^{\mathcal{A}, \text{Ch}}(A)$ obtained from M by erasing all its coercions.

For instance working with $\mathcal{E} = \{\alpha = \alpha \rightarrow \beta\}$ of example 7A.23 and the term $x^{\alpha \rightarrow \gamma}$ one has $\lambda y^{\alpha \rightarrow \beta}. x^{\alpha \rightarrow \gamma} (\text{fold}_\alpha y^{\alpha \rightarrow \beta}) \in \Lambda_{\equiv}^{\mathcal{A}, \text{Ch}}((\alpha \rightarrow \beta) \rightarrow \gamma)$, as $\alpha \rightarrow \gamma =_{\mathcal{E}} (\alpha \rightarrow \beta) \rightarrow \gamma$. See also Exercise 7G.16.

For many interesting terms of $\lambda_{\underline{\underline{}}^A}^{A,\text{Ch}_0}$, however, η -conversion is not needed to obtain the equivalent term in $\lambda_{\underline{\underline{}}^A}^{A,\text{Ch}}$, as in the case of Example 7A.23.

Definition 7A.21 identifies equivalent types, and therefore one term can have infinitely many types (though all equivalent to each other). Such presentations have been called *equi-recursive* in the recent literature Gapeyev, Levin, and Pierce [2002], and are more interesting both from the practical and the theoretical point of view, especially when designing corresponding type checking algorithms. The formulation with explicit coercions is classified as *iso-recursive*, due to the presence of explicit coercions from a recursive type to its unfolding and conversely. We shall not pursue this matter, but refer the reader to Abadi and Fiore [1996] which is, to our knowledge, the only study of this issue, in the context of a call-by-value formulation of the system FPC, see Plotkin [1985].

7B. More on type algebras

Free algebras

7B.1. DEFINITION. 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 $f : \mathbb{A} \rightarrow \mathcal{B}$, there is a unique morphism $f^+ : \mathcal{A} \rightarrow \mathcal{B}$ such that, for any $\alpha \in \mathbb{A}$, one has $f^+(\alpha) = f(\alpha)$; in diagram

$$(1) \quad \begin{array}{ccc} \mathbb{A} & \xrightarrow{f} & \mathcal{B} \\ i \downarrow & \nearrow f^+ & \\ \mathcal{A}, & & \end{array}$$

where $i : \mathbb{A} \rightarrow \mathcal{A}$ is the embedding map.

The following result, see, e.g. Goguen, Thatcher, Wagner, and Wright [1977], Proposition 2.3, characterizes the free type algebra over a set of atoms \mathbb{A} :

7B.2. PROPOSITION. $\langle \mathbb{T}^{\mathbb{A}}, \rightarrow \rangle$ is the free type algebra over \mathbb{A} .

PROOF. Given a map $f : \mathbb{A} \rightarrow \mathcal{B}$, define a morphism $f^+ : \mathbb{T}^{\mathbb{A}} \rightarrow \mathcal{B}$ as follows:

$$\begin{aligned} f^+(\alpha) &= f(\alpha) \\ f^+(A \rightarrow B) &= f^+(A) \rightarrow_B f^+(B). \end{aligned}$$

This is clearly the unique morphism that makes diagram (1) commute. ■

Subalgebras, quotients and morphisms

7B.3. DEFINITION. Let $\mathcal{A} = \langle \mathcal{A}, \rightarrow_{\mathcal{A}} \rangle$, $\mathcal{B} = \langle \mathcal{B}, \rightarrow_{\mathcal{B}} \rangle$ 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

$$\rightarrow_{\mathcal{A}} = \rightarrow_{\mathcal{B}} \upharpoonright \mathcal{A},$$

i.e. for all $a_1, a_2 \in \mathcal{A}$ one has $a_1 \rightarrow_{\mathcal{A}} a_2 = a_1 \rightarrow_{\mathcal{B}} a_2$.

Clearly any subset of \mathcal{B} closed under $\rightarrow_{\mathcal{B}}$ induces a sub type algebra of \mathcal{B} .

7B.4. PROPOSITION. Let \mathcal{A}, \mathcal{B} be type algebras and \approx be a congruence on \mathcal{A} .

(i) Given a morphism $f : \mathcal{A} \rightarrow \mathcal{B}$ such that $\mathcal{B} \models f(\approx)$, i.e. $\mathcal{B} \models \{f(a) = f(a') \mid a \approx a'\}$, then there is a unique morphism $f^\sharp : \mathcal{A}/\approx \rightarrow \mathcal{B}$ such that $f^\sharp([a]_\approx) = f(a)$.

$$\begin{array}{ccc} \mathcal{A} & \xrightarrow{f} & \mathcal{B} \\ & \searrow []_\approx & \nearrow f^\sharp \\ & \mathcal{A}/\approx & \end{array}$$

Moreover, $[]_\approx$ is surjective.

(ii) If $\forall a, a' \in \mathcal{A}. [f(a) = f(a')] \Rightarrow a \approx a'$, then f^\sharp is injective.

(iii) Given a morphism $f : \mathcal{A}/\approx \rightarrow \mathcal{B}$, write $f^\sharp = f \circ []_\approx$.

$$\begin{array}{ccc} \mathcal{A} & \xrightarrow{f^\sharp} & \mathcal{B} \\ & \searrow []_\approx & \nearrow f \\ & \mathcal{A}/\approx & \end{array}$$

Then $f^\sharp : \mathcal{A} \rightarrow \mathcal{B}$ is a morphism such that $\mathcal{B} \models f^\sharp(\approx)$.

(iv) Given a morphism $f : \mathcal{A} \rightarrow \mathcal{B}$ as in (i), then one has $f^{\sharp\sharp} = f$.

(v) Given a morphism $f : \mathcal{A}/\approx \rightarrow \mathcal{B}$ as in (iii), then one has $f^{\sharp\sharp} = f$.

PROOF. (i) The map $f^\sharp([a]_\approx) = f(a)$ is uniquely determined by f and well-defined:

$$\begin{aligned} [a] = [a'] &\Rightarrow a \approx a' \\ &\Rightarrow f(a) = f(a'), \quad \text{as } \mathcal{B} \models f(\approx), \\ &\Rightarrow f^\sharp([a]) = f^\sharp([b]). \end{aligned}$$

The map $[]_\approx$ is surjective by the definition of \mathcal{A}/\approx ; it is a morphism by the definition of \rightarrow_\approx .

(ii)-(v) Equally simple. ■

7B.5. COROLLARY. Let \mathcal{A}, \mathcal{B} be two type algebras and $f : \mathcal{A} \rightarrow \mathcal{B}$ a morphism. Define

$$\begin{aligned} (i) \quad f(\mathcal{A}) &\triangleq \{b \mid \exists a \in \mathcal{A}. f(a) = b\} \subseteq \mathcal{B}; \\ (ii) \quad a \approx_f a' &\iff f(a) = f(a'), \quad \text{for } a, a' \in \mathcal{A}. \end{aligned}$$

Then

- (i) $f(\mathcal{A})$ is a sub-type algebra of \mathcal{B} .
- (ii) The morphisms $[]_{\approx_f} : \mathcal{A} \rightarrow (\mathcal{A}/\approx_f)$ and $f^\sharp : (\mathcal{A}/\approx_f) \rightarrow \mathcal{B}$ are an ‘epi-mono’ factorization of f : $f = f^\sharp \circ []_f$, with $[]_f$ surjective and f^\sharp injective.

$$\begin{array}{ccc} \mathcal{A} & \xrightarrow{f} & \mathcal{B} \\ & \searrow []_{\approx_f} & \nearrow f^\sharp \\ & \mathcal{A}/\approx_f & \end{array}$$

(iii) $(\mathcal{A}/\approx_f) \cong f(\mathcal{A}) \subseteq \mathcal{B}$.

PROOF. (i) $f(\mathcal{A})$ is closed under \rightarrow_B . Indeed, $f(a) \rightarrow_B f(a') = f(a \rightarrow_A a')$.

(ii) By definition of \approx_f one has $\mathcal{B} \models \approx_f$, hence Proposition 7B.4(i) applies.

(iii) Easy. ■

7B.6. REMARK. (i) In case $\mathcal{A} = \mathbb{T}/\approx$ is a syntactic type algebra and $\mathcal{B} = \langle \mathcal{B}, \rightarrow \rangle$, morphisms $h : \mathbb{T}/\approx \rightarrow \mathcal{B}$ correspond exactly to morphisms $h^\sharp : \mathbb{T} \rightarrow \mathcal{B}$ such that for all $A, B \in \mathbb{T}$

$$A \approx B \Rightarrow h^\sharp(A) = h^\sharp(B).$$

The correspondence is given by $h^\sharp(A) = h([A])$. We call such a map h^\sharp a *syntactic* morphism and often identify h and h^\sharp .

(ii) If $\mathbb{T} = \mathbb{T}^{\mathbb{A}}$ for some set \mathbb{A} of atomic types then h^\sharp is uniquely determined by its restriction $h^\sharp \upharpoonright \mathbb{A}$.

(iii) If moreover $\mathcal{B} = \mathbb{T}'/\approx'$ then $h^\sharp(A) = [B]_{\approx'}$ for some $B \in \mathbb{T}'$. Identifying B with its equivalence class in \approx' , we can write simply $h^\sharp(A) = B$. The first condition in (i) then becomes $A \approx B \Rightarrow h^\sharp(A) \approx' h^\sharp(B)$.

7B.7. PROPOSITION. Let \mathcal{E} be a set of equations over \mathcal{A} .

(i) \mathcal{B} justifies $\mathcal{E} \Leftrightarrow$ there is a morphism $g : \mathcal{A}/\mathcal{E} \rightarrow \mathcal{B}$.

(ii) \mathcal{E}' over \mathcal{B} justifies $\mathcal{E} \Leftrightarrow$ there is a morphism $g : \mathcal{A}/\mathcal{E} \rightarrow \mathcal{B}/\mathcal{E}'$.

PROOF. (i) (\Rightarrow) Suppose \mathcal{B} justifies \mathcal{E} . Then there is a morphism $h : \mathcal{A} \rightarrow \mathcal{B}$ such that $\mathcal{B} \models h(\mathcal{E})$. By Proposition 7B.4(i) there is a morphism $h^\sharp : \mathcal{A}/\mathcal{E} \rightarrow \mathcal{B}$. So take $g = h^\sharp$.

(\Leftarrow) Given a morphism $g : \mathcal{A}/\mathcal{E} \rightarrow \mathcal{B}$. Then $h = g^\sharp$ is such that $\mathcal{B} \models h(\mathcal{E})$, according to Proposition 7B.4(iii).

(ii) By (i). ■

Invertible type algebras and prime elements

7B.8. DEFINITION. (i) A relation \sim on a type algebra $\langle \mathcal{A}, \rightarrow \rangle$ is called *invertible* if for all $a, b, a', b' \in \mathcal{A}$

$$(a \rightarrow b) \sim (a' \rightarrow b') \Rightarrow a \sim a' \& b \sim b'.$$

(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.

REMARK. A syntactic type algebra \mathbb{T}/\approx is invertible if one has

$$(A \rightarrow B) \approx (A' \rightarrow B') \Rightarrow A \approx A' \& B \approx B',$$

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 7A.15(ii) for an example of a non-invertible type algebra. Another useful notion concerning type algebras is that of prime element.

7B.9. DEFINITION. Let \mathcal{A} be a type algebra.

(i) An element $a \in \mathcal{A}$ is *prime* if $a \neq (b \rightarrow c)$ for all $b, c \in \mathcal{A}$.

(ii) We write $||\mathcal{A}|| \triangleq \{a \in \mathcal{A} \mid a \text{ is a prime element}\}$.

7B.10. REMARK. If $\mathcal{A} = \mathbb{T}/\approx$ is a syntactic type algebra, then an element $A \in \mathbb{T}$ is prime if $A \not\approx (B \rightarrow C)$ for all $B, C \in \mathbb{T}$. In this case we also say that A is prime with respect to \approx .

In Exercise 7G.17(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 $h : \mathcal{A} \rightarrow \mathcal{B}$ is not uniquely determined by $h \upharpoonright ||\mathcal{A}||$.

Well-founded type algebras

7B.11. DEFINITION. A type algebra \mathcal{A} is *well-founded* if \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 \rightarrow .

The free type algebra $\mathbb{T}^{\mathbb{A}}$ is well-founded, while e.g. $\mathbb{T}^{\{\alpha, \beta\}}[\alpha = \alpha \rightarrow \beta]$ is not. A well-founded invertible type algebra is isomorphic to a free type algebra.

7B.12. PROPOSITION. 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 type 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 7G.17(ii) it will be shown that this embedding is not necessarily surjective: some elements may not be generated by prime elements.

7B.13. PROPOSITION. Let \mathcal{A}, \mathcal{B} be type algebras and let \sim, \approx be congruence relations on \mathcal{A}, \mathcal{B} , respectively.

- (i) Let $h_0 : \mathcal{A} \rightarrow \mathcal{B}$ be a morphism such that

$$\forall x, y \in \mathcal{A}. x \sim y \Rightarrow h_0(x) \approx h_0(y). \quad (1)$$

Then there exists a morphism $h : \mathcal{A}/\sim \rightarrow \mathcal{B}/\approx$ such that

$$\begin{array}{ccc} \mathcal{A} & \xrightarrow{h_0} & \mathcal{B} \\ []_\sim \downarrow & & \downarrow []_\approx \\ \mathcal{A}/\sim & \xrightarrow{h} & \mathcal{B}/\approx \end{array} \quad \forall x \in \mathcal{A}. h([x]_\sim) = [h_0(x)]_\approx. \quad (2)$$

(ii) Suppose moreover that \mathcal{A} is well-founded and invertible. Let $h : \mathcal{A}/\sim \rightarrow \mathcal{B}/\approx$ be a map. Then h is a morphism iff there exists a morphism $h_0 : \mathcal{A} \rightarrow \mathcal{B}$ such that (2) holds.

PROOF. (i) By (1) the equation (2) is a proper definition of h . One easily verifies that h is a morphism.

(ii) (\Rightarrow) Define for $x, y \in \mathcal{A}$

$$\begin{aligned} h_0(x) &\triangleq b, & \text{if } x \in ||\mathcal{A}||, \text{ for some chosen } b \in h([x]_\approx); \\ h_0(x \rightarrow_A y) &\triangleq h_0(x) \rightarrow_B h_0(y). \end{aligned}$$

Then by well-founded induction one has that $h_0(x)$ is defined for all $x \in \mathcal{A}$ and $h([x]_\sim) = [h_0(x)]_\approx$, using also that \mathcal{A} is invertible. The map h_0 is by definition a morphism.

(\Leftarrow) 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.

7B.14. DEFINITION. (i) A type algebra \mathcal{A} is called *enriched* if there are besides \rightarrow 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 $\mathbb{T} = \mathbb{T}_{C_1, \dots, C_k}^{\mathbb{A}}$ is the collection of types freely generated from \mathbb{A} by \rightarrow and some other constructors C_1, \dots, 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 $\langle \mathbb{T}_{1, +, \times}^{\mathbb{A}}, \rightarrow, +, \times, 1 \rangle$ 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

7B.15. PROPOSITION. If \mathcal{E} is a finite set of 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

$$\begin{aligned} A =_n B \Rightarrow & A \equiv B \vee \\ & [A \equiv A_1 \rightarrow A_2 \& B \equiv B_1 \rightarrow B_2 \& \\ & A_1 =_{m_1} B_1 \& A_2 =_{m_2} B_2, \text{ with } m_1, m_2 < n] \vee \\ & [A =_{m_1} A' \& B =_{m_2} B' \& \\ & ((A' = B') \in \mathcal{E} \vee (B' = A') \in \mathcal{E}) \text{ with } m_1, m_2 < n] \end{aligned}$$

(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 7G.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 be solved incrementally.

7B.16. LEMMA. (i) Let $\mathcal{E}_1, \mathcal{E}_2$ be sets of equations over \mathcal{A} . Then

$$\mathcal{A}/(\mathcal{E}_1 \cup \mathcal{E}_2) \cong (\mathcal{A}/\mathcal{E}_1)/\mathcal{E}_2,$$

where \mathcal{E}_{12} is defined by

$$([A]_{\mathcal{E}_1} = [B]_{\mathcal{E}_1}) \in \mathcal{E}_{12} \Leftrightarrow (A = B) \in \mathcal{E}_2.$$

(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

$$\vdash_{\mathcal{E}_1 \cup \mathcal{E}_2} A = B \Leftrightarrow \vdash_{\mathcal{E}_{12}} [A]_{\mathcal{E}_1} = [B]_{\mathcal{E}_1}.$$

It follows that the map $h: \mathbb{T}/(\mathcal{E}_1 \cup \mathcal{E}_2) \rightarrow (\mathbb{T}/\mathcal{E}_1)/\mathcal{E}_{12}$, 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

$$\begin{aligned} \mathcal{E}_1 &\triangleq \{A = B \mid A \approx B\}, \\ \mathcal{E}_2 &\triangleq \{A = B \mid ([A]_\approx = [B]_\approx) \in \mathcal{E}\}. \end{aligned}$$

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 $\mathcal{A} = \mathbb{T}^\mathbb{A}/\approx$ is a syntactic type algebra and A denotes as usual an element of $\mathbb{T}^\mathbb{A}$, then in the formula $A \in \mathcal{A}$ the A stands for $[A]_\approx$. If we consider this \mathcal{A} modulo \mathcal{E} , then $A =_{\mathcal{E}} B$ is equivalent to $A =_{\mathcal{E}'} B$, with \mathcal{E}' as in Lemma 7B.16(ii).

7C. Recursive types via simultaneous recursion

In this section we construct type algebras containing elements satisfying recursive equations, like $a = a \rightarrow b$ or $c = d \rightarrow c$. 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 fixed point 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 fixed point 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 7A.10. Other congruence relations will be defined semantically, using possibly infinite trees (co-inductively), as is done in Section 7E.

Adding indeterminates

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 \rightarrow . Since this definition of $\mathcal{A}(\vec{X})$ is somewhat syntactic we assume, using Prop. 7A.16, that \mathcal{A} is a syntactic type algebra.

Often we will take for \mathcal{A} the free syntactic type algebra $\mathbb{T}^{\mathbb{A}}$ over an arbitrary non-empty set of atomic types \mathbb{A} .

7C.1. DEFINITION. Let $\mathcal{A} = \mathbb{T}^{\mathbb{A}}/\approx$ be a syntactic type algebra. Let $\vec{X} = X_1, \dots, X_n$ ($n \geq 0$) 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

$$\mathcal{A}(\vec{X}) \triangleq \mathbb{T}^{\mathbb{A} \cup \{\vec{X}\}}/\approx.$$

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 7A.16 this construction can be applied to arbitrary type algebras as well.

NOTATION. $A(\vec{X})$ ranges over arbitrary elements of $\mathcal{A}(\vec{X})$.

7C.2. PROPOSITION. $\mathcal{A} \hookrightarrow \mathcal{A}(\vec{X})$.

PROOF. Immediate. ■

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} with 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 $=_{\mathcal{E}}$.

7C.3. DEFINITION. Let \mathcal{A} be a type algebra and $\mathcal{E} = \mathcal{E}(\vec{X})$ be a set of equations over $\mathcal{A}(\vec{X})$. Write $\mathcal{A}[\mathcal{E}] \triangleq \mathcal{A}(\vec{X})/\mathcal{E}$

Satisfying existential equations

Now we want to state for existential statements like $\exists X.a = b \rightarrow X$, with $a, b \in \mathcal{A}$ when they hold in a type structure. We say that $\exists X.a = b \rightarrow X$ holds in \mathcal{A} , notation

$$\mathcal{A} \models \exists X.a = b \rightarrow X,$$

if for some $c \in \mathcal{A}$ one has $a = b \rightarrow c$.

The following definitions are stated for sets of equations \mathcal{E} but apply to a single equation $a = b$ as well, by considering it as a singleton $\{a = b\}$.

7C.4. DEFINITION. Let \mathcal{A} be a type algebra and $\mathcal{E} = \mathcal{E}(\vec{X})$ a set of equations over $\mathcal{A}(\vec{X})$.

- (i) We say \mathcal{A} *solves* \mathcal{E} (or \mathcal{A} *satisfies* $\exists \vec{X}.\mathcal{E}$ or $\exists \vec{X}.\mathcal{E}$ *holds* in \mathcal{A}), notation $\mathcal{A} \models \exists \vec{X}.\mathcal{E}$, if there is a morphism $h:\mathcal{A}(\vec{X}) \rightarrow \mathcal{A}$ such that $h(a) = a$, for all $a \in \mathcal{A}$ and $\mathcal{A} \models h(\mathcal{E}(\vec{X}))$.
- (ii) For any h satisfying (i), the sequence $\langle h(X_1), \dots, h(X_n) \rangle \in \mathcal{A}$ is called a *solution* in \mathcal{A} of $\mathcal{E}(\vec{X})$.

7C.5. REMARK. (i) Note that $\mathcal{A} \models \exists \vec{X}.\mathcal{E}$ iff $\mathcal{A} \models \mathcal{E}[\vec{X}] = \vec{a}$ for some $\vec{a} \in \mathcal{A}$. Indeed, choose $a_i = h(X_i)$ as definition of the \vec{a} or of the morphism h .

(ii) If \mathcal{A} solves $\mathcal{E}(\vec{X})$, then $\mathcal{A}(\vec{X})$ justifies $\mathcal{E}(\vec{X})$, but not conversely. During justification one may reinterpret the constants, via a morphism.

REMARK. (i) The set of equations $\mathcal{E}(\vec{X})$ over $\mathcal{A}(\vec{X})$ is interpreted as a problem of finding the appropriate \vec{X} in \mathcal{A} . This is similar to stating that the polynomial $x^2 - 3 \in \mathbb{R}[x]$ has root $\sqrt{3} \in \mathbb{R}$.

(ii) In the previous Definition we tacitly changed the indeterminates \vec{X} in a bound variable: by $\exists \vec{X}.\mathcal{E}$ or $\exists \vec{X}.\mathcal{E}(\vec{X})$ we intend $\exists \vec{x}.\mathcal{E}(\vec{x})$. We will allow this ‘abus de language’: \vec{X} as bound variables, since it is clear what we mean.

(iii) If $\vec{X} = \emptyset$, then

$$\mathcal{A} \models \exists \vec{X}.\mathcal{E} \Leftrightarrow \mathcal{A} \models \mathcal{E}.$$

EXAMPLE. There exists a type algebra \mathcal{A} such that

$$\mathcal{A} \models \exists X.(X \rightarrow X) = (X \rightarrow X \rightarrow X). \quad (1)$$

Take $\mathcal{A} = \mathbb{T}[\mathcal{E}]$, with $\mathcal{E} = \{X \rightarrow X = X \rightarrow X \rightarrow X\}$, with solution

$$\mathbf{X} = [X]_{\{X \rightarrow X = X \rightarrow X \rightarrow X\}}.$$

7C.6. REMARK. Over $\mathbb{T}^{\{a\}}(X, Y)$ let $\mathcal{R} \triangleq \{X = a \rightarrow X, Y = a \rightarrow a \rightarrow Y\}$. Then $[X]_{\mathcal{R}}, [Y]_{\mathcal{R}} \in \mathbb{T}[\mathcal{R}]$ is a solution of $\exists X Y. \mathcal{R}$. Note that also $[X]_{\mathcal{R}}, [X]_{\mathcal{R}}$ is such a solution and intuitively $[X]_{\mathcal{R}} \neq [Y]_{\mathcal{R}}$, as we will see later more precisely. Hence solutions are not unique.

Simultaneous recursions

In general \mathbb{T}/\mathcal{E} is not invertible. Take e.g. in Example 7A.15(ii) $\mathbb{A}_{\infty} = \{\alpha, \infty\}$. Then in $\mathbb{T}^{\mathbb{A}_{\infty}}/\mathcal{E}_{\infty}$ one has $\alpha \rightarrow \infty = \infty \rightarrow \infty$, but $\alpha \neq \infty$.

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 7A.15 (ii).

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, see [van Wijngaarden \[1981\]](#).

7C.7. DEFINITION. Let \mathcal{A} be a type algebra.

(i) A *simultaneous recursion (sr)* over \mathcal{A} with *indeterminates* $\vec{X} = \{X_1, \dots, X_n\}$ is a finite set $\mathcal{R} = \mathcal{R}(\vec{X})$ of equations over $\mathcal{A}(\vec{X})$ of the form

$$\left. \begin{array}{rcl} X_1 & = & A_1(\vec{X}) \\ & \dots & \\ X_n & = & A_n(\vec{X}) \end{array} \right\} \quad \mathcal{R}$$

where all indeterminates X_1, \dots, X_n are different.

(ii) The *domain* of \mathcal{R} , notation $\text{Dom}(\mathcal{R})$, consists of the set $\{\vec{X}\}$.

(iii) If $\text{Dom}(\mathcal{R}) = \vec{X}$, then \mathcal{R} is said to be an sr over $\mathcal{A}(\vec{X})$.

(iv) The equational theory on $\mathcal{A}(\vec{X})$ axiomatized by \mathcal{R} is denoted by (\mathcal{R}) .

It is useful to consider restricted forms of simultaneous recursion.

7C.8. DEFINITION (Simultaneous recursion). (i) A sr $\mathcal{R}(\vec{X})$ is *proper* if

$$(X_i = X_j) \in \mathcal{R} \Rightarrow i < j.$$

(ii) A sr $\mathcal{R}(\vec{X})$ is *simple* if no equation $X_i = X_j$ occurs in \mathcal{R} .

Note that a simple sr is proper. The definition of proper is intended to rule out circular definitions like $X = X$ or $X = Y, Y = X$. Proper sr are convenient from the Term Rewriting System (TRS) point of view introduced in Section 8C: the reduction relation will be SN. We always can make an sr proper, as will be shown in Proposition 7C.18

EXAMPLE. For example let $\alpha, \beta \in \mathbb{A}$. Then

$$\begin{aligned} X_1 &= \alpha \rightarrow X_2 \\ X_2 &= \beta \rightarrow X_1 \end{aligned}$$

is an sr with indeterminates $\{X_1, X_2\}$ over $\mathbb{T}^{\mathbb{A}}$.

Intuitively it is clear that in this example one has $X_1 =_{\mathcal{R}} \alpha \rightarrow \beta \rightarrow X_1$, but $X_1 \neq_{\mathcal{R}} X_2$. To show this the following is convenient.

An sr can be considered as a TRS, see Klop [1992] or Terese [2003]. The reduction relation is denoted by $\Rightarrow_{\mathcal{R}}^*$; we will later encounter its converse $\Rightarrow_{\mathcal{R}}^{*-1}$ as another useful reduction relation.

7C.9. DEFINITION. Let \mathcal{R} on \mathcal{A} be given.

(i) Define on $\mathcal{A}(\vec{X})$ the *\mathcal{R} -reduction relation*, notation $\Rightarrow_{\mathcal{R}}^*$, induced by the notion of reduction

$$\left. \begin{array}{rcl} X_1 & \Rightarrow_{\mathcal{R}} & A_1(\vec{X}) \\ & \dots & \\ X_n & \Rightarrow_{\mathcal{R}} & A_n(\vec{X}) \end{array} \right\} \quad (\Rightarrow_{\mathcal{R}})$$

So $\Rightarrow_{\mathcal{R}}^*$ is the least reflexive, transitive, and compatible relation on $\mathcal{A}(\vec{X})$ extending $\Rightarrow_{\mathcal{R}}$.

(ii) The relation $=_{\mathcal{R}}$ is the least compatible equivalence relation extending $\Rightarrow_{\mathcal{R}}^*$

(iii) We denote the resulting TRS by $\text{TRS}(\mathcal{R}) = (\mathcal{A}(\vec{X}), \Rightarrow_{\mathcal{R}})$.

It is important to note that the \vec{X} are not variables in the TRS sense: if $a(X) \Rightarrow_{\mathcal{R}}^* b(X)$, then not necessarily $a(c) \Rightarrow_{\mathcal{R}}^* b(c)$. Rewriting in $\text{TRS}(\mathcal{R})$ is between closed expressions.

In general $\Rightarrow_{\mathcal{R}}$ is not normalizing. For example for \mathcal{R} as above one has

$$X_1 \Rightarrow_{\mathcal{R}} (\alpha \rightarrow X_2) \Rightarrow_{\mathcal{R}} (\alpha \rightarrow \beta \rightarrow X_1) \Rightarrow_{\mathcal{R}} \dots$$

Remember that a rewriting system $\langle X, \Rightarrow \rangle$ is *Church-Rosser* (CR) if

$$\forall a, b, c \in X. [a \Rightarrow^* b \& a \Rightarrow^* c \Rightarrow \exists d \in X. [b \Rightarrow^* d \& c \Rightarrow^* d]],$$

where \Rightarrow^* is the transitive reflexive closure of \Rightarrow .

7C.10. PROPOSITION (*Church-Rosser Theorem for \Rightarrow_μ*). *Given an sr \mathcal{R} over \mathcal{A} . Then*

- (i) *For $a, b \in \mathcal{A}$ one has $\mathcal{R} \vdash a = b \Leftrightarrow a =_{\mathcal{R}} b$.*
- (ii) *$\Rightarrow_{\mathcal{R}}$ on $\mathcal{A}(\vec{X})$ is CR.*
- (iii) *Therefore $a =_{\mathcal{R}} b$ iff a, b have a common $\Rightarrow_{\mathcal{R}}^*$ reduct.*

PROOF. (i) See e.g. Terese [2003], Exercise 2.4.3.

- (ii) Easy, the ‘redexes’ are all disjoint.
- (iii) By (ii). ■

So in the example above one has $X_1 \neq_{\mathcal{R}} X_2$ and $X_1 =_{\mathcal{R}} (\alpha \rightarrow \beta \rightarrow X_1)$.

An important property of an sr is that they do not identify elements of \mathcal{A} .

7C.11. LEMMA. *Let $\mathcal{R}(\vec{X})$ be an sr over a type algebra \mathcal{A} . Then for all $a, b \in \mathcal{A}$ we have*

$$a \neq b \Rightarrow a \neq_{\mathcal{R}} b.$$

PROOF. By Proposition 7C.10(ii). ■

Lemma 7C.11 is no longer true, in general, if we start work with a set of equations \mathcal{E} instead of an sr $\mathcal{R}(\vec{X})$. Take e.g. $\mathcal{E} = \{a = a \rightarrow b, b = (a \rightarrow b) \rightarrow b\}$. 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).

Another application of the properties of $\text{TRS}(\mathcal{R})$ is the invertibility of an sr.

7C.12. PROPOSITION. *Let \mathcal{R} be an sr over \mathbb{T} . Then $=_{\mathcal{R}}$ is invertible.*

PROOF. Suppose $A \rightarrow B =_{\mathcal{R}} A' \rightarrow B'$, in order to show $A =_{\mathcal{R}} A' \& B =_{\mathcal{R}} B'$. By the CR property for $\Rightarrow_{\mathcal{R}}^*$ the types $A \rightarrow B$ and $A' \rightarrow B'$ have a common $\Rightarrow_{\mathcal{R}}^*$ -reduct which must be of the form $C \rightarrow D$. Then $A =_{\mathcal{R}} C =_{\mathcal{R}} A'$ and $B =_{\mathcal{R}} D =_{\mathcal{R}} B'$. ■

Note that the images of \mathcal{A} and the $[X_i]$ in $\mathcal{A}(\vec{X})/=_\mathcal{R}$ 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 $[X] = [a]$.

7C.13. DEFINITION. (i) Let $\mathcal{R} = \mathcal{R}(\vec{X})$ be a simultaneous recursion in \vec{X} over a type algebra \mathcal{A} (i.e. a special set of equations over $\mathcal{A}(\vec{X})$). As in Definition 7C.3 write

$$\mathcal{A}[\mathcal{R}] \triangleq \mathcal{A}(\vec{X})/\mathcal{R}$$

- (ii) For X one of the \vec{X} , write $\mathbf{X} \triangleq [X]_{\mathcal{R}}$.
- (iii) We say that $\mathcal{A}[\mathcal{R}]$ is obtained by *adjunction* of the elements \vec{X} to \mathcal{A} .

The method of adjunction then allows us to define recursive types incrementally, according to Lemma 7B.16(i).

REMARK. (i) By Proposition 7C.12 the type algebra $\mathbb{T}[\mathcal{R}]$ is invertible.

- (ii) In general $\mathcal{A}[\mathcal{E}]$ is not invertible, see Example 7A.15(ii).

(iii) Let the indeterminates of \mathcal{R}_1 and \mathcal{R}_2 be disjoint, then $\mathcal{R}_1 \cup \mathcal{R}_2$ is an sr again. By Lemma 7B.16 (i) $\mathcal{A}[\mathcal{R}_1 \cup \mathcal{R}_2] = \mathcal{A}[\mathcal{R}_1][\mathcal{R}_2]$. Recursive types can therefore be defined incrementally.

7C.14. THEOREM. *Let \mathcal{A} be a type algebra and \mathcal{R} an sr over \mathcal{A} . Then*

- (i) $\varphi: \mathcal{A} \hookrightarrow \mathcal{A}[\mathcal{R}]$, where $\varphi(a) = [a]_{\mathcal{R}}$.
- (ii) $\mathcal{A}[\mathcal{R}]$ is generated from (the image under φ of) \mathcal{A} and the $[X_i]_{\mathcal{R}}$.
- (iii) $\mathcal{A}[\mathcal{R}] \models \exists \vec{X}. \mathcal{R}$ and the $\mathbf{X}_1, \dots, \mathbf{X}_n$ form a solution of \mathcal{R} in $\mathcal{A}[\mathcal{R}]$.

PROOF. (i) The canonical map φ is an injective morphism by Lemma 7C.11.

- (ii) Clearly $\mathcal{A}[\mathcal{R}]$ is generated by the \mathbf{X}_i and the $[a]_{\mathcal{R}}$, with $a \in \mathcal{A}$.
- (iii) $\mathcal{A}[\mathcal{R}] \models \exists \vec{X}. \mathcal{R}$ by Lemma 7A.14(ii). ■

In Theorem 7C.14(iii) we stated that the $\mathbf{X}_1, \dots, \mathbf{X}_n$ form a solution of \mathcal{R} . In fact they form a solution of \mathcal{R} translated to $\mathcal{A}[\mathcal{R}](\vec{X})$. Moreover, this translation is trivial, due to the injection $\varphi: \mathcal{A} \hookrightarrow \mathcal{A}[\mathcal{R}]$.

Folding and unfolding

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. Like with a notion of reduction, this operation can also be applied to subterms. 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, possibly on subexpressions of a and b .

7C.15. EXAMPLE. (i) The sr $\mathcal{R}_0 = \{X_0 = A \rightarrow X_0\}$, where $A \in \mathbb{T}$ is a type, specifies a type X_0 which is such that

$$X_0 =_{\mathcal{R}_0} A \rightarrow X_0 =_{\mathcal{R}_0} A \rightarrow A \rightarrow X_0 \dots$$

i.e. $X_0 =_{\mathcal{R}_0} A^n \rightarrow X_0$ for any n . This represents the behavior of a function which can take an arbitrary number of arguments of type A .

(ii) The sr $\mathcal{R}_1 \triangleq \{X_1 = A \rightarrow A \rightarrow X_1\}$ is similar to \mathcal{R}_0 but not all equations modulo \mathcal{R}_0 hold modulo \mathcal{R}_1 . For instance $X_1 \neq_{\mathcal{R}_1} A \rightarrow X_1$ (i.e. we cannot derive $X_1 = A \rightarrow X_1$ from the derivation rules of Definition 7A.10(i)).

REMARK. Note that $=_{\mathcal{R}}$ is the minimal congruence with respect to \rightarrow 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 $\mathcal{R} = \{X = A \rightarrow X, Y = A \rightarrow Y\}$. Then we have $X \neq_{\mathcal{R}} Y$ since we cannot prove $X = Y$ using only the rules of Definition 7A.10(i). These types will instead be identified in the tree equivalence introduced in Section 7E.

We will often consider only proper simultaneous recursions. In order to do this, it is useful to transform an sr into an ‘equivalent’ one. We introduce two notions of equivalence for simultaneous recursion.

7C.16. DEFINITION. 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

$$\forall a, b \in \mathcal{A}[\vec{X}]. a =_{\mathcal{R}} b \Leftrightarrow a =_{\mathcal{R}'} b.$$

REMARK. (i) It is easy to see that \mathcal{R} and \mathcal{R}' over the same \vec{X} are logically equivalent if

$$\mathcal{R} \vdash \mathcal{R}' \text{ and } \mathcal{R}' \vdash \mathcal{R}.$$

- (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.

7C.17. DEFINITION. Let \mathcal{A} be a type algebra. Define $\mathcal{A}_\bullet := \mathcal{A}(\vec{\bullet})$, where $\vec{\bullet}$ are some indeterminates with special names different from all X_i . These $\vec{\bullet}$ are treated as new elements that are said to have been *added* to \mathcal{A} . Indeed, $\mathcal{A} \hookrightarrow \mathcal{A}_\bullet$.

7C.18. PROPOSITION. (i) Every proper sr $\mathcal{R}(\vec{X})$ over \mathcal{A} is equivalent to a simple $\mathcal{R}'(\vec{X}')$, where \vec{X}' is a subset of \vec{X} .

- (ii) Let \mathcal{R} be an sr over \mathcal{A} . Then there is a proper \mathcal{R}' over \mathcal{A}_\bullet such that

$$\mathcal{A}[\mathcal{R}] \cong \mathcal{A}_\bullet[\mathcal{R}'].$$

PROOF. (i) If \mathcal{R} is not simple, then $\mathcal{R} = \mathcal{R}_1 \cup \{X_i = X_j\}$, with $i < j$. Now define

$$\mathcal{R}^-(X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n),$$

by $\mathcal{R}^- \triangleq \mathcal{R}_1[X_i := X_j]$. 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 simple \mathcal{R}' is obtained. One easily proves that

$$\mathcal{A}[\vec{X}]/\mathcal{R} \cong \mathcal{A}[X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n]/\mathcal{R}'$$

as follows. Note that if $\mathcal{R} = \{X_k = A_k(\vec{X}) \mid 1 \leq k \leq n\}$, then

$$\mathcal{R}^- = \{X_k = A_k(\vec{X})[X_i := X_j] \mid k \neq i\}.$$

Define

$$\begin{aligned} g^\natural : \quad & \mathcal{A}(\vec{X}) \rightarrow \mathcal{A}[X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n] \\ h^\natural : \quad & \mathcal{A}[X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n] \rightarrow \mathcal{A}(\vec{X}) \end{aligned}$$

by

$$\begin{aligned} g^\natural(A) &\triangleq A[X_i := X_j], \quad \text{for } A \in \mathcal{A}[\vec{X}], \\ h^\natural(A) &\triangleq A, \quad \text{for } A \in \mathcal{A}[X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n] \end{aligned}$$

and show

$$\begin{aligned} g^\natural(X_k) &= g^\natural(A_k(\vec{X})), \quad \text{for } 1 \leq k \leq n, \\ h^\natural(X_k) &= h^\natural((A_k(\vec{X}))[X_i := X_j]), \quad \text{for } k \neq j. \end{aligned}$$

Then g^\natural, h^\natural induce the required isomorphism g and its inverse h .

(ii) First remove each $X_j = X_i$ from \mathcal{R} and put the X_j in $\vec{\bullet}$. The equations $X_i = X_j$ with $i > j$ are treated in the same way as $X_j = X_i$ in (i). The proof that indeed $\mathcal{A}[\mathcal{R}] \cong \mathcal{A}_\bullet[\mathcal{R}']$ is very easy. Now g^\natural and h^\natural are in fact identities. ■

7C.19. LEMMA. Let $\mathcal{R}(\vec{X})$ be a proper sr over \mathcal{A} . Then all its indeterminates X are such that either $X =_{\mathcal{R}} a$ where $a \in \mathcal{A}$ or $X =_{\mathcal{R}} (b \rightarrow c)$ for some $b, c \in \mathcal{A}[\vec{X}]$.

PROOF. Easy. ■

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.

7C.20. LEMMA. *Let $\mathcal{R}(\vec{X})$ be a proper sr over \mathbb{T}^A . Then*

$$\begin{aligned} ||\mathbb{T}[\mathcal{R}]|| &= \{[\alpha] \mid \alpha \in A\}; \\ [\alpha] &\subseteq \{\alpha\} \cup \{\vec{X}\}, \end{aligned}$$

i.e. $[\alpha]$ consists of α and some of the \vec{X} .

PROOF. The elements of $\mathbb{T}[\mathcal{R}]$ are generated from A and the \vec{X} . Now note that by Lemma 7C.19 (i) an indeterminate X either is such that $X =_{\mathcal{R}} A \rightarrow B$ for some $A, B \in \mathbb{T}^{A \cup \vec{X}}$ (and then $[X]$ is not prime) or $X =_{\mathcal{R}} \alpha$ for some atomic type α . Moreover, by Proposition 7C.10 it follows that 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 $\alpha \in A$. ■

For a proper sr \mathcal{R} we can write, for instance $||\mathbb{T}[\mathcal{R}]|| = A$ choosing α as the representative of $[\alpha]$.

Justifying sets of equations by an sr

Remember that \mathcal{B} justifies a set of equations \mathcal{E} over \mathcal{A} if there is a morphism $h: \mathcal{A} \rightarrow \mathcal{B}$ such that $\mathcal{B} \models h(\mathcal{E})$ and that a set \mathcal{E}' over \mathcal{B} justifies \mathcal{E} over \mathcal{A} iff \mathcal{B}/\mathcal{E}' justifies \mathcal{E} . A particular case is that an sr \mathcal{R} over $\mathcal{B}(\vec{X})$ justifies \mathcal{E} over \mathcal{A} iff $\mathcal{B}[\mathcal{R}]$ justifies \mathcal{E} . Proposition 7B.7 stated that \mathcal{B} justifies a set of equations \mathcal{E} iff there is a morphism $h: \mathcal{A}/\mathcal{E} \rightarrow \mathcal{B}$. Indeed, all the equations in \mathcal{E} become valid after interpreting the elements of \mathcal{A} in the right way in \mathcal{B} .

In Chapter 8 it will be shown that in the right context the notion of justifying is decidable. But decidability only makes sense if \mathcal{B} is given in an effective ‘finitely presented’ way.

7C.21. PROPOSITION. *Let \mathcal{A}, \mathcal{B} be type algebras and let \mathcal{E} be a set of equations over \mathcal{A}*

(i) *Let \mathcal{E}' be a set of equations over \mathcal{B} . Then*

$$\mathcal{E}' \text{ justifies } \mathcal{E} \Leftrightarrow \exists g. g : \mathcal{A}/\mathcal{E} \rightarrow \mathcal{B}/\mathcal{E}'.$$

(ii) *Let \mathcal{R} be an sr over $\mathcal{B}(\vec{X})$. Then*

$$\mathcal{R} \text{ justifies } \mathcal{E} \Leftrightarrow \exists g. g : \mathcal{A}/\mathcal{E} \rightarrow \mathcal{B}[\mathcal{R}].$$

PROOF. (i), (ii). By Proposition 7B.7(ii). ■

EXAMPLE. Let $\mathcal{E} \triangleq \{\alpha \rightarrow \beta = \alpha \rightarrow \alpha \rightarrow \beta\}$. Then $\mathcal{R} = \{X = \alpha \rightarrow X\}$ justifies \mathcal{E} over $\mathbb{T}^{\{\alpha, \beta\}}$ as we have the morphism

$$h : \mathbb{T}^{\{\alpha, \beta\}}/\mathcal{E} \rightarrow \mathbb{T}^{\{\alpha\}}[\mathcal{R}]$$

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 morphism).

7C.22. PROPOSITION. *Let \mathcal{A}, \mathcal{B} be type algebras. Suppose that \mathcal{A} is well-founded and invertible. Let \mathcal{E} be a system of equations over \mathcal{A} and $\mathcal{R}(\vec{X})$ be an sr over \mathcal{B} . Then*

$$\mathcal{R} \text{ justifies } \mathcal{E} \Leftrightarrow \exists h: \mathcal{A} \rightarrow \mathcal{B}(\vec{X}) \forall a, b \in \mathcal{A}. [a =_{\mathcal{E}} b \Rightarrow h(a) =_{\mathcal{R}} h(b)]. \quad (*)$$

PROOF. By Corollary 7B.7(ii) and Proposition 7B.13. ■

As a free type algebra is well-founded and invertible, (*) holds for all \mathbb{T}^A .

Closed type algebras

A last general notion concerning type algebras is the following.

7C.23. DEFINITION. Let \mathcal{A} be a type algebra.

- (i) \mathcal{A} is *closed* if every sr \mathcal{R} over \mathcal{A} can be solved in \mathcal{A} , cf. Definition 7C.4.
- (ii) \mathcal{A} is *uniquely closed*, if every proper sr \mathcal{R} over \mathcal{A} has a unique solution in \mathcal{A} .

7C.24. REMARK. There are type algebras that are closed but not uniquely so. For instance let $\mathcal{A} = \mathbb{T}^{\{a,b\}}/\mathcal{E}$ with $\mathcal{E} \triangleq \{a = a \rightarrow a, b = b \rightarrow b, b = a \rightarrow b, b = b \rightarrow a\}$. Then \mathcal{A} is closed, but not uniquely so. A simple uniquely closed type algebra will be given in section 7E.

From Proposition 7B.15 we know that $=_{\mathcal{R}}$ is decidable for any (finite) \mathcal{R} over $\mathbb{T}^{\mathcal{A}}(\vec{X})$. In Chapter 8 we will prove some other properties of $\mathbb{T}[\mathcal{R}]$, in particular that it is decidable whether an sr \mathcal{R} justifies a set \mathcal{E} of equations.

7D. Recursive types via μ -abstraction

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 7C.23.

7D.1. DEFINITION ($\dot{\mu}$ -types). Let $\mathbb{A} = \mathbb{A}_{\infty}$ be the infinite set of type atoms considered as type variables for the purpose of binding and substitution. The set $\mathbb{T}_{\dot{\mu}}^{\mathbb{A}}$ is defined by the following ‘simplified syntax’, omitting parentheses. The ‘ \cdot ’ on top of the $\dot{\mu}$ indicates that we do not (yet) consider the types modulo α -conversion (renaming of bound variables).

$$\boxed{\mathbb{T}_{\dot{\mu}}^{\mathbb{A}} ::= \mathbb{A} \mid \mathbb{T}_{\dot{\mu}}^{\mathbb{A}} \rightarrow \mathbb{T}_{\dot{\mu}}^{\mathbb{A}} \mid \dot{\mu} \mathbb{A} \mathbb{T}_{\dot{\mu}}^{\mathbb{A}}}$$

Often we write $\mathbb{T}_{\dot{\mu}}$ for $\mathbb{T}_{\dot{\mu}}^{\mathbb{A}}$, leaving \mathbb{A} implicit.

The subset of $\mathbb{T}_{\dot{\mu}}^{\mathbb{A}}$ containing only types without occurrences of the $\dot{\mu}$ operator coincides with the set $\mathbb{T}^{\mathbb{A}}$ of simple types.

NOTATION. (i) Similarly to the case with repeated λ -abstraction we write

$$\dot{\mu}\alpha_1 \cdots \alpha_n.A \triangleq (\dot{\mu}\alpha_1(\dot{\mu}\alpha_2 \cdots (\dot{\mu}\alpha_n(A))..)).$$

(ii) We assume that \rightarrow takes precedence over $\dot{\mu}$, so that e.g. the type $\dot{\mu}\alpha.A \rightarrow B$ should be parsed as $\dot{\mu}\alpha.(A \rightarrow B)$.

According to the intuitive semantics of recursive types, a type expression of the form $\dot{\mu}\alpha.A$ should be regarded as the solution for α in the equation $\alpha = A$, and is then equivalent to the type expression $A[\alpha := \dot{\mu}\alpha.A]$.

Some bureaucracy for renaming and substitution

The reader is advised to skip this subsection at first reading: goto 7D.22.

In $\dot{\mu}\beta.A$ the operator $\dot{\mu}$ binds the variable β . We write $\text{FV}(A)$ for the set of variables occurring free in A , and $\text{BV}(A)$ for the set of variables occurring bound in A .

7D.2. NOTATION. (i) The sets of variables occurring as *bound variables* or as *free variables* in the type $A \in \mathbb{T}_\mu^\mathbb{A}$, notation $\text{BV}(A)$, $\text{FV}(A)$, respectively, are defined inductively as follows.

A	$\text{FV}(A)$	$\text{BV}(A)$
α	$\{\alpha\}$	\emptyset
$A \rightarrow B$	$\text{FV}(A) \cup \text{FV}(B)$	$\text{BV}(A) \cup \text{BV}(B)$
$\dot{\mu}\alpha.A_1$	$\text{FV}(A_1) - \{\alpha\}$	$\text{BV}(A_1) \cup \{\alpha\}$

(ii) If $\beta \notin \text{FV}(A) \cup \text{BV}(A)$ we write $\beta \notin A$.

Bound variables can be renamed by α -conversion: $\dot{\mu}\beta.A \equiv_\alpha \dot{\mu}\gamma.A[\beta := \gamma]$, provided that $\gamma \notin A$. From 7D.22 on we will consider types in $\mathbb{T}_\mu^\mathbb{A}$ modulo α -convertibility, obtaining $\mathbb{T}_\mu^\mathbb{A}$. Towards this goal, items 7D.1–7D.21 are a preparation.

We will often assume that the names of bound and free variables in types are distinct: this can be easily obtained by a renaming of bound variables. Unlike for λ -terms we like to be explicit about this so-called α -conversion. We will distinguish between ‘naive’ substitution $[\beta := A]_\alpha$ in which innocent free variables may be captured and ordinary ‘smart’ substitution $[\beta := A]$ that avoids this.

7D.3. DEFINITION. Let $A, B \in \mathbb{T}_\mu$.

(i) The *naive substitution* operator, notation $A[\beta := B]_\alpha$, is defined as follows.

A	$A[\beta := B]_\alpha$
α	α ,
β	B
$A_1 \rightarrow A_2$	$A_1[\beta := B]_\alpha \rightarrow A_2[\beta := B]_\alpha$
$\dot{\mu}\beta.A$	$\dot{\mu}\beta.A$
$\dot{\mu}\alpha.A$	$\dot{\mu}\alpha.(A[\beta := B]_\alpha)$, if $\alpha \neq \beta$,

The notation $A[\beta := B]_\alpha$ comes from [Endrullis, Grabmayer, Klop, and van Oostrom \[2010\]](#).

(ii) Ordinary ‘smart’ substitution, notation $A[\beta := B]$, that avoids capturing free variables (‘dynamic binding’) is defined by Curry as follows, see [B\[1984\]](#), Definition C.1.

A	$A[\beta := B]$
α	α if $\alpha \neq \beta$
β	B
$A_1 \rightarrow A_2$	$A_1[\beta := B] \rightarrow A_2[\beta := B]$,
$\dot{\mu}\beta.A$	$\dot{\mu}\beta.A$
$\dot{\mu}\alpha.A_1$	$\dot{\mu}\alpha'.(A_1[\alpha := \alpha'][\beta := B])$, if $\alpha \neq \beta$, where $\alpha' = \alpha$ if $\beta \notin \text{FV}(A_1)$ or $\alpha \notin \text{FV}(B)$, else α' is the first variable in the sequence of type variables $\alpha_0, \alpha_1, \alpha_2, \dots$ that is not in $\text{FV}(A_1) \cup \text{FV}(B)$.

7D.4. LEMMA. (i) If $\text{BV}(A) \cap \text{FV}(A) = \emptyset$, then

$$A[\beta := B] \equiv A[\beta := B]_\alpha.$$

(ii) If $\beta \notin \text{FV}(A)$, then

$$A[\beta := B] \equiv A.$$

PROOF. (i) By induction on the structure of A . The interesting case is $A \equiv \dot{\mu}\gamma.C$, with $\gamma \not\equiv \beta$. Then

$$\begin{aligned} (\dot{\mu}\gamma.C)[\beta := B] &\equiv \dot{\mu}\gamma'.C[\gamma := \gamma'][\beta := B], && \text{by Definition 7D.3(ii),} \\ &\equiv \dot{\mu}\gamma.C[\beta := B], && \text{since } \gamma \notin \text{FV}(B), \\ &\equiv \dot{\mu}\gamma.C[\beta := B]_{\not\equiv}, && \text{by the induction hypothesis,} \\ &\equiv (\dot{\mu}\gamma.C)[\beta := B]_{\not\equiv}, && \text{by Definition 7D.3(i).} \end{aligned}$$

(ii) Similarly, the interesting case being $A \equiv \dot{\mu}\gamma.C$, with $\gamma \not\equiv \beta$. Then

$$\begin{aligned} (\dot{\mu}\gamma.C)[\beta := B] &\equiv \dot{\mu}\gamma'.C[\gamma := \gamma'][\beta := B], && \text{by Definition 7D.3(ii),} \\ &\equiv \dot{\mu}\gamma.C[\beta := B], && \text{as } \beta \notin \text{FV}(A) \& \beta \not\equiv \gamma \text{ so } \beta \notin \text{FV}(C), \\ &\equiv \dot{\mu}\gamma.C, && \text{by the induction hypothesis. } \blacksquare \end{aligned}$$

7D.5. DEFINITION (α -conversion). On $\mathbb{T}_{\dot{\mu}}$ we define the notion of α -reduction and α -conversion via the contraction rule

$$\dot{\mu}\alpha.A \mapsto_{\alpha} \dot{\mu}\alpha'.A[\alpha := \alpha'], \text{ provided } \alpha' \notin \text{FV}(A).$$

The relation \Rightarrow_{α} is the least compatible relation containing \mapsto_{α} . The relation \Rightarrow_{α}^* is the transitive reflexive closure of \Rightarrow_{α} . Finally \equiv_{α} the least congruence containing \mapsto_{α} .

For example $\dot{\mu}\alpha.\alpha \rightarrow \alpha \equiv_{\alpha} \dot{\mu}\beta.\beta \rightarrow \beta$. Also $\dot{\mu}\alpha.(\alpha \rightarrow \dot{\mu}\beta.\beta) \equiv_{\alpha} \dot{\mu}\beta.(\beta \rightarrow \dot{\mu}\beta.\beta)$.

7D.6. LEMMA. (i) If $A \Rightarrow_{\alpha} B$, then $B \Rightarrow_{\alpha} A$.

(ii) $A \equiv_{\alpha} B$ implies $A \Rightarrow_{\alpha}^* B \& B \Rightarrow_{\alpha}^* A$.

PROOF. (i) If $\dot{\mu}\alpha.A \Rightarrow_{\alpha} \dot{\mu}\alpha'.A[\alpha := \alpha']$, then $\alpha \notin \text{FV}(A[\alpha := \alpha'])$, so that also

$$\dot{\mu}\alpha'.A[\alpha := \alpha'] \Rightarrow_{\alpha} \dot{\mu}\alpha.A[\alpha := \alpha'][\alpha' := \alpha] \equiv \dot{\mu}\alpha.A.$$

(ii) By (i). \blacksquare

7D.7. DEFINITION. (i) Define on $\mathbb{T}_{\dot{\mu}}$ a notion of $\dot{\mu}$ -reduction via the contraction rule $\mapsto_{\dot{\mu}}$

$$\dot{\mu}\alpha.A \mapsto_{\dot{\mu}} A[\alpha := \dot{\mu}\alpha.A].$$

(ii) A $\dot{\mu}$ -redex is of the form $\dot{\mu}\alpha.A$ and its contraction is $A[\alpha := \dot{\mu}\alpha.A]$.

(iii) The relation $\Rightarrow_{\dot{\mu}} \subseteq \mathbb{T}_{\dot{\mu}} \times \mathbb{T}_{\dot{\mu}}$ is the compatible closure of $\mapsto_{\dot{\mu}}$. That is

$$\begin{aligned} A \Rightarrow_{\dot{\mu}} A' &\Rightarrow A \rightarrow B \Rightarrow_{\dot{\mu}} A' \rightarrow B \\ A \Rightarrow_{\dot{\mu}} A' &\Rightarrow B \rightarrow A \Rightarrow_{\dot{\mu}} B \rightarrow A' \\ A \Rightarrow_{\dot{\mu}} A' &\Rightarrow \dot{\mu}\alpha.A \Rightarrow_{\dot{\mu}} \dot{\mu}\alpha.A'. \end{aligned}$$

(iv) As usual $\Rightarrow_{\dot{\mu}}^n$ denotes reduction in n steps.

(v) The relation $\Rightarrow_{\dot{\mu}}^*$ is the reflexive and transitive closure of $\Rightarrow_{\dot{\mu}}$, i.e. 0 or more reduction steps.

(vi) The relation $\dot{\mu}$ -conversion, notation $=_{\dot{\mu}}$, is the conversion relation generated by $\dot{\mu}$ -reduction, i.e. the least congruence relation containing $\mapsto_{\dot{\mu}}$.

7D.8. LEMMA. Let $A, A', B \in \mathbb{T}_{\dot{\mu}}$. Then

- (i) $A \Rightarrow_{\dot{\mu}} A' \Rightarrow A[\alpha := B] \Rightarrow_{\dot{\mu}} A'[\alpha := B]$.
- (ii) $A \Rightarrow_{\dot{\mu}} A' \Rightarrow B[\alpha := A] \Rightarrow_{\dot{\mu}} B[\alpha := A']$.
- (iii) Both (i) and (ii) hold with $\Rightarrow_{\dot{\mu}}$ replaced by $=_{\dot{\mu}}$.

PROOF. (i) By induction on the derivation of $A \Rightarrow_{\dot{\mu}} A'$.

(ii) By induction on the structure of B .

(iii) By (i) and (ii). \blacksquare

7D.9. LEMMA. Let $A, A', B \in \mathbb{T}_{\dot{\mu}}$. Then

- (i) $A \Rightarrow_{\alpha} A' \Rightarrow A[\alpha := B] \Rightarrow_{\alpha}^* A'[\alpha := B]$.

- (ii) $A \Rightarrow_{\alpha} A' \Rightarrow B[\alpha := A] \Rightarrow_{\alpha}^* B[\alpha := A']$.
- (iii) Both (i) and (ii) hold with \Rightarrow_{α} and \Rightarrow_{α}^* replaced by \equiv_{α} .

PROOF. (i) By induction on the derivation of $A \Rightarrow_{\alpha} A'$.

- (ii) By induction on the structure of B .
- (iii) By (i) and (ii). ■

7D.10. LEMMA (Substitution Lemma). *Let $A \in \mathbb{T}_{\mu}$. Then for all $B, C \in \mathbb{T}_{\mu}$ and type variables β, γ with $\beta \not\equiv \gamma$ and $\beta \notin \text{FV}(C)$ one has*

$$A[\beta := B][\gamma := C] \equiv_{\alpha} A[\gamma := C][\beta := B[\gamma := C]]. \quad (1)$$

Writing $D^{\gamma} = D[\gamma := C]$ this is

$$(A[\beta := B])^{\gamma} \equiv_{\alpha} A^{\gamma}[\beta := B^{\gamma}].$$

PROOF. By induction on the number of symbols in A . The interesting cases are $A \equiv \alpha$ and $A \equiv \dot{\mu}\alpha.A_1$.

Case $A \equiv \alpha$. If $\alpha \notin \{\beta, \gamma\}$, then the result is trivial: $\alpha \equiv_{\alpha} \alpha$. If $\alpha \equiv \beta$, then (1) boils down to $B^{\gamma} \equiv_{\alpha} B^{\gamma}$. If $\alpha \equiv \gamma$, then because of the assumption $\beta \notin \text{FV}(C)$ equation (1) becomes $C \equiv_{\alpha} C$.

Case $A \equiv \dot{\mu}\alpha.A_1$. We must show

$$((\dot{\mu}\alpha.A_1)[\beta := B])^{\gamma} \equiv_{\alpha} (\dot{\mu}\alpha.A_1)^{\gamma}[\beta := B^{\gamma}]. \quad (2)$$

By Lemma 7D.9(iii) it suffices to show (2) for an ‘ α -variant’ $\dot{\mu}\alpha_2.A_2 \equiv_{\alpha} \dot{\mu}\alpha.A_1$. Take α_2 such that $\alpha_2 \notin \{\beta, \gamma\} \cup \text{FV}(B) \cup \text{FV}(C)$. Then by the freshness of α_2

$$\begin{aligned} (\dot{\mu}\alpha_2.A_2)[\beta := B]^{\gamma} &\equiv \dot{\mu}\alpha_2.(A_2[\beta := B])^{\gamma}, && \text{by applying twice Definition 7D.3(ii),} \\ &\equiv_{\alpha} \dot{\mu}\alpha_2.(A_2^{\gamma})[\beta := B^{\gamma}], && \text{by the IH,} \\ &\equiv (\dot{\mu}\alpha_2.A_2)^{\gamma}[\beta := B^{\gamma}]. \blacksquare \end{aligned}$$

7D.11. LEMMA. *Let $A, B \in \mathbb{T}_{\mu}$ and $\alpha' \notin \text{FV}(A)$. Then*

$$A[\alpha := \alpha'][\alpha' := B] \equiv_{\alpha} A[\alpha := B].$$

PROOF. By induction on the structure of A . We treat the case $A \equiv \dot{\mu}\beta.A_1$. Like in the proof of the Substitution Lemma we may assume that $\beta \notin \{\alpha, \alpha'\} \cup \text{FV}(B)$. Then as before

$$\begin{aligned} (\dot{\mu}\beta.A_1)[\alpha := \alpha'][\alpha' := B] &\equiv \dot{\mu}\beta.(A_1[\alpha := \alpha'][\alpha' := B]) \\ &\equiv_{\alpha} \dot{\mu}\beta.A_1[\alpha := B], && \text{by the IH,} \\ &\equiv (\dot{\mu}\beta.A_1)[\alpha := B]. \end{aligned}$$

Avoiding α -conversion on $\dot{\mu}$ -types

As $\dot{\mu}$ -types are built up from the variable binding $\dot{\mu}$ -operator one has to take care that there will be no clash of variables during a $\dot{\mu}$ -reduction. This is similar to the situation with untyped λ -terms and β -reduction. An important difference between those λ -terms and $\dot{\mu}$ -types is the possibility to choose a correct α -variant of the $\dot{\mu}$ -types that remains correct after arbitrary reduction steps. In the (un)typed lambda calculus this is not possible. The term

$$(\lambda x.xx)(\lambda yz.yz) \in \Lambda^{\emptyset}$$

has its bound variables maximally different. But after some reduction steps a clash of variables occurs.

$$\begin{aligned} (\lambda x.xx)(\lambda yz.yz) &\rightarrow_{\beta} (\lambda yz.yz)(\lambda yz.yz) \\ &\rightarrow_{\beta} \lambda z.(\lambda yz.yz)z \\ &\not\rightarrow_{\beta} (\lambda z.(\lambda z.zz)) \equiv \lambda zz.zz, \end{aligned}$$

which should have been $\lambda z z'.z z'$. In order to avoid clashes in the untyped lambda calculus one therefore should be constantly alert and apply α -conversion whenever needed. This example can be modified so that a typable λ -term results, do Exercise 7G.2.

In B[1984] this necessary hygienic discipline is somewhat swept under the carpet via the so-called ‘variable convention’: choosing the names of bound variables maximally fresh. The belief that this is sound came from the calculus with nameless binders in de Bruijn [1972], which is implemented for converting lambda terms in the proof-checker Agda. The variable convention has been cleverly implemented in the Nominal package for Isabelle/HOL, see Urban and Tasson [2005], with as goal to reason formally about binding. They show that the variable convention is compatible with structural inductions over terms with binders. However, Urban, Berghofer, and Norrish [2007] also show that care needs to be taken in rule inductions over inductively defined predicates: if a bound variable occurs free in a conclusion of a rule, then the variable convention can easily lead to unsound reasoning.

7D.12. DEFINITION. Let $A \in \mathbb{T}_{\dot{\mu}}$.

- (i) The type A is called *safe* if for all subtypes $\dot{\mu}\beta.B$ of A the one step reduction

$$\dot{\mu}\beta.B \Rightarrow_{\dot{\mu}} B[\beta := \dot{\mu}\beta.B]_{\not\alpha}$$

does not result in binding a free variable of $\dot{\mu}\beta.B$ in $B[\beta := \dot{\mu}\beta.B]_{\not\alpha}$. One also could state that for all subtypes $\dot{\mu}\beta.B$ of A

$$B[\beta := \dot{\mu}\beta.B] \equiv B[\beta := \dot{\mu}\beta.B]_{\not\alpha}.$$

- (ii) We say that A is *forever safe* if

$$A \Rightarrow_{\dot{\mu}}^* B \Rightarrow B \text{ is safe},$$

for all $B \in \mathbb{T}_{\dot{\mu}}$.

7D.13. EXAMPLE. (i) $\dot{\mu}\alpha.\alpha \rightarrow \beta$ is safe.

(ii) $\dot{\mu}\alpha.(\beta \rightarrow \dot{\mu}\beta.(\alpha \rightarrow \beta))$ is not safe; ‘contracting $\dot{\mu}\alpha$ ’ leads to a clash.

(iii) $\dot{\mu}\alpha.(\beta \rightarrow \dot{\mu}\gamma.(\alpha \rightarrow \gamma))$ is safe.

Like with (un)typed λ -terms one has that every $A \in \mathbb{T}_{\dot{\mu}}$ has an α -variant that is safe, by renaming bound variables by fresh bound variables. We will show something better: the existence of an α -variant that is and remains safe. In general being safe does not warrant remaining safe after $\dot{\mu}$ -reduction.

7D.14. DEFINITION. Let $P \subseteq \mathbb{T}_{\dot{\mu}}$ be a predicate on types.

- (i) P is called *α -flexible* if

$$\forall A \in \mathbb{T}_{\dot{\mu}} \exists A' \in \mathbb{T}_{\dot{\mu}}. [A \equiv_{\alpha} A' \& P(A')].$$

- (ii) P is called *$\dot{\mu}$ -invariant* if for all $A, B \in \mathbb{T}_{\dot{\mu}}$

$$A \Rightarrow_{\dot{\mu}} B \& P(A) \Rightarrow P(B).$$

- (iii) P is called *protective* if for all $A \in \mathbb{T}_{\dot{\mu}}$

$$P(A) \Rightarrow A \text{ is safe}.$$

- (iv) P is called *α -avoiding* if it is α -flexible, protective, and $\dot{\mu}$ -invariant.

Note that if a predicate $P \subseteq \mathbb{T}_\mu$ is protective and invariant, then $P(A)$ implies that A is forever safe. Therefore it will be nice to have an α -avoiding predicate: then every type A can be replaced by an α -equivalent one that is forever safe.

7D.15. REMARK. We try several predicates P_i on \mathbb{T}_μ .

- (i) $P_1(A) \iff \text{FV}(A) \cap \text{BV}(A) = \emptyset$.
- (ii) $P_2(A) \iff P_1(B)$ for all subtypes B of A .
- (iii) $P_3(A) \iff P_1(A)$ and different occurrences $\dot{\mu}$ in A
bind different type variables.
- (iv) $P_4(A) \iff A$ is safe.
- (v) $P_5(A) \iff A$ is forever safe.

Unfortunately none of these predicates is α -avoiding.

7D.16. LEMMA. (i) P_1 is α -flexible, $\dot{\mu}$ -invariant, but not protective.

- (ii) P_2 is α -flexible, protective, but not $\dot{\mu}$ -invariant.
- (iii) P_3 is α -flexible, protective, but not $\dot{\mu}$ -invariant.
- (iv) P_4 is α -flexible, protective, but not $\dot{\mu}$ -invariant.
- (v) P_5 is protective and $\dot{\mu}$ -invariant; it is α -avoiding iff there exists an α -avoiding predicate.

PROOF. For the proof we use a convenient notation of van Oostrom: the binary operation \hookrightarrow is denoted by an implicit ‘multiplication’, with association to the right. E.g. $\dot{\mu}\alpha.\alpha\beta\alpha$ denotes $\dot{\mu}\alpha.\alpha \rightarrow (\beta \rightarrow \alpha)$.

(i) Define $A_1 \equiv \dot{\mu}\alpha.B[\alpha]$, with $B[\alpha] \equiv \dot{\mu}\beta.\alpha\beta(\dot{\mu}\alpha.\beta)$. Then $P_1(A_1)$; but by contracting $B[\alpha]$ a capture of $\alpha \in \text{FV}(B[\alpha])$ occurs:

$$B[\alpha] \Rightarrow_{\dot{\mu}} \alpha B[\alpha](\dot{\mu}\alpha.B[\alpha]),$$

Hence A_1 is not safe, and therefore P_1 is not protective.

(ii) Clearly P_2 is α -flexible and protective. Define $A_2 \equiv \dot{\mu}\alpha\beta.\alpha\beta$. Then $P_2(A_2)$; but

$$A_2 \Rightarrow_{\dot{\mu}} \dot{\mu}\beta.A_2\beta \equiv A'_2.$$

Now $P_2(A'_2)$ fails as its subterm $A_2\beta$ contains β both as free and as bound variable. Therefore P_2 is not $\dot{\mu}$ -invariant.

(iii) Again P_3 is α -flexible and protective. Define $A_3 = \dot{\mu}\alpha.\alpha\alpha$. Then $P_3(A_3)$, but

$$A_3 \Rightarrow_{\dot{\mu}} A_3 A_3$$

and $P(A_3 A_3)$ does not hold. Therefore P_3 is not $\dot{\mu}$ -invariant.

(iv) [van Oostrom.] Also P_4 is α -flexible and protective. In order to show it is not $\dot{\mu}$ -invariant, define $A_4 \equiv \dot{\mu}\alpha\beta.\alpha(\dot{\mu}\gamma.\beta(\dot{\mu}\alpha.\gamma))$. Then $P_4(A_4)$ and

$$\begin{aligned} A_4 &\Rightarrow_{\dot{\mu}} \dot{\mu}\alpha\dot{\mu}\beta.\alpha(\beta(\dot{\mu}\alpha\gamma.\beta(\dot{\mu}\alpha.\gamma))) && \equiv \dot{\mu}\alpha.B[\alpha]. \\ &\Rightarrow_{\dot{\mu}} \dot{\mu}\alpha.\alpha(B[\alpha](\dot{\mu}\alpha\gamma.B[\alpha](\dot{\mu}\alpha.\gamma))), && \text{a variable capture.} \end{aligned}$$

Therefore $\dot{\mu}\alpha\beta.B[\alpha]$ is not safe, and hence P_4 not $\dot{\mu}$ -invariant.

(v) The predicate P_5 is safe and $\dot{\mu}$ -invariant by definition. If there is an α -avoiding predicate P , then P is protective, hence $P \subseteq P_5$, but also P is α -flexible and therefore for all $A \in \mathbb{T}_\mu$ there is an α -variant $A' \in P \subseteq P_5$. The converse is trivial. ■

After these examples we show that there does exist an α -avoiding predicate on \mathbb{T}_μ . The result is due to van Oostrom and was inspired by [Melliès \[1996\]](#).

7D.17. DEFINITION. Let $A \in \mathbb{T}_\mu$.

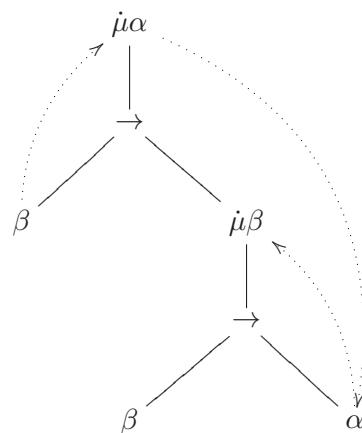
- (i) An occurrence of $\dot{\mu}\alpha$ in A *binds* an occurrence of the variable α if $\dot{\mu}\alpha$ occurs as $\dot{\mu}\alpha.B$ with $\alpha \in \text{FV}(B)$, and α occurs in the scope of $\dot{\mu}\alpha$.
- (ii) An occurrence of $\dot{\mu}\alpha$ in A *captures* an occurrence of the variable $\beta \not\equiv \alpha$ if β freely occurs in the scope of $\dot{\mu}\alpha$.
- (iii) Define for subterm occurrences o_1, o_2 of A of the form $\dot{\mu}\alpha$ or α , with α any type variable, the relation

$$o_1 \prec o_2 \Leftrightarrow o_1 \text{ is captured by } o_2 \text{ or } o_1 \text{ binds } o_2.$$

- (iv) An occurrence α is *self-capturing* if

$$\alpha \prec^* \dot{\mu}\alpha.$$

7D.18. EXAMPLE. Let $A \equiv \dot{\mu}\alpha.\beta \rightarrow (\dot{\mu}\beta.\beta \rightarrow \alpha)$. The following diagram

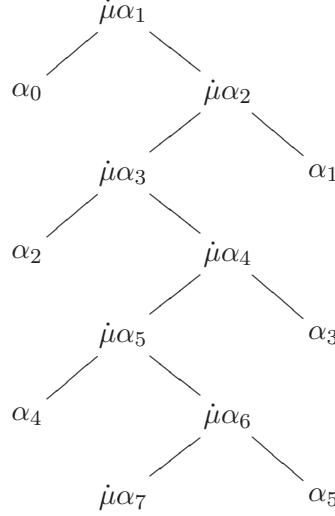


shows that the ‘highest’ occurrence in A of the variable β is self-capturing, but not the other occurrence.

7D.19. EXAMPLE. In the following diagram, where we write $\dot{\mu}\alpha.A \rightarrow B$ as



one has $\alpha_0 \prec^* \dot{\mu}\alpha_k$ for all $k \geq 1$.



Convince yourself that if there is self-capturing, e.g. $\alpha_4 = \alpha_0$, then a naive $\dot{\mu}$ -reduction without changing names of bound variables may result in a variable clash. The diagram shows a typical ‘topology’ causing the phenomenon of self-capturing.

7D.20. PROPOSITION (van Oostrom [2007]). Define $P \subseteq \mathbb{T}_{\dot{\mu}}$ by

$$P(A) \stackrel{\Delta}{\iff} \text{for no variable } \beta \text{ there is a self-capturing occurrence in } A.$$

Then P is α -avoiding.

PROOF. We will show that P is (i) α -flexible, (ii) protecting, and (iii) $\dot{\mu}$ -invariant.

(i) Renaming all binders in A in such a way that they are fresh and pairwise distinct we obtain $A' \equiv_{\alpha} A$ with $P(A')$.

(ii) We have to show that

$$P(A) \Rightarrow A \text{ is safe},$$

for which it suffices that for all subtypes $\dot{\mu}\beta.B$ of A

$$P(\dot{\mu}\beta.B) \Rightarrow B[\beta := \dot{\mu}\beta.B] = B[\beta := \dot{\mu}\beta.B]_{\not\alpha}. \quad (1)$$

It is not clear how to prove this directly. One can prove a stronger statement. Let $Q(\beta, B, C)$ be the statement

$$Q(\beta, B, C) \stackrel{\Delta}{\iff} [\beta \in \text{FV}(B) \Rightarrow \forall \gamma \in \text{BV}(B). [\beta \prec \dot{\mu}\gamma \Rightarrow \gamma \notin \text{FV}(C)]].$$

This states about β , B , and C that if β occurs in $\text{FV}(B)$, which occurrence is captured by an occurrence of $\dot{\mu}\gamma$ in B , then the variable γ does not occur in $\text{FV}(C)$. The stronger statement to be proved is

$$\forall C. [Q(\beta, B, C) \Rightarrow B[\beta := C] = B[\beta := C]_{\not\alpha}]. \quad (2)$$

Now we show (2) by induction on the structure of B . For B a variable or of the form $B = B_1 \rightarrow B_2$ this is easy, noting that

$$B' \subseteq B \Rightarrow Q(\beta, B, C) \Rightarrow Q(\beta, B', C).$$

If $B = \dot{\mu}\gamma.B'$, then we must show

$$Q(\beta, \dot{\mu}\gamma.B', C) \Rightarrow (\dot{\mu}\gamma.B')[\beta := C] = (\dot{\mu}\gamma.B')[\beta := C]_{\not\alpha}.$$

If $\beta \notin \text{FV}(B')$, then this is trivial. Otherwise $\beta \in \text{FV}(B')$, hence $\beta \prec \dot{\mu}\gamma$. By the assumption $Q(\beta, \dot{\mu}\gamma.B', C)$ it follows that $\gamma \notin \text{FV}(C)$. Then

$$\begin{aligned} (\dot{\mu}\gamma.B')[\beta := C] &= \dot{\mu}\gamma.(B'[\beta := C]), && \text{since } \dot{\mu}\gamma \text{ cannot bind a variable in } C, \\ &= \dot{\mu}\gamma.(B'[\beta := C]_{\not\alpha}), && \text{by the induction hypothesis,} \\ &= (\dot{\mu}\gamma.B')[\beta := C]_{\not\alpha}. && \text{by definition of } [\beta := C]_{\not\alpha}. \end{aligned}$$

This establishes (2).

Now for $(\dot{\mu}\beta.B) \subseteq A$ we have

$$P(\dot{\mu}\beta.B) \Rightarrow Q(\beta, B, \dot{\mu}\beta.B). \quad (3)$$

Indeed, if $\beta \in \text{FV}(B)$ and γ occurs both in $\text{BV}(B)$ and in $\text{FV}(\dot{\mu}\beta.B)$, two different occurrences, then $\beta \prec \dot{\mu}\gamma$ is impossible since it implies $\gamma \prec \dot{\mu}\beta \prec \beta \prec \dot{\mu}\gamma$, i.e. γ is self-capturing, contradicting $P(\dot{\mu}\beta.B)$.

Now by (2) and (3) we obtain (1).

(iii) Suppose $A \Rightarrow_{\dot{\mu}} A'$ and $P(A)$, in order to show $P(A')$. Let

$$A = C[\dot{\mu}\beta.B] \Rightarrow_{\dot{\mu}} C[B[\beta := \dot{\mu}\beta.B]],$$

where $C[]$ is some context (with possible binding effects, like $C[] = \dot{\mu}\epsilon.[]$). It suffices to show that if γ is a self-capturing occurrence of a variable in A' , then also A contains a selfcapturing occurrence of γ . Let o'_1, o'_2 be occurrences of an α or $\dot{\mu}\alpha$ in A' . Let o_1, o_2 be the unique occurrences in A such that o'_i is the residual, see Terese [2003], of o_i (for $1 \leq i \leq 2$). The symbols ($\alpha, \dot{\mu}\alpha$, or \rightarrow) written at occurrences o_i and o'_i are the same. We claim that they satisfy

$$o'_1 \prec o'_2 \Rightarrow o_1 \prec^* o_2. \quad (4)$$

Let $\{o'_1, o'_2\} = \{\gamma, \dot{\mu}\delta\}$ and suppose $o'_1 \prec o'_2$.

Case 1. The occurrences o'_1, o'_2 in A' both occur in the same ‘component’, i.e. both in $C[]$, in the ‘body’ B , or in (a copy of) $B[\beta := \dot{\mu}\beta.B]$, respectively. Then also $o_1 \prec o_2$.

Case 2. o'_1 occurs on $C[]$, o'_2 in B . Then $\gamma \neq \beta$, because otherwise $\dot{\mu}\beta.B$ would have been substituted for it. Hence $o_1 \prec o_2$.

Case 3. o'_1 occurs on $C[]$, o'_2 in (a copy of) $\dot{\mu}\beta.B$. Now $\gamma \neq \beta$ holds, because otherwise $o'_1 \not\prec o'_2$. If $\gamma = \delta$, then $o'_1 = \dot{\mu}\gamma, o'_2 = \gamma$ and we also have $o_1 \prec o_2$. If $\gamma \neq \delta$, then $o'_1 = \gamma, o'_2 = \dot{\mu}\delta$ and we have $o_1 \prec o_2$.

Case 4. o'_1 occurs on B , o'_2 in (a copy of) $\dot{\mu}\beta.B$. Then again $\gamma \neq \beta$, for the same reason as in Case 3. If $\gamma = \delta$, then $o'_1 = \dot{\mu}\gamma, o'_2 = \gamma$ and we have $o_1 \prec o_2$. If $\gamma \neq \delta$, then $o'_1 = \gamma, o'_2 = \dot{\mu}\delta$ and we have $o_1 \prec^* o_2$ (this is the only place where we use \prec^*), because

$$\gamma \prec \dot{\mu}\beta \prec \beta \prec \dot{\mu}\delta.$$

This establishes (4). Now suppose $\neg P(A')$. Then A' contains a self-capturing occurrence $\gamma \prec^* \dot{\mu}\gamma$. By (4) and transitivity it follows that also A contains such an occurrence. Therefore indeed $\neg P(A)$. ■

At the end of the day for every $A \in \text{TR}_{\dot{\mu}}$ an α -variant that can be forever $\dot{\mu}$ -reduced naively can be easily found.

7D.21. COROLLARY. Let $A \in \mathbb{T}_{\mu}$. Suppose that all binders in A are pairwise distinct and different from the free variables in A . Then A is forever safe.

PROOF. As in (i) in the proof of the Proposition the condition implies $P(A)$. Therefore A is forever safe. ■

Identifying types equal up to renaming

7D.22. DEFINITION. (i) The set of recursive types, notation \mathbb{T}_μ , is defined as

$$\mathbb{T}_\mu^A \triangleq \mathbb{T}_{\dot{\mu}}^A / \equiv_\alpha,$$

i.e. $\mathbb{T}_\mu^A \triangleq \{[A]_\alpha \mid A \in \mathbb{T}_{\dot{\mu}}^A\}$, where $[A]_\alpha \triangleq \{B \in \mathbb{T}_{\dot{\mu}}^A \mid B \equiv_\alpha A\}$.

(ii) On \mathbb{T}_μ^A one defines

$$\begin{aligned} [A]_\alpha \rightarrow [B]_\alpha &\triangleq [A \rightarrow B]_\alpha, \\ \mu\beta.[A]_\alpha &\triangleq [\dot{\mu}\beta.A]_\alpha. \end{aligned}$$

(iii) For elements of \mathbb{T}_μ^A instead of $[A]_\alpha$ we simply write A .

Note that on \mathbb{T}_μ^A the operations \rightarrow and $\mu\beta$ are well-defined, i.e. independent of the choice of representative in the α -equivalence classes.

Definition 7D.3(ii) of substitution for $\mathbb{T}_{\dot{\mu}}$ can immediately be lifted to \mathbb{T}_μ as follows.

7D.23. DEFINITION. For $A, B \in \mathbb{T}_\mu^A$ substitution is defined as follows.

A	$A[\beta := B]$
α	α if $\alpha \neq \beta$
β	B
$A_1 \rightarrow A_2$	$A_1[\beta := B] \rightarrow A_2[\beta := B]$,
$\mu\alpha.A_1$	$\mu\alpha.(A_1[\beta := B])$, if $\alpha \neq \beta$

Using the variable convention we choose $\alpha \neq \beta$, $\alpha \notin B$ for the case $A \equiv \mu\alpha.A_1$.

7D.24. REMARK. The above definition of substitution is correct, i.e. it is independent of the choice of representatives, and the variable convention makes sense, as follows immediately from Lemma 7D.9(iii).

7D.25. LEMMA (Substitution Lemma). Let $A \in \mathbb{T}_\mu$. Then for all $B, C \in \mathbb{T}_\mu$ and type variables β, γ with $\beta \not\equiv \gamma$ and $\beta \notin \text{FV}(C)$ one has

$$A[\beta := B][\gamma := C] \equiv A[\gamma := C][\beta := B[\gamma := C]]. \quad (1)$$

Writing $D^\gamma = D[\gamma := C]$ this is

$$(A[\beta := B])^\gamma \equiv A^\gamma[\beta := B^\gamma].$$

PROOF. Directly from Lemma 7D.10. ■

7D.26. DEFINITION (Weak equivalence of recursive types). (i) In Fig. 21 the equational theory (μ) is defined. The variable \mathcal{H} stands for a set of equations between two elements of \mathbb{T}_μ . 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) from $\mathcal{H} = \emptyset$.

(ii) We will often identify \mathbb{T}_μ with the (syntactic) type algebra $\mathbb{T}_\mu / =_\mu$, if there is little danger of confusion. If we want to refer to \mathbb{T}_μ not modulo $=_\mu$, we write $\mathbb{T}_\mu^!$.

(ident)	$\mathcal{H} \vdash A = A$
(symm)	$\frac{\mathcal{H} \vdash A = B}{\mathcal{H} \vdash B = A}$
(trans)	$\frac{\mathcal{H} \vdash A = B \quad \mathcal{H} \vdash B = C}{\mathcal{H} \vdash A = C}$
(axiom)	$\mathcal{H} \vdash A = B,$ if $(A = B) \in \mathcal{H},$
(μ -eq)	$\mathcal{H} \vdash \mu\alpha.A = A[\alpha := \mu\alpha.A]$
(μ -cong)	$\frac{\mathcal{H} \vdash A = A'}{\mathcal{H} \vdash \mu\alpha.A = \mu\alpha.A'},$ if α not free in $\mathcal{H},$
(\rightarrow -cong)	$\frac{\mathcal{H} \vdash A = A' \quad \mathcal{H} \vdash B = B'}{\mathcal{H} \vdash A \rightarrow B = A' \rightarrow B'}$

FIGURE 21. The system of equational logic (μ)

REMARK. (i) Rule (μ -eq) is well-defined, as follows immediately from Lemma 7D.9(iii).

(ii) 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.

(iii) Two types in \mathbb{T}_μ are weakly equivalent iff they can be transformed one into the other by a finite number of applications of folding and unfolding, possibly on subexpressions.

7D.27. EXAMPLE. Let $B \triangleq \mu\beta.A \rightarrow \beta$, with $\beta \notin \text{FV}(A)$. Then we have

$$B =_\mu A \rightarrow B =_\mu A \rightarrow A \rightarrow B =_\mu \dots$$

Now let $B' \triangleq \mu\beta.(A \rightarrow A \rightarrow \beta)$. Then

$$B' =_\mu A \rightarrow A \rightarrow B' =_\mu \dots$$

It is easy to see that $B \neq_\mu B'$ (see also Section 8B). Indeed B and B' are the μ -types solving, respectively, the type equations in \mathcal{R}_0 and \mathcal{R}_1 of Example 7C.15.

7D.28. DEFINITION. (i) Define on \mathbb{T}_μ a notion of *μ -reduction* via the contraction rule \mapsto_μ

$$\mu\alpha.A \mapsto_\mu A[\alpha := \mu\alpha.A].$$

(ii) A *μ -redex* is of the form $\mu\alpha.A$ and its *contraction* is $A[\alpha := \mu\alpha.A]$.

(iii) The relation $\Rightarrow_\mu \subseteq \mathbb{T}_\mu \times \mathbb{T}_\mu$ is the compatible closure of \mapsto_μ . That is

$$\begin{aligned} A \Rightarrow_\mu A' &\Rightarrow A \rightarrow B \Rightarrow_\mu A' \rightarrow B; \\ A \Rightarrow_\mu A' &\Rightarrow B \rightarrow A \Rightarrow_\mu B \rightarrow A'; \\ A \Rightarrow_\mu A' &\Rightarrow \mu\alpha.A \Rightarrow_\mu \mu\alpha.A'. \end{aligned}$$

(iv) As usual \Rightarrow_μ^n denotes reduction in n steps.

(v) The relation \Rightarrow_μ^* is the reflexive and transitive closure of \Rightarrow_μ , i.e. 0 or more reduction steps.

(vi) Finally $=_\mu$ *μ -conversion* is the conversion relation generated by μ -reduction, i.e. the least congruence relation containing \mapsto_μ . We will see in Proposition 7D.29(i) that this overloading is justified, as it is the same as the relation in Definition 7D.26.

This turns \mathbb{T}_μ into a *Combinatory Reduction System* (CRS), notation $\langle \mathbb{T}_\mu, \Rightarrow_\mu \rangle$, which generates $=_\mu$ as its convertibility relation. For an introduction to term rewriting systems (TRS) and CRS see e.g. Klop [1992], Klop, van Oostrom, and van Raamsdonk [1993] and Terese [2003].

7D.29. PROPOSITION. (i) *Convertibility corresponding to \Rightarrow_μ is $=_\mu$.*

(ii) *The reduction relation \Rightarrow_μ on \mathbb{T}_μ is CR.*

PROOF. (i) By definition of $=_\mu$; do exercise 8D.2.

(ii) The notion of reduction \Rightarrow_μ induces on μ -types an orthogonal combinatory reduction system, see Terese [2003], Ch. 11. By Theorem 11.6.19 of that book it is CR. ■

Most μ -types can be unfolded to a form with main constructor (a type variable or \rightarrow) at top level, unless they belong to the following pathological subset of \mathbb{T}_μ .

7D.30. DEFINITION. Let $A \in \mathbb{T}_\mu$.

(i) A is called *circular* iff $A \equiv \mu\beta_1 \cdots \beta_n.\beta_i$ for some $1 \leq i \leq n$.

(ii) The most typical circular type is $\bullet \triangleq \mu\alpha.\alpha$. Following Ariola and Klop [1996] the symbol ‘ \bullet ’ is called ‘the blackhole’.

(iii) A is called *contractive in α* if $\mu\alpha.A$ is not circular.

Circular types correspond to non-proper sr.

7D.31. REMARK. In Definition 7D.30(i) we mean in fact that an element of \mathbb{T}_μ is circular if it is of the form $A = [\dot{\mu}\beta_1 \cdots \beta_n.\beta_i]_\alpha$. Suppose that also $A = [B]_\alpha$, for some $B \in \mathbb{T}_\mu$. Then by Lemma 7D.6(ii) one has $\dot{\mu}\beta_1 \cdots \beta_n.\beta_i \Rightarrow_\mu^* B$. It follows that $B \equiv \dot{\mu}\gamma_1 \cdots \gamma_m.\gamma_j$, with $1 \leq j \leq m$. Therefore each representative of A is of the form $\dot{\mu}\beta_1 \cdots \beta_n.\beta_i$. In particular it is decidable whether A is circular.

7D.32. LEMMA. *Let A be a circular type. Then $A =_\mu \mu\alpha.\alpha \equiv \bullet$. Therefore, the circular types form an equivalence class in \mathbb{T}_μ .*

PROOF. Let $A \equiv \mu\beta_1 \cdots \beta_n.\beta_i$. Then

$$\begin{aligned} A &\Rightarrow_\mu^{n-i} \mu\beta_1 \cdots \beta_i.\alpha_i, \\ &\Rightarrow_\mu^{i-1} \mu\beta_i.\beta_i. \end{aligned}$$

Therefore $A \Rightarrow_\mu^{n-1}$. ■

7D.33. LEMMA. *Let $A \in \mathbb{T}_\mu$. One has exactly one of the following cases.*

$$\begin{aligned} A &\mapsto_\mu^* \beta; \\ A &\mapsto_\mu^* (B \rightarrow C); \\ A &\text{ is circular.} \end{aligned}$$

Moreover, β and $B \rightarrow C$ are unique.

PROOF. Write $A \equiv \mu\beta_1 \cdots \beta_n.A_1$, with $A_1 \not\equiv \mu\gamma.A_2$. If $A_1 \in \{\beta_1 \cdots \beta_n\}$, then A is circular. If $A_1 \equiv \beta \notin \{\beta_1 \cdots \beta_n\}$, then $A \mapsto_\mu^n \beta$. If $A_1 \equiv B_1 \rightarrow C_1$, then there exist unique B, C such that $A \mapsto_\mu^n (B \rightarrow C)$. ■

7D.34. DEFINITION. Let $A \in \mathbb{T}_\mu$.

- (i) [Grabmayer [2007]] The *lead symbol* of A , notation $\text{ls}(A)$, is defined as

$$\begin{aligned}\text{ls}(A) &\triangleq \alpha, & \text{if } A \mapsto_{\mu}^* \alpha; \\ \text{ls}(A) &\triangleq \rightarrow, & \text{if } A \mapsto_{\mu}^* (B \rightarrow C); \\ \text{ls}(A) &\triangleq \bullet, & \text{if } A \text{ is circular.}\end{aligned}$$

- (ii) The *principal reduced form* A' of A is defined as follows.

$$\begin{aligned}A' &\equiv \alpha, & \text{if } A \mapsto_{\mu}^* \alpha; \\ A' &\equiv B \rightarrow C, & \text{if } A \mapsto_{\mu}^* (B \rightarrow C); \\ A' &\equiv \bullet, & \text{if } A \text{ is circular.}\end{aligned}$$

(iii) In untyped λ -calculus terms can have several head normal forms but only one principal head normal form. One speaks about the *principal head normal form*. Similarly one can speak about a *reduced form* of A , obtained by μ -reduction, contrasting it with the unique principal reduced form, obtained by μ -contraction in case A is not circular.

The type $B \equiv \mu\beta.A \rightarrow \beta$ has $A \rightarrow B$ as reduced form.

7D.35. LEMMA. (i) If A has $B \rightarrow C$ and $B' \rightarrow C'$ as reduced forms, then $B =_{\mu} B'$ and $C =_{\mu} C'$.

- (ii) If A has α and α' as reduced forms, then $\alpha \equiv \alpha'$.

PROOF. (i), (ii) By the Church-Rosser theorem for \Rightarrow_{μ} . ■

7D.36. LEMMA. (i) Let $\alpha, \beta \in \mathbb{A}$ be different. Then $[\alpha]_{\mu} \neq [\beta]_{\mu}$.

- (ii) The prime elements of $\mathbb{T}_{\mu} = \mathbb{T}_{\mu}^{\mathbb{A}}$ are given by

$$||\mathbb{T}_{\mu}|| = \{[\alpha]_{\mu} \mid \alpha \in \mathbb{A}\} \cup \{[\bullet]\}$$

- (iii) $\mathbb{T}^{\mathbb{A}} \hookrightarrow \mathbb{T}_{\mu}^{\mathbb{A}}$.

PROOF. (i) By Proposition 7D.29.

- (ii) By Lemma 7D.33.

- (iii) One has

$$\begin{aligned}\mathbb{T}^{\mathbb{A}} &\hookrightarrow \mathbb{T}^{||\mathbb{T}_{\mu}||}, & \text{by (i) and (ii),} \\ &\hookrightarrow \mathbb{T}_{\mu}, & \text{by Lemma 7B.12(ii),}\end{aligned}$$

the necessary condition that \mathbb{T}_{μ} is invertible will be proved in Chapter 8. ■

EXAMPLE. The deductions in Proposition 7A.3 can be obtained in $\lambda\mu$ by taking $A \in \mathbb{T}_{\mu}$ arbitrarily and B as $\mu t.t \rightarrow A$. Then one has $B = \mu t.t \rightarrow A =_{\mu} (\mu t.t \rightarrow A) \rightarrow A = B \rightarrow A$.

7D.37. FACT. In Chapter 8 we will see the following properties of \mathbb{T}_{μ} .

- (i) \mathbb{T}_{μ} is closed, i.e. $\mathbb{T}_{\mu} \models \exists \vec{X}. \mathcal{R}$, for all sr \mathcal{R} over \mathbb{T}_{μ} .
- (ii) $=_{\mu}$ is invertible, and hence \mathbb{T}_{μ} is an invertible type algebra.
- (iii) $=_{\mu}$ is decidable.

7E. Recursive types as trees

We now introduce the type-algebra $\text{Tr}_{\text{inf}}^{\mathbb{A}}$ 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 $\mathbb{T}^{\mathbb{A}}$ in a unique way and induce a stronger notion of equivalence between types. For example, if $A = A \rightarrow b$ and $B = (B \rightarrow b) \rightarrow b$, then in Tr_{inf} one has $A = B$.

We first need to recall the basic terminology needed to present (infinite) labeled trees by means of addresses and suitable labelings for them. For more information the reader is referred to [Courcelle \[1983\]](#) for a classic and comprehensive treatment of trees. A node of a tree is represented by the unique path in the tree leading to it. 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.

7E.1. DEFINITION (Trees). (i) A *type-tree* (or just *tree*, for short) over \mathbb{A} is a partial function

$$t : \{0, 1\}^* \nrightarrow (\mathbb{A} \cup \{\rightarrow\} \cup \{\bullet\})$$

satisfying the following conditions.

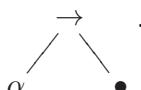
- (1) $\epsilon \in \text{dom}(t)$ (i.e. a tree has at least the root node);
- (2) if $uv \in \text{dom}(t)$, then also $u \in \text{dom}(t)$
(i.e. if a node is in a tree, then all its prefixes are in that tree);
- (3) if $t(u) = \rightarrow$ then $u0, u1 \in \text{dom}(t)$
(i.e. if \rightarrow is given at a node, then it has exactly two successors);
- (4) if $t(u) \in \mathbb{A} \cup \{\bullet\}$ then $uv \notin \text{dom}(t)$ for all $v \neq \epsilon$
(i.e. labels other than ' \rightarrow ' occur only at endpoints).
- (ii) The set of trees over \mathbb{A} will be denoted by $\text{Tr}_{\text{inf}}^{\mathbb{A}}$.
- (iii) Define $\text{Tr}_{\text{fin}}^{\mathbb{A}} \subseteq \text{Tr}_{\text{inf}}^{\mathbb{A}}$ to be the set of *finite* trees, i.e. those with finite domain.

We will often write simply Tr_{inf} , or Tr_{fin} , when \mathbb{A} is understood or not relevant.

EXAMPLE. The function t defined on the (finite) domain $\{\epsilon, 0, 1\}$ as follows:

$$\begin{aligned} t(\epsilon) &\triangleq \rightarrow, \\ t(0) &\triangleq \alpha, \\ t(1) &\triangleq \bullet, \end{aligned}$$

represents the tree



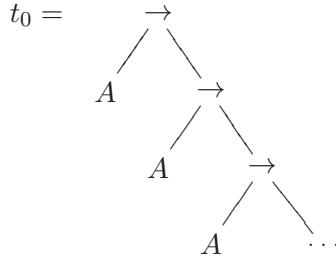
We say that the symbol $c \in \mathbb{A} \cup \{\rightarrow, \bullet\}$ occurs in tree t at node $u \in \Sigma^*$ if $t(u) = c$.

Since $\text{Tr}_{\text{fin}}(\mathbb{A})$ is clearly isomorphic to $\mathbb{T}(\mathbb{A})$, 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 periodic structure, which represent solutions of (systems of) equations of the form

$$\xi = A[\alpha := \xi],$$

where $A \in \text{Tr}_{\text{fin}}$. These will be called the regular trees. See [Courcelle \[1983\]](#), §4.

7E.2. EXAMPLE. Take the equation $\xi = A \rightarrow \xi$ where A is any type (with by the variable convention $\xi \notin \text{FV}(A)$). The solution of this equation is given by $\xi = t_0$ where t_0 is the following infinite tree.



Note that t_0 has only a finite number of distinct subtrees, namely A and t_0 itself. Such a tree is called *regular*.

7E.3. DEFINITION (Regular trees). (i) Let $t \in \text{Tr}_{\text{inf}}$ be a tree and $w \in \text{dom}(t)$ be a word. The *subtree of t at w* , notation $t|w$, is the tree defined by the following conditions.

- (1) $\text{dom}(t|w) = \{u \in \{0, 1\}^* \mid wu \in \text{dom}(t)\}$;
- (2) $(t|w)(u) = t(wu)$, for all $u \in \text{dom}(t|w)$.

A *subtree* of t is $t|w$ for some $w \in \text{dom}(t)$.

(ii) A tree is *regular* if the set of its subtrees is finite. The set of regular trees is denoted by $\text{Tr}_{\text{reg}}^{\mathbb{A}}$ or Tr_{reg} .

7E.4. REMARK. One has the following characterization of $t|u$

$$\begin{aligned} t|\epsilon &= t; \\ t|iv &= t_i, \quad \text{if } i \in \{0, 1\} \text{ and } t = && ; \\ && \begin{array}{ccc} \nearrow & \searrow & \\ t_1 & & t_2 \end{array} \\ &= \uparrow, \quad \text{otherwise.} \end{aligned}$$

Note that $\text{Tr}_{\text{fin}} \subseteq \text{Tr}_{\text{reg}} \subseteq \text{Tr}_{\text{inf}}$. 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 Tr_{inf} , Tr_{reg} and Tr_{fin} as type-algebras. Since \rightarrow is injective, all of them are invertible.

7E.5. LEMMA. Tr_{inf} , Tr_{reg} and Tr_{fin} are invertible type-algebras.

7E.6. DEFINITION. Let $t, u \in \text{Tr}_{\text{inf}}$.

- (i) Define for $k \in \omega$ the tree $(t)_k \in \text{Tr}_{\text{fin}}$, its *truncation* at level k , as follows.

$$\begin{aligned} (t)_0 &\triangleq \bullet; \\ (\alpha)_{k+1} &\triangleq \alpha, \quad \text{for } \alpha \in \mathbb{A}; \\ ((t \rightarrow u))_{k+1} &\triangleq (t)_k \rightarrow (u)_k. \end{aligned}$$

- (ii) Define $t =_k u \iff (t)_k = (u)_k$.

EXAMPLE. Let $t = \alpha \rightarrow \beta$. Then

$$\begin{aligned}(t)_0 &= \bullet; \\ (t)_1 &= \bullet \rightarrow \bullet; \\ (t)_n &= t \quad \text{for } n \geq 2.\end{aligned}$$

The following obvious fact will be useful.

7E.7. LEMMA. Let $t, u, t', u' \in \text{Tr}_{\text{inf}}$.

$$\begin{aligned}(\text{i}) \quad t &= u \iff \forall k \geq 0. t =_k u. \\ (\text{ii}) \quad t &=_0 u. \\ (\text{iii}) \quad t \rightarrow u &=_k t' \rightarrow u' \iff t =_k u \& t' =_k u'.\end{aligned}$$

7E.8. DEFINITION. Let $t, s \in \text{Tr}_{\text{inf}}$. 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 .)

$$\begin{aligned}t[\alpha := s](u) &= s(w) \quad \text{if } u = vw \& t(v) = \alpha, \\ &\quad \text{for some } v, w \in \Sigma^*; \\ &= t(u) \quad \text{else.}\end{aligned}$$

Bisimulation

Equalities among trees can be proved by a proof technique called *coinduction*, which allows to show that two trees are equal whenever there is a bisimulation that relates them.

7E.9. DEFINITION. A relation $R \subseteq \text{Tr}_{\text{inf}}^A \times \text{Tr}_{\text{inf}}^A$ is a *bisimulation* if, for all $s, t \in \text{Tr}_{\text{inf}}$ sRt implies

- either $s, t \in \mathbb{A}$ and $s = t$;
- or $s = s_1 \rightarrow s_2$, $t = t_1 \rightarrow t_2$, and s_1Rt_1, s_2Rt_2 .

The existence of a bisimulation between two trees s and t amounts to a proof that there is no contradiction in assuming that $s = t$. The coinduction principle for trees is as follows.

7E.10. PROPOSITION (Coinduction for trees). *Let R be a bisimulation over $\text{Tr}_{\text{inf}} \times \text{Tr}_{\text{inf}}$. Then*

$$\forall s, t \in \text{Tr}_{\text{inf}}. [sRt \Rightarrow s = t].$$

PROOF. Assume that $\langle s, t \rangle \in R$. We can show that $\text{dom}(s) = \text{dom}(t)$ and that, for all addresses $w \in \text{dom}(s)$, $\langle s|w, t|w \rangle \in R$ by induction on the length of w . We have by Definition 7E.9 either $s|w = t|w$ in \mathbb{A} , hence $s(w) = t(w)$, or $s|w = s_1 \rightarrow s_2$ and $t|w = t_1 \rightarrow t_2$ and then also $s(w) = t(w)$, both sides being equal to \rightarrow . Hence in either case $s(w) = t(w)$. So $s = t$ as partial functions. ■

In particular the equality relation is the largest bisimulation over trees. This coinduction principle may be used whenever we want to prove that two finitary presentations of trees, typically given as terms over some type algebra, have the same value.

Tree-unfolding for invertible general type algebras

Every (possibly non-well-founded) 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 flavor similar to the construction of Böhm-trees for untyped lambda terms, see B[1984], in that both have a co-inductive nature.

7E.11. DEFINITION. Let $\mathcal{A} = \langle |\mathcal{A}|, \rightarrow \rangle$ be an invertible type algebra. Write $\text{Tr}_{\text{inf}}^{\mathcal{A}} = \text{Tr}_{\text{inf}}^{||\mathcal{A}||}$. The *tree-unfolding* of a type $a \in \mathcal{A}$, notation $(a)_{\mathcal{A}}^*$, is defined as follows.

$$\begin{aligned}(a)_{\mathcal{A}}^*(\epsilon) &\triangleq a, & \text{if } a \in ||\mathcal{A}||; \\(a)_{\mathcal{A}}^*(\epsilon) &\triangleq \rightarrow, & \text{if } a = b_0 \rightarrow b_1; \\(a)_{\mathcal{A}}^*(iw) &\triangleq \uparrow, & \text{if } a \in ||\mathcal{A}||; \\(a)_{\mathcal{A}}^*(iw) &\triangleq (b_i)_{\mathcal{A}}^*(w) \quad (i = 0, 1), & \text{if } a = b_0 \rightarrow b_1.\end{aligned}$$

In spite of its technicality, the construction of definition 7E.11 is quite intuitive. The tree $(a)_{\mathcal{A}}^*$, which has the (names of the) prime elements of \mathcal{A} as leaves, corresponds to the (possibly) infinite “unfolding” of a with respect to \rightarrow .

7E.12. LEMMA. For invertible \mathcal{A} the map $(-)_{\mathcal{A}}^* : \mathcal{A} \rightarrow \text{Tr}_{\text{inf}}^{||\mathcal{A}||}$ satisfies the following.

$$\begin{aligned}(a)_{\mathcal{A}}^* &= a, & \text{if } a \in ||\mathcal{A}||; \\(a)_{\mathcal{A}}^* &= \begin{array}{c} \rightarrow \\ / \quad \backslash \\ (b_1)_{\mathcal{A}}^* \quad (b_2)_{\mathcal{A}}^* \end{array}, & \text{if } a = b_1 \rightarrow b_2.\end{aligned}$$

Note that this is not an inductive definition: a may be as complex as $a \rightarrow b$.

PROOF. By Definition 7E.11. ■

This property characterizes the map.

7E.13. REMARK. (i) The map $(-)_{\mathcal{A}}^* : \mathcal{A} \rightarrow \text{Tr}_{\text{inf}}^{\mathcal{A}}$ is a morphism.

(ii) In Section 7F we will see that the simple intuitive characterization of Lemma 7E.12 as official definition. See Remark 7F.18.

7E.14. DEFINITION. Let \mathcal{A} be an invertible type-algebra.

(i) *Strong equality*, notation $=_{\mathcal{A}}^*$, is the relation $=_{\mathcal{A}}^* \subseteq \mathcal{A} \times \mathcal{A}$ defined by

$$a =_{\mathcal{A}}^* b \iff (a)_{\mathcal{A}}^* = (b)_{\mathcal{A}}^*.$$

By contrast the relation $=$ will be called *weak equality*.

(ii) Define $\mathcal{A}^* \triangleq \mathcal{A} / =_{\mathcal{A}}^*$.

(iii) For $a \in \mathcal{A}$ write $a^* \triangleq [a] =_{\mathcal{A}}^*$.

It is immediate from the definition that $=_{\mathcal{A}}^*$ is a congruence with respect to \rightarrow and hence \mathcal{A}^* is well defined.

7E.15. LEMMA. Let \mathcal{A} be an invertible type-algebra.

(i) $=_{\mathcal{A}}^*$ is the greatest invertible congruence over \mathcal{A} such that for all $a \in ||\mathcal{A}||$

$$[a] =_{\mathcal{A}}^* = \{a\}.$$

(ii) \mathcal{A}^* is an invertible type-algebra and $||\mathcal{A}^*|| = \{a^* \mid a \in ||\mathcal{A}||\}$.

PROOF. (i) To prove invertibility note that $a_1 \rightarrow a_2 =^* b_1 \rightarrow b_2$ implies, by definition, $(a_i)^* = (b_i)^*$ ($i = 1, 2$) and then $a_i =^* b_i$. Note also that if $a \in ||\mathcal{A}||$, then $a =^* b$ iff $a = b$. Hence $a^* = \{a\}$ and indeed a^* is prime.

On the other hand, let now \approx be any invertible congruence over \mathcal{A} such that $[a]_{\approx} = \{a\}$ for all $a \in ||\mathcal{A}||$ and $\approx \not\subseteq =_{\mathcal{A}}^*$. Then there are two elements a and b such that $a \approx b$ but $a \neq^* b$. In this case there must be some finite path w such that $(a)_{\mathcal{A}}^*(w) \neq (b)_{\mathcal{A}}^*(w)$. 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 $a' \approx b' \rightarrow c'$ or $a' \approx d'$ for some other prime element d' of \mathcal{A} . In both cases $[a']_{\approx} \neq \{a'\}$.

(ii) By (i). ■

7E.16. PROPOSITION. Let \mathcal{A} be an invertible type-algebra.

(i) There are canonical morphisms

$$\begin{aligned} i_{\mathcal{A}} &: \mathbb{T}^{||\mathcal{A}||} \hookrightarrow \mathcal{A} \\ f &: \mathcal{A} \rightarrow \mathcal{A}^* \\ i_* &: \mathcal{A}^* \hookrightarrow \text{Tr}_{\text{inf}}^{\mathcal{A}} \end{aligned}$$

with f surjective, $i_{\mathcal{A}}, i_*$ injective and $()^* = i_* \circ f$; in diagram

$$\begin{array}{ccccc} \mathbb{T}^{||\mathcal{A}||} & \xrightarrow{i_{\mathcal{A}}} & \mathcal{A} & \xrightarrow{(\)_{\mathcal{A}}^*} & \text{Tr}_{\text{inf}}^{\mathcal{A}} \\ & & \searrow f & & \nearrow i_* \\ & & \mathcal{A}^* & & \end{array}$$

(ii) The maps in (i) are uniquely determined by postulating for all $a \in ||\mathcal{A}||$

$$\begin{aligned} i_{\mathcal{A}}(a) &= a; \\ f(a) &= (a)_{\mathcal{A}}^*; \\ i_*((a)_{\mathcal{A}}^*) &= (a)_{\mathcal{A}}^*. \end{aligned}$$

PROOF. (i) By Propositions 7B.12 and 7B.5(ii).

(ii) By the (easy) proof of these Propositions. ■

Tree-unfolding for invertible syntactic type algebras

The most interesting applications of the tree-unfolding $(-)_{\mathcal{A}}^*$, Definition 7E.11, are for syntactic type-algebras of the form $\mathcal{A} = \mathbb{T}/\approx$. In this case \mathcal{A}^* can easily be described.

7E.17. DEFINITION. Let \mathcal{A} be an invertible syntactic type algebra.

(i) In case $\mathcal{A} = \mathbb{T}^{\mathbb{A}}/\approx$, with invertible relation \approx , write

$$\begin{aligned} (-)_{\approx}^* &\text{ for } (-)_{\mathcal{A}}^* : \mathcal{A} \rightarrow \text{Tr}_{\text{inf}}^{\mathbb{A}}, \\ =^* &\text{ for } =_{\mathcal{A}}^*, \\ \mathbb{T}^{\mathbb{A}}_{\approx}^* &\text{ for } \mathcal{A}/=_{\mathcal{A}}^*. \end{aligned}$$

(ii) In case $\mathcal{A} = \mathbb{T}^{\mathbb{A}}[\mathcal{R}]$, which is always invertible, write

$$\begin{aligned} (-)_{\mathcal{R}}^* &\text{ for } (-)_{\mathcal{A}}^* : \mathcal{A} \rightarrow \text{Tr}_{\text{inf}}^{\mathbb{A}}, \\ =_{\mathcal{R}}^* &\text{ for } =_{\mathcal{A}}^*, \\ \mathbb{T}[\mathcal{R}]^* &\text{ for } \mathcal{A}/=_{\mathcal{A}}^*. \end{aligned}$$

(iii) In case $\mathcal{A} = \mathbb{T}_\mu^\mathbb{A}/=_\mu$, which is always invertible, write

$$\begin{aligned} (-)_\mu^* &\text{ for } (-)_{\mathcal{A}}^* : \mathcal{A} \rightarrow \mathbf{Tr}_{\text{inf}}^\mathbb{A}, \\ =_\mu^* &\text{ for } =_{\mathcal{A}}^*, \\ \mathbb{T}_\mu^* &\text{ for } \mathcal{A}/=_\mu^*. \end{aligned}$$

7E.18. REMARK. In case $\mathcal{A} = \mathbb{T}^\mathbb{A}/\approx$, see exercise 7G.10, Lemma 7E.15 states that $=^*$ is the greatest invertible congruence on \mathbb{T} extending \approx and preserving all prime elements of \mathcal{A} . That is, if \square is an invertible congruence on \mathbb{T} extending \approx , such that

$$[A]_\approx \in ||\mathcal{A}|| \& A \square B \Rightarrow [A]_\approx = [B]_\approx,$$

then $\square \subseteq =^*$.

The map $(-)_{\mathcal{A}}^*$ can be considered as an *interpretation* of types in \mathbb{T} as infinite trees.

7E.19. NOTATION. (i) Working in a syntactic type algebra $\mathbb{T}^\mathbb{A}/\approx$ we view a type as if it is its equivalence class. Therefore we bend the meaning of $(-)_{\approx}^*$ so that it also applies to an element of $\mathbb{T}^\mathbb{A}$:

$$(-)_{\approx}^* : \mathbb{T}^\mathbb{A} \rightarrow \mathbf{Tr}_{\text{inf}}^\mathbb{A}.$$

That is, we identify $(-)_{\approx}^* = (-)_{\approx}^* \natural$, see Lemma 7B.4(ii) for the definition of \natural .

$$\begin{array}{ccc} \mathbb{T}^\mathbb{A} & \xrightarrow{(-)_{\approx}^* = (-)_{\approx}^* \natural} & \mathbf{Tr}_{\text{inf}}^\mathbb{A} \\ & \searrow [-]_\approx & \nearrow (-)_{\approx}^* \\ & \mathbb{T}^\mathbb{A}/\approx & \end{array}$$

(ii) For $\mathcal{A} = \mathbb{T}^\mathbb{A}[\mathcal{R}]$ and $\mathcal{A} = \mathbb{T}_\mu^\mathbb{A}/=_\mu$ this boils down to

$$\begin{aligned} (-)_{\mathcal{R}}^* : \mathbb{T}(\vec{X}) &\rightarrow \mathbf{Tr}_{\text{inf}}^\mathbb{A}; \\ (-)_\mu^* : \mathbb{T}_\mu^\mathbb{A} &\rightarrow \mathbf{Tr}_{\text{inf}}^\mathbb{A}. \end{aligned}$$

(iii) One has by definition

$$\begin{aligned} A =^* B &\Leftrightarrow (A)_{\approx}^* = (B)_{\approx}^*, \\ A =_{\mathcal{R}}^* B &\Leftrightarrow (A)_{\mathcal{R}}^* = (B)_{\mathcal{R}}^*, \\ A =_\mu^* B &\Leftrightarrow (A)_\mu^* = (B)_\mu^*. \end{aligned}$$

7E.20. LEMMA. *In this context we have the following.*

- (i) $\mathbb{T}_{\approx}^* \cong \mathbb{T}/=^*$.
- (ii) $\mathbb{T}[\mathcal{R}]^* \cong \mathbb{T}(\vec{X})/=_\mathcal{R}^*$.
- (iii) $\mathbb{T}_\mu^* \cong \mathbb{T}_\mu/=_\mu^*$.

PROOF. Immediate from Definition 7E.17 and Notation 7E.19. ■

Now we focus on the syntactic type-algebras $\mathbb{T}(\vec{X})/\approx$, $\mathbb{T}[\mathcal{R}]$ and $\mathbb{T}_\mu/=_\mu$. By Theorem 7C.12 we have that $\mathbb{T}[\mathcal{R}]$ is invertible. In Chapter 9A9 it will be proved that the other two type algebras are also invertible.

Tree equivalence for sr

For syntactic type algebras coming from a simultaneous recursion \mathcal{R} , the tree-unfolding $(\)^*_{\mathcal{R}} : \mathbb{T} \rightarrow \text{Tr}_{\text{inf}}$ can be characterized as follows.

7E.21. LEMMA. *Let $\mathcal{R} = \mathcal{R}(\vec{X})$ be an sr over $\mathbb{T} = \mathbb{T}^{\mathbb{A}}$. Then we have the following.*

(i) $\mathbb{T}[\mathcal{R}]$ is invertible.

(ii) Suppose moreover that \mathcal{R} is proper. Then the map $(-)_{\mathcal{R}}^* : \mathbb{T}[\mathcal{R}] \rightarrow \text{Tr}_{\text{inf}}^{\mathbb{A}}$ can be characterized as follows.

$$\begin{aligned} (A)_{\mathcal{R}}^* &= \alpha, & \text{if } A =_{\mathcal{R}} \alpha \text{ with } \alpha \in \mathbb{A}; \\ (A)_{\mathcal{R}}^* &= \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ (B)_{\mathcal{R}}^* \quad (C)_{\mathcal{R}}^* \end{array}, & \text{if } A =_{\mathcal{R}} B \rightarrow C, \\ (A)_{\mathcal{R}}^* &= (B)_{\mathcal{R}}^*, & \text{if } A =_{\mathcal{R}} X \text{ and } (X = B) \in \mathcal{R}. \end{aligned}$$

PROOF. (i) By Theorem 7C.12.

(ii) Note that by Lemma 7C.20 one has $\|\mathbb{T}[\mathcal{R}]\| = \{[\alpha] \mid \alpha \in \mathbb{A}\}$. ■

REMARK. If \mathcal{R} is not proper there may be equations like $X = X$, or $X = Y, Y = X$. In this case the above procedure becomes circular due to the third clause in Lemma 7E.21(ii).

7E.22. EXAMPLE. (i) Let $\mathcal{R} = \{X = A \rightarrow X, Y = A \rightarrow A \rightarrow Y\}$. We have $X \neq_{\mathcal{R}} Y$. But both types unfold to the same tree $(X)_{\mathcal{R}}^* = (Y)_{\mathcal{R}}^* = t_0$, where t_0 is the tree defined in Example 7E.2.

(ii) Let \mathcal{R} be the sr defined by the following equations.

$$\begin{aligned} X_1 &= X_2 \rightarrow X_1 \\ X_2 &= X_1 \rightarrow X_2 \end{aligned}$$

It is easy to see that $(X_1)_{\mathcal{R}}^* = (X_2)_{\mathcal{R}}^* = t_1$ where t_1 is the following infinite tree

$$t_1 = \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \\ \dots \quad \dots \quad \dots \quad \dots \end{array}.$$

Hence $X_1 =_{\mathcal{R}}^* X_2$. Note that $X_1 \neq_{\mathcal{R}} X_2$. Also $(X)_{\mathcal{R}_2}^* = t_1$, where $\mathcal{R}_2 = \{X = X \rightarrow X\}$.

(iii) Take the sr $\mathcal{R} \triangleq \{X = A \rightarrow X, Y = A \rightarrow Y\}$. 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 7E.2. Note that $X \neq_{\mathcal{R}} Y$.

So we see that the relation $=_{\mathcal{R}}^*$ has a more semantic nature than $=_{\mathcal{R}}$. It turns out to be the type equivalence induced by the interpretation of types in continuous models (see Chapter 10). The relation $=_{\mathcal{R}}^*$ can be also characterized as the equational theory of a suitable sr \mathcal{R}^* . In particular in Section 8 we will prove constructively the following property.

7E.23. THEOREM. *Let $\mathcal{R}(\vec{X})$ be an sr over \mathbb{T} . Then there is an sr $\mathcal{R}^*(\vec{X})$ such that for all $A, B \in \mathbb{T}(\vec{X})$ one has $A =_{\mathcal{R}}^* B \Leftrightarrow A =_{\mathcal{R}^*} B$.*

7E.24. COROLLARY. $\mathbb{T}[\mathcal{R}]^* \cong \mathbb{T}[\mathcal{R}^*]$.

PROOF. $\mathbb{T}[\mathcal{R}]^* \cong \mathbb{T}(\vec{X}) / =_{\mathcal{R}}^*$, by Lemma 7E.20(ii),
 $= \mathbb{T}(\vec{X}) / =_{\mathcal{R}}^*$, by the Theorem,
 $= \mathbb{T}[\mathcal{R}^*]$. ■

In Section 8 we will give an alternative axiomatization for the relation $=_{\mathcal{R}}^*$ using a coinduction principle.

As a consequence of Theorem 7C.14 the sr \mathcal{R} is solved not only by the type-algebra $\mathbb{T}[\mathcal{R}]$, but also by its tree-collapse $\mathbb{T}[\mathcal{R}]^*$. But now, as bonus, the solution is unique.

7E.25. THEOREM. *Let $\mathcal{R}(\vec{X}) = \{X_1 = A_1(\vec{X}), \dots, X_n = A_n(\vec{X})\}$ be an sr over \mathbb{T} . Then*

- (i) $\mathbb{T} \hookrightarrow \mathbb{T}[\mathcal{R}]^*$ and $\mathbb{T}[\mathcal{R}]^*$ is generated by (the image of) \mathbb{T} and the \vec{X} .
- (ii) $\mathbb{T}[\mathcal{R}]^* \models \exists \vec{X} \mathcal{R}$.
- (iii) If moreover \mathcal{R} is proper, then \vec{X} , with $\mathbf{X}_i = [X_i] =_{\mathcal{R}}^*$ is the unique solution of \mathcal{R} in $\mathbb{T}[\mathcal{R}]^*$.

PROOF. (i) By Theorem 7C.14(iii) and Lemma 7E.16 (ii).

(ii) \vec{X} is a solution of \mathcal{R} : apply 7C.14(i) and 7E.16(ii).
(iii) Note that $\mathbb{T}[\mathcal{R}]^*$ can be seen as a subset of $\mathsf{Tr}_{\text{inf}}^n$. The map $F: \mathsf{Tr}_{\text{inf}}^n \rightarrow \mathsf{Tr}_{\text{inf}}^n$ defined by $F(\vec{X}) = \langle A_1(\vec{X}), \dots, A_n(\vec{X}) \rangle$ satisfies the conditions of Proposition 7F.8(ii), since \mathcal{R} is proper. Hence by that Proposition it has a unique fixed point. Clearly \vec{X} viewed as element of $\mathsf{Tr}_{\text{inf}}^n$ is this fixed point. ■

Tree equivalence for μ -types

7E.26. FACT. The type algebra $\mathbb{T}_\mu / =_\mu$ is invertible. See Theorem 8B.1.

Therefore the construction of 7E.11 can be applied to this type algebra. For example, from Lemma 7E.20 it follows that $(\mathbb{T}_\mu / =_\mu)^* = \mathbb{T}_\mu / =_\mu^*$.

7E.27. LEMMA. *The map $(-)_\mu^*: \mathbb{T}_\mu^\mathbb{A} \rightarrow \mathsf{Tr}_{\text{inf}}^\mathbb{A}$ can be characterized in the following way.*

$$\begin{aligned} (A)_\mu^* &= \alpha, && \text{if } \alpha \text{ is a reduced form of } A, \\ (A)_\mu^* &= \begin{array}{c} \rightarrow \\ \swarrow \quad \searrow \end{array}, && \text{if } B \rightarrow C \text{ is a reduced form of } A, \\ && (B)_\mu^* \quad (C)_\mu^* \\ (A)_\mu^* &= \bullet, && \text{if } A \text{ is circular.} \end{aligned}$$

PROOF. Similar to that of 7E.21. ■

In particular $=_\mu^*$ is an invertible congruence extending $=_\mu$ and there is a unique morphism $h^*: \mathbb{T}_\mu \rightarrow \mathbb{T}_\mu^*$ (or rather $h^*: \mathbb{T}_\mu / =_\mu \rightarrow \mathbb{T}_\mu^*$) defined by $h^*([A] =_\mu) = [A] =_\mu^*$.

EXAMPLE. Consider the types $B = \mu\beta.A \rightarrow \beta$ and $B' = \mu\beta.A \rightarrow A \rightarrow \beta$ (with by the variable convention $\beta \notin \text{FV}(A)$). We have that $B =_\mu^* B'$. Indeed, one has $(B)_\mu^* = (B')_\mu^* = t_0$, where t_0 is the infinite tree of example 7E.2.

7E.28. LEMMA. $(A[\alpha := B])_\mu^* = (A)_\mu^*[(\alpha)_\mu^* := (B)_\mu^*]$.

PROOF. By induction on the structure of A . ■

7E.29. REMARK. Note that by lemma 7E.28

$$(\mu\alpha.A)_{\mu}^{*} = (A[\alpha := \mu\alpha.A])_{\mu}^{*} = (A)_{\mu}^{*}[\alpha := (\mu\alpha.A)_{\mu}^{*}].$$

So $(\mu\alpha.A)_{\mu}^{*}$ is a solution of the equation $X = (A)_{\mu}^{*}[\alpha := X]$ in Tr_{reg} . If $A \neq \alpha$, this solution in Tr_{inf} is unique, by Proposition 7F.8 and Theorem 7F.5.

In Chapter 8 we will give a complete axiomatization of $=_{\mu}^{*}$. As an application we obtain a constructive proof that $=_{\mu}^{*}$ is decidable.

Types as regular trees

All trees in the codomain of $(-)^{*}_{\mathcal{R}}$ are regular.

7E.30. LEMMA. 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. $(-)_{\mathcal{R}}^{*} : \mathbb{T}[\mathcal{R}] \rightarrow \text{Tr}_{\text{reg}}$.

PROOF. Let $\vec{X} = X_1, \dots, X_n$ ($n \geq 0$) 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 is easy to prove by induction on w that for all $w \in \{0, 1\}^*$ for which $(A)_{\mathcal{R}}^{*}(w)$ is defined we have

$$(A)_{\mathcal{R}}^{*}|w = (C)_{\mathcal{R}}^{*} \text{ for some } C \in \mathcal{S}(A) \cup \bigcup_{1 \leq i \leq n} \{X_i\} \cup \bigcup_{1 \leq i \leq n} \mathcal{S}(B_i).$$

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.

7E.31. THEOREM. (i) Let $\mathcal{R}(\vec{X})$ be a proper sr over \mathbb{T} . Then \mathcal{R} has a unique solution in Tr_{reg} given by $\vec{t} = (X_1)_{\mathcal{R}}^{*}, \dots, (X_n)_{\mathcal{R}}^{*}$.

- (ii) Tr_{inf} is a uniquely closed type algebra.
- (iii) Tr_{reg} is a uniquely closed type algebra.

PROOF. (i) By theorem 7E.25 and 7E.30.

(ii) As the proof of Theorem 7E.25(iii).

(iii) Every \mathcal{R} over Tr_{reg} has a unique solution in Tr_{inf} . That the solution is in Tr_{reg} follows as in the proof of Lemma 7E.30. ■

7E.32. REMARK. (i) On the other hand each regular tree can be obtained as a component of a solution of some sr \mathcal{R} , see Exercise 7G.9. See also [Courcelle \[1983\]](#), Theorem 4.2.1.

(ii) It is easy to see that lemma 7E.30 and theorem 7E.31 hold also if we assume more generally that \mathcal{R} is an sr with coefficients in Tr_{reg} , see also [Courcelle \[1983\]](#), Theorem 4.3.1.

7E.33. REMARK. The mapping $(-)^{*}_{\mathcal{R}}$ can be seen as a *unifier* of \mathcal{R} in Tr_{reg} . It is well known, see [Courcelle \[1983\]](#), Prop. 4.9.5(ii), that $(-)_{\mathcal{R}}^{*}$ is indeed the *most general* unifier of \mathcal{R} in Tr_{reg} . This means that each syntactic morphism $h : \mathbb{T}[\mathcal{R}] \rightarrow \text{Tr}_{\text{reg}}$ can be written as $h = s \circ (-)^{*}_{\mathcal{R}}$ where s is a substitution in Tr_{reg} . Note that any such substitution can be seen as a morphism $s : \text{Tr}_{\text{reg}} \rightarrow \text{Tr}_{\text{reg}}$.

7E.34. REMARK. Simultaneous recursions of a special form can be viewed as $T_{\mathbb{A}}$ -coalgebras, where $T_{\mathbb{A}}$ is the tree functor of Definition 7F.15, as suggested by an analogy of simultaneous recursions with systems of equations defining non-well-founded sets ([Aczel](#)

[1988]). See Section 4 of [Cardone and Coppo \[2003\]](#) for an application of this remark to the decidability of strong equivalence of simultaneous recursions.

The μ -types are just notations for regular trees, see [Courcelle \[1983\]](#), Theorem 4.5.7.

- 7E.35. LEMMA. (i) *If $A \in \mathbb{T}_\mu$ then $(A)_\mu^* \in \mathsf{Tr}_{\text{reg}}$.*
(ii) *Each regular tree t can be written as $t = (A)_\mu^*$, for some type $A \in \mathbb{T}_\mu$.*
(iii) $\mathbb{T}_\mu^* \cong \mathsf{Tr}_{\text{reg}}$.

PROOF. (i) This follows from Remark 7E.32(i).

(ii) By the fact that \mathbb{T}_μ is a closed type-algebra (a proof of this will be given in Section 9).

(iii) Immediate from (i). ■

Type assignment modulo tree equality

7E.36. DEFINITION. As a shorthand we denote the type assignment systems corresponding to \mathbb{T}_μ , \mathbb{T}_μ^* , \mathbb{T}/\mathcal{E} , $\mathbb{T}[\mathcal{R}]$, and $\mathbb{T}[\mathcal{R}]^*$ by $\lambda\mu$, $\lambda\mu^*$, $\lambda\mathcal{E}$, $\lambda\mathcal{R}$, 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.

7E.37. REMARK. (i) It is easy to see that any substitution $s : A \rightarrow \mathbb{T}_\mu$ determines a type algebra morphism of $s : \mathbb{T}_\mu \rightarrow \mathbb{T}_\mu$. So all the typings of Lemma 7A.3 hold in $\lambda\mu$ if we replace α by any type $A \in \mathbb{T}_\mu$ (i.e. taking the substitution $[\alpha := A]$). Note in particular that $(\lambda x.xx)(\lambda x.xx)$ has all types and that the fixed point operator has all the types of the form $(A \rightarrow A) \rightarrow A$. Obviously the same property holds in the other systems, in particular in $\lambda\mu^*$.

(ii) The two systems $\lambda\mu$ and $\lambda\mu^*$ are not equivalent. Take $T_0 = \mu\beta.\alpha \rightarrow \beta$ and $T_1 = \mu\beta.\alpha \rightarrow \alpha \rightarrow \beta$ as in Example 7D.27. Then $\{x : T_1\} \not\vdash_{\lambda\mu} x : T_0$ while $\{x : T_1\} \vdash_{\lambda\mu^*} x : T_0$. Similarly $\{x : T_1\} \not\vdash_{\lambda\mu} x : \alpha \rightarrow T_1$ but $\{x : T_1\} \vdash_{\lambda\mu^*} x : \alpha \rightarrow T_1$.

Notations for type assignment systems

The notion of type algebra has been introduced to unify a number of type assignment systems. The general version is $\lambda\mathcal{A}_\equiv$, that specializes to various cases.

We make the following notation.

7E.38. DEFINITION. (i) The type assignment system $\lambda\mathcal{A}_\equiv$ is also denoted as $\lambda\mathcal{A}$.

(ii) Similar shorthands apply to several other systems.

Type algebra	Type assignment system	Simplified notation
\mathcal{A}	$\lambda_{\equiv}^{\mathcal{A}}$	$\lambda \mathcal{A}$
\mathbb{T}/\approx	$\lambda_{\equiv}^{\mathbb{T}/\approx}$	$\lambda \approx$
\mathbb{T}/ε	$\lambda_{\equiv}^{\mathbb{T}/\varepsilon}$	$\lambda \varepsilon$
$\mathbb{T}[\mathcal{R}]$	$\lambda_{\equiv}^{\mathbb{T}[\mathcal{R}]}$	$\lambda \mathcal{R}$
$\mathbb{T}(\mathcal{R})^*$	$\lambda_{\equiv}^{\mathbb{T}(\mathcal{R})^*}$	$\lambda \mathcal{R}^*$
$\mathbb{T}_\mu / =_\mu$	$\lambda_{\equiv}^{\mathbb{T}_\mu}$	$\lambda \mu$
\mathbb{T}_μ^*	$\lambda_{\equiv}^{\mathbb{T}_\mu^*}$	$\lambda \mu^*$

(iii) The simplified notation is introduced so we can write

$$\vdash_{\lambda \mathcal{A}}, \vdash_{\lambda \approx}, \vdash_{\lambda \varepsilon}, \vdash_{\lambda \mathcal{R}}, \vdash_{\lambda \mathcal{R}^*}, \vdash_{\lambda \mu}, \vdash_{\lambda \mu^*}.$$

The approach we are taking in this Part of the book comes from [Scott \[1975b\]](#): this manifold of type assignment systems can be captured by one idea and a parameter: the type algebra.

Explicitly typed versions

The explicitly typed versions of the systems $\lambda \mathcal{R}$ and $\lambda \mu$ are usually given in ‘Church’ style with coercions (the *fold-unfold* constants) introduced in Definition 7A.22. In the system λ_μ^{Ch} in particular the fold-unfold constants are labeled with the folded version of the corresponding μ -type

$$\begin{aligned} \text{fold}_{\mu t.A} &\in \Lambda_{\mathbb{T}_\mu}^{\text{Ch}}(A[t := \mu t.A] \rightarrow \mu t.A); \\ \text{unfold}_{\mu t.A} &\in \Lambda_{\mathbb{T}_\mu}^{\text{Ch}}(\mu t.A \rightarrow A[t := \mu t.A]). \end{aligned}$$

On the terms extended by the coercion operators one postulates a notion of reduction

$$\begin{aligned} \text{unfold}_{\mu t.A}(\text{fold}_{\mu t.A} M) &\rightarrow M; \\ \text{fold}_{\mu t.A}(\text{unfold}_{\mu t.A} M') &\rightarrow M', \end{aligned}$$

where $M \in \Lambda_{\mathbb{T}_\mu}^{\text{Ch}}(A[t := \mu t.A])$ and $M' \in \Lambda_{\mathbb{T}_\mu}^{\text{Ch}}(\mu t.A)$.

7F. Special views on trees

The set of trees, as introduced in Definition 7E.1, can be viewed as a metric space or as a coalgebra. They are given here for the interested reader, and may be skipped.

Trees as a metric space

We can turn Tr_{inf} into a metric space in the following way, [Courcelle \[1983\]](#).

7F.1. DEFINITION. [Metrics over Tr_{inf}] Let $\alpha, \alpha' \in \text{Tr}_{\text{inf}}$. Define

$$d(\alpha, \alpha') \triangleq \begin{cases} 0 & \text{if } \alpha = \alpha' \\ 2^{-\delta(\alpha, \alpha')} & \text{if } \alpha \neq \alpha' \end{cases}$$

where $\delta(\alpha, \alpha')$ is the length of the minimum path w such that $w \in \text{dom}(\alpha)$, $w \in \text{dom}(\alpha')$ and $\alpha(w) \neq \alpha'(w)$.

7F.2. PROPOSITION (Courcelle [1983]). $\langle \text{Tr}_{\text{inf}}, d \rangle$ is a complete metric space. With respect to the resulting topology, the set Tr_{fin} is a dense subset of Tr_{inf} and Tr_{inf} is the topological completion of Tr_{fin} .

7F.3. REMARK. $\langle \text{Tr}_{\text{inf}}, d \rangle$ is even an *ultrametric space*, i.e. satisfies the strengthened triangle inequality

$$d(x, y) \leq \max\{d(x, z), d(z, y)\}.$$

7F.4. DEFINITION. Let $\langle D, d \rangle$ be a metric space. A map $f : D \rightarrow D$ is called *contractive* if there exists a real number c ($0 \leq c < 1$) such that

$$\forall x, x' \in D \quad d(f(x), f(x')) \leq c \cdot d(x, x').$$

A basic property of complete metric spaces is the following.

7F.5. THEOREM (Banach fixed point theorem). Let $\langle D, d \rangle$ be a complete metric space. Every contractive mapping $f : D \rightarrow D$ has a unique fixed point x in D . This will be denoted by $\text{fix}(f)$.

PROOF. Let $x_0 \in D$. Define $x_n \triangleq f^n(x_0)$. This is a Cauchy sequence. The fixed point x can be defined as the limit of the x_n . If x, y are fixed points, then $d(x, y) = d(f^n(x), f^n(y)) \leq c^n \cdot d(x, y)$. Therefore $d(x, y) = 0$ and hence the fixed point is unique. ■

7F.6. PROPOSITION. Let $D_1 = \langle D_1, d_1 \rangle$, $D_2 = \langle D_2, d_2 \rangle$ be metric spaces. Define on $D = D_1 \times D_2$ the map $d : D \rightarrow \mathbb{R}$ by

$$d((x_1, x_2), (y_1, y_2)) = \max\{d_1(x_1, y_1), d_2(x_2, y_2)\}.$$

Then $D = \langle D, d \rangle$ is a metric space.

- (i) If D_1, D_2 are complete metric spaces, then so is D .
- (ii) If D_1, D_2 are ultrametric spaces, then so is D .

7F.7. DEFINITION. An *algebraic* map on a type algebra \mathcal{A} is defined as usual. For example, if $a, b \in \mathcal{A}$, then f defined by $f(x) = a \rightarrow (b \rightarrow x) \rightarrow x$ is algebraic.

7F.8. PROPOSITION. (i) Let $f : \text{Tr}_{\text{inf}} \rightarrow \text{Tr}_{\text{inf}}$ be algebraic that is not the identity. Then f is contractive.

(ii) Let $f_1, \dots, f_n : \text{Tr}_{\text{inf}}^n \rightarrow \text{Tr}_{\text{inf}}$ be algebraic such that for all $a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n \in \text{Tr}_{\text{inf}}$, with $1 \leq i \leq n$, one has

$$\lambda x_i. f_i(a_1, \dots, a_{i-1}, x_i, a_{i+1}, \dots, a_n) : \text{Tr}_{\text{inf}} \rightarrow \text{Tr}_{\text{inf}}$$

Define

$$F(\vec{x}) = \langle f_1(\vec{x}), \dots, f_n(\vec{x}) \rangle.$$

Then F is contractive on Tr_{inf}^n .

PROOF. (i) It suffices to show that the map g defined by $g(x, y) = x \rightarrow y$ is contractive in its two arguments. Indeed, $d(x \rightarrow y, x' \rightarrow y) = \frac{1}{2}d(x, x')$.

- (ii) For notational simplicity we treat $n = 2$. We must show

$$d(F(x_1, x_2), F(y_1, y_2)) \leq c \cdot d((x_1, x_2), (y_1, y_2)) \tag{*}$$

Now

$$\begin{aligned} d(F(x_1, x_2), F(y_1, y_2)) &= d(\langle f_1(x_1, x_2), f_2(x_1, x_2) \rangle, \langle f_1(y_1, y_2), f_2(y_1, y_2) \rangle) \\ &= \max\{d(f_1(x_1, x_2), f_1(y_1, y_2)), d(f_2(x_1, x_2), f_2(y_1, y_2))\}. \end{aligned}$$

Now for $i = 1, 2$ one has

$$\begin{aligned} d((f_i(x_1, x_2), f_i(y_1, y_2))) &\leq \max\{d(f_i(x_1, x_2), f_i(y_1, x_2)), d(f_i(y_1, x_2), f_i(y_1, y_2))\} \\ &\leq \max\{c_i \cdot d(x_1, y_1), c_i \cdot d(x_2, y_2)\}, \text{ by the assumption and (i),} \\ &= c_i \cdot d((x_1, x_2), (y_1, y_2)). \end{aligned}$$

Now (*) easily follows. ■

Now let $t \in \text{Tr}_{\text{inf}}$ and α a variable occurring in t . If $\alpha \not\equiv t$, then $\lambda x \in \text{Tr}_{\text{inf}}. t[\alpha := x]$ defines a contractive mapping of Tr_{inf} into itself and therefore it has a fixed point.

The following property is also easy to prove, see [Courcelle \[1983\]](#), Theorem 4.3.1.

7F.9. PROPOSITION. *If $\alpha \in \text{Tr}_{\text{reg}}$ and $\alpha \not\equiv t$ then $\text{fix}(\lambda x \in \text{Tr}_{\text{inf}}. t[\alpha := x]) \in \text{Tr}_{\text{reg}}$.*

Trees as a coalgebra

Algebras and coalgebras

The notion of *algebra* to be introduced presently arises as a categorical generalization of the usual set-theoretical notion of algebraic structure. In order to provide a background to the general definitions, we show below how natural numbers provide a simple (but fundamental) example of algebra.

7F.10. EXAMPLE. Natural numbers form a structure $\langle \mathbb{N}, 0, \text{suc} \rangle$ with an element $0 \in \mathbb{N}$ and a unary operator $\text{suc} : \mathbb{N} \rightarrow \mathbb{N}$, the successor function. Towards a categorical formulation of this structure, observe that two functions with a common codomain can be ‘packed’ into one function as follows. First of all, define the disjoint union

$$A + B \triangleq (A \times \{0\}) \cup (B \times \{1\}).$$

There are canonical maps

$$\begin{array}{rcl} \text{inl} : & A & \rightarrow A + B \\ & a & \mapsto \langle a, 0 \rangle \\ \text{inr} : & B & \rightarrow A + B \\ & b & \mapsto \langle b, 1 \rangle \end{array}$$

Then, two functions $f : A \rightarrow C$ and $g : B \rightarrow C$ can be packed together as $[f, g] : A + B \rightarrow C$ by setting

$$\begin{aligned} [f, g](\text{inl}(a)) &\triangleq f(a), \\ [f, g](\text{inr}(b)) &\triangleq g(b). \end{aligned}$$

In the case of natural numbers, we have a mapping $[0, \text{suc}] : 1 + \mathbb{N} \rightarrow \mathbb{N}$, where 1 is any singleton, say $1 = \{*\}$ and $0 : 1 \rightarrow \mathbb{N}$ has as its (unique) value exactly the number 0. In categorical terms, the situation can be described as follows.

7F.11. DEFINITION. Let a functor $T : \text{Set} \rightarrow \text{Set}$ be given. We call it the *signature functor*.

(i) A T -*algebra* consists of a pair $\langle X, \xi \rangle$ with $X \in \text{Set}$ the *carrier* and ξ a morphism

$$\xi : T(X) \rightarrow X.$$

- (ii) If $\langle X, \xi : T(X) \rightarrow X \rangle$ and $\langle Y, \zeta : T(Y) \rightarrow Y \rangle$ are two T -algebras, then a *morphism* is a mapping $f : X \rightarrow Y$ such that the following diagram commutes.

$$\begin{array}{ccc} T(X) & \xrightarrow{\xi} & X \\ T(f) \downarrow & & \downarrow f \\ T(Y) & \xrightarrow{\zeta} & Y \end{array}$$

It is easy to check that the T -algebras with these morphisms form a category.

- (iii) A T -algebra $\langle A, \text{in} \rangle$ is called *initial* if 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 T -algebra morphism $\varphi : \langle A, \text{in} \rangle \rightarrow \langle X, \xi \rangle$.

It is easy to see that natural numbers $\langle \mathbb{N}, [0, \text{suc}] \rangle$ form the initial algebra for the signature functor

$$T_{\mathbb{N}}(X) \triangleq 1 + X = X^0 + X^1.$$

Initiality of \mathbb{N} is equivalent to the iteration principle: the unique morphism $h : \mathbb{N} \rightarrow A$ from the $T_{\mathbb{N}}$ -algebra $\langle \mathbb{N}, [0, \text{suc}] \rangle$ to any other $T_{\mathbb{N}}$ -algebra $\langle A, [a, f] \rangle$ is precisely the unique function from \mathbb{N} to A satisfying the equations (see [Dedekind \[1901\]](#)):

$$\begin{aligned} h(0) &= a \\ h(\text{suc}(n)) &= f(h(n)). \end{aligned}$$

7F.12. DEFINITION. A *polynomial* functor $T : \text{Set} \rightarrow \text{Set}$ is of the form

$$T(X) = A_0 \times X^0 + A_1 \times X^1 + \cdots + A_n \times X^n,$$

where \times denotes the cartesian product and the $A_k \in \text{Set}$ are arbitrary objects.

7F.13. PROPOSITION. For every polynomial functor $T : \text{Set} \rightarrow \text{Set}$ there is an initial algebra $\langle I, \text{in} \rangle$ that is unique up to isomorphism in the category of T -algebras. Moreover, $\text{in} : T(I) \rightarrow I$ is an isomorphism, hence $T(I) \cong I$.

PROOF. Do Exercise 7G.20. ■

We now examine *coalgebras*, which are dual to algebras in the categorical sense and appear in nature as spaces of infinitely proceeding processes. A simple coalgebra is that of streams over a set A , namely infinite lists of elements of A . Another coalgebra is that of *lazy lists*, consisting of both the finite lists and the streams. Some properties of these coalgebras are left as exercises. In this section we shall be concerned only with *trees* as a coalgebra. For more examples and properties of coalgebras, and the general theory, we refer the reader to [Rutten \[2000\]](#).

7F.14. DEFINITION (Coalgebra). Let a functor $T : \text{Set} \rightarrow \text{Set}$ be given.

- (i) A *T -coalgebra* is a pair $\langle X, \xi : X \rightarrow T(X) \rangle$.
(ii) If $\langle X, \xi : X \rightarrow T(X) \rangle$ and $\langle Y, \zeta : Y \rightarrow T(Y) \rangle$ are T -coalgebras, then a *T -coalgebra morphism* is a mapping $f : X \rightarrow Y$ such that the following diagram commutes.

$$\begin{array}{ccc} X & \xrightarrow{\xi} & T(X) \\ f \downarrow & & \downarrow T(f) \\ Y & \xrightarrow{\zeta} & T(Y) \end{array}$$

- (iii) A T -coalgebra $\langle C, \text{out} \rangle$ is called *final* if it is a final object in the category of T -coalgebras, i.e., for any T -coalgebra $\langle X, \xi : X \rightarrow T(X) \rangle$ there is a unique T -coalgebra morphism $\varphi : X \rightarrow C$.

Dualizing the proof of proposition 7F.13, we have that every polynomial endofunctor of Set has a final coalgebra, which is unique up to a T -coalgebra isomorphism. Furthermore, final T -coalgebras are fixed points of T : if $\langle X, \xi : X \rightarrow T(X) \rangle$ is the final T -coalgebra, then ξ is a bijection.

Trees as final coalgebras

We sketch now how it is possible – and in fact convenient – to regard (finite and infinite) trees over a set of atoms \mathbb{A} as the final coalgebra of a suitable functor.

7F.15. DEFINITION (The tree functor). Let \mathbb{A} be a set of atoms. The assignment

$$T_{\mathbb{A}}(X) \triangleq \mathbb{A} + (X \times X)$$

defines a functor over Set that will be called the *tree functor*.

We can define the $T_{\mathbb{A}}$ -coalgebra:

$$\omega : \text{Tr}_{\text{inf}}(\mathbb{A}) \longrightarrow \mathbb{A} + \text{Tr}_{\text{inf}}(\mathbb{A}) \times \text{Tr}_{\text{inf}}(\mathbb{A})$$

by the following clauses:

$$\begin{aligned} \omega(t) &\triangleq \text{inl}(a), & \text{if } t = a \in \mathbb{A}, \\ &\triangleq \text{inr}(\langle t', t'' \rangle), & \text{if } t = t' \rightarrow t''. \end{aligned}$$

7F.16. PROPOSITION. $\langle \text{Tr}_{\text{inf}}(\mathbb{A}), \omega \rangle$ is the final $T_{\mathbb{A}}$ -coalgebra.

PROOF. We only give the details of the construction of the unique $T_{\mathbb{A}}$ -coalgebra morphism toward $\langle \text{Tr}_{\text{inf}}(\mathbb{A}), \omega \rangle$, because this has a concrete description that shall also be exploited later. Let $\xi : X \longrightarrow \mathbb{A} + (X \times X)$ be any $T_{\mathbb{A}}$ -coalgebra. First define, for any $x \in X$, an element $\ell(x) \in \mathbb{A} \cup \{\rightarrow\}$ as follows.

$$\begin{aligned} \ell(x) &\triangleq a, & \text{if } \xi(x) = \text{inl}(a) \text{ for some } a \in \mathbb{A}, \\ &\triangleq \rightarrow, & \text{otherwise.} \end{aligned}$$

For any $x \in X$ we have to construct a tree $\varphi(x)$ in such a way that the resulting mapping $\varphi : X \longrightarrow \text{Tr}_{\text{inf}}(\mathbb{A})$ is a $T_{\mathbb{A}}$ -coalgebra morphism. We define the corresponding partial function (see Definition 7E.1)

$$\varphi(x) : \{0, 1\}^* \rightharpoonup \mathbb{A} \cup \{\rightarrow\}$$

by induction on the length of the tree addresses.

$$\begin{aligned} \varphi(x)(\epsilon) &\triangleq \ell(x), \\ \varphi(x)(iw) &\triangleq \uparrow, & \text{if } \xi(x) = \text{inl}(a) \text{ for some } a \in \mathbb{A}, \\ &\triangleq \varphi(x_i)(w), & \text{if } \xi(x) = \text{inr}(\langle x_0, x_1 \rangle) \text{ for some } x_0, x_1 \in X. \end{aligned}$$

We leave to the reader the verification that φ is indeed the unique $T_{\mathbb{A}}$ -coalgebra morphism from X to Tr_{inf}^A . ■

7F.17. REMARK. A flat simultaneous recursion $\mathcal{R}(\vec{X})$, that is, one whose right-hand sides have either the shape $\alpha \in \mathbb{A}$ or the shape $X' \rightarrow X''$ for some $X', X'' \in \vec{X}$, may be seen directly as a $T_{\mathbb{A}}$ -coalgebra. Explicitly, $\mathcal{R}(\vec{X})$ corresponds to the $T_{\mathbb{A}}$ -coalgebra

$$f_{\mathcal{R}} : \vec{X} \longrightarrow \mathbb{A} + \vec{X} \times \vec{X}$$

defined by the following conditions, for any $X \in \vec{X}$:

$$\begin{aligned} f_{\mathcal{R}}(X) &\triangleq \text{inl}(\alpha), & \text{if } X = \alpha \in \mathbb{A}, \\ &\triangleq \text{inr}(\langle X', X'' \rangle), & \text{if } X = X' \rightarrow X''. \end{aligned}$$

The unique $T_{\mathbb{A}}$ -morphism from \vec{X} to $\text{Tr}_{\text{inf}}(\mathbb{A})$, which exists by finality, is easily seen to be the solution of $\mathcal{R}(\vec{X})$ in $\text{Tr}_{\text{inf}}(\mathbb{A})$.

Coinduction enables us to define maps in a ‘coordinate-free’ fashion, rather than in the way of Definition 7E.11.

7F.18. REMARK. The coalgebraic treatment of trees can also justify the format of 7E.12 as official definition. This is intuitively simpler than that given in Definition 7E.11. If \mathcal{A} is invertible, then there is an obvious map

$$\tau : |\mathcal{A}| \longrightarrow \|\mathcal{A}\| + (|\mathcal{A}| \times |\mathcal{A}|),$$

where the disjoint union matches the case distinction in the definition of function $(-)^*_{\mathcal{A}}$ in Lemma 7E.12. The existence of the unique morphism $(-)^*_{\mathcal{A}} : \mathcal{A} \rightarrow \text{Tr}_{\text{inf}}(\mathbb{A})$ follows then by finality of $\langle \text{Tr}_{\text{inf}}(\mathbb{A}), \omega \rangle$, where $\mathbb{A} = \|\mathcal{A}\|$.

We have seen already a direct proof of the following coinduction principle for trees, see Proposition 7E.10. A categorical proof avoids using argumentation involving maps. This fits with the ‘coordinate-free’ treatment of definitions and propositions, see Section 7E.

7F.19. PROPOSITION (Coinduction for trees). *Let R be a bisimulation over $\text{Tr}_{\text{inf}} \times \text{Tr}_{\text{inf}}$. Then*

$$\forall s, t \in \text{Tr}_{\text{inf}}. [s R t \Rightarrow s = t].$$

PROOF. The following categorical proof, exploiting the finality of the coalgebra of trees, is taken from [Rutten \[2000\]](#), Theorem 8.1. Observe that R is a bisimulation exactly when there is a function $\omega_R : R \longrightarrow T_{\mathbb{A}}(R)$ that makes the diagram

$$\begin{array}{ccccc} \text{Tr}_{\text{inf}} & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & \text{Tr}_{\text{inf}} \\ \omega \downarrow & & \omega_R \downarrow & & \downarrow \omega \\ T_{\mathbb{A}}(\text{Tr}_{\text{inf}}) & \xleftarrow{T_{\mathbb{A}}(\pi_1)} & T_{\mathbb{A}}(R) & \xrightarrow{T_{\mathbb{A}}(\pi_2)} & T_{\mathbb{A}}(\text{Tr}_{\text{inf}}) \end{array}$$

commute, where π_1 (resp. π_2) extracts the first (resp. the second) component of R . Then they both are morphisms to the final coalgebra $\langle \text{Tr}_{\text{inf}}, \omega \rangle$, therefore $\pi_1 = \pi_2$, and all pairs in R have identical components. ■

7G. Exercises

- 7G.1. Let $B = \mu\alpha.\beta \rightarrow \alpha$, with $\alpha \neq \beta$. Show that $\vdash_{\lambda\mu} \text{YK} : B$.
- 7G.2. Show that there is a term $M \in \Lambda_{\rightarrow}^{\emptyset}$ such that all occurrences of λ in M bind different variables, but M cannot be reduced naively, i.e. there is a β -reduct N of M with a redex $(\lambda x.P)Q$ in N such that

$$(\lambda x.P)Q \not\rightarrow_{\beta} P[x := Q]_{\not\in}.$$

[Hint. Consider a term of the form $M \equiv (\lambda x.x(Bx))\mathbf{c}_1$.]

- 7G.3. Let $h : \mathcal{A} \rightarrow \mathcal{S}$ be a morphism. Define the **kernel** of h , notation $\ker(h)$, as follows
 $\ker(h) \triangleq \{a = b \mid h(a) = h(b)\}.$
 - (i) Show that $\ker(h)$ is always a congruence relation and hence

$$(a = b) \in \ker(h) \Leftrightarrow a =_{\ker(h)} b.$$

- (ii) Let $h : \mathcal{A} \rightarrow \mathcal{S}$ be a morphism and \mathcal{E} be a set of equations over \mathcal{A} . Show

$$\mathcal{E} \subseteq \ker(h) \Leftrightarrow (=_{\mathcal{E}}) \subseteq \ker(h).$$

- 7G.4. Show that \mathcal{R} over $\mathbb{T}(\vec{X})$ justifies \mathcal{E} over \mathbb{T} in exactly one of the following cases. See also Exercise 8D.12.
- (i) $\mathcal{R} \triangleq \{X = \alpha \rightarrow \alpha \rightarrow X\};$
 $\mathcal{E} \triangleq \{\beta = \alpha \rightarrow \alpha \rightarrow \alpha \rightarrow \beta, \beta = \alpha \rightarrow \alpha \rightarrow \alpha \rightarrow \alpha \rightarrow \beta\}.$
 - (ii) $\mathcal{R} \triangleq \{X = \alpha \rightarrow \alpha \rightarrow X\}$
 $\mathcal{E} \triangleq \{\beta = \alpha \rightarrow \alpha \rightarrow \alpha \rightarrow \alpha \rightarrow \beta, \beta = \alpha \rightarrow \alpha \rightarrow \alpha \rightarrow \alpha \rightarrow \alpha \rightarrow \beta\}.$
- 7G.5. Let $\mathcal{E} = \mathcal{E}(\vec{X})$ be a set of equations over $\mathcal{A}(\vec{X})$. Show that the following are equivalent.
- (i) \mathcal{B} justifies \mathcal{E} .
 - (ii) $\mathcal{B} \models h^+(\mathcal{E})$, for some $h^+ : \mathcal{A}(\vec{X}) \rightarrow \mathcal{B}$,
 - (iii) $\mathcal{B} \models \exists \vec{X}. h(\mathcal{E})$, for some $h : \mathcal{A} \rightarrow \mathcal{B}$. Here an $h : \mathcal{A} \rightarrow \mathcal{B}$ is extended in the natural way to $h : \mathcal{A}(\vec{X}) \rightarrow \mathcal{B}(\vec{X})$. E.g. $h(a \rightarrow X) = h(a) \rightarrow X$.
 - (iv) There is a morphism $h^\sharp : \mathcal{A}[\mathcal{E}] \rightarrow \mathcal{B}$.
- 7G.6. Let \mathcal{E} be a binary relation on a type algebra \mathcal{A} . Show the following for the category of type algebras with their morphisms.
- (i) Show that a morphism $h : \mathcal{A}/\mathcal{E} \rightarrow \mathcal{S}$ is uniquely determined by a morphism $h^\sharp : \mathcal{A} \rightarrow \mathcal{S}$ such that $\mathcal{E} \subseteq \ker(h^\sharp)$ by $h([a]_\mathcal{E}) = h^\sharp(a)$.
 - (ii) The canonical map $[-]_\mathcal{E} : \mathcal{A} \rightarrow \mathcal{A}/\mathcal{E}$ is an epimorphism having as kernel $=_\mathcal{E}$. Conversely, if $h : \mathcal{A} \rightarrow \mathcal{S}$ is an epimorphism and \mathcal{E} is its kernel, then $\mathcal{S} \cong \mathcal{A}/\mathcal{E}$.
 - (iii) \mathcal{A}/\mathcal{E} is determined, up to isomorphism, as the initial object such that $[-]_\mathcal{E} : \mathcal{A} \rightarrow \mathcal{A}/\mathcal{E}$ has kernel \mathcal{E} . That is, if \mathcal{S} is such that there is a $h : \mathcal{A} \rightarrow \mathcal{S}$ with kernel \mathcal{E} , then there is a unique arrow $k : \mathcal{A}/\mathcal{E} \rightarrow \mathcal{S}$ such that $k \circ [-]_\mathcal{E} = h$.
- 7G.7. Let \mathcal{A}, \mathcal{B} be type algebras and let $h : \mathcal{A} \rightarrow \mathcal{B}$ be a morphism. Show
- (i) $\mathcal{B} \models h(\mathcal{E}) \Leftrightarrow \mathcal{E} \subseteq \ker(h)$.
 - (ii) \mathcal{B} justifies $\mathcal{E} \Leftrightarrow \exists h : \mathcal{A} \rightarrow \mathcal{B}. \mathcal{E} \subseteq \ker(h)$.
- 7G.8. Suppose there is a morphism $h : \mathcal{B} \rightarrow \mathcal{C}$. Show that if $\mathcal{R}[\vec{X}]$ is justified by \mathcal{B} , then also by \mathcal{C} .
- 7G.9. Show that for every regular tree $T \in \mathsf{Tr}_{\text{reg}}$ there exists a proper sr $\mathcal{R}(\vec{X})$ over \mathbb{T} such that $T = (X_1)_\mathcal{R}^*$. [Hint. Let T_1, \dots, T_n , with $T_1 = T$ be the distinct subtrees occurring in T and take $\vec{X} = X_1, \dots, X_n$.]
- 7G.10. Show that the relation $=^*$ introduced in Definition 7E.17(i) is the greatest invertible congruence on \mathbb{T} extending \approx and preserving all prime elements of \mathbb{T}/\approx .
- 7G.11. Show that $\|\mathbb{T}_\mu\| = \{[\alpha] \mid \alpha \in \mathbb{A}\} \cup \{[\mu t.t]\}$.
- 7G.12. Let U denote the type $\mu t.(t \rightarrow t)$. 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 .
- 7G.13. 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 $\lambda\mu$.

- 7G.14. Let \mathcal{A} be a type algebra with elements a, b such that $b = b \rightarrow a$. Provide the de Bruijn version of the terms in Proposition 7A.3: $\omega \triangleq \lambda x.xx$, $\Omega \triangleq \omega\omega$, $\Upsilon \triangleq \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$.

7G.15. 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 a reduction rule similar to $(R_{\mathcal{E}}^{\text{uf}})$ of Definition 7A.22 it can be proved that this has the same reduction properties as Y .

7G.16. Let $M \triangleq \lambda x \gamma \rightarrow \mu \alpha.(\beta \rightarrow \alpha).x \gamma \rightarrow \mu \alpha.(\beta \rightarrow \alpha) \in \Lambda_{\equiv}^{\mathcal{A}, \text{Ch}}((\gamma \rightarrow \mu \alpha.(\beta \rightarrow \alpha)) \rightarrow \gamma \rightarrow \mu \alpha.(\beta \rightarrow \alpha))$. Construct a term M' by adding to some η -expansion of M occurrences of *fold* or *unfold* such that $M' \in \Lambda_{\equiv}^{\mathcal{A}, \text{Ch}}((\gamma \rightarrow \mu \alpha.(\beta \rightarrow \alpha)) \rightarrow \gamma \rightarrow \mu \alpha.(\beta \rightarrow \alpha))$.

In general construct for each $M \in \Lambda_{\equiv}^{\mathcal{A}, \text{Ch}}(A)$ a term $M' \in \Lambda_{\equiv}^{\mathcal{A}, \text{Ch}_0}(A)$ such that $\widehat{M}' =_\eta M$, where \widehat{M}' is the term obtained from M' by erasing all occurrences of *fold* and *unfold*.

7G.17. (i) A type algebra \mathcal{A} is not always generated by its prime elements. [Hint. If $\mathcal{A} = \{A\}$ with $A = A \rightarrow A$, then $\|\mathcal{A}\| = \emptyset$.]
(ii) The embedding 7B.12(i) is not always surjective. [Hint. Use (i).]
(iii) A morphism $h: \mathcal{A} \rightarrow \mathcal{B}$ is not uniquely determined by $h \upharpoonright \|\mathcal{A}\|$.

7G.18. Let \mathcal{E} be a set of equations over the type algebra \mathcal{A} . We say that \mathcal{E} *satisfies* $A = B$, notation $\mathcal{E} \models A = B$, if for all type algebras \mathcal{C} and all morphisms $h: \mathcal{A} \rightarrow \mathcal{C}$ one has $\mathcal{C} \models h(\mathcal{E}) \Rightarrow \mathcal{C} \models h(A) = h(B)$. Show that

$$\mathcal{E} \models A = B \Leftrightarrow \mathcal{E} \vdash A = B.$$

7G.19. This elaborates some points in the proof of Proposition 7B.15.

(i) Show that the following is a ‘cut-free’ (no application of transitivity) axiomatization of $\vdash_{\mathcal{E}}$.

(refl)	$\frac{}{A = A};$
(\rightarrow)	$\frac{A_1 = B_1 \quad A_2 = B_2}{(A_1 \rightarrow A_2) = (B_1 \rightarrow B_2)};$
($\mathcal{E}[P, Q]$)	$\frac{A = P \quad B = Q}{A = B}, \quad \text{if } (P = Q) \in \mathcal{E} \text{ or } (Q = P) \in \mathcal{E}.$

(ii) Given a statement $A = B$, construct a tree (see Figure 22) that describes all possible attempts of backwards derivations of $A = B$, following the cut-free rules. The nodes are labeled with equations $P = Q$ (expressing an equation to be proved) or a symbol ‘ \rightarrow ’ 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 \rightarrow or $\mathcal{E}[P, Q]$ nodes.
2. A node with label $P = Q$ is provable if both targets of at least one of its (\rightarrow or $\mathcal{E}[P', Q']$) targets (an \rightarrow or an $\mathcal{E}[P', Q']$ node) are provable (in the cut-free version of $\vdash_{\mathcal{E}}$). [The $P = Q$ nodes are called ‘or-nodes’ (having dotted lines) and the \rightarrow and $\mathcal{E}[P, Q]$ are called ‘and-nodes’ (having solid lines).]

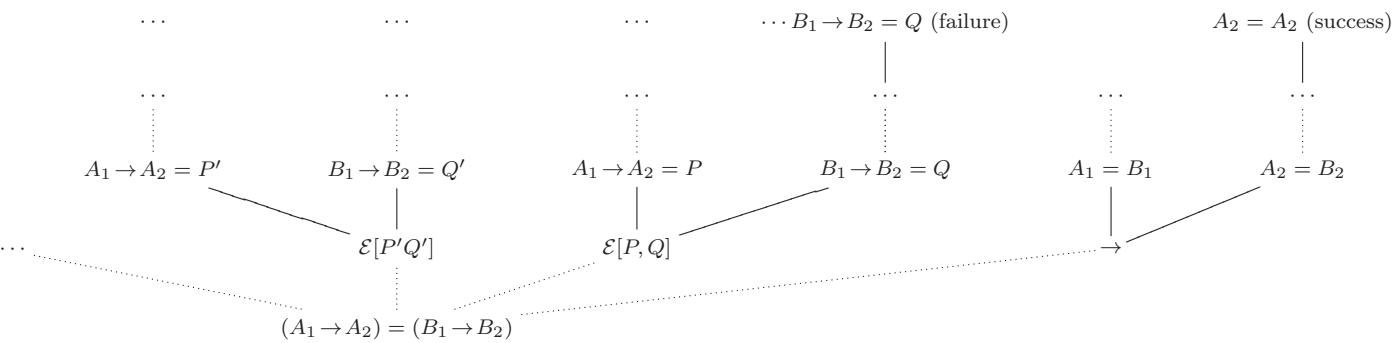


FIGURE 22. Exhaustive search for derivation in the cut-free version of \vdash_{ε} , in this case for the formula $A = B$ of the form $(A_1 \rightarrow A_2) = (B_1 \rightarrow B_2)$.

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.

7G.20. Prove proposition 7F.13.

[Hint. Take I as the direct limit (union with identifications via the $T^k(f)$)

$$0 \xrightarrow{f} T(0) \xrightarrow{T(f)} T^2(0) \xrightarrow{T^2(f)} \dots,$$

where $0 = \emptyset$ is the initial element of **Set** and $f:0 \rightarrow T(0)$ the canonical map. The inverse of the morphism **in** is defined by mapping an element of I , say **in** in $T^k(0)$, via $T^k(f)$ to the next level in I . This is an isomorphism.]

7G.21. Let $T : \text{Set} \rightarrow \text{Set}$ be a functor and let $\langle C, \text{out} \rangle$ be the final T -coalgebra. Prove the following.

- (i) For all $\alpha:X \rightarrow T(X)$ there exists a unique $f:X \rightarrow C$ such that the following diagram commutes

$$\begin{array}{ccc} X & \xrightarrow{\alpha} & T(X) \\ f \downarrow & & \downarrow T(f) \\ C & \xrightarrow{\text{out}} & T(C) \end{array}$$

FIGURE 23. Coiteration

- (ii) For all $\alpha:X \rightarrow T(C + X)$ there exists a unique $f:X \rightarrow C$ such that the following diagram commutes

$$\begin{array}{ccc} T(C) & \xleftarrow{\text{out}} & C \\ T([\text{id}_C, f]) \uparrow & & \uparrow f \\ T(C + X) & \xleftarrow{\alpha} & X \end{array}$$

FIGURE 24. Primitive corecursion

[Hint. Verify that there exists a $g:C + X \rightarrow C$ such that the following diagram commutes

$$\begin{array}{ccc} C + X & \xrightarrow{T(\text{nil}) \circ \text{out} \circ \alpha} & T(C + X) \\ g \downarrow & & \downarrow T(g) \\ C & \xrightarrow{\text{out}} & T(C) \end{array}$$

Show that $g = [\mathbf{1}_C, g \circ \text{nil}]$, i.e. $g \circ \text{nil} = \mathbf{1}_C$.]

7G.22. The collection of **streams** over \mathbb{A} is $\mathbb{S}^{\mathbb{A}}$. An element $s \in \mathbb{S}^{\mathbb{A}}$ is often written as $s = \langle s_0, s_1, s_2, \dots \rangle$, with $s_i = s(i)$. There are two basic operations ('head' and

‘tail’) on $\mathbb{S}^{\mathbb{A}}$ with $\text{hd} : \mathbb{S}^{\mathbb{A}} \rightarrow \mathbb{A}$; $\text{tl} : \mathbb{S}^{\mathbb{A}} \rightarrow \mathbb{S}^{\mathbb{A}}$. defined by

$$\begin{aligned}\text{hd}(\langle s_0, s_1, s_2, \dots \rangle) &\triangleq s_0; \\ \text{tl}(\langle s_0, s_1, s_2, \dots \rangle) &\triangleq \langle s_1, s_2, s_3, \dots \rangle.\end{aligned}$$

The stream $\text{tl}(s)$ is also called the *derivative* of s and is denoted by s' . One has $t = s$ iff $\forall n \in \mathbb{N}. s(n) = t(n)$, for $s, t \in \mathbb{S}$.

- (i) Show that $\mathbb{S}^{\mathbb{A}}$ is the final coalgebra of the functor $T_{\mathbb{S}^{\mathbb{A}}}(X) = \mathbb{A} \times X$.
- (ii) A relation $R \subseteq \mathbb{S}^{\mathbb{A}} \times \mathbb{S}^{\mathbb{A}}$ is called a *bisimulation* iff for all $s, t \in \mathbb{S}^{\mathbb{A}}$

$$sRt \Rightarrow s_0 = t_0 \ \& \ s'Rt'. \quad (\text{ci})$$

Show the principle of coinduction for streams: for R a bisimulation over $\mathbb{S}^{\mathbb{A}}$, and all $s, t \in \mathbb{S}^{\mathbb{A}}$

$$sRt \Rightarrow s = t.$$

- 7G.23. Define maps $\text{even}, \text{odd} : \mathbb{S}^{\mathbb{A}} \rightarrow \mathbb{S}^{\mathbb{A}}$ and $\text{zip} : \mathbb{S}^{\mathbb{A}} \times \mathbb{S}^{\mathbb{A}} \rightarrow \mathbb{S}^{\mathbb{A}}$ as follows, *by corecursion*.

$$\begin{aligned}(\text{even}(s))(0) &\triangleq s(0) \\ (\text{even}(s))' &\triangleq \text{even}(s'') \\ (\text{odd}(s))(0) &\triangleq s(1) \\ (\text{odd}(s))' &\triangleq \text{odd}(s'') \\ (\text{zip}(s, t))(0) &\triangleq s(0) \\ (\text{zip}(s, t))' &\triangleq \text{zip}(t, s').\end{aligned}$$

This has the effect that

$$\begin{aligned}\text{even}(\langle a_0, a_1, \dots \rangle) &= (\langle a_0, a_2, \dots \rangle) \\ \text{odd}(\langle a_0, a_1, \dots \rangle) &= (\langle a_1, a_3, \dots \rangle) \\ \text{zip}(\langle a_0, a_1, \dots \rangle, \langle b_0, b_1, \dots \rangle) &= (\langle a_0, b_0, a_1, b_1, \dots \rangle).\end{aligned}$$

Show that, for all $s, t \in \mathbb{S}^{\mathbb{A}}$, one has

$$\begin{aligned}\text{even}(\text{zip}(s, t)) &= s; \\ \text{odd}(\text{zip}(s, t)) &= t.\end{aligned}$$

[Hint. Show that $R = \{(\text{even}(\text{zip}(s, t)), s) \mid s, t \in \mathbb{S}\}$ is a bisimulation.]

- 7G.24. Show that the maps $\text{even}, \text{odd} : \mathbb{S}^{\mathbb{A}} \rightarrow \mathbb{S}^{\mathbb{A}}$ and $\text{zip} : \mathbb{S}^{\mathbb{A}^2} \rightarrow \mathbb{S}^{\mathbb{A}}$ can be defined by coiteration.

- 7G.25. Using (a slightly modified form of) coinduction, show that

$$\forall s \in \mathbb{S}^{\mathbb{A}}. \text{zip}(\text{even}(s), \text{odd}(s)) = s.$$

- 7G.26. Let $F, G : \mathbb{S}^{\mathbb{A}} \rightarrow \mathbb{S}^{\mathbb{A}}$. Suppose

$$\forall s \in \mathbb{S}. (F s)(0) = (G s)(0) \ \& \ (F s)' = F(s') \ \& \ (G s)' = G(s').$$

Show by coinduction that

$$\forall s \in \mathbb{S}. F s = G s.$$

7G.27. Define $\text{map} : (\mathbb{A} \rightarrow \mathbb{A}) \rightarrow \mathbb{S}^{\mathbb{A}} \rightarrow \mathbb{S}^{\mathbb{A}}$ by

$$\begin{aligned}(\text{map } f \ s)(0) &\triangleq f(s(0)), \\ (\text{map } f \ s)' &\triangleq \text{map } f \ s'\end{aligned}$$

Using Exercise 7G.26 show that

$$\forall f, g \in \mathbb{A} \rightarrow \mathbb{A} \forall s \in \mathbb{S}. \text{map } g \ (\text{map } f \ s) = \text{map } (g \circ f) \ s.$$

7G.28. Construct by primitive corecursion a map $g : \mathbb{S}^{\mathbb{A}} \rightarrow \mathbb{S}^{\mathbb{A}}$ such that

$$g(s) = \langle 0, s_0, s_1, s_2, \dots \rangle.$$

7G.29. (Rutten [2005]) For $s, t \in \mathbb{S}$ define the sum and convolution-product corecursively as follows.

$$\begin{aligned}(s + t)(0) &\triangleq s(0) + t(0), \\ (s + t)' &\triangleq s' + t'; \\ (s \times t)(0) &\triangleq s(0) * t(0), \\ (s \times t)' &\triangleq (s' \times t) + (s \times t').\end{aligned}$$

(i) Show the following for all $s, t, u \in \mathbb{S}$.

$$\begin{array}{lll} \text{(i)} & s + t &= t + s; \\ \text{(ii)} & (s + t) + u &= s + (t + u); \\ \text{(iii)} & (s + t) \times u &= (s \times u) + (t \times u); \\ \text{(iv)} & s \times t &= t \times s; \\ \text{(v)} & (s \times t) \times u &= s \times (t \times u). \end{array}$$

[Hint. Use coinduction.]

(ii) Show that

$$\langle a_0, a_1, a_2, \dots \rangle \times \langle b_0, b_1, b_2, \dots \rangle = \langle c_0, c_1, c_2, \dots \rangle,$$

with $c_n = \sum_{k=0}^n \binom{n}{k} a_k b_{n-k}$.

(iii) Show that

$$\langle 1, r, r^2, \dots \rangle \times \langle 1, s, s^2, \dots \rangle = \langle 1, r+s, (r+s)^2, \dots \rangle.$$

[Hint. Use (ii). Alternatively, define $GS(r) = \langle 1, r, r^2, \dots \rangle$ using corecursion and show $GS(r) \times GS(s) = GS(r+s)$, using coinduction.]

7G.30. Define the coalgebra *lazy lists* consisting of finite and infinite lists. Is there a polynomial functor for which this is the final coalgebra?

7G.31. Show that $\mu\alpha.(\alpha \rightarrow \mu\beta.\beta) \equiv_{\alpha} \mu\beta.(\beta \rightarrow \mu\beta.\beta)$, in spite of the restriction on variables in the axioms (α -conv).

7G.32. [Klop] (i) Draw the complete μ -reduction graph of $A = \mu\alpha\beta\gamma.\beta$, i.e. the set $\{B \mid A \Rightarrow_{\mu}^{*} B\}$ directed by \Rightarrow_{μ} . [Warning. The set has more than 2010 elements.]

(ii) Do the same for the type $A_{n,k} = \mu\alpha_1 \cdots \alpha_n.\alpha_k$.

CHAPTER 8

PROPERTIES OF RECURSIVE TYPES

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 Section 8A we build a solution of an sr using μ -types, and show the essential equivalence of the two notations for recursive types. There are, nevertheless, aspects which are treated more naturally in one approach than in the other. For instance, In Chapter 9, the notion of principal type scheme is formulated and studied in a more natural way using simultaneous recursion. In Sections 8B, 8C we show the decidability of weak and strong equality for μ -types.

8A. Simultaneous recursions vs μ -types

The μ -types and the recursive types defined via an sr turn out to be closely related.

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 $\mathbb{T} = \mathbb{T}^{\mathbb{A}}$. We show that $\mathbb{T}_\mu \models \exists \vec{X}. \mathcal{R}(\vec{X})$, for $\mathbb{T}_\mu = \mathbb{T}_\mu^{\mathbb{A}}$. We do this in a constructive way by building an n -tuple of types $S_1, \dots, S_n \in \mathbb{T}_\mu$ such that $\mathbb{T}_\mu \models \mathcal{R}(\vec{S})$. This means that

$$\begin{aligned} S_1 &=_{\mu} C_1[\vec{X} := \vec{S}] \\ &\dots \\ S_n &=_{\mu} C_n[\vec{X} := \vec{S}], \end{aligned}$$

if \mathcal{R} is of the form

$$\begin{aligned} X_1 &= C_1(\vec{X}) \\ &\dots \\ X_n &= C_n(\vec{X}), \end{aligned} \tag{1}$$

with the $C_i(\vec{X}) \in \mathbb{T}[\vec{X}]$. The following construction is taken from [Bekić \[1984\]](#), p. 39 ‘Bisection Lemma’, and is known as ‘folklore’ of the subject.

8A.1. THEOREM. *Let $\mathcal{R} = \mathcal{R}(\vec{X})$ be an sr over $\mathbb{T} = \mathbb{T}^{\mathbb{A}}$ as (1) above.*

(i) *There is a morphism $h : \mathbb{T}(\vec{X}) \rightarrow \mathbb{T}_\mu$, leaving \mathbb{T} fixed, i.e. $h(A) = A$ for $A \in \mathbb{T}$, such that*

$$S_1 = h(X_1), \dots, S_n = h(X_n)$$

is a solution of $\mathcal{R}(\vec{X})$ in \mathbb{T}_μ . This means that for each $1 \leq i \leq n$

$$S_i =_\mu C_i[\vec{X} := \vec{S}].$$

(ii) *Therefore $\mathbb{T}_\mu \models \exists \vec{X}. \mathcal{R}(\vec{X})$, i.e. \mathbb{T}_μ solves \mathcal{R} .*

PROOF. (i) First define the types $\vec{P} = P_1, \dots, P_n \in \mathbb{T}_\mu$ for $1 \leq i \leq n$.

$$\begin{aligned} P_1 &\triangleq \mu X_1.C_1 \\ P_2 &\triangleq \mu X_2.(C_2[X_1 := P_1]) \\ &\vdots \\ P_n &\triangleq \mu X_n.(C_n[X_1 := P_1] \cdots [X_{n-1} := P_{n-1}]). \end{aligned}$$

Then define the types $\vec{S} = S_1, \dots, S_n \in \mathbb{T}_\mu$ by

$$\begin{aligned} S_n &\triangleq P_n \\ S_{n-1} &\triangleq P_{n-1}[X_n := S_n] \\ &\vdots \\ S_1 &\triangleq P_1[X_2 := S_2, \dots, X_n := S_n]. \end{aligned}$$

Finally set

$$\begin{aligned} h(\alpha) &\triangleq \alpha, & \text{if } \alpha \in \mathbb{A}, \\ h(X_i) &\triangleq S_i, \\ h(A \rightarrow B) &\triangleq h(A) \rightarrow h(B). \end{aligned}$$

Clearly h is a morphism leaving \mathbb{T} fixed.

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 B[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 construction of the \vec{S} as follows.

$$\begin{aligned} P_1^{X_2} &= \mu X_1.C_1(X_1, X_2), \\ P_2 &= \mu X_2.C_2(P_1^{X_2}, X_2), \\ S_2 &= P_2, \\ S_1 &= P_1^{S_2}. \end{aligned}$$

Then

$$\begin{aligned} S_2 &= P_2 \\ &= C_2(P_1^{P_2}, P_2) \\ &= C_2(S_1, S_2); \\ S_1 &= P_1^{S_2} \\ &= C_1(S_1, S_2). \end{aligned}$$

(ii) By (i) and Definition 7C.4. Here we tacitly extend h to a morphism $h : \mathbb{T}_\mu(\vec{X}) \rightarrow \mathbb{T}_\mu$, leaving \mathbb{T}_μ fixed. ■

8A.2. EXAMPLE. Take the following sr \mathcal{R}_1 :

$$\begin{array}{rcl} X_1 & = & X_2 \rightarrow X_1 \\ X_2 & = & X_1 \rightarrow X_2 \end{array}$$

Applying Definition 8A.1 we have:

$$\begin{array}{rcl} P_1 & = & \mu X_1.(X_2 \rightarrow X_1) \\ P_2 & = & \mu X_2.((\mu X_1.(X_2 \rightarrow X_1)) \rightarrow X_2) \end{array}$$

and then

$$\begin{array}{rcl} S_2 & = & \mu X_2.((\mu X_1.(X_2 \rightarrow X_1)) \rightarrow X_2) \\ S_1 & = & \mu X_1.(\mu X_2.((\mu X'_1.(X_2 \rightarrow X'_1)) \rightarrow X_2) \rightarrow X_1) \end{array}$$

It is easy to check that $S_1 =_\mu S_2 \rightarrow S_1$ and $S_2 =_\mu S_1 \rightarrow S_2$, but note that $S_1 \neq_\mu S_2$ as can be proved with the technique developed in Section 8B.

8A.3. REMARK. $\langle S_1, \dots, S_n \rangle$ is not a unique solution in \mathbb{T}_μ (modulo $=_\mu$). In Example 8A.2 both the pair $\langle S_2, S_1 \rangle$, as well as $\langle S, S \rangle$, where $S = \mu X.X \rightarrow X$, solve \mathcal{R}_1 . Note, however, that $(S_1)^*_\mu = (S_2)^*_\mu = (S)^*_\mu$.

Note that in this proof all the needed type equivalences can be obtained by axiom (μ -eq). No other rules of Definition 7D.26 (for instance (\rightarrow -cong)) have been used. So the solution is one in a rather strong sense.

8A.4. COROLLARY. (i) For the morphism $h : \mathbb{T}[\vec{X}] \rightarrow \mathbb{T}_\mu$ of Theorem 8A.1 one has for all $A, B \in \mathbb{T}(\vec{X})$

$$A =_{\mathcal{R}} B \Rightarrow h(A) =_\mu h(B).$$

(ii) Therefore h induces a morphism $h^\sharp : \mathbb{T}[\mathcal{R}] \rightarrow \mathbb{T}_\mu$, satisfying $h^\sharp([\alpha]) = \alpha$ for $\alpha \in \mathbb{A}$ and $h^\sharp([A]) = h(A)$, for $A \in \mathbb{T}(\vec{X})$.

(iii) $\vdash_{\mathbb{T}^{\mathbb{A}}[\mathcal{R}]} M : A \Rightarrow \vdash_{\mathbb{T}_\mu} M : h^\sharp(A)$.

PROOF. (i) By induction on the derivation of $A =_{\mathcal{R}} B$.

(ii) By Proposition 7B.4(i).

(iii) By Lemma 7A.20. ■

8A.5. REMARK. In general the reverse of Corollary 8A.4(i) is not valid. Take e.g.

$$\mathcal{R}_0 \triangleq \{X_1 = X_1 \rightarrow X_1, X_2 = X_2 \rightarrow X_2\}.$$

Applying the construction of Theorem 8A.1 we obtain $S_1 = S_2 = \mu X.(X \rightarrow X)$, but $X_1 \neq_{\mathcal{R}_0} X_2$.

In the case of solutions modulo strong equivalence there are other constructions in the literature which give simpler solutions, see e.g. [Ariola and Klop \[1996\]](#).

From μ -type to sr

In the opposite direction, any μ -type can be represented by an equivalent sr. While the idea underlying this property is intuitively simple, a formal definition in the case of weak equivalence becomes complicated and not especially interesting. In the case

of strong equivalence, however, this notion can be easily formalized by exploiting the interpretation of types as infinite trees.

We define for each $A \in \mathbb{T}_\mu^{\mathbb{A}}$ an sr \mathcal{R} and a $B \in \mathbb{T}^{\mathbb{A}}$ such that $(B)_{\mathcal{R}}^*$ is the same infinite tree as $(A)_\mu^*$. 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 .

8A.6. DEFINITION. We associate to a $A \in \mathbb{T}_\mu$ a type $\mathcal{T}(A) \in \mathbb{T}^{\mathbb{A}}$ and an sr $\mathcal{SR}(A)$ over $\mathbb{T}^{\mathbb{A}}$ defined in the following way.

$$\begin{array}{lll} \mathcal{T}(\alpha) & \triangleq & \alpha \\ \mathcal{T}(A \rightarrow B) & \triangleq & \mathcal{T}(A) \rightarrow \mathcal{T}(B) \\ \mathcal{T}(\mu\alpha A) & \triangleq & \alpha \end{array} \quad \begin{array}{lll} \mathcal{SR}(\alpha) & \triangleq & \emptyset, \quad \text{for } \alpha \in \mathbb{A}, \\ \mathcal{SR}(A \rightarrow B) & \triangleq & \mathcal{SR}(A) \cup \mathcal{SR}(B) \\ \mathcal{SR}(\mu\alpha A) & \triangleq & \mathcal{SR}(A) \cup \{\alpha = \mathcal{T}(A)\}. \end{array}$$

8A.7. THEOREM. $(A)_\mu^* = (\mathcal{T}(A))_{\mathcal{SR}(A)}^*$

PROOF. See Exercise 8D.1. ■

8B. Properties of μ -types

In this section we investigate properties of μ -types both under the weak and the strong equivalence. In particular we show that both relations are invertible and decidable.

Invertibility and decidability of weak equivalence

Invertibility and decidability will be proved using the notion of reduction \Rightarrow_μ , defined in Definition 7D.28. Note that \Rightarrow_μ is not normalizing: types can be infinitely unfolded. However, the CR property, Proposition 7D.29(ii), suffices to prove invertibility.

8B.1. THEOREM. *The relation $=_\mu$ on \mathbb{T}_μ is invertible, i.e. for all $A_1, A_2, B_1, B_2 \in \mathbb{T}_\mu$*

$$A_1 \rightarrow B_1 =_\mu A_2 \rightarrow B_2 \Rightarrow A_1 =_\mu A_2 \& B_1 =_\mu B_2.$$

PROOF. Assume $A_1 \rightarrow B_1 =_\mu A_2 \rightarrow B_2$. By Lemma 7D.29(ii) there is a type C such that $A_1 \rightarrow B_1 \Rightarrow_\mu^* C$ and $A_2 \rightarrow B_2 \Rightarrow_\mu^* C$. But then we must have $C \equiv C_1 \rightarrow C_2$ with $A_i \Rightarrow_\mu^* C_1$ and $B_i \Rightarrow_\mu^* C_2$ for $i = 1, 2$. By 7D.29(i) this implies $A_1 =_\mu A_2$ and $B_1 =_\mu B_2$. ■

Decidability of weak equivalence

The relation $=_\mu$ is also decidable. The proof occupies 8B.3-8B.32. This result is relevant, as one of the typing rules in $\lambda\mu$, see Definition 7E.38, is

$$\vdash_{\lambda\mu} M : A, A =_\mu B \Rightarrow \vdash_{\lambda\mu} M : B.$$

Hence decidability of $=_\mu$ is needed to show the system is recursively axiomatizable.

Surprisingly decidability is not so easy to prove, due to the transitivity of equality (a ‘cut’-rule) and the presence of α -conversion. This makes proof search potentially undecidable, as infinitely many types may have to be tried. In Cardone and Coppo [2003] a proof system without cut-rule was presented. In Endrullis, Grabmayer, Klop, and van Oostrom [2011] it is pointed out that one also needs to be careful with α -conversion, as this may introduce infinitely many candidates in a proof search. This paper gives three different proofs of the decidability of $=_\mu$, two of these based on Cardone and

Coppo [2003] and the third on tree-regular languages. The elementary proof below is a simplification of the first proof of Endrullis et al. It uses a simpler bookkeeping device to give an upper bound to the number of nodes in a derivation tree for $A =_{\mu} B$.

8B.2. REMARK. One obvious attempt to prove decidability of weak equivalence is that of introducing a reduction relation \Rightarrow_{μ}^{-1} , defined by the contraction rule

$$A[\alpha := \mu\alpha.A] \Rightarrow_{\mu}^{-1} \mu\alpha.A.$$

If the relation \Rightarrow_{μ}^{-1} were Church-Rosser, then each $=_{\mu}$ -equivalence class would have a unique minimal representative reachable in a finite number of steps. From this decidability of $=_{\mu}$ would follow. Unfortunately \Rightarrow_{μ}^{-1} is not Church-Rosser, see Exercise 8D.6, therefore this proof strategy fails.

The cut-rule (trans) in the definition of system (μ) in Fig 21 causes problems in the proof of the decidability. In searching a proof of $A = C$ one must find a type B such that $A = B$ and $B = C$. But in general there are many possible choices for B . In Cardone and Coppo [2003] another proofsystem (μ^-) , without rule (trans), is defined that turns out to be equivalent with the proofsystem (μ) .

8B.3. DEFINITION. (The system (μ^-))

(axiom)	$A = A$
(left μ -step)	$\frac{A_1[\alpha := \mu\alpha.A_1] = B}{\mu\alpha.A_1 = B}$
(right μ -step)	$\frac{A = B_1[\alpha := \mu\alpha.B_1]}{A = \mu\alpha.B_1}$
(μ -cong)	$\frac{A_1 = B_1}{\mu\alpha.A_1 = \mu\alpha.B_1}$
(\rightarrow -cong)	$\frac{A_1 = B_1 \quad A_2 = B_2}{A_1 \rightarrow A_2 = B_1 \rightarrow B_2}$

Write $\vdash_{\mu^-} A = B$ if $A = B$ is provable in the system (μ^-) .

In 8B.4-8B.9 it will be shown that

$$A =_{\mu} B \Leftrightarrow \vdash_{\mu^-} A = B.$$

8B.4. DEFINITION. (Standard reduction \Rightarrow_{st}^*)

(i) Consider the \Rightarrow_{μ} -reduction sequence

$$R : A_1 \Rightarrow_{\mu} A_2 \Rightarrow_{\mu} \dots \Rightarrow_{\mu} A_n.$$

In each step $A_i \Rightarrow_{\mu} A_{i+1}$ mark all μ occurring in A_i to the left of the contracted μ with a symbol \diamond . Then R is called a *standard \Rightarrow_{μ} -reduction* if only non-marked μ are contracted in R .

(ii) Write $A \Rightarrow_{st}^* B$ if there is a standard reduction $R : A \Rightarrow_{\mu}^* B$.

8B.5. DEFINITION. Let the length $l(A)$ of a type A be defined inductively by

$$l(\alpha) = 1, \quad l(A_1 \rightarrow A_2) = 1 + l(A_1) + l(A_2), \quad l(\mu\alpha.A_1) = 1 + l(A_1).$$

8B.6. LEMMA. *The relation \Rightarrow_{st}^* is completely determined as follows.*

- (a) $A \Rightarrow_{st}^* A$.
- (b) If $A_1[\alpha := \mu\alpha.A_1] \Rightarrow_{st}^* B$, then $\mu\alpha.A_1 \Rightarrow_{st}^* B$.
- (c) If $A_1 \Rightarrow_{st}^* B_1$, then $\mu\alpha.A_1 \Rightarrow_{st}^* \mu\alpha.B_1$.
- (d) If $A_1 \Rightarrow_{st}^* A'_1$ and $A_2 \Rightarrow_{st}^* A'_2$, then $(A_1 \rightarrow A_2) \Rightarrow_{st}^* (A'_1 \rightarrow A'_2)$.

PROOF. Write $A \Rightarrow'_{st} B$ if (A, B) is in the least relation satisfying (a)-(d). By induction on the generation of \Rightarrow'_{st} one has

$$(A \Rightarrow'_{st} B) \Rightarrow (A \Rightarrow_{st}^* B).$$

Conversely, suppose $A \Rightarrow_{st}^* B$. Then there exist a standard reduction $A \Rightarrow^* B$. By induction on (n, l) , i.e. on the ordinal $\omega n + l$, where n is the length of this reduction and $l = l(A)$, we show $A \Rightarrow'_{st} B$. The only interesting cases are the following.

Case 1. $A \equiv A_1 \rightarrow A_2$ then $B \equiv B_1 \rightarrow B_2$ and $A_1 \Rightarrow_{st}^* B_1, A_2 \Rightarrow_{st}^* B_2$. By the induction hypothesis $A_i \Rightarrow'_{st} B_i$, hence $A_1 \rightarrow A_2 \Rightarrow'_{st} B_1 \rightarrow B_2$.

Case 2. $A \equiv \mu\alpha.A_1$. Subcase 2.1 $\mu\alpha.A_1 \Rightarrow_\mu A_1[\alpha := \mu\alpha.A_1] \Rightarrow_{st}^* B$. By the induction hypothesis $A_1[\alpha := \mu\alpha.A_1] \Rightarrow'_{st} B$ and hence $A \Rightarrow'_{st} B$, by clause (b). Subcase 2.2 $A \Rightarrow_{st}^* B$ is $\mu\alpha.A_1 \Rightarrow_\mu \mu\alpha.A_2 \Rightarrow_{st}^* B$. Then $B \equiv \mu\alpha.A_{n+1}$ and $A_1 \Rightarrow_{st}^* A_{n+1}$. By the induction hypothesis $A_1 \Rightarrow'_{st} A_{n+1}$ hence $A \Rightarrow'_{st} B$. ■

8B.7. EXAMPLE. Let $A = \mu\alpha\mu\beta.C[\alpha, \beta]$, where $C[\alpha, \beta]$ is a type in which possibly α and β occur several times. Let $B = \mu\alpha.C[\alpha, \mu\beta.C[\alpha, \beta]]$. Consider

- (i) $A \Rightarrow_\mu B \Rightarrow_\mu C[B, \mu\beta.C[B, \beta]]$.
- (ii) $A \Rightarrow_\mu \mu\beta.C[A, \beta] \Rightarrow_\mu C[A, \mu\beta.C[A, \beta]] \Rightarrow_\mu C[B, \mu\beta.C[A, \beta]] \Rightarrow_\mu C[B, \mu\beta.C[B, \beta]]$.

Then (i) is not a standard reduction, but (ii) is.

The following standardization theorem for \Rightarrow_μ^* can be obtained as a particular case of the standardization theorem for CRS. For a proof see J.W. Klop, V. van Oostrom, and F. van Raamsdonk [1993].

8B.8. LEMMA. *If $A \Rightarrow_\mu^* B$ then $A \Rightarrow_{st}^* B$.*

8B.9. COROLLARY. $A =_\mu B \Leftrightarrow A$ and B have a common standard reduct.

PROOF. Use that the reduction relation \Rightarrow_μ^* is Church Rosser. ■

8B.10. PROPOSITION. $\vdash_{\mu^-} A = B \Leftrightarrow A$ and B have a common standard reduct.

PROOF. (\Rightarrow) By induction on the derivation of $\vdash_{\mu^-} A = B$, using Lemma 8B.6.

(\Leftarrow) By Lemma 8B.6 we can distinguish the following cases.

Case (a) for both A and B , i.e. $A \equiv C$ and $B \equiv C$.

Case (b) for A (and similarly for B).

Case (c) for A and for B .

Case (d) for A and for B .

This suffices. If we have Case (c) for A and Case (a) for B , then in fact we have Case (c) for both A and B . If we have Case (d) for A and Case (a) for B , then (d) holds for both A and B . Finally, Case (c) for A and Case (d) for B (or vice versa) cannot occur at the same time. In all these four cases we easily conclude $\vdash_{\mu^-} A = B$. ■

8B.11. COROLLARY. $A =_\mu B \Leftrightarrow \vdash_{\mu^-} A = B$.

PROOF. By Corollary 8B.9. ■

Items 8B.3–8B.11 all come from Cardone and Coppo [2003], but their proof of the decidability of $\vdash_{\mu^-} A = B$ incorrectly treated rule (μ -cong).

8B.12. LEMMA. *In a branch of a proof search tree for $\vdash_{\mu^-} A = B$ the number of different nodes is at most*

$$3^{l(A)+l(B)}.$$

PROOF. The proof of this lemma occupies 8B.16–8B.32. ■

8B.13. COROLLARY. *In an exhaustive proof search tree for $\vdash_{\mu^-} A = B$ the number of different nodes is at most*

$$3^{(3^{l(A)+l(B)})}.$$

PROOF. Easy. Note that each node in the tree has at most three daughters. ■

8B.14. COROLLARY. $\vdash_{\mu^-} A = B$ is decidable.

8B.15. THEOREM. $A =_{\mu} B$ is decidable.

PROOF. By the preceding corollary and Corollary 8B.11. ■

In the proof of Lemma 8B.12 the rule (μ -cong) of the proofs system (μ^-) , i.e.

$$\frac{A_1 = B_1}{\mu\alpha.A_1 = \mu\alpha.B_1},$$

gives problems in case $\alpha \in \text{FV}(A_1)$, $\alpha \in \text{FV}(B_1)$. In a proof attempt for $\vdash_{\mu^-} C = D$ one may encounter this rule. Then a proof of $\vdash_{\mu^-} \mu\alpha.A_1 = \mu\alpha.B_1$ follows from a proof of $\vdash_{\mu^-} A_1 = B_1$. But there is an ambiguity here because the equation $\mu\alpha.A_1 = \mu\alpha.B_1$ is the same equation as $\mu\beta.A_1[\alpha := \beta] = \mu\beta.B_1[\alpha := \beta]$ for a fresh variable β and so one could also look for a proof of $\vdash_{\mu^-} A_1[\alpha := \beta] = B_1[\alpha := \beta]$. So we must treat the change of a free variable α by a fresh variable β in A_1 and B_1 at the same time.

For that we define yet another proof system (μ') where the two equations $A_1 = B_1$ and $A_1[\alpha := \beta] = B_1[\alpha := \beta]$ are identified. In case $\alpha \in \text{FV}(A_1) \cup \text{FV}(B_1)$ we replace rule (μ -cong) by

$$\frac{(\alpha)A_1 = (\alpha)B_1}{\mu\alpha.A_1 = \mu\alpha.B_1}.$$

Here $(\alpha)A_1$ and $(\alpha)B_1$ are no longer redexes, but α is not thrown away. The γ in $(\gamma)A$ is considered to be bound and we work modulo α -conversion, $(\gamma)A =_{\alpha} (\delta)A.[\gamma := \delta]$, so the equation $(\alpha)A_1 = (\alpha)B_1$ is the same equation as $(\beta)A_1[\alpha := \beta] = (\beta)B_1[\alpha := \beta]$.

The system (μ') will turn out to be equivalent with the system (μ^-) in the following strong sense. Each proof (attempt) of $\vdash_{\mu^-} A = B$ corresponds to a proof (attempt) of $\vdash_{\mu'} A = B$ of the same structure and vice versa. We will show that Lemma 8B.12 holds with \vdash_{μ^-} replaced by $\vdash_{\mu'}$ and then it clearly also holds for \vdash_{μ^-} itself and we are ready. Now we give the precise definition of the system (μ') .

The inspiration for the use of so called annotated types in the following came from the first proof in Endrullis, Grabmayer, Klop, and van Oostrom [2011]. In this proof the set $\text{AnnTer}(\mu) = \{(\mu\beta_1) \cdots (\mu\beta_n)A \mid A \in \mathbb{T}_{\mu}\}$ is defined. By requiring that the $\vec{\beta}$ are all distinct, that $\beta_i \in \text{FV}(A)$, and working modulo α -conversion the proof below will run more smoothly.

8B.16. DEFINITION (Annotated types $\mathbb{T}_{\mu'}$). (i) The set of *annotated types* is defined as

$$\mathbb{T}_{\mu'} \equiv \{(\beta_1) \cdots (\beta_n)A \mid A \in \mathbb{T}_\mu \text{ with } \beta_i \in FV(A) \text{ and with the } \vec{\beta} \text{ all distinct}\}.$$

Note that $\mathbb{T}_\mu \subset \mathbb{T}_{\mu'}$.

(ii) The β_i in $(\beta_1) \cdots (\beta_n)A$ are bound and we consider the annotated types modulo α -equivalence as follows

$$(\beta_1) \cdots (\beta_n)A \equiv (\beta'_1) \cdots (\beta'_n)A[\beta_i := \beta'_i],$$

where the $\beta'_1, \dots, \beta'_n$ are distinct and fresh, i.e. $\beta'_i \notin (FV(A) - \{\beta_1, \dots, \beta_n\}) \cup BV(A)$.

8B.17. NOTATION. (i) For $(\beta_1) \cdots (\beta_n)A$ we write $(\beta_1 \cdots \beta_n)A$ and often also $(\vec{\beta})A$.

(ii) Let a, b, c, \dots range over $\mathbb{T}_{\mu'}$ and as before A, B, C, \dots over \mathbb{T}_μ .

(iii) If $\vec{\beta}$ is a sequence of (distinct) type variables then $(\vec{\beta})|a$ denotes $(\vec{\beta}')a$, where $\vec{\beta}'$ is the sequence that arises from $\vec{\beta}$ by omitting the variables not occurring freely in a .

(iv) For $S \subseteq \mathbb{T}_{\mu'}$, write $(\vec{\beta})|S = \{(\vec{\beta})|a \mid a \in S\}$.

(v) If X is a finite set, then $|X|$ is the number of its elements.

(vi) If $\vec{\beta}$ is a sequence of variables (assumed distinct), then $\{\vec{\beta}\}$ is its set $\{\beta_1, \dots, \beta_n\}$.

Now we give the definition of the proof system (μ') .

8B.18. DEFINITION. The system (μ') deriving equations $a = b$ for $a, b \in \mathbb{T}_{\mu'}$.

(axiom)	$(\vec{\beta})A = (\vec{\beta})A$
(left μ -step)	$\frac{(\vec{\beta})A_1[\alpha := \mu\alpha.A_1] = (\vec{\beta})B}{(\vec{\beta})\mu\alpha.A_1 = (\vec{\beta})B}$
(right μ -step)	$\frac{(\vec{\beta})A = (\vec{\beta})B_1[\alpha := \mu\alpha.B_1]}{(\vec{\beta})A = (\vec{\beta})\mu\alpha.B_1}$
(μ -cong ₁)	$\frac{(\vec{\beta}\alpha)A_1 = (\vec{\beta}\alpha)B_1}{(\vec{\beta})\mu\alpha.A_1 = (\vec{\beta})\mu\alpha.B_1}$
(μ -cong ₂)	$\frac{(\vec{\beta})A_1 = (\vec{\beta})B_1 \quad \alpha \notin FV(A_1), \alpha \notin FV(B_1)}{(\vec{\beta})\mu\alpha.A_1 = (\vec{\beta})\mu\alpha.B_1}$
(\rightarrow -cong)	$\frac{(\vec{\beta}) A_1 = (\vec{\beta}) B_1 \quad (\vec{\beta}) A_2 = (\vec{\beta}) B_2}{(\vec{\beta})(A_1 \rightarrow A_2) = (\vec{\beta})(B_1 \rightarrow B_2)}$

8B.19. REMARK. (i) One has $FV(A_1)[\alpha := \mu\alpha.A_1] = FV(\mu\alpha.A_1)$ and from that one gets immediately $\vdash_{\mu^-} A = B \Rightarrow FV(A) = FV(B)$ and $\vdash_{\mu'} A = B \Rightarrow FV(A) = FV(B)$. So if $FV(A) \neq FV(B)$ then one cannot have $A =_\mu B$. Therefore from now on we restrict ourselves to equations $A = B$ such that $FV(A) = FV(B)$ and similarly for equations $(\vec{\beta})A = (\vec{\beta})B$.

(ii) Note that in rule $(\mu\text{-cong}_1)$ implicitly $\alpha \in \text{FV}(A_1) = \text{FV}(B_1)$ and that in rule $(\rightarrow\text{-cong})$ one has $\{\vec{\beta}\} \subset \text{FV}(A_1 \rightarrow A_2) = \text{FV}(B_1 \rightarrow B_2)$. Also note that rule $(\mu\text{-cong}_2)$ is superfluous, because it follows by a (left μ -step) and a (right μ -step).

(iii) In rule $(\rightarrow\text{-cong})$ the conclusion is not always uniquely determined by the assumptions: from $(\beta_1\beta_2) \upharpoonright \beta_1 = (\beta_1\beta_2) \upharpoonright \mu\alpha.\beta_1$ and $(\beta_1\beta_2) \upharpoonright \beta_2 = (\beta_1\beta_2) \upharpoonright \beta_2$, which is in fact $(\beta)\beta = (\beta)\mu\alpha.\beta$ and $(\beta)\beta = (\beta)\beta$ one can conclude not only the equality $(\beta_1\beta_2)\beta_1 \rightarrow \beta_2 = (\beta_1\beta_2)(\mu\alpha.\beta_1) \rightarrow \beta_2$, but also $(\beta)\beta \rightarrow \beta = (\beta)(\mu\alpha.\beta) \rightarrow \beta$.

The following is an example of a successful proof attempt of an equality between annotated types. In other cases such an attempt may fail.

8B.20. EXAMPLE.

$$\begin{array}{c}
 \frac{(\beta)\beta = (\beta)\beta}{(\beta)\mu\delta.\beta = (\beta)\beta} \text{ (left } \mu\text{-step)} \quad \frac{(\gamma)\gamma = (\gamma)\gamma}{(\beta\gamma)((\mu\delta.\beta) \rightarrow \gamma) = (\beta\gamma)(\beta \rightarrow \gamma)} \text{ } (\rightarrow\text{-cong}) \\
 \frac{(\beta)\beta = (\beta)\beta}{(\beta)(\mu\delta.\beta) = (\beta)\beta} \text{ (left } \mu\text{-step)} \quad \frac{(\beta\mu\gamma)((\mu\delta.\beta) \rightarrow \gamma) = (\beta\mu\gamma)(\beta \rightarrow \gamma)}{(\beta\mu\gamma)((\mu\delta.\beta) \rightarrow \gamma) = (\beta)(\mu\gamma.\beta \rightarrow \gamma)} \text{ } (\mu\text{-cong}_1) \\
 \hline
 \frac{(\beta)((\mu\delta.\beta) \rightarrow \mu\gamma.((\mu\delta.\beta) \rightarrow \gamma)) = (\beta)(\beta \rightarrow (\mu\gamma.\beta \rightarrow \gamma))}{(\beta\mu\gamma.((\mu\delta.\beta) \rightarrow \gamma) = (\beta)(\beta \rightarrow (\mu\gamma.\beta \rightarrow \gamma)))} \text{ (left } \mu\text{-step)} \\
 \frac{(\beta\mu\gamma.((\mu\delta.\beta) \rightarrow \gamma) = (\beta)(\beta \rightarrow (\mu\gamma.\beta \rightarrow \gamma)))}{(\beta\mu\alpha\gamma.((\mu\delta.\beta) \rightarrow \gamma) = (\beta)\mu\alpha.(\beta \rightarrow (\mu\gamma.\beta \rightarrow \gamma)))} \text{ } (\mu\text{-cong}_2) \\
 \frac{(\beta\mu\alpha\gamma.((\mu\delta.\beta) \rightarrow \gamma) = (\beta)\mu\alpha.(\beta \rightarrow (\mu\gamma.\beta \rightarrow \gamma)))}{\mu\beta\alpha\gamma.((\mu\delta.\beta) \rightarrow \gamma) = \mu\beta\alpha.(\beta \rightarrow (\mu\gamma.\beta \rightarrow \gamma))} \text{ } (\mu\text{-cong}_1)
 \end{array}$$

8B.21. REMARK. (i) Let $A = \mu\beta\alpha\gamma.((\mu\delta\beta) \rightarrow \gamma)$ and $B = \mu\beta\alpha.(\beta \rightarrow (\mu\gamma.\beta \rightarrow \gamma))$. In the example above we gave a proof of $\vdash_{\mu'} A = B$. In fact one could try other proofs. For example starting (from below) with a left μ -step and a right μ -step. Then one encounters at some stage again a node $A = B$. So one could go on for ever and the result would be an infinite tree. But of course one should stop developing the sub-branch at that node.

In general if in a proof attempt for $\vdash_{\mu'} a = b$ we arrive at a node $c = d$ that we encountered already below on the branch to the root, then we stop: we consider only proof attempts without repetitions.

(ii) We go a step further in our restriction on the proof attempts. An (annotated) equation $a = b$ is called a variant of $a' = b'$ if the second results from the first by renaming the free variables on both sides in the same way. For example the following three equations are variants

$$\begin{array}{rcl}
 (\beta)\mu\alpha.\alpha \rightarrow \beta \rightarrow \gamma \rightarrow \delta & = & (\beta)\mu\alpha.\alpha \rightarrow \gamma \\
 (\beta)\mu\alpha.\alpha \rightarrow \beta \rightarrow \gamma' \rightarrow \delta' & = & (\beta)\mu\alpha.\alpha \rightarrow \gamma' \\
 (\beta')\mu\alpha'.\alpha' \rightarrow \beta' \rightarrow \gamma' \rightarrow \delta' & = & (\beta')\mu\alpha'.\alpha' \rightarrow \gamma'
 \end{array}$$

of each other, but not of

$$(\beta)\mu\alpha.\alpha \rightarrow \beta \rightarrow \gamma_1 \rightarrow \delta = (\beta)\mu\alpha.\alpha \rightarrow \gamma_2.$$

Note that variants $a = b$ and $a' = b'$ are equivalent in the sense that from a derivation (attempt) for $a = b$ one obtains one for $a' = b'$ of the same structure by a change of free variables. We only consider proof attempts in (μ') and (μ^-) without repetitions up to variance.

8B.22. LEMMA. *A derivation (attempt) in (μ') corresponds to one in (μ^-) , having the same structure. The converse is also valid.*

PROOF. In a derivation attempt of $\vdash_{\mu'} A = B$ skip all $(\vec{\beta})$, i.e. replace all $(\vec{\beta})C = (\vec{\beta})D$ by $C = D$. By Remark 8B.21(ii) this does not depend on the choice of representatives of the α -equivalence class of annotated types.

Conversely, given a derivation of $\vdash_{\mu^-} A = B$, build a derivation of $\vdash_{\mu'} A = B$ working upwards using the corresponding rules in (μ') . ■

8B.23. REMARK. The map from the derivations in (μ^-) to those in (μ') is a bijection up to variance. For example derivations like

$$\frac{\alpha = \alpha}{\mu\alpha.\alpha = \mu\alpha.\alpha} \text{ and } \frac{\beta = \beta}{\mu\beta.\beta = \mu\beta.\beta}$$

are identified.

Now we will finish the proof of decidability of $=_{\mu}$. We show that the number of different nodes $c = d$ occurring in a proof search tree for $\vdash_{\mu'} A = B$ is bounded by a function in A, B .

8B.24. DEFINITION. (i) The binary relation $\Rightarrow_{\mu'}: \mathbb{T}_{\mu'} \rightarrow \mathbb{T}_{\mu'}$ is defined as follows.

$$\begin{aligned} (\vec{\beta})\mu\alpha.A_1 &\Rightarrow_{\mu'} (\vec{\beta}\alpha)A_1 \text{ if } \alpha \in FV(A_1); \\ (\vec{\beta})\mu\alpha.A_1 &\Rightarrow_{\mu'} (\vec{\beta})A_1[\alpha := \mu\alpha.A_1]; \\ (\vec{\beta})(A_1 \rightarrow A_2) &\Rightarrow_{\mu'} (\vec{\beta})\upharpoonright A_i \quad i = 1, 2; \end{aligned}$$

(ii) For $a \in \mathbb{T}_{\mu'}$ define the set $SC_r^w(a) = \{b \in \mathbb{T}_{\mu'} \mid a \Rightarrow_{\mu'}^* b\}$. The name SC comes from *subterm closure* and we will encounter it in various variants.

8B.25. REMARK. Note that $\Rightarrow_{\mu'}$ is not meant to be a reduction relation, compatible with μ or \rightarrow . For example we do not require $\mu\beta\mu\alpha.A_1 \Rightarrow_{\mu'} \mu\beta.A_1[\alpha := \mu\alpha.A_1]$.

8B.26. LEMMA. *If $c = d$ occurs in a proof attempt in (μ') for $a = b$. Then $c \in SC_r^w(a)$ and $d \in SC_r^w(b)$.*

PROOF. By induction on the length of the proof attempt for $\vdash_{\mu'} a = b$. ■

The essential step towards decidability is showing that $|SC_r^w(a)|$ is bounded by a computable function in a , Lemma 8B.31.

8B.27. LEMMA. (i) *Let $\{\vec{\beta}\} \cap \{\vec{\alpha}\} = \emptyset$. Then $(\vec{\beta}\vec{\alpha})\upharpoonright A = (\vec{\beta})\upharpoonright((\vec{\alpha})\upharpoonright A)$.*

$$(ii) \quad SC_r^w((\vec{\beta})\upharpoonright A) = (\vec{\beta})\upharpoonright SC_r^w(A).$$

PROOF. Immediate. ■

8B.28. LEMMA. $SC_r^w(\alpha) = \{\alpha\};$

$$SC_r^w(\mu\alpha.A_1) = \{\mu\alpha.A_1\} \cup (\alpha)\upharpoonright SC_r^w(A_1) \cup SC_r^w(A_1[\alpha := \mu\alpha.A_1]);$$

$$SC_r^w(A_1 \rightarrow A_2) = \{A_1 \rightarrow A_2\} \cup SC_r^w(A_1) \cup SC_r^w(A_2).$$

PROOF. By induction on the structure of $a \in \mathbb{T}_{\mu'}$, using (ii) of the preceding lemma. Note that the second clause also holds for $\alpha \notin FV(A_1)$, because then

$$(\alpha)\upharpoonright SC_r^w(A_1) = SC_r^w(A_1) = SC_r^w(A_1[\alpha := \mu\alpha.A_1]). \blacksquare$$

The set $SC_r^w(A_1[\alpha := \mu\alpha.A_1])$ is intricate, but by Lemma 8B.30 it will be contained in

$$\{\mu\alpha.A_1\} \cup (\alpha)\upharpoonright SC_r^w(A_1) \cup SC_r^w(A_1)[\alpha := \mu\alpha.A_1]. \quad (1)$$

Note the difference in the brackets in (1) and the second clause in Lemma 8B.28.

The following lemma is an adaptation of Lemma 25 in Endrullis, Grabmayer, Klop, and van Oostrom [2011].

8B.29. LEMMA. *If $A[\alpha := \mu\alpha.A] \Rightarrow_{\mu'}^* b$ then we have either*

- (i) *this reduction is $A[\alpha := \mu\alpha.A] \Rightarrow_{\mu'}^* \mu\alpha.A \Rightarrow_{\mu'}^* b$;*
- (ii) *this reduction is an $[\alpha := \mu\alpha.A]$ instance of $A \Rightarrow_{\mu'}^* a$. So $b = a[\alpha := \mu\alpha.A]$.*

PROOF. By induction on the length of the reduction $A[\alpha := \mu\alpha.A] \Rightarrow_{\mu'}^* b$.

If the length is zero then we have (ii). Otherwise $A[\alpha := \mu\alpha.A] \Rightarrow_{\mu'}^* b' \Rightarrow_{\mu'} b$. The induction hypothesis for $A[\alpha := \mu\alpha.A] \Rightarrow_{\mu'}^* b'$ gives either (i) or (ii) as follows.

- (i) $A[\alpha := \mu\alpha.A] \Rightarrow_{\mu'}^* \mu\alpha.A \Rightarrow_{\mu'}^* b'$. Then we have also case (i) for $A[\alpha := \mu\alpha.A] \Rightarrow_{\mu'}^* b$.
- (ii) The reduction $A[\alpha := \mu\alpha.A] \Rightarrow_{\mu'}^* b'$ is a $[\alpha := \mu\alpha.A]$ instance of $A \Rightarrow_{\mu'}^* a'$. So $b' = a'[\alpha := \mu\alpha.A]$. We distinguish subcases for a' as follows.
 - (iia) $a' = \alpha$. Then we have Case (i) (for $A[\alpha := \mu\alpha.A] \Rightarrow_{\mu'}^* b$).
 - (iib) $a' = \beta \neq \alpha$ or $a' = (\beta)\beta$. These cases cannot occur because then $b' = a'$ is in nf.
 - (iic) $a' = (\vec{\beta})\mu\gamma.A'$ with $\alpha \notin \{\vec{\beta}, \gamma\}$. Then $(\vec{\beta})$ at the root is very innocent. It is frozen and occurs in the same way at the root in all types occurring in the proof. Therefore we may assume $\{\vec{\beta}\} = \emptyset$. Now $b' = \mu\gamma.A'[\alpha := \mu\alpha.A]$. We distinguish two subsubcases.
 - (iic1) $\mu\gamma.A'[\alpha := \mu\alpha.A] \Rightarrow_{\mu'}^* (\gamma)(A'[\alpha := \mu\alpha.A])$ (*). Then $\gamma \in \text{FV}(A'[\alpha := \mu\alpha.A])$, hence also $\gamma \in \text{FV}(A')$, as $\gamma \notin \text{FV}(\mu\alpha.A)$ by the variable convention. So the reduction (*) is an $[\alpha := \mu\alpha.A]$ instance of $\mu\gamma.A' \Rightarrow_{\mu'}^* (\gamma).A'$. Therefore $A[\alpha := \mu\alpha.A] \Rightarrow_{\mu'}^* b' \Rightarrow_{\mu'} b$ is an instance of $A \Rightarrow_{\mu'}^* \mu\gamma.A' \Rightarrow_{\mu'}^* (\gamma).A'$ and we are in Case (ii) (for $A[\alpha := \mu\alpha.A] \Rightarrow_{\mu'}^* b$).
 - (iic2) $b' = \mu\gamma.A'[\alpha := \mu\alpha.A] \Rightarrow_{\mu'} A'[\alpha := \mu\alpha.A][\gamma := \mu\gamma.A'[\alpha := \mu\alpha.A]] = b$. By the substitution lemma $b' = \mu\gamma.A'[\alpha := \mu\alpha.A] \Rightarrow_{\mu'} A'[\gamma := \mu\gamma.A'][\alpha := \mu\alpha.A] = b$. So $A[\alpha := \mu\alpha.A] \Rightarrow_{\mu'}^* b' \Rightarrow_{\mu'} b$ is an instance of $A \Rightarrow_{\mu'}^* a' = \mu\gamma.A' \Rightarrow_{\mu'} a = A'[\gamma := \mu\gamma.A']$ and we are in Case (ii) (for $A[\alpha := \mu\alpha.A] \Rightarrow_{\mu'}^* b$).
 - (iid) $a' = (\vec{\beta})(A_1 \rightarrow A_2)$. As $\alpha \notin \{\vec{\beta}\}$ and $\{\vec{\beta}\} \cap \text{FV}(\mu\alpha.A) = \emptyset$. We have

$$\begin{aligned} b' &= (\vec{\beta})(A_1[\alpha := \mu\alpha.A] \rightarrow A_2[\alpha := \mu\alpha.A]) \\ &\Rightarrow_{\mu'} b = (\vec{\beta})\upharpoonright(A_i[\alpha := \mu\alpha.A]) = ((\vec{\beta})\upharpoonright A_i)[\alpha := \mu\alpha.A]. \end{aligned}$$
 Also $a' \Rightarrow_{\mu'} (\vec{\beta})\upharpoonright A_i$. Again we are in case (ii) (for $A[\alpha := \mu\alpha.A] \Rightarrow_{\mu'}^* b$). ■

For an alternative proof of the following Lemma, see Exercise 8D.7.

8B.30. LEMMA. $\mathcal{SC}_r^w(\mu\alpha.A_1) \subseteq \{\mu\alpha.A_1\} \cup (\alpha)\upharpoonright \mathcal{SC}_r^w(A_1) \cup \mathcal{SC}_r^w(A_1)[\alpha := \mu\alpha.A_1]$.

PROOF. Let $\mu\alpha.A_1 \Rightarrow_{\mu'}^* b$ be a reduction path of minimal length. If the length is zero then $b = \mu\alpha.A_1$ and we are done. If the length is greater than zero, we distinguish cases.

Case (i) $\mu\alpha.A_1 \Rightarrow_{\mu'}^* (\alpha).A_1 \Rightarrow_{\mu'}^* b$. Then we have $\alpha \in \text{FV}(A_1)$ and $b \in \mathcal{SC}_r^w((\alpha).A_1) = (\alpha)\upharpoonright \mathcal{SC}_r^w(A_1)$, by Lemma 8B.27.

Case (ii) $\mu\alpha.A_1 \Rightarrow_{\mu'}^* A_1[\alpha := \mu\alpha.A_1] \Rightarrow_{\mu'}^* b$. By Lemma 8B.29 there are two subcases.

(iia) $\mu\alpha.A_1 \Rightarrow_{\mu'}^* A_1[\alpha := \mu\alpha.A_1] \Rightarrow_{\mu'}^* \mu\alpha.A_1 \Rightarrow_{\mu'}^* b$. By minimality this is impossible.

(iib) $A_1[\alpha := \mu\alpha.A_1] \Rightarrow_{\mu'}^* b$ is an $[\alpha := \mu\alpha.A_1]$ instance of $A_1 \Rightarrow_{\mu'}^* a$. Now $a \in \mathcal{SC}_r^w(A_1)$, hence $b = a[\alpha := \mu\alpha.A_1] \in \mathcal{SC}_r^w(A_1)[\alpha := \mu\alpha.A_1]$. ■

8B.31. LEMMA. $|\mathcal{SC}_r^w(A)| \leq 3^{l(A)}$.

PROOF. By induction on A , using Lemmas 8B.28 and 8B.30. ■

8B.32. COROLLARY. *The number of different nodes in a branch of a proof search tree for $\vdash_{\mu'} A = B$ is at most*

$$3^{l(A)+l(B)}.$$

PROOF. Immediate by Lemma 8B.26. ■

THEOREM 8B.15 $A =_{\mu} B$ is decidable.

PROOF. By Lemma 8B.22(ii) and Corollary 8B.32 proof search is bounded. This proves Lemma 8B.12, hence also the result. ■

Invertibility of strong equivalence

Also strong μ -equivalence is invertible.

8B.33. PROPOSITION. *The type algebra \mathbb{T}_{μ}^* is invertible.*

PROOF. By the fact that $(A \rightarrow B)_{\mu}^* =$

$$\begin{array}{ccc} & \rightarrow & \\ (A)_{\mu}^* & \nearrow & \searrow \\ & (B)_{\mu}^* & \end{array}.$$

Hence $(A \rightarrow B)_{\mu}^* = (A' \rightarrow B')_{\mu}^*$ implies $(A)_{\mu}^* = (A')_{\mu}^*$ and $(B)_{\mu}^* = (B')_{\mu}^*$. ■

Clearly $A =_{\mu}^* B$ if at all nodes u the trees $(A)_{\mu}^*$ and $(B)_{\mu}^*$ have the same label.

Decidability and axiomatization of strong equivalence

We will show that it is decidable whether given two types $A, B \in \mathbb{T}_{\mu}$ their unfolding as infinite trees are equal, i.e. $(A)_{\mu}^* = (B)_{\mu}^*$, also written as $A =_{\mu}^* B$. The decidability is due to Koster [1969], with an exponential algorithm. In unpublished lecture notes by Koster this was improved to an algorithm of complexity $O(n^2)$. In Kozen, Palsberg, and Schwartzbach [1995] another quadratic algorithm is given. In Moller and Smolka [1995] an $O(n \log n)$ algorithm is presented. Automata aspects of both algorithms can be found in ten Eikelder [1991].

While the notion of weak equivalence was introduced by means of a formal inference system, that of strong equivalence was introduced in Chapter 7E 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

$$\vdash_{\text{BH}} A = B \Leftrightarrow A =_{\mu}^* B$$

are taken from Brandt and Henglein [1998]. Other complete formalizations of strong equivalence have been given by Ariola and Klop [1996] and Amadio and Cardelli [1993], and will be discussed below in Definition 8B.60. See Grabmayer [2005] for a proof-theoretical analysis of these formalizations.

Here we follow Brandt and Henglein [1998], who give at the same time an axiomatization and the decidability of strong equality.

8B.34. DEFINITION. For $A \in \mathbb{T}_\mu$ define the set $\mathcal{SC}^s(A) \subseteq \mathbb{T}_\mu$.

$$\begin{aligned}\mathcal{SC}^s(c) &\triangleq \{c\}, \text{ if } c \text{ is a variable or a constant;} \\ \mathcal{SC}^s(A_1 \rightarrow A_2) &\triangleq \{A_1 \rightarrow A_2\} \cup \mathcal{SC}^s(A_1) \cup \mathcal{SC}^s(A_2); \\ \mathcal{SC}^s(\mu\beta A) &\triangleq \{\mu\beta A\} \cup \mathcal{SC}^s(A)[\beta := \mu\beta A].\end{aligned}$$

8B.35. LEMMA. Let $A \in \mathbb{T}_\mu$. Then the cardinality of the set $\mathcal{SC}^s(A)$ is at most the number of symbols in A and hence finite.

PROOF. By induction on the generation of A . ■

8B.36. LEMMA. For all $A \in \mathbb{T}_\mu$ one has

$$\mathcal{SC}^s(\mathcal{SC}^s(A)) = \mathcal{SC}^s(A).$$

PROOF. As to (\subseteq) . First show by induction on A that²

$$\mathcal{SC}^s(A[\beta := B]) \subseteq \mathcal{SC}^s(A)[\beta := B] \cup \mathcal{SC}^s(B). \quad (1)$$

Then show $\mathcal{SC}^s(\mathcal{SC}^s(A)) \subseteq \mathcal{SC}^s(A)$ by induction on A . For $A = \mu\alpha.A_1$ use (1) and the induction hypothesis for A_1 .

As to (\supseteq) . This immediately follows from the fact that $B \in \mathcal{SC}(B)$ for all B . ■

8B.37. DEFINITION. Following Endrullis, Grabmayer, Klop, and van Oostrom [2011] we define for $A \in \mathbb{T}_\mu$ the set $\mathcal{SC}_r^s(A) \subseteq \mathbb{T}_\mu$. Define the binary relation $\rightsquigarrow \subseteq \mathbb{T}_\mu^2$ by

$$\begin{aligned}(A_1 \rightarrow A_2) &\rightsquigarrow A_i; \\ \mu\alpha.A &\rightsquigarrow A[\alpha := \mu\alpha.A].\end{aligned}$$

Now write

$$\mathcal{SC}_r^s(A) \triangleq \{C \in \mathbb{T}_\mu \mid A \rightsquigarrow^* C\},$$

where \rightsquigarrow^* is the transitive reflexive closure of \rightsquigarrow .

8B.38. LEMMA. For all $A \in \mathbb{T}_\mu$ one has $\mathcal{SC}_r^s(A) = \mathcal{SC}^s(A)$.

PROOF. (\subseteq) By Lemma 8B.36.

(\supseteq) By induction on the structure of A , using the Substitution Lemma 7D.25. ■

8B.39. COROLLARY. For every $A \in \mathbb{T}_\mu$ the set $\mathcal{SC}_r^s(A)$ is finite.

PROOF. By Lemma 8B.35. ■

For an alternative proof of this Corollary, see Exercise 8D.9.

²In Brandt and Henglein [1998] Lemma 15, under the condition $\beta \in \text{FV}(A)$, even equality is proved.

8B.40. DEFINITION (Ariola and Klop [1996]). (i) On \mathbb{T}_μ consider the formal system defined by the following axioms and rules.

(ident)	$\mathcal{H} \vdash A = A$
(symm)	$\frac{\mathcal{H} \vdash A = B}{\mathcal{H} \vdash B = A}$
(trans)	$\frac{\mathcal{H} \vdash A = B \quad \mathcal{H} \vdash B = C}{\mathcal{H} \vdash A = C}$
(axiom)	$\mathcal{H} \vdash A = B,$ if $(A = B) \in \mathcal{H},$
(μ -eq)	$\mathcal{H} \vdash \mu\alpha.A = A[\alpha := \mu\alpha.A],$
(deconstr)	$\frac{\mathcal{H} \vdash (A \rightarrow B) = (A' \rightarrow B')}{\mathcal{H} \vdash A = A' \quad \mathcal{H} \vdash (B' = B')}$

Write $\mathcal{H} \vdash_{\mu\text{-dec}} A = B$ if $\mathcal{H} \vdash A = B$ is derivable in this system.

(ii) The *deductive closure* of \mathcal{H} is defined as

$$\{C = D \mid \mathcal{H} \vdash_{\mu\text{-dec}} C = D\}.$$

(iii) Define an equation $A = B$ to be *consistent* if for no $C = D$ in the deductive closure of $\{A = B\}$ one has $\text{ls}(C) \not\equiv \text{ls}(D).$

8B.41. LEMMA. For $A, B \in \mathbb{T}^A$ one has

$$A =_\mu^* B \Leftrightarrow A = B \text{ is consistent.}$$

PROOF. See Exercise 8D.15. ■

8B.42. EXAMPLE. Consider $A \triangleq \mu\alpha.\alpha \rightarrow \alpha$ and $B \triangleq \mu\alpha\beta.\alpha \rightarrow \beta.$ Write $C \triangleq \mu\gamma.B \rightarrow \gamma.$ Then the deductive closure of $\{A = B\}$ is $\{P = Q \mid P, Q \in \{A, B, C\}\}.$ For all P, Q we have $\text{ls}P = \text{ls}Q = \rightarrow.$ Hence $A = B$ is consistent: no inconsistency ‘has appeared’. Therefore $A =_\mu^* B.$

8B.43. LEMMA. Let $C = D$ be in the deductive closure of $\{A = B\}.$ Then

$$C, D \in \mathcal{SC}_r^s(A) \cup \mathcal{SC}_r^s(B).$$

Therefore the cardinality of the deductive closure is at most n^2 , where n is the maximum of the number of symbols in A and in $B.$

PROOF. By induction on the derivation in $\vdash_{\mu\text{-dec}}$ and Lemmas 8B.35 and 8B.38. ■

8B.44. THEOREM (Koster [1969]). The relation $=_\mu^* \subseteq (\mathbb{T}_\mu)^2$ is decidable.

PROOF. Using the system $\vdash_{\mu\text{-dec}}$ one produces the elements of the deductive closure of $\{A = B\}$ as an increasing sequence of sets

$$\mathcal{C}_0 \triangleq \{A = B\} \subseteq \mathcal{C}_1 \subseteq \mathcal{C}_2 \subseteq \cdots \subseteq \mathcal{C}_n$$

such that if $\mathcal{C}_n = \mathcal{C}_{n+1}$, then $\mathcal{C}_n = \mathcal{C}_m$ for all $m > n.$ This is done by adding to already produced equations only new equations produced by the formal system. At a certain moment $\mathcal{C}_n = \mathcal{C}_{n+1}$, by Lemma 8B.43; then the deductive closure is completed and we can decide whether the result is consistent or not. This is sufficient by Lemma 8B.41. ■

(ident)	$\mathcal{H} \vdash A = A$
(symm)	$\frac{\mathcal{H} \vdash A = B}{\mathcal{H} \vdash B = A}$
(trans)	$\frac{\mathcal{H} \vdash A = B \quad \mathcal{H} \vdash B = C}{\mathcal{H} \vdash A = C}$
(axiom)	$\mathcal{H} \vdash A = B,$ if $(A = B) \in \mathcal{H},$
(μ -eq)	$\mathcal{H} \vdash \mu\alpha.A = A[\alpha := \mu\alpha.A]$
(μ -cong)	$\frac{\mathcal{H} \vdash A = A'}{\mathcal{H} \vdash \mu\alpha.A = \mu\alpha.A'},$ if α not free in $\mathcal{H},$
(\rightarrow -cong)	$\frac{\mathcal{H} \vdash A = A' \& \mathcal{H} \vdash B = B'}{\mathcal{H} \vdash (A \rightarrow B) = (A' \rightarrow B')}$
(coind)	$\frac{\mathcal{H}, (A \rightarrow B) = (A' \rightarrow B') \vdash A = A' \quad \mathcal{H}, (A \rightarrow B) = (A' \rightarrow B') \vdash B = B'}{\mathcal{H} \vdash (A \rightarrow B) = (A' \rightarrow B')}$

FIGURE 25. The system (BH).

Axiomatization of strong equality

We will introduce a deductive system (BH) for deriving strong equality. The decidability of $=_\mu^*$ will follow a second time as a Corollary (8B.57).

The proof system for strong equality 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. From the completeness of this formalization we obtain a proof of the decidability of strong equivalence for μ -types. In the next section this proof strategy will be used to show the decidability of strong equivalence $=_{\mathcal{R}}^*$ for a simultaneous recursion \mathcal{R} .

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 $A_1 \rightarrow A_2 = B_1 \rightarrow B_2$, 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 in Theorem 8B.56 that provability in (BH) from $\mathcal{H} = \emptyset$ corresponds exactly to strong equivalence.

8B.45. DEFINITION. Let \mathcal{H} denote a set of statements of the form $A_1 \rightarrow A_2 = B_1 \rightarrow B_2$, where $A_1, B_1, A_2, B_2 \in \mathbb{T}_\mu$. The system (BH) is defined by the rules in Fig. 25.

We write $\mathcal{H} \vdash_{\text{BH}} A = B$ if $\mathcal{H} \vdash A = B$ can be derived by the rules of (BH).

Rule (coind) is obviously the crucial one. It says that if we are able to prove $A = A'$ and $B = B'$ assuming $A \rightarrow B = A' \rightarrow B'$ then we can conclude that $A \rightarrow B = A' \rightarrow B'$.

By rule (coind) the system (BH) exploits its 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, see Definition 7E.9 relating the infinite trees $(A)_\mu^*$ and $(B)_\mu^*$.

8B.46. REMARK. (i) By induction on derivations one can show that weakening is an admissible rule:

$$\mathcal{H} \vdash A = B \Rightarrow \mathcal{H}, \mathcal{H}' \vdash A = B.$$

(ii) Note that the rule (\rightarrow -cong) can be omitted while maintaining the same provability strength. This follows from the rule (coind) and weakening. Indeed, assuming $\mathcal{H} \vdash A = A'$, $\mathcal{H} \vdash B = B'$, one has by weakening $\mathcal{H}, (A \rightarrow B) = (A' \rightarrow B') \vdash A = A'$, $\mathcal{H}, (A \rightarrow B) = (A' \rightarrow B') \vdash B = B'$ and therefore by rule (coind) $\mathcal{H} \vdash (A \rightarrow B) = (A' \rightarrow B')$.

8B.47. EXAMPLE. As an example of a derivation in (BH), consider $S \triangleq \mu\beta.\mu\alpha.\alpha \rightarrow \beta$, $T \triangleq \mu\alpha.\alpha \rightarrow S$, and $B \triangleq \mu\alpha.\alpha \rightarrow \alpha$. Observe that $S =_\mu T =_\mu T \rightarrow S$, and $B =_\mu B \rightarrow B$ but $S \neq_\mu B$. We will show $\vdash_{BH} S = B$. Let \mathcal{D} stand for the following derivation of $\mathcal{H} \vdash T = B$ with $\mathcal{H} = \{T \rightarrow S = B \rightarrow B\}$.

$$\frac{\begin{array}{c} (\mu\text{-eq}) \\ \hline \mathcal{H} \vdash T = T \rightarrow S \end{array} \quad \begin{array}{c} (\text{hyp}) \\ \hline \mathcal{H} \vdash T \rightarrow S = B \rightarrow B \end{array}}{\mathcal{H} \vdash T = B \rightarrow B} \text{(trans)} \quad \frac{\begin{array}{c} (\mu\text{-eq}) \\ \hline \mathcal{H} \vdash B = B \rightarrow B \end{array} \quad \begin{array}{c} (\text{symm}) \\ \hline \mathcal{H} \vdash B \rightarrow B = B \end{array}}{\mathcal{H} \vdash B \rightarrow B = B} \text{(trans)}$$

$$\frac{}{\mathcal{H} \vdash T = B}$$

Consider now the following derivation in (BH).

$$\frac{\begin{array}{c} \mathcal{D} \quad \begin{array}{c} (\mu\text{-eq}) \\ \vdots \end{array} \quad \mathcal{D} \\ \vdots \quad \frac{\begin{array}{c} \mathcal{H} \vdash S = T \quad \mathcal{H} \vdash T = B \\ \hline \mathcal{H} \vdash T = B \end{array} \quad \begin{array}{c} (\text{trans}) \\ \begin{array}{c} (\mu\text{-eq}) \\ \hline \vdash T \rightarrow S = B \rightarrow B \end{array} \quad \begin{array}{c} (\text{coind}) \\ \begin{array}{c} (\mu\text{-eq}) \\ \vdash B = B \rightarrow B \end{array} \quad \begin{array}{c} (\text{symm}) \\ \vdash B \rightarrow B = B \end{array} \end{array} \end{array} \quad \begin{array}{c} (\text{trans}) \\ \begin{array}{c} (\mu\text{-eq}) \\ \vdash T \rightarrow S = B \end{array} \end{array} \end{array} \quad \begin{array}{c} (\text{trans}) \\ \begin{array}{c} (\mu\text{-eq}) \\ \vdash S = T \end{array} \end{array} \end{array} \quad \begin{array}{c} (\text{trans}) \\ \vdash S = B \end{array}$$

Now we will prove the following soundness and completeness result.

$$A =_\mu^* B \Leftrightarrow \vdash_{BH} A = B.$$

The proof occupies 8B.48-8B.56.

8B.48. DEFINITION. (i) Let $A, B \in \mathbb{T}_\mu$ and $k \geq 0$. Define

$$A =_k^* B \triangleq ((A)_\mu^*)_k = ((B)_\mu^*)_k.$$

(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 $A =_\mu^* B$ for all $A = B \in \mathcal{H}$.

Note that $A =_\mu^* B \Leftrightarrow \forall k \geq 0. A =_k^* B$, by Lemma 7E.7(i).

8B.49. LEMMA. *Let \mathcal{H} be valid. Then*

$$\mathcal{H} \vdash_{\text{BH}} A = B \Rightarrow A =_{\mu}^{*} B.$$

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 7E.7(ii). If $k > 0$ the proof is by induction on derivations. Most rules are easy to treat. Rule (μ -cong) follows using Remark 7E.29, showing by induction on p

$$A =_k^{*} B \Rightarrow \forall p \leq k. [\mu\alpha. A =_p^{*} \mu\alpha. B].$$

The most interesting case is when the last applied rule is (coind), i.e.

$$\text{(coind)} \quad \frac{\mathcal{H} \cup \{A \rightarrow B = A' \rightarrow B'\} \vdash A = A' \quad \mathcal{H} \cup \{A \rightarrow B = A' \rightarrow B'\} \vdash B = B'}{\mathcal{H} \vdash A \rightarrow B = A' \rightarrow B'}$$

Since \mathcal{H} is k -valid it is also $(k-1)$ -valid. By the induction hypothesis on k we have $A \rightarrow B =_{k-1}^{*} A' \rightarrow B'$. But then $\mathcal{H} \cup \{A \rightarrow B = A' \rightarrow B'\}$ 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 7E.7(iii) we conclude $A \rightarrow B =_k^{*} A' \rightarrow B'$. ■

8B.50. COROLLARY (Soundness). $\vdash_{\text{BH}} A = B \Rightarrow A =_{\mu}^{*} B$.

PROOF. Take \mathcal{H} empty. ■

The opposite implication is the completeness of (BH). The proof of this fact is given in a constructive way. Below in Definition 8B.51 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

$$\mathcal{S}(\mathcal{H}, A, B) \Rightarrow \mathcal{H} \vdash_{\text{BH}} A = B.$$

Note that it is trivially decidable whether a type A is non-circular or not. If A and B are circular we can easily prove $\vdash_{\text{BH}} A = B$, see Lemma 7D.32. Otherwise A has a reduced form A' as defined in 7D.34 such that, for all \mathcal{H} , $\mathcal{H} \vdash_{\text{BH}} A = A'$ by (μ -eq) and (trans).

8B.51. DEFINITION. Let $A, B \in \mathbb{T}_{\mu}$ be two μ -types and let \mathcal{H} be a set of equations of the form $A_1 \rightarrow A_2 = B_1 \rightarrow B_2$. The predicate $\mathcal{S}(\mathcal{H}, A, B)$ is defined as follows. Let A', B' be the reduced forms of A, B , respectively. Remember Definition 7D.34(i), explaining that these are only of one of the possible forms (a) α , a type atom, (b) $A \rightarrow B$, a function type, or (c) \bullet standing for a circular type. In these cases the lead symbol of the type involved is α, \rightarrow , or \bullet , respectively. Then define

$$\begin{aligned} \mathcal{S}(\mathcal{H}, A, B) &\triangleq \text{true, } \text{if } A' \equiv B' \text{ or if } (A' = B') \in \mathcal{H}; \text{ else} \\ &\triangleq \text{false, } \text{if } A' \text{ and } B' \text{ have different lead symbols; else} \\ &\triangleq \mathcal{S}(\mathcal{H} \cup \{A' = B'\}, A_1, B_1) \& \mathcal{S}(\mathcal{H} \cup \{A' = B'\}, A_2, B_2), \\ &\quad \text{if } A' \equiv A_1 \rightarrow A_2, B' \equiv B_1 \rightarrow B_2. \end{aligned}$$

This $\mathcal{S}(\mathcal{H}, A, B)$ will be seen to be always defined and is a truth value in $\{\text{true}, \text{false}\}$. At the same time \mathcal{S} is considered as a partial computable function denoting this truth

value, that happens to be total. Therefore it is intuitively clear to see $\mathcal{S}(\mathcal{H}, A, B)$ also as an expression reducing to the truth value.

8B.52. 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 $\mathcal{H}' = \mathcal{H} \cup \{A_1 \rightarrow A_2 = B_1 \rightarrow B_2\}$, 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 the last case of the previous definition.

Note that \succ is a “non-deterministic relation”.

(ii) The notations \succ^n and \succ^* have the usual meaning: n -step rewriting and the reflexive transitive closure of \succ .

8B.53. LEMMA. (i) If $\mathcal{S}(\mathcal{H}, A, B) \succ^* \mathcal{S}(\mathcal{H}', A', B')$, then $A' \in \mathbf{SC}_r^s(A)$, $B' \in \mathbf{SC}_r^s(B)$ and all equations in $\mathcal{H}' \setminus \mathcal{H}$ are of the form $A'' = B''$, where $A'' \in \mathbf{SC}_r^s(A)$, $B'' \in \mathbf{SC}_r^s(B)$.

(ii) The relation \succ is well-founded.

(iii) The predicate \mathcal{S} is decidable, i.e., \mathcal{S} seen as map is total and computable.

PROOF. (i) Directly by Definition 8B.37.

(ii) Suppose there exists an infinite sequence

$$\mathcal{S}(\mathcal{H}, A, B) \succ \mathcal{S}(\mathcal{H}_1, A_1, B_1) \succ \mathcal{S}(\mathcal{H}_2, A_2, B_2) \succ \dots .$$

Then $\mathcal{S}(\mathcal{H}_k, A_k, B_k) \succ \mathcal{S}(\mathcal{H}_{k+1}, A_{k+1}, B_{k+1})$, and, for otherwise by the first clause of the definition of $\mathcal{S}(\mathcal{H}, A, B)$ the sequence stops, with $\mathcal{S}(\mathcal{H}_k, A_k, B_k)$ being true. From (i) and Lemma 8B.35 it follows that the sequence terminates.

(iii) By (ii) and König’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. ■

8B.54. LEMMA. If $\mathcal{S}(\mathcal{H}, A, B) = \text{true}$, then $\mathcal{H} \vdash_{\text{BH}} A = B$. If moreover A, B have no circular subterms, then rule (μ -cong) is not needed.

PROOF. 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 the first clause corresponds to a proof, by (μ -eq), (μ -cong), and (trans), that two circular types are equal, see Lemma 7D.32 and the last clause to an application of rule (coind). ■
See also Exercise 8D.3.

8B.55. LEMMA. If $A =_{\mu}^* B$, then $\mathcal{S}(\mathcal{H}, A, B) = \text{true}$.

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 8B.53(ii) we know that such an n must certainly exist. The proof is by induction on n . If $n = 0$, then $A =_{\mu}^* B$ implies that $\mathcal{S}(\mathcal{H}, A, B) = \text{true}$, by Lemma 7E.27. If $n > 0$, then we are in the last case of Definition 8B.51. 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 $m_i < n$. Now use the induction hypotheses for $A_i =_{\mu}^* B_i$ and the fact that $A =_{\mu}^* B$ implies $A_i =_{\mu}^* B_i$, by Lemma 8B.33. ■

Now we can harvest.

8B.56. THEOREM (Completeness). Let $A, B \in \mathbb{T}_{\mu}$. Then the following are equivalent:

- (i) $A =_{\mu}^* B$;
- (ii) $\mathcal{S}(\emptyset, A, B) = \text{true}$;

(iii) $\vdash_{\text{BH}} A = B$.

PROOF. (i) \Rightarrow (ii). By Lemma 8B.55.

(ii) \Rightarrow (iii). By Lemma 8B.54.

(iii) \Rightarrow (i). By Corollary 8B.50. ■

Now Theorem 8B.44, Koster [1969], follows again as a corollary.

8B.57. COROLLARY. Given $A, B \in \mathbb{T}_\mu$, it is decidable whether $A =_\mu^* B$.

PROOF. By Theorem 8B.56 and Lemma 8B.53. ■

8B.58. REMARK. The connection between the present proof of this result via the system of Brandt-Henglein (BH) and the one using the deductive closure presented in 8B.44 is elaborated in Grabmayer [2005], where it is shown that the two are a kind of mirror image of each other.

8B.59. COROLLARY. If $A, B \in \mathbb{T}_\mu$ are not circular and $\vdash_{\text{BH}} A = B$, then there is a derivation of $A = B$ in BH such that rule (μ -cong) is not used.

PROOF. By completeness and Lemma 8B.54. ■

The predicate \mathcal{S} defined in 8B.51 is a computable procedure to test equality of μ -types. In the present form \mathcal{S} is $O(2^{n \times m})$ where n, m are, respectively, the number of arrow types in $\mathcal{SC}^s(A)$ and $\mathcal{SC}^s(B)$. More efficient algorithms are known, as mentioned in the beginning of this subsection.

Other systems

Other systems have been proposed in the literature to give a complete axiomatization of $=_\mu^*$. In Amadio and Cardelli [1993] a formal system is mentioned to prove strong equivalence (that we will denote by (μ_{AC}^*)) defined by adding to the rules of system (μ) , see Definition 7D.26, the following rule.

8B.60. DEFINITION. (i) The rule (AC) is defined as follows.

$$(AC) \quad \frac{A[\beta := B] = B \quad A[\beta := B'] = B'}{B = B'}$$

In Ariola and Klop [1996] an equivalent but slightly different rule has been introduced, see Exercise 8D.13.

(ii) The system μ extended with the rule (AC) is denoted by (μ_{AC}^*) .

The soundness of this system can be proved by a standard induction on derivations using the uniqueness of fixed points in Tr_{inf} Theorem 7F.5, when the last applied rule is (AC). In fact we have that both $(\mu\alpha.B)^*$ (by Remark 7E.29) and $(A)^*$ (by the induction hypothesis and Lemma 7E.28) are fixed points of $\lambda\zeta \in \text{Tr}_{\text{inf}}. (B)^*[\alpha := \zeta]$. In Ariola and Klop [1996] it is proved that also (μ_{AC}^*) is complete with respect to the tree semantics. So (μ_{AC}^*) is equivalent to (BH).

Rule (AC) 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.

8B.61. EXAMPLE. In (μ_{AC}^*) we have $\vdash \mu\alpha.(\beta \rightarrow \alpha) = \mu\alpha.(\beta \rightarrow \beta \rightarrow \alpha)$. Indeed, by two applications of (μ -eq) we have $\vdash \mu\alpha.(\beta \rightarrow \alpha) = \beta \rightarrow \beta \rightarrow \mu\alpha.(\beta \rightarrow \alpha)$. Then we can apply

rule (AC) with $B[\alpha] = \beta \rightarrow \beta \rightarrow \alpha$. Compare this proof with that of Exercises 8D.5 and 8D.4.

Some general properties of strong equality can be easily proved by rule (AC). 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.

8B.62. PROPOSITION. *The following are directly provable by the rule (AC).*

- (i) $\mu\alpha.A = A$, if α does not occur in A ;
- (ii) $\mu\alpha\beta.A = \mu\beta\alpha.A$;
- (iii) $\mu\alpha\beta.A(\alpha, \beta) = \mu\alpha.A(\alpha, \alpha)$.

PROOF. Do Exercise 8D.11. ■

8B.63. REMARK. The notion of bisimulation over infinite trees, see Definition 7E.9, is formulated directly over their representation in \mathbb{T}_μ . A relation $R_\mu \subseteq \mathbb{T}_\mu \times \mathbb{T}_\mu$ is a μ -bisimulation if the following statements hold. Here $\text{ls}(A)$ is the lead-symbol of A , defined in Definition 7D.34.

$$\begin{aligned} A R_\mu B &\Rightarrow \text{ls}(A) \equiv \text{ls}(B); \\ A R_\mu \mu\alpha.B &\Rightarrow A R_\mu B[\alpha := \mu\alpha.B]; \\ \mu\alpha.A R_\mu B &\Rightarrow A[\alpha := \mu\alpha.A] R_\mu B; \\ A \rightarrow B R_\mu A' \rightarrow B' &\Rightarrow A R_\mu A' \& B R_\mu B'. \end{aligned}$$

It is easy to prove that for any pair of recursive types $A, B \in \mathbb{T}_\mu$ and μ -bisimulation R_μ one has

$$A R_\mu B \Rightarrow A =_\mu^* B.$$

8C. Properties of types defined by an sr over \mathbb{T}

In this section we will study some fundamental properties of type algebras defined via type equations and simultaneous recursions over a set $\mathbb{T} = \mathbb{T}^\mathbb{A}$ 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 of weak equivalence for an sr

For an sr \mathcal{R} over \mathbb{T} we already have proved invertibility of $=_{\mathcal{R}}$ in Theorem 7C.12 by introducing the TRS with as notion of reduction $X_i \Rightarrow_{\mathcal{R}} A_i(\vec{X})$. Decidability follows as a particular case of Theorem 7B.15.

Decidability for recursive types defined by an sr \mathcal{R} can also be proved via the ‘inverse’ term rewriting system (TRS), see Statman [1994], Terese [2003], that generates $=_{\mathcal{R}}$ and is complete, i.e. Church-Rosser (CR) and strongly normalizing (SN). We present this proof here since the inverse TRS will be used also in the proof of Theorem 8C.29. Moreover its properties can suggest efficient algorithms to test type equality.

The first step consists of orienting the equations of \mathcal{R} .

8C.1. DEFINITION. Let $\mathcal{R} = \{X_i = A_i \mid 1 \leq i \leq n\}$ be a proper sr over \mathbb{T} . The rewriting system $\text{TRS}^{-1}(\mathcal{R})$ is generated by the notion of reduction

$$\{A_i \Rightarrow_{\mathcal{R}}^{-1} X_i \mid X_i = A_i \in \mathcal{R}\}.$$

Then $=_{\mathcal{R}}^{-1}$ is the convertibility relation generated by $\text{TRS}^{-1}(\mathcal{R})$.

8C.2. PROPOSITION. If \mathcal{R} is proper, then $\text{TRS}^{-1}(\mathcal{R})$ is SN.

PROOF. Each contraction decreases the size of the type to which it is applied, except for $X_j \Rightarrow X_i$. But then we have $j > i$, since \mathcal{R} is proper, Definition 7C.8. Therefore we can make the following argument. Define for $A \in \mathbb{T}(\vec{X})$ the following numbers.

$$\begin{aligned} s(A) &\triangleq \text{number of symbols in } A; \\ n(A) &\triangleq \text{sum of the indices of variables among } \vec{X} \text{ in } A. \end{aligned}$$

Then reducing A , the pair $\langle s(A), n(A) \rangle$ decreases in the lexicographical order. ■

However $\text{TRS}^{-1}(\mathcal{R})$ is, in general, not CR, as we show now.

8C.3. EXAMPLE. Let \mathcal{R} be the sr

$$\begin{aligned} X_0 &= X_0 \rightarrow X_2 \\ X_1 &= (X_0 \rightarrow X_2) \rightarrow X_2 \\ X_2 &= X_0 \rightarrow X_1. \end{aligned}$$

Then $\text{TRS}^{-1}(\mathcal{R})$ consists of the rules

$$\begin{aligned} X_0 \rightarrow X_2 &\Rightarrow_{\mathcal{R}} X_0 \\ (X_0 \rightarrow X_2) \rightarrow X_2 &\Rightarrow_{\mathcal{R}} X_1 \\ X_0 \rightarrow X_1 &\Rightarrow_{\mathcal{R}} X_2. \end{aligned}$$

Observe that the LHS of the first equation is a subterm of the LHS of the second one. In particular $(X_0 \rightarrow X_2) \rightarrow X_2$ can be reduced both to X_1 and to $X_0 \rightarrow X_2$ which further reduces to X_0 : it has then two distinct normal forms X_1 and X_0 . Therefore $\text{TRS}^{-1}(\mathcal{R})$ is not CR.

Expressions like $X_0 \rightarrow X_2$ and $(X_0 \rightarrow X_2) \rightarrow X_2$ in the example above are called *critical pairs* in the literature on term rewriting systems. In $\text{TRS}^{-1}(\mathcal{R})$ there is a critical pair whenever there are i, j such that $i \neq j$ and A_i is a subexpression of A_j . The following result is well-known.

8C.4. THEOREM (Knuth-Bendix). Let \mathcal{T} be a TRS that is SN.

- (i) If all critical pairs of \mathcal{T} have a common reduct, then \mathcal{T} is CR.
- (ii) If \mathcal{T} has no critical pairs, then it is CR.

PROOF. (i) See [Terese \[2003\]](#) Theorem 2.7.16.

(ii) By (i). ■

Now we present an algorithm for transforming any proper sr into a logically equivalent one, see Definition 7C.16 (ii), without critical pairs. The procedure amounts to a simple case of the Knuth-Bendix completion algorithm, see [Terese \[2003\]](#) Theorem 7.4.2, as the equations involved are between closed terms.

8C.5. PROPOSITION. Let \mathcal{R} be a proper sr. Then there exists a proper sr \mathcal{R}^\diamond such that

- (i) \mathcal{R}^\diamond is a proper sr logically equivalent to \mathcal{R} .

(ii) $\text{TRS}^{-1}(\mathcal{R}^\diamond)$ is complete, i.e. SN and CR.

PROOF. Let \mathcal{R} be a proper sr. We define by recursion on n a sequence of sets of equations \mathcal{D}_n , \mathcal{I}_n ($n \geq 0$) such that

- (a) \mathcal{D}_n is a proper sr;
- (b) \mathcal{I}_n is a set of equations of the form $X_i = X_j$ with $i < j$;
- (c) $\mathcal{D}_n \cup \mathcal{I}_n$ is logically equivalent to \mathcal{R} , for all n .

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 as follows.

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 $\mathcal{I}_{n+1} = \mathcal{I}_n$.

4. The algorithm terminates if $\mathcal{D}_{n+1} = \mathcal{D}_n$. Let N be the least n such that $\mathcal{D}_{n+1} = \mathcal{D}_n$. Define

$$\begin{aligned}\mathcal{R}^\diamond &= \mathcal{D}_N \cup \mathcal{I}_N; \\ \text{TRS}^\diamond(\mathcal{R}) &= \text{TRS}^{-1}(\mathcal{R}^\diamond).\end{aligned}$$

This algorithm terminates, as can be seen as follows. In case (2) and in case (1) if $A_j \notin \mathbb{A} \cup \{\vec{X}\}$, the number of symbols in \mathcal{D}_n decreases. In case (1), if $A_j \in \mathbb{A} \cup \{\vec{X}\}$, then

- (i) an α is replaced by X_j ; or
- (ii) an X_k is replaced by X_j , with $j < k$.

After a finite number of applications of (1) or (2) we must eventually apply a rule in which the number of symbols in \mathcal{D}_n decreases and so eventually the process must stop.

(i) By construction each $\mathcal{D}_n \cup \mathcal{I}_n$ is a proper sr that is logically equivalent to \mathcal{R} . In particular this holds for $\mathcal{R}^\diamond = \mathcal{D}_N \cup \mathcal{I}_N$.

(ii) Note that $\text{TRS}^\diamond(\mathcal{R})$ is SN, by Proposition 8C.2 and (i). Claim: $\text{TRS}^\diamond(\mathcal{R})$ has no critical pairs. Indeed $\text{TRS}^{-1}(\mathcal{D}_N)$ has no such pairs, otherwise we could apply step 1 or 2 of the definition of \mathcal{D}_n 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 the definition of \mathcal{D}_n , 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 is put in any \mathcal{I}_n for $n > k$. By Theorem 8C.4(ii) it follows that \Rightarrow is CR. ■

EXAMPLE. Applying the above algorithm to the sr \mathcal{R} defined in Example 8C.3 we obtain (assuming $X_0 < X_1 < X_2$)

$$\begin{aligned}\mathcal{D}_1 &= \{X_0 = X_0 \rightarrow X_2, X_1 = X_0 \rightarrow X_2, X_2 = X_0 \rightarrow X_1\}, \\ \mathcal{I}_1 &= \emptyset;\end{aligned}$$

$$\begin{aligned}\mathcal{D}_2 &= \{X_1 = X_1 \rightarrow X_2, X_2 = X_1 \rightarrow X_1\}, \\ \mathcal{I}_2 &= \{X_0 = X_1\}.\end{aligned}$$

Now 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. We obtain an sr \mathcal{R}^\diamond and $\text{TRS}^\diamond(\mathcal{R})$ as follows.

$$\left. \begin{array}{lcl} X_0 & = & X_1 \\ X_1 & = & X_1 \rightarrow X_2 \\ X_2 & = & X_1 \rightarrow X_1 \end{array} \right\} \mathcal{R}^\diamond; \quad \left. \begin{array}{lcl} X_1 & \Rightarrow_{\mathcal{R}^\diamond} & X_0 \\ X_1 \rightarrow X_2 & \Rightarrow_{\mathcal{R}^\diamond} & X_1 \\ X_1 \rightarrow X_1 & \Rightarrow_{\mathcal{R}^\diamond} & X_2 \end{array} \right\} \text{TRS}^\diamond(\mathcal{R}).$$

8C.6. COROLLARY. Let \mathcal{R} be a proper sr. Then $=_{\mathcal{R}}$ is decidable.

PROOF. Since \mathcal{R}^\diamond is logically equivalent to \mathcal{R} , it follows that also $=_{\mathcal{R}}$ is the convertibility relation generated by $\text{TRS}^\diamond(\mathcal{R})$. For a type C , let C^{nf} be its (unique) normal form with respect to $\text{TRS}^\diamond(\mathcal{R})$. This nf exists and is computable from C , since $\text{TRS}^\diamond(\mathcal{R})$ is complete. Now, given a pair of types $A, B \in \mathbb{T}[\vec{X}]$, we have

$$A =_{\mathcal{R}} B \Leftrightarrow A^{\text{nf}} \equiv B^{\text{nf}}.$$

This is decidable. ■

Alternatively decidability of $=_{\mathcal{R}}$ follows also as a particular case of Proposition 7B.15. A more algebraic but less direct proof is given in [Marz \[1999\]](#).

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 an sr all the results and the techniques valid for the equational theories. To this aim we need some preliminary definitions and lemmas.

8C.7. DEFINITION. Let $\mathcal{R} = \mathcal{R}(\vec{X})$ be an sr over $\mathbb{T}^{\mathbb{A}}$.

(i) An sr $\mathcal{R}[\vec{X}]$ is *flat* if all equations of \mathcal{R} are of the form $X = \alpha$ where $\alpha \in \mathbb{A}$ is an atomic type or $X = Y \rightarrow Z$, where $X, Y, Z \in \vec{X}$

(ii) An indeterminate $X \in \vec{X}$ is called an *orphan* in \mathcal{R} if X does not occur in one of the right hand sides of \mathcal{R} .

8C.8. EXAMPLE. Note that a flat sr is simplified. Any sr can be transformed in an equivalent flat one by adding and possibly removing indeterminates. For example $\mathcal{R}_1 = \{X = X \rightarrow (\alpha \rightarrow \beta), Y = (X \rightarrow X) \rightarrow Y\}$ can be ‘flattened’ to

$$\mathcal{R}'_1 \triangleq \{X = X \rightarrow Z, Z = U \rightarrow V, U = \alpha, V = \beta, Y = W \rightarrow Y, W = X \rightarrow X\}.$$

Note that $\mathcal{R}_2 \triangleq \{X = Y \rightarrow Z, Y = Z \rightarrow Y, Z = Z \rightarrow Z, W = X \rightarrow Y\}$ has W as orphan. Removing it by considering $\mathcal{R}'_2 \triangleq \mathcal{R}_2 - \{W = X \rightarrow Y\}$ yields an equivalent sr, as $\mathbb{T}^{\mathbb{A}}[\mathcal{R}'_2] \cong$

$\mathbb{T}^A[\mathcal{R}_2]$ by mapping $[W]$ onto $[X \rightarrow Y]$. But now X has become an orphan in \mathcal{R}'_2 . We can remove also X and obtain $\mathcal{R}''_2 \triangleq \{Y = Z \rightarrow Y, Z = Z \rightarrow Z\}$, that is without orphans.

8C.9. LEMMA. *Each proper sr can be transformed into an equivalent flat one without orphans.*

PROOF. Let $\mathcal{R}(\vec{X})$ be a proper sr over \mathbb{T} . First transform $\mathcal{R}(\vec{X})$ into an equivalent simplified sr $\mathcal{R}'(\vec{X}')$ applying the construction in the proof of Lemma 7C.18. Now take any equation $X = A_1 \rightarrow A_2 \in \mathcal{R}'$ such that either A_1 or A_2 is not an indeterminate. Assume that A_1 is not an indeterminate. Then replace in \mathcal{R}' the equation $X = A_1 \rightarrow A_2$ by the two equations $X = Y \rightarrow A_2$, $Y = A_1$, where Y is a fresh new indeterminate. Do similarly if A_2 is not an indeterminate. Repeat these steps until the resulting sr is flat; 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. The example given shows that orphans can be removed successively. The proof that at each step one has $\mathbb{T}^A[\mathcal{R}] \cong \mathbb{T}^{A'}[\mathcal{R}']$ can be given in a way similar to that in Proposition 7C.18. ■

8C.10. DEFINITION. Let $\mathcal{R} = \mathcal{R}[\vec{X}]$ be a flat sr. For each equivalence class $[X]$, for $X \in \vec{X}$, with respect to the relation $=_{\mathcal{R}}^*$, choose a representative X' . We define the sr $\mathcal{R}^* = \mathcal{R}^*[\vec{X}]$ in two steps as follows. First let $\mathcal{R}' = \mathcal{R}'[\vec{X}']$ be the flat sr constructed as follows.

$$\begin{aligned}\mathcal{R}' \triangleq \{X' = \alpha \mid (X' = \alpha) \in \mathcal{R}\} \cup \\ \{X' = Y' \rightarrow Z' \mid \exists Y \in [Y'], Z \in [Z']. (X' = Y \rightarrow Z) \in \mathcal{R}\}\end{aligned}$$

Then define $\mathcal{R}^* = \mathcal{R}^*[\vec{X}]$ as

$$\mathcal{R}^* \triangleq \{X = \alpha \mid (X = \alpha) \in \mathcal{R}\} \cup \{X = Y' \rightarrow Z' \mid X \in [X'], X' = Y' \rightarrow Z' \in \mathcal{R}'\}.$$

8C.11. EXAMPLE. For $\mathcal{R} \triangleq \{X = Y \rightarrow Z, Y = Z \rightarrow X, Z = X \rightarrow Y\}$ we have $\mathcal{R}' = \{X = X \rightarrow X\}$ and $\mathcal{R}^* = \{X = X \rightarrow X, Y = X \rightarrow X, Z = X \rightarrow X\}$.

A different construction of \mathcal{R}^* is given in Exercise 8D.17.

We will show that weak equality w.r.t. \mathcal{R}^* is equivalent to strong equality w.r.t. \mathcal{R} .

8C.12. LEMMA. (i) $X \in [X'] \Rightarrow X =_{\mathcal{R}^*} X'$.

(ii) $A =_{\mathcal{R}} B \Rightarrow A =_{\mathcal{R}^*} B$.

PROOF. (i) By the Definition of \mathcal{R}^* from \mathcal{R}' in Definition 8C.10.

(ii) It suffices to show

$$(X = B) \in \mathcal{R} \Rightarrow X =_{\mathcal{R}^*} B.$$

The case $(X = \alpha) \in \mathcal{R}$ is trivial. Now let $(X = Y \rightarrow Z) \in \mathcal{R}$. Then

$$X \in [X_1], Y \in [Y_1], Z \in [Z_1], (X_1 = Y_1 \rightarrow Z_1) \in \mathcal{R}^*.$$

Then the result follows from (i). ■

8C.13. THEOREM. *Given a flat sr $\mathcal{R}(\vec{X})$, define \mathcal{R}^* as in Definition 8C.10. Then*

(i) *For all $A, B \in \mathbb{T}[\vec{X}]$*

$$A =_{\mathcal{R}^*} B \Leftrightarrow A =_{\mathcal{R}}^* B.$$

(ii) $(\mathbb{T}[\mathcal{R}])^* \cong \mathbb{T}[\mathcal{R}^*]$.

PROOF. (i) (\Rightarrow) It suffices to show

$$(X = A(\vec{X})) \in \mathcal{R}^* \Rightarrow X =_{\mathcal{R}}^* A(\vec{X}).$$

Case 1. $A(\vec{X}) \equiv \alpha$. Then $(X = \alpha) \in \mathcal{R}^*$, hence also $(X = \alpha) \in \mathcal{R}$.

Case 2. $A(\vec{X}) = Y \rightarrow Z$. Then $(X = Y \rightarrow Z) \in \mathcal{R}^*$, and $(X_1 = Y_1 \rightarrow Z_1) \in \mathcal{R}$, $X =_{\mathcal{R}}^* X_1$, $Y =_{\mathcal{R}}^* Y_1$, and $Z =_{\mathcal{R}}^* Z_1$. Then clearly $X =_{\mathcal{R}}^* Y_1 \rightarrow Z_1$.

(\Leftarrow) Assume $(A)_{\mathcal{R}}^* = (B)_{\mathcal{R}}^*$. We show $A =_{\mathcal{R}^*} B$ by induction on $s(A) + s(B)$, where $s(C)$ is the number of symbols in $C \in \mathbb{T}[\vec{X}']$. We distinguish the following cases.

- (1) $A = \alpha, B = \alpha$;
- (2) $A = \alpha, B = X, (X = \alpha) \in \mathcal{R}$;
- (3) $A = Y, B = \alpha, (Y = \alpha) \in \mathcal{R}$;
- (4) $A = Y, B = Z$;
- (5) $A = Y, B = B_1 \rightarrow B_2, (Y = A_1 \rightarrow A_2) \in \mathcal{R}$;
- (6) $A = A_1 \rightarrow A_2, B = Y, (Y = B_1 \rightarrow B_2) \in \mathcal{R}$;
- (7) $A = A_1 \rightarrow A_2, B = B_1 \rightarrow B_2$.

The cases (1)-(3) are trivial.

Case (4). Now $Y \in [Y']$, $Z \in [Z']$, where $Y', Z' \in \vec{X}'$. Then $(Y)_{\mathcal{R}}^* = (Z)_{\mathcal{R}}^*$, so $[Y] = [Z]$, and $Y' = Z'$. By Lemma 8C.12 we have $Y =_{\mathcal{R}^*} Y', Z =_{\mathcal{R}^*} Z'$. Therefore $Y =_{\mathcal{R}^*} Z$.

Case (5). $A_1, A_2 \in \vec{X}$, because \mathcal{R} is flat, say $(Y = X_1 \rightarrow X_2) \in \mathcal{R}$. Now $(B_1 \rightarrow B_2)_{\mathcal{R}}^* = (X_1 \rightarrow X_2)_{\mathcal{R}}^*$, hence by invertibility, Lemma 7E.5,

$$B_1 =_{\mathcal{R}}^* X_1, B_2 =_{\mathcal{R}}^* X_2$$

and by the induction hypothesis

$$B_1 =_{\mathcal{R}^*} X_1, B_2 =_{\mathcal{R}^*} X_2.$$

Therefore $B_1 \rightarrow B_2 =_{\mathcal{R}^*} X_1 \rightarrow X_2 =_{\mathcal{R}^*} Y$, by Lemma 8C.12.

Case (6). Similar to Case (5).

Case (7). Similar to Case (5), but easier.

(ii) From (i) we immediately get that the identity morphism $\mathbb{T}[\vec{X}] \rightarrow \mathbb{T}[\vec{X}]$ induces an isomorphism $\mathbb{T}[\mathcal{R}^*] \xrightarrow{\sim} (\mathbb{T}[\mathcal{R}])^*$. ■

8C.14. COROLLARY. *For every proper sr \mathcal{R} there exists a flat sr \mathcal{R}^* such that*

$$\mathbb{T}[\mathcal{R}^*] \cong (\mathbb{T}[\mathcal{R}])^*.$$

PROOF. Let \mathcal{R} be proper. By Lemma 8C.9 there exists a flat sr \mathcal{R}_1 such that $\mathbb{T}[\mathcal{R}] \cong \mathbb{T}[\mathcal{R}_1]$. One easily shows that also $(\mathbb{T}[\mathcal{R}])^* \cong (\mathbb{T}[\mathcal{R}_1])^*$. Now apply the Theorem. ■

Axiomatization of strong equivalence for sr

Given a proper sr \mathcal{R} Theorem 8C.13 (ii) shows that $=_{\mathcal{R}}^*$ can be axiomatized by the rules of Definition 7A.10 simply by taking \mathcal{R}^* instead of \mathcal{R} . However, in a way similar to what we have done in section 8B for $=_{\mu}^*$, we can also define a coinductive system (\mathcal{R}^*) that directly axiomatizes $=_{\mathcal{R}}^*$. Also in this system we have judgments of the form $\mathcal{H} \vdash A = B$ in which \mathcal{H} is a set of equations of the shape $A \rightarrow A' = B \rightarrow B'$. As in system (BH), Definition 8B.45, the crucial point is the introduction of a rule (coind).

8C.15. DEFINITION. Let \mathcal{R} be a proper sr. The system (\mathcal{R}^*) is defined by the following axioms and rules.

(ident)	$\mathcal{H} \vdash A = A$
(symm)	$\frac{\mathcal{H} \vdash A = B}{\mathcal{H} \vdash B = A}$
(trans)	$\frac{\mathcal{H} \vdash A = B \quad \mathcal{H} \vdash B = C}{\mathcal{H} \vdash A = C}$
(axiom)	$\mathcal{H} \vdash A = B, \quad \text{if } A = B \in \mathcal{H},$
(R-eq)	$\mathcal{H} \vdash X = A, \quad \text{if } X = A \in \mathcal{R},$
(coind)	$\frac{\mathcal{H}, (A \rightarrow B) = (A' \rightarrow B') \vdash A = A' \quad \mathcal{H}, (A \rightarrow B) = (A' \rightarrow B') \vdash B = B'}{\mathcal{H} \vdash (A \rightarrow B) = (A' \rightarrow B')}$

FIGURE 26. The system (\mathcal{R}^*)

We write $\mathcal{H} \vdash_{\mathcal{R}}^* A = B$ to mean that $\mathcal{H} \vdash A = B$ can be derived by the above rules.

In this system rule (\rightarrow -cong) is missing but it is easy to prove that it is derivable. Note that in this system there are no types having properties analogous to these of the circular types in \mathbb{T}_μ , \mathbb{T}_μ^* .

EXAMPLE. Let $\mathcal{R}_1 \triangleq \{X = A \rightarrow A \rightarrow X\}$ where A is any type. Then we have $\vdash_{\mathcal{R}_1}^* X = A \rightarrow X$, with the following proof. Let C denote $A \rightarrow A \rightarrow X$.

$$\begin{array}{c}
 & & (\mathcal{R}\text{-eq}) & & (\text{hyp}) \\
 & & \hline
 & (ident) & \{C = A \rightarrow X\} \vdash X = C & \{C = A \rightarrow X\} \vdash C = A \rightarrow X & \\
 & \hline
 & (\mathcal{R}\text{-eq}) & \{C = A \rightarrow X\} \vdash A = A & \{C = A \rightarrow X\} \vdash X = A \rightarrow X & (\text{trans}) \\
 \hline
 & \vdash X = C & & \vdash C = A \rightarrow X & & (\text{coind}), (\text{symm}) \\
 & & & \hline
 & & & \vdash X = A \rightarrow X & & (\text{trans})
 \end{array}$$

With the same technique as used in Section 8B 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 8B.51 (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 $A =_{\mathcal{R}}^* B$. This yields the following.

8C.16. THEOREM. *Let $\mathcal{R}(\vec{X})$ be a proper sr over \mathbb{T} .*

- (i) $\vdash_{\mathcal{R}}^* A = B \Leftrightarrow A =_{\mathcal{R}}^* B$.
- (ii) *Given $A, B \in \mathbb{T}[\vec{X}]$ it is decidable whether $A =_{\mathcal{R}}^* B$.*

Justifying type equations by an sr

In the study of recursive type inference it will be useful to know whether a given sr justifies a given set of equations, according to definition 7A.17(ii). We prove that this is a decidable property, which result is needed in section 9B. The original proof is due to Statman [1994].

In the rest of this subsection we show the decidability of the existence of morphisms between algebras of the form $\mathbb{T}^{\mathbb{A}}[\mathcal{R}]$. By Proposition 7C.18(ii) and Lemma 8C.9 one has $\mathbb{T}^{\mathbb{A}}[\mathcal{R}] \cong \mathbb{T}^{\mathbb{A}'}[\mathcal{R}']$, where \mathcal{R}' is flat and without orphans. Moreover this isomorphism is effective. Hence in a proof that the existence of morphisms is decidable one may assume that the \mathcal{R} are flat and without orphans.

8C.17. DEFINITION. Let $\mathcal{A}, \mathcal{A}'$ be type algebras.

- (i) A set of *constraints* is of the form $\mathcal{C} \subseteq \mathcal{A} \times \mathcal{A}'$.
- (ii) A morphism $h : \mathcal{A} \rightarrow \mathcal{A}'$ is said to *agree with* \mathcal{C} if it satisfies $h(a) = a'$, for all $\langle a, a' \rangle \in \mathcal{C}$.
- (iii) Two sets of constraints $\mathcal{C}, \mathcal{C}' \subseteq \mathcal{A} \times \mathcal{A}'$ are *equivalent*, if

$$\text{for all morphisms } h : \mathcal{A} \rightarrow \mathcal{A}'. [h \text{ agrees with } \mathcal{C} \Leftrightarrow h \text{ agrees with } \mathcal{C}']$$

It will be shown that the existence of an $h : \mathbb{T}^{\mathbb{A}}/\mathcal{E} \rightarrow \mathbb{T}^{\mathbb{A}'}[\mathcal{R}']$ agreeing with a finite set of constraints \mathcal{C} is decidable. The proof occupies 8C.18-8C.28. For the proof substantial help was obtained from Dexter Kozen and Jan Willem Klop (personal communications).

8C.18. DEFINITION. Let \mathcal{A} be a type algebra and $a, b, c \in \mathcal{A}$.

- (i) If $a = b \rightarrow c$, then b and c are called *direct descendants* of a . Direct descendants are *descendants*. And direct descendants of descendants are descendants. Write $a \rightsquigarrow b$ if b is a direct descendant of a and $a \rightsquigarrow^+ b$ if b is a descendant of a . The relation \rightsquigarrow^* is the reflexive transitive closure of \rightsquigarrow .
- (ii) An element $a \in \mathcal{A}$ is called *cyclic* if $a \rightsquigarrow^+ a$.
- (iii) Write $C(\mathcal{A}) = \{a \in \mathcal{A} \mid a \text{ is cyclic}\}$.

For example in $\mathbb{T}^{\{a\}}[X = X \rightarrow a]$ the element $[X]$ is cyclic, as $[X] = [X] \rightarrow [a]$. Notice that not every indeterminate $X \in \vec{X}$ of \mathcal{R} needs to be cyclic. For example if $\mathcal{R} = \{X = X \rightarrow Y, Y = a\}$, then only $[X]$ but not $[Y]$ is cyclic in $\mathbb{T}^{\mathbb{A}}[\mathcal{R}]$.

8C.19. NOTATION. Let \mathcal{R} be an sr over $\mathbb{T}^{\mathbb{A}}$.

- (i) Write $I(\mathcal{R}) \triangleq \{[X] \in \mathbb{T}^{\mathbb{A}}[\mathcal{R}] \mid X \text{ an indeterminate of } \mathcal{R}\}$, the set of indeterminates in $\mathbb{T}^{\mathbb{A}}[\mathcal{R}]$.

- (ii) Write $C(\mathcal{R}) \triangleq C(\mathbb{T}^{\mathbb{A}}[\mathcal{R}])$, the set of cyclic elements of $\mathbb{T}^{\mathbb{A}}[\mathcal{R}]$.

We will see that for a flat sr without orphans \mathcal{R} one has $C(\mathcal{R}) \subseteq I(\mathcal{R})$.

8C.20. LEMMA. Let $h : \mathcal{A} \rightarrow \mathcal{B}$ be a morphism and $a, b \in \mathcal{A}$.

- (i) Suppose \mathcal{B} is invertible. Let $a = b \rightarrow c$. Then $h(b), h(c)$ are uniquely determined by $h(a)$.
- (ii) $a \rightsquigarrow b \Rightarrow h(a) \rightsquigarrow h(b)$.
- (iii) a is cyclic $\Rightarrow h(a)$ is cyclic.

PROOF. (i) Because \mathcal{B} is invertible.

- (ii) Suppose $a \rightsquigarrow b$. If, say, $a = b \rightarrow c$, then $h(a) = h(b) \rightarrow h(c)$, so $h(a) \rightsquigarrow h(b)$.
- (iii) By (ii). ■

8C.21. LEMMA. Let \mathcal{R} be a flat sr over $\mathbb{T}^{\mathbb{A}}$ and $a, b \in \mathbb{T}^{\mathbb{A}}[\mathcal{R}]$, with $a \rightsquigarrow b$ and $a = [A]_{\mathcal{R}}$.

- (i) Then $A \equiv X$ or $A \equiv A_1 \rightarrow A_2$, for some $A_1, A_2 \in \mathbb{T}^{\mathbb{A} \cup \vec{X}}$.
- (ii) In case $A \equiv X$ one has $(X = X_1 \rightarrow X_2) \in \mathcal{R}$ and $b = [X_i]$ for some $i \in \{1, 2\}$.
- (iii) In case $A \equiv A_1 \rightarrow A_2$ one has $b = [A_i]_{\mathcal{R}}$, for some $i \in \{1, 2\}$.

PROOF. Note that A has a unique pair of direct descendants because $\mathbb{T}[\mathcal{R}]$ is invertible by Proposition 7C.12.

(i) Suppose $A \equiv \alpha$. Then $\alpha =_{\mathcal{R}} C \rightarrow D$ for some $C, D \in \mathbb{T}[\vec{X}]$. But then $\alpha \Rightarrow_{\mathcal{R}}^* (C' \rightarrow D')$ by the Church-Rosser Theorem for $\Rightarrow_{\mathcal{R}}^*$, Proposition 7C.10(ii), a contradiction.

(ii) and (iii) The case $(X = \alpha) \in \mathcal{R}$ goes as the case $A \equiv \alpha$ in (i). The rest is immediate by the invertibility of $\mathbb{T}[\mathcal{R}]$. ■

8C.22. LEMMA. *Let \mathcal{R} be flat. Let a, b range over $\mathbb{T}^{\mathbb{A}}[\mathcal{R}]$.*

- (i) $a \rightsquigarrow b \ \& \ a \in I(\mathcal{R}) \Rightarrow b \in I(\mathcal{R})$.
- (ii) *If*

$$a = a_1 \rightsquigarrow a_2 \rightsquigarrow \cdots \rightsquigarrow a_n \rightsquigarrow \cdots$$

is an infinite chain of descendants, then for some k one has $a_k \in I(\mathcal{R})$.

PROOF. (i) Let $a \in I(\mathcal{R})$. Then $a = [X]$, hence by Lemma 8C.21(ii) one has $(X = X_1 \rightarrow X_2) \in \mathcal{R}$ and $b = [X_i]$, for some i .

(ii) Let $a_1 = [A]$. If $A \in \vec{X}$, then we are ready. Else by Lemma 8C.21(i) and (iii) one has $A \equiv A_1 \rightarrow A_2$ and $a_2 = [A_i]$, for some i . Thus continuing, by splitting the type $A_1 \rightarrow A_2 \in \mathbb{T}^{\mathbb{A} \cup \vec{X}}$ in its components, one eventually obtains $a_k = [Y]$, by Lemma 8C.21(i). ■

8C.23. COROLLARY. *Let \mathcal{R} be flat without orphans.*

- (i) $\forall a \in I(\mathcal{R}) \exists b \in C(\mathcal{R}). b \rightsquigarrow^* a$.
- (ii) $C(\mathcal{R}) \subseteq I(\mathcal{R})$.

PROOF. (i) Since \mathcal{R} is without orphans, each $a \in I(\mathcal{R})$ has a ‘father’ $a' \in I(\mathcal{R})$. Going backwards along the ancestors, since the set $I(\mathcal{R})$ is finite, we must end up in a cyclic element $a^{(k)} \in C(\mathcal{R})$. Then $a^{(k)} \rightsquigarrow^* a$, and we are done.

- (ii) Let $a \in C(\mathcal{R})$. Then

$$a \rightsquigarrow^+ a \rightsquigarrow^+ \cdots$$

Hence for some $b \in I(\mathcal{R})$ one has $a \rightsquigarrow^+ b \rightsquigarrow^+ a$, by (ii) of the Lemma. But then $a \in I(\mathcal{R})$, by (i) of the Lemma. ■

8C.24. PROPOSITION. *Let $\mathcal{R}(\vec{X}), \mathcal{R}'(\vec{X}')$ be sr-s, over $\mathbb{T} = \mathbb{T}^{\mathbb{A}}, \mathbb{T}^{\mathbb{A}'}$ respectively, and let $\mathcal{A} = \mathbb{T}^{\mathbb{A}}[\mathcal{R}], \mathcal{A}' = \mathbb{T}^{\mathbb{A}'}[\mathcal{R}']$.*

- (i) *It is decidable whether there exists a morphism $h : \mathcal{A} \rightarrow \mathcal{A}'$.*
- (ii) *Let moreover $\mathcal{C} \subseteq \mathcal{A} \times \mathcal{A}'$ be finite. Then it is decidable whether*

there exists a morphism $h : \mathcal{A} \rightarrow \mathcal{A}'$ agreeing with \mathcal{C} .

PROOF. (i) By Proposition 7C.18(ii) and Lemma 8C.9 we may assume that \mathcal{R} and \mathcal{R}' are flat and without orphans.

Define a *proto-morphism* to be a map $h : I(\mathcal{R}) \rightarrow I(\mathcal{R}')$ such that

$$a = b \rightarrow c \Rightarrow h(a) = h(b) \rightarrow h(c).$$

Claim 1. For a morphism $h : \mathcal{A} \rightarrow \mathcal{A}'$ the restriction $h \upharpoonright I(\mathcal{R})$ is a proto-morphism. We only need to show that if $a \in I(\mathcal{R})$, then $h(a) \in I(\mathcal{R}')$. By Corollary 8C.23(i) there exists a $b \in C(\mathcal{R})$ such that $b \rightsquigarrow^* a$. But then as \mathcal{R}' is flat one has $h(b) \in C(\mathcal{R}') \subseteq I(\mathcal{R}')$,

by Lemma 8C.20(iii) and Corollary 8C.23(ii). By Lemma 8C.20(ii) one has $h(b) \rightsquigarrow^* h(a)$. Therefore $h(a) \in I(\mathcal{R}')$, by Lemma 8C.22(i).

$$\begin{array}{ccc} b \in C(\mathcal{R}) & \xrightarrow{h} & h(b) \in C(\mathcal{R}') \subseteq I(\mathcal{R}') \\ \left\{ \begin{array}{c} * \\ \downarrow \end{array} \right. & & \left\{ \begin{array}{c} * \\ \downarrow \end{array} \right. \\ a \in I(\mathcal{R}) & \xrightarrow{h} & h(a) \in I(\mathcal{R}') \end{array}$$

Claim 2. Any proto-morphism $h : I(\mathcal{R}) \rightarrow I(\mathcal{R}')$ can be extended to a morphism $h^+ : \mathcal{A} \rightarrow \mathcal{A}'$.

For an X occurring in \mathcal{R} choose $h_0(X)$ such that $h([X]) = [h_0(X)]$. Then extend h_0 to a morphism $h_0^+ : \mathbb{T}^{\mathbb{A} \cup \vec{X}} \rightarrow \mathbb{T}^{\mathbb{A}' \cup \vec{X}'}$ by recursion, with a case distinction for the base case.

$$\begin{array}{lll} \text{Basis} & h_0^+(\alpha) & = h_0(X), \quad \text{if } (X = \alpha) \in \mathcal{R}, \\ & & = \text{arbitrary,} \quad \text{otherwise,} \\ & h_0^+(X) & = h_0(X); \\ \text{Recursion} & h_0^+(A \rightarrow B) & = h_0^+(A) \rightarrow h_0^+(B). \end{array}$$

Then $A =_{\mathcal{R}} B \Rightarrow h_0^+(A) =_{\mathcal{R}'} h_0^+(B)$. Hence by Proposition 7B.13(i) h_0^+ induces the required morphism $h^+ : \mathcal{A} \rightarrow \mathcal{A}'$:

$$h^+([A]) = [h_0^+(A)].$$

Claim 3. There exists a morphism $h : \mathcal{A} \rightarrow \mathcal{A}'$ iff there exists a proto-morphism $h_0 : I(\mathcal{R}) \rightarrow I(\mathcal{R}')$. Immediate by Claims 1 and 2.

Claim 4. It is decidable whether there exists a proto-morphism $h_0 : I(\mathcal{R}) \rightarrow I(\mathcal{R}')$. As $I(\mathcal{R}), I(\mathcal{R}')$ are finite there are only finitely many candidates. By Proposition 8C.6 the relation $=_{\mathcal{R}}$ is decidable, hence also the requirement to be a proto-morphism.

Now we are done by Claims 3 and 4.

(ii) Instead of considering equivalence classes, we use elements of $\mathbb{T}^{\mathbb{A} \cup \vec{X}}, \mathbb{T}^{\mathbb{A}' \cup \vec{X}'}$ and work modulo $=_{\mathcal{R}}, =_{\mathcal{R}'}$, respectively. Given a set of constraints \mathcal{C} one can construct an equivalent set \mathcal{C}' such that the elements are of the form

$$\begin{aligned} & \langle \alpha, A' \rangle, \\ & \langle X, \alpha' \rangle, \\ & \langle X, Y' \rightarrow Z' \rangle, \\ & \langle A \rightarrow B, \alpha' \rangle. \end{aligned}$$

Indeed, other possible forms are $\langle X, X' \rangle, \langle A \rightarrow B, X' \rangle$ and $\langle A \rightarrow B, A' \rightarrow B' \rangle$. Using invertibility the last pair can be replaced equivalently by $\{\langle A, A' \rangle, \langle B, B' \rangle\}$. In the first pair $\langle X, X' \rangle$ the type X' can be replaced by the RHS of X' in \mathcal{R}' , and similarly in the second pair. Let us call the resulting \mathcal{C}' a *simplified set of constraints*. A proto-morphism

$h : \mathcal{A} \rightarrow \mathcal{A}'$ is called *consistent with* \mathcal{C}' if the following holds.

$$\begin{aligned}\langle \alpha, A' \rangle \in \mathcal{C}' &\Rightarrow h(X) = A', & \text{in case } (X = \alpha) \in \mathcal{R}, \\ \langle X, \alpha' \rangle \in \mathcal{C}' &\Rightarrow h(X) = \alpha' \\ \langle X, Y' \rightarrow Z' \rangle \in \mathcal{C}' &\Rightarrow h(X) = Y' \rightarrow Z' \\ \langle A \rightarrow B, \alpha' \rangle \notin \mathcal{C}'\end{aligned}$$

The reason to require $\langle A \rightarrow B, \alpha' \rangle \notin \mathcal{C}'$ is that $h(A \rightarrow B) = \alpha$ is impossible for a morphism h , as $\alpha \neq_{\mathcal{R}'} A' \rightarrow B'$.

Now similarly to the claims in (i), one can show the following.

Claim 3'. There exists a morphism $h : \mathcal{A} \rightarrow \mathcal{A}'$ agreeing with \mathcal{C} iff there exists a proto-morphism $h : I(\mathcal{R}) \rightarrow I(\mathcal{R}')$ consistent with \mathcal{C}' , a simplified form of \mathcal{C} .

Claim 4'. It is decidable whether there exists a proto-morphism $h : I(\mathcal{R}) \rightarrow I(\mathcal{R}')$ consistent with a simplified set \mathcal{C}' .

From claims 3', 4' the result follows. ■

In Statman [1994] it is shown that the justifiability of an sr in another sr is an NP-complete problem.

Now we like to obtain this result also for type algebras of the shape $\mathcal{A} = \mathbb{T}^{\mathbb{A}}/\mathcal{E}$. We do this by extending the axiomatization of type equality to force a type algebra generated by a set of equations \mathcal{E} to have the invertibility property.

8C.25. DEFINITION. Write $\mathcal{E} \vdash^{\text{inv}} a = b$ if there is a proof of $\vdash a = b$ by the axioms and rules for equality given in Definition 7A.10 extended by the following two rules for invertibility

$$(\text{inv}) \quad \frac{\vdash a_1 \rightarrow a_2 = b_1 \rightarrow b_2}{\vdash a_i = b_i} \quad (i = 1, 2)$$

We define for $a, b \in \mathbb{T}$ the relation $a =_{\mathcal{E}}^{\text{inv}} b \Leftrightarrow \mathcal{E} \vdash^{\text{inv}} a = b$.

8C.26. LEMMA. Let \mathcal{E} a system of type equations. Then there exists a \mathcal{E}^{inv} which is a proper sr such that

$$\mathcal{E} \vdash^{\text{inv}} \mathcal{E}^{\text{inv}} \& \mathcal{E}^{\text{inv}} \vdash \mathcal{E}.$$

In other words, \mathcal{E}^{inv} is equivalent to \mathcal{E} for provability with the extra rule of invertibility.

PROOF. Given \mathcal{E} over $\mathbb{T}^{\mathbb{A}}$. The relation $=_{\mathcal{E}}^{\text{inv}}$ is the least invertible congruence containing $=_{\mathcal{E}}$. We now define an sr which generates $=_{\mathcal{E}}^{\text{inv}}$.

Let $\alpha_0, \alpha_1, \dots$ denote the elements of \mathbb{A} . As in Definition 8C.5(i) define by induction on n sets of equations $\mathcal{D}_n, \mathcal{I}_n$ ($n \geq 0$). Let $\mathcal{D}_0 \triangleq \mathcal{E}$ and $\mathcal{I}_0 \triangleq \emptyset$. 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 $(A \rightarrow B = A' \rightarrow B') \in \mathcal{D}_n$, then take

$$\mathcal{D}_{n+1} \triangleq \mathcal{D}_n - \{A \rightarrow B = A' \rightarrow B'\} \cup \{A = A', B = B'\}$$

and $\mathcal{I}_{n+1} \triangleq \mathcal{I}_n$;

2. Replace in \mathcal{D}_n all equations of the form $A \rightarrow B = \alpha$ by $\alpha = A \rightarrow B$;
3. If $(\alpha_i = A \rightarrow B), (\alpha_i = A' \rightarrow B') \in \mathcal{D}_n$, then take

$$\mathcal{D}_{n+1} \triangleq \mathcal{D}_n - \{\alpha_i = A' \rightarrow B'\} \cup \{A = A', B = B'\},$$

assuming $\text{dpt}(A \rightarrow B) \leq \text{dpt}(A' \rightarrow B')$, see Definition 1A.21(i) and take $\mathcal{I}_{n+1} \triangleq \mathcal{I}_n$;

4. If $(\alpha_i = \alpha_i) \in \mathcal{D}_n$ for some i , then take $\mathcal{D}_{n+1} \triangleq \mathcal{D}_n - \{\alpha_i = \alpha_i\}$ and $\mathcal{I}_{n+1} \triangleq \mathcal{I}_n$;

5. If $(\alpha_i = \alpha_j) \in \mathcal{D}_n$, with $i \neq j$, then take

$$\begin{aligned}\mathcal{D}_{n+1} &\triangleq \mathcal{D}_n[\alpha_h := \alpha_k] - \{\alpha_i = \alpha_j\} \\ \mathcal{I}_{n+1} &\triangleq \mathcal{I}_n \cup \{\alpha_h = \alpha_k\},\end{aligned}$$

where $h = \min(i, j)$ and $k = \max(i, j)$;

6. Otherwise take $\mathcal{D}_{n+1} \triangleq \mathcal{D}_n$ and $\mathcal{I}_{n+1} \triangleq \mathcal{I}_n$.
7. Let N be the least n such that $\mathcal{D}_{n+1} = \mathcal{D}_n$ and $\mathcal{I}_{n+1} = \mathcal{I}_n$. We will show below that this number exists. Write $\mathcal{D}_{\mathcal{E}} \triangleq \mathcal{D}_N, \mathcal{I}_{\mathcal{E}} \triangleq \mathcal{I}_N$.
8. Finally define $\mathcal{E}^{\text{inv}} \triangleq \mathcal{D}_N \cup \mathcal{I}_N$.

Claim. The number N in step 7 exists. Indeed, to $D = \{A_1 = B_1, \dots, A_n = B_n\}$ we assign

$$S_D = \{d(A_1), d(B_1), \dots, d(A_n), d(B_n)\},$$

considered as multiset. Steps (1), (3)-(6) of the algorithm decrease the multiset order, see Definition 2A.6. Step (2) keeps the multiset fixed. Since the multiset order is well-founded, Lemma 2A.8, it follows that a potential infinite path D_1, D_2, D_3, \dots ends up by only performing step (2). But this is impossible, by counting the number of type atoms on the RHSs.

It is easy to prove, by induction on n , that for all $n \geq 0$

$$\mathcal{D}_n \cup \mathcal{I}_n \vdash \mathcal{E} \text{ and } \mathcal{E} \vdash^{\text{inv}} \mathcal{D}_n \cup \mathcal{I}_n$$

To show that $\mathcal{D}_N \cup \mathcal{I}_N$ is a proper sr, note that, by step (2) and (4) above, \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 now give some applications of this result.

8C.27. PROPOSITION. *Let \mathcal{A} be a type algebra. Then there exists an invertible type algebra \mathcal{A}^{inv} initial under \mathcal{A} . I.e. there is a morphism $k : \mathcal{A} \rightarrow \mathcal{A}^{\text{inv}}$ such that all $h : \mathcal{A} \rightarrow \mathcal{B}$, with \mathcal{B} invertible, can be factored through k ; that is $h = i \circ k$, for some $i : \mathcal{A}^{\text{inv}} \rightarrow \mathcal{B}$.*

$$\begin{array}{ccc}\mathcal{A} & \xrightarrow{h} & \mathcal{B} \\ k \downarrow & \nearrow i & \\ \mathcal{A}^{\text{inv}} & & \end{array}$$

PROOF. By Lemma 7A.16 we can assume that $\mathcal{A} = \mathbb{T}/\sim$ is a syntactic algebra. Then there is a morphism $h^\sharp : \mathbb{T} \rightarrow \mathcal{B}$ corresponding to h according to Proposition 7B.4(iii).

Now define $\mathcal{A}^{\text{inv}} = \mathbb{T}/\approx^{\text{inv}}$ where \approx^{inv} is a shorthand for $=_{\approx}^{\text{inv}}$ defined in 8C.26. Now note that h^\sharp can also be seen as a syntactic morphism from \mathbb{T} to \mathcal{B} preserving \approx^{inv} . In fact, if $A_1 \rightarrow A_2 \approx B_1 \rightarrow B_2$ we must have $A_i \approx^{\text{inv}} B_i$ ($i = 1, 2$) by rule (inv) but also $h^\sharp(A_i) = h^\sharp(B_i)$ since \mathcal{B} is invertible. Now take $k : \mathcal{A} \rightarrow \mathcal{A}^{\text{inv}}$ such that $k([A]_{\approx}) = [A]_{\approx^{\text{inv}}}$ and i such that $i([A]_{\approx^{\text{inv}}}) = h^\sharp(A) = h([A]_{\approx})$. ■

8C.28. COROLLARY. *Let $\mathcal{A} = \mathbb{T}^{\mathbb{A}}/\mathcal{E}$, $\mathcal{A}' = \mathbb{T}^{\mathbb{A}'}[\mathcal{R}']$, with proper \mathcal{R}' , and $\mathcal{C} \subseteq \mathcal{A} \times \mathcal{A}'$. Then it is decidable whether*

$$\text{there exists a morphism } h : \mathcal{A} \rightarrow \mathcal{A}' \text{ agreeing with } \mathcal{C}.$$

PROOF. The existence of an $h : \mathcal{A} \rightarrow \mathcal{A}'$ is by Proposition 8C.27 equivalent to the existence of an $h_1 : \mathcal{A}^{\text{inv}} \rightarrow \mathcal{A}'$. Moreover, h agrees with \mathcal{C} iff h_1 agrees with $\mathcal{C}_1 = \{\langle k(a), a' \rangle \mid \langle a, a' \rangle \in \mathcal{C}\}$. Therefore by Proposition 8C.24(ii) we are done. ■

It is possible to strengthen Proposition 8C.24(i) as follows.

8C.29. THEOREM. Let $\mathcal{E}, \mathcal{E}'$ be finite sets of equations over $\mathbb{T}^{\mathbb{A}}$ and $\mathbb{T}^{\mathbb{A}'}$ respectively. Then it is decidable whether there exists a morphism $h : \mathbb{T}^{\mathbb{A}}/\mathcal{E} \rightarrow \mathbb{T}^{\mathbb{A}'}/\mathcal{E}'$.

PROOF. This is proved in [Kozen \[1977\]](#). ■

8D. Exercises

8D.1. Prove Theorem 8A.7. [Hint. Argue by induction on A . Note that the union of two sr, provided the indeterminates are disjoint, is still an sr.]

8D.2. Prove Proposition 7D.29.

8D.3. Let (triv) be the following rule

$$(\text{triv}) \quad M = N, \text{ for } M, N \text{ circular.}$$

Let $A_1 \equiv \mu\beta.(\mu\alpha.\alpha \rightarrow \beta)$ and $B_1 \equiv \mu\beta.(\mu\alpha.\alpha \rightarrow \beta) \rightarrow \beta$.

(i) Show that one has $\vdash_{\text{BH}} A_1 = B_1$, without using rule (μ -cong).

(ii) Let (BH^\sim) be the system (BH) with (μ -cong) replaced by (triv). Show

$$\vdash_{\text{BH}} A = B \Leftrightarrow \vdash_{\text{BH}^\sim} A = B.$$

[Hint. Use Lemma 8B.54 and Theorem 8B.56(ii), (iii).]

(iii) Show that (triv) is not equivalent with (μ -cong) in system (μ) . [Hint. Let system (μ^\sim) be system (μ) with (μ -cong) replaced by (triv). Show that $A_1 = B_1$ is not derivable in (μ^\sim) .]

8D.4. Show that $\vdash_{\text{BH}} \mu\alpha.(\beta \rightarrow \beta \rightarrow \alpha) = \mu\alpha.(\beta \rightarrow \alpha)$, where β is any type variable, see Examples 7D.27 and 8B.61.

8D.5. (i) Prove $\mu\alpha.((\alpha \rightarrow \alpha) \rightarrow \alpha) = \mu\alpha.(\alpha \rightarrow \alpha \rightarrow \alpha)$ in (BH) .

(ii) Prove directly $\mu\alpha.((\alpha \rightarrow \alpha) \rightarrow \alpha) =_\mu^* \mu\alpha.(\alpha \rightarrow \alpha \rightarrow \alpha)$.

Compare the proofs with that in (μ_{AC}^*) .

8D.6. [Grabmayer] Show that \Rightarrow_μ^{-1} is not CR. [Hint. Take the following type:

$$A = (\mu s.s \rightarrow \mu t.(s \rightarrow t)) \rightarrow \mu t'.((\mu s.s \rightarrow \mu t.(s \rightarrow t)) \rightarrow t').$$

Show that $A = B[t' := \mu t'.B] = C[s := \mu s.C]$ for two types $B \neq C$ irreducible with respect to \Rightarrow_μ^{-1} .]

[[One exercise deleted and replaced by the following.]]

8D.7. Define $\mathcal{SC}^w(a) \subset \mathbb{T}_{\mu'}$ as follows.

$$\mathcal{SC}^w(\alpha) = \{\alpha\};$$

$$\mathcal{SC}^w(A_1 \rightarrow A_2) = \{A_1 \rightarrow A_2\} \cup \mathcal{SC}^w(A_1) \cup \mathcal{SC}^w(A_2);$$

$$\mathcal{SC}^w(\mu\alpha.A_1) = \{\mu\alpha.A_1\} \cup (\alpha) \upharpoonright \mathcal{SC}^w(A_1) \cup \mathcal{SC}^w(A_1)[\alpha := \mu\alpha.A_1].$$

$$\mathcal{SC}^w((\vec{\beta})A) = (\vec{\beta}) \upharpoonright \mathcal{SC}^w(A).$$

(i) Show $\mathcal{SC}^w(A[\alpha := B]) \subseteq (\mathcal{SC}^w(A)[\alpha := B] \cup \mathcal{SC}^w(B))$.

(ii) Show $\mathcal{SC}^w(\mathcal{SC}^w(A)) = \mathcal{SC}^w(A)$.

(iii) Show $\mathcal{SC}^w_r(A) = \mathcal{SC}^w(A)$.

(iv) Show Lemma 8B.30.

- 8D.8. This exercise, inspired by Endrullis, Grabmayer, Klop, and van Oostrom [2011], is about \mathbb{T}_μ . The relation \rightsquigarrow over \mathbb{T}_μ is defined as follows.

$$\begin{aligned} (1) \quad (A \rightarrow B) &\rightsquigarrow A \\ (2) \quad (A \rightarrow B) &\rightsquigarrow B \\ (3) \quad (\dot{\mu}\alpha.A) &\rightsquigarrow A \\ (4) \quad (\dot{\mu}\alpha.A) &\rightsquigarrow (A[\alpha := \dot{\mu}\alpha.A]). \end{aligned}$$

(i) Write \rightsquigarrow^* for the transitive reflexive closure of \rightsquigarrow .

(ii) Write $A \rightsquigarrow_n B$ if $A \rightsquigarrow^* B$ in n steps.

(iii) Define $\mathcal{SC}^*(A) = \{B \in \mathbb{T}_\mu \mid A \rightsquigarrow^* B\}$. Show that $\mathcal{SC}^*(A)$ is always finite.

[Hint. Define the set of underlined $\dot{\mu}$ -types, notation $\underline{\mathbb{T}_\mu}$, for example $(\dot{\mu}\alpha.\alpha \rightarrow \dot{\mu}\beta\dot{\mu}\gamma.(\alpha \rightarrow \beta \rightarrow \gamma))$ and $(\dot{\mu}\alpha.\alpha \rightarrow \dot{\mu}\beta\dot{\mu}\gamma.(\alpha \rightarrow \beta \rightarrow \gamma))$ are in $\underline{\mathbb{T}_\mu}$. For $B \in \underline{\mathbb{T}_\mu}$ write $|B| \in \underline{\mathbb{T}_\mu}$ for the type obtained from B by erasing all underlinings. On $\underline{\mathbb{T}_\mu}$ we define \simeq , a variant of \rightsquigarrow .

$$\begin{aligned} (1) \quad (A \rightarrow B) &\simeq A \\ (2) \quad (A \rightarrow B) &\simeq B \\ (3) \quad (\dot{\mu}\alpha.A) &\simeq A \\ (4) \quad (\dot{\mu}\alpha.A) &\simeq (A[a := \dot{\mu}a.A]). \end{aligned}$$

Show that $\{B \in \underline{\mathbb{T}_\mu} \mid A \simeq B\}$ is finite, for all $A \in \underline{\mathbb{T}_\mu}$. Finally, prove that for every \rightsquigarrow -reduction without repetitions $R : A \equiv B_0 \rightsquigarrow B_1 \rightsquigarrow \dots \rightsquigarrow B_n$, there is a \simeq -reduction $R' : A \equiv B_0 \simeq B'_1 \simeq \dots \simeq B'_n$, such that $|B'_i| \equiv B_i$ for $1 \leq i \leq n$. This yields the conclusion.]

[[One exercise deleted and replaced by the following.]]

- 8D.9. Recall the definition of $\mathcal{SC}_r^s(A) = \{C \in \mathbb{T}_\mu \mid A \rightsquigarrow^* C\}$ in Definition 8B.37.

(i) Show the following. If $A[\alpha := \mu\alpha.A] \rightsquigarrow^* B$ then we have either

- (a) This reduction is $A[\alpha := \mu\alpha.A] \rightsquigarrow^* \mu\alpha.A \rightsquigarrow^* B$;
- (b) This reduction is a $[\alpha := \mu\alpha.A]$ instance of $A \rightsquigarrow^* C$ (So $B = C[\alpha := \mu\alpha.A]$).

(ii) Show $\mathcal{SC}_r^s(\mu\alpha.A_1) \subseteq \{\mu\alpha.A_1\} \cup \mathcal{SC}_r^s(A_1)[\alpha := \mu\alpha.A_1]$.

(iii) Show that $\mathcal{SC}_r^s(A)$ is finite.

- 8D.10. 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 8B.63 then $(A)_\mu^* = (B)_\mu^*$.

- 8D.11. Prove Proposition 8B.62.

- 8D.12. Let $\mathcal{E} \triangleq \{X = \alpha^p \rightarrow X, X = \alpha^q \rightarrow X\}$ and $\mathcal{R} \triangleq \{X = \alpha^r \rightarrow X\}$. Show that $\mathcal{R}(X)$ solves $\mathcal{E}(X)$ over $\mathbb{T}^{\{\alpha\}}(X)$ iff $r | \text{gcd}(p, q)$, where $n|m$ denotes ‘ n divides m ’.

- 8D.13. In Ariola and Klop [1994] the following rule is introduced.

$$(AK) \quad \frac{A[\alpha := B] = B}{\mu\alpha.A = B}$$

Show that (AK) is equivalent to (AC) in Definition 8B.60.

- 8D.14. Give a formulation of rule (AC) for a simultaneous recursion. Use it to prove $\vdash_{\{c=\beta \rightarrow \beta \rightarrow c\}}^* c = \beta \rightarrow c$.

- 8D.15. Prove Lemma 8B.41.

- 8D.16. Prove soundness, completeness and decidability of (\mathcal{R}^*) in Definition 8C.15. This then proves 8C.16.

8D.17. Define, inductively on n , the equivalence relation $R_n \subseteq \vec{X} \times \vec{X}$ ($n \geq 0$) as the minimal equivalence relation such that the following holds.

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 $X = Y \rightarrow Z$, $X' = Y' \rightarrow Z' \in \mathcal{R}$ for some $Y, Y', Z, Z' \in \vec{X}$;
3. $XR_{n+1}X'$ if $X = Y \rightarrow Z$, $X' = Y' \rightarrow Z' \in \mathcal{R}$ and YR_nY' and ZR_nZ' .

Let N be the first N such that $R_N = R_{N+1}$. Show

$$X =_{\mathcal{R}}^* Y \Leftrightarrow XR_NY.$$

The following is due to Grabmayer [2007] and is a direct proof of both the regularity of $(A)_{\mu}^*$ and decidability of $=_{\mu}^*$.

- 8D.18.
- (i) Let $A \in \mathbb{T}_{\mu}$. Then for all subtrees $T \subseteq (A)_{\mu}^*$ of the tree-unfolding of A there exists a $B \in \mathcal{SC}^s(A)$ such that $T = (B)_{\mu}^*$.
 - (ii) Show $\text{card}(\mathcal{SC}^s(A)) \leq l(A)$, where $l(A)$ is defined in Definition 8B.5.
 - (iii) A number n is *large enough* for a regular tree T if all subtrees of T occur at a p of length $< n$. Show that $l(A)$ is large enough for $(A)_{\mu}^*$.
 - (iv) For $A \in \mathbb{T}_{\mu}$ the tree $(A)_{\mu}^*$ is regular with at most $l(A)$ subtrees. [Hint. Use (i).] By (i) there is a surjection of $\mathcal{SC}^s(A)$, which is of cardinality $< l(A)$ by (ii), to the set of subtrees of $(A)_{\mu}^*$. Therefore also the latter set is of cardinality $< l(A)$ and hence finite.
 - (v) Conclude that $=_{\mu}^*$ is decidable.

PROOF. Let $A, B \in \mathbb{T}_{\mu}$. Given a sequence number $p \in \{0, 1\}^*$. By (iii) there are only finitely many different pairs $((A)_{\mu}^*)|p, (B)_{\mu}^*|p$. Therefore, in order to test $(A)_{\mu}^* = (B)_{\mu}^*$, we claim that one needs to examine these trees only up to depth $N = l(A) \cdot l(B) + 1$:

$$A =_{\mu}^* B \Leftrightarrow \forall p < N. (A)_{\mu}^*|p \cong (B)_{\mu}^*|p.$$

Indeed, \Rightarrow is obvious. As to \Leftarrow , assume the RHS and $A \neq_{\mu}^* B$ towards a contradiction. Then $(A)_{\mu}^*|p \neq (B)_{\mu}^*|p$, for some p . ■

CHAPTER 9

PROPERTIES OF TERMS WITH TYPES

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 9A we study the subject reduction property for recursive type systems, in Section 9B the problem of finding types for untyped terms and in Section 9C we study restrictions on recursive types such that strong normalization holds for terms that inhabit these.

9A. First properties of $\lambda_{\equiv}^{\mathcal{A}}$

We will show that for invertible \mathcal{A} the recursive type systems $\lambda_{\equiv}^{\mathcal{A}}$ have the subject reduction property, i.e.

$$\left. \begin{array}{c} \Gamma \vdash_{\mathcal{A}} M : a \\ M \rightarrow_{\beta(\eta)} M' \end{array} \right\} \Rightarrow \Gamma \vdash_{\mathcal{A}} M' : a,$$

where \vdash is defined in Section 7A.

This means that typings are stable with respect to β - or $\beta\eta$ -reduction, which is the fundamental evaluation process for λ -terms.

In general, however, types are not preserved under 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.

9A.1. LEMMA. $\Gamma, x:b \vdash_{\mathcal{A}} M : a, \Gamma \vdash_{\mathcal{A}} N : b \Rightarrow \Gamma \vdash_{\mathcal{A}} M[x:=N] : a$.

PROOF. By induction on the derivation of $\Gamma, x:b \vdash_{\mathcal{A}} M : a$. ■

9A.2. LEMMA. Suppose $x \notin \text{FV}(M)$, then

$$\Gamma, x:b \vdash_{\mathcal{A}} M : a \Rightarrow \Gamma \vdash M : a.$$

PROOF. By induction on the derivation of $\Gamma, x:b \vdash M : a$. ■

9A.3. PROPOSITION (Inversion Lemma). Let \mathcal{A} be a type algebra.

- (i) $\Gamma \vdash_{\mathcal{A}} x : a \Leftrightarrow (x:a) \in \Gamma$.
- (ii) $\Gamma \vdash_{\mathcal{A}} (MN) : a \Leftrightarrow \exists b \in \mathcal{A}. [\Gamma \vdash_{\mathcal{A}} M : (b \rightarrow a) \& \Gamma \vdash_{\mathcal{A}} N : b]$.
- (iii) $\Gamma \vdash_{\mathcal{A}} (\lambda x.M) : a \Leftrightarrow \exists b, c \in \mathcal{A}. [a = (b \rightarrow c) \& \Gamma, x:b \vdash_{\mathcal{A}} M : c]$.

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 ($\rightarrow E$).
- (iii) Now we can only use rule ($\rightarrow I$). ■

9A.4. REMARK. It is good to realize that the Inversion Lemma is a consequence of the following easy observation. For a derivation of e.g. $\Gamma \vdash_{\mathcal{A}} MN : a$, there is a $b \in \mathcal{A}$ such that this statement is a direct consequence of the statements $\Gamma \vdash_{\mathcal{A}} M : (b \rightarrow a)$ and $\Gamma \vdash_{\mathcal{A}} N : b$ in that same derivation. Note that there may be several $b \in \mathcal{A}$ such that $\Gamma \vdash_{\mathcal{A}} M : (b \rightarrow a)$ and $\Gamma \vdash_{\mathcal{A}} N : b$.

For syntactic type algebras this boils down to the following.

9A.5. COROLLARY (Inversion Lemma for syntactic type algebras). *Let $\mathcal{A} = \mathbb{T}/\approx$ be a syntactic type algebra. Then*

- (i) $\Gamma \vdash_{\mathbb{T}/\approx} x : A \Leftrightarrow \exists A' \in \mathbb{T}. [A' \approx A \& x : A' \in \Gamma]$.
- (ii) $\Gamma \vdash_{\mathbb{T}/\approx} (MN) : A \Leftrightarrow \exists B \in \mathbb{T}. [\Gamma \vdash_{\mathbb{T}/\approx} M : B \rightarrow A \& \Gamma \vdash_{\mathbb{T}/\approx} N : B]$.
- (iii) $\Gamma \vdash_{\mathbb{T}/\approx} (\lambda x.M) : A \Leftrightarrow \exists B, C \in \mathbb{T}. [A \approx (B \rightarrow C) \& \Gamma, x : B \vdash_{\mathbb{T}/\approx} M : C]$.

PROOF. From Proposition 9A.3 and Notation 7A.18. ■

9A.6. COROLLARY (Inversion Lemma II). *Let \mathcal{A} be an invertible type algebra. Then*

$$\Gamma \vdash_{\mathcal{A}} \lambda x.M : (a \rightarrow b) \Leftrightarrow \Gamma, x : a \vdash_{\mathcal{A}} M : b.$$

PROOF. (\Rightarrow) If $\Gamma \vdash \lambda x.M : a \rightarrow b$, then by Proposition 9A.3(iii) we have $a \rightarrow b = c \rightarrow d$ and $\Gamma, x : c \vdash M : d$. By invertibility $a = c$ and $b = d$. Hence $\Gamma, x : a \vdash M : b$.

(\Leftarrow) By rule ($\rightarrow I$). ■

Now we can prove the subject reduction property for one step $\beta\eta$ -reduction.

9A.7. LEMMA. (i) *If $\Gamma \vdash_{\mathcal{A}} \lambda x.(Mx) : a$ and $x \notin FV(M)$, then $\Gamma \vdash_{\mathcal{A}} M : a$.*

(ii) *If moreover \mathcal{A} is invertible, then*

$$\Gamma \vdash_{\mathcal{A}} (\lambda x.M)N : a \Rightarrow \Gamma \vdash_{\mathcal{A}} (M[x := N]) : a.$$

PROOF. (i) By the Inversion Lemma 9A.3(iii) we have that $\Gamma, x : b \vdash (Mx) : c$ for some $b, c \in \mathcal{A}$ such that $a = (b \rightarrow c)$. Moreover, by Lemma 9A.3 (ii) and (i) we have that $\Gamma, x : b \vdash M : d \rightarrow c$ and $\Gamma, x : b \vdash x : d$ for some type d such that $d = b$. Then $(d \rightarrow c) = (b \rightarrow c) = a$. Then $\Gamma \vdash M : a$, by Lemma 9A.2.

(ii) Suppose $\Gamma \vdash (\lambda x.M)N : a$. By the Inversion Lemma 9A.3(iii)

$$\Gamma \vdash (\lambda x.M) : b \rightarrow a \& \Gamma \vdash N : b,$$

for some type b . Moreover, by Lemma 9A.3(iii), there are types b' and a' such that $b \rightarrow a = b' \rightarrow a'$ and

$$\Gamma, x : b' \vdash M : a'.$$

Then $b' = b$ and $a' = a$, by invertibility. Hence $\Gamma \vdash N : b'$ and by Lemma 9A.1

$$\Gamma \vdash (M[x := N]) : a'. ■$$

9A.8. COROLLARY (Subject Reduction for $\lambda_{\underline{\mathcal{A}}}$). *Let \mathcal{A} be an invertible type algebra. Then $\lambda_{\underline{\mathcal{A}}}$ satisfies the subject reduction property for $\beta\eta$.*

PROOF. By Lemma 9A.7. ■

Invertibility of \mathcal{A} is a characterization of all the type algebras \mathcal{A} such that $\vdash_{\mathcal{A}}$ has the subject reduction property for β -reduction. The proof is from [Statman \[1994\]](#).

9A.9. THEOREM. *Let \mathcal{A} be a type algebra. Then*

$$\mathcal{A} \text{ is invertible} \Leftrightarrow \lambda_{\leq}^{\mathcal{A}} \text{ satisfies the subject reduction property for } \beta.$$

PROOF. (\Rightarrow) By Lemma 9A.8.

(\Leftarrow) Let $a, a', b, b' \in \mathcal{A}$ and assume $a \rightarrow b = a' \rightarrow b'$. 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 9A.3(i) $b = b'$.

Now take $\Gamma_2 = \{x:(a \rightarrow b), y:(a \rightarrow a), z:a'\}$. 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 9A.3(ii) there is a type $c \in \mathbb{T}$ such that $\Gamma_2 \vdash x : c \rightarrow b'$ and $\Gamma_2 \vdash (yz) : c$. Again by the Inversion Lemma 9A.3 (iii), (i) the second statement implies $a \rightarrow a = a' \rightarrow c$. By the first part of the proof we then have $a = c$ and so $a \rightarrow a = a' \rightarrow a$. Now use this to prove $x:a' \vdash ((\lambda y.y)x) : a$. A final application of subject reduction and the Inversion Lemma yields $a = a'$. ■

9B. Finding and inhabiting types

In this section we consider the problem of finding the (possibly empty) collection of types for a given untyped λ -term $M \in \Lambda$ in a system $\lambda_{\leq}^{\mathcal{A}}$ à la Curry³. The questions that can be asked about typing terms in recursive type systems are similar to those introduced in Section 2C, but now we have one more parameter: the type algebra in which we want to work. For simplicity we work only with closed terms. So assume $M \in \Lambda^o$. The problems are the following.

1. *Type reconstruction*

- [1a] Find all type algebras \mathcal{A} and all types $a \in \mathcal{A}$ such that $\vdash_{\mathcal{A}} M : a$.
- [1b] Given a type algebra \mathcal{A} find all types $a \in \mathcal{A}$ such that $\vdash_{\mathcal{A}} M : a$.

2. *Typability*

The problems can be seen as a particular case of both 1a. and 1b. Now we ask whether the set of all possible types of a given term is empty or not. Thus for a given λ -term M the typability problems corresponding to 1a and 1b are:

- [2a] Does there exist a type algebra \mathcal{A} and $a \in \mathcal{A}$ such that $\vdash_{\mathcal{A}} M : a$?
- [2b] Given \mathcal{A} , does there exist an $a \in \mathcal{A}$ such that $\vdash_{\mathcal{A}} M : a$?

3. *Type checking*

Given \mathcal{A} and $a \in \mathcal{A}$, do we have $\vdash_{\lambda_{\leq}^{\mathcal{A}}} M : a$?

In the next two problems the role of terms and types are reversed: we start with a type algebra \mathcal{A} and element $a \in \mathcal{A}$.

4. *Enumeration*

Find all $M \in \Lambda$ such that $\vdash_{\lambda_{\leq}^{\mathcal{A}}} M : a$.

5. *Inhabitation*

Is there a term $M \in \Lambda$ such that $\vdash_{\lambda_{\leq}^{\mathcal{A}}} M : a$?

³As in the case of simple types this kind of problem is much simpler for explicitly typed terms. The well formed terms of the typed calculi à la Church have 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 (equal). But we have seen in Chapter 8 that all interesting type equivalences are decidable. Type checking is even simpler in the system $\lambda_{\mu}^{\text{Ch}}$ where also type equivalence is explicit. In the case of the typed calculi à la de Bruijn the type information is given only partially, which sometimes leads to undecidability. See [Schubert \[1998\]](#).

Stylistically these problems can be rendered as follows.

Type reconstruction	$\vdash_{\mathcal{A}} M : ?$	$\vdash_{\lambda\mathcal{A}} M : ?$
Typability	$\exists \mathcal{A}, a \in \mathcal{A}. \vdash_{\lambda\mathcal{A}} M : a?$	$\exists a \in \mathcal{A}. \vdash_{\lambda\mathcal{A}} M : a?$
Type checking		$\vdash_{\lambda\mathcal{A}} M : a?$
Enumeration		$\vdash_{\lambda\mathcal{A}} ?: a;$
Inhabitation	$\exists M \in \Lambda. \vdash_{\lambda\mathcal{A}} M : a?$	

Problems 1, 2 could be called ‘finding types’; problem 3 ‘fitting’; and problems 4, 5 ‘finding terms’.

From Corollary 9A.3 it follows that for invertible \mathcal{A} one has

$$x_1:a_1, \dots, x_n:a_n \vdash_{\mathcal{A}} M : a \Leftrightarrow \vdash_{\mathcal{A}} (\lambda x_1 \cdots x_n. M) : (a_1 \rightarrow \dots \rightarrow a_n \rightarrow a).$$

So in this case it is not restrictive to formulate these questions for closed terms only. We will not consider the possibility of having a type environment Γ as parameter for non-invertible type algebras. The reader is invited to study this situation.

The problem of inhabitation is sometimes trivial. As pointed out in Remark 7E.37(i), for instance, in the systems $\lambda\mu$, $\lambda\mu^*$ all types are inhabited by $\Omega = (\lambda x.xx)(\lambda x.xx)$. We will discuss the problem of finding inhabitants in $(\lambda\mathcal{R})$ at the end of this section.

We will present here the basic results concerning type reconstruction considering only pure λ -terms. The generalization to terms containing constants is rather straightforward and will be discussed in Section 11C.

Now we will answer the typing questions for syntactic type algebras \mathcal{A} . The answer to similar questions for μ -types follows easily from this case.

Finding types in type algebras

The most natural way to describe a type algebra is via a set of equations over a set \mathbb{T} of types. So in many of the above questions we will assume (without loss of generality) that \mathcal{A} is of the form \mathbb{T}/\mathcal{E} . Since an sr over \mathcal{A} is a particular set of equations over $\mathcal{A}(\vec{X})$ all results apply immediately to an sr as well. Only Theorem 9B.7 about type checking is specific for type algebras defined via an sr.

Using the notion of type algebra morphism, see Definition 7A.8, we can define, in a quite natural way, a notion of a principal type scheme for type assignments with respect to arbitrary type algebras. It is a quite natural generalization of Corollary 2C.15, the principal type scheme for $\lambda_{\equiv}^{\mathcal{A}}$.

Principal type scheme in $\lambda_{\equiv}^{\mathcal{A}}$

For every $M \in \Lambda$ a *principal triple* will be constructed: a type algebra $\mathcal{A}_M = \mathbb{T}^{\mathbf{c}_M}/\mathcal{E}_M$, where \mathbf{c}_M is a set of type atoms and \mathcal{E}_M is a set of equations over $\mathbb{T}^{\mathbf{c}_M}$, a type $a_M = [\alpha_M]_{\mathcal{E}_M}$, and a basis Γ_M over \mathcal{A}_M such that

- (i) $\Gamma_M \vdash_{\mathcal{A}_M} M : a_M$.
- (ii) $\Gamma \vdash_{\mathcal{A}} M : a \Leftrightarrow$ there is a morphism $h : \mathcal{A}_M \rightarrow \mathcal{A}$ such that $h(\Gamma_M) \subseteq \Gamma$ and $h(a_M) = a$.

Since we are now working with type algebras, we define a type reconstruction algorithm based on the use of type equations, rather than use unification as in Section 2C in Part

I. This approach was first introduced, for type reconstruction in $(\lambda \rightarrow)$, in [Curry \[1969\]](#) and has been followed by many other authors, see e.g. [Wand \[1987\]](#). We start with some simple examples, that explain the method. To make the examples more readable we will use numbers to denote atomic types and we will add to each reconstruction rule used in the deduction the type equation which makes it valid.

9B.1. EXAMPLE. In the following examples we use the notation as indicated in 7A.18, that is elements of $\mathbb{T}^{\mathbb{A}}/\mathcal{E}$ are denoted as elements of $\mathbb{T}^{\mathbb{A}}$, to be considered modulo \mathcal{E} .

(i) Let $M \triangleq \lambda x.xx$. In order to type M , we build from below the following derivation. (We will write α_n just as n .)

$$\frac{x:1 \vdash x : 1 \quad x:1 \vdash x : 1 \quad 1 = 1 \rightarrow 2}{\frac{x:1 \vdash xx : 2 \quad 3 = 1 \rightarrow 2}{\vdash \lambda x.xx : 3}}$$

This gives the triple

$$a_M = 3, \Gamma_M = \emptyset, \mathcal{A}_M = \mathbb{T}^{\mathbf{c}_M}/\mathcal{E}_M,$$

where $\mathbf{c}_M = \{1, 2, 3\}$ and $\mathcal{E}_M = \{1 = 1 \rightarrow 2, 3 = 1 \rightarrow 2\}$. We can simplify this triple to the isomorphic $a_M = 1, \Gamma_M = \emptyset, \mathcal{A}_M = \mathbb{T}^{\{1,2\}}/\{1 = 1 \rightarrow 2\}$, i.e.

$$\mathcal{A}_M = \mathbb{T}^{\{\alpha_1, \alpha_2\}}/\{\alpha_1 = \alpha_1 \rightarrow \alpha_2\}.$$

Indeed

$$\vdash_{\mathcal{A}_M} (\lambda x.xx) : 1$$

In order to show that this assignment is initial, suppose that $\Gamma \vdash_{\lambda \mathcal{A}} M : a_1$. Then one can reconstruct from below a derivation with the same shape, using Proposition 9A.3 and Remark 9A.4.

$$\frac{x:a_1 \vdash x : a_1 \quad x:a_1 \vdash x : a_1 \quad a_1 = a_1 \rightarrow a_2}{\frac{x:a_1 \vdash xx : a_2 \quad a_3 = a_1 \rightarrow a_2}{\vdash \lambda x.xx : a_3}}$$

The required morphism is determined by $h(k) = a_k$ for $k \leq 3$. Indeed one has $h(a_M) = a_3$ and $h(\Gamma_M) \subseteq \Gamma$.

(ii) If we consider the (open) term $M \equiv x(xx)$ we have, using Church notation to represent deduction in a more compact way, the following.

$$\vdash_{\mathcal{A}_M} x^1 (x^1 x^1)^2 : 3,$$

where $a_M = 3, \mathcal{A}_M = \mathbb{T}^{\{1,2,3\}}/\{1 = 1 \rightarrow 2 = 2 \rightarrow 3\}$ and $\Gamma_M = \{x^1\}$.

Moreover, if we want to consider type assignment with respect to invertible type algebras we can convert \mathcal{A}_M to an invertible type algebra \mathcal{A}_M^{inv} . Assuming invertibility we get $1 = 2 = 3$. Then \mathcal{A}_M^{inv} can be simplified to the *trivial* type algebra $\mathbb{T}^{\{1\}}/\{1 = 1 \rightarrow 1\}$.

(iii) If we consider terms having a simple type in $(\lambda \rightarrow)$, then the construction sketched above does not involve recursive definitions as in case (i). Moreover if we assume invertibility the resulting type algebra is isomorphic to a free one and we get the same principal type as in $(\lambda \rightarrow)$. Let $M \triangleq \mathbf{c}_2 \equiv \lambda f x. f(fx)$.

$$\begin{array}{c}
 \frac{f:1, x:2 \vdash f:1 \quad f:1, x:2 \vdash x : 2 \quad 1 = 2 \rightarrow 3}{f:1, x:2 \vdash fx : 3} \qquad 1 = 3 \rightarrow 4 \\
 \frac{}{f:1, x:2 \vdash f(fx) : 4} \qquad 5 = 2 \rightarrow 4 \\
 \frac{}{f:1 \vdash \lambda x.f(fx) : 5} \qquad 6 = 1 \rightarrow 5 \\
 \frac{}{\vdash \lambda fx.f(fx) : 6}
 \end{array}$$

Simplifying this gives the triple

$$\vdash_{\mathbb{T}/\mathcal{E}} \mathbf{c}_2 : (2 \rightarrow 3) \rightarrow (2 \rightarrow 4), \quad (1)$$

with $\mathcal{E} \triangleq \{2 \rightarrow 3 = 3 \rightarrow 4\}$. We can understand this by looking at

$$\vdash \lambda f^{(2 \rightarrow 3) = (3 \rightarrow 4)} x^2.(f^{3 \rightarrow 4}(f^{2 \rightarrow 3}x^2)^3)^4 : (2 \rightarrow 3) \rightarrow 2 \rightarrow 4.$$

Also in $\lambda_{\rightarrow}^{\mathbb{A}}$ this term M can be typed.

$$\vdash_{\lambda_{\rightarrow}^{\mathbb{A}}} \mathbf{c}_2 : (\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha, \quad (2)$$

Note that there is a morphism $h : \mathbb{T}/\mathcal{E} \rightarrow \mathbb{T}^{\alpha}$ determined by

$$h(2) = h(3) = h(4) = \alpha.$$

This h respects the equations in \mathcal{E} . In this way the type assignment (2) is seen to follow from (1), applying Lemma 7A.20.

Again assuming invertibility we get $2 = 3 = 4$ and so \mathbb{T}/\approx contains only identities. Then $\mathcal{A}_{\mathbf{c}_2}^{inv}$ is isomorphic to $\mathbb{T}^{\{\alpha\}}$, the free type algebra over the atomic type α .

(iv) $M \equiv \mathbb{I}$. Bottom-up we construct the following derivation-tree.

$$\frac{\frac{x:1 \vdash x : 1 \quad 2 = 1 \rightarrow 1}{\vdash (\lambda x.x) : 2} \qquad \frac{y:1' \vdash y : 1' \quad 2' = 1' \rightarrow 1'}{\vdash (\lambda y.y) : 2'}}{\vdash (\lambda x.x)(\lambda y.y) : 0} \qquad 2 = 2' \rightarrow 0$$

Hence $\mathcal{E} = \{1 \rightarrow 1 = (1' \rightarrow 1') \rightarrow 0\}$ on $\mathbb{T}^{\{0,1,1'\}}$ we have

$$\vdash_{\mathcal{E}} \mathbb{I} : 0, \quad (3)$$

In λ_{\rightarrow} this term has as principal type $\alpha \rightarrow \alpha$. This can be obtained as image of (3) under the morphism $h : \mathbb{T}^{\{0,1,1'\}} \rightarrow \mathbb{T}^{\{\alpha\}}$ defined by

$$\begin{aligned}
 h(0) &\triangleq \alpha \rightarrow \alpha; \\
 h(1) &\triangleq \alpha \rightarrow \alpha; \\
 h(1') &\triangleq \alpha.
 \end{aligned}$$

Note that it was important to keep the names of the bound variables of the two occurrences of \mathbb{I} different.

We present now the formal algorithm to build the principal triple of a term M . To simplify its definition we make the assumption that the names of all free and bound variables of M are distinct. This can always be achieved by α -conversion.

9B.2. DEFINITION. Let $M \in \Lambda$. Define a set of type constants \mathbf{c}_M , a type α_M , a basis Γ_M , a set of equations \mathcal{E}_M , and a type algebra \mathcal{A}_M with element a_M as follows. We do this by defining first for each subterm-occurrence $L \subseteq M$ for L not a variable a distinct type atom α_L . For variables x occurring in M we choose a fixed type α_x (for different occurrences the same α_x). Then we define \mathcal{E}_L for each subterm-occurrence $L \subseteq M$, obtaining this notion also for M as highest subterm of itself.

L	\mathcal{E}_L
x	\emptyset
PQ	$\mathcal{E}_P \cup \mathcal{E}_Q \cup \{\alpha_P = \alpha_Q \rightarrow \alpha_{PQ}\}$
$\lambda x.P$	$\mathcal{E}_P \cup \{\alpha_{\lambda x.P} = \alpha_x \rightarrow \alpha_P\}$

Define \mathbf{c}_M as the set of all atomic types α_L and α_x occurring in M . Finally we define

$$\begin{aligned}\Gamma_M &\triangleq \{x:[\alpha_x] \mid x \in \text{FV}(M)\}; \\ \mathcal{A}_M &\triangleq \mathbb{T}^{\mathbf{c}_M}/\mathcal{E}_M; \\ a_M &\triangleq [\alpha_M]_{\mathcal{E}_M}.\end{aligned}$$

The type a_M is called the *principal recursive type* of M and \mathcal{A}_M its *principal type algebra*. Γ_M is the *principal recursive basis* of M , which is empty if M is closed. The triple $\langle \Gamma_M, \mathcal{A}_M, a_M \rangle$ is called *principal recursive triple*. Note that \mathcal{E}_M is an sr, if we consider the α_L as indeterminates.

Type reconstruction

Of the following theorem part (iii) solves both versions of the type reconstruction problem.

9B.3. THEOREM. *For every $M \in \Lambda$ the principal recursive triple $\langle \Gamma_M, \mathcal{A}_M, a_M \rangle$ satisfies the following.*

- (i) $\Gamma_M \vdash_{\mathcal{A}_M} M : a_M$.
- (ii) $\Gamma \vdash_{\mathcal{A}} M : a \quad\Leftrightarrow\quad \text{there is a morphism } h : \mathcal{A}_M \rightarrow \mathcal{A} \text{ such that } h(\Gamma_M) \subseteq \Gamma \text{ and } h(a_M) = a.$
- (iii) *For $M \in \Lambda^\varnothing$ this simplifies to*

$$\vdash_{\mathcal{A}} M : a \quad\Leftrightarrow\quad \exists h : \mathcal{A}_M \rightarrow \mathcal{A}. h(a_M) = a.$$

The triple is ‘principal’, as it is the initial one giving M a type satisfying (i) and (ii).

PROOF. Take as triple the one defined in the previous Definition.

(i) By induction on the structure of $L \subseteq M$ we show that this statement holds for M replaced by L and hence also for M itself. Case $L \equiv x$. Then clearly $x:\alpha_x \vdash x : \alpha_x$. Case $L \equiv PQ$, and $\Gamma_P \vdash_{\mathbb{T}/\mathcal{E}_P} P : \alpha_P$, $\Gamma_Q \vdash_{\mathbb{T}/\mathcal{E}_Q} Q : \alpha_Q$. Then $\Gamma_P \cup \Gamma_Q \vdash_{\mathbb{T}/\mathcal{E}} PQ : \alpha_{PQ}$, as $\alpha_P =_{\mathcal{E}} \alpha_Q \rightarrow \alpha_{PQ}$. Case $L \equiv \lambda x.P$ and $\Gamma_P \vdash_{\mathbb{T}/\mathcal{E}_P} P : \alpha_P$. Then $\Gamma_P - \{x:\alpha_x\} \vdash_{\mathbb{T}/\mathcal{E}} \lambda x.P : \alpha_x \rightarrow \alpha_P = \alpha_{\lambda x.P}$.

(ii) (\Leftarrow) By (i), Lemma 7A.20 and weakening, Proposition 7A.4.
(\Rightarrow) By Remark 7B.6 it is enough to define a morphism $h^\natural : \mathbb{T}^{\mathbf{c}_M} \rightarrow \mathcal{A}$ such that $h^\natural(\alpha_M) = a$ and $B = C \in \mathcal{E}_M \Rightarrow h^\natural(B) = h^\natural(C)$, for all $B, C \in \mathbb{T}^{\mathbf{c}_M}$.

Take a deduction \mathcal{D} of $\Gamma \vdash_{\mathcal{A}} M : a$. Note that in definition 9B.2 for every $L \subseteq M$ a type derivation \mathcal{D}_L is constructed in which to each variable x occurring in L we assign a type α_x and to each subterm occurrence of L , that is not a variable, a distinct type variable. Moreover, as in Example 9B.1 \mathcal{D}_M and \mathcal{D} have the same shape corresponding to the structure of M . Now we define h^\natural .

$$h^\natural(\alpha_L) = b_L, \quad \text{for all } \alpha_L \text{ assigned to the subterm occurrences } L \text{ of } M, \\ \text{where } b_L \text{ is assigned in } \mathcal{D} \text{ to the corresponding } L.$$

By induction on the subterm occurrences of M it easily follows, using Proposition 9A.3 and Remark 9A.4, that all equations in \mathcal{E}_M are preserved. Moreover, by construction, $h^\natural(\alpha_M) = a$. Obviously we have $h(\Gamma_M) \subseteq \Gamma$, since Γ_M contains only assumptions for variables occurring in M .

(iii) By (ii), knowing that $\Gamma_M = \emptyset$ for $M \in \Lambda^\varnothing$. ■

9B.4. REMARK. \mathcal{E}_M , Γ_M and a_M are the generalization of the notion of principal type and basis scheme for simple types, see Hindley [1969], 1997. In typing a term in the simple type assignment system we require that \mathcal{E}_M can be solved in an initial type algebra \mathbb{T} . In this case \mathcal{E}_M simply defines the substitution which applied to Γ_M and a_M gives the principal typing of M in the usual sense. Theorem 9B.3 is then a generalization of the Principal Type Theorem.

If we want to consider only invertible type algebras, then by Lemma 8C.27 we can take $\mathcal{B}_M = \mathbb{T}^{\mathbf{c}_M}/\mathcal{E}_M^{\text{inv}}$ as the initial type algebra for M . Let $\mathcal{R}_M = \mathcal{E}_M^{\text{inv}}$. Note that, by Lemma 8C.26, \mathcal{R}_M is a proper sr and we have $\mathcal{B}_M = \mathbb{T}^{\mathbf{c}_M}/\mathcal{R}_M$.

9B.5. COROLLARY. For every $M \in \Lambda$ there exists a so-called invertible principal triple Γ_M , \mathcal{B}_M , b_M such that \mathcal{B}_M is invertible and the following holds.

- (i) $\Gamma_M \vdash_{\mathcal{B}_M} M : b_M$.
- (ii) $\Gamma \vdash_{\mathcal{B}} M : a$, with \mathcal{B} invertible $\Leftrightarrow \exists h : \mathcal{B}_M \rightarrow \mathcal{B}. [h(\Gamma_M) \subseteq \Gamma \& h(b_M) = a]$.

PROOF. Given M , consider its principal type $\Gamma'_M, \mathcal{A}_M, a_M$ and let $\mathcal{B}_M = \mathcal{A}_M^{\text{inv}}$. Let $k : \mathcal{A}_M \rightarrow \mathcal{B}_M$ be the canonical morphism and take $\Gamma_M = k(\Gamma'_M)$, $b_M = k(a_M)$. Then (i) holds by the Theorem and Lemma 7A.20. Property (ii) follows by (ii) of the Theorem and Proposition 8C.27. ■

Let \mathbf{a}_M be the subset of \mathbf{c}_M containing all $\alpha \in \mathbf{c}_M$ such that α does not occur in the right hand of any equation in \mathcal{R}_M . Then we have that $\mathcal{B}_M = \mathbb{T}^{\mathbf{a}_M}[\mathcal{R}_M]$.

The typability problems

Theorem 9B.3 provides the abstract answer to the question whether a term has a type within a given type algebra \mathcal{A} . The first typability question

“Given an untyped $M \in \Lambda^\varnothing$, does there exist \mathcal{A} and $a \in \mathcal{A}$ such that $\vdash_{\lambda\mathcal{A}} M : a$?”,

is trivially checked: this is always the case. Indeed, take $\mathcal{A} = \mathbb{T}^{\{\alpha\}}/\{\alpha = \alpha \rightarrow \alpha\}$ and $a = [\alpha]$; then one has $\vdash_{\lambda\mathcal{A}} M : a$, see Exercise 7G.12. However, this problem is no longer trivial when term constants are considered, see the discussion in Section 11C.

The decidability of the second question, given $M \in \Lambda^\varnothing$ and type algebra \mathcal{A}

“Does there exist an $a \in \mathcal{A}$ such that $\vdash_{\lambda\mathcal{A}} M : a$?”,

only makes sense if \mathcal{A} is ‘finitely presented’, i.e. of the form $\mathbb{T}^{\mathbb{A}}/\mathcal{E}$ with \mathbb{A} and \mathcal{E} finite. Then typability is decidable.

9B.6. THEOREM. *Let \mathcal{E} be a set of equations over $\mathbb{T} = \mathbb{T}^{\mathbb{A}}$, with \mathbb{A} and \mathcal{E} finite. It is decidable whether an $M \in \Lambda^{\varnothing}$ has a type in \mathbb{T}/\mathcal{E} .*

PROOF. By Theorem 9B.3(iii) this boils down to checking whether \mathcal{E} justifies \mathcal{E}_M , which is decidable by Theorem 8C.29. ■

A related but stronger version of this problem will be discussed in the next subsection in the case that \mathcal{E} is an sr \mathcal{R} .

Type checking

We work in the type algebra $\mathbb{T}[\mathcal{R}]$. Writing $x_1:A_1, \dots, x_n:A_n \vdash_{\lambda\mathcal{R}} M : A$ we mean $x_1:[A_1], \dots, x_n:[A_n] \vdash_{\lambda\mathcal{R}} M : [A]_{\mathcal{R}}$, according to Notation 7A.18, identifying $\mathbb{T}(\vec{X})/=_\mathcal{R}$ with $\mathbb{T}(\vec{X})$.

9B.7. THEOREM. *Given $M \in \Lambda$, a proper sr \mathcal{R} over $\mathbb{T}^{\mathbb{A}}$, with \mathbb{A} finite, a type basis Γ and a type $A \in \mathbb{T}$. Then it is decidable whether $\Gamma \vdash_{\lambda\mathcal{R}} M : A$.*

PROOF. W.l.o.g. we can assume $\text{dom}(\Gamma) = \text{FV}(M)$. By Theorem 9B.3(ii) one has $\Gamma \vdash_{\lambda\mathcal{R}} M : A$ iff there is a morphism $h : \mathcal{A}_M \rightarrow \mathbb{T}[\mathcal{R}]$ such that $h(a_M) = A$ and $h(\Gamma_M) \subseteq \Gamma$. This is decidable by Corollary 8C.28. ■

9B.8. REMARK. The set

$$\{\langle h_k(\Gamma_M), h_k(a_M) \rangle \mid \text{there is } h' \text{ which agrees with } \mathcal{S}_i\}$$

can be seen as a finite set of principal pairs for M in \mathcal{R} , in the sense of Theorem 2C.14, from which we can generate all possible other typings of M in \mathcal{R} via morphisms.

Finding inhabitants

We will now discuss the inhabitation problem for $(\lambda\mathcal{R})$. Following a suggestion of R. Statman we will reduce the inhabitation problem for recursive types to the one for simple types discussed in Section 2D. In the following we will consider the syntactic version of the type assignment system for $(\lambda\mathcal{R})$ (see Proposition 7A.19), where the type equivalence relation is given by $=_{\mathcal{R}}$.

9B.9. DEFINITION. Let $\mathcal{R}(\vec{X}) = \{X_i = A_i(\vec{X}) \mid 1 \leq i \leq n\}$ be an sr over \mathbb{T} and $A \in \mathbb{T}[\vec{X}]$. Define the type $((\mathcal{R})) \rightarrow A \in \mathbb{T}[\vec{X}]$ and the environment $\Gamma_{\mathcal{R}}$ as:

- (i) $((\mathcal{R})) \rightarrow A \triangleq (X_1 \rightarrow A_1) \rightarrow (A_1 \rightarrow X_2) \rightarrow \dots \rightarrow A$.
- (ii) $\Gamma_{\mathcal{R}} \triangleq \{c_i : X_i \rightarrow A_i \mid 1 \leq i \leq n\} \cup \{c'_i : A_i \rightarrow X_i \mid 1 \leq i \leq n\}$.

We will show that A is inhabited in $\lambda\mathcal{R}$ if and only if $((\mathcal{R})) \rightarrow A$ is inhabited in λ_{\rightarrow} .

9B.10. LEMMA. *Let $\mathcal{R}(\vec{X})$ be an sr over \mathbb{T} and $A, B \in \mathbb{T}[\vec{X}]$. If $\mathcal{R} \vdash A = B$ then there are terms M and M' such that $\Gamma_{\mathcal{R}} \vdash_{\lambda_{\rightarrow}} M : A \rightarrow B$ and $\Gamma_{\mathcal{R}} \vdash_{\lambda_{\rightarrow}} M' : B \rightarrow A$.*

PROOF. By induction on the proof of $\mathcal{R} \vdash A = B$. If $A = B$ has been obtained by rule (axiom) (i.e. $A = X_i$ and $B = A_i$ where $X_i = A_i \in \mathcal{R}$) then take $M = c_i$ and $M' = c'_i$. If $A = B$ has been obtained by rule (refl) the proof is trivial.

As for the induction step the most interesting case is when $A = B$ has been obtained by rule (\rightarrow -cong). In this case we have $A = A_1 \rightarrow A_2$, $B = B_1 \rightarrow B_2$, $\mathcal{R} \vdash A_1 = B_1$ and

$\mathcal{R} \vdash A_2 = B_2$. By induction hypothesis there are M'_1 and M_2 such that $\Gamma_{\mathcal{R}} \vdash_{\lambda\rightarrow} M'_1 : B_1 \rightarrow A_1$ and $\Gamma_{\mathcal{R}} \vdash_{\lambda\rightarrow} M_2 : A_2 \rightarrow B_2$. Take $M = \lambda x. \lambda y. M_2(x(M'_1 y))$ and note that $\Gamma_{\mathcal{R}} \vdash_{\lambda\rightarrow} M : (A_1 \rightarrow A_2) \rightarrow B_1 \rightarrow B_2$ (assume $x : A_1 \rightarrow A_2$ and $y : B_1$). The term M' can be defined in a similar way.

9B.11. LEMMA. *Let A be a type in $\mathbb{T}[\vec{X}]$ and Γ a type environment over $\mathbb{T}[\vec{X}]$. Then*

$$\exists M. \Gamma \vdash_{\lambda\mathcal{R}} M : A \Leftrightarrow \exists M'. \Gamma, \Gamma_{\mathcal{R}} \vdash_{\lambda\rightarrow} M' : A.$$

PROOF. (\Rightarrow) By induction on deductions in $(\lambda\mathcal{R})$ using Lemma 9B.10.

(\Leftarrow) Take a deduction \mathcal{D} of $\Gamma, \Gamma_{\mathcal{R}} \vdash_{\lambda\rightarrow} M' : A$. To build a deduction of $\Gamma \vdash_{\lambda\mathcal{R}} M : A$ for some term M we only need to get rid of the constants c_i and c'_i . The proof is by induction on the derivation of $\Gamma, \Gamma_{\mathcal{R}} \vdash_{\lambda\rightarrow} M' : A$.

If $M' = x \in \text{dom}(\Gamma)$, then we are done trivially. If $M' = c_i$ then since $X_i =_{\mathcal{R}} A_i$ we have $X_i \rightarrow X_i =_{\mathcal{R}} X_i \rightarrow A_i$. Then we can take $M = \mathbf{l}$. We have $\Gamma \vdash_{\lambda\mathcal{R}} \mathbf{l} : X_i \rightarrow A_i$, using rule (equal). If $M' = c'_i$ the argument is similar.

All induction steps are trivial. ■

Lemma 9B.11 can be generalized in the following way.

9B.12. PROPOSITION. *Given an sr $\mathcal{R}(\vec{X})$ over \mathbb{T} and a type A in $\mathbb{T}[\vec{X}]$*

A is inhabited in $\lambda\mathcal{R}$ iff $((\mathcal{R})) \rightarrow A$ is inhabited in $(\lambda\rightarrow)$.

9B.13. REMARK. The reduction of the inhabitation problem for recursive types to that for simple types is immediate if we consider the system $(\lambda\mathcal{R}^{\text{Ch}_0})$ with explicit constants representing type conversion via folding and unfolding. Here the c_i play the roles of unfold_{X_i} and the c'_i of fold_{X_i} .

Moreover, using Lemma 9B.10 it can easily be shown that any typing in $(\lambda\mathcal{R})$ can be represented by a term of $(\lambda\mathcal{R}^{\text{Ch}_0})$.

Finding μ -types and their inhabitants

Concerning the problems of type reconstruction, typability, type checking and inhabitation in $(\lambda\mu)$ and $(\lambda\mu^*)$, two of these are trivial. Write $\lambda\mu^{(*)}$ to mean that both the $\lambda\mu$ and the $\lambda\mu^*$ versions hold. Typability in $\lambda\mu^{(*)}$, i.e.

$$\exists A \in \mathbb{T}_{\mu}. \vdash_{\lambda\mu^{(*)}} M : A,$$

always holds, taking $A = \mu\alpha. \alpha \rightarrow \alpha$, see Exercise 7G.12. Inhabitation in $\lambda\mu^{(*)}$,

$$\exists M \in \Lambda. \vdash_{\lambda\mu^{(*)}} M : A,$$

always holds, taking $M \equiv \Omega$, see Example 7A.3(ii), using $\mu\alpha. \alpha \rightarrow A$.

The remaining problems, type reconstruction and type checking are first considered for $\lambda\mu^*$ using μ -terms modulo strong equality (as the situation there is somewhat simpler than the one for $\lambda\mu$). Note that both $\lambda\mu^*$ and $\lambda\mu$ are invertible, so we can consider only invertible type algebras.

The notions of principal type and principal pair, see Definitions 2C.13 and 9B.5, can be generalized in a straightforward way to μ -types considered modulo strong equivalence. On the one hand in \mathbb{T}_{μ} all possible recursive types are present and so in considering problems about assigning types to λ -terms there is no parameter \mathcal{R} describing the recursive types available in the system. On the other hand the solutions are not unique

up to weak equivalence $=_\mu$ but only up to $=_{\mu^*}$. Therefore, only for the system $(\lambda\mu^*)$ we know how to state the Principal Type Theorem in a straightforward way.

9B.14. DEFINITION. Recall the definitions of \mathcal{E}_M , a_M , and Γ_M in 9B.2. Let \vec{X} denote the indeterminates of the sr \mathcal{E}_M and $\vec{S} = \langle S_1, \dots, S_n \rangle$ be a solution of \mathcal{E}_M in \mathbb{T}_μ (typically that given by Theorem 8A.1). We define

- (i) $\mathbf{a}_M^\mu \triangleq a_M[\vec{X} := \vec{S}]$;
- (ii) $\Gamma_M^\mu \triangleq \Gamma_M[\vec{X} := \vec{S}]$.

9B.15. THEOREM. (i) $\Gamma_M^\mu \vdash_{\lambda\mu} M : a_M^\mu$.

(ii) Suppose $\Gamma \vdash_{\lambda\mu^*} M : A$. Then there is a substitution $s : \mathbf{a}_M \rightarrow \mathbb{T}_\mu$ such that $A =_\mu^* s(a_M^\mu)$ and $\Gamma \supseteq s(\Gamma_M^\mu)$ (modulo $=_\mu^*$).

PROOF. (i) By Theorem 9B.3(ii) and Corollary 8A.4(ii).

(ii) By Remark 7E.33, since μ -types modulo $=_\mu^*$ are just notations for regular trees. ■

9B.16. COROLLARY. $\Gamma_M^\mu \vdash_{\lambda\mu^*} M : a_M^\mu$.

PROOF. By Theorem 9B.15(i) and Lemma 7A.20(i), taking $h(A) = (A)_\mu^*$. ■

Using a similar argument one can show the following result.

9B.17. THEOREM. For $M \in \Lambda$, basis Γ and $A \in \mathbb{T}_\mu$ it is decidable whether $\Gamma \vdash_{\lambda\mu^*} M : A$.

PROOF. See Exercise 9D.6. ■

As for weak equivalence Theorem 9B.3 implies that the set of possible typings in $\lambda\mu$ of a term M , i.e. the set of all Γ, A such that $\Gamma' \vdash_{\lambda\mu} M : A$ for all $\Gamma' \supseteq \Gamma$, is after coding recursively enumerable. Moreover, by Lemma 7A.20 and Theorem 8A.1, given a typing deduction in $\lambda\mathcal{R}$ (for any sr \mathcal{R}) we can easily build one in $\lambda\mu$ (via the construction in Theorem 8A.1). It is not clear, however, how a principal type can be constructed in $\lambda\mu$, owing to the many incomparable solutions that the same sr can have in \mathbb{T}_μ .

9C. Strong normalization

As shown by the examples in Section 7A type systems with recursive types do not have, in general, the strong normalization property. This is easy: in the presence of a type $A = A \rightarrow A$ 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 holds. The strong normalization theorem in this section comes from [Mendler \[1987\]](#), 1991. We will prove in detail the strong normalization theorem for an assignment system with μ -types. Exploiting the fact, shown in 8A, that any sr can be solved in \mathbb{T}_μ , this result can be extended 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 $\mathbf{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$, or even to unsolvable ones, like $\Omega \equiv (\lambda x.xx)(\lambda x.xx)$. 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}_μ^+ are strongly normalizable. The set \mathbb{T}_μ^+ 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.

9C.1. DEFINITION. (i) The notion of *positive* and *negative* occurrence of a subtype in a type of \mathbb{T}_μ 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 $C \rightarrow B$ and *negative* (*positive*) in $B \rightarrow C$.

(3) If A is *positive* (*negative*) in B then A is *positive* (*negative*) in $\mu t.B$.
(ii) We say that a type variable α is *positive* (*negative*) in A if all free occurrences of α in A are *positive* (*negative*). It is not required that $\alpha \in \text{FV}(A)$, but we do not speak about occurrences of bound variables being positive or negative.

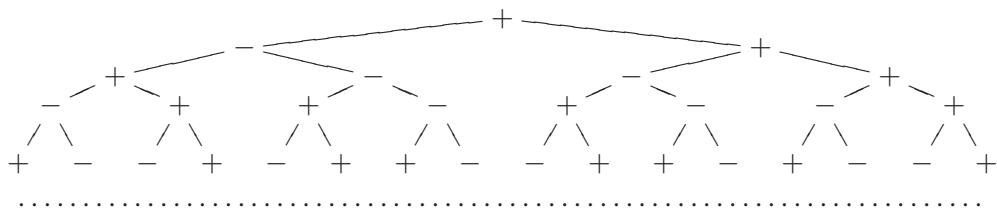


FIGURE 27. The universal binary tree and the \pm nodes

Note that A is *positive* (*negative*) in B if A occurs in B only on the left hand side of an even (odd) number of ' \rightarrow '. Let this number be the *level* of the occurrence of A in B . For instance δ is *positive* in $((\mu\alpha.\alpha \rightarrow \delta) \rightarrow \beta) \rightarrow \gamma$, since the occurrence of δ is at level 2, but not in $((\mu\alpha.\delta \rightarrow \alpha) \rightarrow \beta) \rightarrow \gamma$, since now it is at level 3.

Finite or infinite trees can be considered as subtrees of the universal binary tree. The positive and negative subtypes in a type considered as tree are displayed in Fig. 27. It is interesting to note that the sequences of the signs $+, -$ on one horizontal level, starting all the way to the left with a $+$, form longer and longer initial segments of the Morse-Thue sequence, see Allouche and Shallit [2003].

9C.2. DEFINITION. (i) A type $A \in \mathbb{T}_\mu$ is called *positive* if for every $\mu\alpha.B \subseteq A$ all occurrences of α in B are *positive*.

- (ii) $\mathbb{T}_\mu^+ \triangleq \{A \in \mathbb{T}_\mu \mid A \text{ is positive}\}.$
- (iii) Let $\lambda\mu^+$ be the type assignment system defined as $\lambda\mu$, but restricting the set of types to \mathbb{T}_μ^+ .

9C.3. REMARK. Note that $A, B \in \mathbb{T}_\mu^+ \Leftrightarrow (A \rightarrow B) \in \mathbb{T}_\mu^+$.

9C.4. EXAMPLES. (i) $\mu\beta.\alpha \rightarrow \beta \in \mathbb{T}_\mu^+$.

- (ii) $\mu\beta.\beta \rightarrow \alpha \notin \mathbb{T}_\mu^+$.
- (iii) $\mu\beta.\beta \rightarrow \beta \notin \mathbb{T}_\mu^+$.
- (iv) $((\mu\beta.\alpha \rightarrow \beta) \rightarrow \gamma) \in \mathbb{T}_\mu^+$.
- (v) $(\mu\beta.(\alpha \rightarrow \beta) \rightarrow \gamma) \notin \mathbb{T}_\mu^+$.

Obviously any deduction in $\lambda\mu^+$ is also a deduction in $\lambda\mu$. So any type assignment provable in $\lambda\mu^+$ is also provable in $\lambda\mu$.

Positive recursive types are often called *inductive*, since it is possible to define for them an induction principle, see [Mendler \[1991\]](#). In that paper the result is proved in fact for a stronger system, defined as an extension of second order typed lambda calculus with provable positive recursive types and induction and coinduction operators for each type.

The strong normalization proof for terms typable in $\lambda\mu^+$ is based on the notion of saturated set introduced in [Girard \[1971\]](#), which is in turn based on Tait's method presented in Section 2B provided with an impredicative twist. We will prove it for the Curry version of $\lambda\mu$, but the proof extends to the systems with explicit typing, see Definition 7A.21. For an arithmetical proof of the mentioned SN result, see [David and Nour \[2007\]](#).

Let SN be the set of strongly normalizing type free λ -terms.

9C.5. DEFINITION. (i) A subset $\mathcal{X} \subseteq \text{SN}$ is *saturated* if the following conditions hold.

(1) \mathcal{X} is *closed under reduction*, i.e. for all $M, N \in \Lambda$

$$M \in \mathcal{X} \ \& \ M \rightarrow_{\beta} N \Rightarrow N \in \mathcal{X}.$$

(2) For all variables x one has

$$\forall n \geq 0 \forall R_1, \dots, R_n \in \text{SN}. xR_1 \cdots R_n \in \mathcal{X}.$$

(3) For all $Q \in \text{SN}$ one has

$$P[x := Q]R_1 \cdots R_n \in \mathcal{X} \implies (\lambda x.P)QR_1 \cdots R_n \in \mathcal{X}$$

(ii) Let SAT be the set of all saturated subsets of SN .

9C.6. PROPOSITION. SAT is a complete lattice, see Definition 10A.4(vi) under the operations of set inclusion, with $\text{sup}\{\mathcal{X}, \mathcal{Y}\} = \mathcal{X} \cup \mathcal{Y}$ and $\text{inf}\{\mathcal{X}, \mathcal{Y}\} = \mathcal{X} \cap \mathcal{Y}$.

The top element of SAT is SN while its bottom element $\perp_{\text{SAT}} = \bigcap_{\mathcal{X} \in \text{SAT}} \mathcal{X}$ is the least set containing all terms of the shape $xR_1 \cdots R_n$ ($R_i \in \text{SN}$) and that satisfies (3) of Definition 9C.5(i). We recall that a function $f : \text{SAT} \rightarrow \text{SAT}$ is *monotonic* if $X \subseteq Y$ (where $X, Y \in \text{SAT}$) implies $f(X) \subseteq f(Y)$ and *anti-monotonic* if $X \subseteq Y$ implies $f(Y) \subseteq f(X)$. If $f : \text{SAT} \rightarrow \text{SAT}$ is a monotonic function, then

$$\text{fix}(f) \triangleq \bigcap\{x \mid f(x) \subseteq x\}$$

is the least fixed point of f .

9C.7. DEFINITION. The operation $\Rightarrow : \text{SAT} \times \text{SAT} \rightarrow \text{SAT}$ is defined by

$$(\mathcal{X} \Rightarrow \mathcal{Y}) \triangleq \{M \mid \forall N \in \mathcal{X}. (MN) \in \mathcal{Y}\}$$

The proof of the following proposition is standard.

9C.8. PROPOSITION. \Rightarrow is well defined, i.e. for all $\mathcal{X}, \mathcal{Y} \in \text{SAT}$ $\mathcal{X} \Rightarrow \mathcal{Y} \in \text{SAT}$. 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}_{μ}^{+} as elements of SAT and of terms as elements of Λ . This is inspired by standard semantical notions, see Chapter 10. Let a *type environment* be a function $\tau : \mathbb{A} \rightarrow \text{SAT}$.

9C.9. DEFINITION. The *interpretation of a type* $A \in \mathbb{T}_\mu^+$ under a type environment τ , notation $\llbracket A \rrbracket_\tau$, is defined by

$$\begin{aligned}\llbracket \alpha \rrbracket_\tau &\triangleq \tau(\alpha); \\ \llbracket A \rightarrow B \rrbracket_\tau &\triangleq \llbracket A \rrbracket_\tau \Rightarrow \llbracket B \rrbracket_\tau; \\ \llbracket \mu\alpha.A \rrbracket_\tau &\triangleq \text{fix}(\lambda\mathcal{X}:\text{SAT}. \llbracket A \rrbracket_{\tau[\alpha:=\mathcal{X}]})\end{aligned}$$

9C.10. LEMMA. (i) If α is positive (negative) in $A \in \mathbb{T}_\mu^+$ then

$$\lambda\mathcal{X}:\text{SAT}. \llbracket A \rrbracket_{\tau[\alpha:=\mathcal{X}]}$$

is a monotonic (anti-monotonic) function over the complete lattice SAT .

(ii) $\llbracket - \rrbracket_\tau$ is well defined on \mathbb{T}_μ^+ , i.e. $\llbracket A \rrbracket_\tau \in \text{SAT}$ for all types $A \in \mathbb{T}_\mu^+$ and all type environments τ .

(iii) For circular types A one has $\llbracket A \rrbracket^\tau = \perp_{\text{SAT}}$.

PROOF. (i), (ii) by simultaneous induction on A .

(iii) Notice that $\llbracket \mu\alpha.\alpha \rrbracket^\tau = \perp_{\text{SAT}}$ and $\llbracket \mu\alpha.B \rrbracket^\tau = \llbracket B \rrbracket^\tau$, if $\alpha \notin B$. Then for a circular $A \equiv \mu\alpha_1 \cdots \alpha_n.\alpha_i$ one has $\llbracket A \rrbracket^\tau = \llbracket \mu\alpha_i.\alpha_i \rrbracket^\tau = \perp_{\text{SAT}}$. ■

9C.11. LEMMA. (i) $\llbracket A[\alpha := B] \rrbracket_\tau = \llbracket A \rrbracket_{\tau[\alpha := \llbracket B \rrbracket_\tau]}$

(ii) $\llbracket \mu\alpha.A \rrbracket_\tau = \llbracket A[\alpha := \mu\alpha.A] \rrbracket_\tau$. ■

PROOF. (i) By structural induction on A .

(ii) By definition $\llbracket \mu\alpha.A \rrbracket_\tau$ is a fixed point of $\lambda\mathcal{X}:\text{SAT}. \llbracket A \rrbracket_{\tau[t:=\mathcal{X}]}$. Therefore we have $\llbracket \mu\alpha.A \rrbracket_\tau = \llbracket A \rrbracket_{\tau[t:=\llbracket \mu\alpha.A \rrbracket_\tau]}$, which is $\llbracket A[\alpha := \mu\alpha.A] \rrbracket_\tau$, by (i). ■

By Lemma 9C.11 type interpretation is preserved under weak equivalence.

9C.12. LEMMA. Let $A, B \in \mathbb{T}_\mu^+$. If $A =_\mu B$, then $\llbracket A \rrbracket_\tau = \llbracket B \rrbracket_\tau$.

We define now an interpretation of terms in Λ . This is in fact a *term model* interpretation. Let a *term environment* be a function from the set of term variables V to elements of Λ .

9C.13. DEFINITION. Let $\rho : V \rightarrow \Lambda$ be a term environment. The *evaluation* of a term M under ρ is defined by

$$\llbracket M \rrbracket_\rho \triangleq M[x_1 := \rho(x_1), \dots, x_n := \rho(x_n)],$$

where x_1, \dots, x_n ($n \geq 0$) are the variables free in M .

We define now the notion of satisfiability of a statement $\Gamma \vdash_{\lambda\mu+} M : A$ with respect to the interpretations of types and terms just given.

9C.14. DEFINITION. Let τ, ρ be a type and a term environment, respectively. Define

- (i) $\tau, \rho \models M : A \iff \llbracket M \rrbracket_\rho \in \llbracket A \rrbracket_\tau$.
- (ii) $\tau, \rho \models \Gamma \iff \tau, \rho \models x : A$, for all $x : A \in \Gamma$.
- (iii) $\Gamma \models M : A \iff \tau, \rho \models \Gamma \Rightarrow \tau, \rho \models M : A$, for all τ, ρ .

9C.15. 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 9C.12 for rule (equal). ■

9C.16. THEOREM (Mendler). $\Gamma \vdash_{\lambda\mu+} M : A \implies M \text{ is SN}$.

PROOF. By Lemma 9C.15 one has $\Gamma \models M : A$. Define ρ_0 by $\rho_0(x) = x$ for all $x \in V$. Trivially, for any type environment τ one has $\tau, \rho_0 \models \Gamma$. Hence $\tau, \rho_0 \models M : A$. Therefore $\llbracket M \rrbracket_{\rho_0} = M \in \llbracket A \rrbracket_{\tau} \subseteq SN$. ■

9C.17. REMARK. The converse of this result does not hold. For example $\omega \equiv \lambda x. xx$ is SN, but not typable in $\lambda\mu^+$, see Exercise 9D.9.

We can reformulate Theorem 9C.16 as follows. Define the set of terms typable from a set of types as follows.

9C.18. DEFINITION. Let $\mathcal{X} \subseteq \mathbb{T}_{\mu}$. Then

$$\text{Typable}(\mathcal{X}) = \{M \in \Lambda \mid M \text{ can be typed using only types in } \mathcal{X}\}.$$

By ‘using only types in \mathcal{X} ’ we mean that all types (including those in the basis) in some derivation of $\Gamma \vdash M : A$ are elements of \mathcal{X} .

9C.19. DEFINITION. Let $\mathcal{SC}^+(A)$ be the least set of types containing $\mathcal{SC}^s(A)$, see Definition 8B.34, that is closed under \rightarrow .

- 9C.20. LEMMA. (i) If $A \in \mathbb{T}_{\mu}^+$ and $B \in \mathcal{SC}^s(A)$, then $B \in \mathbb{T}_{\mu}^+$.
(ii) If $A \in \mathbb{T}_{\mu}^+$ and $B \in \mathcal{SC}^s p(A)$, then $B \in \mathbb{T}_{\mu}^+$.

PROOF. (i) By induction on the definition of $\mathcal{SC}^s(A)$.

(ii) By (i) and Remark 9C.3. ■

The following is a consequence of Theorem 9C.16.

9C.21. COROLLARY. If $B \in \mathbb{T}_{\mu}^+$, then $\text{Typable}(\mathcal{SC}^+(B)) \subseteq SN$.

PROOF. Immediate from Theorem 9C.16 and Lemma 9C.20. ■

Note that this Corollary also implies Theorem 9C.16. Indeed, assume $G \vdash_{\lambda\mu^+} M : A$. Let $\mathcal{X} = \{A_1, \dots, A_n\} \subseteq \mathbb{T}_{\mu}^+$ be the set of types used in this deduction. Then we can apply the Corollary to $B = A_1 \rightarrow A_2 \rightarrow \dots \rightarrow A_n \in \mathbb{T}_{\mu}^+$.

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 B which do not belong to \mathbb{T}_{μ}^+ allow to type a non-normalizable term, using only types of $\mathcal{SC}^+(B)$. We see this in the next theorem. The construction is due to Mendler [1991]. As a warm-up, do exercise 9D.11.

Let \vec{A} denote a sequence of types A_1, \dots, A_n ($n \geq 0$) and $\vec{A} \rightarrow B$ denote the type $A_1 \rightarrow \dots \rightarrow A_n \rightarrow B$. Moreover if \vec{M} is a sequence of terms M_1, \dots, M_n then $\vec{M} : \vec{A}$ denotes the set of statements $M_1 : A_1, \dots, M_n : A_n$.

9C.22. THEOREM (Mendler). Let $B \in \mathbb{T}_{\mu} \setminus \mathbb{T}_{\mu}^+$. Then there is a term N without normal form, typable in $\mathcal{SC}^+(B)$. In formula (where N denotes the set of normalizable terms)

$$B \notin \mathbb{T}_{\mu}^+ \Rightarrow \text{Typable}(\mathcal{SC}^+(B)) \not\subseteq \mathsf{N}.$$

PROOF. By assumption there is a subtype $\mu\alpha.T$ of B such that α has a negative occurrence in T . Now 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

$$\begin{aligned} \mu\alpha.T &=_{\mu} Q_{2n+1}; \\ Q_i &\equiv \vec{A}_i \rightarrow Q_{i-1} \rightarrow B_i, \quad \text{with } 1 \leq i \leq 2n+1; \\ Q_0 &\equiv \vec{A}_0 \rightarrow \mu\alpha.T, \end{aligned}$$

where Q_{2n+1} is the head reduced form of $\mu\alpha.T$ (see Definition 7D.34). Clearly all these types are elements of $\mathcal{SC}^+(B)$. See Example 9C.23.

Case $n = 0$ is an exercise.

Case $n > 0$. Define terms $N_i : Q_i$ with $0 \leq i \leq 2n + 1$ in the following order $N_1, N_3, \dots, N_{2n+1}, N_0, N_2, \dots, N_{2n}$.

$$\begin{aligned} N_1 &\triangleq \lambda \vec{x}_1. \lambda y_0. f^{2n+1,1}(y_0 \vec{z}_0 \vec{z}_{2n+1} y_{2n}); \\ N_{2i+1} &\triangleq \lambda \vec{x}_{2i+1} \lambda y_{2i}. f^{2i,2i+1}(y_{2i} \vec{z}_{2i} N_{2i-1}), \quad \text{for } 0 \leq i \leq n, \\ N_0 &\triangleq \lambda \vec{x}_0. N_{2n+1}; \\ N_{2i} &\triangleq \lambda \vec{x}_{2i} \lambda y_{2i-1}. f^{2i-1,2i}(y_{2i-1} \vec{z}_{2i-1} N_{2i-2}), \quad \text{for } 1 \leq i \leq n. \end{aligned}$$

Note that the y_i for $0 \leq i < 2n$ do not occur free in any N_j and that y_{2n} occurs free in $N_1, N_3, \dots, N_{2n-1}$, but not in $N_{2n+1}, N_0, N_2, \dots, N_{2n}$. We do this relative to the basis $\{f^{k,h} : B_k \rightarrow B_h\}$, with $1 \leq k, h \leq 2n + 1$. Moreover in these terms we have the intended types $\vec{x}_i : \vec{A}_i$, $\vec{z}_i : \vec{A}_i$, and $y_i : Q_i$.

Now consider the term $N \triangleq (N_{2n+1} \vec{z}_{2n+1} N_{2n})$. It is straightforward to verify that N can be typed in the following basis.

$$\Gamma_0 \triangleq \{f^{k,h} : B_k \rightarrow B_h \mid 1 \leq k, h \leq 2n + 1\} \cup \{\vec{z}_i : \vec{A}_i \mid 0 \leq i \leq 2n + 1\},$$

using only types of $\mathcal{SC}^+(B)$. The term N has the following (unique) reduction which is infinite.

$$\begin{aligned} N &= N_{2n+1} \vec{z}_{2n+1} N_{2n} \\ &\rightarrow_\beta f^{2n,2n+1}(N_{2n} \vec{z}_{2n} N_{2n-1}[y_{2n} := N_{2n}]) \\ &\rightarrow_\beta f^{2n,2n+1}(f^{2n-1,2n}(N_{2n-1}[y_{2n} := N_{2n}] \vec{z}_{2n-1} N_{2n-2})) \\ &\rightarrow_\beta f^{2n,2n+1}(\dots(f^{1,2}(N_1[y_{2n} := N_{2n}] \vec{z}_1 N_0))\dots) \\ &\rightarrow_\beta f^{2n,2n+1}(\dots(f^{1,2}(f^{2n+1,1}(N_0 \vec{z}_0 \vec{z}_{2n+1} N_{2n})))\dots) \\ &\rightarrow_\beta f^{2n,2n+1}(\dots(f^{1,2}(f^{2n+1,1}(N_{2n+1} \vec{z}_{2n+1} N_{2n})))\dots) \\ &= f^{2n,2n+1}(\dots(f^{1,2}(f^{2n+1,1} N))\dots). \blacksquare \end{aligned}$$

Theorem 9C.22 can be immediately extended to λ_μ^{Ch} and $\lambda_\mu^{\text{Ch}_0}$.

9C.23. EXAMPLE. Let $B \triangleq \mu\alpha.T$, with $T \triangleq ((\alpha \rightarrow \beta) \rightarrow \alpha) \rightarrow \beta$. We consider the leftmost negative occurrence of t . [Considering the rightmost occurrence of t will result in a more simple example.] Now

$$B =_\mu ((B \rightarrow \beta) \rightarrow B) \rightarrow \beta.$$

Then we have $n = 1$, $Q_3 = ((B \rightarrow \beta) \rightarrow B) \rightarrow \beta$, $Q_2 = (B \rightarrow \beta) \rightarrow B$, $Q_1 = B \rightarrow \beta$, and $Q_0 = B$. There are no \vec{A}_i , one has $B_1 \equiv B_3 \equiv \beta$, and $B_2 \equiv B$. Moreover,

$$\begin{aligned} \Gamma_0 &= \{f^{12} : \beta \rightarrow B, f^{2,3} : B \rightarrow \beta, f^{3,1} : \beta \rightarrow \beta\}; \\ N_1 &= \lambda y_0. f^{3,1}(y_0 y_2); \\ N_0 = N_3 &= \lambda y_2. f^{2,3}(y_2 N_1) = \lambda y_2. f^{2,3}(y_2 \lambda y_0. f^{3,1}(y_0 y_2)); \\ N_2 &= \lambda y_1. f^{1,2}(y_1 N_0). \end{aligned}$$

Note that $\Gamma_0, y_2: Q_2 \vdash N_1 : Q_1$, $\Gamma_0 \vdash N_i : Q_i$, for $i \in \{0, 2, 3\}$. Then $\Gamma_0 \vdash_{\lambda\mu} (N_0 N_2) : \alpha$. We have the infinite reduction

$$(N_0 N_2) \rightarrow_{\beta} f^{2,3}(N_2 N_1 [y_2 := N_2]) \rightarrow_{\beta} f^{2,3}(f^{1,2}(f^{3,1}(N_0 N_2))) \rightarrow \dots \blacksquare$$

Theorems 9C.16 and 9C.22 show that \mathbb{T}_{μ}^+ is the largest subset $\mathcal{X} \subseteq \mathbb{T}_{\mu}$ such that if a term can be typed using only the types in \mathcal{X} , then M is strongly normalizing (SN).

9C.24. COROLLARY. Let $B \in \mathbb{T}_{\mu}$. Then

- (i) $\text{Typable}(\mathcal{SC}^+(B)) \subseteq \mathsf{N} \Rightarrow B \in \mathbb{T}_{\mu}^+$.
- (ii) $B \in \mathbb{T}_{\mu}^+ \Leftrightarrow \text{Typable}(\mathcal{SC}^+(B)) \subseteq \mathsf{SN}$.
- $\Leftrightarrow \text{Typable}(\mathcal{SC}^+(B)) \subseteq \mathsf{N}$.

PROOF. (i) By Theorem 9C.22.

- (ii) $B \in \mathbb{T}_{\mu}^+ \Rightarrow \text{Typable}(\mathcal{SC}^+(B)) \subseteq \mathsf{SN}$, by Corollary 9C.21,
- $\Rightarrow \text{Typable}(\mathcal{SC}^+(B)) \subseteq \mathsf{N}$
- $\Rightarrow B \in \mathbb{T}_{\mu}^+$, by (i). \blacksquare

9C.25. COROLLARY. Let $B \in \mathbb{T}_{\mu}$. Then

$$\text{Typable}(\mathcal{SC}^+(B)) \subseteq \mathsf{N} \Leftrightarrow \text{Typable}(\mathcal{SC}^+(B)) \subseteq \mathsf{SN}.$$

This reminds us of the old question whether $M \in \mathsf{N} \Rightarrow M \in \mathsf{SN}$. This is not the case. $\mathbf{K}\Omega$ has a nf, but is not SN. The reason is that there is the subterm Ω 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 $M_0 =_{\beta} Y(\lambda xy.y)$.

The strong normalization theorem holds also for the typed systems $\lambda_{\mu}^{\text{Ch}}$ and $\lambda_{\mu}^{\text{Ch}_0}$ (which is indeed weaker than $\lambda_{\mu}^{\text{Ch}}$). In the case of $\lambda_{\mu}^{\text{Ch}_0}$ it is easy to prove that the reduction rules $(R_{\mathcal{E}}^{\text{uf}})$ and $(R_{\mathcal{E}}^{\text{fu}})$ of Definition 7A.22 cannot cause infinite reduction sequences. On the other hand there does not seem to be a natural definition of positive types for $\vdash_{\lambda\mu^*}$.

Strong normalization for simultaneous recursion

For elements of $\mathbb{T}[\mathcal{R}]$ we define a notion similar to that of a positive occurrence.

9C.26. DEFINITION. We say that an sr \mathcal{R} is *inductive* if the following holds: if $X =_{\mathcal{R}} C$ for an indeterminate X , then X has only positive occurrences in C .

9C.27. EXAMPLE. (i) Let $\mathcal{R}_0 \triangleq \{X_0 = X_1 \rightarrow X_0, X_1 = X_0 \rightarrow X_1\}$. Then \mathcal{R}_0 is inductive. Note that by unfolding we can get $X_0 =_{\mathcal{R}_0} (X_1 \rightarrow X_1) \rightarrow X_0$ (and so on) but both occurrences of X_0 are positive.

(ii) Let $\mathcal{R}_1 \triangleq \{X_0 = X_1 \rightarrow X_1, X_1 = X_0 \rightarrow X_1\}$. Then \mathcal{R}_1 is not inductive. In fact $X_1 =_{\mathcal{R}_1} (X_1 \rightarrow X_1) \rightarrow X_1$.

The following is a useful property.

9C.28. PROPOSITION. An sr \mathcal{R} is inductive iff the solution $\{S_1, \dots, S_n\}$ of \mathcal{R} in \mathbb{T}_{μ} (see Theorem 8A.1) is such that $S_i \in \mathbb{T}_{\mu}^+$, $1 \leq i \leq n$.

PROOF. A routine check. \blacksquare

By this Proposition it is easily decidable if a given sr is inductive. As a consequence of Theorems 9C.16 and 9C.22 we can characterize those sr which can type only terms that are strongly normalizing.

9C.29. THEOREM. *Let \mathcal{R} be an inductive sr. Then*

$$\Gamma \vdash_{\lambda\mathcal{R}} M : A \Rightarrow M \in \text{SN}.$$

PROOF. Let $\mathcal{R} = \mathcal{R}(X_1, \dots, X_n)$ and $h : \mathbb{T}[\mathcal{R}] \rightarrow \mathbb{T}_\mu^+$ be the type algebra morphism defined by $h(X_i) = S_i$, with $1 \leq i \leq n$, where $S_1, \dots, S_n = \text{Sol}_\mu(\mathcal{R})$ and $h(\alpha) = \alpha$, for all other atomic types α . By Proposition 9C.28 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 7A.20 we get $h(\Gamma) \vdash_{\lambda\mu^+} M : h(A)$. Now Theorem 9C.16 applies. ■

9C.30. 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 and some i we have $\Gamma \vdash_{\lambda\mathcal{R}} N : X_i$.*

PROOF. We have $X_i =_{\mathcal{R}} A_i$ for some i with X_i occurring negative in A_i . The proof is the same as for Theorem 9C.22 where X_i replaces $\mu\alpha.T$ and $=_{\mathcal{R}}$ replaces $=_\mu$. ■

Type Inference

The typability of a term M is decidable also with respect to inductive sr (and with respect to $\lambda\mu^+$). This property follows easily from the following Lemma.

9C.31. LEMMA. (i) *Let $\mathcal{A} = \mathbb{T}/\approx$ and $\mathcal{A}' = \mathbb{T}'/\approx'$ be syntactic type algebras and $h : \mathcal{A} \rightarrow \mathcal{A}'$ 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 $\mathcal{R}(\vec{X})$ and $\mathcal{R}'(\vec{X}')$ be proper sr over \mathbb{T} , \mathbb{T}' respectively. Assume that \mathcal{R} is non-inductive and that there is a morphism $h : \mathbb{T}[\mathcal{R}] \rightarrow \mathbb{T}[\mathcal{R}']$. Then \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 7C.12, it follows by a simple induction on $h(X)$ that $h(X)$ must contain a $Y \in \text{dom}(\mathcal{R}')$ such that $Y =_{\mathcal{R}'} C'$ for some C' with negative occurrence Y in it. ■

9C.32. 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} .

9C.33. THEOREM. *Let $M \in \Lambda$. Then*

$$M \text{ can be inductively typed} \Leftrightarrow \mathcal{E}_M \text{ is inductive.}$$

PROOF. We show

$$M \text{ cannot be inductively typed} \Leftrightarrow \mathcal{E}_M \text{ is not inductive.}$$

(\Rightarrow) Immediate by Lemma 9B.3. (\Leftarrow) Assume \mathcal{E}_M is not inductive and that $\Gamma \vdash_{\lambda\mathcal{R}} M : A$. Then, by Theorem 9B.3 and Lemma 9C.31 \mathcal{R} is not inductive. ■

9C.34. THEOREM. *It is decidable whether an $M \in \Lambda$ can be inductively typed.*

PROOF. By Theorem 9C.33 and Proposition 9C.28. ■

9C.35. EXAMPLE. Take the term $\lambda x.xx$. We have seen in Example 9B.1(i) that $\mathcal{E}_{\lambda x.xx}$ contains an equation $\alpha_1 = \alpha_1 \rightarrow \alpha_2$ and then is not inductive. Then there is no inductive sr which can give a type to $\lambda x.xx$. Compare this with Theorems 9C.29 and 9C.30.

An inductive sr can be solved in \mathbb{T}_μ^+ , by Proposition 9C.28. Hence if a term is typable from an inductive sr, then it is also typable in $\lambda\mu^+$. The proof of the converse follows from Exercise 9D.5.

9C.36. THEOREM. *It is decidable whether a given term M can be given a type in $\lambda\mu^+$, i.e. whether $\Gamma \vdash_{\lambda\mu^+} M : A$ for some type A and basis Γ over \mathbb{T}_μ^+ .*

PROOF. By the above and Theorem 9C.34. ■

9D. Exercises

9D.1. Let $M \triangleq \lambda xy.xy(xy)$. Construct the recursive principal type and type algebra for M .

9D.2. Let $M \triangleq \mathbf{c}_2\mathbf{c}_2$.

- (i) Find the principal recursive type and type algebra for M .
- (ii) Show that $\vdash_{\lambda_\rightarrow} M : (\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha$. This is the principal type in λ_\rightarrow and in an invertible type algebra.
- (iii) Show that $\vdash_{\lambda_\equiv^A} M : \beta$, with $\mathcal{A} = \mathbb{T}^{\{\alpha, \beta\}} / \mathcal{E}$, where $\mathcal{E} = \{(\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha = ((\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha) \rightarrow \beta\}$.
- (iv) Find the morphisms mapping the recursive principal type of M onto respectively $(\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha$ and β , according to Theorem 9B.3.

9D.3. Prove that the system λ_μ^{Cho} defined in 7A.22 satisfies the subject-reduction property. Note that this does not follow from Lemma 9A.8, since type equivalence is not defined by a type algebra.

9D.4. Let $(\lambda\mu^-)$ be the system that assigns types in \mathbb{T}_μ to terms in Λ , in which the equivalence on types is not an \rightarrow -congruence, i.e. the rule $\rightarrow\text{-cong}$ is deleted from Definition 7D.26(i). Show that

$$\begin{aligned} x:(A \rightarrow \mu\alpha.B) &\vdash_{\lambda\mu^-} (\lambda y.xy) : (A \rightarrow B[\alpha := \mu\alpha.B]); \\ x:(A \rightarrow \mu\alpha.B) &\not\vdash_{\lambda\mu^-} x : (A \rightarrow B[\alpha := \mu\alpha.B]). \end{aligned}$$

9D.5. Show that if a term M has a type A in $\lambda\mu$, then for some s.r. \mathcal{R} it also has a type B in $\lambda\mathcal{R}$, with the same tree unfolding, i.e. $(A)^* = (B)^*$.

9D.6. Prove Theorem 9B.17. [Hint. Assume for simplicity that M is closed. By Theorem 9B.16 we have to find a substitution s such that $s(a_M^*) = A$. Use Remark 7E.33.]

9D.7. Design an algorithm to decide directly (i.e. without passing through the translation in \mathbb{T}_μ^+), whether a given s.r. is inductive.

9D.8. Show that the typings for the terms in Example 7A.3, are all principal, except the one for \mathbf{Y} .

9D.9. Prove that $\lambda x.xx$ is not typable in $\lambda\mu^+$. [Hint. Use the Inversion Lemma 9A.3.]

9D.10. Show that if $A \in \mathbb{T}_\mu^+$ and $A =_\mu B$, then $B \in \mathbb{T}_\mu^+$.

9D.11. For $M \in \Lambda$ write $\langle M \rangle \triangleq \lambda y.yM$. Define

$$\begin{aligned} N &\triangleq \lambda x.x\langle x \rangle; \\ L &\triangleq N\langle N \rangle. \end{aligned}$$

- (i) Show that L is not normalizing. [Hint. Note that $\langle P \rangle Q \rightarrow_{\beta} QP$.] Historical note. This is very much like Quine's paradox (QP):

$$(QP) \left\{ \begin{array}{l} \text{"yields falsehood when preceded by its own quotation"} \\ \text{yields falsehood when preceded by its own quotation.} \end{array} \right.$$
 (QP) states that a sentence obtained by the given recipe is false; but following that recipe one obtains (QP) itself.
- (ii) Show that $\vdash_{\lambda\mu} L : \alpha$, using $A = \mu\beta.((\beta \rightarrow \alpha) \rightarrow \alpha) \rightarrow \alpha$.

CHAPTER 10

MODELS

Our purpose in the present Chapter is to build concrete type algebras for the interpretation of recursive types. In Section 10A 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. This idea is due to [Scott \[1975a\]](#).

We shall also describe, in Sections 10B and 10C, 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.

10A. Interpretations of type assignments in λ^A

Before constructing a concrete interpretation of type inference systems of the form λ^A introduced in Definition 7A.2, 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 . In Chapter 11 we will indicate how the results can be extended to the situation in which there are constants, both in the types and terms.

10A.1. DEFINITION. Let $\mathcal{D} = \langle \mathcal{D}, \cdot, [\]_\rho^\mathcal{D} \rangle$ be a λ -model, see Definition 3A.31.

(i) We can turn $\mathcal{P}(\mathcal{D})$ into a type algebra by considering $\mathcal{P}(\mathcal{D}) = \langle \mathcal{P}(\mathcal{D}), \Rightarrow \rangle$, where \Rightarrow is defined in Definition 3A.34 as

$$X \Rightarrow Y \triangleq \{d \in \mathcal{D} \mid d \cdot X \subseteq Y\} \triangleq \{d \in \mathcal{D} \mid \forall x \in X. (d \cdot x) \in Y\}.$$

(ii) An *interpretation* of a type algebra \mathcal{A} in \mathcal{D} is a morphism $h : \mathcal{A} \rightarrow \mathcal{P}(\mathcal{D})$.

10A.2. DEFINITION. Let \mathcal{D} be a λ -model, $\rho \in \text{Env}_{\mathcal{D}}$, \mathcal{A} be a type algebra and $h : \mathcal{A} \rightarrow \mathcal{P}(\mathcal{D})$ be a morphism. Let Γ be a basis.

(i) We say that \mathcal{D}, ρ, h *satisfies* the type assignment statement $M : a$, notation

$$\models_{\mathcal{D}, \rho, h} M : a,$$

if $[\![M]\!]_\rho^\mathcal{D} \in h(a)$.

(ii) \mathcal{D}, ρ, h satisfies Γ , notation

$$\models_{\mathcal{D}, \rho, h} \Gamma,$$

if $\models_{\mathcal{D}, \rho, h} x : a$, for all $(x : a) \in \Gamma$.

(iii) Γ satisfies $(M : a)$ with respect to \mathcal{D}, ρ, h , notation

$$\Gamma \models_{\mathcal{D}, \rho, h} M : a,$$

if $\models_{\mathcal{D}, \rho, h} \Gamma \Rightarrow \models_{\mathcal{D}, \rho, h} M : A$.

(iv) Γ satisfies $(M : a)$ with respect to \mathcal{A}, \mathcal{D} , notation

$$\Gamma \models_{\mathcal{A}, \mathcal{D}} M : a,$$

if $\Gamma \models_{\mathcal{D}, \rho, h} M : a$, for all $\rho \in \text{Env}_{\mathcal{D}}, h : \mathcal{A} \rightarrow \mathcal{P}(\mathcal{D})$.

(v) Finally we write $\Gamma \models_{\mathcal{A}} M : a$ if for all \mathcal{D}

$$\Gamma \models_{\mathcal{A}, \mathcal{D}} M : a.$$

10A.3. PROPOSITION (Soundness).

$$\Gamma \vdash_{\lambda^{\mathbb{A}}} M : a \Rightarrow \Gamma \models_{\mathcal{A}} M : a.$$

PROOF. By induction on the length of proof of the LHS. ■

For well-founded type algebras \mathcal{A} morphisms to $\mathcal{P}(\mathcal{D})$ can be obtained by assigning arbitrary values $\xi(a)$ for the prime elements $a \in \mathcal{A}$. In this way morphisms from $\mathbb{T}^{\mathbb{A}}$ are determined by the choice $\xi(\alpha)$ for $\alpha \in \mathbb{A}$.

Some notions of domain theory

For non well-founded type algebras like \mathbb{T}_μ or $\mathbb{T}[\mathcal{R}]$ the existence of morphisms to $\mathcal{P}(\mathcal{D})$ is less obvious. We present some well-known domain theory to be used in Section 10B in order to show that interpretations do exist, exploiting the domain structure of a well-known class of λ -models.

10A.4. DEFINITION. (i) A *partially ordered set*, also called *poset* is a structure $\mathcal{D} = \langle \mathcal{D}, \sqsubseteq \rangle$ satisfying

$$\begin{aligned} x &\sqsubseteq x; \\ x \sqsubseteq y \& y \sqsubseteq z &\Rightarrow x \sqsubseteq z; \\ x \sqsubseteq y \& y \sqsubseteq x &\Rightarrow x = y. \end{aligned}$$

(ii) For $X \subseteq \mathcal{D}$ we say that $d \in \mathcal{D}$ is an *upperbound* of X , notation $X \sqsubseteq d$, if

$$\forall x \in X. x \sqsubseteq d.$$

(iii) A subset X of \mathcal{D} has a *supremum (sup)* $d \in \mathcal{D}$, notation $\sqcup X = d$ if

$$\begin{aligned} X &\sqsubseteq d \\ \forall d' \in \mathcal{D}. X \sqsubseteq d' &\Rightarrow d \sqsubseteq d' \end{aligned}$$

That is, d is the least upper bound of X .

(iv) A subset X of a poset \mathcal{D} is *directed* if X is nonempty and

$$\forall x, y \in X \exists z \in X [x \sqsubseteq z \& y \sqsubseteq z].$$

(v) A *complete partial order (CPO)* is a poset \mathcal{D} such that there is a least element \perp and every directed $X \subseteq \mathcal{D}$ has a sup $\sqcup X \in \mathcal{D}$.

(vi) A *complete lattice* is a poset \mathcal{D} such that every $X \subseteq \mathcal{D}$ has a sup $\sqcup X$ in \mathcal{D} .

10A.5. DEFINITION. Let \mathcal{D} be a complete lattice.

- (i) An element $d \in \mathcal{D}$ is called *compact* (also called *finite*) if for every directed $Z \subseteq \mathcal{D}$ one has

$$d \sqsubseteq \bigsqcup Z \Rightarrow \exists z \in Z. d \sqsubseteq z.$$

- (ii) Write $\mathcal{K}(\mathcal{D}) = \{d \in \mathcal{D} \mid d \text{ is compact}\}$. \mathcal{D} is called an *algebraic lattice* if for all $x \in \mathcal{D}$ the set $\{e \in \mathcal{K}(\mathcal{D}) \mid e \sqsubseteq x\}$ is directed and

$$x = \bigsqcup \{e \in \mathcal{K}(\mathcal{D}) \mid e \sqsubseteq x\}.$$

- (iii) \mathcal{D} is called an *ω -algebraic lattice* if moreover $\mathcal{K}(\mathcal{D})$ is countable.

10A.6. EXAMPLE. (i) For any set X , the powerset $\langle \mathcal{P}(X), \subseteq \rangle$ is a complete lattice under the subset ordering. It is an ω -algebraic lattice iff X is countable.

- (ii) A typical CPO is $\langle \perp, \text{true}, \text{false} \rangle$, where $\perp < \text{true}$ and $\perp < \text{false}$.

- (iii) Also, given two sets X and Y , we can define the CPO of *partial functions* $X \rightharpoonup Y$ (given as graphs) ordered by subset (on the graphs).

Note that for posets $\mathcal{D} = \langle \mathcal{D}, \sqsubseteq \rangle$ we have

$$\begin{aligned} \omega\text{-algebraic lattice} &\Rightarrow \text{algebraic lattice} \\ &\Rightarrow \text{complete lattice} \\ &\Rightarrow \text{complete partial order} \\ &\Rightarrow \text{partially ordered set.} \end{aligned}$$

In Part III we will work mainly with ω -algebraic lattices and in this chapter mainly with CPOs, in order to exercise flexibility of mind.

10A.7. DEFINITION. Let $f : \mathcal{D} \rightarrow \mathcal{D}'$ be a map.

- (i) f is called *strict* if $f(\perp_{\mathcal{D}}) = \perp_{\mathcal{D}'}$.
(ii) f is called *monotonic* if

$$\forall d, d' \in \mathcal{D}. [d \sqsubseteq d' \Rightarrow f(d) \sqsubseteq f(d')].$$

- (iii) f is called *continuous* if for every directed $X \subseteq \mathcal{D}$

$$\bigsqcup f(X) \in \mathcal{D}' \text{ exists and}$$

$$f(\bigsqcup X) = \bigsqcup f(X).$$

- (iv) If $\mathcal{D}, \mathcal{D}'$ are CPOs, then write $[\mathcal{D} \rightarrow \mathcal{D}'] = \{f : \mathcal{D} \rightarrow \mathcal{D}' \mid f \text{ is continuous}\}$.

- (v) The category **CPO** consists of complete partial orders as objects and continuous functions as morphisms.

- (vi) The category **ALG** consists of ω -algebraic complete lattices as objects and continuous functions as morphisms.

A continuous map $f \in [\mathcal{D} \rightarrow \mathcal{D}']$ is always monotonic (consider for $d \sqsubseteq d'$ the directed set $\{d, d'\}$), but does not need to be strict.

10A.8. DEFINITION. (i) Let $\mathcal{D} = \langle \mathcal{D}, \sqsubseteq \rangle$ be a CPO. The *Scott topology* on \mathcal{D} is defined as follows. A set $O \subseteq \mathcal{D}$ is *Scott open* if

$$\begin{aligned} x \in O \ \& \ x \sqsubseteq y \Rightarrow y \in O; \\ X \subseteq \mathcal{D} \text{ directed} \ \& \ \bigsqcup X \in O \Rightarrow \exists x \in X \cap O. \end{aligned}$$

10A.9. REMARK. For each $x \in \mathcal{D}$, the set $U_x = \{z \mid z \not\sqsubseteq x\}$ is open. \mathcal{D} is a T_0 space. Indeed, if $x \neq y$, then, say, $x \not\sqsubseteq y$ and U_y separates x from y . \mathcal{D} is in general not T_1

(if $x \sqsubseteq y$, if every open set containing x also contains y , therefore the partial order is discrete iff the Scott topology is T_1 .).

10A.10. LEMMA. Let $\mathcal{D}, \mathcal{D}'$ be CPOs and $f : \mathcal{D} \rightarrow \mathcal{D}'$.

- (i) f is continuous $\Leftrightarrow f$ is continuous w.r.t. the Scott topology.
- (ii) For $f, g \in [\mathcal{D} \rightarrow \mathcal{D}']$ define

$$f \sqsubseteq g \Leftrightarrow \forall d \in \mathcal{D}. f(d) \sqsubseteq g(d).$$

Then $\langle [\mathcal{D} \rightarrow \mathcal{D}'], \sqsubseteq \rangle$ is a CPO.

PROOF. See e.g. B[1984], Propositions 1.2.6, 1.2.11. ■

The following is a well-known property of CPOs for finding fixed points.

10A.11. THEOREM. Let $\langle \mathcal{D}, \sqsubseteq \rangle$ be a CPO. There is a functional

$$\mathbf{fix} \in [[\mathcal{D} \rightarrow \mathcal{D}] \rightarrow \mathcal{D}]$$

such that for every $f \in [\mathcal{D} \rightarrow \mathcal{D}]$ one has

$$f(\mathbf{fix}(f)) = \mathbf{fix}(f)$$

PROOF. Take **fix** to be the function which assigns to $f : \mathcal{D} \rightarrow \mathcal{D}$ the element

$$\bigcup_{n \in \omega} f^{(n)}(\perp_{\mathcal{D}}).$$

Then **fix** is continuous and that the equation holds is left as an exercise. ■

10A.12. DEFINITION. A *reflexive structure* is of the form

$D = \langle \mathcal{D}, F, G \rangle$, where \mathcal{D} is a CPO and $F : \mathcal{D} \rightarrow [\mathcal{D} \rightarrow \mathcal{D}]$ and $G : [\mathcal{D} \rightarrow \mathcal{D}] \rightarrow \mathcal{D}$, are continuous and satisfy $F \circ G = 1_{[\mathcal{D} \rightarrow \mathcal{D}]}$, i.e. $F(G(f)) = f$, for all $f \in [\mathcal{D} \rightarrow \mathcal{D}]$. We say that \mathcal{D} is reflexive via F, G .

A reflexive structure \mathcal{D} can be turned into a λ -model.

10A.13. DEFINITION. (i) Let \mathcal{D} be a reflexive structure. We turn \mathcal{D} into an applicative structure by defining the binary operation $d \cdot e = F(d)(e)$.

(ii) Then we define for $M \in \Lambda$ and $\rho \in \text{Env}_{\mathcal{D}}$ the interpretation $\llbracket M \rrbracket_{\rho} \in \mathcal{D}$ by induction on the structure of M . This will turn \mathcal{D} into a λ -model in the sense of Definition 3A.31.

$$\begin{aligned} \llbracket x \rrbracket_{\rho} &\triangleq \rho(x), & \text{for } x \in V; \\ \llbracket MN \rrbracket_{\rho} &\triangleq \llbracket M \rrbracket_{\rho} \cdot \llbracket N \rrbracket_{\rho}; \\ \llbracket \lambda x. M \rrbracket_{\rho} &\triangleq G(\lambda d. \llbracket M \rrbracket_{\rho[x \mapsto d]}). \end{aligned}$$

The last equation could also have been written as follows: $\llbracket \lambda x. M \rrbracket_{\rho} \triangleq F(f)$, where $f(d) = \llbracket M \rrbracket_{\rho[x \mapsto d]}$. That this is indeed a continuous function can be proved by induction on the structure of M .

10A.14. PROPOSITION. Let $\mathcal{D} = \langle \mathcal{D}, F, G \rangle$ be a reflexive structure. Define the maps $\cdot : \mathcal{D} \times \mathcal{D} \rightarrow \mathcal{D}$ and $\llbracket \cdot \rrbracket_{\rho}$ as above. Then

- (i) $\llbracket M[x := N] \rrbracket_{\rho} = \llbracket M \rrbracket_{\rho[x \mapsto \llbracket N \rrbracket_{\rho}]}$.
- (ii) $\langle \mathcal{D}, \cdot, \llbracket \cdot \rrbracket_{\rho} \rangle$ is a λ -model.

10B. Interpreting \mathbb{T}_μ and \mathbb{T}_μ^*

In this Section \mathcal{D} will range over λ -models of the form $\langle \mathcal{D}, F, G \rangle$, with \mathcal{D} a reflexive structure. If \mathcal{A} is a well-founded type algebra of the form $\mathbb{T}^\mathbb{A}/\mathcal{E}$, then there are many morphisms $h : \mathcal{A} \rightarrow \mathcal{P}(\mathcal{D})$ determined by the images $h(\alpha)$ for $\alpha \in \mathbb{A}$.

If \mathcal{A} is not well-founded, like \mathbb{T}_μ , \mathbb{T}_μ^* , $\mathbb{T}[\mathcal{R}]$, $\mathbb{T}[\mathcal{R}]^*$, see Definitions 7C.13 and 7E.17, then it is harder to construct morphisms $\mathcal{A} \rightarrow \mathcal{P}(\mathcal{D})$. We will address this in the present Section for the systems \mathbb{T}_μ , \mathbb{T}_μ^* , which are somewhat more general since they contain solutions of all recursive type equations. The same technique can be applied to $\mathbb{T}[\mathcal{R}]$, $\mathbb{T}[\mathcal{R}]^*$ as well, see Exercise 10D.17.

For this construction we have to find a suitable class of subsets of a λ -model \mathcal{D} closed under \Rightarrow , with the property that \mathbb{T}_μ and \mathbb{T}_μ^* can be mapped by a morphism to the type algebra so obtained.

Approximating λ -models

An important tool to interpret recursive types is to work with λ -models having a *notion of approximation*.

10B.1. DEFINITION (Notion of approximation). Let $\langle \mathcal{D}, F, G \rangle$ be a reflexive structure. A family of continuous functions $\{(\cdot)_n : \mathcal{D} \rightarrow \mathcal{D}\}_{n \in \omega}$ is a *notion of approximation* for \mathcal{D} if it satisfies the following conditions. Write d_n for $(d)_n$.

(i) For all $d \in \mathcal{D}$, and all $n, m \in \omega$:

$$(2) \quad \perp_0 = \perp$$

$$(3) \quad n \leq m \Rightarrow d_n \sqsubseteq d_m$$

$$(4) \quad (d_n)_m = d_{\min(n,m)}$$

$$(5) \quad d = \bigsqcup_{n \in \omega} d_n.$$

(ii) For all $d, e \in \mathcal{D}$ and $n \in \omega$:

$$(6) \quad d_0 \cdot e = d_0 = (d \cdot \perp)_0$$

$$(7) \quad d_{n+1} \cdot e_n = d_{n+1} \cdot e = (d \cdot e_n)_n$$

$$(8) \quad d \cdot e = \bigsqcup_{n \in \omega} (d_{n+1} \cdot e_n).$$

(iii) For $X \subseteq \mathcal{D}$ write $X_n \triangleq \{d_n \mid d \in X\}$ for $n \in \omega$.

10B.2. LEMMA. Suppose $k \leq n$. Then

$$\begin{aligned} (X)_k &\subseteq (X)_n. \\ (X_n)_k &= X_k. \end{aligned}$$

PROOF. Since $d_k = (d_k)_n = (d_n)_k$, by (4). ■

The conditions of Definition 10B.1 are satisfied by the λ -models \mathcal{D}_∞ built by the classical construction of Scott, see e.g. B[1984], Lemma 18.2.8 and Proposition 18.2.13, although some of these may fail for the same construction as modified by Park, see B[1984], Exercise 18.4.21. Furthermore, they also apply to λ -models not explicitly obtained by means of an inverse limit construction, like the models \mathcal{D}_A , see Engeler [1981], or the filter λ -model \mathcal{F} introduced in Barendregt, Coppo, and Dezani-Ciancaglini [1983], see Part III. See Exercises 16E.6 and 10D.5.

Complete-uniform sets

NOTATION. From now on in this section we use \mathcal{D} to denote a reflexive structure with a notion of approximation.

One way to interpret elements of \mathbb{T}_μ is the collection of *ideals* of \mathcal{D} , i.e., the non empty closed subsets of \mathcal{D} with respect to the Scott topology (see MacQueen, Plotkin, and Sethi [1986], Coppo [1985], Cardone and Coppo [1991]). Equivalently, these can be described as the non-empty, downward closed subsets X of \mathcal{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 are closed under increasing sequences and *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 Abadi and Plotkin [1990].)

10B.3. 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 B[1984], Chapter 19, §3,

$$\perp_{\mathcal{D}} = \llbracket (\lambda x. xx)(\lambda x. xx) \rrbracket,$$

hence $\perp_{\mathcal{D}}$ belongs to all types, which accounts for non-emptiness, see Example 7A.3(ii). On the other hand, the interpretation of $Y \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$, the fixed point combinator, defines the map $\mathbf{fix} \in [[\mathcal{D} \rightarrow \mathcal{D}] \rightarrow \mathcal{D}]$ introduced in Theorem 10A.11. Now, it was shown in Example 7A.3(iii) that Y has type $(A \rightarrow A) \rightarrow A$, and this motivates completeness, because then every type has to be closed under least upper-bounds of increasing sequences. Concerning uniformity, observe that if X is an ideal of \mathcal{D} , the set $X_n \triangleq \{d_n \mid d \in X\}$ is not, in general, an ideal of \mathcal{D} . For example, in \mathcal{D}_∞ the subset \mathcal{D}_0 is not downward closed.

The construction described below can be performed, more generally, using n -ary complete and uniform relations over \mathcal{D} as interpretations of types. This applies in particular to the interpretation of types as (complete and uniform) *partial equivalence relations* (*PER's*) over \mathcal{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 Exercises 10D.11, 10D.12 and 10D.13). The subsets that interpret types are required to respect the notion of approximation, in the following sense.

10B.4. DEFINITION. (i) A subset $X \subseteq \mathcal{D}$ is *complete* if

- $\perp_{\mathcal{D}} \in X$;
- if $d^0 \sqsubseteq d^1 \sqsubseteq \dots$ are all in X , then so is $\bigsqcup_{n \in \omega} d^n$.

(ii) Let ∇ be a complete subset of \mathcal{D}_0 . A subset $X \subseteq \mathcal{D}$ is called *∇ -uniform* if

- $X_0 = \nabla$,

- $X_n \subseteq X$ for all $n \in \omega$.

(iii) Let $\mathcal{CU}_\nabla(\mathcal{D})$, or just $\mathcal{CU}(\mathcal{D})$ if ∇ is clear from the context, denote the set of complete and ∇ -uniform subsets of \mathcal{D} .

The following provides some properties of this set.

10B.5. PROPOSITION (Properties of complete and uniform subsets). *Let $X, Y \in \mathcal{CU}_\nabla(\mathcal{D})$. Then for all $n \in \omega$.*

- (i) $X_n \in \mathcal{CU}_\nabla(\mathcal{D})$.
- (ii) $X = Y \Leftrightarrow \forall n \in \omega, X_n = Y_n$;

PROOF. (i) X_n is uniform by Definition 10B.1 (4). It also is complete: since $\perp \in X$ also $\perp = \perp_n \in X_n$; and if d^k is an increasing chain in X_n , then $\sqcup_k d^k = \sqcup_k d_n^k = (\sqcup_k d^k)_n \in X_n$, by continuity of the approximation mappings.

- (ii) Easy, using Definitions 10B.4 and 10B.1(5). ■

10B.6. PROPOSITION. *Let the sequence of sets $X^{(n)} \in \mathcal{CU}_\nabla(\mathcal{D})$, $n \in \omega$, satisfy*

$$(9) \quad \forall n \in \omega \left(X^{(n)} = (X^{(n+1)})_n \right).$$

Define $X^{(\infty)} \triangleq \{d \mid d_n \in X^{(n)}\} \in \mathcal{CU}_\nabla(\mathcal{D})$. Then this set is the unique complete and ∇ -uniform subset of \mathcal{D} , such that $(X^{(\infty)})_n = X^{(n)}$, for all $n \in \omega$.

PROOF. We first claim that

- (i) $(X^{(n)})_k = X^{(k)}$, for $k \leq n$;
- (ii) $X^{(k)} \subseteq X^{(n)}$ for $k \leq n$;
- (iii) $(X^{(n)})_k \subseteq X^{(k)}$ for all k, n .

Item (i) is proved by induction on $n - k$. If $n = k$, then

$$\begin{aligned} (X^{(n)})_n &= ((X^{(n+1)})_n)_n \\ &= (X^{(n+1)})_n \\ &= X^{(n)}. \end{aligned}$$

Now we show the equation for $n, k - 1$, assuming it for n, k .

$$\begin{aligned} (X^{(n)})_{k-1} &= ((X^{(n)})_k)_{k-1} \\ &= (X^{(k)})_{k-1}, \quad \text{by the induction hypothesis,} \\ &= X^{(k-1)}. \end{aligned}$$

Item (ii) follows from (i), as $(X^{(n)})_k \subseteq X^{(n)}$. Item (iii) follows for $k \leq n$ from (i), and for $n \leq k$, then one has by (ii)

$$(X^{(n)})_k \subseteq X^{(n)} \subseteq X^{(k)}.$$

To show that $X^{(\infty)} \in \mathcal{CU}_\nabla(\mathcal{D})$, notice $(X^{(\infty)})_0 = \nabla$, because if $d \in \nabla$ then $d \in X^{(i)}$ for all $i \in \omega$ as each $X^{(i)}$ is uniform. But then $d \in (X^{(i)})_i$, hence $d_i = d \in X^{(i)}$ and therefore $\nabla \subseteq (X^{(\infty)})_0$. Conversely, observe that $(X^{(\infty)})_0 \subseteq X^{(0)} = \nabla$, by assumption (9).

As to uniformity of $X^{(\infty)}$, let $d \in X^{(\infty)}$. Then $(d_n)_k \in (X^{(n)})_k \subseteq X^{(k)}$, by (iii). Therefore $d_n \in X^{(\infty)}$.

As to completeness, let $d^{(j)} \in X^{(\infty)}$ be a chain of elements: then $(d^{(j)})_n \in X^{(n)}$, for all $n \in \omega$, and therefore also $\bigsqcup_j (d^{(j)})_n = (\bigsqcup_j d^{(j)})_n \in X^{(n)}$ by continuity of the approximation mappings, which entails that $\bigsqcup_j d^{(j)} \in X^{(\infty)}$. By (iii) we have $X^{(n+1)} \subseteq X^{(\infty)}$, for every n . Hence $X^{(n)} \subseteq (X^{(n+1)})_n \subseteq (X^{(\infty)})_n$. Therefore $X^{(n)} = (X^{(\infty)})_n$, 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 following result shows that each class $\mathcal{CU}_\nabla(\mathcal{D})$ is closed under \Rightarrow .

10B.7. PROPOSITION. *If $X, Y \in \mathcal{CU}(\mathcal{D})$ then $(X \Rightarrow Y) \in \mathcal{CU}(\mathcal{D})$.*

PROOF. We first show that $\nabla = (X \Rightarrow Y)_0$. Assume $d \in \nabla$, and let $a \in X$. Then

$$\begin{aligned} d \cdot a &= d_0 \cdot a, && \text{as } \nabla \subseteq \mathcal{D}_0, \\ &= d, && \text{by Definition 10B.1 (6).} \end{aligned}$$

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 $d \in (X \Rightarrow Y)$. Then

$$d_0 = (d \cdot \perp_{\mathcal{D}})_0, \quad \text{by Definition 10B.1 (6).}$$

but $\perp_{\mathcal{D}} \in \nabla = X_0 \subseteq X$, hence $d \cdot \perp_{\mathcal{D}} \in Y$, because $d \in X \Rightarrow Y$, so $(d \cdot \perp_{\mathcal{D}})_0 \in Y_0 = \nabla$ and finally $d_0 \in \nabla$.

Also we have $\perp_{\mathcal{D}} \in X \Rightarrow Y$, since $\perp_{\mathcal{D}} \in \nabla = (X \Rightarrow Y)_0 \subseteq X \Rightarrow Y$.

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 $a \in X$

$$d_n \cdot a = (d \cdot a_{n-1})_{n-1}, \quad \text{by Definition 10B.1 (7),}$$

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 $X \Rightarrow Y$, in order to show that the sup also is in $X \Rightarrow Y$. Let $a \in X$. Then

$$\left(\bigsqcup_{j \in \omega} d^{(j)} \right) \cdot a = \bigsqcup_{j \in \omega} (d^{(j)} \cdot a) \in Y,$$

by continuity of application and completeness of Y . Therefore $\bigsqcup_{j \in \omega} d^{(j)} \in (X \Rightarrow Y)$. ■

The following property of the interpretation of function types is the key to the whole construction of the interpretation of recursive types.

10B.8. PROPOSITION. *For any $X, Y \in \mathcal{CU}(\mathcal{D})$ 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 10B.1(7), so $d_{n+1} \in X_n \Rightarrow Y_n$ and hence $d_{n+1} \in (X_n \Rightarrow Y_n)_{n+1}$. Conversely, let $d_{n+1} \in (X_n \Rightarrow Y_n)_{n+1}$, and assume that $a \in X$. Then $a_n \in X_n$. Now

$$d_{n+1} \cdot a = d_{n+1} \cdot a_n \in Y_n \subseteq Y,$$

using again Definition 10B.1(7) and uniformity of Y and $X_n \Rightarrow Y_n$, 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.

10B.9. DEFINITION. $\mathcal{S}(\mathcal{D}) \triangleq \langle \mathcal{C}\mathcal{U}(\mathcal{D}), \Rightarrow \rangle$

As an example, let us see how the theory developed so far allows to interpret a type $T = T \rightarrow T$ as a complete and uniform subset $\Xi = \Xi \Rightarrow \Xi$. The latter is built in denumerably many steps

$$\Xi^{(0)}, \Xi^{(1)}, \Xi^{(2)}, \dots$$

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 $n \in \omega$

$$(10) \quad (\Xi^{(n+1)})_n = \Xi^{(n)},$$

then we can exploit Proposition 10B.6 and take $\Xi = \Xi^{(\infty)}$, so that $\Xi_n = \Xi^{(n)}$ for all $n \in \omega$. The proof of (10) is therefore the core of the technique, and appeals in an essential way to Proposition 10B.8. The fact that $\Xi = \Xi \Rightarrow \Xi$ is then a direct consequence of Proposition 10B.5 (ii). 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. The same method will be applied to the interpretation of simultaneous recursions in Exercise 10D.17.

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 $X, Y \in \mathcal{C}\mathcal{U}(\mathcal{D})$ and $n \in \omega$, let $X \Rightarrow^{n+1} Y$ denote $(X \Rightarrow Y)_{n+1}$.

10B.10. LEMMA. (i) If $A \subseteq \mathcal{D}$ is ∇ -uniform, then $(A)_n = A \cap \mathcal{D}_n$.

(ii) Let $X, Y \in \mathcal{C}\mathcal{U}(\mathcal{D})$. Then $X \Rightarrow^{n+1} Y = (X \Rightarrow Y) \cap \mathcal{D}_{n+1}$.

(iii) $X \Rightarrow^{n+1} Y = X_n \Rightarrow^{n+1} Y_n$.

PROOF. (i) Note that $A_n \subseteq A \cap \mathcal{D}_n$ by uniformity. Conversely, if $d \in A \cap \mathcal{D}_n$, then $d = d_n \in A_n$, by the idempotency of $(\cdot)_n$, implied by 10B.1 (4).

(ii) By (i).

(iii) By Proposition 10B.8. ■

10B.11. DEFINITION (Approximate Interpretations of Types). Given a type $A \in \mathbb{T}_\mu$, a number $n \in \omega$, and a type environment $\eta : \mathbb{A} \rightarrow \mathcal{C}\mathcal{U}(\mathcal{D})$, define the n -th *approximation* of the interpretation of A in the environment η , notation $\mathcal{I}^n[A]_\eta$, by induction on n

and the complexity of the type A .

$$\begin{aligned}\mathcal{I}^0[A]_\eta &\triangleq \nabla; \\ \mathcal{I}^{n+1}[\alpha]_\eta &\triangleq (\eta(\alpha))_{n+1}; \\ \mathcal{I}^{n+1}[A_1 \rightarrow A_2]_\eta &\triangleq \mathcal{I}^n[A_1]_\eta \Rightarrow^{n+1} \mathcal{I}^n[A_2]_\eta; \\ \mathcal{I}^{n+1}[\mu\alpha.A_1]_\eta &\triangleq \mathcal{I}^{n+1}[A_1]_{\eta[\alpha \mapsto \mathcal{I}^n[\mu\alpha.A_1]_\eta]}.\end{aligned}$$

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 \mathcal{D} and $\mathcal{I}^n[A]_\eta \subseteq \mathcal{D}_n$. Below we shall make frequent use of the following properties, whose easy inductive proofs are left as an exercise.

10B.12. LEMMA. *Let $A \in \mathbb{T}_\mu$, $\alpha \in \mathbb{A}$, $n \in \omega$ and η be an environment.*

- (i) $\mathcal{I}^n[A]_\eta = \mathcal{I}^n[A]_{(\eta \upharpoonright n)}$, where $(\eta \upharpoonright n)(\alpha) \triangleq (\eta(\alpha))_n$.
- (ii) $\mathcal{I}^{n+1}[A]_\eta = \mathcal{I}^{n+1}[A]_{\eta[\alpha \mapsto (\eta(\alpha))_n]}$, if $\mu\alpha.A$ is non-circular.
- (iii) $\mathcal{I}^n[\mu\alpha.A]_\eta = \nabla$, if $\mu\alpha.A$ is circular. ■

10B.13. 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 we have to prove the conjunction of (a) and (b):

- (a) $\mathcal{I}^n[A]_\eta \subseteq \mathcal{I}^{n+1}[A]_\eta$;
- (b) $d \in \mathcal{I}^{n+1}[A]_\eta \Rightarrow d_n \in \mathcal{I}^n[A]_\eta$.

We do this by induction on n . The basis is obvious. The induction step is proved by induction on the complexity of the type A .

Case $A \equiv \alpha$. Then $\mathcal{I}^n[\alpha]_\eta = (\eta(\alpha))_n$, and we can use Lemma 10B.2.

Case $A \equiv A_1 \rightarrow A_2$. Then we 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 $d \in \mathcal{I}^{n-1}[A_1]_\eta \Rightarrow^n \mathcal{I}^{n-1}[A_2]_\eta$ and $a \in \mathcal{I}^n[A_1]_\eta$. By the induction hypothesis on n for (b) one has $a_{n-1} \in \mathcal{I}^{n-1}[A_1]_\eta$. 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 for (a), and equation (7) of Definition 10B.1. If

$$d \in \mathcal{I}^n[A_1]_\eta \Rightarrow^{n+1} \mathcal{I}^n[A_2]_\eta$$

and $a \in \mathcal{I}^{n-1}[A_1]_\eta$, then $a \in \mathcal{I}^n[A_1]_\eta$ by the induction hypothesis for (a). Hence $d \cdot a \in \mathcal{I}^n[A_2]_\eta$ and by the induction hypothesis for (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 (7) of Definition 10B.1, $(d \cdot a)_{n-1} = d_n \cdot a$, therefore

$$d_n \in \mathcal{I}^{n-1}[A_1]_\eta \Rightarrow^n \mathcal{I}^{n-1}[A_2]_\eta.$$

Case $A \equiv \mu\alpha.A_1$. If A is circular, then the property holds trivially by Lemma 10B.12(iii). Otherwise

$$\begin{aligned} \mathcal{I}^n[\mu\alpha.A_1]_\eta &= \mathcal{I}^n[A_1]_{\eta[\alpha \mapsto \mathcal{I}^{n-1}[\mu\alpha.A_1]_\eta]} \\ &= \mathcal{I}^n[A_1]_{\eta[\alpha \mapsto (\mathcal{I}^n[\mu\alpha.A_1]_\eta)_{n-1}]}, \quad \text{by the IH on } n, \\ &= \mathcal{I}^n[A_1]_{\eta[\alpha \mapsto \mathcal{I}^n[\mu\alpha.A_1]_\eta]}, \quad \text{by Lemma 10B.12(ii),} \\ &\subseteq \mathcal{I}^{n+1}[A_1]_{\eta[\alpha \mapsto \mathcal{I}^n[\mu\alpha.A_1]_\eta]}, \quad \text{by the IH on } A, \\ &= \mathcal{I}^{n+1}[\mu\alpha.A_1]_\eta. \end{aligned}$$

Now, let $d \in \mathcal{I}^{n+1}[\mu\alpha.A_1]_\eta = \mathcal{I}^{n+1}[A_1]_{\eta[\alpha \mapsto \mathcal{I}^n[\mu\alpha.A_1]_\eta]}$. Then by the induction hypothesis on the complexity of the type one has $d_n \in \mathcal{I}^n[A_1]_{\eta[\alpha \mapsto \mathcal{I}^n[\mu\alpha.A_1]_\eta]}$. Hence by Lemma 10B.12 one has $d_n \in \mathcal{I}^n[A_1]_{\eta[\alpha \mapsto (\mathcal{I}^n[\mu\alpha.A_1]_\eta)_{n-1}]}$, 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 the induction hypothesis on n . Therefore

$$d_n \in \mathcal{I}^n[A_1]_{\eta[\alpha \mapsto \mathcal{I}^{n-1}[\mu\alpha.A_1]_\eta]} = \mathcal{I}^n[\mu\alpha.A_1]_\eta. \blacksquare$$

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 10B.6.

10B.14. DEFINITION (Type Interpretations). For every type A and type environment η , define the *type interpretation*

$$\mathcal{I}[A]_\eta \triangleq \mathcal{I}^\infty[A]_\eta,$$

which is $X^{(\infty)}$ for $X^{(n)} = (\mathcal{I}^n[A]_\eta)$.

Observe that the interpretation of types depends on the choice of the base ∇ . For example, when $\mathcal{D} = \mathcal{D}_\infty$ is defined as the inverse limit of a sequence of CPOs (see 16C.1-16C.23), where \mathcal{D}_0 consists of the two points \perp, \top , if $\nabla = \{\perp\}$ then $\top \notin \mathcal{I}[\mu t.t \rightarrow t]_\eta$, whereas $\mathcal{I}[\mu t.t \rightarrow t]_\eta = \mathcal{D}$ when $\nabla = \{\perp, \top\}$. Note that if $\nabla = \{\perp\}$, then we have $\mathcal{D}_\infty \notin \mathcal{CU}(\mathcal{D}_\infty)$. But $\mathcal{I}^\infty[A]_\eta \in \mathcal{CU}(\mathcal{D}_\infty)$, for all A and η .

10B.15. PROPOSITION. For any type A , environment η , and $n \in \omega$ one has

$$(\mathcal{I}[A]_\eta)_n = \mathcal{I}^n[A]_\eta.$$

PROOF. By Proposition 10B.6. \blacksquare

10B.16. PROPOSITION. (i) For any type A and environment η one has

$$\mathcal{I}[\mu\alpha.A]_\eta = \mathcal{I}[A]_{\eta[\alpha \mapsto \mathcal{I}[\mu\alpha.A]_\eta]}.$$

(ii) For any pair of types A, B , any type variable α and any environment η one has

$$\mathcal{I}[A[\alpha := B]]_\eta = \mathcal{I}[A]_{\eta[\alpha \mapsto \mathcal{I}[B]_\eta]}.$$

PROOF. (i) If $\mu\alpha.A$ is circular, then Lemma 10B.12(iii) applies. Otherwise, by Proposition 10B.5 it suffices to show that for all n

$$(\mathcal{I}[\mu\alpha.A]_\eta)_n = (\mathcal{I}[A]_{\eta[\alpha \mapsto \mathcal{I}[\mu\alpha.A]_\eta]})_n.$$

By Proposition 10B.15 for all B, η, n

$$(\mathcal{I}\llbracket B \rrbracket_{\eta})_n = \mathcal{I}^n\llbracket B \rrbracket_{\eta}.$$

Case $n = 0$. Then both sides equal ∇ . Case $n + 1$. Then we have

$$\begin{aligned} \mathcal{I}^{n+1}\llbracket \mu\alpha.A \rrbracket_{\eta} &= \mathcal{I}^{n+1}\llbracket A \rrbracket_{\eta[\alpha \mapsto \mathcal{I}^n\llbracket \mu\alpha.A \rrbracket_{\eta}]}, && \text{by Definition 10B.11} \\ &= \mathcal{I}^{n+1}\llbracket A \rrbracket_{\eta[\alpha \mapsto \mathcal{I}\llbracket \mu\alpha.A \rrbracket_{\eta}]}, && \text{by Lemma 10B.12(ii),} \\ &&& \text{applied to } \eta' = \eta[\alpha \mapsto \mathcal{I}\llbracket \mu\alpha.A \rrbracket_{\eta}]. \end{aligned}$$

(ii) By a double induction, do Exercise 10D.7. ■

10B.17. THEOREM (Properties of Type Interpretations). *The following conditions are satisfied, for any type environment η and all types A, B .*

- (i) $\mathcal{I}\llbracket \alpha \rrbracket_{\eta} = \eta(\alpha)$.
- (ii) $\mathcal{I}\llbracket A \rightarrow B \rrbracket_{\eta} = \mathcal{I}\llbracket A \rrbracket_{\eta} \Rightarrow \mathcal{I}\llbracket B \rrbracket_{\eta}$.
- (iii) $\mathcal{I}\llbracket \mu\alpha.A \rrbracket_{\eta} = \mathcal{I}\llbracket A[\alpha := \mu\alpha.A] \rrbracket_{\eta}$.

PROOF. (i) $d \in \mathcal{I}\llbracket \alpha \rrbracket_{\eta} \Leftrightarrow \forall n \in \omega. d_n \in \mathcal{I}^n\llbracket \alpha \rrbracket_{\eta} = (\eta(\alpha))_n \Leftrightarrow d \in \eta(\alpha)$.

(ii) Observe that for all $n \in \omega$ one has

$$\begin{aligned} (\mathcal{I}\llbracket A \rightarrow B \rrbracket_{\eta})_{n+1} &= \mathcal{I}^{n+1}\llbracket A \rightarrow B \rrbracket_{\eta}, && \text{by Proposition 10B.15,} \\ &= \mathcal{I}^n\llbracket A \rrbracket_{\eta} \Rightarrow^{n+1} \mathcal{I}^n\llbracket B \rrbracket_{\eta} \\ &= (\mathcal{I}\llbracket A \rrbracket_{\eta})_n \Rightarrow^{n+1} (\mathcal{I}\llbracket B \rrbracket_{\eta})_n \\ &= (\mathcal{I}\llbracket A \rrbracket_{\eta} \Rightarrow \mathcal{I}\llbracket B \rrbracket_{\eta})_{n+1}, && \text{by Proposition 10B.8.} \end{aligned}$$

Now the result follows by induction from Proposition 10B.5 (ii), observing that

$$(\mathcal{I}\llbracket A \rightarrow B \rrbracket_{\eta})_0 = \nabla = (\mathcal{I}\llbracket A \rrbracket_{\eta} \Rightarrow \mathcal{I}\llbracket B \rrbracket_{\eta})_0.$$

(iii) By Lemma 10B.16(i),(ii). ■

Soundness and completeness for interpreting \mathbb{T}_{μ} and \mathbb{T}_{μ}^*

Theorem 10B.17 implies that $\mathcal{I}\llbracket - \rrbracket_{\eta} : (\mathbb{T}_{\mu} / =_{\mu}) \longrightarrow \mathcal{D}$ is a type algebra morphism, for all type environments η . We have immediately the following corollary.

10B.18. PROPOSITION (Soundness of $\vdash_{\lambda\mu}$). *For all $\rho \in \text{Env}_{\mathcal{D}}, \eta : \mathbb{A} \rightarrow \mathcal{C}\mathcal{U}(\mathcal{D})$*

$$\Gamma \vdash_{\lambda\mu} M : A \Rightarrow \Gamma \models_{\mathcal{D}, \rho, \mathcal{I}\llbracket - \rrbracket_{\eta}} M : A.$$

PROOF. By Proposition 10A.3. ■

We now proceed to show that the type interpretation introduced in Definition 10B.14 and characterized in Theorem 10B.17 induces a type algebra morphism from \mathbb{T}_{μ}^* , see notation 7E.19, to $\mathcal{S}(\mathcal{D})$, see Definition 7E.17 and Definition 10B.9. To this end, we introduce a notion of approximate interpretation for the regular trees which result from unfolding infinitely often types in \mathbb{T}_{μ} . This interpretation is of interest in its own, and is indeed the notion of interpretation which was taken as basic in [Cardone and Coppo \[1991\]](#).

10B.19. DEFINITION (**Approximate interpretation** of regular trees). Given $n \in \omega$, $t \in \text{Tr}_{\text{reg}}$, $\alpha \in \mathbb{A}$ and $\eta : \mathbb{A} \rightarrow \mathcal{CU}(\mathcal{D})$ a type environment, define the n -th **approximation** of the regular tree t , notation $\mathcal{T}^n[t]_\eta$, by induction on n as follows.

$$\begin{aligned}\mathcal{T}^0[t]_\eta &\triangleq \nabla \\ \mathcal{T}^{n+1}[\alpha]_\eta &\triangleq (\eta(\alpha))_{n+1} \\ \mathcal{T}^{n+1}[\bullet]_\eta &\triangleq \nabla \\ \mathcal{T}^{n+1}[t_1 \rightarrow t_2]_\eta &\triangleq (\mathcal{T}^n[t_1]_\eta \Rightarrow^{n+1} \mathcal{T}^n[t_2]_\eta).\end{aligned}$$

10B.20. 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 $(-)^* = (-)_\mu^* : \mathbb{T}_\mu \rightarrow \text{Tr}_{\text{reg}}$ is given in Notation 7E.19.

PROOF. By induction on n . The basis is obvious, while the induction step is proved by induction on $A \in \mathbb{T}_\mu$. Again the basis is clear. For the induction step we distinguish cases.

Case $A \equiv A_1 \rightarrow A_2$. Then

$$\begin{aligned}\mathcal{T}^{n+1}[(A_1 \rightarrow A_2)^*]_\eta &= \mathcal{T}^{n+1}[(A_1)^* \rightarrow (A_2)^*]_\eta \\ &= \mathcal{T}^n[(A_1)^*]_\eta \Rightarrow^{n+1} \mathcal{T}^n[(A_2)^*]_\eta \\ &= \mathcal{I}^n[A_1]_\eta \Rightarrow^{n+1} \mathcal{I}^n[A_2]_\eta \\ &= \mathcal{I}^{n+1}[A_1 \rightarrow A_2]_\eta;\end{aligned}$$

Case $A \equiv \mu\alpha.A_1$. 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 . ■

10B.21. PROPOSITION. Let $A, B \in \mathbb{T}_\mu$. Then for all type environments η one has

$$A =_\mu^* B \Rightarrow \mathcal{I}[A]_\eta = \mathcal{I}[B]_\eta.$$

PROOF. Let $A, B \in \mathbb{T}_\mu$ and assume $A =_\mu^* B$. Then by Lemma 10B.20 for all n, η

$$\begin{aligned}\mathcal{I}^n[A]_\eta &= \mathcal{T}^n[A^*]_\eta \\ &= \mathcal{T}^n[B^*]_\eta \\ &= \mathcal{I}^n[B]_\eta.\end{aligned}$$

Then the statement follows from Proposition 10B.5(ii). ■

As an immediate consequence we have the soundness of rule (equal) in Definition 7A.2 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.

10B.22. COROLLARY (Soundness of $\vdash_{\lambda\mu^*}$). *For all $\rho \in \text{Env}_{\mathcal{D}}$, $\eta : \mathbb{A} \rightarrow \mathcal{C}\mathcal{U}(\mathcal{D})$*

$$\Gamma \vdash_{\lambda\mu^*} M : A \Rightarrow \Gamma \models_{\mathcal{D}, \rho, \mathcal{I}[_], \eta} M : A.$$

PROOF. Again by Proposition 10A.3. ■

Applications of soundness

One application of the type interpretations described so far is the following, easy proof of a standard result on definability by *untypes* terms, see B[1984], Exercise 19.4.5. Let \mathcal{D} be a λ -model. An element $d \in \mathcal{D}$ is called λ -definable if there exists an $M \in \Lambda^\phi$ such that $d = \llbracket M \rrbracket^{\mathcal{D}}$.

In the sequel we take for \mathcal{D} Scott's λ -model \mathcal{D}_∞ , see 16C.1-16C.23, where

- \mathcal{D}_0 is the two point lattice $\{\perp_{\mathcal{D}_0}, \top_{\mathcal{D}_0}\}$;
- $\langle i_0, j_0 \rangle$ is the so-called standard embedding-projection pair,

$$i_0(d) \triangleq \lambda e. d, \quad j_0(f) \triangleq \perp_{\mathcal{D}_0};$$

- $\nabla \triangleq \{\perp_{\mathcal{D}_0}\}$.

The sets \mathcal{D}_n are considered as subsets of \mathcal{D}_∞ by identifying them with $\Phi_{n\infty}(\mathcal{D}_n)$. Then the maps $\lambda d. d_n : \mathcal{D} \rightarrow \mathcal{D}$ form a notion of approximation for \mathcal{D} . Note that $\top = \top_{\mathcal{D}} = \langle \top_{\mathcal{D}_n} \rangle$, hence $\top_n = \top_{\mathcal{D}_n}$. Moreover, $\perp \in \mathcal{D}$ is λ -definable: $\perp = \llbracket \Omega \rrbracket$.

10B.23. PROPOSITION. $\top = \top_{\mathcal{D}}$ is not λ -definable.

PROOF. Let $L = \mu\alpha.\alpha \rightarrow \alpha$, and recall that every $M \in \Lambda^\phi$ has type L . Then $\llbracket M \rrbracket \in \mathcal{I}[L]$, by soundness, Proposition 10B.18. Now let $d = \llbracket M \rrbracket$. Then

$$d_0 = \llbracket M \rrbracket_0 \in (\mathcal{I}[L])_0 = \mathcal{I}^0[L] = \nabla = \{\perp_0\}.$$

But $\top_0 = \top_{\mathcal{D}_0} \neq \perp_{\mathcal{D}_0} = \perp_0$. Hence $d \neq \top$. ■

Another application of Corollary 10B.22 shows that types of a special form are inhabited only by unsolvable terms, in the sense of B[1984], §8.3. This generalizes the easy observation that, if $\vdash_{\lambda\mu^*} M : \mu\alpha.\alpha$ for a closed term M , then M is unsolvable. In fact $\mathcal{I}[\mu\alpha.\alpha] = \nabla$, so $\llbracket M \rrbracket = \perp$ by soundness (Corollary 10B.22), and in \mathcal{D} any M such that $\llbracket M \rrbracket = \perp$ is unsolvable, see B[1984], Theorem 19.2.4(i).

We need two Lemmas about the standard \mathcal{D}_∞ that do not hold if $i_0(\top_{\mathcal{D}_0}) \neq \top_{\mathcal{D}_1}$.

10B.24. LEMMA. (i) $i_n(\top_n) = \top_{n+1}$, for all $n \geq 0$;
(ii) $\Phi_{n,\infty}(\top_n) = \top$, for all $n \geq 0$.

PROOF. (i) By induction on n .

(ii) By (i). ■

Note that $\mathcal{D} \notin \mathcal{C}\mathcal{U}(\mathcal{D})$, because $\mathcal{D}_0 \neq \nabla$. In the next Lemma we show $\mathcal{D} - \{\top\} \in \mathcal{C}\mathcal{U}(\mathcal{D})$.

10B.25. LEMMA. *Let $\bar{\mathcal{D}} \triangleq \mathcal{D} - \{\top\}$. Then $\bar{\mathcal{D}} \in \mathcal{C}\mathcal{U}(\mathcal{D})$.*

PROOF. First we show that $(\bar{\mathcal{D}})_0 = \nabla$. Let $d \in \bar{\mathcal{D}}$. Then $\Phi_{0\infty}(d_0) = \Phi_{0\infty} \circ \Phi_{\infty 0}(d_0) \sqsubseteq d$. But $\Phi_{0\infty}(\top_0) = \top$, by Lemma 10B.24(ii). Hence $d_0 \neq \top_0$, so $d_0 = \perp_0 \in \nabla$.

Now we show that $\bar{\mathcal{D}}$ is complete. Note that $d_0 = \perp_0$, for $d \in \bar{\mathcal{D}}$, because if $d_0 = \top_0$, then $\Phi_{0\infty}(d_0) = \top$, but $\Phi_{0\infty}(d_0) \sqsubseteq d \neq \top$. Now let $e \triangleq \sup_{\mathcal{D}}\{d \mid d \in \bar{\mathcal{D}}\}$. We must show that $e \in \bar{\mathcal{D}}$, i.e. $e \neq \top$. It suffices to show $e_0 \neq \top_0$. Now indeed

$$\begin{aligned} e_0 &= \Phi_{\infty 0}(e) = \Phi_{\infty 0}(\sup\{d \mid d \in \bar{\mathcal{D}}\}) \\ &= \sup\{\Phi_{\infty 0}(d) \mid d \in \bar{\mathcal{D}}\} = \sup\{d_0 \mid d \in \bar{\mathcal{D}}\} \\ &= \sup\{\perp_0\} = \perp_0 \neq \top_0. \blacksquare \end{aligned}$$

10B.26. THEOREM. Let M be a closed term. Suppose for $n, m \geq 0$

$$\vdash_{\lambda\mu^*} M : \alpha_1 \rightarrow \cdots \rightarrow \alpha_n \rightarrow \mu\alpha.(\beta_1 \rightarrow \cdots \rightarrow \beta_m \rightarrow \alpha).$$

Then M is unsolvable.

PROOF. Define η by $\eta(\gamma) = \bar{\mathcal{D}}$ for all $\gamma \in \mathbb{A}$. Claim

$$(\bar{\mathcal{D}} \Rightarrow \cdots \Rightarrow \bar{\mathcal{D}} \Rightarrow \bar{\mathcal{D}}) \subsetneq \bar{\mathcal{D}}. \quad (1)$$

Observe that $\top \notin \bar{\mathcal{D}} \Rightarrow \cdots \Rightarrow \bar{\mathcal{D}} \Rightarrow \bar{\mathcal{D}}$, hence any element of $\bar{\mathcal{D}} \Rightarrow \cdots \Rightarrow \bar{\mathcal{D}} \Rightarrow \bar{\mathcal{D}}$ is also an element of $\bar{\mathcal{D}}$. In order to show that the inclusion is strict, let the *step function* ($e \mapsto e'$), for compact elements $e, e' \in \mathcal{D}$, be defined by

$$(e \mapsto e')(d) \triangleq \begin{cases} e', & \text{if } e \sqsubseteq d, \\ \perp, & \text{otherwise.} \end{cases}$$

Now consider

$$(\vec{e} \mapsto \top) \triangleq (e_1 \mapsto (e_2 \mapsto \cdots \mapsto (e_p \mapsto \top) \cdots)),$$

where $e_1, \dots, e_p \in \bar{\mathcal{D}}$ are compact elements, with $e_1 \neq \perp$. One has $(\vec{e} \mapsto \top)(\perp) = \perp$, hence $(\vec{e} \mapsto \top) \in \bar{\mathcal{D}}$, but $(\vec{e} \mapsto \top) \notin (\bar{\mathcal{D}}^m \Rightarrow \bar{\mathcal{D}})$, establishing (1).

Now, assume towards a contradiction that M is solvable. Then $MN^\vec{N} = \mathbf{I}$, for some $\vec{N} = N_1, \dots, N_\ell$. Hence $\llbracket M \rrbracket \cdot \llbracket N_1 \rrbracket \cdots \llbracket N_\ell \rrbracket = \llbracket \lambda x.x \rrbracket$. Note that $\llbracket N_i \rrbracket \in \bar{\mathcal{D}}$, by Proposition 10B.23. Let

$$T \triangleq \alpha_1 \rightarrow \cdots \rightarrow \alpha_n \rightarrow \mu\alpha.(\beta_1 \rightarrow \cdots \rightarrow \beta_m \rightarrow \alpha).$$

Case $m = 0$. If $\ell \geq n$ then $\llbracket M \rrbracket \cdot \llbracket N_1 \rrbracket \cdots \llbracket N_n \rrbracket \in \nabla$, which is impossible because then $\llbracket MN_1 \cdots N_n \rrbracket = \perp$, so $MN_1 \cdots N_n$ is unsolvable. Otherwise, $\ell < n$ and

$$\llbracket \lambda x.x \rrbracket \in \bar{\mathcal{D}}^{n-\ell} \Rightarrow \nabla.$$

Then $\llbracket (\lambda x.x)^{n-\ell} \rrbracket \in \nabla$, hence $(\lambda x.x)^{n-\ell} = \mathbf{I}$ would be unsolvable, again a contradiction.

Case $m > 0$. We can assume that $\ell < n$, otherwise take some μ -unfoldings of T . Now

$$\llbracket \lambda x.x \rrbracket \in (\bar{\mathcal{D}}^{n-\ell} \Rightarrow \mathcal{I}[\mu\alpha.(\beta_1 \rightarrow \cdots \rightarrow \beta_m \rightarrow \alpha)]_\eta)$$

because, for $i = 1, \dots, \ell$, $\llbracket N_i \rrbracket \in \bar{\mathcal{D}} = \mathcal{I}[\alpha_i]_\eta$. By applying $\llbracket \lambda x.x \rrbracket$ to $\mathbf{I}^{\sim(n-(\ell+1))}d$, we get

$$\forall d \in \bar{\mathcal{D}}. d \in \mathcal{I}[\mu\alpha. \beta_1 \rightarrow \cdots \rightarrow \beta_m \rightarrow \alpha]_\eta.$$

Therefore $\mathcal{I}[\mu\alpha.\beta_1 \rightarrow \dots \rightarrow \beta_m \rightarrow \alpha]_\eta = \overline{\mathcal{D}}$. But this is impossible because then

$$\begin{aligned}\mathcal{I}[\mu\alpha.\beta_1 \rightarrow \dots \rightarrow \beta_m \rightarrow \alpha]_\eta &= \overline{\mathcal{D}}^m \Rightarrow \mathcal{I}[\mu\alpha.\beta_1 \rightarrow \dots \rightarrow \beta_m \rightarrow \alpha]_\eta \\ &= \overline{\mathcal{D}}^m \Rightarrow \overline{\mathcal{D}} \\ &\subsetneq \overline{\mathcal{D}}, \quad \text{by (1). } \blacksquare\end{aligned}$$

Completeness

Partial converses to the above soundness results have been proved in [Coppo \[1985\]](#) (see also [Cardone and Coppo \[1991\]](#)) for an interpretation of types in a domain \mathcal{D} of the shape $\mathbb{A} + [\mathcal{D} \rightarrow \mathcal{D}]$. On the one hand, the interpretation in \mathcal{D} of every unsolvable λ -term is \perp , see [B\[1984\]](#), §§8.3, 19.2, for the notion of unsolvable terms and their interpretation in topological λ -models. Now \perp is an element of every complete and uniform subset, 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 by just considering the λ -term $\Delta_3\Delta_3$, where $\Delta_3 \triangleq \lambda x.xxx$, which is unsolvable of order 0, yet has principal type scheme $\mu\alpha.\alpha \rightarrow \alpha$. Therefore the system $\vdash_{\lambda\mu^*}$ is incomplete.

Clearly this cannot be remedied by extending $\vdash_{\lambda\mu^*}$ by the following rule, giving the same types to $(\beta\eta)$ -convertible terms.

$$(Eq) \quad \frac{\Gamma \vdash M : A \quad M =_{\beta\eta} N}{\Gamma \vdash N : A}.$$

One can, however, extend the system $\vdash_{\lambda\mu^*}$ to a complete system in another way. The system introduced in [Coppo \[1985\]](#) 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.

10B.27. DEFINITION (Approximants of λ -terms). Let $\Lambda\perp$ be the set of λ -terms with a new constant \perp .

(i) For $M \in \Lambda$, define inductively the *direct approximant* $\omega(M) \in \Lambda\perp$ of M .

$$\begin{aligned}\omega(x) &\triangleq x, && \text{if } x \text{ is a variable,} \\ \omega(\lambda x.M) &\triangleq \lambda x.\omega(M), && \text{if } \omega(M) \neq \perp, \\ &\triangleq \perp, && \text{otherwise,} \\ \omega(xM_1 \cdots M_k) &\triangleq x\omega(M_1) \cdots \omega(M_k), && \text{for } k > 0, \\ \omega((\lambda x.M)M_1 \cdots M_k) &\triangleq \perp, && \text{for } k > 0.\end{aligned}$$

(ii) A term $P \in \Lambda\perp$ is called an *approximant* of a λ -term M if $P = \omega(N)$ for some $N \in \Lambda$ with $M \rightarrow_\beta N$.

(iii) The set of *approximants* of M , notation $\mathcal{A}(M)$, is defined by

$$\mathcal{A}(M) \triangleq \{P \in \Lambda\perp \mid P \text{ is approximant of } M\}.$$

10B.28. DEFINITION. Let \mathcal{D} be a λ -model.

(i) Elements of $\Lambda\perp$ are interpreted in \mathcal{D} by interpreting \perp as the least element of \mathcal{D} .

- (ii) \mathcal{D} satisfies the *approximation* (AT) if for all $M \in \Lambda$ and term environments ρ one has

$$\llbracket M \rrbracket_\rho = \bigsqcup \{\llbracket P \rrbracket_\rho \mid P \in \mathcal{A}(M)\}. \quad (\text{AT})$$

A classical result is the Approximation Theorem by [Wadsworth \[1976\]](#), stating that the property (AT) holds for Scott's \mathcal{D}_∞ models.

We now introduce the promised complete extension of \vdash_{BH} .

10B.29. DEFINITION. The type inference system $\lambda\mu^{*\infty}$ is defined by adding to the system $\vdash_{\lambda\mu^*}$ the following rules.

$(\perp) \quad \Gamma \vdash \perp : A$ $(C) \quad \frac{\Gamma \vdash P : A \text{ for all } P \in \mathcal{A}(M)}{\Gamma \vdash M : A}$
--

FIGURE 28. Extra axiom and rule for $\lambda\mu^{*\infty}$

For the resulting system $\lambda\mu^{*\infty}$ it is possible to prove a form of completeness.

10B.30. THEOREM. Let \mathcal{D} range over λ -models satisfying AT. Then

$$\Gamma \vdash_{\lambda\mu^{*\infty}} M : A \Leftrightarrow \forall \mathcal{D} [\Gamma \models_{\mathbb{T}_{\mu^*}, \mathcal{D}} M : A]$$

see Definition 10A.2(iv) and Lemma 7E.20(iii).

A proof of this result and a thorough discussion of the system $\lambda\mu^{*\infty}$ is contained in [Cardone and Coppo \[1991\]](#), §4.

More directly relevant to applications is another completeness result, which states that $=_\mu^*$ 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 $=_\mu^*$ as the preferred notion of equality of recursive types. This can be proved following the same idea as in [Coppo \[1985\]](#) and [Cardone and Coppo \[1991\]](#), by using a domain $\mathcal{D} \cong \mathbf{A} + [\mathcal{D} \rightarrow \mathcal{D}]$, where \mathbf{A} is a cpo whose elements are basic values, for example the flat cpo of integers or of boolean values.

The following result on the *equivalence of recursive types* will be obtained as Corollary 11A.39.

10B.31. THEOREM. Let $A, B \in \mathbb{T}_\mu$. Then

$$A =_\mu^* B \Leftrightarrow \forall \eta. [\mathcal{I}[\![A]\!]_\eta = \mathcal{I}[\![B]\!]_\eta]. \blacksquare$$

10C. 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, 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 $\mathcal{M}(A \rightarrow B)$ 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 $\mathcal{D} \cong F(\mathcal{D})$, where F

interprets the functor 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 [Scott \[1972\]](#).

In order to overcome some pathologies that arise in the resulting model, for example the fact that for many natural choices of categories of domains the interpretation of type $\mu\alpha.\alpha \rightarrow \alpha$ 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 [Scott \[1976\]](#). Such a construction yields far more than a model for simple recursive types: indeed, it has been used in [McCracken \[1979\]](#) and [Bruce, Meyer, and Mitchell \[1990\]](#) to interpret the polymorphic λ -calculus and in [Barendregt and Reuzs \[1983\]](#) to give a model for calculi with dependent types and a type of all types.

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 10B. Such models are discussed in many references, among them [Amadio \[1991\]](#), [Cardone \[1989\]](#), [Abadi and Plotkin \[1990\]](#). We shall sketch these model constructions in the exercises, see Exercises 10D.11, 10D.12 and 10D.13. We also mention in passing other models that have been proposed recently in the literature, mostly based on operational interpretations that haven't been dealt with at all in this Part, [Birkedal and Harper \[1999\]](#), [Appel and McAllester \[2001\]](#) and [Vouillon and Melliès \[2004\]](#).

In this section we focus in particular on the system $\lambda_\mu^{\text{Ch}_0}$ in which all possible recursive types are present. The advantage of this system is that the terms $\text{fold}_{\mu\alpha.A}$ and $\text{unfold}_{\mu\alpha.A}$ 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

Solving recursive domain equations

The sketch below summarizes the results that we shall need for the model construction; complete details can be found in [Smyth and Plotkin \[1982\]](#) or [Amadio and Curien \[1998\]](#), §7.1.

10C.1. DEFINITION. We associate to the category **CPO** a category **CPO**^{ep} whose morphisms from \mathcal{D} to \mathcal{E} are the embedding-projection pairs $\langle e : \mathcal{D} \rightarrow \mathcal{E}, p : \mathcal{E} \rightarrow \mathcal{D} \rangle$, namely the pairs of continuous functions such that

$$p \circ e = 1_{\mathcal{D}} \quad \text{and} \quad e \circ p \sqsubseteq 1_{\mathcal{E}}.$$

Observe that, if $\langle e_1, p_1 \rangle : \mathcal{D}_1 \rightarrow \mathcal{E}_1$ and $\langle e_2, p_2 \rangle : \mathcal{D}_2 \rightarrow \mathcal{E}_2$ are embedding-projection pairs, then we have an embedding-projection pair

$$\langle [p_1 \rightarrow e_2], [e_1 \rightarrow p_2] \rangle : [\mathcal{D}_1 \rightarrow \mathcal{D}_2] \rightarrow [\mathcal{E}_1 \rightarrow \mathcal{E}_2],$$

where $[p_1 \rightarrow e_2](h) \triangleq e_2 \circ h \circ p_1$ for all $h : \mathcal{D}_1 \rightarrow \mathcal{D}_2$ and $[e_1 \rightarrow p_2](k) \triangleq p_2 \circ k \circ e_1$ for all $k : \mathcal{E}_1 \rightarrow \mathcal{E}_2$.

In the sequel $F : \mathbf{CPO}^{ep} \rightarrow \mathbf{CPO}^{ep}$ is a functor that is continuous, i.e. preserves direct limits of chains of embeddings of the form (13) below. This F associates to each $\mathcal{D} \in \mathbf{CPO}$ an element $F(\mathcal{D})$ and to each $\langle e, p \rangle : \mathcal{D} \rightarrow \mathcal{E}$ a pair $F(\langle e, p \rangle) : F(\mathcal{D}) \rightarrow F(\mathcal{E})$. Note that e and p determine each other completely. Hence if $F(\langle e, p \rangle) = \langle e', p' \rangle$, then e' is completely determined by e and p' by p . Therefore we write loosely $F(e) = e'$ and $F(p) = p'$.

The solution of an equation $\mathcal{D} \cong F(\mathcal{D})$ will be obtained as the inverse limit of a sequence of projections.

$$(11) \quad \mathcal{D}_0 \xleftarrow{j_0} \mathcal{D}_1 \xleftarrow{j_1} \mathcal{D}_2 \xleftarrow{j_2} \cdots \xleftarrow{j_{n-1}} \mathcal{D}_n \xleftarrow{j_n} \mathcal{D}_{n+1} \xleftarrow{j_{n+2}} \cdots$$

or, equivalently, as the direct limit of the sequence of embeddings

$$(12) \quad \mathcal{D}_0 \xrightarrow{i_0} \mathcal{D}_1 \xrightarrow{i_1} \mathcal{D}_2 \xrightarrow{i_2} \cdots \quad \mathcal{D}_n \xrightarrow{i_n} \mathcal{D}_{n+1} \xrightarrow{i_{n+1}} \cdots,$$

where

$$\begin{aligned} \mathcal{D}_0 &\triangleq \mathbf{1}, & \text{the one-element cpo,} \\ \mathcal{D}_{n+1} &\triangleq F(\mathcal{D}_n) \\ \langle i_0, j_0 \rangle &\triangleq \langle \perp_{\mathcal{D}_0 \rightarrow \mathcal{D}_1}, \perp_{\mathcal{D}_1 \rightarrow \mathcal{D}_0} \rangle. \end{aligned}$$

Compare this with Section 16C, where the case $F(\mathcal{D}) \cong [\mathcal{D} \rightarrow \mathcal{D}]$ is treated, starting from a general \mathcal{D}_0 . For the present \mathcal{D}_0 the pair $\langle i_0, j_0 \rangle$ is the only morphism from \mathcal{D}_0 to \mathcal{D}_1 and the pairs $\langle i_n, j_n \rangle$, for $n > 0$, are embedding-projection pairs from \mathcal{D}_n to \mathcal{D}_{n+1} and are defined inductively by

$$\begin{aligned} i_n &\triangleq F(i_{n-1}) : F(\mathcal{D}_{n-1}) \longrightarrow F(\mathcal{D}_n) \\ j_n &\triangleq F(j_{n-1}) : F(\mathcal{D}_n) \longrightarrow F(\mathcal{D}_{n-1}). \end{aligned}$$

The direct limit of the sequence (12) can be described explicitly as

$$\mathcal{D}_\infty \triangleq \lim_{\longrightarrow} \langle \mathcal{D}_n, i_n \rangle = \{ \langle d_n \rangle_{n \in \omega} \mid \forall n \in \omega. d_n \in \mathcal{D}_n \text{ and } j_{n+1}(d_{n+1}) = d_n \}.$$

with embeddings $i_{n\infty} : \mathcal{D}_n \rightarrow \mathcal{D}_\infty$ and projections $j_{\infty n} : \mathcal{D}_\infty \rightarrow \mathcal{D}_n$, for all $n \in \omega$, making all the diagrams

$$\begin{array}{ccc} \mathcal{D}_n & \xrightarrow{i_n} & \mathcal{D}_{n+1} \\ & \searrow i_{n\infty} & \swarrow i_{(n+1)\infty} \\ & \lim_{\longrightarrow} \langle \mathcal{D}_n, i_n \rangle & \end{array}$$

commute. From the continuity of F and universality we obtain:

10C.2. PROPOSITION. *There is a unique isomorphism $\vartheta : F(\mathcal{D}_\infty) \rightarrow \mathcal{D}_\infty$, given by*

$$\bigsqcup_{n \in \omega} (i_{(n+1)\infty} \circ F(j_{\infty n}))$$

whose inverse ϑ^{-1} is

$$\bigsqcup_{n \in \omega} (F(i_{n\infty}) \circ j_{\infty(n+1)}).$$

Note that the suprema do exist indeed, because $\langle i_{(n+1)\infty} \circ F(j_{\infty n}) \rangle$ and $\langle F(i_{n\infty}) \circ j_{\infty(n+1)} \rangle$ are chains. This follows from the fact that the $\langle F(i_n), F(j_n) \rangle$ are morphisms in \mathbf{CPO}^{ep} , hence $F(i_n) \circ F(j_n) \sqsubseteq \text{Id}_{F(\mathcal{D}_{n+1})}$.

For a proof, we refer the reader to Plotkin [1982], Chapter 5. The construction is based on the observation that all the following diagrams (in \mathbf{CPO}^{ep}) commute:

$$(13) \quad \begin{array}{ccc} F(\mathcal{D}_n) & \xrightarrow{F(i_n)} & F(\mathcal{D}_{n+1}); \\ & \searrow F(i_{n\infty}) & \swarrow F(i_{(n+1)\infty}) \\ & F(\lim \overrightarrow{\langle \mathcal{D}_n, i_n \rangle}) & \\ i_{(n+1)\infty} & \downarrow \vartheta & i_{(n+2)\infty} \\ & \lim \overrightarrow{\langle \mathcal{D}_n, i_n \rangle} & \end{array}$$

$$(14) \quad \begin{array}{ccc} \mathcal{D}_n & \xrightarrow{i_n} & \mathcal{D}_{n+1}. \\ & \searrow i_{n\infty} & \swarrow i_{(n+1)\infty} \\ & \lim \overrightarrow{\langle \mathcal{D}_n, i_n \rangle} & \\ F(i_{(n-1)\infty}) & \downarrow \vartheta^{-1} & F(i_{n\infty}) \\ & F(\lim \overrightarrow{\langle \mathcal{D}_n, i_n \rangle}) & \end{array}$$

Interpreting μ -types as domains

With the machinery set up for the solution of recursive domain equations we can now easily define a typed applicative structure, see Definition 3A.1, by interpreting each $A \in \mathbb{T}_\mu = \mathbb{T}_\mu^{\mathbb{A}}$ as $\mathcal{M}^{\mathbf{CPO}}(A) \in \mathbf{CPO}$, by induction on A . If η maps \mathbb{A} to the objects of \mathbf{CPO} , then it can be extended to an interpretation of \mathbb{T}_μ as follows.

$$\begin{aligned} \mathcal{M}_\eta^{\mathbf{CPO}}(\alpha) &\triangleq \eta(\alpha); \\ \mathcal{M}_\eta^{\mathbf{CPO}}(A \rightarrow B) &\triangleq [\mathcal{M}_\eta^{\mathbf{CPO}}(A) \rightarrow \mathcal{M}_\eta^{\mathbf{CPO}}(B)]; \\ \mathcal{M}_\eta^{\mathbf{CPO}}(\mu\alpha.A) &\triangleq \lim_{\rightarrow} \langle F^n(\mathbf{1}), i_n \rangle, \quad \text{where } F(\mathcal{D}) \triangleq \mathcal{M}_{\eta[\alpha \rightarrow \mathcal{D}]}^{\mathbf{CPO}}(A). \end{aligned}$$

The above clauses, especially the last one, are justified by the theory developed in Lehmann and Smyth [1981, §4].

For morphisms $\langle e, p \rangle : \mathcal{D} \rightarrow \mathcal{E}$ we give the definition of $F(\langle e, p \rangle)$ by a few examples.

- If $A = \beta$, then $F(\mathcal{D}) = F(\mathcal{E}) = \eta(\beta)$ and $F(\langle e, p \rangle) = \langle \mathbf{1}, \mathbf{1} \rangle$.
- If $A = \alpha$, then $F(\mathcal{D}) = \mathcal{D}$, $F(\mathcal{E}) = \mathcal{E}$ and $F(\langle e, p \rangle) = \langle e, p \rangle$.
- If $A = \alpha \rightarrow \beta$, then $F(\mathcal{D}) = [\mathcal{D} \rightarrow \eta(\beta)]$, $F(\mathcal{E}) = [\mathcal{E} \rightarrow \eta(\beta)]$, where $[\mathcal{D} \rightarrow \mathcal{E}]$ is the CPO of continuous functions from \mathcal{D} to \mathcal{E} , see Definition 10A.7, and $F(\langle e, p \rangle) = \langle [p \rightarrow \mathbf{1}], [e \rightarrow \mathbf{1}] \rangle$.

Summarizing we have the following result.

10C.3. THEOREM. \mathcal{M}^{CPO} is an applicative type structure.

We can extend in a straightforward way Definition 3A.9 and define an interpretation of every term $M \in \Lambda_{\mathbb{T}_\mu}^{\text{Ch}}(A)$. In particular, define

$$\begin{aligned} \text{fold}_{\mu\alpha.A} &\triangleq \theta : F(\mathcal{M}_\eta^{\text{CPO}}(\mu\alpha.A)) \longrightarrow \mathcal{M}_\eta^{\text{CPO}}(\mu\alpha.A) \\ \text{unfold}_{\mu\alpha.A} &\triangleq \theta^{-1} : \mathcal{M}_\eta^{\text{CPO}}(\mu\alpha.A) \longrightarrow F(\mathcal{M}_\eta^{\text{CPO}}(\mu\alpha.A)) \end{aligned}$$

where $F(\mathcal{D})$ is defined, as above, as $F(\mathcal{D}) \triangleq \mathcal{M}_{\eta[\alpha \mapsto \mathcal{D}]}^{\text{CPO}}(A)$, and θ and θ^{-1} are as in diagrams (13) and (14), respectively. It is immediate to verify that this interpretation is well-defined, and not trivial.

10C.4. THEOREM. \mathcal{M}^{CPO} is a typed λ -model, that models the coercions and their reductions.

The following result is a typical application of the interpretation of terms of $\lambda_\mu^{\text{Ch}_0}$ outlined above. It relates the interpretation of the typed fixed-point combinator \mathbf{Y}_μ^A of Exercise 7G.15 to the iterative construction of fixed-points of continuous endofunctions of a CPO \mathcal{D} (see Theorem 10A.11). It is similar to a well-known result of [Park \[1976\]](#) on the interpretation of the \mathbf{Y} combinator in Scott's \mathcal{D}_∞ models, and was stated for the typed case in [Cosmadakis \[1989\]](#).

10C.5. THEOREM. The interpretation of \mathbf{Y}_μ^A as a term of type $(A \rightarrow A) \rightarrow A$ coincides, for any environment η , with the functional $\mathbf{fix} \in [\mathcal{M}_\eta^{\text{CPO}}(A) \rightarrow \mathcal{M}_\eta^{\text{CPO}}(A)] \rightarrow \mathcal{M}_\eta^{\text{CPO}}(A)$.

PROOF. A straightforward analysis of the interpretation of recursive types in the category **CPO**, whose details are left as an exercise. ■

As $\mathbb{T}_\mu^{\mathbb{A}}$ is an enrichment of the set of simple types \mathbb{T} , the interpretation of $\mathbb{T}_\mu^{\mathbb{A}}$ yields a **CPO** model of simple types. Observe that the interpretation in \mathcal{M}^{CPO} of types like, e.g., $\mu\alpha.\alpha \rightarrow \alpha$ is trivial: its interpretation in \mathcal{M}^{CPO} has only one element. This semantical phenomenon suggests immediately that, at the operational level, terms of trivial type should be observationally equivalent. Programming languages generally possess a notion of *observable type*, for example integers or booleans. Two closed terms M, N of the same type A are *observationally equivalent* if, for any context $C[\]$ bringing a term of type A into an observable type $\alpha \in \mathbb{A}$,

$$C[M] \text{ has a value iff } C[N] \text{ has the same value.}$$

A *trivial type* is a type such that any two inhabitants are observationally equivalent. Somewhat surprisingly, we shall see that there are trivial types in \mathbb{T}_μ . Indeed we can state for trivial types a result similar to the Genericity Lemma for the pure type-free λ -calculus [B\[1984\]](#), Proposition 14.3.24.

10C.6. DEFINITION (Trivial types). A type T is *trivial* if

- (i) either $T =_\mu \mu\alpha_1 \dots \mu\alpha_k.A_1 \rightarrow \dots \rightarrow A_n \rightarrow \alpha_i$ ($k \geq 1, n \geq 0, 1 \leq i \leq k$),
- (ii) or $T =_\mu S \rightarrow T'$ where T' is trivial.

10C.7. LEMMA. let A be a trivial type. Then $\mathcal{M}^{\text{CPO}}(A) \cong \mathbf{1}$.

PROOF. By induction on the structure of types, observing that $\mathcal{M}^{\text{CPO}}(A) \cong \mathbf{1}$ and $A =_\mu A'$ entail that $\mathcal{M}^{\text{CPO}}(A') \cong \mathbf{1}$ by a straightforward induction on the proof that $A =_\mu A'$. $A \equiv \mu\alpha_1 \dots \mu\alpha_k.A_1 \rightarrow \dots \rightarrow A_n \rightarrow \alpha_i$. Then $A =_\mu \mu\alpha_i.A'_1 \rightarrow \dots \rightarrow A'_n \rightarrow \alpha_i$,

and $\mathcal{M}^{\text{CPO}}(\mu\alpha_i.A'_1 \rightarrow \dots \rightarrow A'_n \rightarrow \alpha_i) \cong \mathbf{1}$ because in **CPO** $[\mathcal{D} \rightarrow \mathbf{1}] \cong \mathbf{1}$. Therefore $\mathcal{M}^{\text{CPO}}(A) \cong \mathbf{1}$. $A \equiv S \rightarrow T$, with $\mathcal{M}^{\text{CPO}}(T) \cong \mathbf{1}$ by induction hypothesis. Then also $\mathcal{M}^{\text{CPO}}(S \rightarrow T) \cong \mathbf{1}$. ■

Given a set \mathcal{T} of equations between closed terms of the same type, we write $\mathcal{T} \vdash M = N$ if and only if the equation $M = N$ follows from (the typed version of) β -conversion using the equations in \mathcal{T} as further axioms: this is called the *typed λ -theory generated by \mathcal{T}* . A theory with a set \mathcal{T} of equations is *consistent* if there is a type A and terms M, N of type A such that $\mathcal{T} \not\vdash M = N$. We can now state an easy result that shows that it is consistent to identify terms with the same trivial type.

10C.8. COROLLARY. *The following theory \mathcal{T} is consistent.*

$$\mathcal{T} \triangleq \{M = N \mid M, N \in \Lambda_{\mathbb{T}_\mu}^{\text{Ch}}(B) \text{ and } B \text{ is trivial}\}.$$

PROOF. Observe that by Lemma 10C.7 one has $\mathcal{M}^{\text{CPO}} \models \mathcal{T}$. Let $A \triangleq \alpha \rightarrow \alpha \rightarrow \alpha$ and $M \triangleq \lambda x^\alpha. \lambda y^\alpha.x$, $N \triangleq \lambda x^\alpha. \lambda y^\alpha.y$. Let $\eta(\alpha)$ be a non-trivial CPO. Then $\llbracket M \rrbracket \neq \llbracket N \rrbracket$ in $\eta(\alpha) \Rightarrow \eta(\alpha) \Rightarrow \eta(\alpha)$. Then $\mathcal{M}^{\text{CPO}} \not\models M = N$, hence $\mathcal{T} \not\vdash M = N$. Therefore \mathcal{T} is consistent. ■

In the following it will be often useful to decorate subterms with their types. Let us say that a subterm N^T of a term M , where T is a trivial type, is *maximally trivial* if it is not a subterm of another subterm of M of trivial type. If M is any typed term let \overline{M} be the term obtained by replacing all maximal trivial subterms of M of type T by Ω^T (as shown in Example 7A.3 there are typed versions of Ω at all types).

10C.9. LEMMA. *Let $M \rightarrow_\beta M'$. Then $\overline{M} \rightarrow_{\overline{\beta}} \overline{M}'$, where $\rightarrow_{\overline{\beta}}$ denotes one or zero reduction steps.*

PROOF. A straightforward structural induction on M . The interesting case is $M^A \equiv ((\lambda x^T.P^A)^{T \rightarrow A}Q^T)^A \rightarrow_\beta P[x^T := Q^T]^A$, with T trivial and A not trivial, so that Q^T is maximally trivial in M . We immediately have $((\lambda x^T.\overline{P^A})^{T \rightarrow A}\overline{Q^T})^A \rightarrow_\beta \overline{P}[x^T := \overline{Q^T}]^A$, observing that both x and Q are of the same trivial type T . ■

A type $S \in \mathbb{T}_\mu$ is *hereditarily non-trivial* if no trivial type is a subexpression of S . Say that an occurrence of a subterm P of a normalizable term M is *useless* if M has the same β -normal form whenever P is replaced by any other term of the same type.

10C.10. LEMMA. *Let S be hereditarily non-trivial. For any closed normalizable term $M \in \Lambda_{\mathbb{T}_\mu}^{\text{Ch}}(S)$, any occurrence in M of a subterm P of trivial type T is useless.*

PROOF. Note that if M is in normal form (i.e. $M \equiv \lambda \vec{x}. x_i M_1 \cdots M_m$) the types of all its subterms are hereditarily non-trivial. In fact, $S \equiv S_1 \rightarrow S_2 \rightarrow \dots \rightarrow S_n \rightarrow B$, and $S_i \equiv B_1^i \rightarrow B_2^i \rightarrow \dots \rightarrow B_m^i \rightarrow B$, so B_1^i, \dots, B_m^i, B are hereditarily non-trivial because subexpressions of a hereditarily non-trivial type. Therefore, the types of x_i, M_1, \dots, M_m are hereditarily non-trivial and this holds, recursively, for the subterms of M_1, \dots, M_m . Then it is impossible for a normal form to have a subterm of trivial type.

If M is not in normal form let M' be a term obtained from M by replacing a subterm of a trivial type T by any other subterm of the same type. Note that $\overline{M} = \overline{M}'$ and apply Lemma 10C.9. ■

10D. Exercises

10D.1. Let \mathcal{D} be a CPO.

- (i) Show that a subset $O \subseteq \mathcal{D}$ is closed iff it is closed downwards and closed under sups of directed sets.
- (ii) Show that continuous functions are monotonic.

10D.2. Let \mathcal{D} be a complete lattice.

- (i) Show that \mathcal{D} has a *top* element \top such that $\forall d \in \mathcal{D}. d \sqsubseteq \top$.
- (ii) Show that \mathcal{D} has arbitrary *infima* and the *sup* and *inf* of two elements.
- (iii) Show that $\perp_{\mathcal{D}}$ is compact and that $d, e \in \mathcal{K}(\mathcal{D}) \Rightarrow d \sqcup e \in \mathcal{K}(\mathcal{D})$.
- (iv) In general it is not true that if $d \sqsubseteq e \in \mathcal{K}(\mathcal{D})$, then $d \in \mathcal{K}(\mathcal{D})$. [Hint. Take $\omega + 1$ in the ordinal $\omega + \omega = \{0, 1, 2, \dots, \omega, \omega + 1, \omega + 2, \dots\}$. It is compact, but ω ($\sqsubseteq \omega + 1$) is not.]

10D.3. Let \mathcal{D} be a complete lattice.

- (i) Show that every monotonic $f : \mathcal{D} \rightarrow \mathcal{D}$ has a unique least fixed point. [Hint. Take $x = \bigcap \{z \mid f(z) \sqsubseteq z\}$.]
- (ii) In case f is also continuous, show that the least fixed point is $\text{fix}(f)$.

10D.4. Fill in the details of Theorem 10A.11.

10D.5. Let A be a non-empty set. Define Engeler's model \mathcal{D}_A as follows. \mathcal{D}_A has as underlying set $\mathcal{P}(\mathcal{D}^*)$, where $\mathcal{D}^* = \bigcup_{n \in \omega} \mathcal{D}_n$ and

$$\begin{aligned}\mathcal{D}_0 &\triangleq A \\ \mathcal{D}_1 &\triangleq A \\ \mathcal{D}_{n+2} &\triangleq \mathcal{D}_{n+1} \cup \{(\beta, m) \mid m \in \mathcal{D}_{n+1}, \beta \subseteq_{\text{fin}} \mathcal{D}_{n+1}\}.\end{aligned}$$

Thus, \mathcal{D}^* is the least set X containing A such that if β is a finite subset of X and $m \in X$, then $(\beta, m) \in X$. The complete lattice $\mathcal{D}_A = \langle \mathcal{D}_A, \subseteq \rangle$ can be turned into a reflexive structure by defining, for $d, e \in \mathcal{D}_A$

$$d \cdot e \triangleq \{m \in \mathcal{D}^* \mid \exists \beta \subseteq_{\text{fin}} e. (\beta, m) \in d\}.$$

Show that $d_n \triangleq d \cap \mathcal{D}_n$ defines a notion of approximation for \mathcal{D}_A .

10D.6. Prove Lemma 10B.12.

10D.7. Prove Lemma 10B.16(ii). [Hint. Prove by induction that

$$\forall n \in \omega. \mathcal{I}^n \llbracket A[\alpha := B] \rrbracket_{\eta} = \mathcal{I}^n \llbracket A \rrbracket_{\eta[\alpha \mapsto \mathcal{I} \llbracket B \rrbracket_{\eta}]}.$$

10D.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 10B.

[Hint. Let $\hat{A} \triangleq A[\alpha := \mu\alpha.B]$ and $\hat{B} \triangleq B[\beta := \mu\beta.A]$. Then prove by simultaneous induction on $n \in \omega$ that the following two equations hold.

- (i) $\mathcal{I}^n \llbracket \mu\beta.\hat{A} \rrbracket_{\eta} = \mathcal{I}^n \llbracket \mu\beta.A[\alpha := \mu\alpha.\hat{B}] \rrbracket_{\eta}$,
- (ii) $\mathcal{I}^n \llbracket \mu\alpha.\hat{B} \rrbracket_{\eta} = \mathcal{I}^n \llbracket \mu\alpha.B[\beta := \mu\beta.\hat{A}] \rrbracket_{\eta}$.

10D.9. (Metric interpretation of contractive recursive types) For $X, Y \in \mathcal{CU}$, let

$$\delta(X, Y) \triangleq \begin{cases} \max\{n \mid X_n = Y_n\}, & \text{if } X \neq Y, \\ \infty, & \text{otherwise.} \end{cases}$$

Define a function $d : \mathcal{CU} \times \mathcal{CU} \rightarrow \mathbf{R}^+$ by setting

$$d(X, Y) \triangleq \begin{cases} 2^{-\delta(X, Y)}, & \text{if } X \neq Y; \\ 0, & \text{otherwise.} \end{cases}$$

Show the following, see Theorem 7F.5.

- (1) $\langle \mathcal{CU}, d \rangle$ is a complete ultrametric space.
- (2) The functions $\lambda X \in \mathcal{CU}. X \Rightarrow Y$ for a fixed $Y \in \mathcal{CU}$, $\lambda Y \in \mathcal{CU}. X \Rightarrow Y$ for a fixed $X \in \mathcal{CU}$ and $\lambda X \in \mathcal{CU}. X \Rightarrow X$ are contractive, where the function $\Rightarrow : \mathcal{CU} \times \mathcal{CU} \rightarrow \mathcal{CU}$ is defined in Definition 10A.1.
- (3) For $\mu\alpha.A \in \mathbb{T}_\mu$ define

$$\llbracket \mu\alpha.A \rrbracket_\eta \triangleq \text{fix}(\lambda X \in \mathcal{CU}. \llbracket A \rrbracket_{\eta[\alpha \mapsto X]}).$$

Use these facts to give an interpretation of all $A \in \mathbb{T}_\mu$. [Hint. See MacQueen, Plotkin, and Sethi [1986], Amadio [1991].]

10D.10. Show that $\vdash_{\lambda\mu^*} \lambda y.y((\lambda x.x x)(\lambda x.x x)) : \mu\alpha.T_\alpha \rightarrow \alpha$ where $T_\alpha \triangleq \mu\beta.\beta \rightarrow (\beta \rightarrow \alpha)$. [Recall that $(\lambda x.x x)(\lambda x.x x)$ has all types].

10D.11. A *partial equivalence relation (per)* over \mathcal{D} is any symmetric and transitive relation $R \subseteq \mathcal{D} \times \mathcal{D}$.

Let \mathcal{D} be a λ -model with a notion of approximation and R be a per over \mathcal{D} . Let $R_n \triangleq \{\langle d_n, e_n \rangle \mid \langle d, e \rangle \in R\}$. We say that R is complete and uniform over \mathcal{D} if we have

- $R_0 = \{\langle d, d \rangle \mid d \in \mathcal{D}_0\}$;
- (Completeness) if $\{\langle d^{(1,j)}, d^{(2,j)} \rangle \mid j \in \omega\}$ is an increasing chain such that

$$\left(\forall j \in \mathbb{N}. \langle d^{(1,j)}, d^{(2,j)} \rangle \in R \right), \text{ then } \langle \bigsqcup_{j \in \mathbb{N}} d^{(1,j)}, \bigsqcup_{j \in \mathbb{N}} d^{(2,j)} \rangle \in R;$$

- (Uniformity) $\langle d^1, d^2 \rangle \in R$ implies $\langle d_n^1, d_n^2 \rangle \in R$ for all $n \in \mathbb{N}$.

Generalize Definition 10B.11 to the case where types are interpreted as complete and uniform partial equivalence relations over a λ -model \mathcal{D} with a notion of approximation.

10D.12. Use the interpretation of types as complete and uniform partial equivalence relations of Exercise 10D.11 to build a model of λ_μ^{Ch} . [Hint. The main steps are as follows.

- (i) Prove the following Fundamental Lemma. If $\Gamma \vdash_{\lambda_\mu^{\text{Ch}}} M : A$, η is a type environment and ρ_1, ρ_2 are term environments such that $\langle \rho_1(x_i), \rho_2(x_i) \rangle \in \mathcal{I}[\llbracket A_i \rrbracket_\eta]$ whenever $x_i : A_i \in \Gamma$, then

$$\langle \llbracket M \rrbracket_{\rho_1}, \llbracket M \rrbracket_{\rho_2} \rangle \in \mathcal{I}[\llbracket A \rrbracket_\eta].$$

- (ii) Define the model $\mathcal{M} = \langle T, \{\mathcal{M}(R)\}_{R \in T}, \{\Phi_{RS}\}_{R, S \in T} \rangle$ as follows. T is the set of all complete and uniform per's over \mathcal{D} . For $R, S \in T$ define

$$(R \Rightarrow S) \triangleq \{\langle d_1, d_2 \rangle \mid \forall \langle e_1, e_2 \rangle \in R. \langle d_1 \cdot e_1, d_2 \cdot e_2 \rangle \in S\};$$

$$[d]_R \triangleq \{e \in \mathcal{D} \mid \langle d, e \rangle \in R\}$$

$$\mathcal{M}(R) \triangleq \{[d]_R \mid \langle d, d \rangle \in R\};$$

$$\Phi_{RS}([d]_{R \Rightarrow S})([e]_R) \triangleq [d \cdot e]_S,$$

where $[d]_{R \Rightarrow S} \in \mathcal{M}(R \Rightarrow S)$ and $[e]_R \in \mathcal{M}(R)$. Show that the above construction is well-defined.]

10D.13. Explore the consequences of choosing other definitions of R_0 (e.g., $R_0 = \mathcal{D}_0 \times \mathcal{D}_0$ or $R_0 = \{\langle \perp_{\mathcal{D}}, \perp_{\mathcal{D}} \rangle\}$).

10D.14. (i) In this exercise, we sketch how to build a model \mathcal{M} for the system $\lambda_{\mu}^{\text{Ch}_0}$ where types are interpreted as *closures* over the λ -model $\mathcal{P}\omega$. The construction can be adapted easily to systems of the form $\lambda^{\mathcal{A}-\text{Ch}}$, 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 [Scott \[1976\]](#), §5, and [B\[1984\]](#), §18.1, Exercises 18.4.3–18.4.9. For a related construction see [Bruce, Meyer, and Mitchell \[1990\]](#), §7.2. Define a *closure* (over $\mathcal{P}\omega$) as a continuous function $a : \mathcal{P}\omega \rightarrow \mathcal{P}\omega$ 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 $\text{im}(a)$ coincides with the set $\{x \in \mathcal{P}\omega \mid a(x) = x\}$. It can easily be proved that $\text{im}(a)$ is an ω -algebraic lattice. Moreover, [Scott \[1976\]](#), Theorem 5.2, proves that, if \mathcal{D} is an algebraic lattice with a countable basis, then there is a closure $a_{\mathcal{D}}$ over $\mathcal{P}\omega$ such that $\mathcal{D} \cong \text{im}(a_{\mathcal{D}})$. 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 closure $a \rightsquigarrow b$, defined by $(a \rightsquigarrow b)(x) = b \circ x \circ a$, with a continuous bijection

$$\Phi_{ab} : \text{im}(a \rightsquigarrow b) \xrightarrow{\sim} [\text{im}(a) \rightarrow \text{im}(b)].$$

(ii) Define the mapping $\mathcal{I}[\![\cdot]\!]_{\eta} : \mathbb{T}_{\mu} \rightarrow \mathcal{V}$ as follows, where η is a type environment assigning to each atom a closure.

- (a) $\mathcal{I}[\![\alpha]\!]_{\eta} \triangleq \eta(\alpha);$
- (b) $\mathcal{I}[\![A \rightarrow B]\!]_{\eta} \triangleq \mathcal{I}[\![A]\!]_{\eta} \rightsquigarrow \mathcal{I}[\![B]\!]_{\eta};$
- (c) $\mathcal{I}[\![\mu\alpha.A]\!]_{\eta} \triangleq \text{fix}(\lambda a \in \mathcal{V}. \mathcal{I}[\![A]\!]_{\eta[\alpha \mapsto a]}).$

Show that this mapping is well-defined.

(iii) Let $A, B \in \mathbb{T}_{\mu}$. Show that if $A =_{\mu} B$ in the formal system of Definition 7D.26(i), then $\mathcal{I}[\![A]\!]_{\eta} = \mathcal{I}[\![B]\!]_{\eta}$, for any type environment η .

(iv) For any $a \in \mathcal{V}$, define $\mathcal{M}(a)$ as $\text{im}(a)$. We have isomorphisms

$$\Phi_{ab} : \mathcal{M}(a \rightsquigarrow b) \longrightarrow [\mathcal{M}(a) \rightarrow \mathcal{M}(b)]$$

for any pair $a, b \in \mathcal{V}$ and (trivial) isomorphisms

$$\Psi_{ab} : \mathcal{M}(a) \rightarrow \mathcal{M}(b)$$

whenever $a = b$. Define \mathcal{M} as a structure

$$\mathcal{M} \triangleq \langle \mathcal{V}, \{\mathcal{M}(a)\}_{a \in \mathcal{V}}, \{\Phi_{ab}\}_{a, b \in \mathcal{V}}, \{\Psi_{ab}\}_{a, b \in \mathcal{V}} \rangle.$$

Show that \mathcal{M} is a typed λ -model of $\lambda_{\mu}^{\text{Ch}_0}$.

10D.15. Let Γ be a basis. A Γ, η -environment is a function ρ such that $\rho(x) \in \mathcal{M}(\mathcal{I}[\![A]\!]_{\eta})$, whenever $(x:A) \in \Gamma$. Define the interpretation of a term (w.r.t. a Γ, η -environment)

by induction on its construction tree in the system $\lambda_\mu^{\text{Ch}_0}$:

$$\begin{aligned} \llbracket \Gamma, x:A \vdash_{\lambda_\mu^{\text{Ch}}} x : A \rrbracket_{\eta\rho} &\triangleq \rho(x), \\ \llbracket \Gamma \vdash_{\lambda_\mu^{\text{Ch}}} MN : B \rrbracket_{\eta\rho} &\triangleq \Phi_{ab} \left(\llbracket \Gamma \vdash_{\lambda_\mu^{\text{Ch}}} M : A \rightarrow B \rrbracket_{\eta\rho} \right) \llbracket \Gamma \vdash_{\lambda_\mu^{\text{Ch}}} N : A \rrbracket_{\eta\rho}, \\ \llbracket \Gamma \vdash_{\lambda_\mu^{\text{Ch}}} (\lambda x:A.M) : A \rightarrow B \rrbracket_{\eta\rho} &\triangleq \Phi_{ab}^{-1}(f), \end{aligned}$$

where f is the function defined by the assignment:

$$f(d) \triangleq \llbracket \Gamma, x:A \vdash_{\lambda_\mu^{\text{Ch}}} M : B \rrbracket_{\eta\rho[x \mapsto d]}$$

for any $d \in \mathcal{M}(\mathcal{I}\llbracket A \rrbracket_\eta)$. Now $f \in [\mathcal{M}(a) \rightarrow \mathcal{M}(b)]$, for $a = \mathcal{I}\llbracket A \rrbracket_\eta$ and $b = \mathcal{I}\llbracket B \rrbracket_\eta$.

Let $a \triangleq \mathcal{I}\llbracket \mu\alpha.A \rrbracket_\eta$ and $b \triangleq \mathcal{I}\llbracket A[\alpha := \mu\alpha.A] \rrbracket_\eta$, then we have

$$\begin{aligned} \llbracket \Gamma \vdash_{\lambda_\mu^{\text{Ch}}} \text{fold}_{\mu\alpha.A}(M) : \mu\alpha.A \rrbracket_{\eta\rho} &= \Psi_{ab}^{-1} \left(\llbracket \Gamma \vdash_{\lambda_\mu^{\text{Ch}}} M : A[\alpha := \mu\alpha.A] \rrbracket_{\eta\rho} \right) \\ \llbracket \Gamma \vdash_{\lambda_\mu^{\text{Ch}}} \text{unfold}_{\mu\alpha.A}(M) : A[\alpha := \mu\alpha.A] \rrbracket_{\eta\rho} &= \Psi_{ab} \left(\llbracket \Gamma \vdash_{\lambda_\mu^{\text{Ch}}} M : \mu\alpha.A \rrbracket_{\eta\rho} \right). \end{aligned}$$

Show that this yields a typed λ -model of $\lambda_\mu^{\text{Ch}_0}$.

10D.16. Define a partial order on $\text{Tr}_{\text{inf}}^{\mathbb{A}}$, exploiting the “undefined” tree \bullet , as follows. For $s, t \in \text{Tr}_{\text{inf}}^{\mathbb{A}}$

$$s \preccurlyeq t \Leftrightarrow \text{dom}(s) \subseteq \text{dom}(t), \quad \text{and}$$

$$\forall w \in \text{dom}(s). s(w) \neq \bullet \quad \Rightarrow \quad s(w) = t(w)$$

- (i) Prove that the partially ordered set $\langle \text{Tr}_{\text{inf}}^{\mathbb{A}}, \preccurlyeq \rangle$ is ω -complete and has \bullet as least element.
- (ii) Prove that $\text{Tr}_{\text{inf}}^{\mathbb{A}}$ is the initial ω -complete Σ -algebra, where $\Sigma = \mathbb{A} \cup \{\rightarrow\} \cup \{\bullet\}$.
- (iii) Conclude that for every $\eta : \mathbb{A} \rightarrow \mathcal{V}$, there is a unique continuous Σ -morphism extending η

$$\mathcal{T}\llbracket \cdot \rrbracket_\eta : \text{Tr}_{\text{inf}}^{\mathbb{A}} \rightarrow \mathcal{V},$$

which satisfies the following equations.

$$\begin{aligned} \mathcal{T}\llbracket \alpha \rrbracket_\eta &= \eta(\alpha) \text{ for any } \alpha \in \mathbb{A}; \\ \mathcal{T}\llbracket \bullet \rrbracket_\eta &= \mathbf{I}; \\ \mathcal{T}\llbracket t_1 \rightarrow t_2 \rrbracket_\eta &= \mathcal{T}\llbracket t_1 \rrbracket_\eta \rightsquigarrow \mathcal{T}\llbracket t_2 \rrbracket_\eta. \end{aligned}$$

for all $t_1, t_2 \in \text{Tr}_{\text{inf}}^{\mathbb{A}}$. [Hint. See e.g. [Goguen, Thatcher, Wagner, and Wright \[1977\]](#), Theorem 4.8, or [Courcelle \[1983\]](#).]

- (iv) Prove that for all types $A, B \in \mathbb{T}_\mu^{\mathbb{A}}$ and all type environments η one has

$$\begin{aligned} \text{(i)} \quad \mathcal{T}\llbracket A^* \rrbracket_\eta &= \mathcal{I}\llbracket A \rrbracket_\eta. \\ \text{(ii)} \quad \mathcal{I}\llbracket A \rrbracket_\eta &= \mathcal{I}\llbracket B \rrbracket_\eta, \quad \text{whenever } A =_\mu^* B. \end{aligned}$$

[Hint. By structural induction on $A \in \mathbb{T}_\mu^{\mathbb{A}}$. Show first that for any pair of infinite trees $s, t \in \text{Tr}_{\text{inf}}^{\mathbb{A}}$, we have

$$\mathcal{T}\llbracket s[\alpha := t] \rrbracket_\eta = \mathcal{T}\llbracket s \rrbracket_{\eta[\alpha \mapsto \mathcal{T}\llbracket t \rrbracket_\eta]}$$

and use the fact that $\mathcal{I}[\mu\alpha.A_1]_\eta = \bigsqcup_{n \in \omega} \Theta^{(n)}(\perp)$, where $\Theta^{(0)}(\perp) = \perp$ and $\Theta^{(n+1)}(\perp) = \mathcal{I}[A_1]_{\eta[\alpha \mapsto \Theta^{(n)}(\perp)]}$. See [Cardone \[2002\]](#), where also the converse of this soundness result is proved: for $A, B \in \mathbb{T}_\mu^{\mathbb{A}}$ one has

$$A =_\mu^* B \Leftrightarrow \mathcal{I}[A]_\eta = \mathcal{I}[B]_\eta, \text{ for all } \eta : \mathbb{A} \rightarrow \mathcal{V}.$$

10D.17. Given 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 $i = 1, \dots, n$, show how to construct type algebra morphisms h, k , see Definitions 7C.13(i) and 7E.17(ii), with

$$h : \mathbb{T}[\mathcal{R}] \longrightarrow \mathcal{S}(\mathcal{D})$$

$$k : \mathbb{T}[\mathcal{R}]^* \longrightarrow \mathcal{S}(\mathcal{D}).$$

[Hint. Adapt the technique used in Section 10B. Alternatively, observe that a solution $\vec{\mathbf{X}} = \mathbf{X}_1, \dots, \mathbf{X}_n$ of $\mathcal{R}(\vec{X})$ consists of regular trees so, for each $i = 1, \dots, n$, there is $A_{\mathbf{X}_i} \in \mathbb{T}_\mu^{\mathbb{A}}$ such that $(A_{\mathbf{X}_i})^* = \mathbf{X}_i$. Let η be the type environment defined by the mapping $X_i \mapsto \mathcal{I}[A_{\mathbf{X}_i}]$. Consider the following clauses.

$$\begin{aligned} \mathcal{J}[X_i]_\eta &= \eta(X_i) \\ \mathcal{J}[A' \rightarrow A'']_\eta &= \mathcal{J}[A']_\eta \Rightarrow \mathcal{J}[A'']_\eta \end{aligned}$$

They define inductively the extension of η to the required type algebra morphisms.]

10D.18. A function $f : \mathcal{CU}(\mathcal{D}) \rightarrow \mathcal{CU}(\mathcal{D})$ is called *ideal* if, for all $X \in \mathcal{CU}$, and $n \in \mathbb{N}$

$$(f(X))_{n+1} = (f((X)_n))_{n+1}.$$

Show that such an ideal function f has a unique fixed-point $X \in \mathcal{CU}(\mathcal{D})$.

CHAPTER 11

APPLICATIONS

11A. 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 $A \leq B$, 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 type inequalities for recursive types is sound and complete for this interpretation. We start by introducing some basic systems of subtyping for simple types, showing by way of examples how, assuming that some types are included in others, one can achieve the same effects as having recursive types. Then, we discuss in some more detail the formal system described in [Brandt and Henglein \[1998\]](#) for axiomatizing the subtyping relation on recursive types introduced by [Amadio and Cardelli \[1993\]](#), which is by now standard.

11A.1. DEFINITION. A *type structure* is of the form $\mathcal{S} = \langle |\mathcal{S}|, \leq, \rightarrow \rangle$, where $\langle |\mathcal{S}|, \leq \rangle$ is a poset and $\rightarrow : |\mathcal{S}|^2 \rightarrow |\mathcal{S}|$ is a binary operation such that for $a, b, a', b' \in |\mathcal{S}|$ one has

$$a' \leq a \& b \leq b' \Rightarrow (a \rightarrow b) \leq (a' \rightarrow b').$$

These type structures should not be confused with type structures in Part III of this book, which latter name is an abbreviation of ‘intersection type structures’. Note that a type structure $\mathcal{S} = \langle |\mathcal{S}|, \leq, \rightarrow \rangle$ can be considered as a type algebra $\langle |\mathcal{S}|, \rightarrow \rangle$. We call the latter the type algebra underlying the type structure \mathcal{S} .

11A.2. DEFINITION. Let \mathcal{S} be a type structure and $a, b \in \mathcal{S}$. Then the system of type assignment $\lambda_{\leq}^{\mathcal{S}}$ is defined by the axioms and rules in Fig. 29. Write $\Gamma \vdash_{\lambda_{\leq}^{\mathcal{S}}} M : a$ or $\Gamma \vdash_{\mathcal{S}} M : a$ if $\Gamma \vdash M : a$ can be derived in system $\lambda_{\leq}^{\mathcal{S}}$.

11A.3. PROPOSITION. *Let \mathcal{S} be a type structure. Then*

$$\Gamma \vdash_{\lambda_{\leq}^{\mathcal{S}}} M : A \Rightarrow \Gamma \vdash_{\lambda_{\leq}^{\mathcal{S}}} M : A,$$

where $\Gamma \vdash_{\lambda_{\leq}^{\mathcal{S}}} M : A$ is type assignment using the type algebra underlying \mathcal{S} .

PROOF. Trivial as $\lambda_{\leq}^{\mathcal{S}}$, see Definition 7A.2, has less rules than $\lambda_{\leq}^{\mathcal{S}}$. ■

11A.4. LEMMA. *Suppose that $\Gamma \subseteq \Gamma'$. Then for all $a \in \mathcal{S}$*

$$\Gamma \vdash_{\mathcal{S}} M : a \Rightarrow \Gamma' \vdash_{\mathcal{S}} M : a.$$

(axiom)	$\Gamma \vdash x : a \quad \text{if } (x:a) \in \Gamma$
(\rightarrow E)	$\frac{\Gamma \vdash M : a \rightarrow b \quad \Gamma \vdash N : a}{\Gamma \vdash (MN) : b}$
(\rightarrow I)	$\frac{\Gamma, x:a \vdash M : b}{\Gamma \vdash (\lambda x.M) : (a \rightarrow b)}$
(\leqslant)	$\frac{\Gamma \vdash M : a \quad a \leqslant b}{\Gamma \vdash M : b}$

FIGURE 29. The system $\lambda_{\leqslant}^{\mathcal{S}}$

11A.5. EXAMPLE. Let a, b be elements of a type structure \mathcal{S} .

- (i) If $b \leqslant (b \rightarrow a)$, then $\vdash_{\mathcal{S}} (\lambda x.xx) : (b \rightarrow a)$.
- (ii) If $(b \rightarrow a) \leqslant b$, then $\vdash_{\mathcal{S}} (\lambda x.xx) : (b \rightarrow a) \rightarrow a$.
- (iii) If $((b \rightarrow a) \rightarrow a) \leqslant (b \rightarrow a) \leqslant b$, then $\vdash_{\mathcal{S}} (\lambda x.xx)(\lambda x.xx) : a$.
- (iv) If $a \leqslant (a \rightarrow a)$, then $\vdash_{\mathcal{S}} (\lambda x.x(xx)) : (a \rightarrow a)$.

11A.6. DEFINITION. Let $\mathcal{S}, \mathcal{S}'$ be type structures. A map $h : \mathcal{S} \rightarrow \mathcal{S}'$ is called a *a morphism* of type structures if for all $a, b \in \mathcal{S}$ one has

$$\begin{aligned} a \leq_{\mathcal{S}} b &\Rightarrow h(a) \leq_{\mathcal{S}'} h(b); \\ h(a \rightarrow b) &= h(a) \rightarrow_{\mathcal{S}'} h(b). \end{aligned}$$

11A.7. LEMMA. Let $h : \mathcal{S} \rightarrow \mathcal{S}'$ be a morphism of type structures. Then for all $a \in \mathcal{S}$

$$\Gamma \vdash_{\mathcal{S}} M : a \Rightarrow h(\Gamma) \vdash_{\mathcal{S}'} M : h(a).$$

Type structures from type theories

11A.8. DEFINITION. (i) Now we will be a bit more specific about the set \mathbb{A} of atom on which $\mathbb{T}^{\mathbb{A}}$ is defined. We work with \mathbb{A} countable (finite or countably infinite).

- (ii) The following are some of the type atoms that will be used in different contexts.

c_0, c_1, c_2, \dots	atoms without special role
$\perp, \top, \mathbf{nat}, \mathbf{int}, \mathbf{bool}, \dots$	atoms with special properties.

(iii) The atoms \top and \perp are called *top* and *bottom*, respectively. The intention of the special atoms **nat**, **int**, **bool** is to have primitive natural numbers, integers and booleans. These can be defined, see Section 3.1, but having primitive ones is often more efficient.

11A.9. DEFINITION. (i) Let \mathcal{H} be a set of type inequalities. Define for $A, B \in \mathbb{T}^{\mathbb{A}}$ *derivability* of $A \leqslant B$ from basis \mathcal{H} , notation $\mathcal{H} \vdash A \leqslant B$, by the following axioms and

rules.

(hyp)	$\mathcal{H} \vdash A \leqslant B,$	if $(A \leqslant B) \in \mathcal{H},$
(refl)	$\mathcal{H} \vdash A \leqslant A$	
(trans)	$\frac{\mathcal{H} \vdash A \leqslant B \quad \mathcal{H} \vdash B \leqslant C}{\mathcal{H} \vdash A \leqslant C}$	
(\rightarrow)	$\frac{\mathcal{H} \vdash A' \leqslant A \quad \mathcal{H} \vdash B \leqslant B'}{\mathcal{H} \vdash (A \rightarrow B) \leqslant (A' \rightarrow B')}$	

- (ii) A set of type inequalities \mathcal{T} is a *type theory* if for all $A, B \in \mathbb{T}$

$$\mathcal{T} \vdash A \leqslant B \Rightarrow (A \leqslant B) \in \mathcal{T}.$$

- (iii) If $\mathcal{T} = \{A \leqslant B \mid \mathcal{H} \vdash A \leqslant B\}$, then \mathcal{T} is the type theory *axiomatized* by \mathcal{H} .

11A.10. DEFINITION. (i) A type inequality is called *inflationary* in a type atom α if it is of the form $\alpha \leqslant A$; it is called *deflationary* in α if it is of the form $A \leqslant \alpha$. In both cases, the type atom α is called a *subject* of the judgement. A type inequality is called *atomic* if it is of the form $\alpha \leqslant \beta$.

(ii) A set of type inequalities \mathcal{H} is called *homogeneous in α* if the subtyping judgements in \mathcal{H} of which α is subject are all inflationary or they are all deflationary.

(iii) A set of type inequalities \mathcal{H} is called *homogeneous* if it is homogeneous in α for every type atom α that occurs as subject.

(iv) A type theory \mathcal{T} is called *homogeneous* if it is axiomatized by a homogeneous set \mathcal{H} of type inequalities.

(v) The set \mathcal{H} is called *inflationary* (respectively *deflationary*), if it consists only of inflationary (respectively deflationary) type inequalities.

(vi) A type theory is *inflationary* (respectively *deflationary*), if it is axiomatized by an inflationary (respectively deflationary) set of type inequalities.

11A.11. DEFINITION. Let \mathcal{T} be a type theory over $\mathbb{T} = \mathbb{T}^{\mathbb{A}}$.

- (i) Write for $A, B \in \mathbb{T}$

$$\begin{aligned} A \leqslant_{\mathcal{T}} B &\triangleq (A \leqslant B) \in \mathcal{T}; \\ A \approx_{\mathcal{T}} B &\triangleq A \leqslant_{\mathcal{T}} B \& B \leqslant_{\mathcal{T}} A; \\ [A]_{\mathcal{T}} &\triangleq \{B \in \mathbb{T} \mid A \approx_{\mathcal{T}} B\}. \end{aligned}$$

- (ii) The *type structure determined by \mathcal{T}* is $\mathcal{S}_{\mathcal{T}} = \langle \mathbb{T}/\approx_{\mathcal{T}}, \rightarrow, \leqslant \rangle$, with

$$\begin{aligned} \mathbb{T}/\approx_{\mathcal{T}} &\triangleq \{[A]_{\mathcal{T}} \mid A \in \mathbb{T}\}; \\ [A]_{\mathcal{T}} \rightarrow [B]_{\mathcal{T}} &\triangleq [A \rightarrow B]_{\mathcal{T}}; \\ [A]_{\mathcal{T}} \leqslant [B]_{\mathcal{T}} &\triangleq A \leqslant_{\mathcal{T}} B. \end{aligned}$$

The last two definitions are independent on the choice of representatives.

- (iii) Write $x_1:A_1, \dots, x_n:A_n \vdash_{\mathcal{T}} M : A$, with $A_1, \dots, A_n, A \in \mathbb{T}$ for

$$x_1:[A_1]_{\mathcal{T}}, \dots, x_n:[A_n]_{\mathcal{T}} \vdash_{\mathcal{S}_{\mathcal{T}}} M : [A]_{\mathcal{T}}.$$

The following result relates typability and normalization of λ -terms.

11A.12. PROPOSITION. (i) Let \mathcal{T} be a homogeneous type theory. Then

$$\Gamma \vdash_{\mathcal{T}} M : A \Rightarrow M \text{ is strongly normalizing.}$$

(ii) Let $N \in \Lambda$ be in β -nf. Then there is an inflationary type theory \mathcal{T} such that N can be typed in $\vdash_{\mathcal{T}}$, i.e. for some \mathbb{A} , $A \in \mathbb{T}^{\mathbb{A}}$, and Γ one has

$$\Gamma \vdash_{\mathcal{T}} N : A.$$

PROOF. We only give a sketch of the proofs, leaving most of the details to the reader.

(i) \mathcal{T} is axiomatized by a homogeneous set \mathcal{H} of type inequalities. Interpret types as saturated sets, defined in 9C.5, using the first two clauses of 9C.9 and the type environment τ such that

$$\begin{aligned}\tau(\alpha) &= \perp_{\text{SAT}} = \bigcap_{\mathcal{X} \in \text{SAT}} \mathcal{X}, & \text{if } (\alpha \leq A) \in \mathcal{H} \text{ for some } A; \\ &= \top_{\text{SAT}} = \text{SN}, & \text{if } (A \leq \alpha) \in \mathcal{H} \text{ for some } A. \\ &= X, & \text{with } X \in \text{SAT} \text{ arbitrary, otherwise.}\end{aligned}$$

Observe that by homogeneity of \mathcal{H} the type environment τ is well-defined. By a straightforward induction on the derivations one shows, cf. Definition 9C.14, that

$$\Gamma \vdash_{\mathcal{T}} M : A \Rightarrow \Gamma \models M : A.$$

Therefore, as in the proof of Theorem 9C.16, every typable term belongs to a saturated set, hence is SN.

(ii) Proceed by induction on the structure of the normal form. The only interesting case is when $N \equiv zN_1 \cdots N_k$. Then, by induction hypothesis, there are type theories $\mathcal{T}_1, \dots, \mathcal{T}_k$ generated by inflationary $\mathcal{H}_1, \dots, \mathcal{H}_k$, types A_i , and bases Γ_i such that for each $i = 1, \dots, k$ one has $\Gamma_i \vdash_{\mathcal{T}_i} N_i : A_i$. Let \mathcal{T} be the type theory generated by the inflationary $\mathcal{H}_1 \cup \dots \cup \mathcal{H}_k$. Assume that a variable x has types $X_{j(1)}, \dots, X_{j(k)}$ in bases $\Gamma_{j(1)}, \dots, \Gamma_{j(k)}$. Let for a new type variable t_x the statement $x : t_x$ be in Γ and let \mathcal{H} be $\bigcup_i \mathcal{H}_i$ plus the type inequalities $t_x \leqslant X_{j(1)}, \dots, t_x \leqslant X_{j(k)}$. Thus, we get a basis Γ and an inflationary \mathcal{H} such that, if \mathcal{T} is the type theory generated by \mathcal{H} , $\Gamma, z : A_1 \rightarrow \dots \rightarrow A_k \rightarrow \beta \vdash_{\mathcal{T}} zN_1 \cdots N_k : \beta$. ■

Subject reduction

We show now that the straightforward generalization of the notion of invertibility to subtyping yields the subject reduction property, as for the corresponding property for type equality discussed in Section 9A.

11A.13. DEFINITION (Invertibility for subtyping). A type structure $\mathcal{S} = \langle |\mathcal{S}|, \leqslant, \rightarrow \rangle$ is *invertible* if for all $a, b, a', b' \in |\mathcal{S}|$ one has

$$(a \rightarrow b) \leqslant (a' \rightarrow b') \Rightarrow a' \leqslant a \& b \leqslant b'.$$

The following parallels Corollary 9A.5 of the Inversion Lemma 9A.3 for derivations of system $\lambda_{\leqslant}^{\mathcal{S}}$.

11A.14. LEMMA (Inversion Lemma). *Let \mathcal{S} be a type structure. Then*

- (i) $\Gamma \vdash_{\lambda \mathcal{S}} x : a \Leftrightarrow \exists a' \in \mathcal{S}. [a' \leq a \ \& \ (x:a') \in \Gamma]$.
- (ii) $\Gamma \vdash_{\lambda \mathcal{S}} (MN) : a \Leftrightarrow \exists b \in \mathcal{S}. [\Gamma \vdash_{\lambda \mathcal{S}} M : (b \rightarrow a) \ \& \ \Gamma \vdash_{\lambda \mathcal{S}} N : b]$.
- (iii) $\Gamma \vdash_{\lambda \mathcal{S}} (\lambda x.M) : a \Leftrightarrow \exists b, c \in \mathcal{S}. [a \geq (b \rightarrow c) \ \& \ \Gamma, x:b \vdash_{\lambda \mathcal{S}} M : c]$.

PROOF. (\Leftarrow) immediate from the rules for $\vdash_{\lambda \mathcal{S}}$. (\Rightarrow) The proof is by induction on the height of the derivation of typings $\Gamma \vdash_{\lambda \mathcal{S}} M : a$. The basis is obvious, and the induction step considers the possible shapes of the term M .

(i) $M \equiv x$, then case (i) applies because the last rule must be (\leq).

(ii) $M \equiv PQ$. If the last rule applied is ($\rightarrow E$) then case (ii) applies; if the last rule applied is (\leq) then the derivation has the shape

$$\frac{\Gamma \vdash PQ : a' \quad a' \leq a}{\Gamma \vdash_{\lambda \mathcal{S}} PQ : a}$$

By induction hypothesis there is a type $b \in \mathcal{S}$ such that $\Gamma \vdash_{\lambda \mathcal{S}} P : b \rightarrow a'$ and $\Gamma \vdash_{\lambda \mathcal{S}} Q : b$, and these judgements can be reassembled in a derivation whose last step is an application of rule ($\rightarrow E$):

$$\frac{\Gamma \vdash_{\lambda \mathcal{S}} P : b \rightarrow a \quad \Gamma \vdash_{\lambda \mathcal{S}} Q : b}{\Gamma \vdash_{\lambda \mathcal{S}} PQ : a}.$$

(iii) $M \equiv \lambda x.N$. If the last rule is ($\rightarrow I$), then $a = b \rightarrow c$; if the last rule applied is (\leq), then $\Gamma \vdash_{\lambda \mathcal{S}} \lambda x.N : a'$ and $a' \leq a$. By induction hypothesis we have $\Gamma, x:b' \vdash_{\lambda \mathcal{S}} N : c'$, where $b' \rightarrow c' \leq a'$. Then we can take $b = b'$ and $c = c'$. ■

11A.15. DEFINITION. Let $\Gamma = \{x_1 : a_1, \dots, x_n : a_n\}$ and $\Gamma' = \{x_1 : a'_1, \dots, x_n : a'_n\}$. Then we write $\Gamma' \leq \Gamma$ if $a'_i \leq a_i$ for every $1 \leq i \leq n$.

11A.16. LEMMA. *Let $\Gamma \vdash M : a$ and $\Gamma' \leq \Gamma$. Then $\Gamma' \vdash M : a$.*

PROOF. By induction on the derivation of $\Gamma \vdash M : a$. ■

11A.17. COROLLARY (Inversion Lemma). *Let \mathcal{S} be an invertible type structure. Then*

$$\Gamma \vdash_{\lambda \mathcal{S}} \lambda x.M : (a \rightarrow b) \Leftrightarrow \Gamma, x:a \vdash_{\lambda \mathcal{S}} M : b.$$

PROOF. (\Rightarrow) If $\Gamma \vdash_{\lambda \mathcal{S}} \lambda x.M : a \rightarrow b$, then by Lemma 11A.14 $a \rightarrow b \geq c \rightarrow d$ and $\Gamma, x:c \vdash_{\lambda \mathcal{S}} M : d$. By invertibility $a \leq c$ and $d \leq b$. Hence $\Gamma, x:a \vdash_{\lambda \mathcal{S}} M : b$. (\Leftarrow) By rule ($\rightarrow I$). ■

11A.18. LEMMA. *Suppose $x \notin \text{FV}(M)$, then*

$$\Gamma, x:b \vdash_{\lambda \mathcal{S}} M : a \Rightarrow \Gamma \vdash_{\lambda \mathcal{S}} M : a.$$

PROOF. By induction on the derivation of $\Gamma, x:b \vdash_{\lambda \mathcal{S}} M : a$. ■

Now we can prove the subject reduction property for one step $\beta\eta$ -reduction.

11A.19. LEMMA. (i) *If $\Gamma \vdash_{\lambda \mathcal{S}} \lambda x.(Mx) : a$ and $x \notin \text{FV}(M)$, then $\Gamma \vdash_{\lambda \mathcal{S}} M : a$.*

(ii) *If moreover \mathcal{S} is invertible, then*

$$\Gamma \vdash_{\lambda \mathcal{S}} (\lambda x.M)N : a \Rightarrow \Gamma \vdash_{\lambda \mathcal{S}} (M[x := N]) : a.$$

PROOF. Analogous to the proof of Lemma 9A.7. ■

11A.20. COROLLARY (Subject Reduction). *Let \mathcal{S} be an invertible type structure. Then $\lambda_{\leq}^{\mathcal{S}}$ satisfies the subject reduction property for $\beta\eta$.*

PROOF. By Lemma 11A.19. ■

11A.21. EXAMPLE. (i) We can type self-application by an essential use of rule (\leq) :

$$\{s \leq s \rightarrow t\} \vdash \lambda x.xx : s \rightarrow t$$

Observe that $\{s \leq s \rightarrow t\}$ is inflationary, hence homogeneous because it consists of one type inequality.

(ii) type inequalities can have the same effect as simultaneous recursions:

$$\{s \leq s \rightarrow s, s \rightarrow s \leq s\} \vdash (\lambda x.xx)(\lambda x.xx) : s$$

Actually, the set $\{s \leq s \rightarrow s, s \rightarrow s \leq s\}$ allows to type every pure λ -term, and in this sense is equivalent to the simultaneous recursion $\{s = s \rightarrow s\}$. Note that this set of type inequalities is not homogeneous.

Subtyping recursive types

Now we will define μ -type theories. In the following we summarize the theory of subtyping on recursive types, seen as representations of infinite trees, introduced by [Amadio and Cardelli \[1993\]](#), and the equivalent formulation of [Brandt and Henglein \[1998\]](#).

We can turn the type algebra \mathbb{T}_{μ}^* into a type structure by considering approximations using the elements \perp and \top . For most technical details, the reader is referred to the original papers, and also to the elegant exposition of [Gapeyev, Levin, and Pierce \[2002\]](#).

To introduce the basic notion of subtyping in $\mathbb{T}_{\mu}^{\mathbb{A}}$ we add to the set $\mathbb{T}_{\mu}^{\mathbb{A}}$ two type constants $\perp, \top \in \mathbb{A}$ representing the minimal and maximal elements of the set of types. Obviously other constants and corresponding type inclusions could be added, like $\mathbf{nat} \leq \mathbf{int}$.

First we need to define *lower approximations* $\tau \downarrow n$ and *upper approximations* $\tau \uparrow n$, for all $n \in \omega$, for an infinite tree τ as follows.

11A.22. DEFINITION (Approximations of an infinite tree). Let τ be an infinite tree, its *lower* and *upper* approximations, notation $\tau \downarrow n, \tau \uparrow n$, are the partial types defined simultaneously as follows. Let $n \in \mathbb{N}$.

$$\begin{array}{lll} \tau \downarrow 0 & \triangleq & \perp \\ (\tau' \rightarrow \tau'') \downarrow (n+1) & \triangleq & \tau' \uparrow n \rightarrow \tau'' \downarrow n \\ \alpha \downarrow (n+1) & \triangleq & \alpha \end{array} \quad \begin{array}{lll} \tau \uparrow 0 & \triangleq & \top \\ (\tau' \rightarrow \tau'') \uparrow (n+1) & \triangleq & \tau' \downarrow n \rightarrow \tau'' \uparrow n \\ \alpha \uparrow (n+1) & \triangleq & \alpha, \end{array}$$

where α is a type constant or variable.

11A.23. DEFINITION (Subtyping). (i) The *subtyping relation*, notation \leq_{fin} , on partial types is defined by

$$\begin{aligned} A &\leq_{fin} A; \\ \perp &\leq_{fin} A; \quad A \leq_{fin} \top; \\ \frac{A \leq_{fin} B \quad B \leq_{fin} C}{A \leq_{fin} C}; \quad \frac{A' \leq_{fin} A \quad B \leq_{fin} B'}{A \rightarrow B \leq_{fin} A' \rightarrow B'}. \end{aligned}$$

(ii) For a recursive type A , let $(A)_\mu^*$ be the tree unfolding of A as in Notation 7E.19(ii). The subtyping relation \leq_μ^* is defined by

$$A \leq_\mu^* B \Leftrightarrow \forall n \in \omega. ((A)_\mu^*) \downarrow n \leq_{fin} ((B)_\mu^*) \downarrow n.$$

11A.24. REMARK. Let $A, B \in \mathbb{T}_\mu^{\{\perp, \top\}}$, (where \perp and \top are considered as distinct atomic types). Then for $=_\mu^*$ defined as in Definition 7E.19(iii) one has

$$A =_\mu^* B \Leftrightarrow [A \leq_\mu^* B \& B \leq_\mu^* A].$$

11A.25. DEFINITION. Define the μ -type structure $\mathcal{T}_{BH} = \langle \mathbb{T}_\mu^*, \leq_\mu^*, \rightarrow \rangle$, where \leq_μ^*, \rightarrow are lifted to equivalence classes in an obvious way.

Also for this notion of subtyping we can define, following Brandt and Henglein [1998], a sound and complete coinductive proof system.

11A.26. DEFINITION. (i) Let \mathcal{H} be a set of subtype assumptions of the form $A \leq B$ for $A, B \in \mathbb{T}_\mu$. The system (BH_{\leq}) is defined by the following axioms and rules.

(hyp)	$\mathcal{H} \vdash A \leq B, \quad \text{if } A \leq B \in \mathcal{H}$
(refl)	$\mathcal{H} \vdash A \leq A$
(trans)	$\frac{\mathcal{H} \vdash A \leq B \quad \mathcal{H} \vdash B \leq C}{\mathcal{H} \vdash A \leq C}$
(\rightarrow)	$\frac{\mathcal{H} \vdash A' \leq A \quad \mathcal{H} \vdash B \leq B'}{\mathcal{H} \vdash A \rightarrow B \leq A' \rightarrow B'}$
(unfold)	$\mathcal{H} \vdash \mu\alpha.A \leq A[\alpha := \mu\alpha.A]$
(fold)	$\mathcal{H} \vdash A[\alpha := \mu\alpha.A] \leq \mu\alpha.A$
(bot)	$\mathcal{H} \vdash \perp \leq A$
(top)	$\mathcal{H} \vdash A \leq \top$
(Arrow/Fix)	$\frac{\mathcal{H}, (A \rightarrow B) \leq (A' \rightarrow B') \vdash (A' \leq A) \quad \mathcal{H}, (A \rightarrow B) \leq (A' \rightarrow B') \vdash (B \leq B')}{\mathcal{H} \vdash (A \rightarrow B) \leq (A' \rightarrow B')}$

FIGURE 30. The system (BH_{\leq}) .

Write $\mathcal{H} \vdash_{BH_{\leq}} A \leq B$ if $A \leq B$ is derivable from assumptions in \mathcal{H} in (BH_{\leq}) .

The above system can be proved sound and complete for \leq_μ^* . The proof is a straightforward extension of the corresponding proof for the system (for type equality) discussed in Section 1.

11A.27. THEOREM. (i) $\vdash_{BH_{\leq}} A \leq B \implies A \leq_\mu^* B$.

(ii) $A \leq_\mu^* B \implies \vdash_{BH_{\leq}} A \leq B$.

PROOF. (i) By Brandt and Henglein [1998], Theorem 2.2.

(ii) By [Brandt and Henglein \[1998\]](#), Theorem 2.6. ■

The formalization of [Brandt and Henglein \[1998\]](#) is based on a coinductive analysis of recursive types, ultimately going back to the theory of infinite trees outlined in the last Section of Chapter 7E. The following notion is the key to the extension of this conceptual framework to subtyping.

11A.28. **DEFINITION** (Simulations on recursive types). A *simulation* on recursive types is a binary relation \mathcal{R} that satisfies the following properties.

- (i) $(A \rightarrow B) \mathcal{R} (A' \rightarrow B') \Rightarrow A' \mathcal{R} A \& B \mathcal{R} B';$
- (ii) $\mu t. A \mathcal{R} B \Rightarrow A[t := \mu t. A] \mathcal{R} B;$
- (iii) $A \mathcal{R} \mu t. B \Rightarrow A \mathcal{R} B[t := \mu t. B];$
- (iv) $A \mathcal{R} B \Rightarrow \mathcal{L}(A) \leq^* \mathcal{L}(B),$

where $\mathcal{L}(A)$ is the root symbol of the tree A^* and the ordering on symbols is the least poset with the property that $\perp \leq \top$.

11A.29. **PROPOSITION.** *Let \mathcal{R} be a simulation. Then for $A, B \in \mathbb{T}_\mu$*

$$A \mathcal{R} B \Rightarrow A \leq^*_\mu B.$$

PROOF. [Brandt and Henglein \[1998\]](#), Lemma 2.2. ■

Models of subtyping

In the sequel let \mathcal{D} be a reflexive structure, Definition 10A.12, with a notion of approximation, Definition 10B.1. We will give natural truth conditions for type inequalities so that the system BH_\leq is sound when types are interpreted, as in Section 10B, as complete uniform subsets of a reflexive structure with a notion of approximation \mathcal{D} . This soundness result also motivates the contravariance of arrow types in the first argument w.r.t. the subtyping relation, which is hard to justify using only the definition of \leq^*_μ . Finally, we shall point out under what conditions the reverse implication holds.

Soundness

The following Lemma characterizes inclusion for elements of $\mathcal{C}U$.

11A.30. **LEMMA.** *Let $X, Y \in \mathcal{C}U$. Then*

$$X \subseteq Y \Leftrightarrow \forall n \in \omega. X_n \subseteq Y_n.$$

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 $d = \bigsqcup_{n \in \omega} d_n \in Y$. ■

We now define a notion of satisfaction for type inequalities exploiting the stratification of \mathcal{D} into levels, following the development of Chapter 10B.

11A.31. DEFINITION. Fix a reflexive structure with a notion of approximation \mathcal{D} and let η be a type environment relative to \mathcal{D} and $k \in \omega$:

- (i) $\eta \models_k A \leq B \Leftrightarrow \mathcal{I}^k[A]_\eta \subseteq \mathcal{I}^k[B]_\eta$.
- (ii) $\eta \models_k \mathcal{H} \Leftrightarrow \eta \models_k A \leq B$, for all judgements $A \leq B$ in \mathcal{H} .
- (iii) $\mathcal{H} \models_k A \leq B \Leftrightarrow [\eta \models_k \mathcal{H} \Rightarrow \eta \models_k A \leq B]$, for all η .
- (iv) $\mathcal{H} \models A \leq B \Leftrightarrow \mathcal{H} \models_k A \leq B$, for all $k \in \omega$.

11A.32. LEMMA. $\mathcal{H} \vdash_{\text{BH}_\leq} A \leq B \Rightarrow \mathcal{H} \models A \leq B$

PROOF. By induction on the length of the proof that $\mathcal{H} \vdash_{\text{BH}_\leq} A \leq B$. This is clear if the last rule applied is (const), (hyp), (fold) or (unfold): for the last two cases use Lemma 10B.16. Assume that the last rule applied is (Arrox/Fix). Then the last inference has the following shape.

$$\frac{\mathcal{H}, A \rightarrow B \leq A' \rightarrow B' \vdash A' \leq A \quad \mathcal{H}, A \rightarrow B \leq A' \rightarrow B' \vdash B \leq B'}{\mathcal{H} \vdash A \rightarrow B \leq A' \rightarrow B'}$$

By the induction hypothesis

$$\mathcal{H}, A \rightarrow B \leq A' \rightarrow B' \models A' \leq A$$

$$\mathcal{H}, A \rightarrow B \leq A' \rightarrow B' \models B \leq B'$$

and we have to show that, for all $k \in \omega$,

$$\mathcal{H} \models_k A \rightarrow B \leq A' \rightarrow B'.$$

We do this by induction on k . The base case is obvious, as both sides are interpreted as $\{\perp_D\}$. For the induction step, assume that $\mathcal{H} \models_k A \rightarrow B \leq A' \rightarrow B'$ and assume also that $\eta \models_{k+1} \mathcal{H}$. We have to show that $\mathcal{I}^{k+1}[A \rightarrow B]_\eta \subseteq \mathcal{I}^{k+1}[A' \rightarrow B']_\eta$. By the induction hypothesis (on k) we have $\eta \models_k \mathcal{H}, A \rightarrow B \leq A' \rightarrow B'$, because if $\eta \models_{k+1} \mathcal{H}$, then also $\eta \models_k \mathcal{H}$, by the properties of the approximate interpretation of types and Lemma 11A.30. Hence, by the hypothesis of the main induction, $\mathcal{I}^k[A']_\eta \subseteq \mathcal{I}^k[A]_\eta$ and $\mathcal{I}^k[B]_\eta \subseteq \mathcal{I}^k[B']_\eta$. Now if $d \in \mathcal{I}^{k+1}[A \rightarrow B]_\eta = \mathcal{I}^k[A]_\eta \rightarrow^{n+1} \mathcal{I}^k[B]_\eta$ and $a \in \mathcal{I}^k[A']_\eta$, then $a \in \mathcal{I}^k[A]_\eta$. Hence $d \cdot a \in \mathcal{I}^k[B]_\eta \subseteq \mathcal{I}^k[B']_\eta$. Therefore $d \in \mathcal{I}^{k+1}[A' \rightarrow B']_\eta$. ■

11A.33. COROLLARY (Soundness). Suppose $A \leq_\mu^* B$. Then $\forall \eta (\mathcal{I}[A]_\eta \subseteq \mathcal{I}[B]_\eta)$.

Completeness

The formal system BH_\leq is too weak to achieve completeness, namely the reverse of Corollary 11A.33. For example (see Amadio and Cardelli [1993]), for any reflexive \mathcal{D} with a notion of approximation

$$\mathcal{I}[\alpha \rightarrow \beta]_\eta \subseteq \mathcal{I}[\gamma \rightarrow \top]_\eta,$$

for all type variables α, β, γ , and all type environments η relative to \mathcal{D} , but this fact cannot be proved in BH_\leq . One strategy to remedy this is to add all type inequalities of this form, as do Amadio and Cardelli [1993], who interpret types under what is called the *F-semantics*. Another strategy consists in replacing the atoms \top, \perp , which are only needed

for avoiding the triviality of the subtype relation, by atoms **nat**, **int** with **nat** \leq **int**. We use a domain \mathcal{D} satisfying the equation

$$\mathcal{D} \cong \mathbb{Z}_\perp + [\mathcal{D} \rightarrow \mathcal{D}]_\perp + \{\text{err}\}_\perp,$$

where \mathbb{Z}_\perp is the flat CPO of integers, E_\perp denotes the lifting of E , and $D + E$ is the coalesced sum of CPOs, i.e. the disjoint union of the non-bottom elements of D and E , with a new bottom element. The element **err** is the semantic value of terms that lead to run-time errors, notably those arising from type-incorrect applications. We shall also assume that \mathcal{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 10B.1. Then, types are interpreted as complete and uniform subsets of \mathcal{D} that do not contain **err** and, in addition, are also closed subsets of \mathcal{D} under the Scott topology.

11A.34. DEFINITION. Let $X \subseteq \mathcal{D}$. Then X is called an *ideal* if it is complete, uniform and closed in the Scott topology.

These variations on the standard setting are exploited to show that the natural relation of semantic subtyping induces a simulation on recursive types.

11A.35. LEMMA. *If A, B, A', B' are ideals of \mathcal{D} , then*

$$(A \Rightarrow B) \subseteq (A' \Rightarrow B') \Rightarrow A' \subseteq A \& B \subseteq B'.$$

PROOF. Let $A \Rightarrow B \subseteq A' \Rightarrow B'$ and suppose $A' \not\subseteq A$ towards a contradiction. Then there is a compact element d of D such that $d \in (A' \Rightarrow A)$. The step function $d \mapsto \text{err}$ belongs to $A \Rightarrow B$ because no $x \sqsupseteq d$ is in A , but not to $A' \Rightarrow B'$, contradiction.

Similarly suppose $B \not\subseteq B'$. Take $e \in (B \Rightarrow B')$. Then $(\perp_D \mapsto e) \in (A \Rightarrow B) \setminus (A' \Rightarrow B')$, again a contradiction. ■

11A.36. COROLLARY. *Define on \mathbb{T}_μ the binary relation*

$$A \mathcal{R} B \Leftrightarrow \forall \eta. \mathcal{I}[A]_\eta \subseteq \mathcal{I}[B]_\eta,$$

where η ranges over type environments that interpret type variables as ideals of \mathcal{D} . Then \mathcal{R} is a simulation.

PROOF. This follows easily from Lemma 11A.35 and Theorem 10B.17. ■

11A.37. THEOREM (Completeness). *Let $A, B \in \mathbb{T}_\mu$. Then*

$$\forall \eta \left(\mathcal{I}[A]_\eta \subseteq \mathcal{I}[B]_\eta \right) \Rightarrow A \leq^*_\mu B.$$

PROOF. Define

$$C \mathcal{R} D \Leftrightarrow \forall \eta. \left(\mathcal{I}[C]_\eta \subseteq \mathcal{I}[D]_\eta \right).$$

Then \mathcal{R} is a simulation, by Corollary 11A.36. Now Proposition 11A.29 applies. ■

11A.38. COROLLARY. *Let $A, B \in \mathbb{T}_\mu$. Then the following are equivalent.*

- (i) $A \leq^*_\mu B$
- (ii) $\forall \eta \left(\mathcal{I}[A]_\eta \subseteq \mathcal{I}[B]_\eta \right)$.
- (iii) $\vdash_{\mathbf{BH}_\leq} A \leq B$.

PROOF. (i) \Leftrightarrow (ii). By Theorem 11A.37 and Corollary 11A.33.

(i) \Leftrightarrow (iii). By Theorem 11A.27. ■

Now we obtain Theorem 10B.31, using Remark 11A.24.

11A.39. COROLLARY. $A =_{\mu}^* B \Leftrightarrow \forall \eta. [\mathcal{I}[A]_{\eta} = \mathcal{I}[B]_{\eta}]$.

11B. The principal type structures

The notions and results in this Section come from Andrew Polonsky [2010] with different proofs.

Remember that a *type structure* is of the form $\mathcal{S} = \langle |\mathcal{S}|, \leq_{\mathcal{S}}, \rightarrow_{\mathcal{S}} \rangle$; a type theory \mathcal{T} is a set of type inequalities closed under derivations of the system given in Definition 11A.9. Let \mathcal{S} range over type structures and \mathcal{T} over type theories. For a type theory \mathcal{T} , $\mathcal{S}_{\mathcal{T}}$ is the *type structure generated by \mathcal{T}* , as defined in Definition 11A.11.

Similar to Definition 9B.2 to each untyped $M \in \Lambda$ we assign a triple $\langle \Gamma_M, \mathcal{S}_M, a_M \rangle$, where now \mathcal{S}_M is a type structure, $a_M \in \mathcal{S}_M$, and Γ_M is a basis over \mathcal{S}_M . This \mathcal{S}_M will be the *principal type structure* of M .

11B.1. DEFINITION. Let $M \in \Lambda$. Define a set of type constants \mathbf{c}_M , a type α_M , a basis Γ_M , a set of type inequalities \mathcal{T}_M , and a type algebra \mathcal{A}_M with element a_M as follows. We do this by defining first for each subterm-occurrence $L \subseteq M$ for L not a variable a distinct type atom α_L . For variables x occurring in M we choose a fixed type α_x (for different occurrences the same α_x). Then we define the type theory \mathcal{T}_L for each subterm-occurrence $L \subseteq M$, obtaining this notion also for M as we have $M \subseteq M$.

L	\mathcal{T}_L
x	\emptyset
PQ	$\mathcal{T}_P \cup \mathcal{T}_Q \cup \{\alpha_P \leq (\alpha_Q \rightarrow \alpha_{PQ})\}$
$\lambda x.P$	$\mathcal{T}_P \cup \{\alpha_x \rightarrow \alpha_P \leq (\alpha_{\lambda x.P})\}$

Define \mathbf{c}_M as the set of all atomic types α_L and α_x occurring in M . Finally we define

$$\begin{aligned}\Gamma_M &\triangleq \{x:[\alpha_x] \mid x \in \text{FV}(M)\}, \\ \mathcal{S}_M &\triangleq \langle \mathbb{T}^{\mathbf{c}_M}/\mathcal{T}_M, \leq, \rightarrow \rangle, \\ a_M &\triangleq [\alpha_M]_{\mathcal{T}_M}.\end{aligned}$$

The type a_M is called the *principal recursive type* of M and \mathcal{S}_M its *principal type structure*. Γ_M is the *principal recursive basis* of M , which is empty if M is closed. The triple $\langle \Gamma_M, \mathcal{S}_M, a_M \rangle$ is called *principal recursive triple*. We call \mathcal{T}_M the *principal type theory* of M . If we consider the α_L as indeterminates it is like an sr, but with inequalities.

Below we will derive a principal type theorem for type structures very similar to Theorem 9B.3 for type algebras. We need an extra Definition and Lemma.

11B.2. DEFINITION. Let D be derivation of $\Gamma \vdash_{\mathcal{S}} M : a$. For each subterm L of M let D_L be the smallest subderivation of D whose last judgement has L as subject. Then b_L denotes the type that is assigned to L in the conclusion of D_L .

11B.3. EXAMPLE. Let $M \equiv \lambda y$ and D be the derivation

$$\frac{\frac{x:a \vdash x : a}{\frac{\frac{\vdash I : (a \rightarrow a) \quad a \leq c}{\frac{\vdash I : (a \rightarrow c) \quad y:a \vdash y : a}{\vdash M : c}}}}{}}$$

Then $b_x = a, b_I = a \rightarrow a, b_M = c$.

11B.4. LEMMA. (i) Let $\Gamma \vdash PQ : b_{PQ}$ have subderivations $\Gamma \vdash P : b_P, \Gamma \vdash Q : b_Q$. Then

$$b_P \leq b_Q \rightarrow b_{PQ}.$$

(ii) Let $\Gamma \vdash \lambda x.P : b_{\lambda x.P}$ have as subderivation $\Gamma, x:b_x \vdash P : b_P$. Then

$$b_x \rightarrow b_P \leq b_{\lambda x.P}.$$

PROOF. (i) Let the derivation D of $\Gamma \vdash PQ : b_{PQ}$ be

$$\frac{\begin{array}{c} \vdots \\ \frac{\Gamma \vdash P : b_P}{\vdots \leq} & \frac{\Gamma \vdash Q : b_Q}{\vdots \leq} \\ \hline \frac{\Gamma \vdash P : c \rightarrow b_{PQ}}{\Gamma \vdash Q : c} & \frac{\Gamma \vdash Q : c}{\Gamma \vdash PQ : b_{PQ}} \end{array}}{\Gamma \vdash PQ : b_{PQ}}.$$

Now $b_P \leq c \rightarrow b_{PQ}$ and $b_Q \leq c$. Hence $b_P \leq c \rightarrow b_{PQ} \leq b_Q \rightarrow b_{PQ}$.

(ii) Similarly. ■

11B.5. THEOREM. For $M \in \Lambda$ the principal recursive triple $\langle \Gamma_M, \mathcal{S}_M, a_M \rangle$ satisfies

(i) $\Gamma_M \vdash_{\mathcal{S}_M} M : a_M$.

(ii) $\Gamma \vdash_{\mathcal{S}} M : a \Leftrightarrow$ there is a morphism $h : \mathcal{S}_M \rightarrow \mathcal{S}$ such that
 $h(\Gamma_M) \subseteq \Gamma$ and $h(a_M) \leq a$.

(iii) For closed $M \in \Lambda^{\circ}$ this simplifies to

$$\vdash_{\mathcal{S}} M : a \Leftrightarrow \exists h : \mathcal{S}_M \rightarrow \mathcal{S}. h(a_M) \leq a.$$

PROOF. (i) As for Theorem 9B.3(i) with $=_{\mathcal{E}}$ replaced by \leq_T .

(ii) (\Leftarrow) By (i) and Lemmas 11A.7, 11A.4.

(\Rightarrow) Similar to (\Rightarrow) in Theorem 9B.3, where b_L is given by Definition 11B.2, using Lemma 11B.4.

(iii) By (ii). ■

11B.6. REMARK. (i) Using the principal type theories of $M, N \in \Lambda$ we can define

$$M \not\sim N \iff \text{there exists a morphism } h : \mathcal{T}_M \rightarrow \mathcal{T}_N$$

This introduces a new non-trivial partial order relation on Λ . This relation remains non-trivial over the set of unsolvable terms, and therefore provides a new avenue for studying relations between unsolvable terms. It is not immediately obvious whether the order described above is decidable, because type theories may be equivalent without being the same (for example, $\{\alpha \leq \beta\}$ is equivalent to $\{\alpha \leq \alpha\}$). We conjecture that the $\not\sim$ -order on Λ is decidable.

11C. Recursive types in programming languages

The results proved in the present Part II are relative to (slight variants of) the λ -calculus, while the languages of interest for the programming practice normally include predefined term and type constants and other type constructors, as mentioned in the notion of enriched type algebra in Chapter 7B. Yet, the theoretical work on recursive types expounded in Chapters 7–10 constitutes a mathematical framework for modeling the uses of recursive types in programming: on the one hand, many results carry over literally to the extended languages used in practice (for example the principal type-scheme property). On the other hand, the theoretical framework can be extended to practical languages mostly in a straightforward manner.

In this subsection we review very sketchily the various forms that recursive types take in programming languages. Recursive types are orthogonal to different programming paradigms, so we shall see how they show up in imperative, functional and object-oriented languages. We shall do this mostly by means of examples, referring to the literature for more details.

Extending the language

If we consider an extension of the λ -calculus where term constants c are present, then we must assume a type $\tau(c)$ for each of these belonging to the corresponding enriched type algebra. In this case it is standard to assume that some of the atoms of \mathbb{A} represent data types (for instance an atom **int** representing the set of integers). Each term constant c is then given a type $\tau(c)$ belonging to $\mathbb{T}_\mu(\mathbb{A})$ (for instance $\tau(n) = \mathbf{int}$ for all numerals n , $\tau(+) = \mathbf{int} \rightarrow \mathbf{int} \rightarrow \mathbf{int}$, etc.).

11C.1. REMARK. Observe also that a rich system of types, including an encoding of the types of integer and Boolean values, can be based on recursive types, provided a minimal set of basic type constructors is available. As an example, assume that the set of basic recursive types is extended as described by the following simplified syntax ([Cosmadakis \[1989\]](#)).

$$\boxed{\mathbb{T}_\mu ::= \mathbb{A} \mid \mathbb{T}_\mu \rightarrow \mathbb{T}_\mu \mid \mu \mathbb{A}. \mathbb{T}_\mu \mid \mathbb{T}_\mu \times \mathbb{T}_\mu \mid \mathbb{T}_\mu \oplus \mathbb{T}_\mu \mid (\mathbb{T}_\mu)_\perp}$$

Then we can define the following data types:

$$\begin{aligned} triv &\triangleq \mu t. t \\ O &\triangleq triv_\perp \\ bool &\triangleq O \oplus O \\ nat &\triangleq \mu t. O \oplus t \\ obliq &\triangleq \mu t. O \oplus t_\perp \\ vert &\triangleq \mu t. t_\perp \\ lamb &\triangleq \mu t. O \oplus (t \rightarrow t) \\ lazy &\triangleq \mu t. (t \rightarrow t)_\perp \end{aligned}$$

Within a formulation with explicit typing as in Section 0, we can add typed constants representing constructors and destructors for each of the new types. So, for example, we may have constants $up : A \rightarrow A_\perp$ and $down : A_\perp \rightarrow A$ (possibly with a new conversion rule $down(up(x)) = x$). This set of types is useful in formalizing a metalanguage for denotational semantics. Type *lamb* then

encodes the terms of the pure untyped λ -calculus, whereas *lazy* does the same for terms of the lazy λ -calculus ([Abramsky \[1990\]](#)). The types *nat*, *vert* and *obliq* represent three versions of a type of natural numbers.

Finding and checking types

Assume for the moment that only constants and constant types, and no other type constructors are added. Generalization to other standard type constructors is straightforward. In this case usually one must take into account that constant types cannot be equated to anything else but themselves. We do not allow, for instance, that **int** be equivalent to an arrow type. Moreover invertibility is always assumed. We say that the (principal) set of type equations \mathcal{E}_M is *consistent* if it does not imply any equation $\kappa = A$ where k is a constant type and A is either a different constant type or a non atomic type expression. Then, as a consequence of Theorem 9B.3 a term M can be typed (w.r.t. some invertible type algebra) iff \mathcal{E}_M is consistent. We can take into account constants in the construction of \mathcal{E}_M in Definition 9B.2 by taking $\mathcal{E}_c = \{\alpha_c = \tau(c)\}$ for each constant c occurring in M . We then can easily prove that \mathcal{E}_M is consistent iff for each atomic constant type κ the sr \mathcal{E}_M does not contain equations $\kappa = a_1, a_1 = a_2, \dots, a_n = C$ where $n \geq 0$ and C is either a constant type different from κ or a non atomic type expression. This also yields an algorithm for deciding whether \mathcal{E}_M is consistent.

All the results given in Section 9B still hold if we consider terms with additional constants. In some cases this makes these results more interesting. For instance, the problem of deciding whether a given term has a type w.r.t. some sr is no longer trivial since there are terms, like (3 3), which have no type at all w.r.t. any sr. By the subject reduction theorem, however, we can still prove that a term to which a type can be given in any system with recursive types can not produce incorrect applications during its evaluation. This is one of the motivations for the practical uses of these systems in programming languages [Milner \[1978\]](#).

In real programming languages recursive types are usually defined via recursive type equations, so the problem of typability with respect to a given sr is the closest one to real cases. However the declaration of constructors which is usually included in recursive type definitions make type inference much easier.

Imperative programming

Historically, a form of recursive type definition was introduced in ALGOL 68, see [van Wijngaarden \[1981\]](#), as “recursive mode declarations”, e.g.

```
mode cell = struct(ref cell next, int 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, see [Koster \[1969\]](#), or, equivalently, of components of solutions of finite systems of equations, [Pair \[1970\]](#); the latter method is thoroughly investigated in [Courcelle, Kahn, and Vuillemin \[1974\]](#), whereas the first anticipates some ideas on which the coinductive proof-system of [Brandt and Henglein \[1998\]](#) is based: in particular, also this algorithm allows a certain amount of circularity

because in order to decide whether two modes are equivalent, one assumes that they are and shows that this assumption does not lead to a contradiction.

Functional programming

Typed functional programming languages in the tradition of ML [Gordon, Milner, and Wadsworth \[1979\]](#), including Miranda and Haskell, allow recursive definitions of data-types which the type-checker exploits in inferring type schemes for programs, see [Peyton-Jones \[1987\]](#), Chs. 8–9. A typical example of such definitions, say, in Haskell [Peyton Jones \[editor\], Hughes \[editor\], Augustsson, Barton, Boutel, Burton, Fraser, Fasel, Hammond, Hinze, \[1999\]](#), 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 [Hudak, Peterson, and Fasel \[1999\]](#), §2.2.1.

Another example from SML (a call-by-value language) is a type declaration intended to define the notion of polymorphic streams.

```
type 'a stream = Scons of (unit -> 'a * 'a stream)
```

where `unit` is the one-element type. The function type is introduced to suspend evaluation.

Definitions like the example above also declare *constructors* (like `Leaf` and `Branch` in the example) that can be used in defining functions over elements of the data-type by pattern-matching, [Burstall \[1969\]](#). In type inference, however, constructors are treated like constants and have then a specific type (or type scheme) assigned a priori.

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 \[1988\]](#); we just discuss a few motivating examples.

11C.2. DEFINITION. We extend $\mathbb{T}_\mu = \mathbb{T}_\mu^{\mathbb{A}}$ with *record types* $\{\ell_1 : A_1, \dots, \ell_n : A_n\}$ as indicated by the following simplified syntax.

$$\boxed{\mathbb{T}_\mu ::= \mathbb{A} \mid \mathbb{T}_\mu \rightarrow \mathbb{T}_\mu \mid \mu \mathbb{A}. \mathbb{T}_\mu \mid \{\ell_1 : \mathbb{T}_\mu, \dots, \ell_n : \mathbb{T}_\mu\}}$$

where ℓ_1, \dots, ℓ_n are distinct *labels*, and we assume that \mathbb{A} contains atoms for base types like `nat`, `int`, `bool`, \dots .

Correspondingly, the set of terms is extended with *record structures* of the form $\{\ell_1 = M_1, \dots, \ell_n = M_n\}$ where M_1, \dots, M_n are terms, and constants of base types.

Record structures are tuples whose components are accessed via their labels: if $\{\ell_1 = M_1, \dots, \ell_n = M_n\}$ is such a structure, then $\{\ell_1 = M_1, \dots, \ell_n = M_n\}.\ell_i$ has the same meaning as M_i .

Recursive record structures can be interpreted as *objects*. For example, we may have a point object in the plane defined as follows.

$$\mathbf{Y} \left(\lambda s. \{x = 1.0, y = 2.0, \text{dist} = \lambda p. \sqrt{(p.x - s.x)^2 + (p.y - s.y)^2}\} \right)$$

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

$$\text{Point} \equiv \{x : \text{real}, y : \text{real}, \text{dist} : \text{Point} \rightarrow \text{real}\},$$

a recursive record type. Further developments of this approach to modeling objects are thoroughly described in [Abadi and Cardelli \[1996\]](#).

11D. Further reading

This Section collects further references, in addition to those indicated in the main text, where the reader may find additional information on the topics discussed in this Part.

Historical

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, [McCarthy \[1963\]](#), §2.6, 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 $S = 1 + A + A^2 + A^3 + \dots$, which describes a sequence as 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 [Fiore \[2004\]](#) for an introduction to these, 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 ([Smyth and Plotkin \[1982\]](#), [Gunter and Scott \[1990\]](#), [Freyd \[1990\]](#), [Freyd \[1991\]](#), [Freyd \[1992\]](#)) can be regarded as comprehensive semantical universes for recursive types, as may be seen already from the straightforward use of the category of CPOs in Section 10C.

On the syntactical side, recursive types appeared in J.H. Morris’ thesis [Morris \[1968\]](#), who considered the possibility of allowing, in the simply typed λ -calculus, “circular” type expressions like the solution of the equation $X = X \rightarrow \alpha$. At about the same time, a form of recursive type definition was introduced in ALGOL 68 as “recursive mode declarations” mentioned above, in Section 11C.

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 algorithms of [Curry \[1969\]](#), [Hindley \[1969\]](#) and [Milner \[1978\]](#), as remarked already in [Wand \[1987\]](#) (see [Aho, Sethi, and Ullman \[1986\]](#) 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 [Courcelle \[1983\]](#), §4, 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 [Cardone and Coppo \[1991\]](#), 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 endofunctor over the category of sets canonically associated with the signature, as in Proposition 7F.16. This is a well-known fact in the theory of coalgebras [Rutten \[2000\]](#), and has been exploited in [Fiore \[1996\]](#).

Models

The first model for the implicitly typed systems is given in [MacQueen, Plotkin, and Sethi \[1986\]](#), where types are interpreted as ideals over a Scott domain of the form $D \cong V + [D \rightarrow D]$. 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 the Banach fixed point theorem, Theorem 7F.5.

The interpretation of recursive types that we have described in §10B, exploiting the approximation structure of domains like D above induced by their construction as inverse limits by the technique in [Scott \[1972\]](#), stems from [Coppo \[1985\]](#), where also the completeness theorems 10B.30 and 10B.31 were first 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 [Cardone, Dezani-Ciancaglini, and de'Liguoro \[1994\]](#). On the other hand, it can be extended to second-order polymorphic types [Abadi and Plotkin \[1990\]](#) and even to bounded polymorphic types with subtyping [Cardone \[1991\]](#). 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. Concerning the interpretation of the explicitly typed systems, the domain models presented in Section 10C belong to the folklore of the subject.

Subtyping

The theory of subtyping recursive types has received much attention in recent years, mainly for the interest it has for the design of object-oriented programming languages. The seminal paper on this is [Amadio and Cardelli \[1993\]](#). The presentation of subtyping has been refined in [Brandt and Henglein \[1998\]](#), who have exploited in an elegant way the coinductive basis of every reasoning on infinite objects, like the infinite trees of which recursive types can be regarded as finite notations. See also [Grabmayer \[2005\], \[2007\]](#) for work in this direction. Algorithms for deciding subtyping between μ -types, with the relative complexity issues, have been studied first in [Kozen, Palsberg, and Schwartzbach \[1995\]](#). We have not dealt at all with this topic; pointers to the relevant

literature can be found in [Gapeyev, Levin, and Pierce \[2002\]](#). In [Jay \[2009\]](#) the (typed) lambda calculus is extended with patterns as a first class citizen.

11E. Exercises

- 11E.1. Call type theories \mathcal{T} , \mathcal{T}' *equivalent* if $\mathcal{T} \preccurlyeq \mathcal{T}'$ and $\mathcal{T}' \preccurlyeq \mathcal{T}$. In each of the following, show equivalence of the listed theories.
1. $\emptyset, \{\alpha \leq \alpha\}, \{\alpha \leq \beta\}, \{\alpha \leq A\}$, where A is a type with $\alpha \notin A$.
 2. $\{\alpha \leq (\alpha \rightarrow \beta)\}, \{(\alpha \rightarrow \beta) \leq \alpha\}, \{\gamma \leq \alpha, \gamma \leq (\alpha \rightarrow \beta)\}$.
- 11E.2. Let **TA** be the category of type algebras and **TS** be the category of type structures. Let $U : \mathbf{TS} \rightarrow \mathbf{TA}$ be the forgetful map that sends (S, \rightarrow, \leq) to (S, \rightarrow) .
1. Show that there is a functor $F : \mathbf{TA} \rightarrow \mathbf{TS}$ such that U is right adjoint to F .
 2. Show that F also has a left adjoint C .
 3. Let M be a lambda term, \mathcal{T} the principal type theory of M . Show that $C(S_{\mathcal{T}})$ is isomorphic to the principal type algebra of M in the sense of Definition 9B.2.
 4. Show that \mathcal{T} is a weakly initial object in **TS**. Show that $S_{\{\alpha \leq (\alpha \rightarrow \alpha), (\alpha \rightarrow \alpha) \leq \alpha\}}$ is a weakly terminal object.
 5. Show that **TS** has arbitrary coproducts (and finite products?)
 6. *Show that the Subtyping order refines the Typability order induced by \preccurlyeq between type algebras.*
 7. Show that there exist two unsolvables whose principal type theories are \preccurlyeq -incompatible.

Part 3

INTERSECTION TYPES λ_{\cap}^S

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* in the context of identifications and subtyping 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 flavor: 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, Coppo, and Dezani-Ciancaglini \[1983\]](#) 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 explored.

The interested reader can find a continuously updated bibliography maintained by Joe Wells on intersection types at URL <www.macs.hw.ac.uk/~jbw/itrs/bibliography.html>. Introductions to intersection types are in [Cardone and Coppo \[1990\]](#) and [Hindley \[1992\]](#).

CHAPTER 12

AN EXEMPLARY SYSTEM

There are several systems that assign intersection types to untyped lambda terms. These will be collectively denoted by λ_{\cap} . In this section we consider one particular system of this family, λ_{\cap}^{BCD} in order to outline the concepts and related properties. Definitions and the statements 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 λ_{\rightarrow} in such a way that not only subject reduction, but also subject expansion holds. The problem of subject expansion is the following. Suppose $\vdash_{\lambda_{\rightarrow}} M : A$ and that $M' \rightarrow_{\beta} M$. Does one have $\vdash_{\lambda_{\rightarrow}} M' : A$? Let us focus on one β -step. So let $M' \equiv (\lambda x.P)Q$ be a redex and suppose

$$\vdash_{\lambda_{\rightarrow}} P[x := Q] : A. \quad (1)$$

Do we have $\vdash_{\lambda_{\rightarrow}} (\lambda x.P)Q : A$? 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 $B \rightarrow A$ and therefore $\vdash_{\lambda_{\rightarrow}} (\lambda x.P)Q : A$. The mistake is that in (1) there may be several occurrences of Q , say $Q_1 \equiv Q_2 \equiv \dots \equiv Q_n$, having as types respectively B_1, \dots, 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

$$\begin{aligned} &\vdash_{\lambda_{\rightarrow}^{\mathbb{A}}} (\lambda x.\mathbf{I}(\mathbf{K}x)(\mathbf{I}x)) : A \rightarrow A, \\ &\not\vdash_{\lambda_{\rightarrow}^{\mathbb{A}}} (\lambda xy.x(\mathbf{K}y)(xy))\mathbf{I} : A \rightarrow A. \end{aligned}$$

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, \dots, B_n , we give them all of these types:

$$\vdash Q : B_1 \cap \dots \cap B_n,$$

implying that for all i one has $Q : B_i$. Then we will get

$$\begin{aligned} &\vdash (\lambda x.P) : B_1 \cap \dots \cap B_n \rightarrow A \quad \text{and} \\ &\vdash ((\lambda x.P)Q) : A. \end{aligned}$$

There is, however, a second problem. In the $\lambda\mathbf{K}$ -calculus, with its terms $\lambda x.P$ such that $x \notin \text{FV}(P)$ 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 $B_1 \cap \dots \cap B_n$ also for $n = 0$ and writing this type as \mathbf{U} , to be considered as the universal type, i.e. assigned to all terms.

Then in case $x \notin \text{FV}(P)$ one has

$$\begin{aligned}\vdash Q &: \mathbb{U} \\ \vdash (\lambda x.P) &: \mathbb{U} \rightarrow A \quad \text{and} \\ \vdash ((\lambda x.P)Q) &: A.\end{aligned}$$

This is the motivation to introduce a \leq relation on types with largest element \mathbb{U} and intersections such that $A \cap B \leq A$, $A \cap B \leq B$ and the extension of the type assignment by the subsumption 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 $((A \rightarrow B) \cap A) \rightarrow B$, while $(\lambda x.xx)(\lambda x.xx)$ only gets \mathbb{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. It is worth noticing that these models are models of the untyped lambda calculus.

The type assignment system $\lambda_{\cap}^{\text{BCD}}$ will be introduced in Section 12A and the corresponding filter model in 12B. Results are often stated without proof, as these will appear in later Chapters.

12A. The type assignment system $\lambda_{\cap}^{\text{BCD}}$

A typical member of the family of intersection type assignment systems is $\lambda_{\cap}^{\text{BCD}}$. This system is introduced in [Barendregt, Coppo, and Dezani-Ciancaglini \[1983\]](#) as an extension of the initial system in [Coppo and Dezani-Ciancaglini \[1980\]](#).

12A.1. DEFINITION. (i) Define the following sets of type atoms.

$$\begin{aligned}\mathbb{A}_{\infty} &\triangleq \{c_0, c_1, c_2, \dots\} \\ \mathbb{A}_{\infty}^{\mathbb{U}} &\triangleq \mathbb{A}_{\infty} \cup \{\mathbb{U}\}, \\ \mathbb{A}^{\text{BCD}} &\triangleq \mathbb{A}_{\infty}^{\mathbb{U}},\end{aligned}$$

where the type atom $\mathbb{U} \notin \mathbb{A}_{\infty}$ is a special symbol called [universe](#) or [universal top](#).

(ii) The [intersection type language](#) over \mathbb{A}^{BCD} , denoted by $\mathbb{T} = \mathbb{T}_{\cap}^{\text{BCD}}$ is defined by the following simplified syntax.

$$\boxed{\mathbb{T} ::= \mathbb{A}^{\text{BCD}} \mid \mathbb{T} \rightarrow \mathbb{T} \mid \mathbb{T} \cap \mathbb{T}}$$

NOTATION. (i) Greek letters α, β, \dots will denote arbitrary atoms in \mathbb{A}^{BCD} .

(ii) A, B, C, D, E range over arbitrary types in \mathbb{T} .

(iii) In [Barendregt, Coppo, and Dezani-Ciancaglini \[1983\]](#) the universe used to be denoted by ω .

(iv) When writing intersection types we shall use the following convention: the constructor \cap takes precedence over the constructor \rightarrow and the constructor \rightarrow associates to the right. For example

$$(A \rightarrow B \rightarrow C) \cap A \rightarrow B \rightarrow C \equiv ((A \rightarrow (B \rightarrow C)) \cap A) \rightarrow (B \rightarrow C).$$

12A.2. REMARK. In this Part III other sets \mathbb{T} of types will be formed by replacing the set \mathbb{A}^{BCD} of type atoms by an arbitrary set \mathbb{A} (finite or countably infinite). In this Chapter, however, we take $\mathbb{A} = \mathbb{A}^{\text{BCD}} = \mathbb{A}_{\infty}^{\mathbb{U}}$.

The following deductive system has as intention to introduce an appropriate pre-order on \mathbb{T} , compatible with the operator \rightarrow , such that $A \cap B$ is a greatest lower bound of A and B , for each A, B .

12A.3. DEFINITION (Intersection type preorder). The *intersection type theory* BCD is the set of all statements $A \leq B$ (to be read as “ A sub B ”), with $A, B \in \mathbb{T}$, derivable from the following axioms and rules, where $A, B, C, \dots \in \mathbb{T}$.

(refl)	$A \leq A$
(incl _L)	$A \cap B \leq A$
(incl _R)	$A \cap B \leq B$
(glb)	$\frac{C \leq A \quad C \leq B}{C \leq A \cap B}$
(trans)	$\frac{A \leq B \quad B \leq C}{A \leq C}$
(U _{top})	$A \leq \mathbb{U}$
(U \rightarrow)	$\mathbb{U} \leq A \rightarrow \mathbb{U}$
(\rightarrow \cap)	$(A \rightarrow B) \cap (A \rightarrow C) \leq A \rightarrow (B \cap C)$
(\rightarrow)	$\frac{A' \leq A \quad B \leq B'}{(A \rightarrow B) \leq (A' \rightarrow B')}$

12A.4. NOTATION. (i) For $(A \leq B) \in \text{BCD}$ we write $A \leq_{\text{BCD}} B$ or $\vdash_{\text{BCD}} A \leq B$ (or often just $A \leq B$ if there is little danger of confusion).

(ii) Write $A =_{\text{BCD}} B$ (or $A = B$) for $A \leq_{\text{BCD}} B \& B \leq_{\text{BCD}} A$.

(iii) We write $[\mathbb{T}]$ for the set \mathbb{T} modulo $=_{\text{BCD}}$. For types in BCD we usually work with $[\mathbb{T}]$.

(iv) We write $A \equiv B$ for syntactic identity. E.g. $A \cap B = A \cap B$, but $A \cap B \not\equiv B \cap A$.

12A.5. REMARK. All systems in Part III have the first five axioms and rules of Definition 12A.3. They differ in the extra axioms and rules and the set of atoms.

12A.6. PROPOSITION. The equivalence relation $=$ defined in Notation 12A.4(ii) is a congruence, i.e. $=$ is compatible with \cap and \rightarrow .

PROOF. By rules (trans), (incl_L), (incl_R) and (glb) one has

$$A = A' \& B = B' \Rightarrow (A \cap B) = (A' \cap B').$$

By rule (\rightarrow) one has

$$A = A' \& B = B' \Rightarrow (A \rightarrow B) = (A' \rightarrow B'). \blacksquare$$

12A.7. REMARK. The theory BCD can be seen as a structure with a pre-order

$$\text{BCD} = \langle \mathbb{T}, \leq, \cap, \rightarrow, \mathbb{U} \rangle.$$

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.

$$[A] \leq [B] \Leftrightarrow A \leq B.$$

By Proposition 12A.6, the operators \cap and \rightarrow can be defined on $[\mathbb{T}]$ by

$$\begin{aligned} [A] \cap [B] &\triangleq [A \cap B]; \\ [A] \rightarrow [B] &\triangleq [A \rightarrow B]. \end{aligned}$$

We obtain a type structure

$$[\mathbf{BCD}] \triangleq \langle [\mathbb{T}], \leq, \cap, \rightarrow, [\mathbb{U}] \rangle.$$

In this structure, $[\mathbb{U}]$ is the largest element (also called *top*) and $[A] \cap [B]$ is the *greatest lower bound* of $[A]$ and $[B]$.

12A.8. DEFINITION. (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 $\lambda_{\cap}^{\mathbf{BCD}}$ 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 $(x:A) \in \Gamma$
(\rightarrow I)	$\frac{\Gamma, x:A \vdash M : B}{\Gamma \vdash (\lambda x.M) : (A \rightarrow B)}$	if $(x:A) \notin \Gamma$
(\rightarrow E)	$\frac{\Gamma \vdash M : (A \rightarrow B) \quad \Gamma \vdash N : A}{\Gamma \vdash (MN) : B}$	
(\cap I)	$\frac{\Gamma \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash M : (A \cap B)}$	
(\leq)	$\frac{\Gamma \vdash M : A}{\Gamma \vdash M : B}$	if $A \leq_{\mathbf{BCD}} B$
(\mathbb{U})	$\Gamma \vdash M : \mathbb{U}$	

(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 $\lambda_{\cap}^{\mathbf{BCD}}$. In this case we write $\Gamma \vdash_{\cap}^{\mathbf{BCD}} M : A$ or just $\Gamma \vdash M : A$, if there is little danger of confusion.

12A.9. REMARK. All systems of type assignment in Part III have the first five axioms and rules of Definition 12A.8.

In Proposition 12A.11 we need the notions of admissible and derivable rule.

12A.10. DEFINITION. Consider an unspecified rule of the form (possibly with several assumptions or side-conditions)

$$\frac{\Gamma \vdash M : A}{\Gamma' \vdash M' : A'} \quad (R) \quad \text{if } p(\Gamma, M, A)$$

where $p(\Gamma, M, A)$ is a predicate on Γ, M, A .

(i) R is called *admissible* if one has

$$\Gamma \vdash M : A \text{ and } p(\Gamma, M, A) \Rightarrow \Gamma' \vdash M' : A'.$$

(ii) R is called *derivable* if $p(\Gamma, M, A)$ is always true and there is a derivation starting from $\Gamma \vdash M : A$ that ends in $\Gamma' \vdash M' : A'$.

A derivable rule is always admissible but the converse does not hold. If

$$\frac{\Gamma \vdash M : A}{\Gamma' \vdash M' : A'}$$

is a derivable rule, then for all Δ one has that

$$\frac{\Gamma \cup \Delta \vdash M : A}{\Gamma' \cup \Delta \vdash M' : A'}$$

is also derivable. Hence derivable rules are closed under theory extension. We will only be concerned with admissible and derivable rules for theories of type assignment. The statements of next proposition are easy to prove. For instance, derivability of the rules $(\cap E)$ follows immediately from rule (\leq) . The other proofs are left to the reader.

12A.11. PROPOSITION. (i) *The rules $(\cap E)$*

$$\frac{\Gamma \vdash M : (A \cap B)}{\Gamma \vdash M : A} \quad \frac{\Gamma \vdash M : (A \cap B)}{\Gamma \vdash M : B}$$

are derivable in $\lambda_{\cap}^{\text{BCD}}$.

(ii) *The following rules are admissible in the type assignment system $\lambda_{\cap}^{\text{BCD}}$.*

$(weakening)$ $\frac{\Gamma \vdash M : A \quad x \notin \Gamma}{\Gamma, x:B \vdash M : A}$
$(strengthening)$ $\frac{\Gamma, x:B \vdash M : A \quad x \notin FV(M)}{\Gamma \vdash M : A}$
(cut) $\frac{\Gamma, x:B \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash (M[x := N]) : A}$
$(\leq\text{-L})$ $\frac{\Gamma, x:B \vdash M : A \quad C \leq B}{\Gamma, x:C \vdash M : A}$
$(\rightarrow\text{-L})$ $\frac{\Gamma, y:B \vdash M : A \quad \Gamma \vdash N : C \quad x \notin \Gamma}{\Gamma, x:(C \rightarrow B) \vdash (M[y := xN]) : A}$
$(\cap\text{-L})$ $\frac{\Gamma, x:A \vdash M : B}{\Gamma, x:(A \cap C) \vdash M : B}$

Next theorem is a particular case of an “Inversion Lemma” (see Theorem 14A.1).

12A.12. THEOREM. In (i) assume $A \neq U$. Then

- (i) $\Gamma \vdash x : A \Leftrightarrow \exists B \in \mathbb{T}. [(x:B \in \Gamma \& B \leq A)]$.
- (ii) $\Gamma \vdash (MN) : A \Leftrightarrow \exists B \in \mathbb{T}. [\Gamma \vdash M : (B \rightarrow A) \& \Gamma \vdash N : B]$.
- (iii) $\Gamma \vdash \lambda x.M : A \Leftrightarrow \exists n > 0 \exists B_1, \dots, B_n, C_1, \dots, C_n \in \mathbb{T}$
 $\forall i \in \{1, \dots, n\}. [\Gamma, x:B_i \vdash M : C_i \&$
 $(B_1 \rightarrow C_1) \cap \dots \cap (B_n \rightarrow C_n) \leq A]$.
- (iv) $\Gamma \vdash \lambda x.M : B \rightarrow C \Leftrightarrow \Gamma, x:B \vdash M : C$.

12A.13. DEFINITION. Let R be a notion of reduction. Consider the following rules.

$(R\text{-red})$	$\frac{\Gamma \vdash M : A \quad M \rightarrow_R N}{\Gamma \vdash N : A}$
$(R\text{-exp})$	$\frac{\Gamma \vdash M : A \quad M \leftarrow_R N}{\Gamma \vdash N : A}$

General results of Section 14B imply next proposition. More in details, Corollary 14B.7(ii) implies admissibility of $(\beta\text{-red})$ and $(\beta\text{-exp})$, Corollary 14B.9 admissibility of $(\eta\text{-red})$. The negative result on $(\eta\text{-exp})$ follows from Theorem 14B.11(ii) together with Proposition 14B.12(ii).

12A.14. PROPOSITION. The rules $(\beta\text{-red})$, $(\beta\text{-exp})$ and $(\eta\text{-red})$ are admissible in $\lambda_{\cap}^{\text{BCD}}$. The rule $(\eta\text{-exp})$ is not.

The following result characterizes notions related to normalization in terms of type assignment in the system $\lambda_{\cap}^{\text{BCD}}$. The notation $U \notin A$ means that U does not occur in A . The result follows from Theorem 17B.15(i) and (ii).

12A.15. THEOREM. Let $M \in \Lambda^{\circ}$.

- (i) M has a head normal form $\Leftrightarrow \exists A \in \mathbb{T}. [A \neq_{\text{BCD}} U \& \vdash M : A]$.
- (ii) M has a normal form $\Leftrightarrow \exists A \in \mathbb{T}. [U \notin A \& \vdash M : A]$.

Let M be a lambda term. For the notion ‘approximant of M ’, see B[1984] Section 14.3. The *approximants of a term* M are roughly obtained from the Böhm tree $\text{BT}(M)$ of M by removing some branches and replacing these by a new symbol \perp . The set of approximants of M is denoted by $\mathcal{A}(M)$. We have e.g. for the fixed-point combinator Y

$$\mathcal{A}(\text{Y}) = \{\perp\} \cup \{\lambda f. f^n \perp \mid n > 0\}.$$

Approximants can be typed (for details see Section 17C) by extending the typing rules from terms to approximants. For example it will be shown that

$$\begin{aligned} &\vdash \perp : U \\ &\vdash \lambda f. f \perp : (U \rightarrow A_1) \rightarrow A_1 \\ &\vdash \lambda f. f(f \perp) : (U \rightarrow A_1) \cap (A_1 \rightarrow A_2) \rightarrow A_2 \\ &\dots \\ &\vdash \lambda f. f^n \perp : (U \rightarrow A_1) \cap (A_1 \rightarrow A_2) \cap \dots \cap (A_{n-1} \rightarrow A_n) \rightarrow A_n \\ &\dots \end{aligned}$$

The set of types of a term M will be shown to coincide 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. Theorem 17C.17 is the next theorem in a more general context.

12A.16. THEOREM. $\Gamma \vdash M : A \Leftrightarrow \exists P \in \mathcal{A}(M). \Gamma \vdash P : A$.

For example since $\lambda f.f^n\perp$ is for all n an approximant of Y , we have that all types of the shape $(\mathsf{U} \rightarrow A_1) \cap \dots \cap (A_{n-1} \rightarrow A_n) \rightarrow A_n$ can be derived for Y .

Finally the question whether an intersection type is inhabited is undecidable: the proof is the content of Section 17E, see Corollary 17E.32.

12A.17. THEOREM. *The set $\{A \in \mathbb{T} \mid \exists M \in \Lambda^\circ \vdash M : A\}$ is undecidable.*

12B. The filter model \mathcal{F}^{BCD}

12B.1. DEFINITION. (i) A *filter* over $\mathbb{T} = \mathbb{T}^{\text{BCD}}$ is a non-empty set $X \subseteq \mathbb{T}$ such that

- (1) $A \in X \ \& \ A \leq_{\text{BCD}} B \Rightarrow B \in X$;
- (2) $A, B \in X \Rightarrow (A \cap B) \in X$.

(ii) \mathcal{F}^{BCD} , or just \mathcal{F} , denotes the set of filters over \mathbb{T} .

12B.2. 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 $\uparrow A = \{B \mid A \leq B\}$.

Remember the definition of the notion of (ω -algebraic) complete lattice in 10A.4 and 10A.5. The following proposition easily follows from the above definitions.

12B.3. PROPOSITION. (i) $\mathcal{F} = \langle \mathcal{F}, \subseteq \rangle$ is an ω -algebraic complete lattice.

(ii) \mathcal{F} has as bottom element $\uparrow \mathsf{U}$ and as top element \mathbb{T} .

(iii) The compact elements of \mathcal{F} are exactly the principal filters.

A reflexive element of **ALG**, see Definitions 10A.7 and 10A.12, is also a model of the *untyped* λ -calculus in which the term interpretation is naturally defined as follows, see B[1984], Section 5.4.

12B.4. DEFINITION (Interpretation of terms). Let \mathcal{D} be reflexive via F, G .

(i) A *term environment* in \mathcal{D} is a map $\rho : \mathbf{Var} \rightarrow \mathcal{D}$. We denote by $\mathbf{Env}_{\mathcal{D}}$ the set of term environments.

(ii) If ρ is a term environment and $d \in \mathcal{D}$, then $\rho[x := d]$ is the term environment ρ' defined by

$$\begin{aligned} \rho'(y) &\triangleq \rho(y) && \text{if } y \not\equiv x; \\ \rho'(x) &\triangleq d. \end{aligned}$$

(iii) Given such ρ , the interpretation $\llbracket \cdot \rrbracket_{\rho}^{\mathcal{D}} : \Lambda \rightarrow \mathcal{D}$ is defined as follows.

$$\begin{aligned} \llbracket x \rrbracket_{\rho}^{\mathcal{D}} &\triangleq \rho(x); \\ \llbracket MN \rrbracket_{\rho}^{\mathcal{D}} &\triangleq F(\llbracket M \rrbracket_{\rho}^{\mathcal{D}})(\llbracket N \rrbracket_{\rho}^{\mathcal{D}}); \\ \llbracket \lambda x. M \rrbracket_{\rho}^{\mathcal{D}} &\triangleq G(\mathbb{A} d \in \mathcal{D}. \llbracket M \rrbracket_{\rho(x:=d)}^{\mathcal{D}}). \end{aligned}$$

(iv) Let $M, N \in \Lambda$. Then $M = N$, is *true in \mathcal{D}* , notation $\mathcal{D} \models M = N$, if

$$\forall \rho \in \text{Env}_{\mathcal{D}}. \llbracket M \rrbracket_{\rho}^{\mathcal{D}} = \llbracket N \rrbracket_{\rho}^{\mathcal{D}}.$$

Remember the notion of a λ -model \mathcal{D} given in Definition 3A.10.

12B.5. THEOREM. *Let \mathcal{D} be reflexive via F, G . Then \mathcal{D} is a λ -model, in particular for all $M, N \in \Lambda$*

$$\mathcal{D} \models (\lambda x.M)N = M[x := N].$$

We state now properties of \mathcal{F} which are implied by more general results proved in the following sections. More precisely Proposition 12B.6 follows from Corollary 16B.10(i), Theorem 12B.7 follows from Theorem 16B.7, and Theorem 12B.8 follows from Theorem 16B.18.

12B.6. PROPOSITION. *Define maps $F : \mathcal{F} \rightarrow [\mathcal{F} \rightarrow \mathcal{F}]$ and $G : [\mathcal{F} \rightarrow \mathcal{F}] \rightarrow \mathcal{F}$ by*

$$\begin{aligned} F(X)(Y) &\triangleq \uparrow\{B \mid \exists A \in Y. (A \rightarrow B) \in X\} \\ G(f) &\triangleq \uparrow\{A \rightarrow B \mid B \in f(\uparrow A)\}. \end{aligned}$$

Then \mathcal{F} is reflexive via F, G . Therefore \mathcal{F} is a model of the untyped λ -calculus.

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.

12B.7. THEOREM. *For all λ -terms M one has*

$$\llbracket M \rrbracket_{\rho}^{\mathcal{F}} = \{A \mid \exists \Gamma \models \rho. \Gamma \vdash M : A\},$$

where $\Gamma \models \rho$ iff for all $(x:B) \in \Gamma$ one has $B \in \rho(x)$.

Lastly we notice that all continuous functions are representable.

12B.8. THEOREM.

$$[\mathcal{F} \rightarrow \mathcal{F}] = \{f : \mathcal{F} \rightarrow \mathcal{F} \mid f \text{ is representable}\},$$

*where $f \in \mathcal{F} \rightarrow \mathcal{F}$ is called *representable* if for some $X \in \mathcal{F}$ one has*

$$\forall Y \in \mathcal{F}. f(Y) = F(X)(Y).$$

12C. Completeness of type assignment

12C.1. DEFINITION (*Interpretation of types*). Let \mathcal{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 \mathcal{D} is a map $\xi : \mathbb{A}_{\infty} \rightarrow \mathcal{P}(\mathcal{D})$.
- (ii) For $X, Y \in \mathcal{P}(\mathcal{D})$ define

$$(X \Rightarrow Y) \triangleq \{d \in \mathcal{D} \mid d \cdot X \subseteq Y\} \triangleq \{d \in \mathcal{D} \mid \forall x \in X. d \cdot x \in Y\}.$$

(iii) Given a type environment ξ , the interpretation $\llbracket \cdot \rrbracket_{\xi} : \mathbb{T} \rightarrow \mathcal{P}(\mathcal{D})$ is defined as follows.

$$\begin{aligned} \llbracket \mathbf{U} \rrbracket_{\xi}^{\mathcal{D}} &\triangleq \mathcal{D}; \\ \llbracket \alpha \rrbracket_{\xi}^{\mathcal{D}} &\triangleq \xi(\alpha), \quad \text{for } \alpha \in \mathbb{A}_{\infty}; \\ \llbracket A \rightarrow B \rrbracket_{\xi}^{\mathcal{D}} &\triangleq \llbracket A \rrbracket_{\xi}^{\mathcal{D}} \Rightarrow \llbracket B \rrbracket_{\xi}^{\mathcal{D}}; \\ \llbracket A \cap B \rrbracket_{\xi}^{\mathcal{D}} &\triangleq \llbracket A \rrbracket_{\xi}^{\mathcal{D}} \cap \llbracket B \rrbracket_{\xi}^{\mathcal{D}}. \end{aligned}$$

12C.2. DEFINITION (**Satisfaction**). (i) Given a λ -model \mathcal{D} , a term environment ρ and a type environment ξ one defines the following.

$$\begin{aligned}\mathcal{D}, \rho, \xi \models M : A &\iff [\![M]\!]_{\rho}^{\mathcal{D}} \in [\![A]\!]_{\xi}^{\mathcal{D}}. \\ \mathcal{D}, \rho, \xi \models \Gamma &\iff \mathcal{D}, \rho, \xi \models x : B, \quad \text{for all } (x:B) \in \Gamma.\end{aligned}$$

(ii) $\Gamma \models M : A \iff \forall \mathcal{D}, \rho, \xi. [\mathcal{D}, \rho, \xi \models \Gamma \Rightarrow \rho, \xi \models M : A].$

12C.3. THEOREM (**Soundness**).

$$\Gamma \vdash M : A \Rightarrow \Gamma \models M : A.$$

12C.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 Section 17A, where soundness and completeness are proved.

CHAPTER 13

TYPE ASSIGNMENT SYSTEMS

This chapter defines a family of systems $\lambda_{\cap}^{\mathcal{T}}$ 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 exemplary system λ_{\cap}^{BCD} presented in Chapter 12.

In Section 13A, we start by defining a set $\mathbb{T}^{\mathbb{A}}$ of intersection types similar to \mathbb{T}^{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 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 $U \leq A$, then also A is a universal element. So it is natural (but not strictly necessary) to require that U is the top element. The class of intersection type theories with a universal and top element is denoted by TT^U and the one without by TT^{-U} . For the (disjoint) union we write $TT = TT^U \cup TT^{-U}$.

Fig. 31 shows the thirteen specific examples of type theories that will be considered, where BCD is amongst them. These theories are denoted by names (respectively acronyms) 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 theories 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 in 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. 31 have the universal type U and the remaining three do not, i.e.

$$\begin{array}{ll} \text{Scott, Park, CDZ, HR, DHM, BCD, AO, Plotkin, Engeler, CDS} & \in TT^U \\ \text{HL, CDV, CD} & \in TT^{-U} \end{array}$$

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 $0 < 1 < U$.

\mathcal{T}	$\lambda_{\cap}^{\mathcal{T}}$	$\mathcal{F}^{\mathcal{T}}$	Reference
Scott	$\lambda_{\cap}^{\text{Scott}}$	$\mathcal{F}^{\text{Scott}}$	Scott [1972]
Park	$\lambda_{\cap}^{\text{Park}}$	$\mathcal{F}^{\text{Park}}$	Park [1976]
CDZ	$\lambda_{\cap}^{\text{CDZ}}$	\mathcal{F}^{CDZ}	Coppo, Dezani-Ciancaglini, and Zacchi [1987]
HR	$\lambda_{\cap}^{\text{HR}}$	\mathcal{F}^{HR}	Honsell and Ronchi Della Rocca [1992]
DHM	$\lambda_{\cap}^{\text{DHM}}$	\mathcal{F}^{DHM}	Dezani-Ciancaglini, Honsell, and Motohama [2005]
BCD	$\lambda_{\cap}^{\text{BCD}}$	\mathcal{F}^{BCD}	Barendregt, Coppo, and Dezani-Ciancaglini [1983]
AO	$\lambda_{\cap}^{\text{AO}}$	\mathcal{F}^{AO}	Abramsky and Ong [1993]
Plotkin	$\lambda_{\cap}^{\text{Plotkin}}$	$\mathcal{F}^{\text{Plotkin}}$	Plotkin [1993]
Engeler	$\lambda_{\cap}^{\text{Engeler}}$	$\mathcal{F}^{\text{Engeler}}$	Engeler [1981]
CDS	$\lambda_{\cap}^{\text{CDS}}$	\mathcal{F}^{CDS}	Coppo, Dezani-Ciancaglini, and Sallé [1979]
HL	$\lambda_{\cap}^{\text{HL}}$	\mathcal{F}^{HL}	Honsell and Lenisa [1999]
CDV	$\lambda_{\cap}^{\text{CDV}}$	\mathcal{F}^{CDV}	Coppo, Dezani-Ciancaglini, and Venneri [1981]
CD	$\lambda_{\cap}^{\text{CD}}$	\mathcal{F}^{CD}	Coppo and Dezani-Ciancaglini [1980]

FIGURE 31. Specific type theories, type assignment systems and filter models

In Section 13B we will assign types in $\mathbb{T}^{\mathbb{A}}$ to lambda terms in Λ . Given a type theory \mathcal{T} , we will derive assertions of the form $\Gamma \vdash_{\cap}^{\mathcal{T}} M : A$ where $M \in \Lambda$, $A \in \mathbb{T}^{\mathbb{A}}$ 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 $\lambda_{\cap}^{\mathcal{T}}$ and called *type assignment system over \mathcal{T}* . This can be seen as a mapping

$$\mathcal{T} \in \text{TT} \mapsto \text{set } \lambda_{\cap}^{\mathcal{T}} \text{ of typing rules (and axioms) parametric in } \mathcal{T}.$$

The parameter \mathcal{T} appears in rule (\leq) and axiom (U_{top}). The rule (\leq) states that a lambda term has type B if it has type A and $A \leq_{\mathcal{T}} B$. The axiom (U_{top}) states that all lambda terms have type U in case $\mathcal{T} \in \text{TT}^U$. In particular, the type assignments of the first ten type theories of Fig. 31 contain (U_{top}) and the remaining three do not.

The systems $\lambda_{\cap}^{\mathcal{T}}$ also share a set of non-parametric rules for assigning lambda terms to the types $(A \cap B)$ and $(A \rightarrow B)$. The particular use of intersection is that if a lambda term has both type A and type B , then it also has type $(A \cap B)$. The type $(A \rightarrow B)$ plays the same role as in the simply typed lambda calculus to cover the abstraction terms.

We have an infinite collection $\{\lambda_{\cap}^{\mathcal{T}} \mid \mathcal{T} \in \text{TT}\}$ 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. 31.
2. We can study general properties of $\lambda_{\cap}^{\mathcal{T}}$ that hold for all $\mathcal{T} \in \text{TT}$ as it will be done in Chapter 14.

In Section 13C we define the notion of intersection type structure

$$\mathcal{S} = \langle |\mathcal{S}|, \leq, \cap, \rightarrow \rangle$$

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 \rightarrow . Then we speak of a *compatible* type theory, having a corresponding *type structure*

$$[\mathcal{T}] = \langle [\mathbb{T}], \leq, \cap, \rightarrow \rangle.$$

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 13D we introduce the notion of *filter* over \mathcal{T} as a set of types closed under intersection \cap and pre-order \leq . If $\mathcal{T} \in \text{TT}^U$, then filters are non-empty and the smallest filter (ordered by subset inclusion) is the filter generated by $\{U\}$. If $\mathcal{T} \in \text{TT}^{-U}$, 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

$$\mathcal{T} \in \text{TT} \mapsto \text{set } \mathcal{F}^{\mathcal{T}} \text{ of filters over } \mathcal{T}.$$

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

$$\mathcal{T} \in \text{TT} \mapsto \mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle.$$

In Chapter 15 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 type structures and algebraic lattices.

In Chapter 16 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. 31 which appear in the literature. The first ten are models of the λ -calculus (when $\mathcal{T} \in \text{TT}^U$) and the remaining three are models of the λI -calculus (when $\mathcal{T} \in \text{TT}^{-U}$).

13A. Type theories

13A.1. DEFINITION. Let \mathbb{A} be a (usually countable, i.e. finite or countably infinite) set of symbols, called *(type) atoms*. The set of *intersection types* over \mathbb{A} , notation $\mathbb{T}_{\cap}^{\mathbb{A}}$ (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 simplified syntax.

$$\boxed{\mathbb{T} ::= \mathbb{A} \mid \mathbb{T} \rightarrow \mathbb{T} \mid \mathbb{T} \cap \mathbb{T}}$$

13A.2. REMARK. (i) The set \mathbb{A} varies in the applications of intersection types.

(ii) In Chapter 12 the set of intersection types \mathbb{T} was defined over the set of atoms $\mathbb{A}_{\infty}^U = \{c_0, c_1, \dots\} \cup \{U\}$. We write this as

$$\mathbb{A}^{BCD} = \mathbb{A}_{\infty}^U \text{ and } \mathbb{T}^{BCD} = \mathbb{T}^{\mathbb{A}_{\infty}^U}.$$

(iii) The following are some of the type atoms that will be used in different contexts.

c_0, c_1, c_2, \dots	indiscernible atoms
$0, 1, U$	atoms with special properties.

(iv) The atom \mathbf{U} is called *universe*. Its intention is to host all lambda terms. The intention of the special atom 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 Fig. 43 at page 572.

NOTATION. (i) Greek letters α, β, \dots range over arbitrary atoms in \mathbb{A} .

- (ii) The letters A, B, C, \dots range over types in $\mathbb{T}^{\mathbb{A}}$.
- (iii) Some papers on intersection types such as [Coppo, Dezani-Ciancaglini, and Sallé \[1979\]](#) 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, Honsell, and Alessi \[2003\]](#) use Ω for the universal type and ω and φ for 0 and 1.

13A.3. DEFINITION. (i) An *intersection type theory* over a set of type atoms \mathbb{A} is a set \mathcal{T} of sentences of the form $A \leq B$ (to be read: A sub 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)	$\frac{C \leq A \quad C \leq B}{C \leq A \cap B}$
(trans)	$\frac{A \leq B \quad B \leq C}{A \leq C}$

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, $\mathbf{TT}^{\mathbf{U}}$ and $\mathbf{TT}^{-\mathbf{U}}$.

13A.4. DEFINITION. (i) The collection of *intersection type theories with universe* \mathbf{U} notation $\mathbf{TT}^{\mathbf{U}}$, consists of the type theories \mathcal{T} over a \mathbb{A} such that $\mathbf{U} \in \mathbb{A}$ and \mathbf{U} is the *top*, i.e. the following holds for all types A

$$(\mathbf{U}_{\text{top}}) \quad A \leq \mathbf{U}$$

In the corresponding type structures treated in 13C the universal element will be visible in the signature.

(ii) The collection of *intersection type theories without universe*, notation $\mathbf{TT}^{-\mathbf{U}}$, consists of the type theories without such special atom \mathbf{U} that is a top.

13A.5. REMARK. (i) An intersection type theory \mathcal{T} can fail in three ways to be in $\mathbf{TT}^{\mathbf{U}}$: (a) there is a universe \mathbf{U} , assigned to all lambda terms M , but it is not a top; (b) there is a top \top , but it is not declared as a universal element; (c) there is neither special element \mathbf{U} , nor a top \top . In all these cases $\mathcal{T} \notin \mathbf{TT}^{\mathbf{U}}$.

(ii) The intuition behind the special atom \mathbf{U} , and its name, will become clear in the next section 13B on type assignment. The types in a TT will be assigned to untyped lambda terms. Arrow types like $A \rightarrow B$ will be assigned to abstraction terms of the form $\lambda x.M$. For $\mathcal{T} \in \text{TT}^{\mathbf{U}}$ we will postulate the assignment

$$\Gamma \vdash M : \mathbf{U},$$

for all Γ and all $M \in \Lambda$. So \mathbf{U} will be a *universal type*, i.e. a type assigned to all terms. Another rule of type assignment will be

$$\frac{\Gamma \vdash M : A \quad A \leq B}{\Gamma \vdash M : B}.$$

Hence if $\mathbf{U} \leq A$, then also A will be a universal type. Therefore, for type theories in which we wish to have both a universe \mathbf{U} and a top \top , we **have chosen in 13A.4 that $\mathbf{U} = \top$** , i.e. that for all types A

$$A \leq \mathbf{U}.$$

There will be $\mathcal{T} \in \text{TT}$ 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 $\lambda_{\cap}^{\text{Krivine}^{\mathbf{U}}}$ of Krivine, see Exercise 13E.10, is such an example.

13A.6. REMARK. (i) In this Part of the book \mathcal{T} ranges over elements of TT.

(ii) If $\mathcal{T} \in \text{TT}$ over \mathbb{A} , then we also write $\mathbb{T}^{\mathcal{T}} = \mathbb{T}_{\cap}^{\mathbb{A}}$ and $\mathbb{A}^{\mathcal{T}} = \mathbb{A}$.

13A.7. REMARK. Most $\mathcal{T} \in \text{TT}$ have some extra axioms or rules, the above set in Definition 13A.3 being the minimum requirement. For example the theory BCD over $\mathbb{A} = \mathbb{A}_{\infty}^{\mathbf{U}}$, introduced in Chapter 12, is a TT $^{\mathbf{U}}$ and has the extra axioms $(\mathbf{U} \rightarrow)$ and $(\rightarrow \cap)$ and rule (\rightarrow) .

13A.8. NOTATION. Let $\mathcal{T} \in \text{TT}$. 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. $A \cap B = A \cap B$, 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 $A_1 \cap \dots \cap A_n$ to denote the intersection of a sequence of n types A_i for $i \in \{1, \dots, n\}$ associating to the right. If $n = 3$ then $A_1 \cap \dots \cap A_n$ denotes $(A_1 \cap (A_2 \cap A_3))$. In case $n = 0$ and $\mathcal{T} \in \text{TT}^{\mathbf{U}}$, we intend that the sequence is empty and $A_1 \cap \dots \cap A_n$ denotes \mathbf{U} .

(viii) For a finite $I = \{1, \dots, n\}$ we may also use the notation $\bigcap_{i \in I} A_i$ to denote $A_1 \cap \dots \cap A_n$.

13A.9. PROPOSITION. *The following rule is derivable.*

(mon)	$\frac{A \leq A' \quad B \leq B'}{A \cap B \leq A' \cap B'}$
-------	--

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 \rightarrow we define the following.

13A.10. DEFINITION. \mathcal{T} is called *compatible* if the following rule is admissible.

$$\boxed{(\rightarrow^=) \quad \frac{A = A' \quad B = B'}{(A \rightarrow B) =_{\mathcal{T}} (A' \rightarrow B')}}$$

This means $A =_{\mathcal{T}} A' \& B =_{\mathcal{T}} B' \Rightarrow (A \rightarrow B) =_{\mathcal{T}} (A' \rightarrow B')$. One way to insure this is to adopt $(\rightarrow^=)$ as rule determining \mathcal{T} .

13A.11. LEMMA. (i) For any \mathcal{T} one has $A \cap B =_{\mathcal{T}} B \cap A$.

(ii) If \mathcal{T} is compatible, then $(A \cap B) \rightarrow C =_{\mathcal{T}} (B \cap A) \rightarrow C$.

PROOF. (i) By (incl_L), (incl_R) and (glb).

(ii) By (i). ■

13A.12. REMARK. Similarly to Remark 12A.7 any $\mathcal{T} \in \text{TT}$ can be seen as a structure with a pre-order $\mathcal{T} = \langle \mathbb{T}, \leq, \cap, \rightarrow \rangle$. This means that \leq is reflexive and transitive, but not necessarily anti-symmetric

$$A \leq B \& B \leq A \text{ does not always imply } A = B.$$

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 13A.9, \cap is always well defined on $[\mathbb{T}]$ by $[A] \cap [B] = [A \cap B]$. To ensure that \rightarrow is well defined by $[A] \rightarrow [B] = [A \rightarrow B]$, we need to require that $=$ is compatible with \rightarrow . This is the case if \mathcal{T} is compatible and one obtains a type structure

$$[\mathcal{T}] = \langle [\mathbb{T}], \leq, \cap, \rightarrow \rangle.$$

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 $\mathcal{T} \in \text{TT}^U$, then $[\mathcal{T}]$ can be enriched to a type structure with universe $[\mathbb{U}]$ of the form

$$[\mathcal{T}] = \langle [\mathbb{T}], \leq, \cap, \rightarrow, [\mathbb{U}] \rangle.$$

This will be done in Section 13C.

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})	$(U \rightarrow 0) = 0$
(0 _{Park})	$(0 \rightarrow 0) = 0$
(01)	$0 \leq 1$
(1 \rightarrow 0)	$(1 \rightarrow 0) = 0$
(0 \rightarrow 1)	$(0 \rightarrow 1) = 1$
(I)	$(1 \rightarrow 1) \cap (0 \rightarrow 0) = 1$

Axiom schemes	
(U _{top})	$A \leq U$
(U \rightarrow)	$U \leq (A \rightarrow U)$
(U _{lazy})	$(A \rightarrow B) \leq (U \rightarrow U)$
($\rightarrow \cap$)	$(A \rightarrow B) \cap (A \rightarrow C) \leq A \rightarrow B \cap C$
($\rightarrow \cap =$)	$(A \rightarrow B) \cap (A \rightarrow C) = A \rightarrow B \cap C$

Rule schemes	
(\rightarrow)	$\frac{A' \leq A \quad B \leq B'}{(A \rightarrow B) \leq (A' \rightarrow B')}$
($\rightarrow =$)	$\frac{A' = A \quad B = B'}{(A \rightarrow B) = (A' \rightarrow B')}$

FIGURE 32. Possible Axioms and Rules concerning \leq .

13A.13. REMARK. (i) The axiom scheme (U_{top}) states that the universe U is a top element.

(ii) In the presence of (\rightarrow) the axiom-scheme $(U \rightarrow)$ is equivalent with the axiom $U \leq (U \rightarrow U)$. Also in that case the axiom-scheme $(\rightarrow \cap)$ is equivalent with $(\rightarrow \cap =)$. See Exercise 13E.1.

13A.14. DEFINITION. In Fig. 33 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 13A.3) that satisfies the rules and axioms of \mathcal{T} shown in Fig. 33.

\mathcal{T}	$\mathbb{A}^{\mathcal{T}}$	Rules	Axiom Schemes	Axioms
Scott	$\{\mathbf{U}, 0\}$	(\rightarrow)	$(\rightarrow \cap), (\mathbf{U}_{\text{top}}), (\mathbf{U} \rightarrow)$	(0_{Scott})
Park	$\{\mathbf{U}, 0\}$	(\rightarrow)	$(\rightarrow \cap), (\mathbf{U}_{\text{top}}), (\mathbf{U} \rightarrow)$	(0_{Park})
CDZ	$\{\mathbf{U}, 1, 0\}$	(\rightarrow)	$(\rightarrow \cap), (\mathbf{U}_{\text{top}}), (\mathbf{U} \rightarrow)$	$(01), (1 \rightarrow 0), (0 \rightarrow 1)$
HR	$\{\mathbf{U}, 1, 0\}$	(\rightarrow)	$(\rightarrow \cap), (\mathbf{U}_{\text{top}}), (\mathbf{U} \rightarrow)$	$(01), (1 \rightarrow 0), (\mathbf{I})$
DHM	$\{\mathbf{U}, 1, 0\}$	(\rightarrow)	$(\rightarrow \cap), (\mathbf{U}_{\text{top}}), (\mathbf{U} \rightarrow)$	$(01), (0 \rightarrow 1), (0_{\text{Scott}})$
BCD	$\mathbb{A}_{\infty}^{\mathbf{U}}$	(\rightarrow)	$(\rightarrow \cap), (\mathbf{U}_{\text{top}}), (\mathbf{U} \rightarrow)$	
AO	$\{\mathbf{U}\}$	(\rightarrow)	$(\rightarrow \cap), (\mathbf{U}_{\text{top}}), (\mathbf{U}_{\text{lazy}})$	
Plotkin	$\{\mathbf{U}, 0\}$	$(\rightarrow =)$	$(\mathbf{U}_{\text{top}})$	—
Engeler	$\mathbb{A}_{\infty}^{\mathbf{U}}$	$(\rightarrow =)$	$(\rightarrow \cap =), (\mathbf{U}_{\text{top}}), (\mathbf{U} \rightarrow)$	—
CDS	$\mathbb{A}_{\infty}^{\mathbf{U}}$	—	$(\mathbf{U}_{\text{top}})$	—
HL	$\{1, 0\}$	(\rightarrow)	$(\rightarrow \cap)$	$(01), (0 \rightarrow 1), (1 \rightarrow 0)$
CDV	\mathbb{A}_{∞}	(\rightarrow)	$(\rightarrow \cap)$	—
CD	\mathbb{A}_{∞}	—	—	—

FIGURE 33. Various type theories

13A.15. REMARK. (i) Note that CDS and CD are non-compatible, while the other eleven type theories are compatible.

(ii) The first ten type theories of Fig. 33 belong to $\text{TT}^{\mathbf{U}}$ and the last three to $\text{TT}^{-\mathbf{U}}$. In Lemma 13A.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 13A.22(ii) and (iii).

13A.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 17B and 17D), and for describing different λ -models (see Section 16C).

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 \mathcal{D} (the interpretation of the universe \mathbf{U}), which is endowed with a (partial) application $\cdot : \mathcal{D} \times \mathcal{D} \rightarrow \mathcal{D}$, and interpret \cap as set-theoretic intersection, \leq as set inclusion, and give \rightarrow the *realizability interpretation* $\llbracket A \rrbracket_{\xi} \subseteq \mathcal{D}$ for each type A . One starts by interpreting types in \mathbb{T}_{\rightarrow} .

$$\llbracket \alpha \rrbracket_{\xi} \triangleq \xi(\alpha), \text{ for } \alpha \not\equiv \mathbf{U};$$

$$\llbracket A \rightarrow B \rrbracket_{\xi} \triangleq (\llbracket A \rrbracket_{\xi} \Rightarrow \llbracket B \rrbracket_{\xi}) \triangleq \{d \in \mathcal{D} \mid d \cdot \llbracket A \rrbracket_{\xi} \subseteq \llbracket B \rrbracket_{\xi}\}.$$

This semantics, due to Scott, can be extended to intersection types by

$$\llbracket \mathbf{U} \rrbracket_{\xi} \triangleq \mathcal{D};$$

$$\llbracket A \cap B \rrbracket_{\xi} \triangleq \llbracket A \rrbracket_{\xi} \cap \llbracket B \rrbracket_{\xi}.$$

Given the right TT and right domain the following holds.

$$A \leq B \Leftrightarrow \forall \xi. \llbracket A \rrbracket_\xi \subseteq \llbracket B \rrbracket_\xi.$$

This type of semantics will be studied in Section 17A.

The type $U \rightarrow U$ is in the set-theoretic interpretation the set of functions which applied to an arbitrary element return again an arbitrary element. Then axiom scheme $(U \rightarrow)$ expresses the fact that all the objects in our domain of discourse are total functions, i.e. that U is equal to $A \rightarrow U$, hence $A \rightarrow U = B \rightarrow U$ for all A, B (Barendregt, Coppo, and Dezani-Ciancaglini [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 $(U \rightarrow)$. One still may postulate the weaker property (U_{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 $(\rightarrow \cap)$, 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 of B and C (i.e. into $B \cap C$), see Barendregt, Coppo, and Dezani-Ciancaglini [1983].

Rule (\rightarrow) 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, Coppo, and Dezani-Ciancaglini [1983].

The rule $(\rightarrow \cap^=)$ is similar to the rule $(\rightarrow \cap)$. It captures properties of the graph models for the untyped lambda calculus, see Plotkin [1975] and Engeler [1981], as we shall discuss in Section 16C.

For Scott, Park, CDZ, HR, DHM, the axioms express peculiar properties of D_∞ -like inverse limit models (see Section 16C). For Park, CDZ, HR, DHM as well as for HL, the axioms also express properties of subsets of λ -terms (see Fig. 43 at page 572 and Proposition 17B.13).

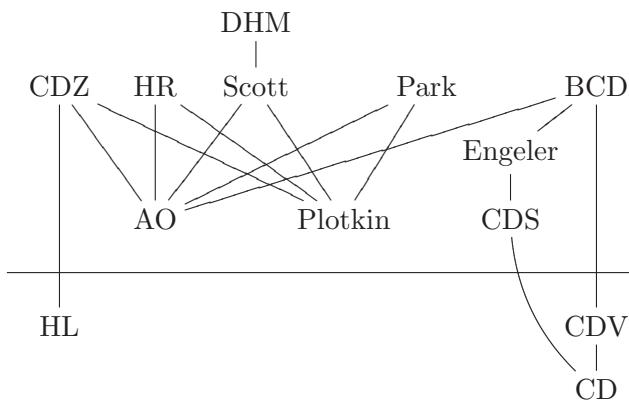


FIGURE 34. Inclusion among some intersection type theories.

13A.17. REMARK. In Fig. 34 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 13E.15 it is shown that the inclusions are strict. Above the horizontal line we find the elements of TT^U , below of TT^{-U} .

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.

13A.18. DEFINITION. We define special subclasses of TT .

Class	Defining axiom(-scheme)(s) or rule(-scheme)(s)
<i>graph</i>	$(\rightarrow^=), (\mathbf{U}_{\text{top}})$
<i>lazy</i>	$(\rightarrow), (\rightarrow \cap), (\mathbf{U}_{\text{top}}), (\mathbf{U}_{\text{lazy}})$
<i>natural</i>	$(\rightarrow), (\rightarrow \cap), (\mathbf{U}_{\text{top}}), (\mathbf{U} \rightarrow)$
<i>proper</i>	$(\rightarrow), (\rightarrow \cap)$

13A.19. NOTATION. The sets of graph, lazy, natural and proper type theories are denoted by respectively GTT^U , LTT^U , NTT^U and PTT .

The following subset inclusions are easily deduced from their definition.

$$\text{NTT}^U \subset \text{LTT}^U \subset \text{GTT}^U \subset \text{TT}^U \subset \text{TT}$$

and $\text{LTT}^U \subset \text{PTT}$.

13A.20. REMARK. The type theories of Fig. 33 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 NTT^U is the smallest set that contains CDZ.

Some properties about specific TTs

Results about proper type theories

All type theories of Fig. 33 are proper, except CD, CDS, Plotkin, Engeler.

13A.21. PROPOSITION. Let \mathcal{T} be proper. Then we have

- (i) $(A \rightarrow B) \cap (A' \rightarrow B') \leq (A \cap A') \rightarrow (B \cap B');$
- (ii) $(A_1 \rightarrow B_1) \cap \dots \cap (A_n \rightarrow B_n) \leq (A_1 \cap \dots \cap A_n) \rightarrow (B_1 \cap \dots \cap B_n);$
- (iii) $(A \rightarrow B_1) \cap \dots \cap (A \rightarrow B_n) = A \rightarrow (B_1 \cap \dots \cap B_n).$

$$\begin{aligned} \text{PROOF. (i)} \quad (A \rightarrow B) \cap (A' \rightarrow B') &\leq ((A \cap A') \rightarrow B) \cap ((A \cap A') \rightarrow B') \\ &\leq (A \cap A') \rightarrow (B \cap B'), \end{aligned}$$

by respectively (\rightarrow) and $(\rightarrow \cap)$.

(ii) Similarly (i.e. by induction on $n > 1$, using (i) for the induction step).

(iii) By (ii) one has $(A \rightarrow B_1) \cap \dots \cap (A \rightarrow B_n) \leq A \rightarrow (B_1 \cap \dots \cap B_n)$. For \geq use (\rightarrow) to show that $A \rightarrow (B_1 \cap \dots \cap B_n) \leq (A \rightarrow B_i)$, 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 Fig. 33

13A.22. LEMMA. (i) 1 is the top and 0 the bottom element in HL.

- (ii) CDV has no top element.
- (iii) CD has no top element.

PROOF. (i) By induction on the generation of \mathbb{T}^{HL} one shows that $0 \leq A \leq 1$ for all $A \in \mathbb{T}^{\text{HL}}$.

(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_{CDV} that $A \leq_{\text{CDV}} B \Rightarrow B \in \mathcal{B}_\alpha$. Hence if $\alpha \leq_{\text{CDV}} B$, then $B \in \mathcal{B}_\alpha$. Since \mathcal{B}_{α_1} and \mathcal{B}_{α_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. ■

13A.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 TT^U .

In the following Lemmas 13A.24-13A.28 we study the positions of the atoms 0, and 1 in the TTs introduced in Fig. 33. The principal result is that $0 < 1$ in HL and, as far as applicable,

$$0 < 1 < U,$$

in the theories Scott, Park, CDZ, HR, DHM and Plotkin.

13A.24. LEMMA. Let $\mathcal{T} \in \{\text{Scott, Park, CDZ, HR, DHM, BCD, Engeler}\}$. Define inductively the followin collection of types.

$$\boxed{\mathcal{B} ::= U \mid \mathbb{T}^{\mathcal{T}} \rightarrow \mathcal{B} \mid \mathcal{B} \cap \mathcal{B}}$$

Then we have $\mathcal{B} = \{A \in \mathbb{T}^{\mathcal{T}} \mid A =_{\mathcal{T}} U\}$.

PROOF. By induction on the generation of $A \leq_{\mathcal{T}} B$ one proves that \mathcal{B} is closed upwards. This gives $U \leq_{\mathcal{T}} A \Rightarrow A \in \mathcal{B}$.

By induction on the definition of \mathcal{B} one shows, using (\rightarrow) or $(\rightarrow^=)$, (U_{top}) and $(U \rightarrow)$, that $A \in \mathcal{B} \Rightarrow A =_{\mathcal{T}} U$.

Therefore

$$A =_{\mathcal{T}} U \Leftrightarrow A \in \mathcal{B}. \blacksquare$$

13A.25. LEMMA. For $\mathcal{T} \in \{\text{AO, Plotkin}\}$ define inductively

$$\boxed{\mathcal{B} ::= U \mid \mathcal{B} \cap \mathcal{B}}$$

Then $\mathcal{B} = \{A \in \mathbb{T}^{\mathcal{T}} \mid A =_{\mathcal{T}} \mathbf{U}\}$, hence $\mathbf{U} \rightarrow \mathbf{U} \neq_{\mathcal{T}} \mathbf{U}$.

PROOF. Similar to the proof of 13A.22, but easier. ■

13A.26. LEMMA. For $\mathcal{T} \in \{\text{CDZ}, \text{HR}, \text{DHM}\}$ define by mutual induction

$$\boxed{\begin{array}{lcl} \mathcal{B} & ::= & 1 \mid \mathbf{U} \mid \mathbb{T}^{\mathcal{T}} \rightarrow \mathcal{B} \mid \mathcal{H} \rightarrow \mathbb{T}^{\mathcal{T}} \mid \mathcal{B} \cap \mathcal{B} \\ \mathcal{H} & ::= & 0 \mid \mathcal{B} \rightarrow \mathcal{H} \mid \mathcal{H} \cap \mathbb{T}^{\mathcal{T}} \mid \mathbb{T}^{\mathcal{T}} \cap \mathcal{H}. \end{array}}$$

Then

$$\begin{aligned} 1 \leq B &\Rightarrow B \in \mathcal{B}, \\ A \leq 0 &\Rightarrow A \in \mathcal{H}. \end{aligned}$$

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. ■

13A.27. LEMMA. We work with the theory HL.

(i) Define by mutual induction

$$\boxed{\begin{array}{lcl} \mathcal{B} & ::= & 1 \mid \mathcal{H} \rightarrow \mathcal{B} \mid \mathcal{B} \cap \mathcal{B} \\ \mathcal{H} & ::= & 0 \mid \mathcal{B} \rightarrow \mathcal{H} \mid \mathcal{H} \cap \mathbb{T}^{\text{HL}} \mid \mathbb{T}^{\text{HL}} \cap \mathcal{H} \end{array}}$$

Then

$$\begin{aligned} \mathcal{B} &= \{A \in \mathbb{T}^{\text{HL}} \mid A =_{\text{HL}} 1\}; \\ \mathcal{H} &= \{A \in \mathbb{T}^{\text{HL}} \mid A =_{\text{HL}} 0\} \end{aligned}$$

(ii) $0 \neq_{\text{HL}} 1$ and hence $0 <_{\text{HL}} 1$.

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 13A.22(i),

$$(B \in \mathcal{B} \Rightarrow B = 1) \& (A \in \mathcal{H} \Rightarrow A = 0).$$

Now the assertion follows immediately.

(ii) By (i). ■

13A.28. PROPOSITION. In HL we have $0 < 1$. For the other members of Fig. 33 as far as applicable

$$0 < 1 < \mathbf{U}.$$

More precisely, in HL one has

$$(i) \quad 0 < 1.$$

In CDZ, HR, DHM one has

$$(ii) \quad 0 < 1 < \mathbf{U}.$$

PROOF. (i) By rule (01) and Lemma 13A.27.

(ii) By rules (01), $(\mathbf{U}_{\text{top}})$ and Lemmas 13A.24-13A.26. ■

13B. Type assignment

Assignment of types from type theories

In this subsection we define an infinite collection $\{\lambda_n^T \mid T \in \text{TT}\}$ of type assignment systems by giving a *uniform* set of typing rules parametric in T .

13B.1. DEFINITION. Let $T \in \text{TT}$.

- (i) A *T -statement* is of the form $M : A$, with $M \in \Lambda$ and $A \in \text{TT}^T$.
- (ii) A *T -declaration* is a T -statement of the form $x : A$.
- (iii) A *T -basis* Γ is a finite set of T -declarations, with all term variables distinct.
- (iv) A *T -assertion* is of the form

$$\Gamma \vdash M : A,$$

where $M : A$ is a T -statement and Γ is a T -basis.

13B.2. REMARK. Let $M : A$ be a T -statement.

- (i) The term M is called the *subject* of this statement.
- (ii) The type A is called its *predicate*.

13B.3. DEFINITION. Let $T \in \text{TT}$. The *type assignment* system λ_n^T derives T -assertions by the following axioms and rules.

(Ax)	$\Gamma \vdash x : A$	if $(x:A) \in \Gamma$
(\rightarrow I)	$\frac{\Gamma, x:A \vdash M : B}{\Gamma \vdash \lambda x.M : A \rightarrow B}$	
(\rightarrow E)	$\frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$	
(\cap I)	$\frac{\Gamma \vdash M : A \quad \Gamma \vdash M : B}{\Gamma \vdash M : A \cap B}$	
(\leq)	$\frac{\Gamma \vdash M : A \quad A \leq_T B}{\Gamma \vdash M : B}$	
(U)	$\Gamma \vdash M : \mathbf{U}$	if $T \in \text{TT}^U$

Note that the parameter T appears only in the last two rules.

13B.4. NOTATION. (i) We write $\Gamma \vdash_n^T M : A$ if $\Gamma \vdash M : A$ is derivable in λ_n^T .

(ii) The assertion \vdash_n^T may also be written as \vdash_n^T , as \vdash_n or simply as \vdash , if there is little danger of confusion.

(iii) λ_n^T may be denoted simply by λ_n .

13B.5. REMARK. Given a type theory T , the following two options are mutually exclusive: either the type assignment system λ_n^T contains the axiom (U_{top}) or it does not. For the specific type theories in Fig. 33, 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 (U_{top}) , since they all belong to TT^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 TT^{U} .

13B.6. REMARK. As suggested in the introduction of Chapter 12, the type assignment systems with the axiom (U_{top}) are closed under β -expansions and can be used to construct models of the $\lambda\mathbf{K}$ -calculus. On the other hand the systems without this axiom are closed under $\beta\mathbf{I}$ -expansions (not necessarily under β) and can be used to construct models specifically for the $\lambda\mathbf{I}$ -calculus.

13B.7. EXAMPLE. The statements in this Example will be proved in Exercises 14C.7 and 14C.8. Define $\omega \triangleq \lambda x.xx$, $\Omega \triangleq \omega\omega$, $V \triangleq \lambda yz.\mathbf{K}z(yz)$ and $K_* \triangleq \lambda yz.z$. Then $V \rightarrow_{\beta} K_*$. We have the following for arbitrary $A, B \in \mathbb{T}$.

- (i) For all $\mathcal{T} \in \text{TT}$.

$$\vdash_{\cap}^{\mathcal{T}} \omega : A \cap (A \rightarrow B) \rightarrow B.$$

$$\vdash_{\cap}^{\mathcal{T}} V : (B \rightarrow A) \rightarrow B \rightarrow B.$$

$$\vdash_{\cap}^{\mathcal{T}} K_* : A \rightarrow B \rightarrow B.$$

- (ii) For some $\mathcal{T} \in \text{TT}^{\text{U}}$, for example for CDV and CD, one has

$$\nvdash_{\cap}^{\mathcal{T}} V : \alpha \rightarrow \beta \rightarrow \beta.$$

Conclude that

$$M \rightarrow_{\beta} N \& \Gamma \vdash_{\cap}^{\mathcal{T}} N : A \not\Rightarrow \Gamma \vdash_{\cap}^{\mathcal{T}} M : A,$$

i.e. in the absence of U subject expansion fails, even if the expanded term is typable, as observed in [van Bakel \[1993\]](#); this phenomenon also occurs in λ_{\rightarrow} .

- (iii) For some $\mathcal{T} \in \text{TT}^{\text{U}}$, for example for HL, CDV and CD, one has for all A

$$\nvdash_{\cap}^{\mathcal{T}} K\mathbf{I}\Omega : A;$$

$$\nvdash_{\cap}^{\mathcal{T}} \Omega : A.$$

- (iv) $\nvdash^{\text{CD}} I : ((\alpha \cap \beta) \rightarrow \gamma) \rightarrow ((\beta \cap \alpha) \rightarrow \gamma)$.

- (v) Let $\mathcal{T} \in \text{TT}^{\text{U}}$. Then

$$\vdash_{\cap}^{\mathcal{T}} \Omega : \text{U}.$$

$$\vdash_{\cap}^{\mathcal{T}} K\mathbf{I}\Omega : (A \rightarrow A).$$

$$\vdash_{\cap}^{\mathcal{T}} V : A \rightarrow B \rightarrow B.$$

13B.8. DEFINITION. Define the rules $(\cap E)$

$$\frac{\Gamma \vdash M : (A \cap B)}{\Gamma \vdash M : A} \quad \frac{\Gamma \vdash M : (A \cap B)}{\Gamma \vdash M : B}$$

Notice that these rules are derived in $\lambda_{\cap}^{\mathcal{T}}$ for all \mathcal{T} .

13B.9. LEMMA. For $\mathcal{T} \in \text{TT}^{\text{U}}$ one has the following.

- (i) $\Gamma \vdash_{\cap}^{\mathcal{T}} M : A \Rightarrow \text{FV}(M) \subseteq \text{dom}(\Gamma)$.

- (ii) $\Gamma \vdash_{\cap}^{\mathcal{T}} M : A \Rightarrow (\Gamma \upharpoonright \text{FV}(M)) \vdash M : A$.

PROOF. (i), (ii) By straightforward induction on the derivation. ■

Notice that $\Gamma \vdash M : A \Rightarrow \text{FV}(M) \subseteq \text{dom}(\Gamma)$ does not hold for $\mathcal{T} \in \text{TT}^{\text{U}}$, since by axiom (U_{top}) we have $\vdash_{\cap}^{\mathcal{T}} M : \text{U}$ for all M .

Admissible rules

13B.10. PROPOSITION. *The following rules are admissible in λ_{\cap}^T .*

$(weakening)$	$\frac{\Gamma \vdash M : A \quad x \notin \Gamma}{\Gamma, x:B \vdash M : A};$
$(strengthening)$	$\frac{\Gamma, x:B \vdash M : A \quad x \notin FV(M)}{\Gamma \vdash M : A};$
(cut)	$\frac{\Gamma, x:B \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash (M[x := N]) : A};$
$(\leq\text{-L})$	$\frac{\Gamma, x:B \vdash M : A \quad C \leq_T B}{\Gamma, x:C \vdash M : A};$
$(\rightarrow\text{-L})$	$\frac{\Gamma, y:B \vdash M : A \quad \Gamma \vdash N : C \quad x \notin \Gamma}{\Gamma, x:(C \rightarrow B) \vdash (M[y := xN]) : A};$
$(\cap\text{-L})$	$\frac{\Gamma, x:A \vdash M : B}{\Gamma, x:(A \cap C) \vdash M : B}.$

FIGURE 35. Various admissible rules.

PROOF. By straightforward 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 T -bases are additive. We could have taken, just as well, a sequent style presentation and replace rule $(\rightarrow\text{-E})$ with the three rules $(\rightarrow\text{-L})$, $(\cap\text{-L})$ and (cut) occurring in Proposition 13B.10, see [Barbanera, Dezani-Ciancaglini, and de'Liguoro \[1995\]](#), [Barendregt and Ghilezan \[2000\]](#). Next to this we could have formulated the rules so that T -bases “multiply”. Notice that because of the presence of the type constructor \cap , a special notion of *multiplication of T -bases* is useful.

13B.11. DEFINITION (Multiplication of T -bases).

$$\begin{aligned} \Gamma \uplus \Gamma' &\triangleq \{x: A \cap B \mid x: A \in \Gamma \text{ and } x: B \in \Gamma'\} \\ &\cup \{x: A \mid x: A \in \Gamma \text{ and } x \notin \Gamma'\} \\ &\cup \{x: B \mid x: B \in \Gamma' \text{ and } x \notin \Gamma\}. \blacksquare \end{aligned}$$

Accordingly we define:

$$\Gamma \underline{\oplus} \Gamma' \iff \exists \Gamma''. \Gamma \uplus \Gamma'' = \Gamma'.$$

For example, $\{x:A, y:B\} \uplus \{x:C, z:D\} = \{x:A \cap C, y:B, z:D\}$.

13B.12. PROPOSITION. *The following rules are admissible in all $\lambda_{\cap}^{\mathcal{T}}$.*

$(multiple \text{ } weakening)$	$\frac{\Gamma_1 \vdash M : A}{\Gamma_1 \uplus \Gamma_2 \vdash M : A}$
$(relevant \rightarrow E)$	$\frac{\Gamma_1 \vdash M : A \rightarrow B \quad \Gamma_2 \vdash N : A}{\Gamma_1 \uplus \Gamma_2 \vdash MN : B}$
$(relevant \cap I)$	$\frac{\Gamma_1 \vdash M : A \quad \Gamma_2 \vdash M : B}{\Gamma_1 \uplus \Gamma_2 \vdash M : A \cap B}$

PROOF. By straightforward induction on derivations. ■

In Exercise 14C.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” lemmas for $\lambda_{\cap}^{\mathcal{T}}$. 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 $\lambda_{\cap}^{\mathcal{T}}$, 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 $(relevant \rightarrow E)$ 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, Lafont, and Taylor \[1989\]](#). 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

13B.13. DEFINITION. The type theory [EHR](#) is defined with $A^{\text{EHR}} = \{V\}$ and the extra rule (\rightarrow) and axioms $(\rightarrow \cap)$ and

$$A \rightarrow B \leq V.$$

The type assignment system $\lambda_{\cap V}^{\text{EHR}}$ is defined by the axiom and rules of Definition 13B.3 with the extra axiom

$$(V) \quad \Gamma \vdash (\lambda x.M) : V.$$

The type theory EHR has a top, namely V , but it is not an element of TT^U . Hence $\lambda_{\cap V}^{\text{EHR}}$ does not contain the axiom (U_{top}) . Note also that the axiom (V) is different from (U_{top}) . This type assignment system is suitable for modelling the call-by-value λ -calculus: it has been extensively studied in [Ronchi Della Rocca and Paolini \[2004\]](#).

13C. Type structures

Intersection type structures

Remember that a type algebra \mathcal{A} , see Definition 7A.1, is of the form $\mathcal{A} = \langle |\mathcal{A}|, \rightarrow \rangle$, i.e. just an arbitrary set $|\mathcal{A}|$ with a binary operation \rightarrow on it.

13C.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 . MSL^U is the set of meet semi-lattices.

(ii) A *meet semi-lattice with universe* is a similar structure

$$\mathcal{M} = \langle |\mathcal{M}|, \leq, \cap, U \rangle,$$

with U the (unique) top element of \mathcal{M} . MSL^U is the set of meet semi-lattices with universe.

(iii) $\text{MSL} = \text{MSL}^U \cup \text{MSL}^{-U}$ is the set of meet semi-lattices with or without universe.

(iv) We have $\text{MSL}^U \cap \text{MSL}^{-U} = \emptyset$, as the signatures are different.

13C.2. DEFINITION. (i) An *(intersection) type structure* (without universe) is a type algebra with the additional structure of a meet semi-lattice

$$\mathcal{S} = \langle |\mathcal{S}|, \leq, \cap, \rightarrow \rangle.$$

TS^U is the set of type structures without universe. The relation \leq and the operation \rightarrow have a priori no relation with each other, but in special structures this will be the case.

(ii) A *type structure with universe U* is a type algebra that is also a meet semi-lattice with universe.

$$\mathcal{S} = \langle |\mathcal{S}|, \leq, \cap, \rightarrow, U \rangle.$$

TS^U is the set of type structures with universe U .

(iii) $\text{TS} = \text{TS}^U \cup \text{TS}^{-U}$ is the set of type structures with or without universe. As before $\text{TS}^{-U} \cap \text{TS}^U = \emptyset$.

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, \dots 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 \rightarrow 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.

13C.3. PROPOSITION. Let $\mathcal{T} \in \text{TT}$ be compatible. Then \mathcal{T} induces a type structure $[\mathcal{T}]$ defined as follows.

$$[\mathcal{T}] = \langle [\mathbb{T}], \leq, \cap, \rightarrow \rangle,$$

by defining on the $=_{\mathcal{T}}$ -equivalence classes

$$\begin{aligned} [A] \leq [B] &\triangleq A \leq B; \\ [A] \cap [B] &\triangleq [A \cap B]; \\ [A] \rightarrow [B] &\triangleq [A \rightarrow B]^4 \end{aligned}$$

where A, B, C range over \mathbb{T} . If moreover \mathcal{T} has universe U , then $[\mathcal{T}]$ is a type structure with $[U]$ as universe.

PROOF. See Remark 13A.12. ■

⁴Here we misuse notation in a suggestive way, by using the same notation \rightarrow for equivalence classes as for types.

Let $\mathcal{T} \in \text{TT}$. Then the type structure $[\mathcal{T}]$ is called a *syntactical type structure*.

13C.4. PROPOSITION. *Every type structure is isomorphic to a syntactical one.*

PROOF. For a type structure \mathcal{S} , define a type theory $\text{Th}(\mathcal{S})$ as follows. Take $\mathbb{A} = \mathbb{A}^{\text{Th}(\mathcal{S})} = \{\underline{c} \mid c \in \mathcal{S}\}$. Then define $g : \mathbb{T}^{\mathbb{A}} \rightarrow \mathcal{S}$ by

$$\begin{aligned} g(\underline{c}) &\triangleq c; \\ g(A \rightarrow B) &\triangleq g(A) \rightarrow g(B); \\ g(A \cap B) &\triangleq g(A) \cap g(B). \end{aligned}$$

Define $A \leq_{\text{Th}(\mathcal{S})} B \iff g(A) \leq_{\mathcal{S}} g(B)$. Then g induces a bijective morphism

$$\bar{g} : [\text{Th}(\mathcal{S})] \rightarrow \mathcal{S}$$

where the inverse f is defined by $f(a) \triangleq [a]$. Moreover, if \mathcal{S} has a universe U , then $[U]$ is the universe of $[\text{Th}(\mathcal{S})]$ and $\bar{g}([U]) = U$. ■

13C.5. REMARK. (i) Each of the eleven compatible type theories \mathcal{T} in Figure 33 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 explored in Chapters 14, 16 and 17. 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 explored in Chapter 15.

(iii) Note that in a type theory there is up to $=_{\mathcal{T}}$ at most one universe 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.

13C.6. DEFINITION. Let $\mathcal{S} \in \text{TS}$.

(i) The notion of an *\mathcal{S} -statement* $M : A$, an *\mathcal{S} -declaration* $x:A$, an *\mathcal{S} -basis* and an *\mathcal{S} -assertion* $\Gamma \vdash M : A$ is as in Definition 13B.1, now for $A \in \mathcal{S}$ an element of the type structure \mathcal{S} .

(ii) The notion $\Gamma \vdash_{\cap}^{\mathcal{S}} M : A$ is defined by the same set of axioms and rules of $\lambda_{\cap}^{\mathcal{T}}$ in Definition 13B.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.

13C.7. PROPOSITION. *Let $\mathcal{T} \in \text{TT}$ be compatible and write*

$$[\mathcal{T}] = \langle [\mathbb{T}], \leq, \cap, \rightarrow, ([U]) \rangle$$

its corresponding type structure possibly with universe. For a type $A \in \mathcal{T}$ write its equivalence class as $[A] \in [\mathcal{T}]$. For $\Gamma = \{x_1 : B_1, \dots, x_n : B_n\}$ a \mathcal{T} -basis write $[\Gamma] = \{x_1 : [B_1], \dots, x_n : [B_n]\}$, a $[\mathcal{T}]$ -basis. Then

$$\Gamma \vdash_{\cap}^{\mathcal{T}} M : A \Leftrightarrow [\Gamma] \vdash_{\cap}^{[\mathcal{T}]} M : [A].$$

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 $\Gamma' = \{x_1 : B'_1, \dots, x_n : B'_n\}$, 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 15 we will introduce some categories related to given classes of type structures.

13C.8. DEFINITION. (i) The category \mathbf{MSL}^U has as objects the elements of \mathbf{MSL}^U and as morphisms maps $f : \mathcal{M} \rightarrow \mathcal{M}'$, 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) The category \mathbf{MSL}^U has as objects the elements of \mathbf{MSL}^U and as morphisms maps that preserve \leq, \cap but also U , i.e. $f(U) = U'$.

There is no natural category corresponding to \mathbf{MSL} , as it is a hybrid set consisting of structures with and without universe.

13C.9. DEFINITION. (i) The category \mathbf{TS}^U has as objects type structures and as morphisms maps $f : \mathcal{M} \rightarrow \mathcal{M}'$, which satisfy restrictions based on inequalities:

$$\begin{aligned} (m1) \quad A \leq B &\Rightarrow f(A) \leq' f(B); \\ (m2) \quad f(A) \cap' f(B) &\leq' f(A \cap B); \\ (m3) \quad f(A) \rightarrow' f(B) &\leq' f(A \rightarrow B). \end{aligned}$$

Because of monotonicity (m1) of f , we obtain that (m2) implies $f(A \cap B) = f(A) \cap' f(B)$.

(ii) The category \mathbf{TS}^U is as \mathbf{TS}^U , but based on type structures with universe. For morphisms we require

$$(m4) \quad U' \leq' f(U)$$

or, equivalently, $U' = f(U)$. Again there is no category corresponding to \mathbf{TS} .

13C.10. DEFINITION. The definitions of graph, lazy, natural and proper type theory translate immediately to type structures. For example, a type structure \mathcal{S} such that $(\rightarrow^=), (U_{\text{top}})$ hold is called a *graph type structure*.

13C.11. DEFINITION. We define three full subcategories of \mathbf{TS}^U and one full subcategory of \mathbf{TS}^U by specifying in each case the objects.

- (i) \mathbf{GTS}^U whose objects are the graph type structures.
- (ii) \mathbf{LTS}^U whose objects are the lazy type structures.
- (iii) \mathbf{NTS}^U whose objects are the natural type structures.
- (iv) \mathbf{PTS}^U whose objects are the proper type structures.

13D. Filters

In this section we define the notions of filter and filter structure.

13D.1. DEFINITION. (i) Let $\mathcal{T} \in \text{TT}^U$ and $X \subseteq \text{TT}^\mathcal{T}$. 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 $\mathcal{T} \in \text{TT}^{-U}$ and $X \subseteq \text{TT}^\mathcal{T}$. 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 $\mathcal{F}^\mathcal{T} = \{X \subseteq \text{TT}^\mathcal{T} \mid X \text{ is a filter over } \mathcal{T}\}$.

If $\mathcal{T} \in \text{TT}^U$, then filters are sets of types containing U and closed under \leq and \cap . If $\mathcal{T} \in \text{TT}^{-U}$, then filters may be empty.

13D.2. DEFINITION. Let $\mathcal{T} \in \text{TT}$.

- (i) For $A \in \text{TT}^\mathcal{T}$ write $\uparrow A \triangleq \{B \in \text{TT}^\mathcal{T} \mid A \leq B\}$.
- (ii) For $X \subseteq \text{TT}^\mathcal{T}$ define $\uparrow X$ to be the smallest filter over \mathcal{T} containing X .

13D.3. REMARK. (i) If X is non-empty,

$$\uparrow X = \{B \in \text{TT}^\mathcal{T} \mid \exists n \geq 1 \exists A_1, \dots, A_n \in X. A_1 \cap \dots \cap A_n \leq B\}.$$

(ii) For $X = \emptyset$, we have that

$$\uparrow \emptyset = \begin{cases} \{A \mid A = U\} = [U] & \text{if } \mathcal{T} \in \text{TT}^U \\ \emptyset & \text{if } \mathcal{T} \in \text{TT}^{-U} \end{cases}$$

(iii) $C \in \uparrow \{B_i \mid i \in I \neq \emptyset\} \Leftrightarrow \exists J \subseteq_{\text{fin}} I. [J \neq \emptyset \ \& \ \bigcap_{j \in J} B_j \leq C]$.

Complete lattices and the category **ALG** were introduced in Definition 10A.4.

13D.4. PROPOSITION. Let $\mathcal{T} \in \text{TT}$.

- (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 $A \in \text{TT}^\mathcal{T}$ one has $\uparrow A = \uparrow\{A\}$ and $\uparrow A \in \mathcal{F}^\mathcal{T}$.
- (iii) For $A, B \in \text{TT}^\mathcal{T}$ one has $\uparrow A \sqcup \uparrow B = \uparrow(A \cap B)$.
- (iv) For $A_i \in \text{TT}^\mathcal{T}$ ($i \in I$) one has $\bigsqcup\{\uparrow A_i \mid i \in I\} = \uparrow\{A_i \mid i \in I\}$.
- (v) For $X \in \mathcal{F}^\mathcal{T}$ one has

$$\begin{aligned} X &= \bigsqcup\{\uparrow A \mid A \in X\} = \bigsqcup\{\uparrow A \mid \uparrow A \subseteq X\} \\ &= \bigcup\{\uparrow A \mid A \in X\} = \bigcup\{\uparrow A \mid \uparrow A \subseteq X\}. \end{aligned}$$

(vi) The set $\mathcal{K}(\mathcal{F}^\mathcal{T})$ of finite (i.e. compact) elements of $\mathcal{F}^\mathcal{T}$ is given by

$$\mathcal{K}(\mathcal{F}^\mathcal{T}) = \begin{cases} \{\uparrow A \mid A \in \text{TT}^\mathcal{T}\} & \text{if } \mathcal{T} \in \text{TT}^U \\ \{\uparrow A \mid A \in \text{TT}^\mathcal{T}\} \cup \emptyset & \text{if } \mathcal{T} \in \text{TT}^{-U}. \end{cases}$$

(vii) $\mathcal{F}^\mathcal{T} \in \mathbf{ALG}$.

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, Coppo, and Dezani-Ciancaglini \[1983\]](#), this notion has played a major role in the study of the mathematical semantics of lambda calculus.

13D.5. DEFINITION. Let $\mathcal{T} \in \text{TT}$. Define

$$\begin{aligned} F^{\mathcal{T}} &\in [\mathcal{F}^{\mathcal{T}} \rightarrow [\mathcal{F}^{\mathcal{T}} \rightarrow \mathcal{F}^{\mathcal{T}}]], \quad \text{and} \\ G^{\mathcal{T}} &\in [[\mathcal{F}^{\mathcal{T}} \rightarrow \mathcal{F}^{\mathcal{T}}] \rightarrow \mathcal{F}^{\mathcal{T}}] \end{aligned}$$

as follows

$$\begin{aligned} F^{\mathcal{T}}(X)(Y) &\triangleq \uparrow\{B \in \mathbb{T}^{\mathcal{T}} \mid \exists A \in Y. (A \rightarrow B) \in X\}; \\ G^{\mathcal{T}}(f) &\triangleq \uparrow\{A \rightarrow B \mid B \in f(\uparrow A)\}. \end{aligned}$$

Then, $\mathcal{F}^{\mathcal{T}} \triangleq \langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle$ is called the *filter structure* over \mathcal{T} .

It is easy to show that

$$F^{\mathcal{T}} \in [\mathcal{F}^{\mathcal{T}} \rightarrow [\mathcal{F}^{\mathcal{T}} \rightarrow \mathcal{F}^{\mathcal{T}}]] \ \& \ G^{\mathcal{T}} \in [[\mathcal{F}^{\mathcal{T}} \rightarrow \mathcal{F}^{\mathcal{T}}] \rightarrow \mathcal{F}^{\mathcal{T}}].$$

13D.6. REMARK. The items 13A.18-13B.12 and 13D.1-13D.5 are about type theories, but can be translated immediately to structures and if no \rightarrow are involved to meet-semi lattices. For example Proposition 13A.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 13A.22-13A.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 13B.7-13B.12 hold for type structures, as follows immediately from Propositions 13C.4 and 13C.7. Finally 13D.1-13D.5 can be translated immediately to type structures and meet semi-lattices. In Chapter 15 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.

13D.7. LEMMA. 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 . ■

13E. Exercises

13E.1. Let \mathcal{T} be a type theory that satisfies (\rightarrow) .

- (i) Show that if (U_{top}) holds, then the scheme axiom $(U \rightarrow)$ is equivalent to the following axiom:

$$U \leq U \rightarrow U.$$

- (ii) Show that $(\rightarrow \cap)$ is equivalent with $(\rightarrow \cap =)$.

13E.2. Let $\mathcal{T} \in \{\text{Scott}, \text{DHM}\}$. Prove that 0 is a bottom element in \mathcal{T} , i.e. $0 \leq_{\mathcal{T}} A$ for all $A \in \mathbb{T}^{\mathcal{T}}$.

13E.3. Let $\mathcal{T} \in \{\text{CDZ}, \text{HR}\}$. Show that $\mathbf{U} \rightarrow 0 < 0$. Conclude that 0 is not a bottom element in \mathcal{T} .

13E.4. Prove that for all types $A \in \mathbb{T}^{\text{AO}}$ there is an n such that

$$\mathbf{U}^n \rightarrow \mathbf{U} \leq_{\text{AO}} A.$$

13E.5. Show that $\Gamma, x:\mathbf{U} \vdash_{\cap}^{\mathcal{T}} M : A \Rightarrow \Gamma \vdash_{\cap}^{\mathcal{T}} M : A$.

13E.6. For \mathcal{T} a type theory, $M, N \in \Lambda$ and $x \notin \text{dom}(\Gamma)$ show

$$\Gamma \vdash_{\cap}^{\mathcal{T}} M : A \Rightarrow \Gamma \vdash_{\cap}^{\mathcal{T}} M[x := N] : A.$$

13E.7. Prove that if $(01), (1 \rightarrow 0)$ and $(0 \rightarrow 1)$ are axioms in \mathcal{T} , then for all M in normal form $\{x_1 : 1, \dots, x_n : 1\} \vdash^{\mathcal{T}} M : 0$, where $\{x_1, \dots, x_n\} \supseteq \text{FV}(M)$.

13E.8. Show that

$$M \text{ is a closed term} \Rightarrow \vdash_{\cap}^{\text{Park}} M : 0.$$

Later we will show the converse (Theorem 17D.3).

13E.9. Suppose in \mathcal{T} there is a type A such that $A = A \rightarrow A$. Then

$$\text{FV}(M) \subseteq \text{dom}(\Gamma) \Rightarrow \Gamma \vdash_{\cap}^{\mathcal{T}} M : A.$$

13E.10. The type theory [Krivine](#) and the type assignment system $\lambda_{\cap}^{\text{Krivine}}$ of [Krivine](#) [1990] are CD and $\lambda_{\cap}^{\text{CD}}$, but with rule (\leq) replaced by

$$\boxed{(\cap E) \quad \frac{\Gamma \vdash M : A \cap B}{\Gamma \vdash M : A} \quad \frac{\Gamma \vdash M : A \cap B}{\Gamma \vdash M : B}}$$

Similarly [Krivine^U](#) and $\lambda_{\cap}^{\text{Krivine}^U}$ are CDS and $\lambda_{\cap}^{\text{CDS}}$, with (\leq) replaced by $(\cap E)$.

Show that

$$(i) \quad \Gamma \vdash^{\text{Krivine}} M : A \Leftrightarrow \Gamma \vdash_{\cap}^{\text{CD}} M : A.$$

$$(ii) \quad \Gamma \vdash^{\text{Krivine}^U} M : A \Leftrightarrow \Gamma \vdash_{\cap}^{\text{CDS}} M : A.$$

13E.11. (i) Show that $\lambda x. xxx$ and $(\lambda x. xx)I$ are typable in $\lambda_{\cap}^{\text{Krivine}}$.

(ii) Show that all closed terms in normal form are typable in $\lambda_{\cap}^{\text{Krivine}}$.

13E.12. Show the following:

$$(i) \quad \vdash^{\text{Krivine}} \lambda z. \mathbf{KI}(zz) : (A \rightarrow B) \cap A \rightarrow C \rightarrow C.$$

$$(ii) \quad \vdash^{\text{Krivine}^U} \lambda z. \mathbf{KI}(zz) : \mathbf{U} \rightarrow C \rightarrow C.$$

$$(iii) \quad \vdash_{\cap}^{\text{BCD}} \lambda z. \mathbf{KI}(zz) : \mathbf{U} \rightarrow (A \rightarrow B \cap C) \rightarrow A \rightarrow B.$$

13E.13. Consider HL as type structure with universe $\langle [\mathbb{T}^{\text{HL}}], \leq_{\text{HL}}, \cap, \rightarrow, [1] \rangle$, where $[1]$ is taken as the universe. Show that $\text{HL} \notin \mathbf{LTS}^U$ and $\text{HL} \notin \mathbf{NTS}^U$ [Hint. Show that $1 \rightarrow 1 <_{\text{HL}} 0 \rightarrow 1$].

13E.14. Let $\mathcal{D} = \langle \mathcal{D}, \cdot \rangle$ be an applicative structure. Consider $(\mathcal{P}(\mathcal{D}), \Rightarrow, \subseteq, \cap, \mathcal{D})$, where $\mathcal{P}(\mathcal{D})$ is the power set of \mathcal{D} , \Rightarrow is defined in 3A.34, \subseteq and \cap are the usual set theoretic notions, and \mathcal{D} is the top of $\mathcal{P}(\mathcal{D})$. Show

- $(\mathcal{P}(\mathcal{D}), \Rightarrow, \subseteq, \cap)$ is a proper type structure.
- $\mathcal{D} = \mathcal{D} \Rightarrow \mathcal{D}$.
- $(\mathcal{P}(\mathcal{D}), \Rightarrow, \subseteq, \cap, \mathcal{D})$ is a natural type structure.

13E.15. Show that the inclusions suggested in Fig. 34 are strict.

CHAPTER 14

BASIC PROPERTIES

This chapter studies meta-theoretical properties of the type assignment systems. They will be crucial for the development of filter lambda models induced by these systems in Chapter 16. The most important question here is whether the type assignment system satisfies β -, η -reduction or β -, η -expansion. Due to the intrinsic syntactic nature of the filter models, any property on the type assignment system is transferred into the filter structure. This is revealed by the Type Semantics Theorem (Theorem 16B.7) which says that the interpretation of a term is the set of its types. The amazing consequence of this is that we know whether the filter structure preserves the meaning of λ -terms under β - or η - reduction or expansion, exactly when the type assignment does.

Inversion lemmas are proved in Section 14A and they essentially state when an assertion $\Gamma \vdash^{\mathcal{T}} M : A$ holds depending on the form of M . This is a convenient technical tool for doing proofs by induction (or by cases) on M when the typing is involved. Instead of doing a proof by induction on the derivation, the inversion lemma gives the possibility of doing induction on the structure of the term.

Fig. 36 summarizes the characterizations of the admissibility of β and η in the type assignment systems. This topic is developed in Section 14B.

Property of $\mathcal{T} \in \text{TT}$	versus	property of $\lambda_{\cap}^{\mathcal{T}}$
β -sound	\Rightarrow	β -red
$\mathcal{T} \in \text{TT}$	\Rightarrow	βl -exp
$\mathcal{T} \in \text{TT}^U$	\Rightarrow	β -exp
$\mathcal{T} \in \text{TT}^U$ & proper	\Leftrightarrow	η -red
$\mathcal{T} \in \text{TT}^U$ & natural	\Leftrightarrow	η -red
$\mathcal{T} \in \text{TT}^U$ & η -sound	\Leftrightarrow	η -exp
$\mathcal{T} \in \text{TT}^U$ & η^U -sound	\Leftrightarrow	η -exp

FIGURE 36. Characterizing reduction and expansion

On the left hand side of the table, we find sufficient conditions (sometimes necessary as well) on the type theory to preserve the typing after reducing or expanding.

The notions of natural and proper type theory were introduced in Definition 13A.18 because they characterize η -reduction. The idea is to deduce which conditions are necessary to preserve the typing after η -reducing terms. What we find turns out to be a sufficient set of conditions as well. Let $\Gamma = \{x : (A \rightarrow B)\}$. If $A \geq A'$ and $B \leq B'$, then $\Gamma \vdash \lambda y. xy : A' \rightarrow B'$ and $\lambda y. xy$ η -reduces to x . Then, it is easy to see that to deduce $\Gamma \vdash x : A' \rightarrow B'$, we need the axiom (\rightarrow) . A similar argument can be used to show that the axiom $(\rightarrow \cap)$ is necessary as well.

New classes of type theories will be given for completing the picture. The notions of η -sound and η^U -sound are introduced in Definition 14B.10 to characterize η -expansion. Roughly speaking, the condition of η -soundness states that any type A should be equal to an intersection of arrow types. Let $\Gamma = \{x : A\}$. The term $\lambda y. xy$ will have type A in Γ if we impose the condition that there exist B, C, D, E such that

$$\begin{aligned} (B \rightarrow C) &\leq A \\ A &\leq (D \rightarrow E) \\ B &\leq D \ \& \ A \leq C. \end{aligned}$$

It is easy to see that if the type theory is also natural, then $A = B \rightarrow C$. The above condition is a simplification of the actual condition of η -soundness that has to consider intersections.

For β -expansion, there is no need to impose any condition, except the presence of the universal type. As explained in the introduction of Chapter 12, this was the reason to introduce intersection types. Without (U_{top}) , the type assignment system only preserves βI -expansion.

For β -reduction, the condition of β -soundness is introduced in Definition 14A.4. To illustrate the idea, we consider a simplification of β -soundness:

$$(A \rightarrow B) \cap (C \rightarrow D) \leq E \rightarrow F \Rightarrow E \leq A \cap C \ \& \ B \cap D \leq F.$$

If we know that $(\lambda x. M)N$ is typable, then we can apply the Inversion Lemmas and reach a point where the condition of β -soundness appears naturally to be able to continue with the proof and deduce that $M[x := N]$ is typable too. Suppose that

$$\begin{aligned} \vdash (\lambda x. M) : E \rightarrow F \\ \vdash N : E. \end{aligned}$$

This does not necessarily mean that $x : E \vdash M : F$. We could have the situation where

$$\begin{aligned} (A \rightarrow B) \cap (C \rightarrow D) &\leq E \rightarrow F \\ x : A \vdash M : B \\ x : C \vdash M : D. \end{aligned}$$

Our goal is to conclude that $x : E \vdash M : F$ and apply (cut) to finally have that $\vdash M[x := N] : F$. At this point, we see that it would be enough to have the simplified condition of β -soundness to conclude our goal.

The condition of β -soundness is sufficient for preserving β -red but it is not necessary. In Definition 16B.19, an example will be given of a type theory that preserves β -reduction but is not β -sound.

Fig. 37 summarizes the results on β and η for the specific type theories of Fig. 33. The proofs can be found in Sections 14B and 14C. The symbol ‘ \checkmark ’ stands for “holds” and ‘ \times ’ for “fails”.

\mathcal{T}	β -red	β -exp	βI -exp	η -red	η -exp
Scott	✓	✓	✓	✓	✓
Park	✓	✓	✓	✓	✓
CDZ	✓	✓	✓	✓	✓
HR	✓	✓	✓	✓	✓
DHM	✓	✓	✓	✓	✓
BCD	✓	✓	✓	✓	✗
AO	✓	✓	✓	✗	✓
Plotkin	✓	✓	✓	✗	✗
Engeler	✓	✓	✓	✗	✗
CDS	✓	✓	✓	✗	✗
HL	✓	✗	✓	✓	✓
CDV	✓	✗	✓	✓	✗
CD	✓	✗	✓	✗	✗

FIGURE 37. Reduction and expansion in the type assignment systems

Since all type theories of Fig. 37 are β -sound, they all preserve β -reduction and obviously, they also preserve βI -reduction. Since the type assignment systems induced by the first ten type theories contain (U_{top}) , they are all closed under β -expansions. The type assignment systems for the last three are closed under βI -expansions.

It is easy to see that BCD, Engeler, CDS, CDV, CD cannot be closed under η -expansion because a constant in A_∞ cannot be decomposed into an intersection of arrow types. A similar argument applies to Plotkin where 0 does not have any axiom associated to it.

14A. Inversion lemmas

In the style of [Coppo, Dezani-Ciancaglini, Honsell, and Longo \[1984\]](#) and [Alessi, Barbanera, and Dezani-Ciancaglini \[2003\]](#), [Alessi, Barbanera, and Dezani-Ciancaglini \[2006\]](#) we shall isolate special properties which allow to ‘reverse’ some of the rules of the type assignment system $\vdash_{\mathcal{T}}$, 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 14A.1 and 14A.9.

14A.1. THEOREM ([Inversion Lemma for \$\lambda_{\cap}^{\mathcal{T}}\$](#)). *Let $\mathcal{T} \in \text{TT}$ and let \vdash denote $\vdash_{\mathcal{T}}$. If $\mathcal{T} \in \text{TT}^{-U}$, then the following statements hold unconditionally; if $\mathcal{T} \in \text{TT}^U$, then they hold under the*

assumption that $A \neq \mathbb{U}$ in (i) and (ii).

- (i) $\Gamma \vdash x : A \Leftrightarrow \Gamma(x) \leq A.$
- (ii) $\Gamma \vdash MN : A \Leftrightarrow \exists k \geq 1 \exists B_1, \dots, B_k, C_1, \dots, C_k$
 $[C_1 \cap \dots \cap C_k \leq A \& \forall i \in \{1, \dots, k\}$
 $\Gamma \vdash M : B_i \rightarrow C_i \& \Gamma \vdash N : B_i].$
- (iii) $\Gamma \vdash \lambda x.M : A \Leftrightarrow \exists k \geq 1 \exists B_1, \dots, B_k, C_1, \dots, C_k$
 $[(B_1 \rightarrow C_1) \cap \dots \cap (B_k \rightarrow C_k) \leq A$
 $\& \forall i \in \{1, \dots, k\}. \Gamma, x:B_i \vdash M : C_i].$

PROOF. We only treat (\Rightarrow) in (i)-(iii), as (\Leftarrow) is trivial. First consider $\mathcal{T} \in \text{TT}^{-\mathbb{U}}$.

(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 ($\rightarrow E$), (\leq) or ($\cap I$). In the first case the last applied rule is

$$(\rightarrow E) \quad \frac{\Gamma \vdash M : D \rightarrow A \quad \Gamma \vdash N : D}{\Gamma \vdash MN : A}.$$

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

$$(\cap I) \quad \frac{\Gamma \vdash MN : A_1 \quad \Gamma \vdash MN : A_2}{\Gamma \vdash MN : A_1 \cap A_2}.$$

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

$$\begin{aligned} \Gamma \vdash M : B_i \rightarrow C_i, & \quad \Gamma \vdash N : B_i, \\ \Gamma \vdash M : D_j \rightarrow E_j, & \quad \Gamma \vdash N : D_j, \\ C_1 \cap \dots \cap C_k \leq A_1, & \quad E_1 \cap \dots \cap E_{k'} \leq A_2. \end{aligned}$$

Hence we are done, as $C_1 \cap \dots \cap C_k \cap E_1 \cap \dots \cap E_{k'} \leq A$.

(iii) Again by induction on derivations. We only treat the case $A \equiv A_1 \cap A_2$ and the last applied rule is ($\cap I$):

$$(\cap I) \quad \frac{\Gamma \vdash \lambda x.M : A_1 \quad \Gamma \vdash \lambda x.M : A_2}{\Gamma \vdash \lambda x.M : A_1 \cap A_2}.$$

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

$$\begin{aligned} \Gamma, x:B_i \vdash M : C_i, & \quad (B_1 \rightarrow C_1) \cap \dots \cap (B_k \rightarrow C_k) \leq A_1, \\ \Gamma, x:D_j \vdash M : E_j, & \quad (D_1 \rightarrow E_1) \cap \dots \cap (D_{k'} \rightarrow E_{k'}) \leq A_2. \end{aligned}$$

We are done, since $(B_1 \rightarrow C_1) \cap \dots \cap (B_k \rightarrow C_k) \cap (D_1 \rightarrow E_1) \cap \dots \cap (D_{k'} \rightarrow E_{k'}) \leq A$.

Now we prove (\Rightarrow) in (i)-(iii) for $\mathcal{T} \in \text{TT}^{\mathbb{U}}$.

(i), (ii) The condition $A \neq \mathbb{U}$ implies that axiom $(\mathbf{U}_{\text{top}})$ cannot have been used in the last step. Hence the reasoning above suffices.

(iii) If $A = \mathbb{U}$, then (\Rightarrow) in (iii) holds as $\mathbb{U} \rightarrow \mathbb{U} \leq \mathbb{U}$ and $\Gamma, x:\mathbb{U} \vdash M : \mathbb{U}$. So we may assume that $A \neq \mathbb{U}$. Then the only interesting rule is $(\cap I)$. Condition $A \neq \mathbb{U}$ implies that we cannot have $A_1 = A_2 = \mathbb{U}$. In case $A_1 \neq \mathbb{U}$ and $A_2 \neq \mathbb{U}$ 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 13A), the Inversion Lemma can be restated in a more memorable form. This will be done in Theorem 14A.9.

14A.2. COROLLARY (Subformula property). *Let $\mathcal{T} \in \text{TT}$. Assume*

$$\Gamma \vdash_{\cap}^{\mathcal{T}} M : A \text{ and } N \text{ is a subterm of } M.$$

Then N is typable in an extension $\Gamma^+ = \Gamma, x_1:B_1, \dots, x_n:B_n$ in which also the variables $\{x_1, \dots, x_n\} = \text{FV}(N) - \text{FV}(M)$ get a type assigned.

PROOF. We can write $M \equiv C[N]$. If $\mathcal{T} \in \text{TT}^{\mathbb{U}}$, then the statement is trivial as $\vdash N : \mathbb{U}$. Otherwise the statement is proved by induction on the structure of $C[]$, using Theorem 14A.1. ■

14A.3. PROPOSITION. *Let $\mathcal{T} \in \text{TT}$. Writing \vdash for $\vdash_{\cap}^{\mathcal{T}}$, we have for $y \notin \text{dom}(\Gamma)$*

$$\begin{aligned} \exists B [\Gamma \vdash N : B \ \& \ \Gamma \vdash M[x := N] : A] &\Rightarrow \\ \exists C [\Gamma \vdash N : C \ \& \ \Gamma, y:C \vdash M[x := y] : A]. \end{aligned}$$

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 $(\rightarrow I)$ and preservation of β -reduction.

14A.4. DEFINITION. \mathcal{T} is called **β -sound** if

$$\forall k \geq 1 \forall A_1, \dots, A_k, B_1, \dots, B_k, C, D,$$

$$\begin{aligned} (A_1 \rightarrow B_1) \cap \dots \cap (A_k \rightarrow B_k) \leq (C \rightarrow D) \ \& \ D \neq \mathbb{U} \Rightarrow \\ C \leq A_{i_1} \cap \dots \cap A_{i_p} \ \& \ B_{i_1} \cap \dots \cap B_{i_p} \leq D, \\ \text{for some } p \geq 1 \text{ and } 1 \leq i_1, \dots, i_p \leq k. \end{aligned}$$

This definition translates immediately to type structures. The notion of β -soundness is important to prove invertibility of the rule $(\rightarrow I)$, which is crucial for the next section.

In particular the condition of β -soundness for $k=1$ is expressed as follows:

$$B' \neq \mathbb{U} \ \& \ A \rightarrow B \leq A' \rightarrow B' \Rightarrow A' \leq A \ \& \ B \leq B'.$$

When $B' = \mathbb{U}$ and $A \rightarrow B \leq A' \rightarrow B'$, 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 Fig. 33 are β -sound. The proof occupies 14A.5-14A.7.

14A.5. REMARK. Note that in a TT every type A can be written uniquely, modulo the order, as

$$A \equiv \alpha_1 \cap \dots \cap \alpha_n \cap (B_1 \rightarrow C_1) \cap \dots \cap (B_k \rightarrow C_k) \tag{+},$$

i.e. an intersection of atoms ($\alpha_i \in \mathbb{A}$) and arrow types.

For some of our \mathcal{T} the shape (+) in Remark 14A.5 can be simplified.

14A.6. DEFINITION. For the type theories \mathcal{T} of Fig. 33 we define for each $A \in \mathbb{T}^{\mathcal{T}}$ its *canonical form*, notation $\text{cf}(A)$, as follows.

(i) If $\mathcal{T} \in \{\text{BCD}, \text{AO}, \text{Plotkin}, \text{Engeler}, \text{CDV}, \text{CDS}, \text{CD}\}$, then

$$\text{cf}(A) \equiv A.$$

(ii) If $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{HL}\}$ then the definition is by induction on A . For an atom α the canonical form $\text{cf}(\alpha)$ depends on the type theory in question; moreover the mapping cf preserves \rightarrow, \cap and \mathbb{U} .

System \mathcal{T}	A	$\text{cf}(A)$
Scott	0	$\mathbb{U} \rightarrow 0$
Park	0	$0 \rightarrow 0$
CDZ, HL	0	$1 \rightarrow 0$
	1	$0 \rightarrow 1$
HR	0	$1 \rightarrow 0$
	1	$(0 \rightarrow 0) \cap (1 \rightarrow 1)$
DHM	0	$\mathbb{U} \rightarrow 0$
	1	$0 \rightarrow 1$
All systems	\mathbb{U}	\mathbb{U}
All systems	$B \rightarrow C$	$B \rightarrow C$
All systems	$B \cap C$	$\text{cf}(B) \cap \text{cf}(C)$

14A.7. THEOREM. All type theories of Fig. 33 are β -sound.

PROOF. We prove the following stronger statement (induction loading). Let

$$A \leq A',$$

$$\text{cf}(A) = \alpha_1 \cap \dots \cap \alpha_n \cap (B_1 \rightarrow C_1) \cap \dots \cap (B_k \rightarrow C_k),$$

$$\text{cf}(A') = \alpha'_1 \cap \dots \cap \alpha'_{n'} \cap (B'_1 \rightarrow C'_1) \cap \dots \cap (B'_{k'} \rightarrow C'_{k'})$$

where $n, n' \geq 0, k, k' \geq 1$. Then

$$\forall j \in \{1, \dots, k'\}. [C'_j \neq \mathbb{U} \Rightarrow$$

$$\exists p \geq 1 \exists i_1, \dots, i_p \in \{1, \dots, k\}. [B'_j \leq B_{i_1} \cap \dots \cap B_{i_p} \& C_{i_1} \cap \dots \cap C_{i_p} \leq C'_j]].$$

The proof of the statement is by induction on the generation of $A \leq A'$. From it β -soundness follows easily. ■

14A.8. REMARK. From Theorem 14A.7 it follows immediately that for the compatible theories of Fig. 33 the corresponding type structures are β -sound.

14A.9. THEOREM (Inversion Lemma II). Let $\mathcal{T} \in \text{TT}$. Of the following properties (i) holds in general, (ii) provided that $\mathcal{T} \in \text{PTT}$ and if $\mathcal{T} \in \text{TT}^{\mathbb{U}}$, then $A \neq \mathbb{U}$, and (iii) provided that \mathcal{T} is β -sound.

- (i) $\Gamma, x:A \vdash x : B \Leftrightarrow A \leq B.$
- (ii) $\Gamma \vdash (MN) : A \Leftrightarrow \exists B [\Gamma \vdash M : (B \rightarrow A) \& \Gamma \vdash N : B].$
- (iii) $\Gamma \vdash (\lambda x.M) : (B \rightarrow C) \Leftrightarrow \Gamma, x:B \vdash M : C.$

PROOF. The proof of each (\Leftarrow) is easy. So we only treat (\Rightarrow).

(i) If $B \neq \mathbb{U}$, then the conclusion follows from Theorem 14A.1(i). If $B = \mathbb{U}$, then the conclusion holds trivially.

(ii) Suppose $\Gamma \vdash MN : A$. Then by Theorem 14A.1(ii) there are $B_1, \dots, B_k, C_1, \dots, C_k$, with $k \geq 1$, such that $C_1 \cap \dots \cap C_k \leq A$, $\Gamma \vdash M : B_i \rightarrow C_i$ and $\Gamma \vdash N : B_i$ for $1 \leq i \leq k$. Hence $\Gamma \vdash N : B_1 \cap \dots \cap B_k$ and

$$\begin{aligned}\Gamma \vdash M : & (B_1 \rightarrow C_1) \cap \dots \cap (B_k \rightarrow C_k) \\ & \leq (B_1 \cap \dots \cap B_k) \rightarrow (C_1 \cap \dots \cap C_k) \\ & \leq (B_1 \cap \dots \cap B_k) \rightarrow A,\end{aligned}$$

by Lemma 13A.21. So we can take $B \equiv (B_1 \cap \dots \cap B_k)$.

(iii) Suppose $\Gamma \vdash (\lambda x.M) : (B \rightarrow C)$. Then Theorem 14A.1(iii) applies and we have for some $k \geq 1$ and $B_1, \dots, B_k, C_1, \dots, C_k$

$$\begin{aligned}(B_1 \rightarrow C_1) \cap \dots \cap (B_k \rightarrow C_k) &\leq B \rightarrow C, \\ \Gamma, x:B_i &\vdash M : C_i \text{ for all } i.\end{aligned}$$

If $C = \mathbb{U}$, then the assertion holds trivially, so let $C \neq \mathbb{U}$. Then by β -soundness there are $1 \leq i_1, \dots, i_p \leq k$, $p \geq 1$ such that

$$\begin{aligned}B &\leq B_{i_1} \cap \dots \cap B_{i_p}, \\ C_{i_1} \cap \dots \cap C_{i_p} &\leq C.\end{aligned}$$

Applying (\leq -L) we get

$$\begin{aligned}\Gamma, x:B &\vdash M : C_{i_j}, \quad 1 \leq j \leq p, \\ \Gamma, x:B &\vdash M : C_{i_1} \cap \dots \cap C_{i_p} \leq C. \blacksquare\end{aligned}$$

We give a simple example which shows that in general rule (\rightarrow E) cannot be reversed, i.e. that if $\Gamma \vdash MN : B$, then it is not always true that there exists A such that $\Gamma \vdash M : A \rightarrow B$ and $\Gamma \vdash N : A$.

14A.10. EXAMPLE. Let $\mathcal{T} = \text{Engeler}$, one of the intersection type theories of Fig. 33. Let $\Gamma \triangleq \{x:(c_0 \rightarrow c_1) \cap (c_2 \rightarrow c_3), y:(c_0 \cap c_2)\}$. Then one has

$$\Gamma \vdash_{\cap}^{\mathcal{T}} xy : c_1 \cap c_3.$$

But for no type B

$$\Gamma \vdash_{\cap}^{\mathcal{T}} x : B \rightarrow (c_1 \cap c_3) \text{ and } \Gamma \vdash_{\cap}^{\mathcal{T}} y : B.$$

14A.11. REMARK. Note that in general

$$\Gamma \vdash_{\cap}^{\mathcal{T}} (\lambda x.M) : A \not\Rightarrow \exists B, C. A = (B \rightarrow C) \& \Gamma, x:B \vdash_{\cap}^{\mathcal{T}} M : C.$$

Consider $\vdash_{\cap}^{\text{BCD}} I : (c_1 \rightarrow c_1) \cap (c_2 \rightarrow c_2)$, with c_1, c_2 different type atoms.

14A.12. PROPOSITION. For all \mathcal{T} in Fig. 13A.14 except AO properties (i), (ii) and (iii) of Theorem 14A.9 hold unconditionally. For $\mathcal{T} = \text{AO}$ they hold under the condition that $A \neq \mathbb{U}$ in (ii).

PROOF. Since these \mathcal{T} are proper and β -sound, by Theorem 14A.7, we can apply Theorem 14A.9. Moreover, by axiom (\rightarrow U) we have $\Gamma \vdash_{\cap}^{\mathcal{T}} M : \mathbb{U} \rightarrow \mathbb{U}$ for all Γ, M , hence we do not need to assume $A \neq \mathbb{U}$ for $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{BCD}\}$. ■

14B. Subject reduction and expansion

Various subject reduction and expansion properties are proved, for the classical β , βI and η notions of reduction. Other results can be found in Alessi, Barbanera, and Dezani-Ciancaglini [2003], Alessi, Barbanera, and Dezani-Ciancaglini [2006]. We consider the following rules.

$$(R\text{-}red) \quad \frac{M \rightarrow_R N \quad \Gamma \vdash M : A}{\Gamma \vdash N : A}$$

$$(R\text{-}exp) \quad \frac{M_R \leftarrow N \quad \Gamma \vdash M : A}{\Gamma \vdash N : A}$$

where R is a notion of reduction, notably β -, βI , or η -reduction. If one of these rules holds in λ_n^T , we write $\lambda_n^T \models (R\text{-}\{\text{exp}, \text{red}\})$, respectively. If both hold we write $\lambda_n^T \models (R\text{-cnv})$. These properties will be crucial in Section 16B, where we will discuss (untyped) λ -models induced by these systems.

Recall that $(\lambda x.M)N$ is a βI -redex if $x \in \text{FV}(M)$, Curry and Feys [1958].

β -conversion

We first investigate when $\lambda_n^T \models (\beta(\text{I})\text{-red})$.

14B.1. PROPOSITION. Let $T \in \text{TT}$. Then we have the following.

$$(i) \quad \lambda_n^T \models (\beta\text{I}-\text{red}) \Leftrightarrow$$

$$[\Gamma \vdash^T (\lambda x.M) : (B \rightarrow A) \& x \in \text{FV}(M) \Rightarrow \Gamma, x:B \vdash^T M : A].$$

$$(ii) \quad \lambda_n^T \models (\beta-\text{red}) \Leftrightarrow$$

$$[\Gamma \vdash^T (\lambda x.M) : (B \rightarrow A) \Rightarrow \Gamma, x:B \vdash^T M : A].$$

PROOF. (i) (\Rightarrow) Assume $\Gamma \vdash \lambda x.M : B \rightarrow A$ & $x \in \text{FV}(M)$, which implies $\Gamma, y:B \vdash (\lambda x.M)y : A$, by weakening and rule $(\rightarrow E)$ for a fresh y . Now rule $(\beta\text{I}-\text{red})$ gives us $\Gamma, y:B \vdash M[x:=y] : A$. Hence $\Gamma, x:B \vdash M : A$.

(\Leftarrow) Suppose $\Gamma \vdash (\lambda x.M)N : A$ & $x \in \text{FV}(M)$, in order to show that $\Gamma \vdash M[x:=N] : A$. We may assume $A \neq \text{U}$. Then Theorem 14A.1(ii) implies $\Gamma \vdash \lambda x.M : B_i \rightarrow C_i$, $\Gamma \vdash N : B_i$ and $C_1 \cap \dots \cap C_k \leq A$, for some $B_1, \dots, B_k, C_1, \dots, C_k$. By assumption $\Gamma, x:B_i \vdash M : C_i$. Hence by rule (cut) , Proposition 13B.10, one has $\Gamma \vdash M[x:=N] : C_i$. Therefore $\Gamma \vdash M[x:=N] : A$, using rules $(\cap I)$ and (\leq) .

(ii) Similarly. ■

14B.2. COROLLARY. Let $T \in \text{TT}$ be β -sound. Then $\lambda_n^T \models (\beta\text{-red})$.

PROOF. Using Theorem 14A.9(iii). ■

The converse of Corollary 14B.2 does not hold. In Definition 16B.19 we will introduce a type theory that is not β -sound, but nevertheless induces a type assignment system satisfying $(\beta\text{-red})$.

14B.3. COROLLARY. Let T be one of the TT in Fig. 13A.14. Then

$$\lambda_n^T \models (\beta\text{-red}).$$

PROOF. By Corollary 14B.2 and Theorem 14A.7. ■

Now we investigate when $\lambda_n^{\mathcal{T}} \models (\beta\text{-exp})$. 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 λ_{\rightarrow} this problem causes the failure of rule $(\beta\text{-exp})$. But in the intersection type theories one has $N : B_1 \cap \dots \cap B_k$ if $N : B_1, \dots, N : B_k$. 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 λK -redex. We would like to assign as type to N the intersection over an empty sequence, i.e. the universe U . This makes $(\beta\text{-exp})$ invalid for $\mathcal{T} \in TT^{-U}$, but valid for $\mathcal{T} \in TT^U$.

14B.4. PROPOSITION. *Let $\mathcal{T} \in TT$. Then we have the following.*

(i) *Suppose $\Gamma \vdash^{\mathcal{T}} M[x:=N] : A$. Then*

$$\Gamma \vdash^{\mathcal{T}} (\lambda x.M)N : A \Leftrightarrow N \text{ is typable in context } \Gamma.$$

(ii) $\lambda_n^{\mathcal{T}} \models (\beta I\text{-exp}) \Leftrightarrow \forall \Gamma, M, N, A \text{ with } x \in FV(M)$

$$[\Gamma \vdash^{\mathcal{T}} M[x:=N] : A \Rightarrow N \text{ is typable in context } \Gamma].$$

(iii) $\lambda_n^{\mathcal{T}} \models (\beta\text{-exp}) \Leftrightarrow \forall \Gamma, M, N, A$

$$[\Gamma \vdash^{\mathcal{T}} M[x:=N] : A \Rightarrow N \text{ is typable in context } \Gamma].$$

PROOF. (i) (\Rightarrow) By Theorem 14A.1(ii). (\Leftarrow) Let $\Gamma \vdash M[x:=N] : A$ and suppose N is typable in context Γ . By Proposition 14A.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 14A.1. ■

14B.5. COROLLARY. (i) *Let $\mathcal{T} \in TT$. Then $\lambda_n^{\mathcal{T}} \models (\beta I\text{-exp})$.*

(ii) *Let $\mathcal{T} \in TT^U$. Then $\lambda_n^{\mathcal{T}} \models (\beta\text{-exp})$.*

PROOF. (i) By the subformula property (Corollary 14A.2).

(ii) Trivial, since every term has type U . ■

Now we can harvest results towards closure under β -conversion.

14B.6. THEOREM. (i) *Let $\mathcal{T} \in TT$. Then*

$$\mathcal{T} \text{ is } \beta\text{-sound} \Rightarrow \lambda_n^{\mathcal{T}} \models (\beta I\text{-cnv}).$$

(ii) *Let $\mathcal{T} \in TT^U$. Then*

$$\mathcal{T} \text{ is } \beta\text{-sound} \Rightarrow \lambda_n^{\mathcal{T}} \models (\beta\text{-cnv}).$$

PROOF. (i) By Corollaries 14B.2 and 14B.5(i).

(ii) By Corollaries 14B.2 and 14B.5(ii). ■

14B.7. COROLLARY. (i) *Let \mathcal{T} be a TT of Fig. 13A.14. Then*

$$\lambda_n^{\mathcal{T}} \models (\beta I\text{-cnv}).$$

(ii) *Let $\mathcal{T} \in \{\text{Scott, Park, CDZ, HR, DHM, BCD, AO, Plotkin, Engeler, CDS}\}$. Then $\lambda_n^{\mathcal{T}} \models (\beta\text{-cnv})$.*

PROOF. (i) By Theorems 14A.7 and 14B.6(i).

(ii) Similarly, by Theorem 14B.6(ii). ■

η -conversion

First we give necessary and sufficient conditions for a system $\lambda_{\cap}^{\mathcal{T}}$ to satisfy the rule (η -red).

14B.8. THEOREM (Characterization of η -red). (i) Let $\mathcal{T} \in \text{TT}^{\mathbb{U}}$. Then

$$\lambda_{\cap}^{\mathcal{T}} \models (\eta\text{-red}) \Leftrightarrow \mathcal{T} \text{ is proper.}$$

(ii) Let $\mathcal{T} \in \text{TT}^{\mathbb{U}}$. Then

$$\lambda_{\cap}^{\mathcal{T}} \models (\eta\text{-red}) \Leftrightarrow \mathcal{T} \text{ is natural.}$$

PROOF. (i) Similarly, but simpler than (ii).

(ii) (\Rightarrow) Assume $\lambda_{\cap}^{\mathcal{T}} \models (\eta\text{-red})$ towards ($\rightarrow\cap$), (\rightarrow) and ($\mathbb{U}\rightarrow$).

As to ($\rightarrow\cap$), one has

$$x:(A\rightarrow B) \cap (A\rightarrow C), y:A \vdash xy : B \cap C,$$

hence by ($\rightarrow I$) it follows that $x:(A\rightarrow B) \cap (A\rightarrow C) \vdash \lambda y.xy : A\rightarrow(B \cap C)$. Therefore $x:(A\rightarrow B) \cap (A\rightarrow C) \vdash x : A\rightarrow(B \cap C)$, by (η -red). By Theorem 14A.9(i) one can conclude $(A\rightarrow B) \cap (A\rightarrow C) \leq A\rightarrow(B \cap C)$.

As to (\rightarrow), suppose that $A \leq B$ and $C \leq D$, in order to show $B\rightarrow C \leq A\rightarrow D$. One has $x:B\rightarrow C, y:A \vdash xy : C \leq D$, so $x:B\rightarrow C \vdash \lambda y.xy : A\rightarrow D$. Therefore by (η -red) it follows that $x:B\rightarrow C \vdash x : A\rightarrow D$ and we are done as before.

As to $\mathbb{U} \leq \mathbb{U}\rightarrow\mathbb{U}$, notice that $x:\mathbb{U}, y:\mathbb{U} \vdash xy : \mathbb{U}$, so we have $x:\mathbb{U} \vdash \lambda y.xy : \mathbb{U}\rightarrow\mathbb{U}$. Therefore $x:\mathbb{U} \vdash x : \mathbb{U}\rightarrow\mathbb{U}$ and again we are done.

(\Leftarrow) Let \mathcal{T} be natural. Assume that $\Gamma \vdash \lambda x.Mx : A$, with $x \notin \text{FV}(M)$, in order to show $\Gamma \vdash M : A$. If $A = \mathbb{U}$, we are done. Otherwise,

$$\begin{aligned} \Gamma \vdash \lambda x.Mx : A &\Rightarrow \Gamma, x:B_i \vdash Mx : C_i, 1 \leq i \leq k, \& \\ &(B_1\rightarrow C_1) \cap \dots \cap (B_k\rightarrow C_k) \leq A, \\ &\text{for some } B_1, \dots, B_k, C_1, \dots, C_k, \end{aligned}$$

by Theorem 14A.1(iii). We can suppose that $C_i \neq \mathbb{U}$ for all i . If there exists i such $C_i = \mathbb{U}$ then, by (\mathbb{U}) and ($\mathbb{U}\rightarrow$), we have $(B_i \rightarrow C_i) \cap D = \mathbb{U} \cap D = D$, for any type D . On the other hand, there is at least one $C_i \neq \mathbb{U}$, since otherwise $A \geq (B_1\rightarrow\mathbb{U}) \cap \dots \cap (B_k\rightarrow\mathbb{U}) = \mathbb{U}$, and we would have $A = \mathbb{U}$. Hence by Theorem 14A.9(ii)

$$\begin{aligned} \Rightarrow &\quad \Gamma, x:B_i \vdash M : D_i\rightarrow C_i \text{ and} \\ &\quad \Gamma, x:B_i \vdash x : D_i, \quad \text{for some } D_1, \dots, D_k, \\ \Rightarrow &\quad B_i \leq D_i, \quad \text{by Theorem 14A.9(i),} \\ \Rightarrow &\quad \Gamma \vdash M : (B_i\rightarrow C_i), \quad \text{by } (\leq\text{-L}) \text{ and } (\rightarrow), \\ \Rightarrow &\quad \Gamma \vdash M : ((B_1\rightarrow C_1) \cap \dots \cap (B_k\rightarrow C_k)) \leq A. \blacksquare \end{aligned}$$

14B.9. COROLLARY. Let $\mathcal{T} \in \{\text{Scott, Park, CDZ, HR, DHM, BCD, HL, CDV}\}$. Then $\lambda_{\cap}^{\mathcal{T}} \models (\eta\text{-red})$.

In order to characterize the admissibility of rule $(\eta\text{-exp})$, 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 14B.11.

14B.10. DEFINITION. Let $\mathcal{T} \in \text{TT}$.

(i) \mathcal{T} is called **η -sound** if for all A there are $k \geq 1$, $m_1, \dots, m_k \geq 1$ and B_1, \dots, B_k , C_1, \dots, C_k ,

$$\begin{pmatrix} D_{11} \dots D_{1m_1} \\ \dots \\ D_{k1} \dots D_{km_k} \end{pmatrix} \text{ and } \begin{pmatrix} E_{11} \dots E_{1m_1} \\ \dots \\ E_{k1} \dots E_{km_k} \end{pmatrix}$$

with

$$\begin{aligned} (B_1 \rightarrow C_1) \cap \dots \cap (B_k \rightarrow C_k) &\leq A \\ \& A \leq (D_{11} \rightarrow E_{11}) \cap \dots \cap (D_{1m_1} \rightarrow E_{1m_1}) \cap \dots \\ &\quad (D_{k1} \rightarrow E_{k1}) \cap \dots \cap (D_{km_k} \rightarrow E_{km_k}) \\ \& B_i \leq D_{i1} \cap \dots \cap D_{im_i} \& E_{i1} \cap \dots \cap E_{im_i} \leq C_i, \\ &\quad \text{for } 1 \leq i \leq k. \end{aligned}$$

(ii) Let $\mathcal{T} \in \text{TT}^{\mathbb{U}}$. Then \mathcal{T} is called **$\eta^{\mathbb{U}}$ -sound** if for all $A \neq \mathbb{U}$ at least one of the following two conditions holds.

- (1) There are types B_1, \dots, B_n with $(B_1 \rightarrow \mathbb{U}) \cap \dots \cap (B_n \rightarrow \mathbb{U}) \leq A$;
- (2) There are $n \geq k \geq 1$, $m_1, \dots, m_k \geq 1$ and B_1, \dots, B_k , C_1, \dots, C_k ,

$$\begin{pmatrix} D_{11} \dots D_{1m_1} \\ \dots \\ D_{k1} \dots D_{km_k} \end{pmatrix} \text{ and } \begin{pmatrix} E_{11} \dots E_{1m_1} \\ \dots \\ E_{k1} \dots E_{km_k} \end{pmatrix}$$

with

$$\begin{aligned} (B_1 \rightarrow C_1) \cap \dots \cap (B_k \rightarrow C_k) \cap \\ \cap (B_{k+1} \rightarrow \mathbb{U}) \cap \dots \cap (B_n \rightarrow \mathbb{U}) &\leq A \\ \& A \leq (D_{11} \rightarrow E_{11}) \cap \dots \cap (D_{1m_1} \rightarrow E_{1m_1}) \cap \dots \\ &\quad (D_{k1} \rightarrow E_{k1}) \cap \dots \cap (D_{km_k} \rightarrow E_{km_k}) \\ \& B_i \leq D_{i1} \cap \dots \cap D_{im_i} \& E_{i1} \cap \dots \cap E_{im_i} \leq C_i, \\ &\quad \text{for } 1 \leq i \leq k. \end{aligned}$$

The validity of η -expansion can be characterized as follows.

14B.11. THEOREM (Characterization of η -exp). (i) Let $\mathcal{T} \in \text{TT}^{-\mathbb{U}}$. Then

$$\lambda_{\cap}^{\mathcal{T}} \models (\eta\text{-exp}) \Leftrightarrow \mathcal{T} \text{ is } \eta\text{-sound}.$$

(ii) Let $\mathcal{T} \in \text{TT}^{\mathbb{U}}$. Then

$$\lambda_{\cap}^{\mathcal{T}} \models (\eta\text{-exp}) \Leftrightarrow \mathcal{T} \text{ is } \eta^{\mathbb{U}}\text{-sound}.$$

PROOF. (i) (\Rightarrow) Assume $\lambda_{\cap}^{\mathcal{T}} \models (\eta\text{-exp})$. As $x:A \vdash x : A$, by assumption we have $x:A \vdash \lambda y.xy : A$. From Theorem 14A.1(iii) it follows that $x:A, y:B_i \vdash xy : C_i$ and $(B_1 \rightarrow C_1) \cap \dots \cap (B_k \rightarrow C_k) \leq A$ for some B_i, C_i . By Theorem 14A.1(ii) for each i there exist D_{ij}, E_{ij} , such that for each j one has $x:A, y:B_i \vdash x : (D_{ij} \rightarrow E_{ij})$, and $x:A, y:B_i \vdash y : D_{ij}$ and $E_{i1} \cap \dots \cap E_{im_i} \leq C_i$. Hence by Theorem 14A.1(i) we have $A \leq (D_{ij} \rightarrow E_{ij})$ and $B_i \leq D_{ij}$ for all i and j . Therefore we obtain the condition of Definition 14B.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 14B.10(i).

$$\begin{aligned} (B_1 \rightarrow C_1) \cap \cdots \cap (B_k \rightarrow C_k) &\leq A \\ \& A \leq (D_{11} \rightarrow E_{11}) \cap \cdots \cap (D_{1m_1} \rightarrow E_{1m_1}) \cap \cdots \\ &\quad (D_{k1} \rightarrow E_{k1}) \cap \cdots \cap (D_{km_k} \rightarrow E_{km_k}) \\ \& B_i \leq D_{i1} \cap \cdots \cap D_{im_i} \& E_{i1} \cap \cdots \cap E_{im_i} \leq C_i, \\ &\text{for } 1 \leq i \leq k. \end{aligned}$$

By rule (\leq) for all i, j we have $\Gamma \vdash M : D_{ij} \rightarrow E_{ij}$ and so $\Gamma, x:D_{ij} \vdash Mx : E_{ij}$ by rule $(\rightarrow E)$. From $(\leq L)$, $(\cap I)$ and (\leq) we get $\Gamma, x:B_i \vdash Mx : C_i$ and this implies $\Gamma \vdash \lambda x.Mx : B_i \rightarrow C_i$, using rule $(\rightarrow I)$. 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 $(B_1 \rightarrow C_1) \cap \cdots \cap (B_k \rightarrow C_k) \leq A$ for some B_i, C_i . If all $C_i = U$, then A satisfies the first condition of Definition 14B.10(ii). Otherwise, consider the i such that $C_i \neq U$ 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 14B.10(ii), that is $(B_1 \rightarrow U) \cap \cdots \cap (B_n \rightarrow U) \leq A$, then by (U_{top}) it follows that $\Gamma, x:B_i \vdash Mx : U$, hence $\Gamma \vdash \lambda x.Mx : (B_1 \rightarrow U) \cap \cdots \cap (B_n \rightarrow U) \leq A$. Now let A satisfy the second condition. Then we reason as for (\Leftarrow) in (i). ■

For most intersection type theories of interest the condition of $\eta^{(U)}$ -soundness is deduced from the following proposition.

14B.12. PROPOSITION. Let $\mathcal{T} \in TT$ be proper, with set \mathbb{A} of atoms.

$$(i) \quad \mathcal{T} \text{ is } \eta\text{-sound} \Leftrightarrow \forall \alpha \in \mathbb{A} \exists k \geq 1 \exists B_1, \dots, B_k, C_1, \dots, C_k \alpha = (B_1 \rightarrow C_1) \cap \cdots \cap (B_k \rightarrow C_k).$$

(ii) Let $\mathcal{T} \in TT^U$. Then

$$\begin{aligned} \mathcal{T} \text{ is } \eta^U\text{-sound} \Leftrightarrow & \forall \alpha \in \mathbb{A} [U \rightarrow U \leq \alpha \vee \exists k \geq 1 \exists B_1, \dots, B_k, C_1, \dots, C_k \\ & [(B_1 \rightarrow C_1) \cap \cdots \cap (B_k \rightarrow C_k) \cap (U \rightarrow U) \leq \alpha \\ & \& \alpha \leq (B_1 \rightarrow C_1) \cap \cdots \cap (B_k \rightarrow C_k)]]. \end{aligned}$$

(iii) Let $\mathcal{T} \in NTT^U$. Then

$$\mathcal{T} \text{ is } \eta^U\text{-sound} \Leftrightarrow \mathcal{T} \text{ is } \eta\text{-sound}.$$

PROOF. (i) (\Rightarrow) Suppose \mathcal{T} is η -sound. Let $\alpha \in \mathbb{A}$. Then α satisfies the condition of Definition 14B.10(i), for some $B_1, \dots, B_k, C_1, \dots, C_k$,

$D_{11}, \dots, D_{1m_1}, \dots, D_{k1}, \dots, D_{km_k}, E_{11}, \dots, E_{1m_1}, \dots, E_{k1}, \dots, E_{km_k}$. By $(\rightarrow \cap)$ and (\rightarrow) , using Proposition 13A.21, it follows that

$$\begin{aligned} \alpha &\leq (D_{11} \cap \cdots \cap D_{1m_1} \rightarrow E_{11} \cap \cdots \cap E_{1m_1}) \cap \cdots \cap \\ &\quad (D_{k1} \cap \cdots \cap D_{km_k} \rightarrow E_{k1} \cap \cdots \cap E_{km_k}) \\ &\leq (B_1 \rightarrow C_1) \cap \cdots \cap (B_k \rightarrow C_k), \end{aligned}$$

hence $\alpha =_{\mathcal{T}} (B_1 \rightarrow C_1) \cap \cdots \cap (B_k \rightarrow C_k)$.

(\Leftarrow) By induction on the generation of A one can show that A satisfies the condition of η -soundness. The case $A_1 \rightarrow A_2$ is trivial and the case $A \equiv A_1 \cap A_2$ follows by the induction hypothesis and Rule (mon).

(ii) Similarly. Note that $(U \rightarrow U) \leq (B \rightarrow U)$ for all B .

(iii) Immediately by (ii) using rule $(U \rightarrow)$. ■

14B.13. COROLLARY. (i) Let $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{AO}\}$. Then \mathcal{T} is η^U -sound.

(ii) HL is η -sound.

PROOF. Easy. For AO in (i) one applies (ii) of Proposition 14B.12. ■

14B.14. COROLLARY. Let $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{AO}, \text{HL}\}$. Then

$$\lambda_n^{\mathcal{T}} \models (\eta\text{-exp}).$$

PROOF. By the previous Corollary and Theorem 14B.11. ■

Exercise 14C.21 shows that the remaining systems of Fig. 33 do not satisfy $(\eta\text{-exp})$.

Now we can harvest results towards closure under η -conversion.

14B.15. THEOREM. (i) Let $\mathcal{T} \in \text{TT}^{-U}$. Then

$$\lambda_n^{\mathcal{T}} \models (\eta\text{-cnv}) \Leftrightarrow \mathcal{T} \text{ is proper and } \eta\text{-sound}.$$

(ii) Let $\mathcal{T} \in \text{TT}^U$. Then

$$\lambda_n^{\mathcal{T}} \models (\eta\text{-cnv}) \Leftrightarrow \mathcal{T} \text{ is natural and } \eta^U\text{-sound}.$$

PROOF. (i) By Theorems 14B.8(i) and 14B.11(i).

(ii) By Theorems 14B.8(ii) and 14B.11(ii). ■

14B.16. THEOREM. For $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{HL}\}$ one has

$$\lambda_n^{\mathcal{T}} \models (\eta\text{-cnv}).$$

PROOF. By Corollaries 14B.9 and 14B.14. ■

14C. Exercises

14C.1. Let \subset be the inclusion relation as considered in Remark 13A.17. Prove that $\mathcal{T}_1 \subset \mathcal{T}_2 \Leftrightarrow \forall \Gamma, M, A [\Gamma \vdash^{\mathcal{T}_1} M : A \Rightarrow \Gamma \vdash^{\mathcal{T}_2} M : A]$.

14C.2. Show that for each number $n \in \mathbb{N}$ there is a type $A_n \in \text{TT}^{\text{CD}}$ such that for the Church numerals \mathbf{c}_n one has $\Gamma \vdash_n^{\text{CD}} \mathbf{c}_{n+1} : A_n$, but $\Gamma \not\vdash_n^{\text{CD}} \mathbf{c}_n : A_n$.

14C.3. Show that $S(KI)(II)$ and $(\lambda x.xxx)S$ are typable in \vdash_n^{CD} .

14C.4. Derive $\vdash_n^{\text{CDZ}} (\lambda x.xxx)S : 1$ and $y:0, z:0 \vdash_n^{\text{CDZ}} (\lambda x.xxx)(Syz) : 0$.

14C.5. Using the Inversion Lemmas show the following.

(i) $\vdash_n^{\text{CD}} \mathbf{c}_1 : \alpha \rightarrow \alpha$, where α is any constant.

(ii) $\vdash_n^{\text{HL}} K : 0$.

(iii) $\vdash_n^{\text{Scott}} I : 0$.

(iv) $\vdash_n^{\text{Plotkin}} I : 0$.

14C.6. Let \vdash be \vdash^{CD} and \leq be \leq_{CD} . Show

(i) Let

$$\begin{aligned} A &\equiv \alpha_1 \cap \cdots \cap \alpha_n \cap (B_1 \rightarrow C_1) \cap \cdots \cap (B_m \rightarrow C_m), \\ A' &\equiv \alpha'_1 \cap \cdots \cap \alpha'_{n'} \cap (B'_1 \rightarrow C'_1) \cap \cdots \cap (B'_{m'} \rightarrow C'_{m'}). \end{aligned}$$

Suppose $A \leq A'$. Then every α'_i is one of $\alpha_1, \dots, \alpha_n$ and every $B'_i \rightarrow C'_i$ is one of $(B_1 \rightarrow C_1), \dots, (B_m \rightarrow C_m)$.

(ii) Let $k \geq 1$. Then

$$\begin{aligned} (B_1 \rightarrow C_1) \cap \cdots \cap (B_k \rightarrow C_k) \leq (B \rightarrow C) &\Rightarrow \\ (B \rightarrow C) &\equiv (B_j \rightarrow C_j), \text{ for some } 1 \leq j \leq k. \end{aligned}$$

(iii) $\Gamma \vdash \lambda x.M : A \rightarrow B \Rightarrow \Gamma, x:A \vdash M : B$.

(iv) $\Gamma \vdash MN : A \Rightarrow \exists B, C [\Gamma \vdash M : B \rightarrow C \& \Gamma \vdash N : B]$.

14C.7. Let $\omega = \lambda x.xx$ and $\Omega = \omega\omega$. For \vdash being \vdash^{CD} we want to show

$$\begin{aligned} \not\vdash \Omega : A, &\quad \text{for all types } A; \\ \not\vdash K\Omega : A, &\quad \text{for all types } A. \end{aligned}$$

Prove this using the following steps.

(i) $\vdash \Omega : A \Rightarrow \exists B, C \vdash \omega : (B \rightarrow C) \cap B$.

(ii) $\vdash \omega : (B \rightarrow C) \cap B \Rightarrow \exists B', C'. \vdash \omega : (B' \rightarrow C') \cap B' \& B' \text{ is a proper subtype of } B$.

14C.8. Let $M \triangleq \lambda xy.Kx(xy)$. Show that for \vdash being \vdash^{CD} one has the following.

$$\begin{aligned} \not\vdash M : \alpha \rightarrow \beta \rightarrow \alpha; \\ \not\vdash I : ((\alpha \cap \beta) \rightarrow \gamma) \rightarrow ((\beta \cap \alpha) \rightarrow \gamma). \end{aligned}$$

[Hint. Use 14C.6.]

14C.9. We say that M and M' have the same types in Γ , notation $M \sim_{\Gamma} M'$ if

$$\forall A [\Gamma \vdash M : A \Leftrightarrow \Gamma \vdash M' : A].$$

Prove that $M \sim_{\Gamma} M' \Rightarrow M\vec{N} \sim_{\Gamma} M'\vec{N}$ for all \vec{N} .

14C.10. Let \mathcal{T} be a β -sound type theory that satisfies (U) and (U \rightarrow). Prove that for all A, B we have that $A \rightarrow B = \mathbb{U} \Leftrightarrow B = \mathbb{U}$.

14C.11. Using β -soundness, find out whether the following types are related or not with respect to \leq_{CDZ} .

$$(0 \rightarrow (1 \rightarrow 1) \rightarrow 0) \cap ((1 \rightarrow 1) \rightarrow 1), 0 \rightarrow 0 \rightarrow 0, (0 \rightarrow 0) \rightarrow 0 \text{ and } 1 \rightarrow (0 \rightarrow 0) \rightarrow 1.$$

14C.12. Let $\mathcal{T} \in \{\text{CDZ}, \text{HR}\}$. Consider the sequence of types defined by $A_0 = 0$ and $A_{n+1} = \mathbb{U} \rightarrow A_n$. Using Exercise 13E.3 and β -soundness, prove that \mathcal{T} does not have a bottom element at all.

14C.13. Show that Plotkin is β -sound by checking that it satisfies the following stronger condition.

$$\begin{aligned} (A_1 \rightarrow B_1) \cap \cdots \cap (A_n \rightarrow B_n) \leq C \rightarrow D &\Rightarrow \\ \exists k \neq 0 \exists i_1, \dots, i_k. 1 \leq i_j \leq n \& C = A_{i_j} \& B_{i_1} \cap \cdots \cap B_{i_k} = D. \end{aligned}$$

14C.14. Show that Engeler is β -sound by checking that it satisfies the following stronger condition:

$$(A_1 \rightarrow B_1) \cap \dots \cap (A_n \rightarrow B_n) \leq C \rightarrow D \& D \neq \mathbb{U} \Rightarrow \\ \exists k \neq 0 \exists i_1, \dots, i_k. 1 \leq i_j \leq n \& C = A_{i_j} \& B_{i_1} \cap \dots \cap B_{i_k} = D.$$

14C.15. Let $\mathbb{A}^{\mathcal{T}} = \{\mathbb{U}, 0\}$ 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 $\mathbb{U} \neq 0$].

14C.16. Prove that Theorem 14A.9(ii) still holds if the condition of properness is replaced by the following two conditions

$$A \leq_{\mathcal{T}} B \Rightarrow C \rightarrow A \leq_{\mathcal{T}} C \rightarrow B$$

$$(A \rightarrow B) \cap (C \rightarrow D) \leq_{\mathcal{T}} A \cap C \rightarrow B \cap D.$$

14C.17. Show that for $\mathcal{T} \in \text{TT}^{\mathbb{U}}$ the following condition

$$A \rightarrow B =_{\mathcal{T}} \mathbb{U} \rightarrow \mathbb{U} \Rightarrow B =_{\mathcal{T}} \mathbb{U}$$

is necessary for the admissibility of rule (β -red) in $\lambda_{\cap}^{\mathcal{T}}$. [Hint: Use Proposition 14B.1(ii).]

14C.18. Remember that the systems $\lambda_{\cap}^{\text{Krivine}}$ and $\lambda_{\cap}^{\text{Krivine}^{\mathbb{U}}}$ are defined in Exercise 13E.10.

- (i) Show that rules (β -red) and (β l-exp) are admissible in $\lambda_{\cap}^{\text{Krivine}}$, while (β -exp) is not admissible.
- (ii) Show that rules (β -red) and (β -exp) are admissible in $\lambda_{\cap}^{\text{Krivine}^{\mathbb{U}}}$.

14C.19. Show that for $\mathcal{T} \in \{\text{AO, Plotkin, Engeler, CDS, CD}\}$ one has

$$\lambda_{\cap}^{\mathcal{T}} \not\models (\eta\text{-red}).$$

14C.20. Verify the following.

- (i) Let $\mathcal{T} \in \{\text{BCD, Plotkin, Engeler, CDS}\}$. Then \mathcal{T} is not $\eta^{\mathbb{U}}$ -sound.
- (ii) Let $\mathcal{T} \in \{\text{CDV, CD}\}$. Then \mathcal{T} is not η -sound.

14C.21. Show that for $\mathcal{T} \in \{\text{BCD, Plotkin, Engeler, CDS, CDV, CD}\}$ one has

$$\lambda_{\cap}^{\mathcal{T}} \not\models (\eta\text{-exp}).$$

14C.22. Show that rules (η -red) and (η -exp) are not admissible in the systems $\lambda_{\cap}^{\text{Krivine}}$ and $\lambda_{\cap}^{\text{Krivine}^{\mathbb{U}}}$ as defined in Exercises 13E.10.

14C.23. Let \vdash denote derivability in the system obtained from the system $\lambda_{\cap}^{\text{CDV}}$ by replacing rule (\leq) by the rules ($\cap E$), see Definition 13B.8, and adding the rule

$$(R\eta) \quad \frac{\Gamma \vdash \lambda x. Mx : A}{\Gamma \vdash M : A} \quad \text{if } x \notin \text{FV}(M).$$

Show that $\Gamma \vdash_{\cap}^{\text{CDV}} M : A \Leftrightarrow \Gamma \vdash M : A$.

14C.24. (Barendregt, Coppo, and Dezani-Ciancaglini [1983]) Let \vdash denote derivability in the system obtained from $\lambda_{\cap}^{\text{BCD}}$ by replacing rule (\leq) by the rules ($\cap E$) and adding ($R\eta$) as defined in Exercise 14C.23. Verify that

$$\Gamma \vdash_{\cap}^{\text{BCD}} M : A \Leftrightarrow \Gamma \vdash M : A.$$

14C.25. Let Δ be a basis that is allowed to be infinite. We define $\Delta \vdash M : A$ if 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 $(\rightarrow I)$.
- (ii) Suppose $\text{dom}(\Delta)$ is the set of all the variables. Show that the rule $(\rightarrow I)$ is derivable if it is reformulated as

$$\Delta_x, x:A \vdash M : B \Rightarrow \Delta \vdash (\lambda x.M) : (A \rightarrow B),$$

with Δ_x the result of removing any $x:C$ from Δ .

- (iii) Reformulate and prove Propositions 13B.10, 13B.12, Theorems 14A.1 and 14A.9 for infinite bases.

14C.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 $\Delta \vdash M : A$ if there exists a singled (only one declaration per variable) basis $\Gamma \subseteq \Delta$ such that $\Gamma \vdash M : A$.

- (i) Show that $x : \alpha_1, x : \alpha_2 \not\vdash^{\text{CD}} x : \alpha_1 \cap \alpha_2$.
- (ii) Show that $x : \alpha_1 \rightarrow \alpha_2, x : \alpha_1 \not\vdash^{\text{CD}} xx : \alpha_2$.
- (iii) Consider $\Delta = \{x : \alpha_1 \cap \alpha_2, x : \alpha_1\}$; Show that $\Delta, x : A \vdash^{\text{CD}} M : B$, but $A = \alpha_2$;
 $B = (\alpha_1 \rightarrow \alpha_2 \rightarrow \alpha_3) \rightarrow \alpha_3$;
 $M = \lambda y.yxx$.
 $\Delta \not\vdash^{\text{CD}} (\lambda x.M) : (A \rightarrow B)$.
- (iv) We say that a multi-basis is closed under \cap if for all $x \in \text{dom}(\Delta)$ 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 13B.3, except for $(\rightarrow I)$, are derivable for (possibly infinite) multi-bases that are closed under \cap .
- (v) Let Δ be closed under \cap . We define

$$\Delta[x := X] \triangleq \{y : \Delta(y) \mid y \neq x\} \cup \{x : A \mid A \in X\}.$$

Prove that the following reformulation of $(\rightarrow I)$ using principal filters is derivable

$$\frac{\Delta[x := \uparrow B] \vdash N : C}{\Delta \vdash \lambda x.N : B \rightarrow C}.$$

- (vi) Prove Propositions 13B.10, 13B.12, Theorems 14A.1 and 14A.9 for (possible infinite) multi-bases reformulating the statements whenever it is necessary.
- (vii) Prove that if $\Delta(x)$ is a filter then $\{A \mid \Delta \vdash x : A\} = \Delta(x)$.

14C.27.

- (i) Prove that $F^\mathcal{T} \circ G^\mathcal{T} \supseteq \text{Id}_{\mathcal{F}\mathcal{T}}$ for all $\mathcal{T} \in \text{TT}$.
- (ii) Prove that $F^\mathcal{T} \circ G^\mathcal{T} \subseteq \text{Id}_{\mathcal{F}\mathcal{T}}$ iff \mathcal{T} is β -sound.
- (iii) Construct a natural type theory \mathcal{T} such that $F^\mathcal{T} \circ G^\mathcal{T} \neq \text{Id}_{\mathcal{F}\mathcal{T}}$.

14C.28. Let $\mathcal{T} \in \text{TT}^{\text{U}}$.

- (i) Prove that $G^\mathcal{T} \circ F^\mathcal{T} \subseteq \text{Id}_{\mathcal{F}\mathcal{T}}$ iff \mathcal{T} is proper.
- (ii) Prove that $G^\mathcal{T} \circ F^\mathcal{T} \supseteq \text{Id}_{\mathcal{F}\mathcal{T}}$ iff \mathcal{T} is η -sound.

14C.29. Let $\mathcal{T} \in \text{TT}^{\mathbb{U}}$.

- (i) Prove that $G^{\mathcal{T}} \circ F^{\mathcal{T}} \subseteq \text{Id}_{\mathcal{F}\mathcal{T}}$ iff \mathcal{T} is natural.
- (ii) Prove that $G^{\mathcal{T}} \circ F^{\mathcal{T}} \supseteq \text{Id}_{\mathcal{F}\mathcal{T}}$ iff \mathcal{T} is $\eta^{\mathbb{U}}$ -sound.

CHAPTER 15

TYPE AND LAMBDA STRUCTURES

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 16C 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 15B.

As a warm-up, a proto-type result in this direction is Corollary 15A.20, stating the equivalence between the categories \mathbf{MSL}^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 \mathcal{D} an element d is fully determined by its compact, see Definition 10A.4(iii), approximations

$$d = \bigsqcup \{a \mid a \sqsubseteq d \ \& \ a \text{ compact}\}.$$

Writing $\mathcal{K}(\mathcal{D}) = \{a \in \mathcal{D} \mid a \text{ compact}\}$ and $\uparrow a = \{d \in \mathcal{D} \mid a \sqsubseteq d\}$ one has

$$a \sqsubseteq b \Leftrightarrow \uparrow b \subseteq \uparrow a.$$

Therefore it is natural to write for $a, b \in \mathcal{K}(\mathcal{D})$

$$b \leq a \Leftrightarrow a \sqsubseteq b.$$

The complete lattice \mathcal{D} can be reconstructed from the meet semi-lattice \mathcal{S} by a general construction that assigns to \mathcal{S} the collection $\mathcal{F}^{\mathcal{S}}$ of *filters* (defined in Definition 13D.1 for type theories, see also Remark 13D.6 for the translation to type structures), forming an algebraic lattice. We will show the isomorphisms

$$\begin{aligned} \mathcal{S} &\cong \mathcal{K}(\mathcal{F}^{\mathcal{S}}); \\ \mathcal{D} &\cong \mathcal{F}^{\mathcal{K}(\mathcal{D})}. \end{aligned}$$

In fact the isomorphisms are functional and constitute an equivalence between the categories \mathbf{MSL}^U and \mathbf{ALG}_a . Similarly, for the strict case, the equivalence between \mathbf{MSL}^U and \mathbf{ALG}_a^s is proved. See Fig. 38. For the injectivity of the map establishing $\mathcal{S} \cong \mathcal{K}(\mathcal{F}^{\mathcal{S}})$ in Proposition 15A.19 one needs that \leq is a partial order. Therefore the results of this section do not work for type theories in general.

$$\begin{array}{ccc}
 & \mathbf{MSL}^U \cong \mathbf{ALG}_a & \\
 \text{meet semi-} & | & | \\
 \text{lattices} & \mathbf{MSL}^U \cong \mathbf{ALG}_a^s & \text{algebraic} \\
 & | & | \\
 & \mathbf{ALG}_a & \text{lattices}
 \end{array}$$

FIGURE 38. Equivalences proved in Section 15A

To interpret untyped λ -terms the objects \mathcal{D} in \mathbf{ALG} are not enough. We need some more structure, \mathcal{D} enriched to $\langle \mathcal{D}, F, G \rangle$, where $F \in [\mathcal{D} \rightarrow [\mathcal{D} \rightarrow \mathcal{D}]]$ and $G \in [[\mathcal{D} \rightarrow \mathcal{D}] \rightarrow \mathcal{D}]$. These are called lambda structures. There is a canonical way to define such a pair F, G for \mathcal{D} being a filter structure obtained from a type theory, see Definition 13D.5 and Exercise 14C.27. Then we always have

$$F \circ G \sqsupseteq \text{Id}_{\mathcal{D}}$$

This does not yet imply the validity of the β -axiom

$$(\beta) (\lambda x.M)N = M[x := N]$$

For β to hold it suffices to require

$$(\text{fun-}\beta) F \circ G = \text{Id}_{\mathcal{D}}$$

Looking for the ‘right’ category of type structures corresponding to lambda structures there are other considerations. The axioms (\rightarrow) , $(\rightarrow\cap)$ and $(U\rightarrow)$ are natural considering the intended semantics of λ_{\rightarrow} given by [Scott \[1975a\]](#) generalized to λ_{\cap} . The ‘natural type structures’, satisfying (\rightarrow) , $(\rightarrow\cap)$ and $(U\rightarrow)$, exactly correspond to ‘natural lambda structures’ satisfying

- (1) $F \circ G \sqsupseteq \text{Id}_{[\mathcal{D} \rightarrow \mathcal{D}]}$
- (2) $G \circ F \sqsubseteq \text{Id}_{\mathcal{D}}$

as suggested in Exercise 14C.29. Moreover, these axioms suffice to make the connection between type structures and lambda structures categorical, see Section 15B.

$$\mathbf{NTS}^U \cong \mathbf{NLS}$$

FIGURE 39. Equivalence proved in Section 15B

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 33 all satisfy $(\text{fun-}\beta)$ and hence are λ -models. See Exercise 14C.27 for a natural lambda structure that is not a λ -model.

Weakening one of the axioms for natural type structures, taking (U_{lazy}) instead of $(U\rightarrow)$, 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 16D.

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 $P\omega$ by Scott and \mathcal{D}_A by

Engeler (see respectively Section 18.1 and 18.4.2 of B[1984]). This can be done, but we cannot rely on $P\omega$ or \mathcal{D}_A as lambda structures for the following reason. The essence of the graph models is that there are maps

$$i : \mathcal{K}(\mathcal{D}) \times \text{Prime}(\mathcal{D}) \rightarrow \text{Prime}(\mathcal{D}),$$

where

$$\text{Prime}(\mathcal{D}) \triangleq \{d \in \mathcal{D} \mid \forall A \subseteq \mathcal{D}. d \sqsubseteq \bigsqcup A \Rightarrow \exists a \in A. d \sqsubseteq a\}.$$

In the case of $P\omega$ and \mathcal{D}_A the subset $\text{Prime}(\mathcal{D})$ consists of the singletons. Although we can define from these maps i appropriate F_i, G_i , from these one cannot reconstruct i .

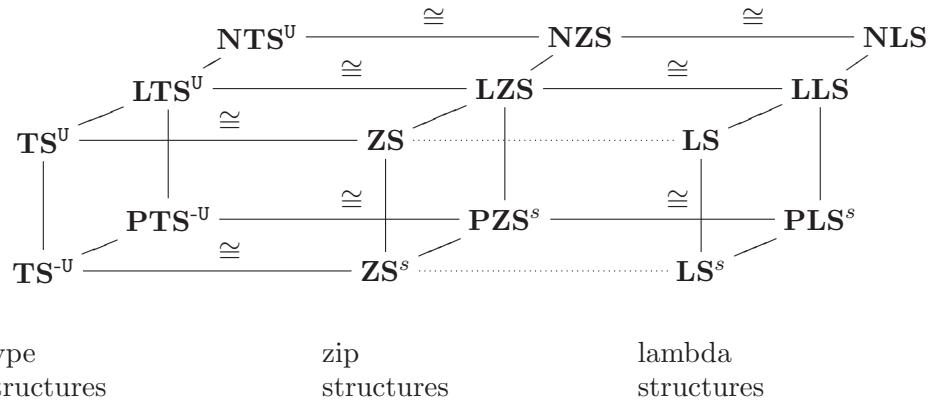


FIGURE 40. Equivalences proved in Sections 15C and 15D

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

$$Z : \mathcal{K}(\mathcal{D}) \times \mathcal{K}(\mathcal{D}) \rightarrow \mathcal{K}(\mathcal{D})$$

in order to be able to relate it in a simple way to type structures: $Z(a, b)$ roughly corresponds to $(a \rightarrow b)$. In this way we obtain the so-called zip structures $\langle \mathcal{D}, Z \rangle$. For these structures one can define a corresponding lambda structure $\langle \mathcal{D}, F_Z, G_Z \rangle$ by

$$F_Z(x)(y) \triangleq \bigsqcup \{b \mid \exists a \sqsubseteq y. Z(a, b) \sqsubseteq x\},$$

$$G_Z(f) \triangleq \bigsqcup \{Z(a, b) \mid b \sqsubseteq f(a)\}.$$

Finally the type structures whose filter models are $P\omega$ and \mathcal{D}_A , respectively, can be defined by weakening the conditions (\rightarrow) , $(\rightarrow \cap)$.

Even if we can establish an equivalence between the categories of type and zip structures, we lose the equivalence between the general categories of type structures and lambda structures. This situation is studied in detail in Sections 15C and 15D. 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 \rightarrow ; the zip and lambda structures are expansions of algebraic lattices \mathcal{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 $F : \mathcal{D} \rightarrow [\mathcal{D} \rightarrow \mathcal{D}]$ and $G : [\mathcal{D} \rightarrow \mathcal{D}] \rightarrow \mathcal{D}$. 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 Fig.s 38, 39 and 40.

15A. Meet semi-lattices and algebraic lattices

The results of this section are either known or relatively simple modifications of known ones, see [Gierz, Hofmann, Keimel, Lawson, Mislove, and Scott \[1980\]](#).

Categories of meet semi-lattices

Remember the following notions, see Definitions 13C.8-13C.11. The category **MSL** has as objects at most countable meet semi-lattices and as morphisms maps preserving \leq and \cap .

The category **MSL^U** is as **MSL**, but based on top meet semi-lattices. So now morphisms also should preserve the top.

The category **TS^U** has as objects the at most countable type structures and as morphisms maps $f : \mathcal{S} \rightarrow \mathcal{S}'$, preserving \leq and \cap , and which moreover satisfy the following:

$$(\text{mon-}\rightarrow) \quad f(s) \rightarrow f(t) \leq f(s \rightarrow t).$$

The category **TS^U** is as **TS^U**, but based on top type structures. Now also morphisms should preserve the top.

In Definition 13C.11 we defined three full subcategories of **TS^U** and one of **TS^U**, by specifying in each case the objects: **GTS^U** with as objects the graph top type structures; **LTS^U** with as objects the lazy top type structures; **NTS^U** with as objects the natural top type structures; **PTS^U** with as objects the proper type structures.

Categories of algebraic lattices

Remember the notions from domain theory presented in 10A.4-10A.12.

15A.1. DEFINITION. Let \mathcal{D} be a complete lattice.

- (i) For $x \in \mathcal{D}$ define $\mathcal{K}(x) \triangleq \{d \in \mathcal{K}(\mathcal{D}) \mid d \sqsubseteq x\}$.
- (ii) On $\mathcal{K}(\mathcal{D})$ define

$$d \leq e \iff e \sqsubseteq d.$$

- (iii) A function $f \in [\mathcal{D} \rightarrow \mathcal{D}]$ is called *strict* if $f(\perp_{\mathcal{D}}) = \perp_{\mathcal{D}}$.
- (iv) Write $[\mathcal{D} \rightarrow_s \mathcal{D}] \triangleq \{f \in [\mathcal{D} \rightarrow \mathcal{D}] \mid f \text{ is strict}\}$.
- (v) $\mathcal{K}^s(\mathcal{D}) \triangleq \mathcal{K}(\mathcal{D}) - \{\perp_{\mathcal{D}}\}$.

When useful we will decorate $\sqsubseteq, \sqcup, \sqcap, \perp, \top, \sqcup$ and \sqcap with \mathcal{D} , e.g. $\sqsubseteq_{\mathcal{D}}$ etcetera. In this Chapter $a, b, c, d \dots$ often range over compact elements in lattices, while $x, y, z \dots$ range over general elements.

The following connects the notion of compact element to the notion of compact subset of a topological space.

15A.2. LEMMA. Let \mathcal{D} be a complete lattice.

(i) $d \in \mathcal{D}$ is compact if

$$\forall Z \subseteq \mathcal{D}. [d \sqsubseteq \bigsqcup Z \Rightarrow \exists Z_0 \subseteq Z. [Z_0 \text{ is finite} \& d \sqsubseteq \bigsqcup Z_0]].$$

(ii) If a, b are compact, then $a \sqcup b$ is compact.

(iii) For $a, b \in \mathcal{K}(\mathcal{D})$ one has $a \sqcap_{\mathcal{K}(\mathcal{D})} b = a \sqcup_{\mathcal{D}} b$.

(iv) $(\mathcal{K}(\mathcal{D}), \leq) \in \mathbf{MSL}^{\mathbb{U}}$.

PROOF. (i) (\Rightarrow) Suppose $d \in \mathcal{D}$ is compact. Given $Z \subseteq \mathcal{D}$, 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

$$\begin{aligned} d \sqsubseteq \bigsqcup Z &\Rightarrow d \sqsubseteq \bigsqcup Z^+ \\ &\Rightarrow \exists z^+ \in Z^+. d \sqsubseteq z^+ \\ &\Rightarrow \exists Z_0 \subseteq Z. d \sqsubseteq \bigsqcup Z_0 \& Z_0 \text{ is finite.} \end{aligned}$$

(\Leftarrow) Suppose $d \sqsubseteq \bigsqcup Z$ with $Z \subseteq \mathcal{D}$ 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 $d = \perp_{\mathcal{D}}$ and we can take an arbitrary element z in the non-empty Z satisfying $d \sqsubseteq z$.

(ii) If $a \sqcup b$ is ‘covered’ (in the sense of \sqsubseteq) by the union of a family Z , then each of a, b is covered by a finite subset of Z by (i). Therefore also $a \sqcup b$ is covered by a finite subset of Z .

(iii) By (ii) $(a \sqcup_{\mathcal{D}} b) \in \mathcal{K}(\mathcal{D})$; now turn things around.

(iv) Immediate from (iii), noticing that $\sqcup_{\mathcal{K}(\mathcal{D})} = \perp_{\mathcal{D}} \in \mathcal{K}(\mathcal{D})$. ■

Instead of \sqcap_{\leq} we often write \sqcap_{\leq} or simply \sqcap .

15A.3. PROPOSITION. Let $\mathcal{D}, \mathcal{D}'$ be algebraic lattices.

(i) Let $f \in [\mathcal{D} \rightarrow \mathcal{D}']$. Then for $x \in \mathcal{D}$

$$f(x) = \bigsqcup \{f(a) \mid a \in \mathcal{K}(x)\}.$$

(ii) Let $f, g \in [\mathcal{D} \rightarrow \mathcal{D}']$. Suppose $f \upharpoonright \mathcal{K}(\mathcal{D}) = g \upharpoonright \mathcal{K}(\mathcal{D})$. 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). ■

15A.4. DEFINITION. If $e \in \mathcal{D}$, $e' \in \mathcal{D}'$, then $e \mapsto e'$ is the *step function* defined by

$$(e \mapsto e')(d) \triangleq \begin{cases} e' & \text{if } e \sqsubseteq d, \\ \perp_{\mathcal{D}'} & \text{otherwise.} \end{cases}$$

15A.5. LEMMA. $[\mathcal{D} \rightarrow \mathcal{D}']$ is a complete lattice with

$$(\bigsqcup_{f \in X} f)(d) = \bigsqcup_{f \in X} f(d).$$

15A.6. LEMMA. For $d, e \in \mathcal{D}$, $d', e' \in \mathcal{D}'$ and $f \in [\mathcal{D} \rightarrow \mathcal{D}']$ one has

- (i) d compact $\Rightarrow d \mapsto d'$ is continuous.
- (ii) $d \mapsto d'$ is continuous and $d' \neq \perp \Rightarrow d$ is compact.
- (iii) d' compact $\Leftrightarrow d \mapsto d'$ compact.
- (iv) $d' \sqsubseteq f(d) \Leftrightarrow (d \mapsto d') \sqsubseteq f$.
- (v) $e \sqsubseteq d \& d' \sqsubseteq e' \Rightarrow (d \mapsto d') \sqsubseteq (e \mapsto e')$.

$$(vi) (d \mapsto d') \sqcup (e \mapsto e') \sqsubseteq (d \sqcap e) \mapsto (d' \sqcup e').$$

PROOF. Easy. ■

15A.7. LEMMA. For all $e, d_1, \dots, d_n \in \mathcal{D}$, $e', d'_1, \dots, d'_n \in \mathcal{D}'$

$$\begin{aligned} (e \mapsto e') \sqsubseteq (d_1 \mapsto d'_1) \sqcup \dots \sqcup (d_n \mapsto d'_n) &\Leftrightarrow \\ &\Leftrightarrow \exists I \subseteq \{1, \dots, n\} \bigsqcup_{i \in I} d_i \sqsubseteq e \ \& \ e' \sqsubseteq \bigsqcup_{i \in I} d'_i]. \end{aligned}$$

Clearly in (\Rightarrow) we have $I \neq \emptyset$ if $e' \neq \perp_{\mathcal{D}'}$.

PROOF. Easy. ■

15A.8. PROPOSITION. Let $\mathcal{D}, \mathcal{D}' \in \mathbf{ALG}$.

- (i) For $f \in [\mathcal{D} \rightarrow \mathcal{D}']$ one has $f = \bigsqcup \{a \mapsto a' \mid a' \sqsubseteq f(a), a \in \mathcal{K}(\mathcal{D}), a' \in \mathcal{K}(\mathcal{D}')\}$.
- (ii) Let $\mathcal{D} \in \mathbf{ALG}$ and let $f \in [\mathcal{D} \rightarrow \mathcal{D}']$ be compact. Then

$$f = (a_1 \mapsto a'_1) \sqcup \dots \sqcup (a_n \mapsto a'_n),$$

for some $a_1, \dots, a_n \in \mathcal{K}(\mathcal{D})$, $a'_1, \dots, a'_n \in \mathcal{K}(\mathcal{D}')$.

- (iii) $[\mathcal{D} \rightarrow \mathcal{D}'] \in \mathbf{ALG}$.

PROOF. (i) It suffices to show that RHS and LHS are equal when applied to an arbitrary element $d \in \mathcal{D}$.

$$\begin{aligned} f(d) &= f(\bigsqcup \{a \mid a \sqsubseteq d \ \& \ a \in \mathcal{K}(\mathcal{D})\}) \\ &= \bigsqcup \{f(a) \mid a \sqsubseteq d \ \& \ a \in \mathcal{K}(\mathcal{D})\} \\ &= \bigsqcup \{a' \mid a' \sqsubseteq f(a) \ \& \ a \sqsubseteq d \ \& \ a \in \mathcal{K}(\mathcal{D}), a' \in \mathcal{K}(\mathcal{D}')\} \\ &= \bigsqcup \{(a \mapsto a')(d) \mid a' \sqsubseteq f(a) \ \& \ a \sqsubseteq d \ \& \ a \in \mathcal{K}(\mathcal{D}), a' \in \mathcal{K}(\mathcal{D}')\} \\ &= \bigsqcup \{(a \mapsto a')(d) \mid a' \sqsubseteq f(a) \ \& \ a \in \mathcal{K}(\mathcal{D}), a' \in \mathcal{K}(\mathcal{D}')\} \\ &= \bigsqcup \{(a \mapsto a') \mid a' \sqsubseteq f(a) \ \& \ a \in \mathcal{K}(\mathcal{D}), a' \in \mathcal{K}(\mathcal{D}')\}(d). \end{aligned}$$

(ii) For f compact one has $f = \bigsqcup \{a \mapsto a' \mid a' \sqsubseteq f(a) \ \& \ a \in \mathcal{K}(\mathcal{D}), a' \in \mathcal{K}(\mathcal{D}')\}$, by (i). Hence by Lemma 15A.2(i) for some $a_1, \dots, a_n \in \mathcal{K}(\mathcal{D})$, $a'_1, \dots, a'_n \in \mathcal{K}(\mathcal{D}')$

$$(1) \quad f = (a_1 \mapsto a'_1) \sqcup \dots \sqcup (a_n \mapsto a'_n).$$

(iii) It remains to show that there are only countably many compact elements in $[\mathcal{D} \rightarrow \mathcal{D}]$. Since $\mathcal{K}(\mathcal{D})$ is countable, there are only countably many expressions in the RHS of (1). (The cardinality is $\leq \Sigma_n n \cdot \aleph_0^2 = \aleph_0$.) Therefore there are countable many compact $f \in [\mathcal{D} \rightarrow \mathcal{D}]$. (There may be more expressions on the RHS for one f , but this results in less compact elements.) ■

15A.9. DEFINITION. (i) The category \mathbf{ALG}_a has the same objects as \mathbf{ALG} and as morphisms $\mathbf{ALG}_a(\mathcal{D}, \mathcal{D}')$ maps $f \in [\mathcal{D} \rightarrow \mathcal{D}']$ that satisfy the properties ‘compactness preserving’ and ‘additive’:

$$\begin{aligned} (\text{cmp-pres}) \quad &\forall a \in \mathcal{K}(\mathcal{D}). f(a) \in \mathcal{K}(\mathcal{D}'); \\ (\text{add}) \quad &\forall X \subseteq \mathcal{D}. f(\bigsqcup X) = \bigsqcup f(X). \end{aligned}$$

(ii) The category \mathbf{ALG}_a^s has the same objects as \mathbf{ALG}_a and as morphisms $\mathbf{ALG}_a^s(\mathcal{D}, \mathcal{D}')$ maps $f \in [\mathcal{D} \rightarrow \mathcal{D}']$ satisfying (cmp-pres), (add) and

$$(s) \quad \forall d \in \mathcal{D}. [f(d) = \perp_{\mathcal{D}'} \Rightarrow d = \perp_{\mathcal{D}}].$$

15A.10. REMARK. (i) Note that the requirement (add) implies that a morphism f is continuous (preserving sups for directed subsets X) and strict ($f(\perp_{\mathcal{D}}) = \perp_{\mathcal{D}'}$).

(ii) Remember that $\mathcal{K}^s(\mathcal{D}) = \mathcal{K}(\mathcal{D}) - \{\perp_{\mathcal{D}}\}$. Note that $\mathbf{ALG}_a^s(\mathcal{D}, \mathcal{D}')$ consists of maps satisfying

$$\begin{aligned} (\text{cmp-pres}^s) \quad & \forall a \in \mathcal{K}^s(\mathcal{D}). f(a) \in \mathcal{K}^s(\mathcal{D}'); \\ (\text{add}) \quad & \forall X \subseteq \mathcal{D}. f(\sqcup X) = \sqcup f(X). \end{aligned}$$

(iii) By contrast to Proposition 15A.8(iii) $\mathbf{ALG}_a(\mathcal{D}, \mathcal{D}') \notin \mathbf{ALG}_a$, 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.

15A.11. LEMMA. (i) Let $f : \mathcal{D} \rightarrow \mathcal{D}'$ be a continuous function with $\mathcal{D}, \mathcal{D}' \in \mathbf{ALG}$. Then, for any $X \subseteq \mathcal{D}$, $b' \in \mathcal{K}(\mathcal{D}')$,

$$b' \sqsubseteq f(\sqcup X) \Leftrightarrow \exists Z \subseteq_{fin} X \cap \mathcal{K}(\mathcal{D}). b' \sqsubseteq f(\sqcup Z).$$

(ii) A map $f \in [\mathcal{D} \rightarrow \mathcal{D}']$ satisfies (add) iff f is Scott continuous and

$$(2) \quad \forall X \subseteq_{fin} \mathcal{K}(\mathcal{D}). f(\sqcup X) = \sqcup f(X).$$

PROOF. (i) Note that $\Xi = \{\sqcup Z \mid Z \subseteq_{fin} X \cap \mathcal{K}(\mathcal{D})\}$ is a directed set and $\sqcup \Xi = \sqcup X$. Moreover, by monotonicity of f , also the set $\{f(\sqcup Z) \mid Z \subseteq_{fin} X \cap \mathcal{K}(\mathcal{D})\}$ is directed. Therefore

$$\begin{aligned} b' \sqsubseteq f(\sqcup X) &\Leftrightarrow b' \sqsubseteq f(\sqcup \Xi) \\ &\Leftrightarrow b' \sqsubseteq \sqcup f(\Xi), \quad \text{since } f \text{ is continuous,} \\ &\Leftrightarrow b' \sqsubseteq \sqcup \{f(\sqcup Z) \mid Z \subseteq_{fin} X \cap \mathcal{K}(\mathcal{D})\}, \quad \text{by definition of } \Xi, \\ &\Leftrightarrow \exists Z \subseteq_{fin} X \cap \mathcal{K}(\mathcal{D}). b' \sqsubseteq f(\sqcup Z), \quad \text{since } b' \text{ is compact.} \end{aligned}$$

(ii) The non-trivial direction is to show, assuming f is Scott continuous and satisfies (2), that f is additive. By monotonicity of f we only need to show for all $X \subseteq \mathcal{D}$

$$f(\sqcup X) \sqsubseteq \sqcup f(X).$$

As \mathcal{D}' is algebraic, it suffices to assume $b' \sqsubseteq f(\sqcup X)$ and conclude $b' \sqsubseteq \sqcup f(X)$. By (i) $\exists Z \subseteq_{fin} X \cap \mathcal{K}(\mathcal{D}). b' \sqsubseteq f(\sqcup Z) = \sqcup f(Z)$, so $b' \sqsubseteq \sqcup f(X)$. ■

15A.12. LEMMA. Let $\mathcal{S} \in \mathbf{MSL}^U$ 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

$$\sqcup_{i \in I} \uparrow s_i = \uparrow \bigcap_{i \in I} s_i.$$

(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. ■

15A.13. LEMMA. (i) Let $\mathcal{X} \subseteq \mathcal{F}^{\mathcal{K}(\mathcal{D})}$. Then taking sups in \mathcal{D} one has

$$\sqcup(\bigcup \mathcal{X}) = \bigsqcup_{X \in \mathcal{X}} (\sqcup X).$$

(ii) Let $\theta \subseteq \mathcal{K}(\mathcal{D})$ be non-empty. Then taking the sups in \mathcal{D} one has

$$\sqcup(\uparrow\theta) = \sqcup\theta.$$

PROOF. (i) Realizing that a sup (in \mathcal{D}) of a union of $\{Y_i\}_{i \in I} \subseteq \mathcal{K}(\mathcal{D})$ is the sup of the sups $\sqcup Y_i$, one has

$$\sqcup(\bigcup_{i \in I} Y_i) = \bigsqcup_{i \in I} (\sqcup Y_i).$$

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 $\mathcal{K}(\mathcal{D})$. Now the order in this \mathbf{MSL}^U is the reverse of the one induced by \mathcal{D} , therefore $\uparrow\theta$ is obtained by taking smaller elements and unions (in \mathcal{D}). But then taking the big union the result is the same. ■

We now will establish the following equivalences of categories.

$$\begin{aligned}\mathbf{MSL}^U &\cong \mathbf{ALG}_a; \\ \mathbf{MSL}^{-U} &\cong \mathbf{ALG}_a^s.\end{aligned}$$

Equivalence between \mathbf{MSL}^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 13D.1-13D.4 can be translated immediately to meet semi-lattices.

15A.14. DEFINITION. We define a map $\text{Flt} : \mathbf{MSL}^U \rightarrow \mathbf{ALG}_a$, that will turn out to be a functor, as follows.

(i) On objects $\mathcal{S} \in \mathbf{MSL}^U$ one defines

$$\text{Flt}(\mathcal{S}) \triangleq \langle \mathcal{F}^{\mathcal{S}}, \subseteq \rangle.$$

(ii) On morphisms $f : \mathcal{S} \rightarrow \mathcal{S}'$ one defines $\text{Flt}(f) : \mathcal{F}^{\mathcal{S}} \rightarrow \mathcal{F}^{\mathcal{S}'}$ by

$$\text{Flt}(f)(X) \triangleq \{s' \mid \exists s \in X. f(s) \leq s'\}.$$

15A.15. LEMMA. Let $f \in \mathbf{MSL}^U(\mathcal{S}, \mathcal{S}')$.

(i) For $X \subseteq \mathcal{S}$, one has $\text{Flt}(f)(\uparrow X) = \uparrow \{f(s) \mid s \in X\}$.

(ii) For $s \in \mathcal{S}$ one has $\text{Flt}(f)(\uparrow s) = \uparrow f(s)$. ■

PROOF. We only prove (ii).

$$\begin{aligned}\text{Flt}(f)(\uparrow s) &= \{s' \mid \exists t \in \uparrow s. f(t) \leq s'\}, \\ &= \{s' \mid \exists t \geq s. f(t) \leq s'\}, \\ &= \{s' \mid f(s) \leq s'\}, \quad \text{since } f \text{ is monotone,} \\ &= \uparrow f(s).\end{aligned}$$

15A.16. PROPOSITION. Flt is a functor from \mathbf{MSL}^U to \mathbf{ALG}_a .

PROOF. We have to prove that Flt transforms a morphism in \mathbf{MSL}^U into a morphism in \mathbf{ALG}_a . Let $f \in \mathbf{MSL}^U(\mathcal{S}, \mathcal{S}')$, $\uparrow s \in K(\mathcal{F}^{\mathcal{S}})$. By Lemma 15A.15(ii) $\text{Flt}(f)(\uparrow s) = \uparrow f(s)$, which is compact in $\mathcal{F}^{\mathcal{S}'}$, hence $\text{Flt}(f)$ satisfies (cmp-pres).

$\text{Flt}(f)$ satisfies (add). Indeed, by Lemma 15A.11(ii) and the fact that $\text{Flt}(f)$ 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

$$\begin{aligned}\text{Flt}(f)(\bigsqcup_{i \in I} \uparrow s_i) &= \text{Flt}(f)(\uparrow \bigcap_{i \in I} s_i), \quad \text{by Lemma 15A.12(ii)}, \\ &= \uparrow f(\bigcap_{i \in I} s_i), \quad \text{by Lemma 15A.15(ii)}, \\ &= \uparrow \bigcap_{i \in I} f(s_i), \quad \text{since } f \text{ commutes with } \cap, \\ &= \bigsqcup_{i \in I} \uparrow f(s_i), \quad \text{by Lemma 15A.12(ii)}, \\ &= \bigsqcup_{i \in I} \text{Flt}(f)(\uparrow s_i) \quad \text{by Lemma 15A.15(ii)}.\end{aligned}$$

If I is empty, and U, U' are respectively the universal tops of $\mathcal{S}, \mathcal{S}'$, then

$$\begin{aligned}\text{Flt}(f)(\bigsqcup_{\emptyset} \uparrow s_i) &= \text{Flt}(f)(\uparrow U), \\ &= \uparrow f(U), \quad \text{by Lemma 15A.15(ii)}, \\ &= \uparrow U', \quad \text{since } f \text{ preserves tops}, \\ &= \bigsqcup_{\emptyset} \text{Flt}(f)(\uparrow s_i).\end{aligned}$$

So $\text{Flt}(f)$ satisfies (add). ■

It is possible to leave out conditions (cmp-pres) and (add), obtaining the category \mathbf{ALG} . Then one needs to consider *approximable maps* as morphisms in the category \mathbf{MSL}^U . See [Abramsky \[1991\]](#).

15A.17. DEFINITION. We define a map $\text{Cmp} : \mathbf{ALG}_a \rightarrow \mathbf{MSL}^U$, that will turn out to be a functor, as follows.

(i) On objects $\mathcal{D} \in \mathbf{ALG}_a$ one defines Cmp by

$$\text{Cmp}(\mathcal{D}) \triangleq (\mathcal{K}(\mathcal{D}), \leq).$$

(ii) On morphisms $f \in \mathbf{ALG}_a(\mathcal{D}, \mathcal{D}')$ one defines $\text{Cmp}(f)$ by

$$\text{Cmp}(f)(d) \triangleq f(d).$$

15A.18. LEMMA. *The map Cmp is a functor.*

PROOF. Let $f \in \mathbf{ALG}_a(\mathcal{D}, \mathcal{D}')$. Note that $\text{Cmp}(f) = f \upharpoonright \mathcal{K}(\mathcal{D}) : \mathcal{K}(\mathcal{D}) \rightarrow \mathcal{K}(\mathcal{D}')$, by (cmp-pres). By the fact that f is additive one has $f(\perp_{\mathcal{D}}) = \perp_{\mathcal{D}'}$, which is $f(U_{\mathcal{K}(\mathcal{D})}) = U_{\mathcal{K}(\mathcal{D}')}$, and

$$f(a \cap_{\mathcal{K}(\mathcal{D})} b) = f(a \sqcup_{\mathcal{D}} b) = f(a) \sqcup_{\mathcal{D}'} f(b) = f(a) \cap_{\mathcal{K}(\mathcal{D})} f(b).$$

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}^U and \mathbf{ALG}_a .

15A.19. PROPOSITION. (i) Let $\mathcal{S} \in \mathbf{MSL}^U$. Then $\sigma = \sigma_{\mathcal{S}} : \mathcal{S} \rightarrow \mathcal{K}(\mathcal{F}^{\mathcal{S}})$ defined by $\sigma_{\mathcal{S}}(s) = \uparrow s$ is an \mathbf{MSL}^U isomorphism.

$$\begin{array}{ccc} \mathcal{S} & \xrightarrow{\text{Flt}} & \mathcal{F}^{\mathcal{S}} \\ \downarrow \sigma = \uparrow & \nearrow \text{Cmp} & \\ \mathcal{K}(\mathcal{F}^{\mathcal{S}}) & & \end{array}$$

Therefore $\mathcal{S} \cong \mathcal{K}(\mathcal{F}^{\mathcal{S}})$.

(ii) Let $\mathcal{D} \in \mathbf{ALG}_a$. Then $\tau = \tau_{\mathcal{D}} : \mathcal{F}^{\mathcal{K}(\mathcal{D})} \rightarrow \mathcal{D}$ defined by $\tau(X) = \bigsqcup X$, where \bigsqcup is taken in \mathcal{D} , is an \mathbf{ALG}_a isomorphism with inverse $\tau^{-1} : \mathcal{D} \rightarrow \mathcal{F}^{\mathcal{K}(\mathcal{D})}$ defined by $\tau^{-1}(x) = \{c \in \mathcal{K}(\mathcal{D}) \mid c \sqsubseteq x\}$.

$$\begin{array}{ccc} \mathcal{K}(\mathcal{D}) & \xleftarrow{\text{Cmp}} & \mathcal{D} \\ \downarrow \text{Flt} & \nearrow \tau = \bigsqcup & \\ \mathcal{F}^{\mathcal{K}(\mathcal{D})} & & \end{array}$$

Therefore $\mathcal{D} \cong \mathcal{F}^{\mathcal{K}(\mathcal{D})}$.

PROOF. (i) By Proposition 13D.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 :

$$\begin{aligned} s \leq t &\Leftrightarrow \uparrow t \subseteq \uparrow s \\ &\Leftrightarrow \uparrow s \leq \uparrow t, \quad \text{by definition of } \leq \text{ on } \mathcal{K}(\mathcal{F}^{\mathcal{S}}). \end{aligned}$$

Also σ preserves \cap :

$$\begin{aligned} \sigma(s \cap t) &= \uparrow(s \cap t) \\ &= \uparrow s \sqcup \uparrow t && \text{in } \mathcal{F}^{\mathcal{S}}, \text{ by 13D.4(iii),} \\ &= \uparrow s \cap \uparrow t && \text{in } \mathcal{K}(\mathcal{F}^{\mathcal{S}}), \text{ by definition of } \leq \text{ on } \mathcal{K}(\mathcal{F}^{\mathcal{S}}), \\ &= \sigma(s) \cap \sigma(t). \end{aligned}$$

Then $\sigma(U_{\mathcal{S}}) = \uparrow U_{\mathcal{S}} = U_{\mathcal{K}(\mathcal{F}^{\mathcal{S}})}$, since $\{U_{\mathcal{S}}\}$ is the \leq -smallest filter; hence σ preserves tops.

Finally σ^{-1} preserves $\leq, \sqcap_{\mathcal{K}(\mathcal{F}^{\mathcal{S}})}$ and $U_{\mathcal{K}(\mathcal{F}^{\mathcal{S}})}$, as by Lemma 13D.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 $\tau \circ \tau^{-1} = \mathbf{1}_{\mathcal{D}}$ and $\tau^{-1} \circ \tau = \mathbf{1}_{\mathcal{F}^{\mathcal{K}(\mathcal{D})}}$:

$$\begin{aligned} \tau(\tau^{-1}(x)) &= \bigsqcup \{c \in \mathcal{K}(\mathcal{D}) \mid c \sqsubseteq x\} \\ &= x, && \text{since } \mathcal{D} \in \mathbf{ALG}_a. \\ \tau^{-1}(\tau(X)) &= \{c \mid c \sqsubseteq \bigsqcup X\} \\ &= \{c \mid c \in X\}, && \text{since one has} \\ c \sqsubseteq_{\mathcal{D}} \bigsqcup X &\Leftrightarrow \exists x \in X \ c \sqsubseteq_{\mathcal{D}} x, && \text{as } c \text{ is compact and } X \subseteq \mathcal{K}(\mathcal{D}) \subseteq \mathcal{D} \text{ is} \\ &&& \text{a filter w.r.t. } \leq, \text{ so directed w.r.t. } \sqsubseteq \\ &\Leftrightarrow \exists x \in X \ x \leq_{\mathcal{K}(\mathcal{D})} c \\ &\Leftrightarrow c \in X, && \text{as } X \text{ is a filter on } \mathcal{K}(\mathcal{D}). \end{aligned}$$

We still have to show that τ and τ^{-1} are morphisms. One easily sees that τ satisfies (cmp-pres). The map τ is also additive, i.e. $\sqcup \tau(\mathcal{X}) = \tau(\sqcup \mathcal{X})$ for arbitrary $\mathcal{X} \subseteq \mathcal{F}^{\mathcal{K}(\mathcal{D})}$. Indeed,

$$\begin{aligned}\sqcup \tau(\mathcal{X}) &= \bigsqcup_{X \in \mathcal{X}} (\sqcup X), && \text{by definition of } \tau, \\ &= \sqcup (\bigcup \mathcal{X}), && \text{by Proposition 15A.13(i),} \\ &= \sqcup (\uparrow (\bigcup \mathcal{X})), && \text{by Proposition 15A.13(ii),} \\ &= \tau(\uparrow (\bigcup \mathcal{X})), && \text{by definition of } \tau, \\ &= \tau(\sqcup \mathcal{X}), && \text{by Proposition 13D.4(i).}\end{aligned}$$

Now we have to prove that τ^{-1} satisfies (cmp-pres) and (add). As to (cmp-pres), assume that $b \in \mathcal{K}(\mathcal{D})$ and $\tau^{-1}(b) \sqsubseteq \sqcup X$, with X directed. Then $b \sqsubseteq \sqcup \tau(X)$, 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 $X \subseteq \mathcal{D}$. Then

$$\begin{aligned}\tau^{-1}(\sqcup X) &= \tau^{-1}(\sqcup \{\tau(\tau^{-1}(x)) \mid x \in X\}), \\ &= \tau^{-1}(\tau(\sqcup \{\tau^{-1}(x) \mid x \in X\})), && \text{since } \tau \text{ satisfies (add),} \\ &= \sqcup \{\tau^{-1}(x) \mid x \in X\}. \blacksquare\end{aligned}$$

15A.20. COROLLARY. *The categories \mathbf{MSL}^U and \mathbf{ALG}_a are equivalent; in fact the isomorphisms in Proposition 15A.19 form natural isomorphisms showing*

$$\mathbf{Cmp} \circ \mathbf{Flt} \cong \mathbf{Id}_{\mathbf{MSL}^U} \& \mathbf{Flt} \circ \mathbf{Cmp} \cong \mathbf{Id}_{\mathbf{ALG}_a}.$$

PROOF. First one has to show that in \mathbf{MSL}^U the following diagram commutes.

$$\begin{array}{ccc} \mathcal{S} & \xrightarrow{\uparrow} & \mathcal{K}(\mathcal{F}^{\mathcal{S}}) \\ f \downarrow & & \downarrow \mathbf{Cmp}(\mathbf{Flt}(f)) \\ \mathcal{S}' & \xrightarrow{\uparrow} & \mathcal{K}(\mathcal{F}^{\mathcal{S}'}) \end{array}$$

One must show $\mathbf{Cmp}(\mathbf{Flt}(f))(\uparrow s) = \uparrow(f(s))$. This follows from Lemma 15A.15(ii) and the definition of \mathbf{Cmp} .

Secondly one has to show that in \mathbf{ALG}_a the following diagram commutes.

$$\begin{array}{ccc} \mathcal{F}^{\mathcal{K}(\mathcal{D})} & \xrightarrow{\sqcup} & \mathcal{D} \\ \mathbf{Flt}(\mathbf{Cmp}(f)) \downarrow & & \downarrow f \\ \mathcal{F}^{\mathcal{K}(\mathcal{D}')}& \xrightarrow{\sqcup} & \mathcal{D}' \end{array}$$

Now for $X \in \mathcal{F}^{\mathcal{K}(\mathcal{D})}$ one has

$$\begin{aligned}\text{Flt}(\text{Cmp}(f))(X) &= \{d' \in \mathcal{K}(\mathcal{D}') \mid \exists d \in X. f(d) \leq d'\} \\ &= \{d' \in \mathcal{K}(\mathcal{D}') \mid \exists d \in X. d' \sqsubseteq f(d)\}.\end{aligned}$$

Hence, using also the continuity of f and the fact that X as subset of $\langle \mathcal{D}, \sqsubseteq \rangle$ is directed,

$$\bigsqcup(\text{Flt}(\text{Cmp}(f))(X)) = \bigsqcup_{d \in X} f(d) = f(\bigsqcup X),$$

and the diagram commutes. As before we have $\text{Flt} \circ \text{Cmp} \cong \text{Id}_{\mathbf{ALG}_a}$. ■

This result is a special case of Stone duality, cf. [Johnstone \[1986\]](#) (II, 3.3).

Equivalence between \mathbf{MSL}^{-U} and \mathbf{ALG}_a^s

We prove that $\mathbf{MSL}^{-U} \cong \mathbf{ALG}_a^s$. The proof uses the functors Flt and a small variant of Cmp , denoted by Cmp_s . The functor Flt is given in Definition 15A.14, where now \mathcal{F}^S is taken with $S \in \mathbf{MSL}^{-U}$. But now $\emptyset \in \mathcal{K}(\mathcal{F}^S)$ by Definition 13D.1 and hence, $S \not\in \mathcal{K}(\mathcal{F}^S)$. To obtain an isomorphism, the functor Cmp_s is defined by considering the set $\mathcal{K}^s(\mathcal{D})$ of compact elements of \mathcal{D} without \perp .

15A.21. DEFINITION. Let $\mathcal{D} \in \mathbf{ALG}_a^s$.

(i) The functor $\text{Cmp}_s : \mathbf{ALG}_a^s \rightarrow \mathbf{MSL}^{-U}$ is defined as follows

$$\text{Cmp}_s(\mathcal{D}) \triangleq (\mathcal{K}^s(\mathcal{D}), \leq).$$

For a morphism $f \in \mathbf{ALG}_a^s(\mathcal{D}, \mathcal{D}')$ define $\text{Cmp}_s(f) : \mathcal{K}^s(\mathcal{D}) \rightarrow \mathcal{K}^s(\mathcal{D}')$ by

$$\text{Cmp}_s(f)(d) \triangleq f(d).$$

15A.22. PROPOSITION. (i) Let $S \in \mathbf{MSL}^{-U}$. Then $\sigma : S \rightarrow \mathcal{K}^s(\mathcal{F}^S)$ defined by $\sigma(s) \triangleq \uparrow s$ is an \mathbf{MSL}^{-U} isomorphism.

$$\begin{array}{ccc} S & \xrightarrow{\text{Flt}} & \mathcal{F}^S \\ \uparrow \downarrow & \swarrow \text{Cmp}_s & \\ \mathcal{K}^s(\mathcal{F}^S) & & \end{array}$$

Therefore $S \cong \mathcal{K}^s(\mathcal{F}^S)$.

(ii) Let $\mathcal{D} \in \mathbf{ALG}_a^s$. Then $\tau : \mathcal{F}^{\mathcal{K}^s(\mathcal{D})} \rightarrow \mathcal{D}$ defined by $\tau(X) \triangleq \bigsqcup X$, is an \mathbf{ALG}_a^s isomorphism with inverse $\tau^{-1} : \mathcal{D} \rightarrow \mathcal{F}^{\mathcal{K}^s(\mathcal{D})}$ defined by

$$\tau^{-1}(d) \triangleq \{c \in \mathcal{K}^s(\mathcal{D}) \mid c \sqsubseteq d\}.$$

In diagram,

$$\begin{array}{ccc} \mathcal{K}^s(\mathcal{D}) & \xleftarrow{\text{Cmp}_s} & \mathcal{D} \\ & \searrow \text{Flt} & \nearrow \bigsqcup \\ & \mathcal{F}^{\mathcal{K}^s(\mathcal{D})} & \end{array}$$

Therefore $\mathcal{D} \cong \mathcal{F}^{\mathcal{K}^s(\mathcal{D})}$.

PROOF. Similar to the proof of Proposition 15A.19. ■

15A.23. COROLLARY. The categories \mathbf{MSL}^{-U} and \mathbf{ALG}_a^s are equivalent.

15A.24. REMARK. The map $\rho : \mathcal{F}^{\mathcal{K}^s(\mathcal{D})} \rightarrow \mathcal{F}^{\mathcal{K}(\mathcal{D})}$, given by $\rho(X) \triangleq X \cup \{\perp_{\mathcal{D}}\}$ is an isomorphism in the category \mathbf{ALG}_a^s , hence also in \mathbf{ALG}_a .

15B. Natural type structures and lambda structures

In this section we prove for the natural type and lambda structures that

$$\begin{aligned}\mathcal{S} &\simeq \mathcal{K}(\mathcal{F}^{\mathcal{S}}), \\ \mathcal{D} &\simeq \mathcal{F}^{\mathcal{K}(\mathcal{D})},\end{aligned}$$

such that there is a congruence between the categories $\mathbf{NTS}^U \cong \mathbf{NLS}$. The results of this Section will be generalized using zip structures in Sections 15C and 15D. 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.

15B.1. DEFINITION. Let $\mathcal{D}, \mathcal{D}' \in \mathbf{ALG}$. A *Galois connection* between \mathcal{D} and \mathcal{D}' , notation $\langle m, n \rangle : \mathcal{D} \rightarrow \mathcal{D}'$, is a pair of continuous functions $\langle m, n \rangle$ with $m : \mathcal{D} \rightarrow \mathcal{D}'$, $n : \mathcal{D}' \rightarrow \mathcal{D}$ such that

$$\begin{aligned}(\text{Galois-1}) \quad n \circ m &\sqsupseteq \text{Id}_{\mathcal{D}}, \\ (\text{Galois-2}) \quad m \circ n &\sqsubseteq \text{Id}_{\mathcal{D}'};\end{aligned}$$

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.

$$(\text{Galois}) \quad \forall x \in \mathcal{D}, x' \in \mathcal{D}'. m(x) \sqsubseteq x' \Leftrightarrow x \sqsubseteq n(x').$$

See Exercise 15E.7.

Each adjoint in a Galois connection determines the other, see Exercise 15E.8. For this reason often one writes m^R for n , and n^L for m . From now on \underline{m} is short for $\langle m, m^R \rangle$.

The next lemma provides some properties of Galois connections.

15B.2. LEMMA. Let $\underline{m} : \mathcal{D} \rightarrow \mathcal{D}'$ be a Galois connection. Then

(i) m is *additive*:

$$\forall X \subseteq \mathcal{D}, m(\bigsqcup X) = \bigsqcup m(X).$$

In particular m is strict, $m(\perp) = \perp$, as $\perp = \bigsqcup \emptyset$.

(ii) $m^R(\top) = \top$.

(iii) $d \in \mathcal{K}(\mathcal{D}) \Rightarrow m(d) \in \mathcal{K}(\mathcal{D}')$.

(iv) Let $d, e \in \mathcal{K}(\mathcal{D})$. Then

$$m \circ (d \mapsto e) \circ m^R = (m(d) \mapsto m(e)).$$

PROOF. (i) As m is monotone we have $\bigsqcup m(X) \sqsubseteq m(\bigsqcup X)$. On the other hand

$$\begin{aligned}m(\bigsqcup X) &\sqsubseteq m(\bigsqcup m^R \circ m(X)), \quad \text{by (Galois-1),} \\ &\sqsubseteq m(m^R(\bigsqcup m(X))), \quad \text{since } m^R \text{ is monotone,} \\ &\sqsubseteq \bigsqcup m(X), \quad \text{by (Galois-2).}\end{aligned}$$

(ii) By (Galois) applied to $x' = \top$.

(iii) Let $d \in \mathcal{K}(\mathcal{D})$ and let $m(d) \sqsubseteq \bigsqcup Z$, where $Z \subseteq \mathcal{D}'$ is any directed set. Then, by (Galois), $d \sqsubseteq m^R(\bigsqcup Z) = \bigsqcup m^R(Z)$. Since d is compact, there exists $z \in Z$ such that $d \sqsubseteq m^R(z)$, hence, by (Galois), $m(d) \sqsubseteq z$. This proves that $m(d)$ is compact.

(iv) Let $y \in \mathcal{D}'$. We have

$$(\mathbf{m} \circ (d \mapsto e) \circ \mathbf{m}^R)(y) = \begin{cases} \mathbf{m}(e), & \text{if } \mathbf{m}^R(y) \sqsupseteq d; \\ \mathbf{m}(\perp), & \text{otherwise.} \end{cases}$$

Note that by (Galois), $d \sqsubseteq \mathbf{m}^R(y)$ is equivalent to $\mathbf{m}(d) \sqsubseteq y$. Moreover, as previously noted, $\mathbf{m}(\perp) = \perp$. So we have

$$(\mathbf{m} \circ (d \mapsto e) \circ \mathbf{m}^R)(y) = \begin{cases} \mathbf{m}(e), & \text{if } \mathbf{m}(d) \sqsubseteq y; \\ \perp, & \text{otherwise.} \end{cases}$$

The RHS above is the definition of the step function $(\mathbf{m}(d) \mapsto \mathbf{m}(e))$. ■

15B.3. DEFINITION. (i) A triple $\mathcal{D} = \langle \mathcal{D}, F, G \rangle$ is called a *lambda structure*, notation LS, if $\mathcal{D} \in \mathbf{ALG}$ and $F : \mathcal{D} \rightarrow [\mathcal{D} \rightarrow \mathcal{D}]$ and $G : [\mathcal{D} \rightarrow \mathcal{D}] \rightarrow \mathcal{D}$ are continuous.

(ii) The lambda structure $\mathcal{D} = \langle \mathcal{D}, F, G \rangle$ is *natural* if $\langle G, F \rangle$ is a Galois connection, with $F = G^R$, i.e. $F \circ G \sqsupseteq \mathbf{Id}_{\mathcal{D}}$ and $G \circ F \sqsubseteq \mathbf{Id}_{\mathcal{D}}$.

Note that we are using the notation $\langle \mathcal{D}, F, G \rangle$ 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 $\langle \mathcal{D}, F, G \rangle$, then (Galois) implies

$$(\text{func-Galois}) \quad \forall d, e \in \mathcal{K}(\mathcal{D}), x \in \mathcal{D}. \ e \sqsubseteq F(x)(d) \Leftrightarrow G(d \mapsto e) \sqsubseteq x.$$

The next lemma shows how to build Galois connections in \mathbf{ALG} from morphisms between \mathbf{NTS}^U 's.

15B.4. LEMMA. Let $f \in \mathbf{NTS}^U(\mathcal{S}, \mathcal{S}')$. Let \bar{f} be short for $\mathbf{Flt}(f)$. Define $\bar{f}^R : \mathbf{ALG}(\mathcal{F}^{\mathcal{S}'}, \mathcal{F}^{\mathcal{S}})$ by

$$\bar{f}^R(d') \triangleq \{s \mid f(s) \in d'\}.$$

Then $\langle \bar{f}, \bar{f}^R \rangle : \mathcal{F}^{\mathcal{S}} \rightarrow \mathcal{F}^{\mathcal{S}'}$ is a Galois connection (so the name \bar{f}^R is appropriate).

PROOF. We leave as an exercise to prove that \bar{f} is well defined and continuous. Note that

$$(3) \quad [\forall s \in \mathcal{S}. \ s \in d \Rightarrow f(s) \in d'] \Leftrightarrow [\forall s' \in \mathcal{S}'. \ (\exists s \in d. f(s) \leq s') \Rightarrow s' \in d']$$

(\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 $\langle \bar{f}, \bar{f}^R \rangle$.

$$\begin{aligned} \bar{f}(x) \subseteq x' &\Leftrightarrow \{s' \mid \exists s \in x. f(s) \leq s'\} \subseteq x' \\ &\Leftrightarrow \forall s' \in \mathcal{S}'. (\exists s \in x. f(s) \leq s') \Rightarrow s' \in x' \\ &\Leftrightarrow \forall s \in \mathcal{S}. s \in x \Rightarrow f(s) \in x', && \text{by (3),} \\ &\Leftrightarrow x \subseteq \{s \mid f(s) \in x'\} \\ &\Leftrightarrow x \subseteq \bar{f}^R(x'). \blacksquare \end{aligned}$$

From now on \bar{f} denotes for $f \in \mathbf{NTS}^U(\mathcal{S}, \mathcal{S}')$ the Galois connection $\langle \bar{f}, \bar{f}^R \rangle$.

Next definition is necessary for introducing morphisms between lambda structures. We will explain this choice in Section 15D.

15B.5. DEFINITION. Let $\mathcal{D} = \langle \mathcal{D}, F, G \rangle$, $\mathcal{D}' = \langle \mathcal{D}', F, G \rangle$ be lambda structures. A *lambda Galois connection* is a Galois connection $\underline{\mathbf{m}} = \langle \mathbf{m}, \mathbf{m}^R \rangle : \mathcal{D} \rightarrow \mathcal{D}'$ such that

- (lambda-gc1) $\forall f \in [\mathcal{D} \rightarrow \mathcal{D}]. \mathbf{m}(G(f)) \sqsubseteq G'(\mathbf{m} \circ f \circ \mathbf{m}^R);$
- (lambda-gc2) $\forall x' \in \mathcal{D}', x \in \mathcal{D}. F(\mathbf{m}^R(x'))(x) \sqsubseteq \mathbf{m}^R(F'(x'))(\mathbf{m}(x)).$

If we write $f^{\mathbf{m}} \triangleq \mathbf{m} \circ f \circ \mathbf{m}^R$, then we can reformulate these conditions as

- (lambda-gc1) $\forall f \in [\mathcal{D} \rightarrow \mathcal{D}]. \mathbf{m}(G(f)) \sqsubseteq G'(f^{\mathbf{m}});$
- (lambda-gc2) $\forall x' \in \mathcal{D}', x \in \mathcal{D}. \mathbf{m}^R(x') \cdot x \sqsubseteq \mathbf{m}^R(x' \cdot (\mathbf{m}(x))).$

See also Lemma 16A.12.

15B.6. DEFINITION. (i) The category **LS** consists of lambda structures as objects and lambda Galois connections as morphisms. The composition between morphisms $\langle \mathbf{m}, \mathbf{m}^R \rangle : \mathcal{D} \rightarrow \mathcal{D}'$ and $\langle \mathbf{n}, \mathbf{n}^R \rangle : \mathcal{D}' \rightarrow \mathcal{D}''$ is given by $\langle \mathbf{n} \circ \mathbf{m}, \mathbf{m}^R \circ \mathbf{n}^R \rangle$.

(ii) The category of *natural lambda structures*, notation **NLS**, is the full subcategory of **LS** which has as objects natural lambda structures.

For $\mathcal{S} \in \text{TS}^U$ the maps $F^{\mathcal{S}} : \mathcal{F}^{\mathcal{S}} \rightarrow [\mathcal{F}^{\mathcal{S}} \rightarrow \mathcal{F}^{\mathcal{S}}]$ and $G^{\mathcal{S}} : [\mathcal{F}^{\mathcal{S}} \rightarrow \mathcal{F}^{\mathcal{S}}] \rightarrow \mathcal{F}^{\mathcal{S}}$ are two continuous functions, defined in Definition 13D.5 as follows.

$$\begin{aligned} \forall x \in \mathcal{F}^{\mathcal{S}}. F^{\mathcal{S}}(x) &\triangleq \lambda y \in \mathcal{S}. \uparrow \{t \mid \exists s \in y. s \rightarrow t \in x\}; \\ \forall f \in [\mathcal{F}^{\mathcal{S}} \rightarrow \mathcal{F}^{\mathcal{S}}]. G^{\mathcal{S}}(f) &\triangleq \uparrow \{s \rightarrow t \mid t \in f(\uparrow s)\}. \end{aligned}$$

Also remember that $x \cdot y \triangleq F^{\mathcal{S}}(x)(y)$, for any $x, y \in \mathcal{F}^{\mathcal{S}}$.

15B.7. LEMMA. Let $\mathcal{S} \in \text{NTS}^U$, $s, t \in \mathcal{S}$, $x, y \in \mathcal{F}^{\mathcal{S}}$, and $f : \mathcal{F}^{\mathcal{S}} \rightarrow \mathcal{F}^{\mathcal{S}}$. Then

- (i) $x \cdot y = \{t \mid \exists s \in y. s \rightarrow t \in x\}.$
- (ii) $t \in x \cdot \uparrow s \Leftrightarrow s \rightarrow t \in x.$
- (iii) $t \in f(\uparrow s) \Rightarrow (s \rightarrow t) \in G^{\mathcal{S}}(f).$
- (iv) $G^{\mathcal{S}}(\uparrow s \mapsto \uparrow t) = \uparrow (s \rightarrow t).$

PROOF. (i) Let $\{x \cdot y\} = \{t \mid \exists s \in y. s \rightarrow t \in x\}$. 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 $U \in \{x \cdot y\}$ by ($U \rightarrow$).

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 $(s \rightarrow t) \in x$. But $(s \rightarrow t) \leq (s \rightarrow t')$ by (\rightarrow), 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 $(s \rightarrow t), (s' \rightarrow t') \in x$ for some $s, s' \in y$. Then $(s \rightarrow t) \cap (s' \rightarrow t') \in x$ and $s \cap s' \in y$, as x, y are filters. By Proposition 13A.21(ii) one has $(s \rightarrow t) \cap (s' \rightarrow t') \leq (s \cap s') \rightarrow (t \cap t')$, 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 $(s' \rightarrow t) \in x$. By (\rightarrow), this is equivalent to $(s \rightarrow t) \in x$.

(iii) Easy.

(iv) We have

$$\begin{aligned} G^{\mathcal{S}}(\uparrow s \mapsto \uparrow t) &= \uparrow \{s' \rightarrow t' \mid t' \in (\uparrow s \mapsto \uparrow t)(\uparrow s')\} \\ &= \uparrow \{s' \rightarrow t' \mid [\uparrow s' \supseteq \uparrow s \& t' \in \uparrow t] \text{ or } t' = U\} \\ &= \uparrow (s \rightarrow t). \end{aligned}$$

As to the last equality, \supseteq holds trivially, and \subseteq by (\rightarrow), ($U \rightarrow$). ■

15B.8. LEMMA. Let \mathcal{S} be in NTS^U . Write $\text{Flt}_{\text{NTS}^U}(\mathcal{S}) = \langle \mathcal{F}^{\mathcal{S}}, F^{\mathcal{S}}, G^{\mathcal{S}} \rangle$. Then

- (i) $\text{Flt}_{\text{NTS}^U}(\mathcal{S}) \in \text{LS}$.
- (ii) $\text{Flt}_{\text{NTS}^U}(\mathcal{S}) \in \text{NLS}$.

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 $f \in [\mathcal{S} \rightarrow \mathcal{S}]$, $s \in \mathcal{S}$. We have

$$\begin{aligned} F^{\mathcal{S}}(G^{\mathcal{S}}(f))(\uparrow s) &= \{t \mid s \rightarrow t \in G^{\mathcal{S}}(f)\}, && \text{by Lemma 15B.7(ii),} \\ &\supseteq \{t \mid t \in f(\uparrow s)\}, && \text{by Lemma 15B.7(iii),} \\ &= f(\uparrow s). \end{aligned}$$

On the other hand, let $x \in \mathcal{F}^{\mathcal{S}}$. We have

$$\begin{aligned} G^{\mathcal{S}}(F^{\mathcal{S}}(x)) &= \uparrow \{s \rightarrow t \mid t \in x \cdot \uparrow s\} \\ &= \uparrow \{s \rightarrow t \mid s \rightarrow t \in x\}, && \text{by Lemma 15B.7(ii),} \\ &\subseteq x. \blacksquare \end{aligned}$$

15B.9. THEOREM. Define the action of $\text{Flt}_{\text{NTS}^U}$ on morphisms $f \in \text{NTS}^U(\mathcal{S}, \mathcal{S}')$ by

$$\text{Flt}_{\text{NTS}^U}(f) \triangleq \langle \text{Flt}(f), \text{Flt}(f)^R \rangle.$$

Then $\text{Flt}_{\text{NTS}^U} : \text{NTS}^U \rightarrow \text{NLS}$ is a functor.

PROOF. The proof will descend from the results of Sections 15C and 15D (see in particular Proposition 15D.27).

15B.10. DEFINITION. (i) Given a natural lambda structure \mathcal{D} , we define

$$\text{Flt}_{\text{NLS}}(\mathcal{D}) \triangleq \langle \mathcal{K}(\mathcal{D}), \leq, \cap, \rightarrow_{\mathcal{D}}, U \rangle$$

with $a \leq b \triangleq b \sqsubseteq_{\mathcal{K}(\mathcal{D})} a$, $a \cap b \triangleq a \sqcup_{\mathcal{D}} b$, $U \triangleq \perp_{\mathcal{D}}$, and $a \rightarrow_{\mathcal{K}(\mathcal{D})} b \triangleq G(a \mapsto b)$.

(ii) Given $\underline{m} \in \text{NLS}(\mathcal{D}, \mathcal{D}')$, we define

$$\text{Flt}_{\text{NLS}}(\underline{m}) \triangleq \underline{m} \upharpoonright \mathcal{K}(\mathcal{D}) : \mathcal{K}(\mathcal{D}) \rightarrow \mathcal{K}(\mathcal{D}').$$

15B.11. LEMMA. Flt_{NLS} is a functor from NLS to NTS^U .

PROOF. The proof will descend from the result of Sections 15C and 15D (see in particular Theorem 15B.14).

Now we will prove that NTS^U and NLS are equivalent categories.

15B.12. PROPOSITION. Let $\mathcal{S} \in \text{NTS}^U$. Define $\sigma_{\mathcal{S}} : S \rightarrow \mathcal{K}(\mathcal{F}^{\mathcal{S}})$ by

$$\sigma_{\mathcal{S}}(s) \triangleq \uparrow s.$$

Then $\sigma_{\mathcal{S}}$ is an isomorphism in NTS^U . Hence $\mathcal{S} \cong \mathcal{K}(\mathcal{F}^{\mathcal{S}})$.

PROOF. We write simply σ for $\sigma_{\mathcal{S}}$. We know from Proposition 15A.19 that σ is an isomorphism of meet-semilattices, so it preserves intersections and top elements, and so does its inverse σ^{-1} . We will prove that $\sigma(s \rightarrow t) = \sigma(s) \rightarrow_{\mathcal{K}(\mathcal{F}^{\mathcal{S}})} \sigma(t)$, for any $s, t \in \mathcal{S}$. We have

$$\begin{aligned} \sigma(s) \rightarrow_{\mathcal{K}(\mathcal{F}^{\mathcal{S}})} \sigma(t) &= \uparrow s \rightarrow_{\mathcal{K}(\mathcal{F}^{\mathcal{S}})} \uparrow t \\ &= G^{\mathcal{S}}(\uparrow s \mapsto \uparrow t), && \text{by Definition 15B.10,} \\ &= \uparrow (s \rightarrow t), && \text{by Lemma 15B.8(i),} \\ &= \sigma(s \rightarrow t). \end{aligned}$$

Similarly, $\sigma^{-1}(\uparrow s \rightarrow_{\mathcal{K}(\mathcal{F}^{\mathcal{S}})} \uparrow t) = s \rightarrow t$. ■

15B.13. PROPOSITION. Let $\mathcal{D} \in \mathbf{NLS}$. Define $\tau_{\mathcal{D}} : \mathcal{F}^{\mathcal{K}(\mathcal{D})} \rightarrow \mathcal{D}$ by

$$\tau_{\mathcal{D}}(x) = \bigsqcup x.$$

Then $\tau_{\mathcal{D}}$ is an isomorphism in \mathbf{NLS} . Hence $\mathcal{D} \cong \mathcal{F}^{\mathcal{K}(\mathcal{D})}$.

PROOF. Write $\tau = \tau_{\mathcal{D}}$. By Proposition 15A.19 τ is an isomorphism of lattices. The pair $\langle \tau, \tau^{-1} \rangle$ is a Galois connection. We show that

$$\langle \tau, \tau^{-1} \rangle : \langle \mathcal{F}^{\mathcal{K}(\mathcal{D})}, F^{\mathcal{K}(\mathcal{D})}, G^{\mathcal{K}(\mathcal{D})} \rangle \rightarrow \langle \mathcal{D}, F, G \rangle$$

is lambda. Let $f : \mathcal{F}^{\mathcal{K}(\mathcal{D})} \rightarrow \mathcal{F}^{\mathcal{K}(\mathcal{D})}$ be a continuous function. We have:

$$\begin{aligned} \tau(G^{\mathcal{K}(\mathcal{D})}(f)) &= \tau(\uparrow \{a \rightarrow_{\mathcal{K}(\mathcal{D})} b \mid b \in f(\uparrow a)\}) \\ &= \bigsqcup \{a \rightarrow_{\mathcal{K}(\mathcal{D})} b \mid b \in f(\uparrow a)\}, \\ &\quad \text{since } \bigsqcup \uparrow x = \bigsqcup x, \text{ for any } x \subseteq \mathcal{K}(\mathcal{D}), \\ &= \bigsqcup \{G(a \mapsto b \mid b \in f(\uparrow a))\}, \text{ by definition of } \rightarrow_{\mathcal{K}(\mathcal{D})}, \\ &= \bigsqcup \{G(a \mapsto b) \mid b \sqsubseteq \bigsqcup(f(\uparrow a))\} \\ &= \bigsqcup \{G(a \mapsto b) \mid b \sqsubseteq \tau \circ f \circ \tau^{-1}(a)\} \\ &= G(\bigsqcup \{a \mapsto b \mid b \sqsubseteq \tau \circ f \circ \tau^{-1}(a)\}), \\ &\quad \text{since } G \text{ is additive by Lemma 15B.2(i),} \\ &= G(\tau \circ f \circ \tau^{-1}). \end{aligned}$$

The above shows in particular that $\langle \tau, \tau^{-1} \rangle$ satisfies (lambda-gc1) in Definition 15B.5. We now prove (lambda-gc2). Of course it is sufficient to reason on compact elements, so let $a, b \in \mathcal{K}(\mathcal{D})$. Taking, into (lambda-gc2), $x' = a$ and $x = \uparrow b$, we have to prove $F^{\mathcal{K}(\mathcal{D})}(\tau^{-1}(a))(\uparrow b) \sqsubseteq \tau^{-1}(F(a)(\tau(\uparrow b)))$, that is

$$F^{\mathcal{K}(\mathcal{D})}(\uparrow a)(\uparrow b) \sqsubseteq \tau^{-1}(F(a)(b)).$$

We have

$$\begin{aligned} F^{\mathcal{K}(\mathcal{D})}(\uparrow a)(\uparrow b) &= \{t \in \mathcal{K}(\mathcal{D}) \mid b \rightarrow_{\mathcal{K}(\mathcal{F}\mathcal{S})} t \in \uparrow a\}, \text{ by Lemma 15B.7(ii),} \\ &= \{t \in \mathcal{K}(\mathcal{D}) \mid G(b \mapsto t) \sqsubseteq a\}, \text{ by Definition 15B.10,} \\ &= \{t \in \mathcal{K}(\mathcal{D}) \mid b \mapsto t \sqsubseteq F(a)\}, \text{ since } \langle F, G \rangle \text{ is a Galois} \\ &\quad \text{connection,} \\ &= \{t \in \mathcal{K}(\mathcal{D}) \mid t \sqsubseteq F(a)(b)\} \\ &= \tau^{-1}(F(a)(b)). \end{aligned}$$

As a consequence of the above, we have that $\langle \tau, \tau^{-1} \rangle$ satisfies (lambda-gc2). Similarly, also $\langle \tau^{-1}, \tau \rangle$ is a lambda Galois connection from \mathcal{D} to $\mathcal{F}^{\mathcal{K}(\mathcal{D})}$, and it is of course the inverse of $\langle \tau, \tau^{-1} \rangle$.

15B.14. THEOREM. The categories \mathbf{NTS}^U and \mathbf{NLS} are equivalent.

PROOF. This follows from Propositions 15B.12 and 15B.13 almost in the same way as Corollary 15A.20 follows from Proposition 15A.19. There is one extra case. If $\langle m, m^R \rangle :$

$\mathcal{D} \rightarrow \mathcal{D}'$, then we must show the commutativity of the following diagram

$$\begin{array}{ccc} & \sqcup & \\ \mathcal{D} & \xleftarrow{\quad} & \mathcal{F}^{\mathcal{K}(\mathcal{D})} \\ \uparrow m^R & & \uparrow m \upharpoonright \mathcal{K}(\mathcal{D})^R \\ \mathcal{D}' & \xleftarrow{\quad} & \mathcal{F}^{\mathcal{K}(\mathcal{D}')} \end{array}$$

This is done in Exercise 15E.11. ■

15C. Type and zip structures

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 $\langle \mathcal{D}, Z \rangle$, where \mathcal{D} is an object in **ALG** and $Z : \mathcal{K}(\mathcal{D}) \times \mathcal{K}(\mathcal{D}) \rightarrow \mathcal{K}(\mathcal{D})$ 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.

15C.1. DEFINITION. (i) A *zip structure* is a pair $\langle \mathcal{D}, Z \rangle$ with $\mathcal{D} \in \mathbf{ALG}$ and

$$Z : \mathcal{K}(\mathcal{D}) \times \mathcal{K}(\mathcal{D}) \rightarrow \mathcal{K}(\mathcal{D})$$

an arbitrary map.

(ii) The category **ZS** has zip structures as objects and maps $f : \mathcal{D} \rightarrow \mathcal{D}'$ such that (a, b, c, \dots) ranging over $\mathcal{K}(\mathcal{D})$)

$$\begin{aligned} (\text{cmp-pres}) \quad & \forall a. f(a) \in \mathcal{K}(\mathcal{D}'); \\ (\text{add}) \quad & \forall X \subseteq \mathcal{D}. f(\bigsqcup X) = \bigsqcup f(X); \\ (\text{Z-mon}) \quad & \forall a, b. f(Z(a, b)) \sqsubseteq Z'(f(a), f(b)) \end{aligned}$$

as morphisms $\mathbf{ZS}(\langle \mathcal{D}, Z \rangle, \langle \mathcal{D}', Z' \rangle)$. The second requirement implies that a morphism f is continuous (only required to preserve sups for directed sets X) and strict, i.e. $f(\perp_{\mathcal{D}}) = \perp_{\mathcal{D}'}$.

We now specialize this general framework to special zip structures.

15C.2. DEFINITION. Let $\langle \mathcal{D}, Z \rangle$ be a zip structure.

- (i) Then $\langle \mathcal{D}, Z \rangle$ is a *lazy zip structure* if the following holds.
 - (1) (Z-contr) $a \sqsubseteq a' \& b' \sqsubseteq b \Rightarrow Z(a', b') \sqsubseteq Z(a, b);$
 - (2) (Z-add) $Z(a, b_1 \sqcup b_2) = Z(a, b_1) \sqcup Z(a, b_2);$
 - (3) (Z-lazy) $Z(\perp_{\mathcal{D}}, \perp_{\mathcal{D}}) \sqsubseteq Z(a, b).$

(ii) **LZS** is the full subcategory of **ZS** consisting of lazy zip structures.

15C.3. DEFINITION. Let $\langle \mathcal{D}, Z \rangle \in \mathbf{ZS}$.

- (i) Then $\langle \mathcal{D}, Z \rangle$ is a *natural zip structure* if $\langle \mathcal{D}, Z \rangle \in \mathbf{LZS}$ and moreover

$$(Z\text{-bot}) \quad Z(\perp_{\mathcal{D}}, \perp_{\mathcal{D}}) = \perp_{\mathcal{D}}.$$

(ii) **NZS** is the full subcategory of **LZS** consisting of natural zip structures.

15C.4. REMARK. Since condition $(Z\text{-bot})$ is stronger than $(Z\text{-lazy})$, \mathcal{D} is natural if it satisfies $(Z\text{-contr})$, $(Z\text{-add})$ and $(Z\text{-bot})$. In fact, $(Z\text{-bot})$ corresponds to $(U\rightarrow)$, and $(Z\text{-lazy})$ to the weaker (U_{lazy}) .

Strict zip structures

15C.5. DEFINITION. Let $\mathcal{D} \in \mathbf{ALG}$. Remember that $\mathcal{K}^s(\mathcal{D}) = \mathcal{K}(\mathcal{D}) - \{\perp_{\mathcal{D}}\}$.

- (i) A *strict zip structure* is of the form $\langle \mathcal{D}, Z \rangle$, with

$$Z : (\mathcal{K}^s(\mathcal{D}) \times \mathcal{K}^s(\mathcal{D})) \rightarrow \mathcal{K}^s(\mathcal{D}).$$

- (ii) If we write $Z(a, b)$, then it is always understood that $(a, b) \in \text{dom}(Z)$.
- (iii) The category \mathbf{ZS}^s consists of strict zip structures as objects and as morphisms maps f satisfying

$$\begin{aligned} (\text{cmp-pres}^s) \quad & \forall a \in \mathcal{K}^s(\mathcal{D}). f(a) \in \mathcal{K}^s(\mathcal{D}'); \\ (\text{add}) \quad & \forall X \subseteq \mathcal{D}. f(\bigsqcup X) = \bigsqcup f(X); \\ (Z\text{-mon}) \quad & \forall a, b. f(Z(a, b)) \sqsubseteq Z'(f(a), f(b)). \end{aligned}$$

15C.6. DEFINITION. (i) Let $\langle \mathcal{D}, Z \rangle \in \mathbf{ZS}^s$. Then $\langle \mathcal{D}, Z \rangle$ is called a *proper strict zip structure*, if it satisfies the following conditions.

$$\begin{aligned} (\text{Z-contr}) \quad & a \sqsubseteq a' \& b' \sqsubseteq b \Rightarrow Z(a', b') \sqsubseteq Z(a, b); \\ (\text{Z-add}) \quad & Z(a, b_1 \sqcup b_2) = Z(a, b_1) \sqcup Z(a, b_2). \end{aligned}$$

- (ii) \mathbf{PZS}^s is the full subcategory of \mathbf{ZS}^s consisting of proper strict zip structures.

15C.7. REMARK. In Section 13C we introduced the names $\text{MSL}^U, \text{MSL}^{-U}, \text{TS}^U, \text{TS}^{-U}$ for collections of meet semi-lattices and type structures. The superscript U in these names denotes that the structures in the relevant collections have a top element.

In Definitions 15A.9, 15C.1 and 15C.5, we introduced $\mathbf{ALG}_a, \mathbf{ALG}_a^s, \mathbf{ZS}, \mathbf{ZS}^s$. The superscript s , to be read as ‘strict’, concerns the top element of $\mathcal{K}(\mathcal{D})_{\leq}$ (see Definition 15A.1(ii)) and suggests that we are not interested in it.

Equivalences between type and zip structures

Now we will extend the equivalences of Section 15A to the various categories of type and zip structures. We will show the following equivalences of categories.

$$\begin{aligned} \mathbf{TS}^U &\cong \mathbf{ZS}; \\ \mathbf{LTS}^U &\cong \mathbf{LZS}; \\ \mathbf{NTS}^U &\cong \mathbf{NZS}; \\ \mathbf{TS}^{-U} &\cong \mathbf{ZS}^s; \\ \mathbf{PTS}^{-U} &\cong \mathbf{PZS}^s. \end{aligned}$$

Note that under this correspondence there is a sort of adjunction in the superscripts of the names due to the fact that in the left-hand 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 $\mathcal{K}(\mathcal{D})_{\leq}$). In particular, note that \mathbf{TS} does not correspond with \mathbf{ZS} .

Since the proofs are standard, we will give the full details only for the equivalence between \mathbf{TS}^U and \mathbf{ZS} , whilst the other cases are done in Exercises 15E.15 and 15E.16.

Equivalence between \mathbf{TS}^U and \mathbf{ZS}

First we see how the functors Flt and Cmp between \mathbf{MSL}^U and \mathbf{ALG}_a induce new functors between the richer categories \mathbf{TS}^U and \mathbf{ZS} .

15C.8. DEFINITION. (i) For $\mathcal{S} \in \mathbf{TS}^U$, define $\mathcal{Q}(\mathcal{S}) \in \mathbf{ZS}$ by

$$\mathcal{Q}(\mathcal{S}) \triangleq \langle \mathcal{F}^{\mathcal{S}}, Z^{\mathcal{S}} \rangle,$$

with $Z^{\mathcal{S}} : \mathcal{K}(\mathcal{F}^{\mathcal{S}}) \times \mathcal{K}(\mathcal{F}^{\mathcal{S}}) \rightarrow \mathcal{K}(\mathcal{F}^{\mathcal{S}})$ defined by

$$Z^{\mathcal{S}}(\uparrow s, \uparrow t) \triangleq \uparrow(s \rightarrow t).$$

(ii) For $\langle \mathcal{D}, Z \rangle \in \mathbf{ZS}$, define $\mathcal{M}(\langle \mathcal{D}, Z \rangle) \in \mathbf{TS}^U$ by

$$\mathcal{M}(\langle \mathcal{D}, Z \rangle) \triangleq \langle \mathcal{K}(\mathcal{D}), \leq, \cap, \rightarrow_Z, U \rangle,$$

with $a \leq b \iff b \sqsubseteq_{\mathcal{D}} a$, $a \cap b \triangleq a \sqcup_{\mathcal{D}} b$, and $U \triangleq \perp_{\mathcal{D}}$, as in Definition 15A.17 and

$$a \rightarrow_Z b \triangleq Z(a, b).$$

(iii) The actions of the maps $\mathcal{M} : \mathbf{ZS} \rightarrow \mathbf{TS}^U$ and $\mathcal{Q} : \mathbf{TS}^U \rightarrow \mathbf{ZS}$ on morphisms are defined as follows. Given $f \in \mathbf{TS}^U(\mathcal{S}, \mathcal{S}')$, $X \in \mathcal{F}^{\mathcal{S}}$ and $g \in \mathbf{ZS}(\langle \mathcal{D}, Z \rangle, \langle \mathcal{D}', Z' \rangle)$ define

$$\begin{aligned} \mathcal{Q}(f) &\triangleq \lambda X. \{t \mid \exists s \in X. f(s) \leq t\}; \\ \mathcal{M}(g) &\triangleq g \upharpoonright \mathcal{K}(\mathcal{D}). \end{aligned}$$

15C.9. LEMMA. If $\mathcal{S} \in \mathbf{TS}^U$ and $Z^{\mathcal{S}}$ are defined as above in Definition 15C.8(i), then $\langle \mathcal{K}(\mathcal{F}^{\mathcal{S}}), Z^{\mathcal{S}} \rangle \in \mathbf{ZS}$. Moreover if \rightarrow_Z is defined for $\langle \mathcal{K}(\mathcal{F}^{\mathcal{S}}), Z^{\mathcal{S}} \rangle \in \mathbf{ZS}$ as in Definition 15C.8(ii), then

$$\uparrow s \rightarrow_Z \uparrow t = \uparrow(s \rightarrow t).$$

PROOF. Immediate from Definition 15C.8. ■

15C.10. PROPOSITION. \mathcal{M} is a functor from \mathbf{ZS} to \mathbf{TS}^U .

PROOF. We just have to prove that \mathcal{M} transforms a morphism in \mathbf{ZS} into a morphism into \mathbf{TS}^U . Let $m : \langle \mathcal{D}, Z \rangle \rightarrow \langle \mathcal{D}', Z' \rangle$ be a morphism. By Lemma 15A.18 we only need to show that $\mathcal{M}(m)$ satisfies condition (m3) of Definition 13C.9. We have

$$\begin{aligned} \mathcal{M}(m)(a \rightarrow_Z b) &= m(a \rightarrow_Z b), && \text{by definition of } \mathcal{M}(m), \\ &= m(Z(a, b)), && \text{by definition of } \rightarrow_Z, \\ &\geq Z'(m(a), m(b)), && \text{since } m \text{ satisfies (Z-mon)}, \\ &= m(a) \rightarrow_{Z'} m(b), \\ &= \mathcal{M}(m(a)) \rightarrow_{Z'} \mathcal{M}(m(b)), \\ &= \mathcal{M}(m)(a) \rightarrow_{Z'} \mathcal{M}(m)(b). \blacksquare \end{aligned}$$

15C.11. PROPOSITION. \mathcal{Q} is a functor from \mathbf{TS}^U to \mathbf{ZS} .

PROOF. We have to prove that \mathcal{Q} transforms a morphism in \mathbf{TS}^U into a morphism in \mathbf{ZS} . Let $f \in \mathbf{TS}^U(\mathcal{S}, \mathcal{S}')$ with arrows $\rightarrow_{Z^{\mathcal{S}}}$ and $\rightarrow_{Z^{\mathcal{S}'}}$ corresponding to $Z^{\mathcal{S}}$ and $Z^{\mathcal{S}'}$, respectively. By Proposition 15A.16 we only need to show that $\mathcal{Q}(f)$ satisfies (Z -mon). Indeed,

$$\begin{aligned}\mathcal{Q}(f)(Z^{\mathcal{S}}(\uparrow s, \uparrow t)) &= \mathcal{Q}(f)(\uparrow(s \rightarrow_{Z^{\mathcal{S}}} t)), && \text{by definition of } Z^{\mathcal{S}}, \\ &= \uparrow f(s \rightarrow_{Z^{\mathcal{S}}} t), && \text{by Lemma 15A.15,} \\ &\subseteq \uparrow(f(s) \rightarrow_{Z^{\mathcal{S}'}} f(t)), && \text{since } f \in \mathbf{TS}^U(\mathcal{S}, \mathcal{S}'), \\ &= Z^{\mathcal{S}'}(\uparrow f(s), \uparrow f(t)), && \text{by definition of } Z^{\mathcal{S}'}, \\ &= Z^{\mathcal{S}'}(\mathcal{Q}(f)(\uparrow s), \mathcal{Q}(f)(\uparrow t)), && \text{by Lemma 15A.15.} \blacksquare\end{aligned}$$

Now we will prove that \mathbf{ZS} and \mathbf{TS}^U are equivalent. To this aim we show that natural isomorphisms $\text{Id}_{\mathbf{TS}^U} \simeq \mathcal{M} \circ \mathcal{Q}$ and $\mathcal{Q} \circ \mathcal{M} \simeq \text{Id}_{\mathbf{ZS}}$ are given by σ and τ respectively, exactly as in the case of the equivalence between the categories \mathbf{MSL}^U and \mathbf{ALG}_a .

15C.12. PROPOSITION. *Let \mathcal{S} be a top type structure. Let $\sigma : \mathcal{S} \rightarrow \mathcal{K}(\mathcal{F}^{\mathcal{S}})$ be the map such that $\sigma(s) = \uparrow s$. Then σ is an isomorphism in \mathbf{TS}^U .*

PROOF. By Proposition 15A.19(i) we only need to show that σ and σ^{-1} commute with arrows. Now σ is a bijection, hence the following suffices.

$$\begin{aligned}\sigma(s \rightarrow t) &= \uparrow(s \rightarrow t), \\ &= \uparrow s \rightarrow_{Z^{\mathcal{S}}} \uparrow t, && \text{by Lemma 15C.9,} \\ &= \sigma(s) \rightarrow_{Z^{\mathcal{S}}} \sigma(t). \blacksquare\end{aligned}$$

15C.13. PROPOSITION. *Let $\langle \mathcal{D}, Z \rangle \in \mathbf{ZS}$. Define $\tau : \mathcal{F}^{\mathcal{K}(\mathcal{D})} \rightarrow \mathcal{D}$ by*

$$\tau(x) = \bigsqcup x, \quad \text{where the sup is taken in } \mathcal{D}.$$

Then τ is an isomorphism in \mathbf{ZS} .

PROOF. By Proposition 15A.19(ii) we only need to show that τ and τ^{-1} satisfy (Z -comm). As to τ , we have

$$\begin{aligned}\tau(Z^{\mathcal{K}(\mathcal{D})}(\uparrow a, \uparrow b)) &= \tau(\uparrow(a \rightarrow_Z b)), && \text{by definition of } Z^{\mathcal{K}(\mathcal{D})}, \\ &= \tau(\uparrow Z(a, b)), && \text{by definition of } \rightarrow_Z, \\ &= \bigsqcup(\uparrow Z(a, b)) \\ &= Z(a, b) \\ &= Z(\bigsqcup \uparrow a, \bigsqcup \uparrow b) \\ &= Z(\tau(\uparrow a), \tau(\uparrow b)).\end{aligned}$$

As to τ^{-1} , we must show

$$\tau^{-1}(Z(a, b)) = Z^{\mathcal{K}(\mathcal{D})}(\tau^{-1}(a), \tau^{-1}(b)).$$

This follows by applying τ^{-1} to both sides of the equality above, using the fact that it is one-to-one. \blacksquare

15C.14. THEOREM. *The categories \mathbf{TS}^U and \mathbf{ZS} are equivalent.*

PROOF. As in Corollary 15A.20, using Propositions 15C.10-15C.13. \blacksquare

15D. Zip and lambda structures

In this section we do not proceed to a direct comparison between type and lambda structures, but put zip structures to work and compare them with lambda structures. 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 combining the results of this section and the previous one. We gain clarity when comparing lambda structures with the intermediate zip structures since both have some components in common such as the ω -algebraic lattices and their compact elements. We also avoid the confusion which arises with the reversed order and the use of filters when passing from type to lambda structures or vice-versa.

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.

15D.1. DEFINITION. Let $\langle \mathcal{D}, Z \rangle \in \mathbf{ZS}$.

- (i) Define $\Theta_{\mathcal{D}_Z}(x, y) \triangleq \{b \mid \exists a \sqsubseteq y. Z(a, b) \sqsubseteq x\}$, for $x, y \in \mathcal{D}$.
- (ii) $\mathbf{F}_{\mathbf{ZS}}$ is defined as the lambda structure $\langle \mathcal{D}, F_Z, G_Z \rangle$, with the two continuous functions $F_Z : \mathcal{D} \rightarrow [\mathcal{D} \rightarrow \mathcal{D}]$ and $G_Z : [\mathcal{D} \rightarrow \mathcal{D}] \rightarrow \mathcal{D}$ defined by

$$\begin{aligned} F_Z(x) &\triangleq \lambda y. \sqcup \Theta_{\mathcal{D}_Z}(x, y); \\ G_Z(f) &\triangleq \sqcup \{Z(a, b) \mid b \sqsubseteq f(a)\}. \end{aligned}$$

- (iii) Moreover define $\cdot_Z : \mathcal{D}^2 \rightarrow \mathcal{D}$, by

$$x \cdot_Z y \triangleq F_Z(x)(y)$$

15D.2. PROPOSITION. Let $\mathbf{m} \in \mathbf{ZS}(\langle \mathcal{D}, Z \rangle, \langle \mathcal{D}', Z' \rangle)$. Then $\mathbf{m}^R(x') = \sqcup \{x \mid \mathbf{m}(x) \sqsubseteq x'\}$ is the right adjoint of \mathbf{m} and $\underline{\mathbf{m}} = \langle \mathbf{m}, \mathbf{m}^R \rangle$ satisfies the following properties, for all $f : \mathcal{D} \rightarrow \mathcal{D}$, $x \in \mathcal{D}$, $x' \in \mathcal{D}'$:

$$\begin{aligned} \mathbf{m}(G_Z(f)) &\sqsubseteq G'_{Z'}(\mathbf{m} \circ f \circ \mathbf{m}^R) \\ F_Z(\mathbf{m}^R(x'))(x) &\sqsubseteq \mathbf{m}^R(F'_{Z'}(x')(\mathbf{m}(x))). \end{aligned}$$

PROOF. In order to simplify notation, we omit all the subscripts. We have

$$\begin{aligned} \mathbf{m}(G(f)) &= \mathbf{m}(\sqcup \{Z(a, b) \mid b \sqsubseteq f(a)\}) \\ &= \sqcup \{\mathbf{m}(Z(a, b)) \mid b \sqsubseteq f(a)\}, && \text{by 15B.2(i),} \\ &\sqsubseteq \sqcup \{Z'(\mathbf{m}(a), \mathbf{m}(b)) \mid b \sqsubseteq f(a)\}, && \text{by (Z-mon),} \\ &\sqsubseteq \sqcup \{Z'(a', b') \mid \exists a, b. \mathbf{m}(a) \sqsubseteq a' \& b' \sqsubseteq \mathbf{m}(b) \& b \sqsubseteq f(a)\} \\ &= \sqcup \{Z'(a', b') \mid \exists a. \mathbf{m}(a) \sqsubseteq a' \& b' \sqsubseteq \mathbf{m}(f(a))\}, && \text{since } f \text{ is continuous,} \\ &= \sqcup \{Z'(a', b') \mid b' \sqsubseteq \mathbf{m}(f(\mathbf{m}^R(a')))\}, && \text{by (Galois),} \\ &= G'(\mathbf{m} \circ f \circ \mathbf{m}^R). \end{aligned}$$

$$\begin{aligned}
F(\mathbf{m}^R(x'))(x) &= \bigsqcup\{b \mid \exists a \sqsubseteq x. Z(a, b) \sqsubseteq \mathbf{m}^R(x')\} \\
&\sqsubseteq \bigsqcup\{b \mid \exists a \sqsubseteq x. \mathbf{m}(Z(a, b)) \sqsubseteq x'\}, && \text{by (Galois-2),} \\
&\sqsubseteq \bigsqcup\{b \mid \exists a \sqsubseteq x. Z'(\mathbf{m}(a), \mathbf{m}(b)) \sqsubseteq x'\}, && \text{by (Z-mon),} \\
&\sqsubseteq \bigsqcup\{b \mid \exists a' \sqsubseteq \mathbf{m}(x). Z'(a', \mathbf{m}(b)) \sqsubseteq x'\} \\
&\sqsubseteq \bigsqcup\{b \mid \mathbf{m}(b) \sqsubseteq \bigsqcup\{b' \mid \exists a' \sqsubseteq \mathbf{m}(x). Z'(a', b') \sqsubseteq x'\}\} \\
&= \mathbf{m}^R(\bigsqcup\{b' \mid \exists a' \sqsubseteq \mathbf{m}(x). Z'(a', b') \sqsubseteq x'\}), && \text{see 15E.8,} \\
&= \mathbf{m}^R(F'(x'))(\mathbf{m}(x))). \blacksquare
\end{aligned}$$

We recall that the category **LS** has been defined in Definition 15B.6, using Definition 15B.5. Note that the conditions (lambda-gc1) and (lambda-gc2) of Definition 15B.5 are expressed in the thesis of previous Proposition 15D.2. As a consequence, it is immediate that $F_{\mathbf{ZS}}$ can be extended to a functor.

15D.3. PROPOSITION. *Given $\mathbf{m} \in \mathbf{ZS}(\langle \mathcal{D}, Z \rangle, \langle \mathcal{D}', Z' \rangle)$, define $F_{\mathbf{ZS}}(\mathbf{m}) \triangleq \underline{\mathbf{m}}$. Then $F_{\mathbf{ZS}} : \mathbf{ZS} \rightarrow \mathbf{LS}$ is a functor.*

PROOF. We only need to show that $F_{\mathbf{ZS}}(\mathbf{m}) = \underline{\mathbf{m}}$ is in $\mathbf{LS}(F_{\mathbf{ZS}}(\mathcal{D}), F_{\mathbf{ZS}}(\mathcal{D}'))$, when $\mathbf{m} \in \mathbf{ZS}(\mathcal{D}, \mathcal{D}')$. This follows from Proposition 15D.2. ■

As a consequence of last proposition, the fundamental operation $\mathcal{A} = F_{\mathbf{ZS}} \circ Q$ is actually a functor from \mathbf{TS}^U to **LS** (see Theorem 15D.27). 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 $\mathbf{m} : \langle \mathcal{D}, F, G \rangle \rightarrow \langle \mathcal{D}', F', G' \rangle$ between lambda structures consists of a pair of continuous functions, $\mathbf{m} : \mathcal{D} \rightarrow \mathcal{D}'$ and $\mathbf{n} : \mathcal{D}' \rightarrow \mathcal{D}$ such that

$$(4) \quad \begin{aligned} \mathbf{m} \circ G &= G' \circ (\mathbf{n} \rightarrow \mathbf{m}) \\ (\mathbf{m} \rightarrow \mathbf{n}) \circ F' &= F \circ \mathbf{n} \end{aligned}$$

where $\mathbf{n} \rightarrow \mathbf{m} : [\mathcal{D} \rightarrow \mathcal{D}] \rightarrow [\mathcal{D}' \rightarrow \mathcal{D}']$ is defined, for every $f : \mathcal{D} \rightarrow \mathcal{D}$, by $(\mathbf{n} \rightarrow \mathbf{m})(f) = \mathbf{m} \circ f \circ \mathbf{n} : \mathcal{D}' \rightarrow \mathcal{D}'$ (and similarly is defined $\mathbf{n} \circ \mathbf{m} : [\mathcal{D}' \rightarrow \mathcal{D}'] \rightarrow [\mathcal{D} \rightarrow \mathcal{D}]$). This choice allows to look at lambda structures as *dialgebras*, but does not guarantees a functorial behavior 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 15D.2 and they are obviously weaker than (4).

Equivalence between **ZS** and **LS**?

We do not know whether the categories **ZS** and **LS** are equivalent, nor do we have a counter example. Restricting both categories we do obtain equivalent categories, as we will see below.

Various categories of lambda structures

15D.4. DEFINITION. Let $\langle \mathcal{D}, F, G \rangle$ be a lambda structure. Write

$$\delta \triangleq G(\perp \mapsto \perp).$$

We say that $\langle \mathcal{D}, F, G \rangle$ is a *lazy lambda structure* if the following holds.

- (i) (δ -comp) $\delta \in \mathcal{K}(\mathcal{D})$;
- (ii) (adj1) $\forall f \in [\mathcal{D} \rightarrow \mathcal{D}]. F(G(f)) \sqsupseteq f$;
- (iii) (adj2) $\forall x \in \mathcal{D}. \delta \sqsubseteq x \Rightarrow G(F(x)) \sqsubseteq x$;
- (iv) ($\delta \perp$) $\forall x \in \mathcal{D}. \delta \not\sqsubseteq x \Rightarrow F(x) = \perp \mapsto \perp$.

15D.5. DEFINITION. A *strict lambda structure*, notation \mathbf{LS}^s , is a triple $\langle \mathcal{D}, F, G \rangle$, where $\mathcal{D} \in \mathbf{ALG}$, and $F : \mathcal{D} \rightarrow_s [\mathcal{D} \rightarrow_s \mathcal{D}]$ and $G : [\mathcal{D} \rightarrow_s \mathcal{D}] \rightarrow_s \mathcal{D}$ are continuous.

We now give the definition of various categories of lambda structures. This definition is an expansion of Definition 15B.6. By sake of completeness, we repeat the definition of the categories of lambda structures and natural lambda structures.

15D.6. DEFINITION. (i) The category \mathbf{LS} consists of lambda structures as objects and lambda Galois connections as morphisms. The composition between morphisms $\langle m, m^R \rangle : \mathcal{D} \rightarrow \mathcal{D}'$ and $\langle n, n^R \rangle : \mathcal{D}' \rightarrow \mathcal{D}''$ is given by $\langle n \circ m, m^R \circ n^R \rangle$.

(ii) The category \mathbf{LLS} is the full subcategory of \mathbf{LS} which has as objects lazy lambda structures.

(iii) The category of *natural lambda structures*, notation \mathbf{NLS} , is the full subcategory of \mathbf{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 \underline{m} such that m and m^R are strict.

(v) A *proper* lambda structure, notation \mathbf{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.

$$\mathbf{LZS} \cong \mathbf{LLS};$$

$$\mathbf{NZS} \cong \mathbf{NLS};$$

$$\mathbf{PZS}^s \cong \mathbf{PLS}^s.$$

The three equivalences are actually isomorphisms. The first two will be proved in Theorems 15D.18 and 15D.22. The third one will be proved in Exercise 15E.17.

Isomorphism between \mathbf{LZS} and \mathbf{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 results.

15D.7. LEMMA. Let $\langle \mathcal{D}, F, G \rangle$ be a lazy lambda structure. Then

$$\forall f \in [\mathcal{D} \rightarrow \mathcal{D}'] \forall x \in \mathcal{D}. f \neq (\perp \mapsto \perp) \Rightarrow [G(f) \sqsubseteq x \Leftrightarrow f \sqsubseteq F(x)].$$

PROOF. Do Exercise 15E.12. ■

Recall that $\Theta_{\mathcal{D}_Z}$ is defined in Definition 15D.1.

15D.8. REMARK. Note that by (Z-contr) and (Z-add) one has in \mathbf{LZS}

- (i) $\forall a \in \mathcal{K}(\mathcal{D}). Z(a, \perp) = Z(\perp, \perp)$.
- (ii) $\Theta_{\mathcal{D}_Z}(x, y) \neq \emptyset \Leftrightarrow \Theta_{\mathcal{D}_Z}(x, y)$ is directed.

15D.9. LEMMA. Let $\langle \mathcal{D}, Z \rangle \in \mathbf{LZS}$ and let $a, b \in \mathcal{K}(\mathcal{D})$, $x \in \mathcal{D}$, with $b \neq \perp$. 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

$$\begin{aligned} b \sqsubseteq x \cdot a &\Leftrightarrow b \sqsubseteq \bigsqcup \Theta_{\mathcal{D}_Z}(x, a), \\ &\Rightarrow \exists a_1, b_1. b \sqsubseteq b_1 \& a_1 \sqsubseteq a \& Z(a_1, b_1) \sqsubseteq x, \text{ by Remark 15D.7,} \\ &\Rightarrow Z(a, b) \sqsubseteq x. \blacksquare \end{aligned}$$

15D.10. PROPOSITION. Let $\langle \mathcal{D}, Z \rangle$ be a lazy zip structure. Then

- (i) $G_Z(a \mapsto b) = Z(a, b)$.
- (ii) $\mathsf{Fzs}(\langle \mathcal{D}, Z \rangle) = \langle \mathcal{D}, \mathsf{Fz}, \mathsf{Gz} \rangle$ is a lazy lambda structure.

PROOF. (i) We have the following.

$$\begin{aligned} G_Z(a \mapsto b) &= \bigsqcup \{Z(a', b') \mid b' \sqsubseteq (a \mapsto b)a'\} \\ &= \bigsqcup \{Z(a', b') \mid a \sqsubseteq a' \& b' \sqsubseteq b\}, \quad \text{by Remark 15D.8(i),} \\ &= Z(a, b), \quad \text{by } (Z\text{-contr).} \end{aligned}$$

(ii) We prove that $\mathsf{Fzs}(\langle \mathcal{D}, Z \rangle)$ satisfies the four points of Definition 15D.4. We have $G(\perp \mapsto \perp) = Z(\perp, \perp)$, by (i). Therefore $(\delta\text{-comp})$ holds, since $Z(\perp, \perp)$ is compact.

As to (adj1) , it is sufficient to reason about compact elements, and prove that for every a, b ,

$$b \sqsubseteq f(a) \Rightarrow b \sqsubseteq F(G(f))(a).$$

Notice that if $b \sqsubseteq f(a)$, then $b \in \Theta_{\mathcal{D}_Z}(G(f), a)$, so

$$\begin{aligned} b &\sqsubseteq \bigsqcup \Theta_{\mathcal{D}_Z}(G(f), a), \\ &= G(f) \cdot a, \\ &= F(G(f))(a). \end{aligned}$$

Now we prove (adj2) . Suppose $\delta \sqsubseteq x$, that is $Z(a, \perp) \sqsubseteq x$ for every 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 $b = \perp$, then the thesis follows from the hypothesis. If $b \neq \perp$, then the thesis follows from Lemma 15D.9.

Finally we prove $(\delta\perp)$. By $(Z\text{-lazy})$ it follows that $Z(a, b) \not\sqsubseteq x$, for all a, b . So $F(x)(y) = \perp$, for every y . \blacksquare

From Propositions 15D.10 and 15D.3 we get the following.

15D.11. THEOREM. Fzs restricts to a functor from \mathbf{LZS} to \mathbf{LLS} .

Going in the other direction, from every lazy lambda structure one can define a lazy zip structure. Before showing that, we need to extend Lemma 15B.2(i), (ii) to lazy lambda structures.

15D.12. LEMMA. Let $\langle \mathcal{D}, F, G \rangle$ be a lazy lambda structure. Then

- (i) G is additive.
- (ii) $\forall f \in \mathcal{K}([\mathcal{D} \rightarrow \mathcal{D}]). G(f) \in \mathcal{K}(\mathcal{D})$.

PROOF. (i) Similar to the proof of Lemma 15B.2(i).

(ii) If $f = (\perp \mapsto \perp)$ then $G(f) \in \mathcal{K}(\mathcal{D})$ by Definition 15D.4(i). If on the other hand $f \neq (\perp \mapsto \perp)$, then the proof is similar to that of Lemma 15B.2(ii), using Lemma 15D.7. ■

15D.13. DEFINITION. Let $\mathcal{D} = \langle \mathcal{D}, F, G \rangle$ be a lazy lambda structure. Then we can define, for every $a, b \in \mathcal{K}(\mathcal{D})$,

$$Z_{F,G}(a, b) \triangleq G(a \mapsto b)$$

and

$$\mathcal{R}(\mathcal{D}) \triangleq \langle \mathcal{D}, Z_{F,G} \rangle.$$

Because of Lemma 15D.12(ii), $\mathcal{R}(\mathcal{D})$ is a zip structure.

15D.14. PROPOSITION. Let $\mathcal{D} = \langle \mathcal{D}, F, G \rangle$ 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 15D.12(ii), $Z(a, b)$ is a compact element.

We prove (Z -contr). Let $a \sqsubseteq a'$, $b' \sqsubseteq b$. Then, in $[\mathcal{D} \rightarrow \mathcal{D}]$, $a' \mapsto b' \sqsubseteq a \mapsto b$, 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

$$\begin{aligned} Z(a, b_1 \sqcup b_2) &= G(a \mapsto (b_1 \sqcup b_2)), && \text{by definition,} \\ &= G(a \mapsto b_1) \sqcup G(a \mapsto b_2), && \text{by Lemma 15D.12(i).} \end{aligned}$$

Finally, (Z -lazy) is immediate by monotonicity of G and the fact that

$(\perp \mapsto \perp) \sqsubseteq (a \mapsto b)$, for all $a, b \in \mathcal{K}(\mathcal{D})$. ■

15D.15. LEMMA. Let $\underline{m} = \langle m, m^R \rangle \in \mathbf{LLS}(\mathcal{D}, \mathcal{D}'_{F', G'})$, where $\mathcal{D} = \langle \mathcal{D}, F, G \rangle$ and $\mathcal{D}' = \langle \mathcal{D}', F', G' \rangle$. Define $\mathcal{R}(\underline{m}) \triangleq m$. 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 15D.12. So we are left to prove that m satisfies (Z -mon), that is, for every $a, b \in K(D)$,

$$m(Z(a, b)) \sqsubseteq Z'(m(a), m(b))$$

where $Z = Z_{F,G}$, $Z' = Z_{F',G'}$. We have

$$\begin{aligned} m(Z(a, b)) &= m(G(a \mapsto b)), && \text{by definition of } Z, \\ &\sqsubseteq G'(m \circ (a \mapsto b) \circ m^R), && \text{by Definition 15B.5(i),} \\ &= G'(m(a) \mapsto m(b)), && \text{by Lemma 15B.2(iii),} \\ &= Z'(m(a), m(b)). \end{aligned}$$

From Proposition 15D.14 and Lemma 15D.15 we obtain the following.

15D.16. THEOREM. $\mathcal{R} : \mathbf{LLS} \rightarrow \mathbf{LZS}$ is a functor. ■

Actually, \mathcal{R} and $\mathcal{F}_{\mathbf{ZS}}$ set up an isomorphism between \mathbf{LLS} and \mathbf{LZS} . So the correspondence between \mathbf{LLS} and \mathbf{LZS} is perfect.

15D.17. THEOREM. (i) $\mathcal{F}_{\mathbf{ZS}} \circ \mathcal{R} = \text{Id}_{\mathbf{LLS}}$.

(ii) $\mathcal{R} \circ \mathcal{F}_{\mathbf{ZS}} = \text{Id}_{\mathbf{LZS}}$.

PROOF. (i) We have to prove that for every lazy lambda structure $\mathcal{D} = \langle \mathcal{D}, F, G \rangle$, $\mathcal{F}_{\mathbf{ZS}}(\mathcal{R}(\mathcal{D})) = \mathcal{D}$. 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 every lazy zip structure

$\langle \mathcal{D}, Z \rangle$, we have that $\mathcal{R}(\mathcal{A}(\langle \mathcal{D}, Z \rangle)) = \langle \mathcal{D}, Z \rangle$. For this aim, it is enough to prove that $Z_{G_Z} = Z$.

$$\begin{aligned} Z_{G_Z}(a, b) &= G_Z(a \mapsto b) \\ &= Z(a, b), \quad \text{by Proposition 15D.10(i).} \blacksquare \end{aligned}$$

15D.18. THEOREM. *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 $\mathcal{D} = \langle \mathcal{D}, F, G \rangle$, F and G set up a Galois connection iff \mathcal{D} is a lazy lambda structure with $\delta = \perp$.

15D.19. LEMMA. *Let $\mathcal{D} = \langle \mathcal{D}, F, G \rangle$ be a lambda structure. Then \mathcal{D} is natural (that is $\langle G, F \rangle$ is a Galois connection) iff \mathcal{D} is a lazy lambda structure with $\delta = \perp$.*

15D.20. PROPOSITION. *Let $\langle \mathcal{D}, Z \rangle$ be a natural zip structure. Then*

$$\mathsf{F}_{\mathbf{ZS}}(\langle \mathcal{D}, Z \rangle) = \langle \mathcal{D}, \mathsf{F}_Z, \mathsf{G}_Z \rangle$$

is a natural lambda structure.

PROOF. By Lemma 15D.19, since $Z(\perp, \perp) = \perp$, hence $G_Z(\perp \mapsto \perp) = \perp$. ■

So we get the following.

15D.21. THEOREM. $\mathsf{F}_{\mathbf{ZS}}$ restricts to a functor from **NZS** to **NLS**.

We now prove the other direction.

15D.22. PROPOSITION. *Let $\mathcal{D} = \langle \mathcal{D}, F, G \rangle$ be a natural lambda structure. Then $\mathcal{R}(\mathcal{D}) = \langle \mathcal{D}, Z_{F,G} \rangle$ is a natural zip structure.*

PROOF. By Proposition 15D.14, we just have to prove that $Z_{F,G}(\perp, \perp) = \perp$. By Lemma 15D.19 we know that $G(\perp \mapsto \perp) = \perp$, so we are done by the definition of $Z_{F,G}$. ■

15D.23. COROLLARY. \mathcal{R} restricts to a functor from **NLS** to **NZS**.

15D.24. THEOREM. *The categories **NLS** and **NZS** are isomorphic.*

PROOF. From Theorem 15D.18 and the fact that **NLS** and **NZS** are full subcategories of **LLS** and **LZS** respectively. ■

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 $\langle \mathcal{F}^{\mathcal{S}}, \mathcal{F}^{\mathcal{S}}, \mathcal{G}^{\mathcal{S}} \rangle$ over a type structure \mathcal{S} given in Definition 13D.5 can be seen as a functor \mathcal{A} from **TS^U** to **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 every lambda structure as the filter space of a suitable type structure. In the general case, lambda structures are not captured by type structures, and no categorical equivalence result seems 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.

15D.25. DEFINITION. Let $\mathcal{S} \in \mathbf{TS}^U$. We define the functor $\text{Flt}_{\mathbf{TS}^U}$ analogous to the one in Lemma 15B.8 and Theorem 15B.9.

1. For $\mathcal{S} \in \mathbf{TS}^U$, define $\text{Flt}_{\mathbf{TS}^U}(\mathcal{S}) \triangleq \langle \mathcal{F}^{\mathcal{S}}, F^{\mathcal{S}}, G^{\mathcal{S}} \rangle$.
2. For $f \in \mathbf{TS}^U(\mathcal{S}, \mathcal{S}')$, define $\text{Flt}_{\mathbf{TS}^U}(f) \triangleq \langle \text{Flt}(f), \text{Flt}(f)^R \rangle$

15D.26. LEMMA. \mathcal{A} is the composition of \mathcal{Q} and $\mathcal{F}_{\mathbf{ZS}}$.

PROOF. Given $f \in \mathbf{TS}^U(\mathcal{S}, \mathcal{S}')$, it is easy to see that $\mathcal{F}_{\mathbf{ZS}}(\mathcal{Q}(f)) = \mathcal{A}(f)$. Let $\mathcal{S} \in \mathbf{TS}^U$. We will prove that $\mathcal{F}_{\mathbf{ZS}}(\mathcal{Q}(\mathcal{S})) = \mathcal{A}(\mathcal{S})$. By Definitions 15C.8 and 15D.1, we have to prove that $F^{\mathcal{S}} = F_{ZS}$ and $G^{\mathcal{S}} = G_{ZS}$. Taking the suprema in $\mathcal{F}^{\mathcal{S}}$ one has

$$\begin{aligned} F^{\mathcal{S}}(X)(Y) &= \uparrow\{\uparrow A \mid \exists B \in Y. (B \rightarrow A) \in X\} \\ &= \sqcup\{\uparrow A \mid \exists B \in Y. \uparrow(B \rightarrow A) \subseteq X\} \\ &= \sqcup\{\uparrow A \mid \exists \uparrow B \subseteq Y. Z^{\mathcal{S}}(\uparrow B, \uparrow A) \subseteq X\} \\ &= F_{ZS}(X)(Y). \end{aligned}$$

Moreover,

$$\begin{aligned} G^{\mathcal{S}}(f) &= \uparrow\{B \rightarrow A \mid A \in f(\uparrow B)\} \\ &= \sqcup\{\uparrow(B \rightarrow A) \mid A \in f(\uparrow B)\} \\ &= \sqcup\{Z(\uparrow B, \uparrow A) \mid \uparrow A \subseteq f(\uparrow B)\} \\ &= G_{ZS}(f). \blacksquare \end{aligned}$$

15D.27. THEOREM. (i) $\mathcal{A} : \mathbf{TS}^U \rightarrow \mathbf{LS}$ is a functor.

(ii) \mathcal{A} restricts to a functor from \mathbf{LTS}^U to \mathbf{LLS} .

(iii) \mathcal{A} restricts to a functor from \mathbf{NTS}^U to \mathbf{NLS} .

PROOF. (i) By Lemma 15D.26, \mathcal{A} is a functor since it is the composition of two functors.

(ii), (iii) By Exercise 15E.15 along with Theorem 15D.11 for the lazy case, and Theorem 15D.21 for the natural case. ■

15D.28. THEOREM. Define $\text{Flt}_{\mathbf{NLS}} : \mathbf{LLS} \rightarrow \mathbf{LTS}^U$ as in Definition 15B.10.

(i) Given a lazy lambda structure \mathcal{D} , we define

$$\text{Flt}_{\mathbf{NLS}}(\mathcal{D}) \triangleq \langle \mathcal{K}(\mathcal{D}), \leq, \cap, \rightarrow_{\mathcal{D}}, U \rangle$$

(ii) Given $\underline{m} \in \mathbf{LLS}(\mathcal{D}, \mathcal{D}')$, we define

$$\text{Flt}_{\mathbf{NLS}}(\underline{m}) \triangleq \underline{m} \upharpoonright \mathcal{K}(\mathcal{D}) : \mathcal{K}(\mathcal{D}) \rightarrow \mathcal{K}(\mathcal{D}').$$

Then $\text{Flt}_{\mathbf{NLS}}$ is a functor. Moreover $\text{Flt}_{\mathbf{NLS}}$ restricts to a functor from \mathbf{NLS} to \mathbf{NTS}^U .

PROOF. It follows from the fact that $\text{Flt}_{\mathbf{NLS}}$ is a composition of \mathcal{R} with \mathcal{M} , which are functors as proved in Theorem 15D.16 and Corollary 15D.23 and Exercise 15E.15, both for the lazy and natural case. ■

15D.29. THEOREM. The categories \mathbf{LTS}^U and \mathbf{LLS} are equivalent.

PROOF. By Theorem 15D.18 and Exercise 15E.15 ■

15D.30. THEOREM. *The categories \mathbf{NTS}^U and \mathbf{NLS} are equivalent.*

PROOF. By Theorem 15D.24 and Exercise 15E.15. ■

15D.31. THEOREM. *The categories \mathbf{PTS}^U and \mathbf{PLS}^s are equivalent.*

PROOF. By Exercise 15E.17 and Exercise 15E.16. ■

15E. Exercises

15E.1. Let $\mathcal{S} \in \mathbf{TS}^U$. Show that

$$\emptyset \cdot X = \emptyset \text{ for all } X \in \mathcal{F}^{\mathcal{S}}$$

$$X \cdot \emptyset = \emptyset \text{ for all } X \in \mathcal{F}^{\mathcal{S}}.$$

15E.2. Let \mathcal{S} be a natural and β -sound type structure. Show that

$$\uparrow \bigcap_{i \in I} (s_i \rightarrow t_i) \cdot \uparrow s' = \bigcap_{j \in J} t_j,$$

$$\text{where } J = \{i \in I \mid s' \leq s_i\}.$$

15E.3. Let \mathcal{S} be a top type structure. Show that

$$\begin{aligned} F^{\mathcal{S}}(G^{\mathcal{S}}(\perp \mapsto \perp)) &= (\perp \mapsto \perp) \Leftrightarrow \\ (s_1 \rightarrow U) \cap \cdots \cap (s_n \rightarrow U) &\leq (s' \rightarrow t') \Rightarrow t' = U. \end{aligned}$$

15E.4. Let \mathcal{S} be an arbitrary type structure. Show that

$$G^{\mathcal{S}}(\bigsqcup_{i \in I} (\uparrow s_i \mapsto \uparrow t_i)) \supseteq \uparrow \bigcap_{i \in I} (s_i \rightarrow t_i).$$

15E.5. Let \mathcal{S} be a proper type structure. Show that

$$G^{\mathcal{S}}(\bigsqcup_{i \in I} (\uparrow s_i \mapsto \uparrow t_i)) = \uparrow \bigcap_{i \in I} (s_i \rightarrow t_i).$$

15E.6. Let \mathcal{S} be a natural type structure. Show that $G^{\mathcal{S}}(\uparrow U \mapsto \uparrow U) = \uparrow U$.

15E.7. Show that in Definition 15B.1 one has: (Galois-1) & (Galois-2) \Leftrightarrow (Galois).

15E.8. Let $\langle m, m^R \rangle : \mathcal{D} \rightarrow \mathcal{D}'$ be a Galois connection. Show that

$$(i) \quad m(x) = \prod\{x' \mid x \sqsubseteq m^R(x')\};$$

$$(ii) \quad m^R(x') = \bigsqcup\{x \mid m(x) \sqsubseteq x'\}.$$

15E.9. We consider the following dual conditions to Definition 15B.5.

$$(\text{lambda-gc1}^*) \quad \forall f' \in [\mathcal{D}' \rightarrow \mathcal{D}']. m^R(G'(f')) \sqsupseteq G(m^R \circ f' \circ m);$$

$$(\text{lambda-gc2}^*) \quad \forall x \in \mathcal{D}, x' \in \mathcal{D}'. F'(m(x))(x') \sqsupseteq m(F(x)(m^R(x'))).$$

Prove the following equivalences.

$$(i) \quad (\text{lambda-gc1}) \Leftrightarrow (\text{lambda-gc1}^*).$$

$$(ii) \quad (\text{lambda-gc2}) \Leftrightarrow (\text{lambda-gc2}^*).$$

15E.10. Let $f \in \mathbf{NTS}^U(\mathcal{S}, \mathcal{S}')$ and $f' \in \mathbf{NTS}^U(\mathcal{S}', \mathcal{S}'')$. Show that

$$\begin{aligned} \overline{f \circ f'} &= \bar{f} \circ \bar{f'}; \\ \overline{f \circ f'}^R &= \bar{f'}^R \circ \bar{f}^R; \\ \langle \overline{Id}, \overline{Id}^R \rangle &= \langle Id, Id \rangle, \quad \text{for } Id = Id_{\mathcal{S}} : \mathcal{S} \rightarrow \mathcal{S}. \end{aligned}$$

15E.11. Let $\langle m, m^R \rangle : \mathcal{D} \rightarrow \mathcal{D}'$ be a morphism in \mathbf{LS} . Show the commutativity of the following diagram

$$\begin{array}{ccc}
& \sqcup & \\
\mathcal{D} & \xleftarrow{\quad} & \mathcal{F}^{\mathcal{K}(\mathcal{D})} \\
\mathbf{m}^R \uparrow & & \uparrow \mathbf{m}|\mathcal{K}(\mathcal{D})^R \\
& \sqcup & \\
\mathcal{D}' & \xleftarrow{\quad} & \mathcal{F}^{\mathcal{K}(\mathcal{D}')}.
\end{array}$$

Note that a filter X' on $(\mathcal{K}(\mathcal{D}'), \leq)$ is a directed subset of $(\mathcal{D}', \sqsubseteq)$.

15E.12. Let $\langle \mathcal{D}, F, G \rangle$ be a lazy lambda structure, $f \in [\mathcal{D} \rightarrow \mathcal{D}]$ and $x \in \mathcal{D}$. Assume $f \neq \perp \mapsto \perp$. Show that

$$G(f) \sqsubseteq x \Leftrightarrow f \sqsubseteq F(x).$$

15E.13. Let $\langle \mathcal{D}, F, G \rangle$ and $\langle \mathcal{D}', F', G' \rangle$ be natural lambda structures and let $\underline{\mathbf{m}} = \langle m, m^R \rangle : \mathcal{D} \rightarrow \mathcal{D}'$ be a Galois connection. Show that

$$[\forall x, y \in \mathcal{D}. m(Fxy) \sqsubseteq F'(mx)(my)] \Rightarrow$$

$$[\forall f \in [\mathcal{D} \rightarrow \mathcal{D}]. m(Gf) \sqsupseteq G'(m \circ f \circ m^R)].$$

15E.14. Let $\langle \mathcal{D}, F, G \rangle$ and $\langle \mathcal{D}', F', G' \rangle$ be lambda structures. Assume $\mathbf{m} : \mathcal{D} \rightarrow \mathcal{D}'$ is a bijection such that \mathbf{m} and \mathbf{m}^{-1} are continuous and the following conditions hold for \mathbf{m} .

$$\begin{aligned}
\mathbf{m}(G(f)) &= G'(\mathbf{m} \circ f \circ \mathbf{m}^{-1}); \\
\mathbf{m}(F(d)(e)) &= F'(\mathbf{m}(d))(\mathbf{m}(e)).
\end{aligned}$$

Prove that \mathbf{m} induces an isomorphism of lambda structures $\underline{\mathbf{m}} = \langle \mathbf{m}, \mathbf{m}^{-1} \rangle : \langle \mathcal{D}, F, G \rangle \rightarrow \langle \mathcal{D}', F', G' \rangle$.

15E.15. Using restrictions of the functors \mathcal{Q} and \mathcal{M} , prove the following equivalences of categories:

$$\mathbf{LTS}^U \cong \mathbf{LZS}$$

$$\mathbf{NTS}^U \cong \mathbf{NZS}.$$

15E.16. Prove the following equivalences of categories:

$$\mathbf{TS}^{-U} \cong \mathbf{ZS}^s$$

$$\mathbf{PTS}^{-U} \cong \mathbf{PZS}^s$$

using the functors \mathcal{Q}_s and \mathcal{M}_s defined as follows.

1. For $\mathcal{S} \in \mathbf{TS}^{-U}$, define $\mathcal{Q}_s(\mathcal{S}) \in \mathbf{ZS}^s$ by $\mathcal{Q}_s(\mathcal{S}) \triangleq (\mathcal{F}^{\mathcal{S}}, Z^{\mathcal{S}})$, with $Z^{\mathcal{S}}$ as in Definition 15C.8. Note that $Z^{\mathcal{S}} : \mathcal{K}^s(\mathcal{F}^{\mathcal{S}}) \times \mathcal{K}^s(\mathcal{F}^{\mathcal{S}}) \rightarrow \mathcal{K}^s(\mathcal{F}^{\mathcal{S}})$.

For $(\mathcal{D}, Z) \in \mathbf{ZS}^s$, define $\mathcal{M}_s((\mathcal{D}, Z)) \in \mathbf{TS}^{-U}$ by

$$\mathcal{M}_s((\mathcal{D}, Z)) \triangleq \langle K^s(\mathcal{D}), \leq, \cap, \rightarrow_Z \rangle,$$

with $\leq, \cap, \rightarrow_Z$ as in Definition 15C.8.

2. Given $f \in \mathbf{TS}^{-U}(\mathcal{S}, \mathcal{S}')$, $X \in \mathcal{F}^{\mathcal{S}}$, and $g \in \mathbf{ZS}^s((\mathcal{D}, Z), (\mathcal{D}', Z'))$, define

$$\mathcal{Q}_s(f)(X) \triangleq \begin{cases} \{t \mid \exists s \in X. f(s) \leq t\}, & \text{if } X \neq \perp (= \emptyset), \\ \perp, & \text{else;} \end{cases}$$

$$\mathcal{M}_s(g) \triangleq g \upharpoonright \mathcal{K}(\mathcal{D}).$$

15E.17. (i) Let $\langle \mathcal{D}, Z \rangle \in \mathbf{ZS}^s$. Prove that the mappings F_Z and G_Z given in Definition 15D.1 are strict and that $F_Z : \mathcal{D} \rightarrow_s [\mathcal{D} \rightarrow_s \mathcal{D}]$ and $G_Z : [\mathcal{D} \rightarrow_s \mathcal{D}] \rightarrow_s \mathcal{D}$.

- (ii) Let $\langle \mathcal{D}, F, G \rangle \in \mathbf{PLS}^s$. Show that if $f \neq \perp \mapsto \perp$ then $G(f) \neq \perp$ and conclude that $Z_{F,G}$ introduced in Definition 15D.13 is a mapping from $\mathcal{K}^s(\mathcal{D}) \times \mathcal{K}^s(\mathcal{D})$ to $\mathcal{K}^s(\mathcal{D})$.
- (iii) Prove that $F_{\mathbf{ZS}}$ is a functor from \mathbf{PZS}^s to \mathbf{PLS}^s (see Definition 15D.1 and Proposition 15D.3).
- (iv) Prove that \mathcal{R} is a functor from \mathbf{PLS}^s to \mathbf{PZS}^s (see Definition 15D.13 and Lemma 15D.15).
- (v) Prove that the categories \mathbf{PZS}^s and \mathbf{PLS}^s are isomorphic.

CHAPTER 16

FILTER MODELS

Filter models are models of the untyped lambda calculus where terms are interpreted as sets of types. The domain of a filter model will be the set $\mathcal{F}^{\mathcal{T}}$ of filters as defined in Section 13D, i.e.

$$[\![M]\!]^{\mathcal{F}^{\mathcal{T}}} \in \mathcal{F}^{\mathcal{T}}.$$

The application and the abstraction will be interpreted using the functions $F^{\mathcal{T}}$ and $G^{\mathcal{T}}$ of a filter structure, see Definition 13D.5. Variations on \mathcal{T} induce different interpretations $[\![\]]^{\mathcal{F}^{\mathcal{T}}}$ on λ -terms which may or may not satisfy β -conversion. This leads to define the following different classes of filter models.

Classes of $\mathcal{F}^{\mathcal{T}}$	Rule satisfied by $[\![\]]^{\mathcal{F}^{\mathcal{T}}}$
<i>Filter λ-model</i>	β -conversion
<i>Filter λI-model</i>	βI -conversion

Moreover, filter models could be *extensional* or *non-extensional* depending on whether $[\![\]]^{\mathcal{F}^{\mathcal{T}}}$ satisfies η -conversion or not.

The first important property that will be proved in Section 16B is the so-called Type-semantics Theorem. This theorem states that the interpretation of a closed term is the set of its types. More formally, if M is closed,

$$[\![M]\!]^{\mathcal{F}^{\mathcal{T}}} = \{A \mid \vdash_{\cap}^{\mathcal{T}} M : A\}.$$

A first consequence of this theorem is that the interpretation on a filter structure satisfies $\beta(I)(\eta)$ -conversion exactly when the type assignment system does. Hence, Fig. 41 and Fig. 42 can be easily deduced from Fig. 36 and Fig. 37, respectively.

Property of $\mathcal{T} \in \text{TT}$	versus	property of $\mathcal{F}^{\mathcal{T}}$
$\mathcal{T} \in \text{TT}^U, \beta\text{-sound}$	\Rightarrow	filter λ -model
$\mathcal{T} \in \text{TT}^{-U}, \beta\text{-sound}$	\Rightarrow	filter λI -model
$\mathcal{T} \in \text{TT}^U, \beta\text{-sound, natural, } \eta^U\text{-sound}$	\Rightarrow	extensional filter λ -model
$\mathcal{T} \in \text{TT}^{-U}, \beta\text{-sound, proper, } \eta\text{-sound}$	\Rightarrow	extensional filter λI -model

FIGURE 41. Conditions on type theories for inducing filter models

Class of filter model $\mathcal{F}^{\mathcal{T}}$	Type theory \mathcal{T}
Extensional filter λ -model	Scott, Park, CDZ, HR, DHM
Extensional filter $\lambda\text{-l}$ -model	HL
Non-extensional filter λ -model	BCD, AO, Plotkin, Engeler, CDS
Non-extensional filter $\lambda\text{-l}$ -model	CDV, CD

FIGURE 42. Classification of filter models

Section 16B also studies representability of continuous functions ([Coppo, Dezani-Ciancaglini, Honsell, and Longo \[1984\]](#), [Alessi, Barbanera, and Dezani-Ciancaglini \[2004\]](#)). We prove that \mathcal{T} is β -sound iff all continuous functions are representable in $\mathcal{F}^{\mathcal{T}}$. At the end of the section, we show an example of a non β -sound type theory called ABD which induces a filter lambda model. As a consequence, not all continuous functions are representable in \mathcal{F}^{ABD} . This example also shows that the condition of β -soundness does not have the full power to characterize the type theories whose type assignment is closed under β -reduction, or equivalently, whose filter structure is a λ -model.

Section 16C shows the relation between Scott's \mathcal{D}_{∞} -models (see Section 18.2 of [B\[1984\]](#) for definition and properties of \mathcal{D}_{∞}) and filter models. It is well known that Scott's \mathcal{D}_{∞} -models are models of the lambda calculus that satisfy the recursive domain equation $\mathcal{D} = [\mathcal{D} \rightarrow \mathcal{D}]$. The construction of \mathcal{D}_{∞} does not only depend on the initial \mathcal{D}_0 , but also on the projection pair i_0, j_0 ⁵ which gives the start of the \mathcal{D}_{∞} construction:

$$i_0 : \mathcal{D}_0 \rightarrow \mathcal{D}_1, \quad j_0 : \mathcal{D}_1 \rightarrow \mathcal{D}_0,$$

where $\mathcal{D}_1 = [\mathcal{D}_0 \rightarrow \mathcal{D}_0]$. Given the triple $t = (\mathcal{D}_0, i_0, j_0)$, we write $\mathcal{D}_{\infty} = \mathcal{D}_{\infty}^t$ to emphasize the dependency on t . Variations of t define different \mathcal{D}_{∞}^t -models. Some instances of $t = (\mathcal{D}_0, i_0, j_0)$ have given rise to some specific filter models such as $\mathcal{F}^{\text{Scott}}$ and $\mathcal{F}^{\text{Park}}$, see [Scott \[1972\]](#), [Park \[1976\]](#), [Barendregt, Coppo, and Dezani-Ciancaglini \[1983\]](#). Actually we will show that any \mathcal{D}_{∞}^t model in the category **ALG** of ω -algebraic complete lattices can be described as a filter model $\mathcal{F}^{\text{CDHL}(t)}$ by considering the compact elements of \mathcal{D}_0 as atomic types and defining a type theory $\text{CDHL}(t)$ that contains both the order of \mathcal{D}_0 and i_0 , see [Barendregt, Coppo, and Dezani-Ciancaglini \[1983\]](#), [Coppo, Dezani-Ciancaglini, Honsell, and Longo \[1984\]](#), [Coppo, Dezani-Ciancaglini, and Zacchi \[1987\]](#), [Alessi, Dezani-Ciancaglini, and Honsell \[2004\]](#). We will first prove that

$$\mathcal{K}(\mathcal{D}_{\infty}^t) \cong [\text{CDHL}(t)].$$

Then, by Proposition 15B.13, we get our result that \mathcal{D}_{∞}^t can be described as $\mathcal{F}^{\text{CDHL}(t)}$, i.e.

$$\mathcal{D}_{\infty}^t \cong \mathcal{F}^{[\text{CDHL}(t)]} \cong \mathcal{F}^{\text{CDHL}(t)}.$$

The converse is obviously not true. Filter models are in a sense weaker structures than \mathcal{D}_{∞} -models. Not all of them satisfy the recursive domain equation $\mathcal{D} = [\mathcal{D} \rightarrow \mathcal{D}]$ for they can be non-extensional. If we restrict our attention to the extensional filter models of Fig. 42, then all of them, i.e. $\mathcal{F}^{\text{Scott}}$, $\mathcal{F}^{\text{Park}}$, \mathcal{F}^{CDZ} , \mathcal{F}^{HR} , and \mathcal{F}^{DHM} , can be described as \mathcal{D}_{∞} models by choosing an appropriate t , see [Coppo, Dezani-Ciancaglini, and Longo \[1983\]](#),

⁵In [B\[1984\]](#), Definition 18.2.1, the maps i_0 and j_0 are called respectively φ_0 and ψ_0 .

Coppo, Dezani-Ciancaglini, Honsell, and Longo [1984], Alessi [1991], Dezani-Ciancaglini, Ghilezan, and Likavec [2004]. One obtains the following versions of \mathcal{D}_∞ ,

$$\mathcal{D}_\infty^{\text{Scott}}, \mathcal{D}_\infty^{\text{Park}}, \mathcal{D}_\infty^{\text{CDZ}}, \mathcal{D}_\infty^{\text{DHM}} \text{ and } \mathcal{D}_\infty^{\text{HR}}.$$

Given $\mathcal{T} \in \{\text{Scott, Park, CDZ, DHM, HR}\}$, we will associate a triple $\text{init}(\mathcal{T}) = (\mathcal{D}_0, i_0, j_0)$ such that $\text{CDHL}(\text{init}(\mathcal{T})) = \mathcal{T}$. Then,

$$\mathcal{D}_\infty^{\text{init}(\mathcal{T})} \cong \mathcal{F}^{\text{CDHL}(\text{init}(\mathcal{T}))} = \mathcal{F}^\mathcal{T}.$$

We will write $\mathcal{D}_\infty^\mathcal{T} := \mathcal{D}_\infty^{\text{init}(\mathcal{T})}$. Then the equation becomes

$$\mathcal{D}_\infty^\mathcal{T} \cong \mathcal{F}^{\text{CDHL}(\text{init}(\mathcal{T}))} = \mathcal{F}^\mathcal{T}.$$

The pleasant fact is that \mathcal{T} and the triple $t = (\mathcal{D}_0, i_0, j_0)$ correspond to each other in a canonical way. For each of the theories $\mathcal{T} \in \{\text{Scott, Park}\}$ the model $\mathcal{D}_\infty^\mathcal{T}$ was constructed first and the natural type theory \mathcal{T} came later. For $\mathcal{T} \in \{\text{CDZ, DHM, HR}\}$ one constructed this natural type theory in order to obtain the model $\mathcal{D}_\infty^\mathcal{T}$ satisfying certain properties.

Although the non-extensional filter models do not satisfy $\mathcal{D} = [\mathcal{D} \rightarrow \mathcal{D}]$, in some cases it is possible to find other recursive domain equations for them, see Alessi [1991]. For instance, the non-extensional filter model \mathcal{F}^{AO} satisfies the equation $\mathcal{D} = [\mathcal{D} \rightarrow \mathcal{D}]_\perp$ and \mathcal{F}^{BCD} satisfies the equation $\mathcal{D} = [\mathcal{D} \rightarrow \mathcal{D}] \times \mathbf{P}(\mathbb{A}_\infty)$.

Section 16D studies other filter models. The first model considered is AO which is shown to be computationally adequate for the lazy operational semantics, see Abramsky and Ong [1993]. The second one is used as an application of intersection types to prove consistency of certain equations in the λ -calculus. Following Alessi, Dezani-Ciancaglini, and Honsell [2001] we show that $\Omega := (\lambda x.xx)(\lambda x.xx)$ is an *easy* lambda term in the sense of Jacopini and Venturini-Zilli: $\lambda\beta \cup \{\Omega = M\}$, is consistent for all $M \in \Lambda$. This has been shown in various ways, see e.g. Jacopini [1975], Baeten and Boerboom [1979] or Mitschke's proof in B[1984], Proposition 15.3.9. Given any λ -term M , we inductively build natural intersection type theories $\text{ADH}_n(M)$ in such a way that the union of these theories, called $\text{ADH}(M)$, forces the interpretation of M to coincide with the interpretation of Ω , i.e.

$$\mathcal{F}^{\text{ADH}(M)} \models M = \Omega.$$

Further applications of intersection types consist of necessary conditions for filter λ -models to be *sensible* or *semi-sensible*. We will not consider this issue, see Zylberajch [1991]. At the end of this section, we describe some graph models as filter models.

16A. Lambda models

In this Section we generalize the basic notions and properties of λ -models. That definition was given in Section 3A. We now introduce also *quasi λ -models*. This makes it possible to differentiate between models for the λ -calculus and for the λI -calculus.

16A.1. DEFINITION. (i) Let \mathcal{D} be a set and V the set of variables of the untyped lambda calculus. An *environment in \mathcal{D}* is a total map

$$\rho : V \rightarrow \mathcal{D}.$$

The set of environments in \mathcal{D} is denoted by $\text{Env}_{\mathcal{D}}$.

(ii) If $\rho \in \text{Env}_{\mathcal{D}}$ and $d \in \mathcal{D}$, then $\rho[x := d]$ is the $\rho' \in \text{Env}_{\mathcal{D}}$ defined by

$$\rho'(y) = \begin{cases} d & \text{if } y = x, \\ \rho(y) & \text{otherwise.} \end{cases}$$

Remember that an applicative structure is a pair $\langle \mathcal{D}, \cdot \rangle$, consisting of a set \mathcal{D} together with a binary operation $\cdot : \mathcal{D} \times \mathcal{D} \rightarrow \mathcal{D}$ on it.

16A.2. DEFINITION. (i)

(ii) A *quasi λ -model* is of the form

$$\mathcal{D} = \langle \mathcal{D}, \cdot, [\]^{\mathcal{D}} \rangle,$$

where $\langle \mathcal{D}, \cdot \rangle$ is an applicative structure and $[\]^{\mathcal{D}} : \Lambda \times \text{Env}_{\mathcal{D}} \rightarrow \mathcal{D}$ satisfies the following.

$$(1) \quad [x]_{\rho}^{\mathcal{D}} = \rho(x);$$

$$(2) \quad [MN]_{\rho}^{\mathcal{D}} = [M]_{\rho}^{\mathcal{D}} \cdot [N]_{\rho}^{\mathcal{D}};$$

$$(3) \quad [\lambda x.M]_{\rho}^{\mathcal{D}} = [\lambda y.M[x := y]]_{\rho}^{\mathcal{D}}, \quad (\alpha)$$

provided $y \notin \text{FV}(M)$;

$$(4) \quad \forall d \in \mathcal{D}. [M]_{\rho[x := d]}^{\mathcal{D}} = [N]_{\rho[x := d]}^{\mathcal{D}} \Rightarrow [\lambda x.M]_{\rho}^{\mathcal{D}} = [\lambda x.N]_{\rho}^{\mathcal{D}}; \quad (\xi)$$

$$(5) \quad \rho \upharpoonright \text{FV}(M) = \rho' \upharpoonright \text{FV}(M) \Rightarrow [M]_{\rho}^{\mathcal{D}} = [M]_{\rho'}^{\mathcal{D}}.$$

(iii) A *λ -model* is a quasi λ -model which satisfies

$$(6) \quad [\lambda x.M]_{\rho}^{\mathcal{D}} \cdot d = [M]_{\rho[x := d]}^{\mathcal{D}} \quad (\beta)$$

This is consistent with Definition 3A.31.

(iv) A *λI -model* is a quasi λ -model which satisfies

$$(6') \quad x \in \text{FV}(M) \Rightarrow [\lambda x.M]_{\rho}^{\mathcal{D}} \cdot d = [M]_{\rho[x := d]}^{\mathcal{D}} \quad (\beta I)$$

16A.3. REMARK. As noticed by R. Hindley the present definition of λI -model requires that every λK term have an interpretation, even though it plays no “active” role. But K is not in the λI language (as defined by most people, e.g. Barendregt [1984] p.185). On the other hand, this approach agrees with Church’s original λ -calculus Church [1932], [1933], in which K was allowed in the language but not permitted to be active in redexes.

We will write simply $[\]_{\rho}$ instead of $[\]_{\rho}^{\mathcal{D}}$ when there is no danger of confusion.

We have the following implications.

$$\mathcal{D} \text{ } \lambda\text{-model} \implies \mathcal{D} \text{ } \lambda I\text{-model} \implies \mathcal{D} \text{ quasi } \lambda\text{-model.}$$

The first class of applicative structures satisfies (β) in general, the second only for λI -redexes and the third class does not need to satisfy (β) at all, but there is an interpretation for λ -terms.

16A.4. DEFINITION. Let $\mathcal{D} = \langle \mathcal{D}, \cdot, [\] \rangle$ be a quasi λ -model.

(i) The statement $M = N$, for M, N untyped lambda terms, is *true in \mathcal{D}* , notation $\mathcal{D} \models M = N$ if

$$\forall \rho \in \text{Env}_{\mathcal{D}}. [M]_{\rho} = [N]_{\rho}.$$

(ii) As usual one defines $\mathcal{D} \models \chi$, where χ is any statement built up using first order predicate logic from equations between untyped lambda terms.

(iii) A $\lambda(\mathbb{I})$ -model \mathcal{D} is called *extensional* if

$$\mathcal{D} \models (\forall x. Mx = Nx) \Rightarrow M = N.$$

(iv) A $\lambda(\mathbb{I})$ -model \mathcal{D} is called an *η -model* if

$$\mathcal{D} \models \lambda x. Mx = M, \text{ for } x \notin \text{FV}(M). \quad (\eta)$$

We will see now how the notions of $\lambda(\mathbb{I})$ -model and (strict) lambda structure are related, see Definition 15B.3 and Definition 15D.5.

16A.5. DEFINITION. (i) Let $\langle \mathcal{D}, F, G \rangle$ be a lambda structure, see Definition 15B.3. We define the triple $\langle \mathcal{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 $d, e \in \mathcal{D}$

$$d \cdot_F e \triangleq F(d)(e).$$

- Then the map $[\]^{F,G} : \Lambda \times \text{Env}_{\mathcal{D}} \rightarrow \mathcal{D}$ is defined as follows.

$$\begin{aligned} [x]_{\rho}^{F,G} &\triangleq \rho(x); \\ [MN]_{\rho}^{F,G} &\triangleq F([M]_{\rho}^{F,G})([N]_{\rho}^{F,G}); \\ [\lambda x. M]_{\rho}^{F,G} &\triangleq G(\lambda d \in \mathcal{D}. [M]_{\rho[x:=d]}^{F,G}), \end{aligned}$$

noticing that the map $\lambda d \in \mathcal{D}. [M]_{\rho[x:=d]}$ used for $[\lambda x. M]_{\rho}$ is continuous.

(ii) Let $\langle \mathcal{D}, F, G \rangle$ be a *strict* lambda structure. We define the triple $\langle \mathcal{D}, \cdot_F, [\]^{F,G} \rangle$ as above, changing the clause for $[\lambda x. M]_{\rho}^{F,G}$ into

$$[\lambda x. M]_{\rho}^{F,G} \triangleq G(\lambda d \in \mathcal{D}. \text{if } d = \perp_{\mathcal{D}} \text{ then } \perp_{\mathcal{D}} \text{ else } [M]_{\rho[x:=d]}^{F,G}).$$

16A.6. PROPOSITION. Let $\langle \mathcal{D}, F, G \rangle$ be a (strict) lambda structure. Then $\langle \mathcal{D}, \cdot_F, [\]^{F,G} \rangle$ is a quasi λ -model.

PROOF. Easy. ■

16A.7. DEFINITION. Let $\langle \mathcal{D}, F, G \rangle$ be a (strict) lambda structure. Then,

$$\mathcal{D} = \langle \mathcal{D}, \cdot_F, [\]^{F,G} \rangle$$

is called the *quasi λ -model* induced by $\langle \mathcal{D}, F, G \rangle$. 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 $\lambda(\mathbb{I})$ -model is the axiom $(\beta(\mathbb{I}))$.

16A.8. PROPOSITION. (i) Let $\mathcal{D} = \langle \mathcal{D}, \cdot_F, [\]^{F,G} \rangle$ be the quasi λ -model induced by the lambda structure $\langle \mathcal{D}, F, G \rangle$. Then the following statements are equivalent.

- (1) $\mathcal{D} \models (\lambda x. M)N = M[x := N]$, for all $M, N \in \Lambda$;
- (2) $[\lambda x. M]_{\rho}^{F,G} \cdot d = [M]_{\rho(x:=d)}^{F,G}$, for all $M \in \Lambda$ and $d \in \mathcal{D}$;
- (3) \mathcal{D} is a λ -model;

- (4) $\mathcal{D} \models \{M = N \mid \lambda\beta \vdash M = N\}.$
- (ii) Let $\mathcal{D} = \langle \mathcal{D}, \cdot_F, \llbracket \cdot \rrbracket^{F,G} \rangle$ be the quasi λ -model induced by the strict lambda structure $\langle \mathcal{D}, F, G \rangle$. Then the following statements are equivalent.
 - (1) $\mathcal{D} \models (\lambda x.M)N = M[x := N]$, for all $M, N \in \Lambda$ such that $x \in \text{FV}(M)$;
 - (2) $\llbracket \lambda x.M \rrbracket_\rho^{F,G} \cdot d = \llbracket M \rrbracket_{\rho(x:=d)}^{F,G}$, for all $M \in \Lambda$ with $x \in \text{FV}(M)$, and $d \in \mathcal{D}$;
 - (3) \mathcal{D} is a λl -model;
 - (4) $\mathcal{D} \models \{M = N \mid \lambda\beta l \vdash M = N\}.$

PROOF. (i) (1) \Rightarrow (2). By (1) one has $\llbracket (\lambda x.M)N \rrbracket_\rho^{F,G} = \llbracket M[x := N] \rrbracket_\rho^{F,G}$. Taking $N \equiv x$ and $\rho' = \rho(x := d)$ one obtains

$$\llbracket (\lambda x.M)x \rrbracket_{\rho'}^{F,G} = \llbracket M \rrbracket_{\rho'}^{F,G},$$

hence

$$\llbracket \lambda x.M \rrbracket_\rho^{F,G} \cdot d = \llbracket M \rrbracket_{\rho'}^{F,G},$$

as $\rho \upharpoonright \text{FV}(\lambda x.M) = \rho' \upharpoonright \text{FV}(\lambda x.M)$.

(2) \Rightarrow (3). By (ii), Definition 16A.5 and Proposition 16A.6 all conditions for being a λ -model are fulfilled, see Definition 16A.2.

(3) \Rightarrow (4). By Theorem 5.3.4 in B[1984].

(4) \Rightarrow (1). Trivial.

(ii) Similarly. ■

In Part I, Definition 10A.13, we required $F \circ G = \text{Id}_{[\mathcal{D} \rightarrow \mathcal{D}]}$ for a lambda structure to be a lambda model. This condition implies representability of all continuous functions, see Lemma 16B.15. The theory ABD defined in Definition 16B.19 gives rise to a lambda model, where not all continuous functions are representable.

16A.9. COROLLARY. Let \mathcal{D} be the $\lambda(l)$ -model induced by the (strict) lambda structure $\langle \mathcal{D}, F, G \rangle$. Then

$$\mathcal{D} \text{ is a } \lambda(l)\eta\text{-model} \Leftrightarrow \mathcal{D} \text{ is an extensional } \lambda(l)\text{-model.}$$

PROOF. (\Rightarrow) Suppose that for some ρ one has for all $d \in \mathcal{D}$

$$\llbracket Mx \rrbracket_{\rho[x:=d]}^{F,G} = \llbracket Nx \rrbracket_{\rho[x:=d]}^{F,G}.$$

Then by (η) and Proposition 16A.6(ii) one has

$$\llbracket M \rrbracket_\rho^{F,G} = \llbracket \lambda x.Mx \rrbracket_\rho^{F,G} = \llbracket \lambda x.Nx \rrbracket_\rho^{F,G} = \llbracket N \rrbracket_\rho^{F,G}.$$

(\Leftarrow) Note that by ($\beta(l)$) one has $\mathcal{D} \models (\lambda x.Mx)y = My$, where x is fresh. Hence by extensionality one has $\mathcal{D} \models \lambda x.Mx = M$. ■

Isomorphisms of λ -models

This section relates isomorphisms between lambda structures and lambda models.

16A.10. DEFINITION. We say that \mathcal{D} and \mathcal{D}' are **isomorphic λ -models** (via \mathbf{m}), notation $(\mathbf{m} :) \mathcal{D} \cong \mathcal{D}'$, if \mathbf{m} is a bijection and for all λ -terms M and environments ρ :

$$\mathbf{m}(\llbracket M \rrbracket_{\rho}^{\mathcal{D}}) = \llbracket M \rrbracket_{\mathbf{m} \circ \rho}^{\mathcal{D}'}$$

16A.11. LEMMA. *If two λ -models \mathcal{D} and \mathcal{D}' are isomorphic, then they equate the same terms, i.e. $\mathcal{D} \models M = N \Leftrightarrow \mathcal{D}' \models M = N$.*

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 15E.14.

16A.12. LEMMA. *Let $\underline{\mathbf{m}} = \langle \mathbf{m}, \mathbf{m}^R \rangle : \langle \mathcal{D}, F, G \rangle \rightarrow \langle \mathcal{D}', F', G' \rangle$ be an isomorphism between lambda structures. Then $\mathbf{m} : \mathcal{D} \rightarrow \mathcal{D}'$ is a bijective continuous map such that*

$$\begin{aligned} (\text{iso-ls1}) \quad \mathbf{m}(G(f)) &= G'(\mathbf{m} \circ f \circ \mathbf{m}^R); \\ (\text{iso-ls2}) \quad \mathbf{m}(F(d)(e)) &= F'(\mathbf{m}(d))(\mathbf{m}(e)). \end{aligned}$$

If we write $f^{\mathbf{m}} = \mathbf{m} \circ f \circ \mathbf{m}^{-1}$ then we can reformulate these conditions as

$$\begin{aligned} \mathbf{m}(G(f)) &= G'(f^{\mathbf{m}}); \\ \mathbf{m}(d \cdot_F e) &= \mathbf{m}(d) \cdot_{F'} \mathbf{m}(e). \end{aligned}$$

PROOF. By Definition 15B.5 we get:

$$\begin{aligned} (\text{lambda-gc1}) \quad \forall f \in [\mathcal{D} \rightarrow \mathcal{D}]. \mathbf{m}(G(f)) &\sqsubseteq G'(\mathbf{m} \circ f \circ \mathbf{m}^R); \\ (\text{lambda-gc2}) \quad \forall x' \in \mathcal{D}', x \in \mathcal{D}. F(m^R(x'))(x) &\sqsubseteq m^R(F'(x'))(\mathbf{m}(x)). \end{aligned}$$

As to (iso-ls1). Since $\underline{\mathbf{m}}$ is an isomorphism, we have that besides the lambda Galois connection

$$\langle \mathbf{m}, \mathbf{m}^R \rangle : \mathcal{D} \rightarrow \mathcal{D}',$$

there is another lambda Galois connection $\underline{\mathbf{m}}^{-1}$, which we call $\underline{\mathbf{n}} = \langle \mathbf{n}, \mathbf{n}^R \rangle : \mathcal{D}' \rightarrow \mathcal{D}$ such that

$$\begin{aligned} \underline{\mathbf{n}} \circ \underline{\mathbf{m}} &= \langle \text{Id}_{\mathcal{D}}, \text{Id}_{\mathcal{D}} \rangle, \\ \underline{\mathbf{m}} \circ \underline{\mathbf{n}} &= \langle \text{Id}_{\mathcal{D}'}, \text{Id}_{\mathcal{D}'} \rangle. \end{aligned}$$

Using composition between Galois connections, this amounts to saying

$$(1) \quad \begin{aligned} \mathbf{n} \circ \mathbf{m} &= \text{Id}_{\mathcal{D}}, \\ \mathbf{m}^R \circ \mathbf{n}^R &= \text{Id}_{\mathcal{D}}, \\ \mathbf{m} \circ \mathbf{n} &= \text{Id}_{\mathcal{D}'}, \\ \mathbf{n}^R \circ \mathbf{m}^R &= \text{Id}_{\mathcal{D}'}. \end{aligned}$$

Looking at (1), we see that compositions of \mathbf{m} and \mathbf{n} give the identities. So $\mathbf{n} = \mathbf{m}^{-1}$. This implies that \mathbf{n} is a right adjoint of \mathbf{m} . But the right adjoint is unique, so $\mathbf{m}^{-1} = \mathbf{n} = \mathbf{m}^R$. For the same reason $\mathbf{n}^R = \mathbf{m}$. Therefore we have proved that $(\underline{\mathbf{m}}^{-1} = \underline{\mathbf{n}} =) \langle \mathbf{n}, \mathbf{n}^R \rangle = \langle \mathbf{m}^R, \mathbf{m} \rangle$. Note that, as $\underline{\mathbf{n}}$ is a lambda Galois connection, we have that

$$(2) \quad \begin{cases} \mathbf{m}^R \text{ is (also) the left adjoint of } \mathbf{m}, \text{ and} \\ \mathbf{m} \text{ is (also) the right adjoint of } \mathbf{m}^R. \end{cases}$$

We are now in the position to prove (1). The inequality

$$\mathbf{m}(G(f)) \sqsubseteq G'(\mathbf{m} \circ f \circ \mathbf{m}^R)$$

is (lambda-gc1). As to the other inequality, first of all note that, exploiting (2), we have that conditions (lambda-gc1) and (lambda-gc2) induce, for any $f' : \mathcal{D}' \rightarrow \mathcal{D}'$, and $x \in \mathcal{D}, x' \in \mathcal{D}'$,

$$(3) \quad \begin{aligned} \mathbf{m}^R(G'(f')) &\sqsubseteq G(\mathbf{m}^R \circ f' \circ \mathbf{m}), \\ F'(\mathbf{m}(x))(x') &\sqsubseteq \mathbf{m}(F(x)(\mathbf{m}^R(x'))). \end{aligned}$$

So we have

$$\begin{aligned} G'(\mathbf{m} \circ f \circ \mathbf{m}^R) &= \mathbf{m} \circ \mathbf{m}^R \circ G'(\mathbf{m} \circ f \circ \mathbf{m}^R), && \text{by (1),} \\ &\sqsubseteq \mathbf{m} \circ G(\mathbf{m}^R \circ \mathbf{m} \circ f \circ \mathbf{m}^R \circ \mathbf{m}), && f' = \mathbf{m} \circ f \circ \mathbf{m}^R \text{ in (b),} \\ &= \mathbf{m} \circ G(f), && \text{since } \mathbf{m}^R = \mathbf{m}^{-1}. \end{aligned}$$

Therefore we have proved $\mathbf{m}(G(f)) = G'(\mathbf{m} \circ f \circ \mathbf{m}^{-1})$.

As to (iso-ls2). Notice that $\mathbf{m}(F(d))(e) \sqsubseteq F'(\mathbf{m}(d))(\mathbf{m}(e))$, since

$$\begin{aligned} \mathbf{m}(F(d))(e) &= \mathbf{m}(F(\mathbf{m}^R(\mathbf{m}(d)))(e)), && \text{by (1),} \\ &\sqsubseteq \mathbf{m}(\mathbf{m}^R(F'(\mathbf{m}(d))(\mathbf{m}(e)))), && \text{by (lambda-gc2),} \\ &= F'(\mathbf{m}(d))(\mathbf{m}(e)), && \text{by (1).} \end{aligned}$$

On the other hand, we have also $F'(\mathbf{m}(d))(\mathbf{m}(e)) \sqsubseteq \mathbf{m}(F(d)(e))$. In fact

$$\begin{aligned} F'(\mathbf{m}(d))(\mathbf{m}(e)) &\sqsubseteq \mathbf{m}(F(d)(\mathbf{m}^R(\mathbf{m}(e)))), && \text{by (3),} \\ &= \mathbf{m}(F(d)(e)), && \text{by (1).} \end{aligned}$$

The following proposition will be used in Corollary 16C.31 to prove that the models \mathcal{D}_∞ and $\mathcal{F}^{\text{Scott}}$ equate the same terms.

16A.13. PROPOSITION. *Let \mathcal{D} and \mathcal{D}' be isomorphic as lambda structures. Then*

- (i) \mathcal{D} and \mathcal{D}' are isomorphic as λ -models.
- (ii) They equate the same terms, i.e. $\mathcal{D} \models M = N \Leftrightarrow \mathcal{D}' \models M = N$.

PROOF. (i) By induction on M using Lemma 16A.12.

(ii) Using (i) and Lemma 16A.11. ■

16B. 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 14, 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.

16B.1. DEFINITION. Let $\mathcal{T} \in \text{TT}$. The *filter quasi λ -model* of \mathcal{T} is a quasi λ -model, denoted by $\mathcal{F}^\mathcal{T}$, where $\llbracket \cdot \rrbracket_{\rho}^{\mathcal{F}^\mathcal{T}} : \Lambda \times \text{Env}_{\mathcal{F}^\mathcal{T}} \rightarrow \mathcal{F}^\mathcal{T}$ is defined by

$$\begin{aligned} \llbracket x \rrbracket_{\rho}^{\mathcal{F}^\mathcal{T}} &\triangleq \rho(x); \\ \llbracket MN \rrbracket_{\rho}^{\mathcal{F}^\mathcal{T}} &\triangleq \uparrow\{B \in \mathbb{T}^\mathcal{T} \mid \exists A \in \llbracket N \rrbracket_{\rho}^{\mathcal{F}^\mathcal{T}} . (A \rightarrow B) \in \llbracket M \rrbracket_{\rho}^{\mathcal{F}^\mathcal{T}}\}; \\ \llbracket \lambda x. M \rrbracket_{\rho}^{\mathcal{F}^\mathcal{T}} &\triangleq \uparrow\{A \rightarrow B \mid B \in \llbracket M \rrbracket_{\rho[x:=\uparrow A]}^{\mathcal{F}^\mathcal{T}}\}. \end{aligned}$$

The notion of filter structure $\langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle$ given in Definition 13D.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 16B.1. This leads to the following lemma:

16B.2. LEMMA. *Let $\mathcal{T} \in \text{TT}$. 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 16B.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 15B.3 and the Remark just after 13D.5. In case $\mathcal{T} \in \text{TT}^{-U}$, we have that $\langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle$ is also a strict lambda structure, i.e.

$$F^{\mathcal{T}} \in [\mathcal{F}^{\mathcal{T}} \rightarrow_s [\mathcal{F}^{\mathcal{T}} \rightarrow_s \mathcal{F}^{\mathcal{T}}]] \text{ and } G^{\mathcal{T}} \in [[\mathcal{F}^{\mathcal{T}} \rightarrow_s \mathcal{F}^{\mathcal{T}}] \rightarrow_s \mathcal{F}^{\mathcal{T}}],$$

see Definition 15D.5. Hence $\langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle$ induces a quasi λ -model, by Proposition 16A.6. It is easy to see that $\llbracket \] \llbracket^{F^{\mathcal{T}}, G^{\mathcal{T}}} = \llbracket \] \llbracket^{\mathcal{F}^{\mathcal{T}}}$. ■

We now define two classes of filter models: filter λ -models and filter λI -models.

16B.3. DEFINITION. (i) Let $\mathcal{T} \in \text{TT}^U$. We say that $\mathcal{F}^{\mathcal{T}}$ is a *filter model* if the filter quasi λ -model $\mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, \cdot, \llbracket \] \llbracket^{\mathcal{F}^{\mathcal{T}}} \rangle$ is a λ -model.

(ii) Let $\mathcal{T} \in \text{TT}^{-U}$. We say that $\mathcal{F}^{\mathcal{T}}$ is a *filter model* for λI if the filter quasi λ -model $\mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, \cdot, \llbracket \] \llbracket^{\mathcal{F}^{\mathcal{T}}} \rangle$ is a λI -model.

16B.4. PROPOSITION. (i) *Let $\mathcal{T} \in \text{TT}^U$. Then $\mathcal{F}^{\mathcal{T}}$ is a filter λ -model iff for all $M, N \in \Lambda$*

$$\llbracket (\lambda x.M)N \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}} = \llbracket M[x := N] \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}}.$$

(ii) *Let $\mathcal{T} \in \text{TT}^{-U}$. Then $\mathcal{F}^{\mathcal{T}}$ is a filter λI -model iff for all $M, N \in \Lambda$*

$$x \in \text{FV}(M) \Rightarrow \llbracket (\lambda x.M)N \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}} = \llbracket M[x := N] \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}}.$$

PROOF. Both equivalences follow from Proposition 16A.8. ■

The following Type-semantics Theorem 16B.7 is important. It has as consequence that for a closed untyped lambda term M and a $\mathcal{T} \in \text{TT}$ one has

$$\llbracket M \rrbracket^{\mathcal{F}^{\mathcal{T}}} = \{A \mid \vdash_{\cap}^{\mathcal{T}} M : A\},$$

i.e. the semantical meaning of M is the collection of its types.

16B.5. DEFINITION. Let Γ be a context and $\rho \in \text{Env}_{\mathcal{F}^{\mathcal{T}}}$. Then Γ *agrees* ρ , notation $\Gamma \models \rho$, if

$$(x : A) \in \Gamma \Rightarrow A \in \rho(x).$$

16B.6. PROPOSITION. (i) $\Gamma \models \rho \& \Gamma' \models \rho \Rightarrow \Gamma \uplus \Gamma' \models \rho$.

(ii) $\Gamma \models \rho[x := \uparrow A] \Rightarrow \Gamma \setminus x \models \rho$.

PROOF. Immediate. ■

16B.7. THEOREM (Type-semantics Theorem). *Let $\mathcal{T} \in \text{TT}$ and $\langle \mathcal{F}^{\mathcal{T}}, \cdot, \llbracket \] \llbracket^{\mathcal{F}^{\mathcal{T}}} \rangle$ its corresponding filter quasi λ -model. Then, for any $M \in \Lambda$ and $\rho \in \text{Env}_{\mathcal{F}^{\mathcal{T}}}$,*

$$\llbracket M \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}} = \{A \mid \Gamma \vdash_{\cap}^{\mathcal{T}} M : A \text{ for some } \Gamma \models \rho\}.$$

PROOF. We have two cases:

(i) Let $\mathcal{T} \in \text{TT}^{\mathbb{U}}$. By induction on the structure of M .

Case $M \equiv x$. Then

$$\begin{aligned}\llbracket M \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}} &= \rho(x) \\ &= \{A \mid A \in \rho(x)\} \\ &= \{A \mid A \in \rho(x) \& x : A \vdash_{\cap}^{\mathcal{T}} x : A\} \\ &= \{A \mid \Gamma \vdash_{\cap}^{\mathcal{T}} x : A \text{ for some } \Gamma \models \rho\},\end{aligned}$$

by Definition 16B.5 and the Inversion Lemma 14A.1(i).

Case $M \equiv NL$. Then

$$\begin{aligned}\llbracket M \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}} &= \llbracket N \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}} \cdot \llbracket L \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}} \\ &= \uparrow\{A \mid \exists B \in \llbracket L \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}}.(B \rightarrow A) \in \llbracket N \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}}\} \\ &= \{A \mid \exists k > 0 \exists B_1, \dots, B_k, C_1, \dots, C_k. \\ &\quad [(B_i \rightarrow C_i) \in \llbracket N \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}} \& B_i \in \llbracket L \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}} \& (\bigcap_{1 \leq i \leq k} C_i) \leq A]\} \cup \uparrow\{\mathbb{U}\}, \\ &\quad \text{by definition of } \uparrow, \\ &= \{A \mid \exists k > 0 \exists B_1, \dots, B_k, C_1, \dots, C_k, \Gamma_1, \dots, \Gamma_k, \Delta_1, \dots, \Delta_k \\ &\quad [\Gamma_i, \Delta_i \models \rho \& \Gamma_i \vdash_{\cap}^{\mathcal{T}} N : (B_i \rightarrow C_i) \\ &\quad \& \Delta_i \vdash_{\cap}^{\mathcal{T}} L : B_i \& C_1 \cap \dots \cap C_k \leq A]\} \cup \uparrow\{\mathbb{U}\}, \\ &\quad \text{by the induction hypothesis,} \\ &= \{A \mid \Gamma \vdash_{\cap}^{\mathcal{T}} NL : A \text{ for some } \Gamma \models \rho\}, \\ &\quad \text{taking } \Gamma = \Gamma_1 \uplus \dots \uplus \Gamma_k \uplus \dots \uplus \Delta_1 \uplus \dots \uplus \Delta_k, \\ &\quad \text{by Theorem 14A.1(ii) and Proposition 16B.6(i).}\end{aligned}$$

Case $M \equiv \lambda x.N$. Then

$$\begin{aligned}\llbracket \lambda x.N \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}} &= G^{\mathcal{T}}(\lambda X \in \mathcal{F}^{\mathcal{T}}. \llbracket N \rrbracket_{\rho[x:=X]}^{\mathcal{F}^{\mathcal{T}}}) \\ &= \uparrow\{(B \rightarrow C) \mid C \in \llbracket N \rrbracket_{\rho[x:=\uparrow B]}^{\mathcal{F}^{\mathcal{T}}}\} \\ &= \{A \mid \exists k > 0 \exists B_1, \dots, B_k, C_1, \dots, C_k, \Gamma_1, \dots, \Gamma_k. \Gamma_i \models \rho[x := \uparrow B_i] \\ &\quad \& \Gamma_i, x:B_i \vdash_{\cap}^{\mathcal{T}} N : C_i \& (B_1 \rightarrow C_1) \cap \dots \cap (B_k \rightarrow C_k) \leq A\}, \\ &\quad \text{by the induction hypothesis,} \\ &= \{A \mid \Gamma \vdash_{\cap}^{\mathcal{T}} \lambda x.N : A \text{ for some } \Gamma \models \rho\}, \\ &\quad \text{taking } \Gamma = (\Gamma_1 \uplus \dots \uplus \Gamma_k) \setminus x, \text{ by Theorem 14A.1(iii), rule } (\leq) \\ &\quad \text{and Proposition 16B.6.}\end{aligned}$$

(ii) Let $\mathcal{T} \in \text{TT}^{-\mathbb{U}}$. Similarly. Note that in the case $M = NL$ we drop ‘ $\cup \uparrow\{\mathbb{U}\}$ ’ both times. ■

16B.8. COROLLARY. (i) Let $\mathcal{T} \in \text{TT}^U$. Then

$$\mathcal{F}^\mathcal{T} \text{ is a filter } \lambda\text{-model} \Leftrightarrow [\Gamma \vdash_{\cap}^{\mathcal{T}} (\lambda x.M) : (B \rightarrow A) \Rightarrow \Gamma, x:B \vdash_{\cap}^{\mathcal{T}} M : A].$$

(ii) Let $\mathcal{T} \in \text{TT}^{-U}$. Then

$$\mathcal{F}^\mathcal{T} \text{ is a filter } \lambda I\text{-model} \Leftrightarrow$$

$$[\Gamma \vdash_{\cap}^{\mathcal{T}} (\lambda x.M) : (B \rightarrow A) \& x \in \text{FV}(M) \Rightarrow \Gamma, x:B \vdash_{\cap}^{\mathcal{T}} M : A].$$

PROOF. (i) By Propositions 16B.4(i), 14B.1(i) and Corollary 14B.5(i).

(ii) By Propositions 16B.4(ii), 14B.1(ii) and Corollary 14B.5(ii). ■

16B.9. COROLLARY. (i) Let $\mathcal{T} \in \text{TT}^U$. Then

$$\mathcal{T} \text{ is } \beta\text{-sound} \Rightarrow \mathcal{F}^\mathcal{T} \text{ is a filter } \lambda\text{-model.}$$

(ii) Let $\mathcal{T} \in \text{TT}^{-U}$. Then

$$\mathcal{T} \text{ is } \beta\text{-sound} \Rightarrow \mathcal{F}^\mathcal{T} \text{ is a filter } \lambda I\text{-model.}$$

PROOF. By the Corollary above and Theorem 14A.9(iii). ■

16B.10. COROLLARY. (i) Let $\mathcal{T} \in \{\text{Scott, Park, CDZ, HR, DHM, BCD, AO, Plotkin, Engeler, CDS}\}$. Then

$$\mathcal{F}^\mathcal{T} \text{ is a filter } \lambda\text{-model.}$$

(ii) Let $\mathcal{T} \in \{\text{HL, CDV, CD}\}$. Then

$$\mathcal{F}^\mathcal{T} \text{ is a filter } \lambda I\text{-model.}$$

PROOF. (i) By (i) of the previous Corollary and Theorem 14A.7.

(ii) By (ii) of the Corollary, using Theorem 14A.7. ■

16B.11. PROPOSITION. (i) Let $\mathcal{T} \in \text{TT}^U$. Then

$$\mathcal{T} \text{ is natural and } \beta\text{- and } \eta^U\text{-sound} \Rightarrow \mathcal{F}^\mathcal{T} \text{ is an extensional filter } \lambda\text{-model.}$$

(ii) Let $\mathcal{T} \in \text{TT}^{-U}$. Then

$$\mathcal{T} \text{ is proper and } \beta\text{- and } \eta\text{-sound} \Rightarrow \mathcal{F}^\mathcal{T} \text{ is an extensional filter } \lambda I\text{-model.}$$

PROOF. (i) and (ii). $\mathcal{F}^\mathcal{T}$ is a $\lambda(I)$ -model by Corollary 16B.9(i)((ii)). As to extensionality it suffices by Corollary 16A.9 to verify for $x \notin \text{FV}(M)$ that

$$\llbracket \lambda x.Mx \rrbracket_\rho = \llbracket M \rrbracket_\rho. \tag{\eta}$$

This follows from Theorems 16B.7, and 14B.15. ■

16B.12. COROLLARY. (i) Let $\mathcal{T} \in \{\text{Scott, Park, CDZ, HR, DHM}\}$. Then

$$\mathcal{F}^\mathcal{T} \text{ is an extensional filter } \lambda\text{-model.}$$

(ii) Let $\mathcal{T} = \text{HL}$. Then

$$\mathcal{F}^\mathcal{T} \text{ is an extensional filter } \lambda I\text{-model.}$$

PROOF. (i) and (ii) follow from Corollary 14B.13. ■

As shown in [Meyer \[1982\]](#), see also [B\[1984\] Ch.4](#), a lambda structure \mathcal{D} is a λ -model provided that there are elements $K, S, \varepsilon \in \mathcal{D}$, satisfying certain properties. Thus, a condition for being a filter λ -model can be obtained by simply requiring the existence of such elements. This yields a characterization of the natural type theories which induce λ -models. See [Alessi \[1991\]](#) for the rather technical proof.

16B.13. THEOREM. Let $\mathcal{T} \in \text{NTT}^U$.

(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*

$$C \leq E \Leftrightarrow \begin{aligned} & \forall D \exists k \geq 1, A_1, \dots, A_k, B_1, \dots, B_k. \\ & (A_1 \rightarrow B_1 \rightarrow A_1) \cap \dots \cap (A_k \rightarrow B_k \rightarrow A_k) \leq C \rightarrow D \rightarrow E. \end{aligned}$$

(S) *For all D, E, F, G one has*

$$\exists H. [E \leq F \rightarrow H \& D \leq F \rightarrow H \rightarrow G] \Leftrightarrow \left[\begin{array}{l} \exists k \geq 1, A_1, \dots, A_k, B_1, \dots, B_k, C_1, \dots, C_k. \\ [((A_1 \rightarrow B_1 \rightarrow C_1) \rightarrow (A_1 \rightarrow B_1) \rightarrow A_1 \rightarrow C_1) \cap \\ \dots \\ \cap ((A_k \rightarrow B_k \rightarrow C_k) \rightarrow (A_k \rightarrow B_k) \rightarrow A_k \rightarrow C_k)] \leq D \rightarrow E \rightarrow F \rightarrow G \end{array} \right].$$

(ε) *For all C, D one has*

$$\left[\begin{array}{l} \exists k \geq 1, A_1, \dots, A_k, B_1, \dots, B_k. \\ ((A_1 \rightarrow B_1) \rightarrow A_1 \rightarrow B_1) \cap \dots \cap ((A_k \rightarrow B_k) \rightarrow A_k \rightarrow B_k) \leq (C \rightarrow D) \end{array} \right] \Leftrightarrow \exists m \geq 1, E_1, \dots, E_m, F_1, \dots, F_m. C \leq (E_1 \rightarrow F_1) \cap \dots \cap (E_m \rightarrow F_m) \leq D.$$

(ii) *The structure $\mathcal{F}^\mathcal{T}$ is an extensional filter λ -model iff the third condition above is replaced by the following two.*

- (ε₁) $\forall A \exists k \geq 1, A_1, \dots, A_k, B_1, \dots, B_k. A = (A_1 \rightarrow B_1) \cap \dots \cap (A_k \rightarrow B_k);$
- (ε₂) $\forall A, B \exists k \geq 1, A_1, \dots, A_k. [(A_1 \rightarrow A_1) \cap \dots \cap (A_k \rightarrow A_k) \leq (A \rightarrow B)$
 $\Leftrightarrow A \leq B]. \blacksquare$

Representability of continuous functions

In this subsection following [Alessi, Barbanera, and Dezani-Ciancaglini \[2004\]](#) we will isolate a number of conditions on a $\mathcal{T} \in \text{NTT}^U$ to characterize properties of the set of *representable functions* in $\langle \mathcal{F}^\mathcal{T}, \mathcal{F}^\mathcal{T}, \mathcal{G}^\mathcal{T} \rangle$, i.e. the set of functions in the image of $\mathcal{F}^\mathcal{T}$.

16B.14. DEFINITION. A function $f : \mathcal{D} \rightarrow \mathcal{D}$ is called *representable* in the lambda structure $\langle \mathcal{D}, \mathcal{F}, \mathcal{G} \rangle$ if $f = F(d)$ for some $d \in \mathcal{D}$.

Note that since $F : \mathcal{D} \rightarrow [\mathcal{D} \rightarrow \mathcal{D}]$, all representable functions are continuous.

16B.15. LEMMA. Let $\mathcal{T} \in \text{NTT}^U$ and let $f \in [\mathcal{F}^\mathcal{T} \rightarrow \mathcal{F}^\mathcal{T}]$. Then

$$f \text{ is representable in } \langle \mathcal{F}^\mathcal{T}, \mathcal{F}^\mathcal{T}, \mathcal{G}^\mathcal{T} \rangle \Leftrightarrow F^\mathcal{T} \circ G^\mathcal{T}(f) = f.$$

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 $G^\mathcal{T}(F^\mathcal{T}(X)) = \uparrow\{A \rightarrow B \mid A \rightarrow B \in X\}$. Hence

$$A \rightarrow B \in G^\mathcal{T}(F^\mathcal{T}(X)) \Leftrightarrow A \rightarrow B \in X.$$

So $\forall Y.F^{\mathcal{T}}(G^{\mathcal{T}}(F^{\mathcal{T}}(X)))(Y) = F^{\mathcal{T}}(X)(Y)$, hence $F^{\mathcal{T}}(G^{\mathcal{T}}(f)) = f$. ■

16B.16. LEMMA. Let $\mathcal{T} \in \text{NTT}^U$. Let $A, B \in \mathbb{T}^{\mathcal{T}}$. Then

$$G^{\mathcal{T}}(\uparrow A \Rightarrow \uparrow B) = \uparrow(A \rightarrow B).$$

PROOF.

$$\begin{aligned} G^{\mathcal{T}}(\uparrow A \Rightarrow \uparrow B) &= \uparrow\{(C \rightarrow D) \mid D \in (\uparrow A \Rightarrow \uparrow B)(\uparrow C)\} \\ &= \uparrow(A \rightarrow B). \end{aligned}$$

In the last step the inclusion \supseteq is trivial: $(A \rightarrow B)$ is one of the $C \rightarrow D$. Now suppose $C \rightarrow D$ 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 $A \rightarrow B \leq C \rightarrow D$ by rule (\rightarrow) . Case $\uparrow A \not\subseteq \uparrow C$. Then $D = U$, hence $C \rightarrow D = U$, by rules (\rightarrow) , $(U \rightarrow)$, and again $A \rightarrow B \leq C \rightarrow D$. Therefore also \subseteq holds in the last equation. ■

16B.17. LEMMA. Let $\mathcal{T} \in \text{NTT}^U$ and define the function $h : \mathcal{F}^{\mathcal{T}} \rightarrow \mathcal{F}^{\mathcal{T}}$ by

$$h \triangleq (\uparrow A_1 \Rightarrow \uparrow B_1) \sqcup \cdots \sqcup (\uparrow A_n \Rightarrow \uparrow B_n).$$

Then, for all $C \in \mathbb{T}^{\mathcal{T}}$ we have that

$$(i) \quad h(\uparrow C) = \{D \mid \exists k \geq 1 \exists i_1 \cdots i_k. [B_{i_1} \cap \cdots \cap B_{i_k} \leq D \ \& \ C \leq A_{i_1} \cap \cdots \cap A_{i_k}]\} \cup \uparrow\{U\}.$$

$$(ii) \quad (F^{\mathcal{T}} \circ G^{\mathcal{T}})(h)(\uparrow C) = \{D \mid (A_1 \rightarrow B_1) \cap \cdots \cap (A_n \rightarrow B_n) \leq (C \rightarrow D)\}.$$

$$\begin{aligned} \text{PROOF. } (i) \quad h(\uparrow C) &= \bigcup\{\uparrow B_i \mid \uparrow A_i \subseteq \uparrow C \ \& \ 1 \leq i \leq n\} \\ &= \uparrow B_{i_1} \sqcup \cdots \sqcup \uparrow B_{i_k} \\ &\quad \text{for } \{i_1, \dots, i_k\} = \{i \mid C \leq A_i \ \& \ 1 \leq i \leq n\} \\ &= \uparrow(B_{i_1} \cap \cdots \cap B_{i_k}) \cup \uparrow U, \text{ by Proposition 13D.4(iii),} \\ &= \{D \mid \exists k \geq 1 \exists i_1 \cdots i_k. \\ &\quad B_{i_1} \cap \cdots \cap B_{i_k} \leq D \ \& \ C \leq A_{i_1} \cap \cdots \cap A_{i_k}\} \cup \uparrow\{U\}. \end{aligned}$$

$$\begin{aligned} (ii) \quad (F^{\mathcal{T}} \circ G^{\mathcal{T}})(h)(\uparrow C) &= \\ &= F^{\mathcal{T}}(G^{\mathcal{T}}(\uparrow A_1 \Rightarrow \uparrow B_1) \sqcup \cdots \sqcup G^{\mathcal{T}}(\uparrow A_n \Rightarrow \uparrow B_n))(\uparrow C), \text{ by Lemma 15B.2(ii),} \\ &= F^{\mathcal{T}}(\uparrow(A_1 \rightarrow B_1) \sqcup \cdots \sqcup \uparrow(A_n \rightarrow B_n))(\uparrow C), \text{ by Lemma 16B.16,} \\ &= F^{\mathcal{T}}(\uparrow((A_1 \rightarrow B_1) \cap \cdots \cap (A_n \rightarrow B_n)))(\uparrow C), \text{ by Proposition 13D.4(iii),} \\ &= \{D \mid \exists E \in \uparrow C. (E \rightarrow D) \in \uparrow((A_1 \rightarrow B_1) \cap \cdots \cap (A_n \rightarrow B_n))\}, \\ &\quad \text{see Definition 13D.5(ii),} \\ &= \{D \mid \exists E \geq C. (A_1 \rightarrow B_1) \cap \cdots \cap (A_n \rightarrow B_n) \leq (E \rightarrow D)\} \\ &= \{D \mid (A_1 \rightarrow B_1) \cap \cdots \cap (A_n \rightarrow B_n) \leq (C \rightarrow D)\}, \text{ by } (\rightarrow). \blacksquare \end{aligned}$$

NOTATION. We define the function $\mathbb{K}^{\mathcal{T}} \in \mathcal{F}^{\mathcal{T}} \rightarrow \mathcal{F}^{\mathcal{T}} \rightarrow \mathcal{F}^{\mathcal{T}}$ as

$$\mathbb{K}^{\mathcal{T}} \triangleq \lambda X \in \mathcal{F}^{\mathcal{T}}. \lambda Y \in \mathcal{F}^{\mathcal{T}}. X.$$

For each $X \in \mathcal{F}^{\mathcal{T}}$, $\mathbb{K}^{\mathcal{T}}(X)$ is a constant function in $[\mathcal{F}^{\mathcal{T}} \rightarrow \mathcal{F}^{\mathcal{T}}]$.

16B.18. THEOREM. Let $\mathcal{T} \in \text{NTT}^U$. Let $\mathbf{R} \subseteq [\mathcal{F}^{\mathcal{T}} \rightarrow \mathcal{F}^{\mathcal{T}}]$ 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) \mathbf{R} contains the bottom function $\mathbb{K}^{\mathcal{T}}(\perp_{\mathcal{F}\mathcal{T}})$ iff for all C, D

$$\mathbb{U} \leq C \rightarrow D \Rightarrow \mathbb{U} \leq D.$$

(ii) \mathbf{R} contains all constant functions iff for all B, C, D

$$\mathbb{U} \rightarrow B \leq C \rightarrow D \Rightarrow B \leq D.$$

(iii) \mathbf{R} contains all continuous step functions iff for all A, B, C, D

$$A \rightarrow B \leq C \rightarrow D \& D \neq \mathbb{U} \Rightarrow C \leq A \& B \leq D.$$

(iv) \mathbf{R} contains (i.e. is the set of) all continuous functions iff \mathcal{T} is β -sound.

PROOF. (i) Assume that $\mathbb{K}^{\mathcal{T}}(\perp_{\mathcal{F}\mathcal{T}}) \in \mathbf{R}$. Then $F^{\mathcal{T}}(G^{\mathcal{T}}(\mathbb{K}^{\mathcal{T}}(\perp_{\mathcal{F}\mathcal{T}}))) = \mathbb{K}^{\mathcal{T}}(\perp_{\mathcal{F}\mathcal{T}})$, by Lemma 16B.15. Observe that

$$\begin{aligned} G^{\mathcal{T}}(\mathbb{K}^{\mathcal{T}}(\perp_{\mathcal{F}\mathcal{T}})) &= \uparrow\{A \rightarrow \mathbb{U} \mid A \in \mathbb{T}\}, && \text{by Definition 13D.5(i),} \\ &= \uparrow\mathbb{U}, && \text{since } \mathcal{T} \text{ is natural.} \end{aligned}$$

Hence $F^{\mathcal{T}}(\perp_{\mathcal{F}\mathcal{T}}) = \mathbb{K}^{\mathcal{T}}(\perp_{\mathcal{F}\mathcal{T}})$, so in particular $F^{\mathcal{T}}(\perp_{\mathcal{F}\mathcal{T}})(\uparrow C) = \perp_{\mathcal{F}\mathcal{T}}$, and hence

$$\begin{aligned} \{D \mid \mathbb{U} \leq D\} &= \uparrow\mathbb{U} = \perp_{\mathcal{F}\mathcal{T}} \\ &= F^{\mathcal{T}}(\perp_{\mathcal{F}\mathcal{T}})(\uparrow C) \\ &= F^{\mathcal{T}}(\uparrow\mathbb{U})(\uparrow C) \\ &= \{D \mid \mathbb{U} \leq (C \rightarrow D)\}, && \text{by Definition 13D.5(i).} \end{aligned}$$

But then $\mathbb{U} \leq C \rightarrow D \Rightarrow \mathbb{U} \leq D$.

(\Leftarrow) Suppose $\mathbb{U} \leq C \rightarrow D \Rightarrow \mathbb{U} \leq D$. We show that $F^{\mathcal{T}}(\perp_{\mathcal{F}\mathcal{T}}) = \mathbb{K}^{\mathcal{T}}(\perp_{\mathcal{F}\mathcal{T}})$. Indeed,

$$\begin{aligned} F^{\mathcal{T}}(\perp_{\mathcal{F}\mathcal{T}})(X) &= \{B \mid \exists A \in X. (A \rightarrow B) \in \perp_{\mathcal{F}\mathcal{T}}\} \\ &= \{B \mid \exists A \in X. (A \rightarrow B) \in \uparrow\mathbb{U}\} \\ &= \{B \mid \exists A \in X. \mathbb{U} \leq (A \rightarrow B)\} \\ &= \{B \mid \mathbb{U} \leq B\}, && \text{by the assumption,} \\ &= \uparrow\mathbb{U} = \perp_{\mathcal{F}\mathcal{T}}. \end{aligned}$$

(ii) Suppose that $\mathbb{U} \rightarrow B \leq C \rightarrow D \Rightarrow B \leq D$. We first show that each compact constant function $\mathbb{K}^{\mathcal{T}}(\uparrow B)$ is represented by $\uparrow(\mathbb{U} \rightarrow B)$. Indeed,

$$\begin{aligned} D \in \uparrow(\mathbb{U} \rightarrow B) \cdot \uparrow C &\Leftrightarrow C \rightarrow D \in \uparrow(\mathbb{U} \rightarrow B), && \text{by } (\rightarrow), \\ &\Leftrightarrow \mathbb{U} \rightarrow B \leq C \rightarrow D \\ &\Leftrightarrow B \leq D, && \text{using the assumption,} \\ &\Leftrightarrow D \in \uparrow B = \mathbb{K}^{\mathcal{T}}(\uparrow B)(\uparrow C). \end{aligned}$$

Now we show that an arbitrary constant function $\mathbb{K}^{\mathcal{T}}(X)$ is representable. Then $X = \bigcup\{\uparrow B \mid B \in X\}$, where $\{\uparrow B \mid B \in X\}$ is directed. Notice that $\mathbb{K}^{\mathcal{T}}(X) = \bigsqcup_{B \in X} \mathbb{K}^{\mathcal{T}}(\uparrow B)$.

Therefore by the representability of $\mathbb{K}^{\mathcal{T}}(\uparrow B)$ just proved, Lemma 16B.15 and the continuity of $F^{\mathcal{T}} \circ G^{\mathcal{T}}$ we get

$$\begin{aligned}\mathbb{K}^{\mathcal{T}}(X) &= \bigsqcup_{B \in X} \mathbb{K}^{\mathcal{T}}(\uparrow B) \\ &= \bigsqcup_{B \in X} (F^{\mathcal{T}} \circ G^{\mathcal{T}})(\mathbb{K}^{\mathcal{T}}(\uparrow B)) \\ &= (F^{\mathcal{T}} \circ G^{\mathcal{T}})\left(\bigsqcup_{B \in X} \mathbb{K}^{\mathcal{T}}(\uparrow B)\right) \\ &= (F^{\mathcal{T}} \circ G^{\mathcal{T}})(\mathbb{K}^{\mathcal{T}}(X)),\end{aligned}$$

hence again by Lemma 16B.15 the constant map $\mathbb{K}^{\mathcal{T}}(X)$ is representable.

Conversely, suppose that all constant functions are representable. Then $F^{\mathcal{T}} \circ G^{\mathcal{T}}(\mathbb{K}^{\mathcal{T}}(\uparrow B)) = \mathbb{K}^{\mathcal{T}}(\uparrow B)$, by Lemma 16B.15. Therefore

$$\begin{aligned}\mathbb{U} \rightarrow B \leq C \rightarrow D &\Rightarrow (C \rightarrow D) \in \uparrow(\mathbb{U} \rightarrow B) \\ &\Rightarrow D \in \uparrow(\mathbb{U} \rightarrow B) \cdot \uparrow C \\ &\Rightarrow D \in ((F^{\mathcal{T}} \circ G^{\mathcal{T}})(\mathbb{K}^{\mathcal{T}}(\uparrow B)))(\uparrow C), \\ &\quad \text{since } \uparrow(\mathbb{U} \rightarrow B) \subseteq \mathbb{K}^{\mathcal{T}}(\uparrow B) \text{ by } (\rightarrow) \text{ and } (\mathbb{U}), \\ &\Rightarrow D \in (\mathbb{K}^{\mathcal{T}}(\uparrow B))(\uparrow C) = \uparrow B \\ &\Rightarrow B \leq D.\end{aligned}$$

(iii) (\Rightarrow) Suppose all continuous step functions are representable. Suppose $A \rightarrow B \leq C \rightarrow D$, $D \neq \mathbb{U}$. Take $h = \uparrow A \Rightarrow \uparrow B$. By Lemma 16B.17(ii) we have

$$\begin{aligned}(F^{\mathcal{T}} \circ G^{\mathcal{T}})(h)(\uparrow C) &= \{E \mid A \rightarrow B \leq C \rightarrow E\} \\ h(\uparrow C) &= \{E \mid B \leq E \& C \leq A\} \cup \uparrow \mathbb{U}.\end{aligned}$$

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

$$(4) \quad (F^{\mathcal{T}} \circ G^{\mathcal{T}})(h) = h.$$

By Lemma 15A.3(ii) it suffices to show this for compact h . If $Y \neq \perp_{\mathcal{F}^{\mathcal{T}}}$, then by 15A.6 both X, Y are compact, so $h = \uparrow A \rightarrow \uparrow B$. Then (4) holds by Lemma 16B.17 and the assumption. If $Y = \perp_{\mathcal{F}^{\mathcal{T}}}$, then h is the bottom function and hence representable (the assumption in (iii) implies the assumption in (i)).

(iv) Let $\mathcal{T} \in \text{NTT}$. Let $h \in \mathcal{K}([\mathcal{F}^{\mathcal{T}} \rightarrow \mathcal{F}^{\mathcal{T}}])$. By Proposition 15A.8(ii) it follows that for some $A_1, \dots, A_n, B_1, \dots, B_n \in \mathcal{T}$

$$h = (\uparrow A_1 \Rightarrow \uparrow B_1) \sqcup \dots \sqcup (\uparrow A_n \Rightarrow \uparrow B_n),$$

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 16B.15, one has

$$(F^{\mathcal{T}} \circ G^{\mathcal{T}})(h)(\uparrow C) = h(\uparrow C).$$

It follows from Lemma 16B.17(i),(ii) that \mathcal{T} is β -sound.

(\Leftarrow) Suppose \mathcal{T} is β -sound. Again by Lemma 16B.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 15A.3 they coincide everywhere. But then it follows again that $f = (F^{\mathcal{T}} \circ G^{\mathcal{T}})(f)$ for every continuous $f : \mathcal{F}^{\mathcal{T}} \rightarrow \mathcal{F}^{\mathcal{T}}$. Hence Lemma 16B.15 applies. ■

To induce a filter model β -soundness is not necessary

The intersection type theories $\mathcal{T} \in \{\text{Scott, Park, BCD, CDZ, HR, AO, DHM}\}$ all induce filter λ -models, by Corollary 16B.10(i). These type theories are all natural and β -sound. Therefore by Theorem 16B.18(iv) all continuous functions are representable in these $\mathcal{F}^{\mathcal{T}}$. In Sections 16C and 16D 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}^{ABD} . Therefore by the same theorem not all continuous functions are representable in \mathcal{F}^{ABD} . The model builds on an idea in [Coppo, Dezani-Ciancaglini, Honsell, and Longo \[1984\]](#). In Exercise 16E.4 another such model, due to Alessi [1993], is constructed. See also [Alessi, Barbanera, and Dezani-Ciancaglini \[2004\]](#).

The theory ABD

16B.19. DEFINITION. Let $\mathbb{A}^{\text{ABD}} \triangleq \{\text{U}, \diamond, \heartsuit\}$. We define **ABD** as the smallest natural type theory⁶ that contains the axiom (\diamond) where

$$(\diamond) \quad A \leq_{\text{ABD}} A[\diamond := \heartsuit].$$

16B.20. LEMMA. (i) $A \leq_{\text{ABD}} B \Rightarrow A[\diamond := \heartsuit] \leq_{\text{ABD}} B[\diamond := \heartsuit]$.

$$(ii) \quad \Gamma \vdash^{\text{ABD}} M : A \Rightarrow \Gamma[\diamond := \heartsuit] \vdash^{\text{ABD}} M : A[\diamond := \heartsuit].$$

$$(iii) \quad \Gamma, \Gamma' \vdash^{\text{ABD}} M : A \Rightarrow \Gamma, \Gamma'[\diamond := \heartsuit] \vdash^{\text{ABD}} M : A[\diamond := \heartsuit].$$

$$(iv) \quad \Gamma, x:A_i \vdash^{\text{ABD}} M : B_i \text{ for } 1 \leq i \leq n \& \\ (A_1 \rightarrow B_1) \cap \dots \cap (A_n \rightarrow B_n) \leq_{\text{ABD}} C \rightarrow D \Rightarrow \Gamma, x:C \vdash^{\text{ABD}} M : D.$$

PROOF. (i) By induction on the definition of \leq_{ABD} .

(ii) By induction on derivations using (i) for rule (\leq_{ABD}) .

(iii) From (ii) and rule $(\leq_{\text{ABD}}\text{-L})$, taking into account that if $(x:B) \in \Gamma$, then $(x:B[\diamond := \heartsuit]) \in \Gamma[\diamond := \heartsuit]$ and $B \leq_{\text{ABD}} B[\diamond := \heartsuit]$.

(iv) Let $\alpha_1, \dots, \alpha_n, \alpha'_1, \dots, \alpha'_{n'} \in \mathbb{A}^{\text{ABD}}$. We show by induction on the definition of \leq_{ABD} that if the following statements hold

$$A \leq_{\text{ABD}} A',$$

$$A = \alpha_1 \cap \dots \cap \alpha_n \cap (B_1 \rightarrow C_1) \cap \dots \cap (B_k \rightarrow C_k),$$

$$A' = \alpha'_1 \cap \dots \cap \alpha'_{n'} \cap (B'_1 \rightarrow C'_1) \cap \dots \cap (B'_{k'} \rightarrow C'_{k'}),$$

$$\Gamma, x:B_i \vdash^{\text{ABD}} M : C_i \quad \forall i \in \{1, \dots, k\},$$

then

$$\Gamma, x:B'_j \vdash^{\text{ABD}} M : C'_j \quad \forall j \in \{1, \dots, k'\}.$$

⁶ABD contains the axioms and rules of Definitions 13A.3 and 13A.18.

The only interesting case is when the applied rule is (\Diamond) , i.e. we have

$$\begin{aligned} A &\leq_{ABD} A[\Diamond := \heartsuit]; \\ A &= \alpha_1 \cap \cdots \cap \alpha_n \cap (B_1 \rightarrow C_1) \cap \cdots \cap (B_k \rightarrow C_k); \\ A' &= A[\Diamond := \heartsuit]. \end{aligned}$$

By hypothesis $\Gamma, x:B_i \vdash^{ABD} M : C_i$ for all $i \in \{1, \dots, k\}$, so we are done by (iii). ■

16B.21. THEOREM. (i) ABD is a TT that is not β -sound.

(ii) Nevertheless \mathcal{F}^{ABD} is a filter λ -model.

PROOF. (i) By definition ABD is a TT. We have $\Diamond \rightarrow \Diamond \leq_{ABD} \heartsuit \rightarrow \heartsuit$, but $\heartsuit \not\leq_{ABD} \Diamond$, so it is not β -sound.

(ii) To show that \mathcal{F}^{ABD} is a λ -model, it suffices, by Proposition 16B.8, to verify that $\Gamma \vdash^{ABD} \lambda x.M : A \rightarrow B \Rightarrow \Gamma, x:A \vdash^{ABD} M : B$. Suppose that $\Gamma \vdash^{ABD} \lambda x.M : A \rightarrow B$. By Lemma 14A.1(iii), there are $C_1, \dots, C_n, D_1, \dots, D_n$ such that

$$\begin{aligned} (C_1 \rightarrow D_1) \cap \cdots \cap (C_n \rightarrow D_n) &\leq_{ABD} A \rightarrow B \\ \forall i \in \{1, \dots, n\} \Gamma, x:C_i &\vdash^{ABD} M : D_i. \end{aligned}$$

So, we are done by Lemma 16B.20(iv). ■

For example the step function $\uparrow \Diamond \Rightarrow \uparrow \Diamond$ is not representable in \mathcal{F}^{ABD} .

16C. \mathcal{D}_∞ models as filter models

This section shows the connection between filter models and \mathcal{D}_∞ models, see [Scott \[1972\]](#) or [B\[1984\]](#). 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 \mathcal{D}_∞ -models in their full generality.

\mathcal{D}_∞ models

This subsection recalls some basic concepts of the standard \mathcal{D}_∞ construction and fixes some notations, see [Scott \[1972\]](#), [B\[1984\]](#), [Gierz, Hofmann, Keimel, Lawson, Mislove, and Scott \[1980\]](#). See also Definition 10C.1, where the construction is given in a more categorical setting.

In the following of this subsection we will recall main definitions and results presented in Section 18.2 of [B\[1984\]](#).

16C.1. DEFINITION. (i) Let \mathcal{D}_0 be an ω -algebraic complete lattice and

$$\langle i_0, j_0 \rangle$$

be an *embedding-projection* pair between \mathcal{D}_0 and $[\mathcal{D}_0 \rightarrow \mathcal{D}_0]$, i.e.

$$\begin{aligned} i_0 : \mathcal{D}_0 &\rightarrow [\mathcal{D}_0 \rightarrow \mathcal{D}_0] \\ j_0 : [\mathcal{D}_0 \rightarrow \mathcal{D}_0] &\rightarrow \mathcal{D}_0 \end{aligned}$$

are Scott continuous maps satisfying

$$\begin{aligned} i_0 \circ j_0 &\sqsubseteq \text{Id}_{[\mathcal{D}_0 \rightarrow \mathcal{D}_0]} \\ j_0 \circ i_0 &= \text{Id}_{\mathcal{D}_0}. \end{aligned}$$

(ii) Define a *tower* $\langle i_n, j_n \rangle : \mathcal{D}_n \rightarrow \mathcal{D}_{n+1}$ in the following way:

- $\mathcal{D}_{n+1} = [\mathcal{D}_n \rightarrow \mathcal{D}_n]$;
- $i_n(f) = i_{n-1} \circ f \circ j_{n-1}$ for any $f \in \mathcal{D}_n$;
- $j_n(g) = j_{n-1} \circ g \circ i_{n-1}$ for any $g \in \mathcal{D}_{n+1}$.

(iii) For $d \in \prod_{n \in \mathbb{N}} \mathcal{D}_n$ write $\textcolor{blue}{d}_n = d(n)$. The set \mathcal{D}_∞ is defined by

$$\mathcal{D}_\infty = \{d \in \prod_{n \in \mathbb{N}} \mathcal{D}_n \mid \forall n \in \mathbb{N}. d_n \in \mathcal{D}_n \& j_n(d_{n+1}) = d_n\},$$

NOTATION. d_n denotes the projection on \mathcal{D}_n , while $\textcolor{blue}{d}^n$ is an element of \mathcal{D}_n .

16C.2. DEFINITION. (i) The *ordering* on \mathcal{D}_∞ is given by

$$d \sqsubseteq e \iff \forall k \in \mathbb{N}. d_k \sqsubseteq e_k.$$

(ii) Let $\langle \Phi_{m\infty}, \Phi_{\infty m} \rangle$ denote the standard embedding-projection pair between \mathcal{D}_m and \mathcal{D}_∞ defined as follows. For $d^m \in \mathcal{D}_m$, $d \in \mathcal{D}_\infty$ write

$$\Phi_{mn}(d^m) \triangleq \begin{cases} j_n(\dots(j_{m-1}(d^m))), & \text{if } m > n; \\ d^m & \text{if } m = n; \\ i_{n-1}(\dots(i_m(d^m))), & \text{if } m < n; \end{cases}$$

and take

$$\begin{aligned} \Phi_{m\infty}(d^m) &\triangleq \langle \Phi_{m0}(d^m), \Phi_{m1}(d^m), \dots, \Phi_{mn}(d^m), \dots \rangle; \\ \Phi_{\infty m}(d) &\triangleq d_m = d(m). \end{aligned}$$

16C.3. LEMMA. $\bigsqcup X$ exists for all $X \subseteq \mathcal{D}_\infty$.

PROOF. Let $d^n = \bigsqcup \{x_n \mid x \in X\}$;
 $e^n = \bigsqcup \{\Phi_{mn}(d^m) \mid m \in \mathbb{N}\}$.

The set $\{\Phi_{mn}(d^m) \mid m \in \mathbb{N}\}$ is directed by monotonicity of i_m, j_m , and the fact that $i_m \circ j_m \sqsubseteq \text{Id}_{\mathcal{D}_{m+1}}$ and $j_m \circ i_m = \text{Id}_{\mathcal{D}_m}$. Define

$$\bigsqcup X = \lambda n \in \mathbb{N}. e^n.$$

Then, by continuity of j_n , we have that $\bigsqcup X \in \mathcal{D}_\infty$, if $X \neq \emptyset$. If $X = \emptyset$, then the continuity cannot be applied, but $\bigsqcup \emptyset = \lambda n \in \mathbb{N}. \perp_{\mathcal{D}_n}$ (use $i_n(\perp_{\mathcal{D}_n}) = \perp_{\mathcal{D}_{n+1}}$ so that $j_{n+1}(\perp_{\mathcal{D}_{n+1}}) = \perp_{\mathcal{D}_n}$). ■

16C.4. LEMMA. (i) $i_n \circ j_n \sqsubseteq \text{Id}_{[\mathcal{D}_n \rightarrow \mathcal{D}_n]}$, $j_n \circ i_n = \text{Id}_{\mathcal{D}_n}$.

(ii) $\forall p, q \in \mathcal{D}_n [i_{n+1}(p \mapsto q) = (i_n(p) \mapsto i_n(q)) \& j_{n+1}(i_n(p) \mapsto i_n(q)) = (p \mapsto q)]$.

(iii) $\Phi_{m\infty} \circ \Phi_{\infty m} \sqsubseteq \text{Id}_\infty$ and $\Phi_{\infty m} \circ \Phi_{m\infty} = \text{Id}_{\mathcal{D}_m}$.

(iv) $\forall e \in \mathcal{K}(\mathcal{D}_n) [i_n(e) \in \mathcal{K}(\mathcal{D}_{n+1})]$.

(v) $\forall e \in \mathcal{K}(\mathcal{D}_n) [m \geq n \Rightarrow \Phi_{nm}(e) \in \mathcal{K}(\mathcal{D}_m)]$.

(vi) $\forall e \in \mathcal{K}(\mathcal{D}_n) [\Phi_{n\infty}(e) \in \mathcal{K}(\mathcal{D}_\infty)]$.

(vii) If $n \leq k \leq m$ and $d \in \mathcal{D}_n$, $e \in \mathcal{D}_k$, then

$$\Phi_{nk}(d) \sqsubseteq e \Leftrightarrow \Phi_{nm}(d) \sqsubseteq \Phi_{km}(e) \Leftrightarrow \Phi_{n\infty}(d) \sqsubseteq \Phi_{k\infty}(e).$$

(viii) $\Phi_{mn} = \Phi_{\infty n} \circ \Phi_{m\infty}$.

(ix) $\forall a, b \in \mathcal{D}_n [(\Phi_{n\infty}(a) \mapsto \Phi_{n\infty}(b)) = \Phi_{n\infty} \circ (a \mapsto b) \circ \Phi_{\infty n}]$.

PROOF. (i) and (ii): By induction on n .

(iii) follows from (i).

(iv) and (v) and (vi): By Lemma 15B.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 15B.2(iii). ■

16C.5. LEMMA. $\bigsqcup_{n \in \mathbb{N}} \Phi_{n\infty} \circ \Phi_{\infty n} = \text{Id}_{\mathcal{D}_\infty}$.

PROOF. Since $\langle \Phi_{n\infty}, \Phi_{\infty n} \rangle$ is an embedding-projection pair, we have for all $n \in \mathbb{N}$ $\Phi_{n\infty} \circ \Phi_{\infty n} \sqsubseteq \text{Id}_{\mathcal{D}_\infty}$, hence for all $d \in \mathcal{D}_\infty$

$$\bigsqcup_{n \in \mathbb{N}} \Phi_{n\infty} \circ \Phi_{\infty n}(d) \sqsubseteq d.$$

On the other hand, for all $k \in \mathbb{N}$, we have

$$\begin{aligned} (\bigsqcup_{n \in \mathbb{N}} \Phi_{n\infty} \circ \Phi_{\infty n}(d))_k &\sqsupseteq (\Phi_{k\infty} \circ \Phi_{\infty k}(d))_k, && \text{because } (-)_k \text{ is monotone,} \\ &= \Phi_{\infty k}(d), && \text{as } \forall x \in \mathcal{D}_k. (\Phi_{k\infty}(x))_k = x, \\ &= d_k. \end{aligned}$$

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 \mathcal{D}_∞ and $[\mathcal{D}_\infty \rightarrow \mathcal{D}_\infty]$.

16C.6. LEMMA. (i) $d \in \mathcal{K}(\mathcal{D}_\infty) \Leftrightarrow \exists k, e \in \mathcal{K}(\mathcal{D}_k). \Phi_{k\infty}(e) = d$.

(ii) $f \in \mathcal{K}([\mathcal{D}_\infty \rightarrow \mathcal{D}_\infty]) \Leftrightarrow \exists k, g \in \mathcal{K}(\mathcal{D}_{k+1}). f = \Phi_{k\infty} \circ g \circ \Phi_{\infty k}$.

PROOF. (i) (\Rightarrow) Let $d \in \mathcal{K}(\mathcal{D}_\infty)$. Then $d = \bigsqcup_{n \in \mathbb{N}} \Phi_{n\infty}(d_n)$, by Lemma 16C.5. Since d is compact, there exists $k \in \mathbb{N}$ such that $d = \Phi_{k\infty}(d_k)$. Now we prove that $d_k \in \mathcal{K}(\mathcal{D}_k)$. Let $X \subseteq \mathcal{D}_k$ be directed. Then

$$\begin{aligned} d_k \sqsubseteq \bigsqcup X &\Rightarrow d \sqsubseteq \Phi_{k\infty}(\bigsqcup X) \\ &\Rightarrow d \sqsubseteq \bigsqcup \Phi_{k\infty}(X), && \text{since } \Phi_{k\infty} \text{ is continuous,} \\ &\Rightarrow \exists x \in X. d \sqsubseteq \Phi_{k\infty}(x), && \text{for some } k \text{ since } d \text{ is compact,} \\ &\Rightarrow \Phi_{\infty k}(d) \sqsubseteq \Phi_{\infty k} \circ \Phi_{k\infty}(x) \\ &\Rightarrow d_k \sqsubseteq x. \end{aligned}$$

This proves that $d_k \in \mathcal{K}(\mathcal{D}_k)$. (\Leftarrow) By Lemma 16C.4(vi).

(ii) (\Rightarrow) By Lemma 16C.5, 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

- $\exists k \in \mathbb{N}. f = \Phi_{k\infty} \circ (\Phi_{\infty k} \circ f \circ \Phi_{k\infty}) \circ \Phi_{\infty k}$,
- $(\Phi_{\infty k} \circ f \circ \Phi_{k\infty}) \in \mathcal{K}(\mathcal{D}_{k+1})$.

Put $g = (\Phi_{\infty k} \circ f \circ \Phi_{k\infty})$. (\Leftarrow) Easy. ■

16C.7. LEMMA. (i) $\forall x \in \mathcal{D}_\infty. x = \bigsqcup \{e \in \mathcal{K}(\mathcal{D}_\infty) \mid e \sqsubseteq x\}$.

(ii) $\mathcal{K}(\mathcal{D}_\infty)$ is countable.

(iii) $\mathcal{D}_\infty \in \mathbf{ALG}$.

PROOF. (i) Let $x \in \mathcal{D}_\infty$ and $U_x = \{e \in \mathcal{K}(\mathcal{D}_\infty) \mid e \sqsubseteq x\}$. Clearly, $\bigsqcup U_x \sqsubseteq x$. Now let $f = \bigsqcup U_x$ in order to show $x \sqsubseteq f$. By definition of sup in \mathcal{D}_∞ we have

$$f_n = \bigsqcup V(n, x), \quad \text{where } V(n, x) = \{e_n \in \mathcal{D}_n \mid e \in \mathcal{K}(\mathcal{D}_\infty) \& e \sqsubseteq x\}.$$

Since \mathcal{D}_n is algebraic, we have that

$$x_n = \bigsqcup W(n, x), \quad \text{where } W(n, x) = \{d \in \mathcal{K}(\mathcal{D}_n) \mid d \sqsubseteq x_n\}.$$

We will prove that $W(n, x) \subseteq V(n, x)$. Suppose $d \in W(n, x)$. Then $d \in \mathcal{K}(\mathcal{D}_n)$ 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) $e = \Phi_{n\infty}(d) \in \mathcal{K}(\mathcal{D}_\infty)$, by Lemma 16C.4(vi).

(3) $e = \Phi_{n\infty}(d) \sqsubseteq \Phi_{n\infty}(x_n) \sqsubseteq x$, by monotonicity of $\Phi_{n\infty}$ and Lemma 16C.4(iii).

Hence $d \in V(n, x)$. Now indeed $x \sqsubseteq f$, as clearly $x_n \sqsubseteq f_n$.

(ii) By Proposition 15A.8 one has $\mathcal{D}_n \in \mathbf{ALG}$ for each n . Hence $\mathcal{K}(\mathcal{D}_n)$ is countable for each n . But then also $\mathcal{K}(\mathcal{D}_\infty)$ is countable, by Lemma 16C.6(i).

(iii) By (i), and (ii). ■

16C.8. DEFINITION. Let

$$\begin{aligned} F_\infty : \mathcal{D}_\infty &\rightarrow [\mathcal{D}_\infty \rightarrow \mathcal{D}_\infty] \\ G_\infty : [\mathcal{D}_\infty \rightarrow \mathcal{D}_\infty] &\rightarrow \mathcal{D}_\infty \end{aligned}$$

be defined as follows

$$\begin{aligned} F_\infty(d) &\triangleq \bigsqcup_{n \in \mathbb{N}} (\Phi_{n\infty} \circ d_{n+1} \circ \Phi_{\infty n}); \\ G_\infty(f) &\triangleq \bigsqcup_{n \in \mathbb{N}} \Phi_{(n+1)\infty}(\Phi_{\infty n} \circ f \circ \Phi_{n\infty}). \end{aligned}$$

16C.9. THEOREM. ([Scott \[1972\]](#)) Let \mathcal{D}_∞ be constructed from $\mathcal{D}_0 \in \mathbf{ALG}$ and a projection pair i_0, j_0 . Then $\mathcal{D}_\infty \in \mathbf{ALG}$ and \mathcal{D}_∞ with F_∞, G_∞ is reflexive. Moreover,

$$F_\infty \circ G_\infty = \text{Id}_{[\mathcal{D}_\infty \rightarrow \mathcal{D}_\infty]} \quad \& \quad G_\infty \circ F_\infty = \text{Id}_{\mathcal{D}_\infty}.$$

It follows that \mathcal{D}_∞ is an extensional λ -model.

PROOF. For the proof that F_∞ and G_∞ are inverse of each other for Scott's model \mathcal{D}_∞ , with embedding-projection pair $i_0(d) = \lambda e. d$ and $j_0(f) = f(\perp)$, see [B\[1984\]](#), Theorem 18.2.16. For a proof in the general case see [Plotkin \[1982\]](#). By Proposition 16A.9 it follows that \mathcal{D}_∞ is an extensional λ -model. ■

16C.10. COROLLARY. Let \mathcal{D}_∞ be constructed from $\mathcal{D}_0 \in \mathbf{ALG}$ and a projection pair i_0, j_0 . Then $\langle \mathcal{D}_\infty, F_\infty, G_\infty \rangle$ is in \mathbf{NLS} .

PROOF. Immediate from the Theorem. ■

\mathcal{D}_∞ as a filter λ -model

In this subsection we follow Alessi [1991], Alessi, Dezani-Ciancaglini, and Honsell [2004]. Let \mathcal{D}_∞ be constructed from the triple $t = (\mathcal{D}_0, i_0, j_0)$. To emphasize the dependence on t we write $\mathcal{D}_\infty^t = \mathcal{D}_\infty^t$. From the previous corollary and Proposition 15B.13 it follows that $\mathcal{D}_\infty^t \cong \mathcal{F}^{\mathcal{K}(\mathcal{D}_\infty^t)}$. In this subsection we associate with $t = (\mathcal{D}_0, i_0, j_0)$ a family of intersection type theories $\text{CDHL}(t)$. These type theories are compatible and they can be considered as type structures, denoted also by $\text{CDHL}(t)$. We will show that

$$\mathcal{K}(\mathcal{D}_\infty^t) \cong \text{CDHL}(t).$$

Hence

$$\mathcal{D}_\infty^t \cong \mathcal{F}^{\text{CDHL}(t)}.$$

The name of the family of type theories $\text{CDHL}(t)$ is due to Coppo, Dezani-Ciancaglini, Honsell, and Longo [1984] where this construction was first discussed. Other relevant references are Coppo, Dezani-Ciancaglini, and Zacchi [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.

16C.11. DEFINITION. Let $t = (\mathcal{D}_0, i_0, j_0)$ be given. We define the family of type theories $\text{CDHL}(t)$ as follows.

- (i) The partial order \leq_0 on $\mathcal{K}(\mathcal{D}_0)$ is defined by

$$d \leq_0 e \iff d \sqsupseteq e, \quad \text{where } d, e \in \mathcal{K}(\mathcal{D}_0);$$

$$(ii) \quad \mathbb{T}^{\text{CDHL}(t)} \triangleq \mathcal{K}(\mathcal{D}_0) \mid \mathbb{T}^{\text{CDHL}(t)} \rightarrow \mathbb{T}^{\text{CDHL}(t)} \mid \mathbb{T}^{\text{CDHL}(t)} \cap \mathbb{T}^{\text{CDHL}(t)}.$$

(iii) Write \mathbb{U} for $\perp_{\mathcal{D}_0}$. Let $\text{CDHL}(t)$ be the smallest natural type theory⁷ on $\mathbb{T}^{\text{CDHL}(t)}$ that contains the following extra axiom and rules.

$$(\sqcup) \quad c \cap d =_{\text{CDHL}(t)} c \sqcup d,$$

$$(\leq_0) \quad \frac{c \leq_0 d}{c \leq_{\text{CDHL}(t)} d} \quad (i_0) \quad \frac{i_0(e) = (c_1 \mapsto d_1) \sqcup \cdots \sqcup (c_n \mapsto d_n)}{e =_{\text{CDHL}(t)} (c_1 \rightarrow d_1) \cap \cdots \cap (c_n \rightarrow d_n)},$$

where $c, d, e, c_1, d_1, \dots, c_n, d_n \in \mathcal{K}(\mathcal{D}_0)$ and \sqcup is the least upper bound for the ordering \sqsubseteq on $\mathcal{K}(\mathcal{D}_0)$. Note that for $c, d, e \in \mathcal{K}(\mathcal{D}_0)$ one has $(c \mapsto d), i_0(e) \in \mathcal{D}_1$.

(iv) Since $\text{CDHL}(t)$ is compatible, it can be considered as a type structure denoted by $[\text{CDHL}(t)]$. Note that $[\text{CDHL}(t)] \in \mathbf{NTS}^{\mathbb{U}}$.

The proof of the next lemma follows easily from Definition 16C.11.

16C.12. LEMMA. $c_1 \cap \cdots \cap c_n =_{\text{CDHL}(t)} c_1 \sqcup \cdots \sqcup c_n$. ■

The proof of $\mathcal{K}(\mathcal{D}_\infty^t) \cong \text{CDHL}(t)$ occupies 16C.13-16C.18. First we classify the types in $\mathbb{T}^{\text{CDHL}(t)}$ according to the maximal number of nested arrow occurrences they may contain.

⁷ $\text{CDHL}(t)$ contains the axiom and rules of Definitions 13A.3 and 13A.18.

16C.13. DEFINITION. (i) We define the map $\text{depth} : \mathbb{T}^{\text{CDHL}(t)} \rightarrow \mathbb{N}$ by:

$$\begin{aligned}\text{depth}(c) &\triangleq 0, & \text{for } c \in \mathcal{K}(\mathcal{D}_0); \\ \text{depth}(A \rightarrow B) &\triangleq \max\{\text{depth}(A), \text{depth}(B)\} + 1; \\ \text{depth}(A \cap B) &\triangleq \max\{\text{depth}(A), \text{depth}(B)\}.\end{aligned}$$

(ii) Let $\mathbb{T}_n^{\text{CDHL}(t)} \triangleq \{A \in \mathbb{T}^{\text{CDHL}(t)} \mid \text{depth}(A) \leq n\}$.

Notice that depth differs from the map dpt in Definition 1A.21.

We can associate to each type in $\mathbb{T}_n^{\text{CDHL}(t)}$ an element in \mathcal{D}_n : this will be crucial for defining the required isomorphism (see Definition 16C.20).

16C.14. DEFINITION. We define, for each $n \in \mathbb{N}$, a map $w_n : \mathbb{T}_n^{\text{CDHL}(t)} \rightarrow \mathcal{K}(\mathcal{D}_n^t)$ by a double induction on n and on the construction of types in $\mathbb{T}^{\text{CDHL}(t)}$:

$$\begin{aligned}w_n(c) &\triangleq \Phi_{0n}(c); \\ w_n(A \cap B) &\triangleq w_n(A) \sqcup w_n(B); \\ w_n(A \rightarrow B) &\triangleq (w_{n-1}(A) \mapsto w_{n-1}(B)).\end{aligned}$$

16C.15. REMARK. From Lemma 15A.6 we get $\forall A \in \mathbb{T}_n^{\text{CDHL}(t)}. w_n(A) \in \mathcal{K}(\mathcal{D}_n^t)$.

16C.16. LEMMA. Let $n \leq m$ and $A \in \mathbb{T}_n^{\text{CDHL}(t)}$. 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 $w_{n+1}(A) = i_n(w_n(A))$. Then the desired equality follows from the definition of the function Φ . The only interesting case is when $A \equiv B \rightarrow C$. We get

$$\begin{aligned}w_{n+1}(B \rightarrow C) &= w_n(B) \mapsto w_n(C), & \text{by definition,} \\ &= i_{n-1}(w_{n-1}(B)) \mapsto i_{n-1}(w_{n-1}(C)), & \text{by induction,} \\ &= i_n(w_{n-1}(B) \mapsto w_{n-1}(C)), & \text{by Lemma 16C.4(ii),} \\ &= i_n(w_n(B \rightarrow C)), & \text{by Definition 16C.14.} \blacksquare\end{aligned}$$

The maps w_n reverse the order between types.

16C.17. LEMMA. Let $\text{depth}(A \cap B) \leq n$. Then

$$A \leq_{\text{CDHL}(t)} B \Rightarrow w_n(B) \sqsubseteq w_n(A).$$

PROOF. The proof is by induction on the definition of $\leq_{\text{CDHL}(t)}$. We consider only two cases.

Case (\rightarrow). Let $A \leq_{\text{CDHL}(t)} B$ follows from $A \equiv C \rightarrow D$, $B \equiv E \rightarrow F$, $E \leq_{\text{CDHL}(t)} C$ and $D \leq_{\text{CDHL}(t)} F$. Then

$$\begin{aligned}E \leq_{\text{CDHL}(t)} C \& D \leq_{\text{CDHL}(t)} F \Rightarrow \\ \Rightarrow \quad w_{n-1}(C) &\sqsubseteq w_{n-1}(E) \& w_{n-1}(F) \sqsubseteq w_{n-1}(D), \\ &\text{by the induction hypothesis,} \\ \Rightarrow \quad w_{n-1}(E) &\mapsto w_{n-1}(F) \sqsubseteq w_{n-1}(C) \mapsto w_{n-1}(D) \\ \Rightarrow \quad w_n(B) &\sqsubseteq w_n(A).\end{aligned}$$

Case $e =_{\text{CDHL}(t)} (c_1 \rightarrow d_1) \cap \dots \cap (c_k \rightarrow d_k)$ follows from $i_0(e) = (c_1 \mapsto d_1) \sqcup \dots \sqcup (c_k \mapsto d_k)$.

We show by induction on $n \geq 1$ the following.

$$w_n(e) = (w_{n-1}(c_1) \mapsto w_{n-1}(d_1)) \sqcup \cdots \sqcup (w_{n-1}(c_k) \mapsto w_{n-1}(d_k)).$$

It trivially holds for $n = 1$, so let $n > 1$.

$$\begin{aligned} w_n(e) &= i_{n-1}(w_{n-1}(e)) \\ &= i_{n-1}((w_{n-2}(c_1) \mapsto w_{n-2}(d_1)) \sqcup \cdots \sqcup (w_{n-2}(c_k) \mapsto w_{n-2}(d_k))) \\ &= i_{n-1}(w_{n-2}(c_1) \mapsto w_{n-2}(d_1)) \sqcup \cdots \sqcup i_{n-1}(w_{n-2}(c_k) \mapsto w_{n-2}(d_k)) \\ &= (i_{n-2}(w_{n-2}(c_1)) \mapsto i_{n-2}(w_{n-2}(d_1))) \sqcup \cdots \\ &\quad \cdots \sqcup (i_{n-2}(w_{n-2}(c_k)) \mapsto i_{n-2}(w_{n-2}(d_k))) \\ &= (w_{n-1}(c_1) \mapsto w_{n-1}(d_1)) \sqcup \cdots \sqcup (w_{n-1}(c_k) \mapsto w_{n-1}(d_k)). \blacksquare \end{aligned}$$

Also the reverse implication of Lemma 16C.17 holds.

16C.18. LEMMA. *Let $\text{depth}(A \cap B) \leq n$. Then*

$$w_n(B) \sqsubseteq w_n(A) \Rightarrow A \leq_{\text{CDHL}(t)} B.$$

PROOF. By induction on $\text{depth}(A \cap B)$.

If $\text{depth}(A \cap B) = 0$ we have $A \equiv \bigcap_{i \in I} c_i$, $B = \bigcap_{j \in J} d_j$. Then

$$\begin{aligned} w_n(B) \sqsubseteq w_n(A) &\Rightarrow \bigsqcup_{j \in J} \Phi_{0n}(d_j) \sqsubseteq \bigsqcup_{i \in I} \Phi_{0n}(c_i) \\ &\Rightarrow \Phi_{n0}(\bigsqcup_{j \in J} \Phi_{0n}(d_j)) \sqsubseteq \Phi_{n0}(\bigsqcup_{i \in I} \Phi_{0n}(c_i)) \\ &\Rightarrow \bigsqcup_{j \in J} (\Phi_{n0} \circ \Phi_{0n})(d_j) \sqsubseteq \bigsqcup_{i \in I} (\Phi_{n0} \circ \Phi_{0n})(c_i) \\ &\Rightarrow \bigsqcup_{j \in J} d_j \sqsubseteq \bigsqcup_{i \in I} c_i \\ &\Rightarrow A \leq_{\text{CDHL}(t)} B. \end{aligned}$$

Otherwise, let

$$\begin{aligned} A &\equiv \left(\bigcap_{i \in I} c_i \right) \cap \left(\bigcap_{l \in L} (C_l \rightarrow D_l) \right), \\ B &\equiv \left(\bigcap_{h \in H} d_h \right) \cap \left(\bigcap_{m \in M} (E_m \rightarrow F_m) \right). \end{aligned}$$

By rule (i₀), we have that

$$c_i =_{\text{CDHL}(t)} \bigcap_{j \in J_i} (a_j \rightarrow b_j), \quad d_h =_{\text{CDHL}(t)} \bigcap_{k \in K_h} (e_k \rightarrow f_k),$$

where $a_j, b_j, e_k, f_k \in \mathcal{K}(\mathcal{D}_0)$. Now for all $n \geq 1$

$$\begin{aligned} w_n(c_i) &= \bigsqcup_{j \in J_i} (w_{n-1}(a_j) \mapsto w_{n-1}(b_j)), \\ w_n(d_h) &= \left(\bigsqcup_{k \in K_h} (w_{n-1}(e_k) \mapsto w_{n-1}(f_k)) \right), \end{aligned}$$

since by Lemma 16C.17 the function w_n identifies elements in the equivalence classes of $=_{\text{CDHL}(t)}$. Therefore

$$\bigsqcup_{h \in H} (\bigsqcup_{k \in K_h} w_{n-1}(e_k) \mapsto w_{n-1}(f_k)) \sqcup (\bigsqcup_{m \in M} w_{n-1}(E_m) \mapsto w_{n-1}(F_m)) \sqsubseteq \\ \bigsqcup_{i \in I} (\bigsqcup_{j \in J_i} w_{n-1}(a_j) \mapsto w_{n-1}(b_j)) \sqcup (\bigsqcup_{l \in L} w_{n-1}(C_l) \mapsto w_{n-1}(D_l)).$$

Hence for each $h \in H$, $k \in K_h$ we have

$$(w_{n-1}(e_k) \mapsto w_{n-1}(f_k)) \sqsubseteq \bigsqcup_{i \in I} (\bigsqcup_{j \in J_i} w_{n-1}(a_j) \mapsto w_{n-1}(b_j)) \sqcup \\ (\bigsqcup_{l \in L} w_{n-1}(C_l) \mapsto w_{n-1}(D_l)).$$

Suppose $w_{n-1}(f_k) \neq \perp_{\mathcal{D}_n}$. By Lemma 15A.7 there exist $I' \subseteq I$, $J'_i \subseteq J_i$, $L' \subseteq L$ such that

$$\bigsqcup_{i \in I'} (\bigsqcup_{j \in J'_i} w_{n-1}(a_j)) \sqcup (\bigsqcup_{l \in L'} w_{n-1}(C_l)) \sqsubseteq w_{n-1}(e_k), \\ \bigsqcup_{i \in I'} (\bigsqcup_{j \in J'_i} w_{n-1}(b_j)) \sqcup (\bigsqcup_{l \in L'} w_{n-1}(D_l)) \sqsupseteq w_{n-1}(f_k).$$

Notice that all types involved in the two above judgments have depths strictly less than $\text{depth}(A \cap B)$:

- (i) the depth of a_j, b_j, e_k, f_k is 0, since they are all atoms in $\mathcal{K}(\mathcal{D}_0)$;
- (ii) the depth of C_l, D_l is strictly smaller than the one of $A \cap B$, since they are subterms of an arrow in A .

Then by induction and by Lemma 16C.12 we obtain

$$e_k \leq_{\text{CDHL}(t)} \bigcap_{i \in I'} \left(\bigcap_{j \in J'_i} a_j \right) \cap \bigcap_{l \in L'} C_l, \\ f_k \geq_{\text{CDHL}(t)} \bigcap_{i \in I'} \left(\bigcap_{j \in J'_i} b_j \right) \cap \bigcap_{l \in L'} D_l.$$

Therefore, by (\rightarrow) and Proposition 13A.21, we have $A \leq_{\text{CDHL}(t)} e_k \rightarrow f_k$.

If $w_{n-1}(f_k) = \perp_{\mathcal{D}_n}$, then $w_{n-1}(f_k) = \Phi_{0n}(f_k)$ since $f_k \in \mathcal{K}(\mathcal{D}_0)$. This gives $f_k = \Phi_{n0} \circ \Phi_{0n}(f_k) = \Phi_{n0}(\perp_{\mathcal{D}_n}) = \perp_{\mathcal{D}_0}$ because $j_n(\perp_{\mathcal{D}_{n+1}}) = \perp_{\mathcal{D}_n}$. Since $f_k = \perp_{\mathcal{D}_0}$ implies $A \leq_{\text{CDHL}(t)} e_k \rightarrow f_k$ we are done.

In a similar way we can prove that $A \leq_{\text{CDHL}(t)} E_m \rightarrow F_m$, for any $m \in M$. Putting together these results we get $A \leq_{\text{CDHL}(t)} B$. ■

16C.19. PROPOSITION. $\langle \mathcal{K}(\mathcal{D}_\infty^t), \leq_\infty, \cap, \rightarrow_\infty, \top \rangle$ is a natural type structure, where \leq_∞ is the reverse order on $\mathcal{K}(\mathcal{D}_\infty^t)$, $a \rightarrow_\infty b = G_\infty(a \mapsto b)$, \cap is the least upper bound of \mathcal{D}_∞^t and $\top = \perp_{\mathcal{D}_\infty^t}$.

PROOF. By Lemma 15B.11. ■

We can now prove the isomorphism in **NTS**^U between $\mathcal{K}(\mathcal{D}_\infty^t)$ and $\text{CDHL}(t)$ seen as type structure, i.e. $[\text{CDHL}(t)]$.

16C.20. DEFINITION. For $A \in \mathbb{T}^{\text{CDHL}(t)}$ write

$$\textcolor{blue}{f}([A]) \triangleq \Phi_{r\infty}(w_r(A)),$$

where $r \geq \text{depth}(A)$.

16C.21. THEOREM. In $\mathbf{NTS}^\mathbb{U}$ one has $[\text{CDHL}(t)] \cong \mathcal{K}(\mathcal{D}_\infty^t)$ via f .

PROOF. First of all notice that f is well defined, in the sense the it does not depend on either the type chosen in $[A]$ or the depth r . In fact let $B, B' \in [A]$, and let $p \geq \text{depth}(B)$, $p' \geq \text{depth}(B')$. Fix any $q \geq p, p'$. Then we have

$$\begin{aligned} \Phi_{p\infty}(w_p(B)) &= \Phi_{q\infty}(w_q(B)), && \text{by Lemma 16C.16,} \\ &= \Phi_{q\infty}(w_q(B')), && \text{by Lemma 16C.17,} \\ &= \Phi_{p'\infty}(w_{p'}(B')), && \text{by Lemma 16C.16.} \end{aligned}$$

Write $f(A)$ for $f([A])$. Now f is injective by Lemma 16C.18 and monotone by Lemma 16C.17. From Lemma 15A.8(ii) we get immediately

$$\mathcal{K}(\mathcal{D}_{n+1}) = \{c_1 \mapsto d_1 \sqcup \cdots \sqcup c_n \mapsto d_n \mid c_i, d_i \in \mathcal{K}(\mathcal{D}_n)\}.$$

it is easily proved by induction on n that w_n is surjective on $\mathcal{K}(\mathcal{D}_n)$, hence f is surjective by Lemma 16C.6(i). The function f^{-1} is monotone by Lemma 16C.18. Taking into account that the order \leq_∞ on $\mathcal{K}(\mathcal{D}_\infty^t)$ is the reversed of \sqsubseteq of \mathcal{D}_∞^t and that $d \rightarrow_\infty e = G_\infty(d \mapsto e)$, we need to show

$$(5) \quad A \leq_{\text{CDHL}(t)} B \rightarrow C \Leftrightarrow f(A) \sqsupseteq G_\infty(f(B) \mapsto f(C))$$

In order to prove (5), let $r \geq \max\{\text{depth}(A), \text{depth}(B \rightarrow C)\}$ (in particular it follows $\text{depth}(B), \text{depth}(C) \leq r - 1$). We have

$$\begin{aligned} G_\infty(f(B) \mapsto f(C)) &= G_\infty(\Phi_{(r-1)\infty}(w_{r-1}(B)) \mapsto \Phi_{(r-1)\infty}(w_{r-1}(C))) \\ &= G_\infty(\Phi_{(r-1)\infty} \circ (w_{r-1}(B) \mapsto w_{r-1}(C)) \circ \Phi_{\infty(r-1)}), \\ &\quad \text{by Lemma 16C.4(viii),} \\ &= \Phi_{r\infty}(w_{r-1}(B) \mapsto w_{r-1}(C)), \quad \text{by Lemma 16C.6(iii),} \\ &= \Phi_{r\infty}(w_r(B \rightarrow C)), \quad \text{by definition of } w_r. \end{aligned}$$

Finally we have

$$\begin{aligned} A \leq_{\text{CDHL}(t)} B \rightarrow C &\Leftrightarrow w_r(A) \sqsupseteq w_r(B \rightarrow C), \\ &\quad \text{by Lemmas 16C.17 and 16C.18,} \\ &\Leftrightarrow \Phi_{r\infty}(w_r(A)) \sqsupseteq \Phi_{r\infty}(w_r(B \rightarrow C)), \\ &\quad \text{since } \Phi_{r\infty} \text{ is an embedding,} \\ &\Leftrightarrow \Phi_{r\infty}(w_r(A)) \sqsupseteq G_\infty(f(B) \mapsto f(C)), \quad \text{as above,} \\ &\Leftrightarrow f(A) \sqsupseteq G_\infty(f(B) \mapsto f(C)). \\ &\Leftrightarrow f(A) \leq_\infty f(B) \rightarrow_\infty f(C) \end{aligned}$$

So we have proved (5) and the proof is complete. ■

16C.22. THEOREM. $\mathcal{F}^{[\text{CDHL}(t)]} \cong \mathcal{D}_\infty^t$ in \mathbf{NLS} , via the map

$$\hat{f}(X) \triangleq \bigsqcup\{f(B) \mid B \in X\},$$

satisfying $\hat{f}(\uparrow A) = f(A)$.

PROOF. Let $f : [\text{CDHL}(t)] \rightarrow \mathcal{K}(\mathcal{D}_\infty^t)$ be the isomorphism in $\mathbf{NTS}^\mathbb{U}$. By Proposition 15B.9 we know that \mathcal{A} is a functor from $\mathbf{NTS}^\mathbb{U}$ to \mathbf{NLS} . Then

$$\mathcal{A}(f) : \mathcal{F}^{[\text{CDHL}(t)]} \rightarrow \mathcal{F}^{\mathcal{K}(\mathcal{D}_\infty^t)}$$

is an isomorphism in \mathbf{NLS} where $\mathcal{A}(f)(X) = \{B \mid \exists A \in X. f(A) \sqsubseteq B\}$.

By Proposition 15B.13 we have that

$$\tau : \mathcal{F}^{\mathcal{K}(\mathcal{D}_\infty^t)} \rightarrow \mathcal{D}_\infty^t$$

is an isomorphism in \mathbf{NLS} where $\tau(X) = \bigsqcup X$.

The composition of τ and $\mathcal{A}(f)$ is an isomorphism from $\mathcal{F}^{[\text{CDHL}(t)]}$ to \mathcal{D}_∞^t explicitly given by

$$\begin{aligned}\tau \circ \mathcal{A}(f)(X) &= \bigsqcup \{B \mid \exists A \in X. f(A) \sqsubseteq B\} \\ &= \bigsqcup \{f(A) \mid A \in X\} = \hat{f}(X).\blacksquare\end{aligned}$$

16C.23. COROLLARY. $\mathcal{F}^{[\text{CDHL}(t)]} \cong \mathcal{D}_\infty^t$ in \mathbf{NLS} .

PROOF. By Lemma 13D.7. ■

Specific models \mathcal{D}_∞ as filter models

In this subsection we specialize Theorem 16C.22 to \mathcal{D}_∞^t models constructed from specific triples $t = (\mathcal{D}_0, i_0, j_0)$, 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 $\text{init}(\mathcal{T}) = (\mathcal{D}_0, i_0, j_0)$ such that $\text{CDHL}(\text{init}(\mathcal{T})) = \mathcal{T}$. By Corollary 16C.23,

$$\mathcal{D}_\infty^{\text{init}(\mathcal{T})} \cong \mathcal{F}^{[\text{CDHL}(\text{init}(\mathcal{T}))]} = \mathcal{F}^{\mathcal{T}}.$$

We will write $\mathcal{D}_\infty^{\mathcal{T}} := \mathcal{D}_\infty^{\text{init}(\mathcal{T})}$, so that this reads smoothly as $\mathcal{D}_\infty^{\mathcal{T}} \cong \mathcal{F}^{\mathcal{T}}$.

16C.24. REMARK. (i) Remember the following type theoretic axioms.

(0 _{Scott})	(U → 0) = 0
(0 _{Park})	(0 → 0) = 0
(01)	0 ≤ 1
(1 → 0)	(1 → 0) = 0
(0 → 1)	(0 → 1) = 1
(I)	(1 → 1) ∩ (0 → 0) = 1

(ii) In Definition 13A.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 $(\rightarrow \cap)$, (U_{top}))

and $(U \rightarrow)$ and the rule (\rightarrow) .

\mathcal{T}	Atoms $A^\mathcal{T}$	Axioms of \mathcal{T}
Scott	{U, 0}	(0_{Scott})
Park	{U, 0}	(0_{Park})
CDZ	{U, 0, 1}	$(01), (1 \rightarrow 0), (0 \rightarrow 1)$
DHM	{U, 0, 1}	$(01), (0 \rightarrow 1), (0_{\text{Scott}})$
HR	{U, 0, 1}	$(01), (1 \rightarrow 0), (I)$

16C.25. DEFINITION. For each $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{DHM}, \text{HR}\}$, we associate a triple $\text{init}(\mathcal{T}) = (\mathcal{D}_0^\mathcal{T}, i_0^\mathcal{T}, j_0^\mathcal{T})$ defined as follows.

1. Define $\mathcal{D}_0^\mathcal{T}$ as either a two point chain or a three point chain depending on \mathcal{T} as follows.

\mathcal{T}	$\mathcal{D}_0^\mathcal{T}$
Scott, Park	$U \sqsubseteq 0$
CDZ, DHM, HR	$U \sqsubseteq 1 \sqsubseteq 0$

2. Define $i_0^\mathcal{T} : \mathcal{D}_0^\mathcal{T} \rightarrow [\mathcal{D}_0^\mathcal{T} \rightarrow \mathcal{D}_0^\mathcal{T}]$ as follows.

$$\begin{aligned} i_0^\mathcal{T}(U) &\triangleq U \mapsto U \text{ for any } \mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{DHM}, \text{HR}\} \\ i_0^\mathcal{T}(1) &\triangleq \begin{cases} 0 \mapsto 1 & \text{if } \mathcal{T} \in \{\text{CDZ}, \text{DHM}\} \\ (1 \mapsto 1) \sqcup (0 \mapsto 0) & \text{if } \mathcal{T} \in \{\text{HR}\} \end{cases} \\ i_0^\mathcal{T}(0) &\triangleq \begin{cases} U \mapsto 0 & \text{if } \mathcal{T} \in \{\text{Scott}, \text{DHM}\} \\ 0 \mapsto 0 & \text{if } \mathcal{T} \in \{\text{Park}\} \\ 1 \mapsto 0 & \text{if } \mathcal{T} \in \{\text{CDZ}, \text{HR}\} \end{cases} \end{aligned}$$

3. Then define $j_0^\mathcal{T} : [\mathcal{D}_0^\mathcal{T} \rightarrow \mathcal{D}_0^\mathcal{T}] \rightarrow \mathcal{D}_0^\mathcal{T}$ as follows.

$$j_0^\mathcal{T}(f) \triangleq \bigsqcup\{d \in \mathcal{D}_0^\mathcal{T} \mid i_0^\mathcal{T}(d) \sqsubseteq f\}, \quad \text{for } f \in [\mathcal{D}_0^\mathcal{T} \rightarrow \mathcal{D}_0^\mathcal{T}].$$

It is easy to prove that $\langle i_0^\mathcal{T}, j_0^\mathcal{T} \rangle$ is an embedding-projection pair from $\mathcal{D}_0^\mathcal{T}$ to $[\mathcal{D}_0^\mathcal{T} \rightarrow \mathcal{D}_0^\mathcal{T}]$, so we can build $\mathcal{D}_\infty^{\text{init}(\mathcal{T})}$ following the steps outlined in Definition 16C.1.

16C.26. LEMMA. Let $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{DHM}, \text{HR}\}$ and $c_1, \dots, c_n, d_1, \dots, d_n, e_1, \dots, e_k, f_1, \dots, f_k \in \mathcal{D}_0^\mathcal{T}$. Then

$$\begin{aligned} (e_1 \rightarrow f_1) \cap \dots \cap (e_k \rightarrow f_k) =_{\mathcal{T}} (c_1 \rightarrow d_1) \cap \dots \cap (c_n \rightarrow d_n) &\Leftrightarrow \\ (c_1 \mapsto d_1) \sqcup \dots \sqcup (c_n \mapsto d_n) = (e_1 \mapsto f_1) \sqcup \dots \sqcup (e_k \mapsto f_k). \end{aligned}$$

PROOF. It suffices to prove

$$(c \mapsto d) \sqsubseteq (e_1 \mapsto f_1) \sqcup \dots \sqcup (e_k \mapsto f_k) \Leftrightarrow (e_1 \rightarrow f_1) \cap \dots \cap (e_k \rightarrow f_k) \leq_{\mathcal{T}} (c \rightarrow d).$$

Now, $(c \mapsto d) \sqsubseteq (e_1 \mapsto f_1) \sqcup \dots \sqcup (e_k \mapsto f_k)$

$$\begin{aligned} &\Leftrightarrow \exists I \subseteq \{1, \dots, k\} [\sqcup_{i \in I} e_i \sqsubseteq c \& d \sqsubseteq \sqcup_{i \in I} f_i], \text{ by Lemma 15A.7,} \\ &\Leftrightarrow \exists i_1, \dots, i_p \in \{1, \dots, k\} [c \leq_{\mathcal{T}} e_{i_1} \cap \dots \cap e_{i_p} \& f_{i_1} \cap \dots \cap f_{i_p} \leq_{\mathcal{T}} d], \\ &\Leftrightarrow (e_1 \rightarrow f_1) \cap \dots \cap (e_k \rightarrow f_k) \leq_{\mathcal{T}} (c \rightarrow d), \text{ by } \beta\text{-soundness, } (\rightarrow) \text{ and } (\rightarrow \cap). \blacksquare \end{aligned}$$

16C.27. COROLLARY. *The definition of $i_0^{\mathcal{T}}$ is canonical. By this we mean that we could have given equivalently the following definition.*

$$i_o^{\mathcal{T}}(e) = (c_1 \mapsto d_1) \sqcup \cdots \sqcup (c_n \mapsto d_n) \Leftrightarrow e =_{\mathcal{T}} (c_1 \rightarrow d_1) \cap \cdots \cap (c_n \rightarrow d_n).$$

PROOF. Immediate, by the definition of $i_0^{\mathcal{T}}$, the axioms (U_{top}) and $(U \rightarrow)$, the special axioms $(0_{Scott}), (0_{Park}), (1 \rightarrow 0), (0 \rightarrow 1), (I)$ respectively, and the previous Lemma. ■

16C.28. PROPOSITION. *For $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{DHM}, \text{HR}\}$ one has*

$$\mathcal{T} = \text{CDHL}(\text{init}(\mathcal{T})).$$

PROOF. First of all, we have that $\mathbb{P}^{\mathcal{T}} = \mathbb{P}^{\text{CDHL}(\text{init}(\mathcal{T}))}$ because

$$\begin{aligned} \mathbb{A}^{\mathcal{T}} &= \mathcal{K}(\mathcal{D}_0^{\mathcal{T}}) && \text{by Definition 16C.25} \\ &= \mathbb{A}^{\text{CDHL}(\text{init}(\mathcal{T}))} && \text{by Definition 16C.11} \end{aligned}$$

since each $\mathcal{D}_0^{\mathcal{T}}$ only contains compact elements. It remains to show that

$$A \leq_{\text{CDHL}(\text{init}(\mathcal{T}))} B \Leftrightarrow A \leq_{\mathcal{T}} B.$$

As to (\Rightarrow) , this follows by induction on the generation of $\leq_{\text{CDHL}(t)}$ where $t = \text{init}(\mathcal{T})$. Now \mathcal{T} satisfies the axioms $(\rightarrow \cap), (U_{top})$ and is closed under rule (\rightarrow) , 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, $c \cap d =_{\text{CDHL}(t)} c \sqcup d$, with $c, d \in \mathcal{K}(\mathcal{D}_0^{\mathcal{T}}) = \mathcal{D}_0^{\mathcal{T}}$, 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, $c \leq_{\text{CDHL}(t)} d$ because $d \sqsubseteq c$, we have $d = U, c = 0$ or $d = c$. Then in all cases $c \leq_{\mathcal{T}} d$, by the axioms (U_{top}) and (01) .

Rule (i_0) where $i_0 = i_0^{\mathcal{T}}$. Suppose $e =_{\text{CDHL}(t)} (c_1 \rightarrow d_1) \cap \cdots \cap (c_n \rightarrow d_n)$, because

$$i_0^{\mathcal{T}}(e) = (c_1 \mapsto d_1) \sqcup \cdots \sqcup (c_n \mapsto d_n).$$

Then $e =_{\mathcal{T}} (c_1 \rightarrow d_1) \cap \cdots \cap (c_n \rightarrow d_n)$, by Corollary 16C.27.

As to (\Leftarrow) , the axioms and rules $(U_{top}), (U \rightarrow), (\rightarrow \cap)$ and (\rightarrow) hold for \mathcal{T} by definition. Moreover, all axioms extra of \mathcal{T} hold in $\text{CDHL}(\text{init}(\mathcal{T}))$, by Definition 16C.25 and the rules $(\leq_0), (i_0)$ of Definition 16C.11. ■

Now we can obtain the following result

16C.29. COROLLARY. *Let $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{DHM}, \text{HR}\}$. Then in the category **NLS** we have*

$$\mathcal{F}^{\mathcal{T}} \cong \mathcal{D}_{\infty}^{\mathcal{T}}.$$

PROOF. $\mathcal{F}^{\mathcal{T}} = \mathcal{F}^{\text{CDHL}(\text{init}(\mathcal{T}))}$, by Proposition 16C.28,
 $\cong \mathcal{D}_{\infty}^{\mathcal{T}}$, by Corollary 16C.23. ■

We will end this subsection by telling what is the interest of the various models \mathcal{D}_{∞}^t . In B[1984], Theorem 19.2.9, the following result is proved.

16C.30. THEOREM (Hyland and Wadsworth). *Let $t = (\mathcal{D}_0, i_0, j_0)$, where \mathcal{D}_0 is a cpo (or object of **ALG**) with at least two elements and*

$$\begin{aligned} i_0(d) &= \lambda e \in \mathcal{D}_0. d, && \text{for } d \in \mathcal{D}_0, \\ j_0(f) &= f(\perp_{\mathcal{D}_0}), && \text{for } f \in [\mathcal{D}_0 \rightarrow \mathcal{D}_0]. \end{aligned}$$

Then for $M, N \in \Lambda$ (untyped lambda terms) and $C[\cdot]$ ranging over contexts

$$\mathcal{D}_\infty^t \models M = N \Leftrightarrow \forall C[\cdot].(C[M] \text{ is solvable} \Leftrightarrow C[N] \text{ is solvable}).$$

In particular, the local structure of \mathcal{D}_∞^t (i.e. $\{M = N \mid \mathcal{D}_\infty^t \models M = N\}$) is independent of the initial \mathcal{D}_0 . ■

16C.31. COROLLARY. For t as in the theorem one has for closed terms M, N

$$\mathcal{D}_\infty^t \models M = N \Leftrightarrow \forall A \in \mathbb{T}^{\text{Scott}} [\vdash_{\cap}^{\text{Scott}} M : A \Leftrightarrow \vdash_{\cap}^{\text{Scott}} N : A].$$

PROOF. Let $M, N \in \Lambda^\phi$. Then

$$\begin{aligned} \mathcal{D}_\infty^t \models M = N &\Leftrightarrow \mathcal{D}_\infty^{\text{Scott}} \models M = N, \text{ by Theorem 16C.30,} \\ &\Leftrightarrow \mathcal{F}^{\text{Scott}} \models M = N, \\ &\quad \text{by Corollary 16C.29 and Proposition 16A.13,} \\ &\Leftrightarrow \forall A \in \mathbb{T}^{\text{Scott}} [\vdash_{\cap}^{\text{Scott}} M : A \Leftrightarrow \vdash_{\cap}^{\text{Scott}} N : A], \\ &\quad \text{by Theorem 16B.7. ■} \end{aligned}$$

The model $\mathcal{D}_\infty^{\text{Park}}$ has been introduced to contrast the following result, see B[1984], 19.3.6.

16C.32. THEOREM (Park). Let t be as in 16C.30. Then for the untyped λ -term

$$Y_{\text{Curry}} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$$

one has

$$[Y_{\text{Curry}}]^{\mathcal{D}_\infty^t} = Y_{\text{Tarski}},$$

where Y_{Tarski} is the least fixed-point combinator on \mathcal{D}_∞^t . ■

The model $\mathcal{D}_\infty^{\text{Park}}$ has been constructed to give Y_{Curry} a meaning different from Y_{Tarski} .

16C.33. THEOREM (Park). $[Y_{\text{Curry}}]^{\mathcal{D}_\infty^{\text{Park}}} \neq Y_{\text{Tarski}}$. ■

Now this model can be obtained as a simple filter model $\mathcal{D}_\infty^{\text{Park}} \cong \mathcal{F}^{\text{Park}}$ and therefore, by Corollary 16C.29, one has

$$[Y_{\text{Curry}}]^{\mathcal{F}^{\text{Park}}} \neq Y_{\text{Tarski}}.$$

Other domain equations

Results similar to Theorem 16C.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

$$\begin{aligned} \mathcal{D} &= [\mathcal{D} \rightarrow \mathcal{D}] \times A \\ \mathcal{D} &= [\mathcal{D} \rightarrow \mathcal{D}] + A \\ \mathcal{D} &= [\mathcal{D} \rightarrow_{\perp} \mathcal{D}] \times A \\ \mathcal{D} &= [\mathcal{D} \rightarrow_{\perp} \mathcal{D}] + A, \end{aligned}$$

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, Dezani-Ciancaglini, and Longo \[1983\]](#). Solutions of these equations will be discussed below. At least the following

result is worthwhile mentioning in this respect, see [Coppo, Dezani-Ciancaglini, Honsell, and Longo \[1984\]](#) for a proof.

16C.34. PROPOSITION. *The filter λ -model \mathcal{F}^{BCD} is isomorphic to $\langle \mathcal{D}, F, G \rangle$, where \mathcal{D} is the initial solution of the domain equation*

$$\mathcal{D} = [\mathcal{D} \rightarrow \mathcal{D}] \times \mathbb{P}(\mathbb{A}_\infty)$$

the pair $\langle F, G \rangle$ set up a Galois connection and G is the map that takes the minimal elements in the extensionality classes of functions. ■

16D. 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.

16D.1. DEFINITION. (i) The *order* of an untyped lambda term is

$$\text{order}(M) \triangleq \sup\{n \mid \exists N. M \rightarrow_\beta \lambda x_1 \cdots x_n. N\},$$

i.e. the upper-bound of the number of its initial abstractions modulo β -conversion. So $\text{order}(M) \in \mathbb{N} \cup \infty$.

(ii) A λ -model \mathcal{D} is *lazy* if

$$[\mathcal{D} \models M = N \& \text{order}(M) = k] \Rightarrow \text{order}(N) = k.$$

For example $\text{order}(\Omega)=0$, $\text{order}(\mathsf{K})=2$ and $\text{order}(\mathsf{YK})=\infty$.

16D.2. PROPOSITION. *Let $\mathcal{F}^\mathcal{T}$ be a filter λ -model. Then $\mathcal{F}^\mathcal{T}$ is lazy iff*

$$\forall \Gamma, A. [\Gamma \vdash_{\cap}^\mathcal{T} M : A \Leftrightarrow \Gamma \vdash_{\cap}^\mathcal{T} N : A] \Rightarrow \text{order}(M) = \text{order}(N),$$

i.e. if M and N have the same types, then they have the same order.

PROOF. By 16B.7(i). ■

A very simple type theory is AO, see Fig. 13A.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 $\mathcal{D} \in \mathbf{ALG}$ its lifting \mathcal{D}_\perp is defined as the domain obtained by adding a new bottom element.

16D.3. THEOREM ([Abramsky and Ong \[1993\]](#)). *Let $\mathcal{D}_\infty^{\text{lazy}}$ be the initial solution of the domain equation $\mathcal{D} \cong [\mathcal{D} \rightarrow \mathcal{D}]_\perp$ in \mathbf{ALG} . Then $\mathcal{D}_\infty^{\text{lazy}} \cong \mathcal{F}^{\text{AO}}$. ■*

The theory AO can also be used to prove the completeness of the so called *F-semantics*, see [Dezani-Ciancaglini and Margaria \[1986\]](#).

The λl -calculus.

Models of the λl -calculus are considered in [Honsell and Lenisa \[1993\], \[1999\]](#). Similar to Theorem 16C.21 one has the following.

16D.4. THEOREM (Honsell and Lenisa [1999]). *Let \mathcal{D}_∞^I be the inverse limit solutions of the domain equation $[\mathcal{D} \rightarrow_{\perp} \mathcal{D}] \cong \mathcal{D}$. Then $\mathcal{D}_\infty^I \cong \mathcal{F}^T$, for some proper type theory T with $\mathbb{A}^T = \mathcal{K}(\mathcal{D}_0)$.*

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 λ -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 Ω is easy, i.e. for any closed lambda term M the equation $\Omega = M$ is consistent. This fact was proved by a Church-Rosser argument by Mitschke, see his Mitschke [1976] or B[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

$$\mathcal{P}(\omega) \models \Omega = M,$$

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, Dezani-Ciancaglini, and Honsell [2001], using intersection types. For an arbitrary closed λ -term M we will build a filter model $\mathcal{F}^{\text{ADH}(M)}$ such that

$$\mathcal{F}^{\text{ADH}(M)} \models \Omega = M.$$

We first examine which types can be assigned to $\omega := \lambda x.xx$ and $\Omega := \omega\omega$.

16D.5. LEMMA. *Let T be a natural type theory that is β -sound.*

- (i) $\vdash_n^T \omega : A \rightarrow B \Leftrightarrow A \leq_T A \rightarrow B$.
- (ii) $\vdash_n^T \Omega : B \Leftrightarrow \exists A \in \mathbb{T}^T. \vdash_n^T \omega : A \leq_T (A \rightarrow B)$.

PROOF. (i) (\Leftarrow) Suppose $A \leq_T (A \rightarrow B)$. Then

$$\begin{aligned} &x:A \vdash_n^T x : (A \rightarrow B) \\ &x:A \vdash_n^T xx : B \\ &\vdash_n^T \lambda x.xx : (A \rightarrow B). \end{aligned}$$

(\Rightarrow) Suppose $\vdash_n^T \omega : (A \rightarrow B)$. If $B =_T U$, then $A \leq_T U =_T (A \rightarrow B)$, by axiom $(U \rightarrow)$. Otherwise, by Theorem 14A.9,

$$\begin{aligned} \vdash_n^T \lambda x.xx : (A \rightarrow B) &\Rightarrow x:A \vdash_n^T xx : B, \\ &\Rightarrow x:A \vdash_n^T x : C, \quad x:A \vdash_n^T x : (C \rightarrow B), \text{ for some } C, \\ &\Rightarrow A \leq_T (C \rightarrow B) \leq_T (A \rightarrow B), \quad \text{by } (\rightarrow). \end{aligned}$$

(ii) (\Leftarrow) Immediate. (\Rightarrow) If $B =_T U$, then $\vdash_n^T \omega : U \leq_T U \rightarrow B$. If $B \neq_T U$, then by Theorem 14A.9(ii) one has $\vdash_n^T \omega : (A \rightarrow B)$, $\vdash_n^T \omega : A$, for some A . By (i) one has $A \leq_T A \rightarrow B$.

We associate to each type the maximum number of nested arrows in the leftmost path.

16D.6. DEFINITION. Let \mathcal{T} be a type theory. For $A \in \mathbb{T}^{\mathcal{T}}$ its *type nesting*, notation $\square(A)$, is defined inductively on types as follows:

$$\begin{aligned}\square(A) &\triangleq 0 && \text{if } A \in \mathbb{A}^{\mathcal{T}}; \\ \square(A \rightarrow B) &\triangleq \square(A) + 1; \\ \square(A \cap B) &\triangleq \max\{\square(A), \square(B)\}.\end{aligned}$$

For η^U -sound and natural type theories Lemma 16D.5(ii) can be strengthened using type nesting. First we need the following lemma that shows that, any type A with $\square(A) \geq 1$ is equivalent to an intersection of arrows with the same type nesting.

16D.7. LEMMA. *Let \mathcal{T} be a η^U -sound and natural type theory. Then for all $A \in \mathbb{T}^{\mathcal{T}}$ with $\square(A) \geq 1$, there exist $C_1, \dots, C_m, D_1, \dots, D_m$ such that*

$$\begin{aligned}A &=_{\mathcal{T}} (C_1 \rightarrow D_1) \cap \dots \cap (C_m \rightarrow D_m); \\ \square(A) &= \square((C_1 \rightarrow D_1) \cap \dots \cap (C_m \rightarrow D_m)).\end{aligned}$$

PROOF. Every type A is an intersection of arrow types and atoms. Since \mathcal{T} is η^U -sound and natural, the atoms can be replaced by an intersection of arrows between atoms. As $\square(A) \geq 1$ this does not increase the type nesting. ■

16D.8. LEMMA. *Let \mathcal{T} be a natural type theory which is β and η^U -sound. Then*

$$\vdash_{\cap}^{\mathcal{T}} \Omega : B \Rightarrow \exists A \in \mathbb{T}^{\mathcal{T}} [\vdash_{\cap}^{\mathcal{T}} \omega : A \leq_{\mathcal{T}} A \rightarrow B \ \& \ \square(A) = 0].$$

PROOF. Let $\vdash_{\cap}^{\mathcal{T}} \Omega : B$. If $B =_{\mathcal{T}} U$ take $A \equiv U$. Otherwise, by Lemma 16D.5(ii), there exists $A \in \mathbb{T}^{\mathcal{T}}$ such that $\vdash_{\cap}^{\mathcal{T}} \omega : A$ and $A \leq_{\mathcal{T}} A \rightarrow B$. We show by course of value induction on $n = \square(A)$ that we can take an alternative A' with $\square(A') = 0$. If $n = 0$ we are done, so suppose $n \geq 1$. By Lemma 16D.7, we may assume that A is of the form $A \equiv (C_1 \rightarrow D_1) \cap \dots \cap (C_m \rightarrow D_m)$. Now $A \leq_{\mathcal{T}} A \rightarrow B$, hence $A \leq_{\mathcal{T}} C_{i_1} \cap \dots \cap C_{i_p}$ and $D_{i_1} \cap \dots \cap D_{i_p} \leq_{\mathcal{T}} B$, with $p > 0$ and $1 \leq i_1, \dots, i_p \leq m$, since \mathcal{T} is β -sound. Hence,

$$\begin{aligned}\vdash_{\cap}^{\mathcal{T}} \omega : A &\Rightarrow \vdash_{\cap}^{\mathcal{T}} \omega : (C_{i_k} \rightarrow D_{i_k}), 1 \leq k \leq p, \\ &\Rightarrow C_{i_k} \leq_{\mathcal{T}} (C_{i_k} \rightarrow D_{i_k}), \text{ by 16D.5(i),} \\ &\Rightarrow C_{i_1} \cap \dots \cap C_{i_p} \leq_{\mathcal{T}} (C_{i_1} \rightarrow D_{i_1}) \cap \dots \cap (C_{i_p} \rightarrow D_{i_p}) \\ &\leq_{\mathcal{T}} C_{i_1} \cap \dots \cap C_{i_p} \rightarrow D_{i_1} \cap \dots \cap D_{i_p}, \text{ since } \mathcal{T} \text{ is natural,} \\ &\leq_{\mathcal{T}} (C_{i_1} \cap \dots \cap C_{i_p} \rightarrow B), \text{ as } D_{i_1} \cap \dots \cap D_{i_p} \leq_{\mathcal{T}} B.\end{aligned}$$

Now take $A' \equiv C_{i_1} \cap \dots \cap C_{i_p}$. Then $\square(A') < n$ and we are done by the IH. ■

Now let $M \in \Lambda^o$. We will build the desired model satisfying $\models \Omega = M$ by taking the union of a countable sequence of type theories $\text{ADH}_n(M)$ defined in a suitable way to force the final interpretation of M to coincide with the interpretation of Ω . In the following $\langle \cdot, \cdot \rangle$ denotes any bijection between $\mathbb{N} \times \mathbb{N}$ and \mathbb{N} .

16D.9. DEFINITION. (i) Define the following increasing sequence of intersection type theories $\text{ADH}_n(M)$ by induction on $n \in \mathbb{N}$, specifying the atoms, axioms and rules.

- $\mathbb{A}^{\text{ADH}_0(M)} = \{U, 0\}$;
- $\text{ADH}_0(M) = \text{Scott}$, the smallest natural type theory that contains (0_{Scott}) ;
- $\mathbb{A}^{\text{ADH}_{n+1}(M)} = \mathbb{A}^{\text{ADH}_n(M)} \cup \{\xi_{\langle n, m \rangle} \mid m \in \mathbb{N}\}$;

- $\text{ADH}_{n+1}(M)$ is the smallest natural type theory that contains $\text{ADH}_n(M)$ and the following infinite set of axioms

$$\xi_{\langle n, m \rangle} = (\xi_{\langle n, m \rangle} \rightarrow W_{\langle n, m \rangle}) \text{ for } m \in \mathbb{N},$$

where $\langle W_{\langle n, m \rangle} \rangle_{m \in \mathbb{N}}$ is any enumeration of the countably infinite set

$$\{A \mid \vdash_{\cap}^{\text{ADH}_n(M)} M : A\}.$$

(ii) We define $\text{ADH}(M)$ as follows:

$$\mathbb{A}^{\text{ADH}(M)} \triangleq \bigcup_{n \in \mathbb{N}} \mathbb{A}^{\text{ADH}_n(M)}; \quad \text{ADH}(M) \triangleq \bigcup_{n \in \mathbb{N}} \text{ADH}_n(M).$$

16D.10. PROPOSITION. $\text{ADH}(M)$ is a β, η^U -sound natural type theory.

PROOF. It is immediate to check that $\text{ADH}(M)$ is β and η^U -sound: they hold for $\text{ADH}_0(M)$, and not adding \leq between arrows β is preserved, while each fresh constant being equated to an arrow assures η . By construction all the $\text{ADH}_n(M)$ are natural type theories. The validity of rule (\rightarrow) in $\text{ADH}(M)$ follows by a ‘compactness’ argument: if $A' \leq_{\text{ADH}(M)} A$ and $B \leq_{\text{ADH}(M)} B'$, then $A' \leq_{\text{ADH}_n(M)} A$ and $B \leq_{\text{ADH}_m(M)} B'$; but then we have $(A \rightarrow B) \leq_{\text{ADH}_{\max\{n, m\}}(M)} (A' \rightarrow B')$ and hence $(A \rightarrow B) \leq_{\text{ADH}(M)} (A' \rightarrow B')$. Similarly one verifies that $(\rightarrow \cap), (U_{\text{top}})$, and $(U \rightarrow)$ all hold in $\text{ADH}(M)$. Therefore the type theory $\text{ADH}(M)$ is natural. ■

16D.11. THEOREM. $\mathcal{F}^{\text{ADH}(M)}$ is an extensional filter λ -model.

PROOF. By Propositions 16D.10 and 16B.11. ■

We now need to show that some types cannot be deduced for ω .

16D.12. LEMMA. $\not\vdash_{\cap}^{\text{ADH}(M)} \omega : 0$ and $\not\vdash_{\cap}^{\text{ADH}(M)} \omega : (0 \rightarrow 0) \rightarrow 0 \rightarrow 0$.

PROOF. Define the set $\mathcal{E}_U \subseteq \mathbb{T}(\mathbb{A}^{\text{ADH}(M)})$ as the minimal set such that:

$$\begin{aligned} U &\in \mathcal{E}_U; \\ A \in \mathbb{T}^{\text{ADH}(M)}, B \in \mathcal{E}_U &\Rightarrow (A \rightarrow B) \in \mathcal{E}_U; \\ A, B \in \mathcal{E}_U &\Rightarrow (A \cap B) \in \mathcal{E}_U; \\ W_i \in \mathcal{E}_U &\Rightarrow \xi_i \in \mathcal{E}_U. \end{aligned}$$

Claim: $A \in \mathcal{E}_U \Leftrightarrow A =_{\text{ADH}(M)} U$.

(\Rightarrow) By induction on the definition of \mathcal{E}_U , using axiom $(U \rightarrow)$.

(\Leftarrow) By induction on $\leq_{\text{ADH}(M)}$ it follows that

$$\mathcal{E}_U \ni B \leq_{\text{ADH}(M)} A \Rightarrow A \in \mathcal{E}_U.$$

Hence if $A =_{\text{ADH}(M)} U$, one has $\mathcal{E}_U \ni U \leq_{\text{ADH}(M)} A$ and thus $A \in \mathcal{E}_U$.

As $0 \notin \mathcal{E}_U$, it follows by the claim that

$$(6) \quad 0 \neq_{\text{ADH}(M)} U.$$

Similarly one has $0 \rightarrow 0 \notin \mathcal{E}_U$, hence $0 \rightarrow 0 \neq_{\text{ADH}(M)} U$. Suppose towards a contradiction that $\vdash_{\cap}^{\text{ADH}(M)} \omega : 0$. Then $\vdash_{\cap}^{\text{ADH}(M)} \omega : U \rightarrow 0$, by (0_{Scott}) . By Lemma 16D.5(i) we get $U \leq_{\text{ADH}(M)} (U \rightarrow 0) =_{\text{ADH}(M)} 0 \leq_{\text{ADH}(M)} U$, i.e. $U =_{\text{ADH}(M)} 0$, contradicting (6).

Similarly from $\vdash_{\cap}^{\text{ADH}(M)} \omega : (0 \rightarrow 0) \rightarrow 0 \rightarrow 0$, by Lemma 16D.5(i), we get $0 \rightarrow 0 \leq_{\text{ADH}(M)} (0 \rightarrow 0) \rightarrow (0 \rightarrow 0)$, which implies $0 \rightarrow 0 \leq_{\text{ADH}(M)} 0 \leq_{\text{ADH}(M)} U \rightarrow 0$, by β -soundness and (0_{Scott}) . Therefore $U =_{\text{ADH}(M)} 0$, contradicting (6). ■

We finally are able to prove the main theorem.

16D.13. THEOREM. *Let $M \in \Lambda^\ell$. Then $\mathcal{F}^{\text{ADH}(M)}$ is a non-trivial extensional λ -model such that $\mathcal{F}^{\text{ADH}(M)} \models M = \Omega$.*

PROOF. The model is non-trivial since clearly $\vdash_{\cap}^{\text{ADH}(M)} I : (0 \rightarrow 0) \rightarrow 0 \rightarrow 0$ and by Lemma 16D.12 $\nvdash_{\cap}^{\text{ADH}(M)} \omega : (0 \rightarrow 0) \rightarrow 0 \rightarrow 0$, hence $\mathcal{F}^{\text{ADH}(M)} \not\models I = \omega$.

We must show that $\llbracket M \rrbracket = \llbracket \Omega \rrbracket$. Suppose that $W \in \llbracket M \rrbracket$. Then

$$\begin{aligned} \vdash_{\cap}^{\text{ADH}(M)} M : W &\Rightarrow \vdash_{\cap}^{\text{ADH}_n(M)} M : W, && \text{for some } n, \\ &\Rightarrow \xi_i =_{\text{ADH}_{n+1}(M)} (\xi_i \rightarrow W), && \text{for some } i, \\ &\Rightarrow \vdash_{\cap}^{\text{ADH}(M)} \Omega : W, \\ &\Rightarrow W \in \llbracket \Omega \rrbracket. \end{aligned}$$

This proves $\llbracket M \rrbracket \subseteq \llbracket \Omega \rrbracket$.

Now suppose $B \in \llbracket \Omega \rrbracket$, i.e. $\vdash_{\cap}^{\text{ADH}(M)} \Omega : B$. Then by Lemma 16D.8 there exists A such that $\square(A) = 0$ and $\vdash_{\cap}^{\text{ADH}(M)} \omega : A \leq_{\text{ADH}(M)} A \rightarrow B$. Let $A \equiv \bigcap_{i \in I} \psi_i$, with $\psi_i \in \mathbb{A} = \{U, 0, \xi_0, \dots\}$. By Lemma 16D.12 $\psi_i \neq_{\text{ADH}(M)} 0$. Hence it follows that $A =_{\text{ADH}(M)} U$ or $A =_{\text{ADH}(M)} \bigcap_{j \in J} (\xi_j)$, for some finite $J \subseteq \mathbb{N}$. Since $U =_{\text{ADH}(M)} (U \rightarrow U)$ and $\xi_j =_{\text{ADH}(M)} (\xi_j \rightarrow W)$ we get $A =_{\text{ADH}(M)} (U \rightarrow U)$ or $A =_{\text{ADH}(M)} \bigcap_{j \in J} (\xi_j \rightarrow W_j)$. Since $A \leq_{\text{ADH}(M)} A \rightarrow B$ it follows by β -soundness that in the first case $U \leq_{\text{ADH}(M)} B$ or in the second case $\bigcap_{j \in L} W_j \leq_{\text{ADH}(M)} B$, for some $L \subseteq J$. Since each W_j is in $\llbracket M \rrbracket$, we have in both cases $B \in \llbracket M \rrbracket$. This shows $\llbracket \Omega \rrbracket \subseteq \llbracket M \rrbracket$ and we are done. ■

Graph models as filter models

For a set X we use $\mathcal{P}(X)$ to denote the power-set of X and $\mathcal{P}_{fin}(X)$ to denote the set of finite subsets of X .

Engeler's Model

16D.14. DEFINITION ([Engeler \[1981\]](#)). Let \mathbb{A}_∞ be a countable set of atoms.

- (i) Define \mathbf{Em} as the least set satisfying $\mathbf{Em} = \mathbb{A}_\infty \cup (\mathcal{P}_{fin}(\mathbf{Em}) \times \mathbf{Em})$.
- (ii) Define $F_{\mathbf{Em}} : \mathcal{P}(\mathbf{Em}) \rightarrow [\mathcal{P}(\mathbf{Em}) \rightarrow \mathcal{P}(\mathbf{Em})]$
 $G_{\mathbf{Em}} : [\mathcal{P}(\mathbf{Em}) \rightarrow \mathcal{P}(\mathbf{Em})] \rightarrow \mathcal{P}(\mathbf{Em})$
by

$$\begin{aligned} F_{\mathbf{Em}}(X) &= \bigsqcup \{u \mapsto e \mid \langle u, e \rangle \in X\} \\ G_{\mathbf{Em}}(f) &= \{\langle u, e \rangle \mid e \in f(u)\}. \end{aligned}$$

16D.15. THEOREM. $F_{\mathbf{Em}}, G_{\mathbf{Em}}$ satisfy $F_{\mathbf{Em}} \circ G_{\mathbf{Em}} = \text{Id}$, making $\mathcal{P}(\mathbf{Em})$ a λ -model.

PROOF. See [Engeler \[1981\]](#). ■

16D.16. THEOREM. Let the type theory Engeler be as defined in Definition 13A.14. Then $\mathcal{P}(\text{Em}) \cong \mathcal{F}^{\text{Engeler}}$ are isomorphic as λ -structures (λ -models). ■

PROOF. See Plotkin [1993]. ■

Scott's $\mathcal{P}(\omega)$ model

Following the original notation by Scott, we use ω to denote the set of natural numbers.

16D.17. NOTATION. (i) Let $\mathbb{A}nm.\langle n, m \rangle : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ be the polynomially defined bijection

$$\langle n, m \rangle \triangleq \frac{1}{2}(n+m)(n+m+1) + m.$$

(ii) Let $\mathbb{A}n.e_n : \mathbb{N} \rightarrow \mathcal{P}_{\text{fin}}(\omega)$ be a bijection, e.g. the one defined by

$$e_n \triangleq \{k_0, \dots, k_{m-1}\} \text{ with } k_0 < k_1 < \dots < k_{m-1} \Leftrightarrow n = \sum_{i < m} 2^{k_i}.$$

16D.18. DEFINITION. [Scott [1972]] Let $\gamma : \mathcal{P}_{\text{fin}}(\omega) \times \omega \rightarrow \omega$ be the bijection defined by

$$\gamma(e_n, m) \triangleq \langle n, m \rangle.$$

(i) Define $F_\omega : [\mathcal{P}\omega \rightarrow \mathcal{P}\omega] \rightarrow [\mathcal{P}\omega \rightarrow \mathcal{P}\omega]$ by

$$F_\omega(X) \triangleq \bigsqcup \{u \mapsto i \mid \gamma(u, i) \in X\}.$$

(ii) $G_\omega : [\mathcal{P}\omega \rightarrow \mathcal{P}\omega] \rightarrow \mathcal{P}\omega$ by

$$G_\omega(f) \triangleq \{\gamma(u, i) \mid i \in f(u)\}$$

for all $f \in [\mathcal{P}\omega \rightarrow \mathcal{P}\omega]$.

16D.19. PROPOSITION. Define for $X, Y \in \mathcal{P}\omega$ the application

$$X \cdot_{\mathcal{P}\omega} Y \triangleq \{m \mid \exists e_n \subseteq Y \ \langle n, m \rangle \in X\}.$$

Then $F_\omega(X)(Y) = X \cdot_{\mathcal{P}\omega} Y$ is a (more common) equivalent definition for F_ω .

PROOF. Do exercise 16E.5. ■

16D.20. THEOREM. $\mathcal{P}\omega$ is a λ -model via F_ω, G_ω .

PROOF. See Scott [1972]. ■

16D.21. THEOREM. Define

$$\begin{aligned} \mathbb{A}^{\text{Scott-}\omega} &\triangleq \omega \\ \text{Scott-}\omega &\triangleq \text{Engeler} \cup \{\bigcap_{k \in e} (k \rightarrow n) = \gamma(e, n) \mid e \in \mathcal{P}_{\text{fin}}(\omega), n \in \omega\}. \end{aligned}$$

Then $\mathcal{P}\omega \cong \mathcal{F}^{\text{Scott-}\omega}$ are isomorphic as natural λ -structures (λ -models).

PROOF. See Alessi [1991]. ■

Plotkin's Model

16D.22. DEFINITION (Plotkin [1993]). Let 0 be an atom.

(i) Define Pm as the least set such that

$$\text{Pm} = \{0\} \cup (\mathcal{P}_{fin}(\text{Pm}) \times \mathcal{P}_{fin}(\text{Pm})).$$

(ii) Define $F_{\text{Pm}} : \mathcal{P}(\text{Pm}) \rightarrow [\mathcal{P}(\text{Pm}) \rightarrow \mathcal{P}(\text{Pm})]$ by
 $G_{\text{Pm}} : [\mathcal{P}(\text{Pm}) \rightarrow \mathcal{P}(\text{Pm})] \rightarrow \mathcal{P}(\text{Pm})$

$$\begin{aligned} F_{\text{Pm}}(X) &\triangleq \bigsqcup\{u \mapsto v \mid \langle u, v \rangle \in X\} \\ G_{\text{Pm}}(f) &\triangleq \{\langle u, v \rangle \mid v \subseteq f(u)\}. \end{aligned}$$

16D.23. THEOREM. $F_{\text{Pm}}, G_{\text{Pm}}$ satisfy $F_{\text{Pm}} \circ G_{\text{Pm}} = \text{Id}$, turning $\mathcal{P}(\text{Pm})$ into a λ -model.

PROOF. See Plotkin [1993]. ■

16D.24. THEOREM. Let the type theory Plotkin be as defined in Definition 13A.14. Then $\mathcal{P}(\text{Pm}) \cong \mathcal{F}^{\text{Plotkin}}$ are isomorphic as natural λ -structures (λ -models).

PROOF. See Plotkin [1993]. ■

16E. Exercises

16E.1. Check the following equalities:

$$\begin{aligned} \llbracket \lambda x.y \rrbracket_{\rho_0}^{\mathcal{F}^{\text{CDV}}} &= \emptyset; \\ \llbracket \lambda x.y \rrbracket_{\rho_1}^{\mathcal{F}^{\text{HL}}} &= \uparrow 0; \\ \llbracket \lambda x.y \rrbracket_{\rho_0}^{\mathcal{F}^{\text{AO}}} &= \uparrow (\text{U} \rightarrow \text{U}); \\ \llbracket \lambda x.y \rrbracket_{\rho_0}^{\mathcal{F}^{\text{EHR}}} &= \uparrow \text{V}, \end{aligned}$$

where $\rho_0(y) = \uparrow \emptyset$ and $\rho_1(y) = \uparrow 0$.

- 16E.2. (i) Define $K^\infty \triangleq YK$. 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 13E.4.]

16E.3. Prove using the results of Exercise 16E.2 that $\llbracket K^\infty \rrbracket_\rho^{\mathcal{F}^{\text{AO}}} = \mathcal{F}^{\text{AO}}$.

16E.4. Define $t : \mathbb{T}(\{0, 1, \text{U}\}) \rightarrow \mathbb{T}(\{0, 1, \text{U}\})$ inductively:

$$\begin{aligned} t(\alpha) &\triangleq \alpha, & \text{where } \alpha \in \{\text{U}, 0\}; \\ t(1) &\triangleq \text{U}; \\ t(A \rightarrow B) &\triangleq A \rightarrow t(B); \\ t(A \cap B) &\triangleq t(A) \cap t(B). \end{aligned}$$

The intersection type theory Alessi is axiomatized by rule (\rightarrow) and axioms $(\rightarrow \cap)$, (U_{top}) , $(\text{U} \rightarrow)$, (01) , $(1 \rightarrow 0)$, $(0 \rightarrow 1)$, see Fig. 13A.14, and (t) , $(t \rightarrow)$, where

$$\begin{aligned} (t) \quad &A \leq t(A); \\ (t \rightarrow) \quad &A \rightarrow B \leq t(A) \rightarrow t(B). \end{aligned}$$

If $\Gamma = \{x_1:A_1, \dots, x_n:A_n\}$, then write $\mathbf{t}(\Gamma) \triangleq \{x_1:\mathbf{t}(A_1), \dots, x_n:\mathbf{t}(A_n)\}$. Show the following.

- (i) The map \mathbf{t} is idempotent, i.e. $\mathbf{t}(\mathbf{t}(A)) = \mathbf{t}(A)$.
- (ii) $A \rightarrow \mathbf{t}(B) =_{\text{Alessi}} \mathbf{t}(A) \rightarrow \mathbf{t}(B)$.
- (iii) $A \leq_{\text{Alessi}} B \Rightarrow \mathbf{t}(A) \leq_{\text{Alessi}} \mathbf{t}(B)$.
- (iv) $\Gamma \vdash_{\text{Alessi}} M : A \Rightarrow \mathbf{t}(\Gamma) \vdash_{\text{Alessi}} M : \mathbf{t}(A)$.
- (v) $\Gamma, \Gamma' \vdash_{\text{Alessi}} M : A \Rightarrow \Gamma, \mathbf{t}(\Gamma') \vdash_{\text{Alessi}} M : \mathbf{t}(A)$.
- (vi) $\forall i \in I. \Gamma, x:A_i \vdash_{\text{Alessi}} M : B_i \& \bigcap_{i \in I} (A_i \rightarrow B_i) \leq_{\text{Alessi}} C \rightarrow D \Rightarrow \Gamma, x:C \vdash_{\text{Alessi}} M : D$.
- (vii) Alessi is not β -sound. [Hint. $1 \rightarrow 0 \leq_{\text{Alessi}} U \rightarrow 0$.]
- (viii) $\mathcal{F}^{\text{Alessi}}$ is a filter λ -model. [Hint. Modify Theorem 16B.21, and Lemma 16B.20(vi).]
- (ix) The step function $\uparrow 1 \Rightarrow \uparrow 0$ is not representable in $\mathcal{F}^{\text{Alessi}}$.

Actually, $\mathcal{F}^{\text{Alessi}}$ is the inverse limit solution of the domain equation $\mathcal{D} \simeq [\mathcal{D} \rightarrow \mathcal{D}]$ taken in the category of \mathbf{t} -lattices, whose objects are ω -algebraic lattices \mathcal{D} endowed with a finitary additive projection $\delta : \mathcal{D} \rightarrow \mathcal{D}$ and whose morphisms $f : (\mathcal{D}, \delta) \rightarrow (\mathcal{D}', \delta')$ are continuous functions such that $\delta' \circ f \sqsubseteq f \circ \delta$. See Alessi [1993], Alessi, Barbanera, and Dezani-Ciancaglini [2004] for details.

16E.5. Show Proposition 16D.19.

16E.6. Show Theorem 16D.16 using the mapping $f : \mathcal{P}_{fin}(\mathsf{Em}) \rightarrow \mathbb{T}^{\text{Engeler}}$ defined as

$$\begin{aligned} f(\emptyset) &\triangleq U, \\ f(\{a\}) &\triangleq a, \\ f(u \cup \{e\}) &\triangleq f(u) \cap f(\{e\}), \\ f(\{\langle u, e \rangle\}) &\triangleq f(u) \rightarrow f(\{e\}), \end{aligned}$$

where $u \in \mathcal{P}_{fin}(\mathsf{Em})$, $e \in \mathsf{Em}$, $a \in \mathbb{A}_\infty$.

16E.7. Show Theorem 16D.21 using the mapping $f : \mathcal{P}_\omega \rightarrow \mathbb{T}^{\text{Alessi}}$ defined as

$$\begin{aligned} f(\emptyset) &\triangleq U, \\ f(\{i\}) &\triangleq i, \\ f(u \cup \{i\}) &\triangleq f(u) \cap i. \end{aligned}$$

where $u \in \mathcal{P}_{fin}(\omega)$, $i \in \omega$.

16E.8. Show Theorem 16D.24 using the mapping $f : \mathcal{P}_{fin}(\mathsf{Pm}) \rightarrow \mathbb{T}^{\text{Plotkin}}$ defined as

$$\begin{aligned} f(\emptyset) &\triangleq U, \\ f(\{\omega\}) &\triangleq \omega, \\ f(u \cup \{a\}) &\triangleq f(u) \cap f(\{a\}), \\ f(\{\langle u, v \rangle\}) &\triangleq f(u) \rightarrow f(v). \end{aligned}$$

where $u, v \in \mathcal{P}_{fin}(\mathsf{Pm})$, $a \in \mathsf{Pm}$.

16E.9. Let $\mathcal{T} \in \mathbf{TT}^{-U}$. Prove the following statements.

1. The filter quasi λ -model $\mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, \cdot, \llbracket \cdot \rrbracket^{\mathcal{F}^{\mathcal{T}}} \rangle$ is not a λ -model [Hint. Consider the constant function $\lambda x.y$.]
2. The filter quasi λ -model $\mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, \cdot, \llbracket \cdot \rrbracket^{\mathcal{F}^{\mathcal{T}}} \rangle$ 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 $\mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, F^{\mathcal{T}}, G^{\mathcal{T}} \rangle$.

16E.10. Let $\mathcal{F} = \mathcal{F}^{BCD}$ be the model described in Section 16B. To every intersection type $A \in \mathbb{T}^{BCD}$ associate a *height*, notation $|A|$, as follows.

- (i) $|\mathbf{U}| \triangleq 0$,
- (ii) $|\mathbf{c}_i| \triangleq 1$, for all $\mathbf{c}_i \in \mathbb{A}^{BCD}$ such that $\mathbf{c}_i \neq \mathbf{U}$,
- (iii) $|A \cap B| \triangleq \max\{|A|, |B|\}$,
- (iv) $|A \rightarrow B| \triangleq 1 + \max\{|A|, |B|\}$.

Given a filter $d \in \mathcal{F}$, define

$$d[n] \triangleq \{A \in d \mid |A| \leq n\}$$

and set $d_n \triangleq \uparrow d[n]$, where $\uparrow X$ is the filter generated by the set X (this is the intersection of all filters containing X). Prove that the mappings $d \mapsto d_n : \mathcal{F} \rightarrow \mathcal{F}$ define a notion of approximation over \mathcal{F} . [Hint. (i) 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 $A \in d[n]$.

(ii) Prove that $d_0 = \uparrow \{\mathbf{U}\}$.
 (iii) 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 $B \neq \mathbf{U}$ and $A' \rightarrow B' \leq A \rightarrow B$, then $A \leq A'$ and $B' \leq B$.
- If $C \leq A \rightarrow B$ and $|C| \leq n+1$, then there exist A', B' such that $|A'|, |B'| \leq n$ and $C \leq A' \rightarrow B' \leq A \rightarrow B$.

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 $A \leq B$ iff $A(\lessdot)^* B$.]

CHAPTER 17

ADVANCED PROPERTIES AND APPLICATIONS

This chapter proves some properties of intersection types in relation to terms and models.

Section 17A defines a realizability interpretation of types ([Barendregt, Coppo, and Dezani-Ciancaglini \[1983\]](#)). Types are interpreted as subsets of a domain of discourse \mathcal{D} . Assuming a (partial) application $\cdot : \mathcal{D} \times \mathcal{D} \rightarrow \mathcal{D}$, and a type environment ξ , i.e. a mapping from type atoms to subsets of \mathcal{D} , we can define an interpretation $\llbracket A \rrbracket_\xi = \llbracket A \rrbracket_\xi^\mathcal{T} \subseteq \mathcal{D}$ for each type A in \mathbb{T}_\rightarrow by giving \rightarrow the *realizability interpretation*, i.e.

$$\begin{aligned}\llbracket \alpha \rrbracket_\xi &\triangleq \xi(\alpha); \\ \llbracket A \rightarrow B \rrbracket_\xi &\triangleq \llbracket A \rrbracket_\xi \rightarrow \llbracket B \rrbracket_\xi \triangleq \{d \in \mathcal{D} \mid d \cdot \llbracket A \rrbracket_\xi \subseteq \llbracket B \rrbracket_\xi\}.\end{aligned}$$

This semantics, due to [Scott \[1975a\]](#), can be extended to intersection types by interpreting \mathbb{U} as the domain of discourse and the intersection \cap on types as ordinary intersection:

$$\begin{aligned}\llbracket \mathbb{U} \rrbracket_\xi &\triangleq \mathcal{D}; \\ \llbracket A \cap B \rrbracket_\xi &\triangleq \llbracket A \rrbracket_\xi \cap \llbracket B \rrbracket_\xi.\end{aligned}$$

The first requirement will be met by considering only ξ with $\xi(\mathbb{U}) = \mathcal{D}$. Then, \leq is interpreted as inclusion between sets. For interpreting λ -terms, it is enough to require that \mathcal{D} is a quasi λ -model, depending on a valuation ρ of the term variables in \mathcal{D} . One says that \mathcal{D} satisfies $M : A$ under ρ, ξ , notation $\mathcal{D}, \rho, \xi \models^\mathcal{T} M : A$ or simply $\mathcal{D}, \rho, \xi \models M : A$, if

$$\llbracket M \rrbracket_\rho \in \llbracket A \rrbracket_\xi^\mathcal{T}.$$

Then Γ satisfies $M : A$, notation $\Gamma \models M : A$, is defined by

$$\Gamma \models M : A \stackrel{\Delta}{\iff} \forall \mathcal{D}, \rho, \xi. [\mathcal{D}, \rho, \xi \models \Gamma \Rightarrow \mathcal{D}, \rho, \xi \models M : A].$$

First soundness is proved, i.e.

$$\Gamma \vdash_{\cap}^\mathcal{T} M : A \Rightarrow \Gamma \models_{\cap}^\mathcal{T} M : A.$$

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.

$$\Gamma \models_{\cap}^\mathcal{T} M : A \Rightarrow \Gamma \vdash_{\cap}^\mathcal{T} M : A] \Leftrightarrow \mathcal{T} \in \text{NTT}^{\mathbb{U}}.$$

The proof of completeness for a natural type theory \mathcal{T} follows by taking $\mathcal{D} = \mathcal{F}^\mathcal{T}$ where $\mathcal{F}^\mathcal{T}$ is the filter quasi λ -model over \mathcal{T} .

In [Barendregt, Coppo, and Dezani-Ciancaglini \[1983\]](#) the completeness of the type theory BCD was used to show the completeness of simple types: for all $M \in \Lambda^\varnothing$ and $A \in \mathbb{T}^{\mathbb{A}}_\rightarrow$

$$\vdash_{\lambda_\rightarrow} M : A \Leftrightarrow \models M : A.$$

In Sections 17B and 17D 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 *persistent* property P if for all appropriate (to be defined in Section 17B) arguments \vec{N} the term $M\vec{N}$ has property P . The sets of untyped lambda terms having these properties are denoted respectively by SN, N, HN, PSN, PN, PHN. For a set of terms $X \subseteq \Lambda$ write

$$X \uparrow \beta = \{M \mid \exists N \in X. M \rightarrow_\beta N\}.$$

We denote by Γ_α the set of type declarations which associate α to all variables. For $\mathcal{T} \in \{\text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{HL}\}$ Fig. 43 defines a context $\text{Ctx}^\mathcal{T}$ and a subset $\text{Set}^\mathcal{T}(c)$ of lambda terms for each type atom $c \in \{0, 1, \text{U}\}$.

\mathcal{T}	$\text{Ctx}^\mathcal{T}$	$\text{Set}^\mathcal{T}(0)$	$\text{Set}^\mathcal{T}(1)$	$\text{Set}^\mathcal{T}(\text{U})$
Park	\emptyset	$\Lambda^\varnothing \uparrow \beta$	—	Λ
CDZ	Γ_0	PN	N	Λ
HR	Γ_1	\emptyset	$\Lambda^1 \uparrow \beta$	Λ
DHM	Γ_0	PHN	HN	Λ
HL	Γ_0	PSN	SN	—

FIGURE 43. Context and Set associated to 0, 1 and U.

We will show the following characterizations.

$$M \in \text{Set}^\mathcal{T}(c) \Leftrightarrow \text{Ctx}^\mathcal{T} \vdash^\mathcal{T} M : c.$$

The characterizations for U are immediate; those for 1 are given in Theorems 17B.15 (CDZ, DHM, HL) and 17D.9 (HR); those for 0 in Theorem 17D.3 (Park), Lemma 17D.6 (HR); the remaining ones can be found in [Dezani-Ciancaglini, Honsell, and Motohamo \[2005\]](#) (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. Section 17B uses the type interpretation defined in Section 17A and the standard technique of type stable sets.
2. Section 17D uses the Approximation Theorem presented in Section 17C.

In Section 17C, we introduce appropriate notions of approximants for almost all the type theories of Fig. 33. 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 \perp 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 \perp . In case of AO, the notion of approximant is relaxed and abstractions are not replaced by \perp . 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, Honsell, and Motohamo [2001]). The proof technique used is a variant of stable sets over a Kripke applicative structure. In Section 17D some applications of the Approximation Theorem are given. Amongst these, the characterizations for Park and HL, mentioned in Fig. 43.

Finally in Section 17E 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 $\Gamma \vdash^{\text{CDV}} M : A$. 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 16E.2, therefore inhabitation is decidable. The question of decidability of inhabitation for several other type theories remains open.

Notice that the contents of the first four sections of this chapter can also be regarded as applications of intersection types.

17A. Realizability interpretation of types

The natural set-theoretic semantics for type assignment in λ_{\rightarrow} based on untyped λ -models is given in Scott [1975a] where it was shown that

$$\Gamma \vdash_{\lambda_{\rightarrow}} M : A \Rightarrow \Gamma \models M : A.$$

Scott asked whether the converse (completeness) holds. In Barendregt, Coppo, and Dezani-Ciancaglini [1983] the notion of semantics was extended to intersection types and completeness was proved for $\lambda_{\cap}^{\text{BCD}}$ via the corresponding filter model. Completeness for λ_{\rightarrow} follows by a conservativity result. In Hindley [1983] an alternative proof of completeness for λ_{\rightarrow} was given directly, using a term model. Variations of the semantics are presented in Dezani-Ciancaglini, Honsell, and Alessi [2003].

Recall that quasi λ -models are defined in Definition 16A.2 and are λ -models without the requirement that (β) holds. Using this notion one can distinguish between models for the λI -calculus and the full λ -calculus.

17A.1. DEFINITION (Type Interpretation). Let $\mathcal{D} = \langle \mathcal{D}, \cdot, \llbracket \cdot \rrbracket^{\mathcal{D}} \rangle$ be a quasi λ -model and let \mathcal{T} be an intersection type theory over the atoms $\mathbb{A}^{\mathcal{T}}$.

(i) Remember that for $X, Y \subseteq \mathcal{D}$ we defined

$$(X \Rightarrow Y) \triangleq \{d \in \mathcal{D} \mid \forall e \in X. d \cdot e \in Y\}.$$

(ii) The *type interpretation* induced by the type environment $\xi : \mathbb{A}^{\mathcal{T}} \rightarrow \mathcal{P}(\mathcal{D})$, with $\xi(U) = \mathcal{D}$ if $\mathcal{T} \in \text{TT}^U$, is the map $\llbracket \cdot \rrbracket_{\xi}^{\mathcal{D}} : \mathbb{A}^{\mathcal{T}} \rightarrow \mathcal{P}(\mathcal{D})$ defined as follows.

$$\llbracket \alpha \rrbracket_{\xi}^{\mathcal{D}} \triangleq \xi(\alpha),$$

$$\llbracket A \rightarrow B \rrbracket_{\xi}^{\mathcal{D}} \triangleq \llbracket A \rrbracket_{\xi}^{\mathcal{D}} \Rightarrow \llbracket B \rrbracket_{\xi}^{\mathcal{D}},$$

$$\llbracket A \cap B \rrbracket_{\xi}^{\mathcal{D}} \triangleq \llbracket A \rrbracket_{\xi}^{\mathcal{D}} \cap \llbracket B \rrbracket_{\xi}^{\mathcal{D}}.$$

The above definition is the extension to intersection-types of the *simple semantics* for simple types of Scott [1975a], generalized by allowing \mathcal{D} to be just a quasi λ -model instead of a λ -model.

It is easy to verify that $\llbracket U \rightarrow U \rrbracket_{\xi}^{\mathcal{D}} = \mathcal{D}$ for all \mathcal{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 \rightarrow is interpreted as the arrow induced by the application \cdot on \mathcal{D} . The following definition is necessary to require that the interpretation preserves the remaining two typability rules: abstraction and subtyping.

17A.2. DEFINITION. Let $\mathcal{D} = \langle \mathcal{D}, \cdot, []^{\mathcal{D}} \rangle$ be a quasi λ -model and $\xi : \mathbb{A}^{\mathcal{T}} \rightarrow \mathbf{P}(\mathcal{D})$ be a type environment.

(i) The pair (\mathcal{D}, ξ) is *\rightarrow -good* if for all $A, B \in \mathbb{T}^{\mathcal{T}}$ for all environments ρ , terms M and variables x

$$[\forall d \in [A]_{\xi}^{\mathcal{D}}. [M]_{\rho[x:=d]}^{\mathcal{D}} \in [B]_{\xi}^{\mathcal{D}}] \Rightarrow [\lambda x. M]_{\rho}^{\mathcal{D}} \in [A]_{\xi}^{\mathcal{D}} \rightarrow [B]_{\xi}^{\mathcal{D}};$$

(ii) The pair (\mathcal{D}, ξ) *preserves* $\leq_{\mathcal{T}}$ if for all $A, B \in \mathbb{T}^{\mathcal{T}}$ one has

$$A \leq_{\mathcal{T}} B \Rightarrow [A]_{\xi}^{\mathcal{D}} \subseteq [B]_{\xi}^{\mathcal{D}}.$$

We now introduce the semantics of type assignment.

17A.3. DEFINITION (Semantic Satisfiability). Let $\mathcal{T} \in \text{TT}$.

(i) Let $\mathcal{D} = \langle \mathcal{D}, \cdot, []^{\mathcal{D}} \rangle$ be a quasi λ -model. Define

$$\begin{aligned} \mathcal{D}, \rho, \xi \models M : A &\iff [M]_{\rho}^{\mathcal{D}} \in [A]_{\xi}^{\mathcal{D}}; \\ \mathcal{D}, \rho, \xi \models \Gamma &\iff \mathcal{D}, \rho, \xi \models x : B, \text{ for all } (x:B) \in \Gamma. \end{aligned}$$

(ii) We say that Γ *satisfies* $M : A$, notation $\Gamma \models_{\cap}^{\mathcal{T}} M : A$, if

$$\mathcal{D}, \rho, \xi \models \Gamma \Rightarrow \mathcal{D}, \rho, \xi \models M : A,$$

for all \mathcal{D}, ξ, ρ such that (\mathcal{D}, ξ) is \rightarrow -good and preserves $\leq_{\mathcal{T}}$.

Derivability in the type system implies semantic satisfiability, as shown in the next theorem.

17A.4. THEOREM (Soundness). *For all $\mathcal{T} \in \text{TT}$ one has*

$$\Gamma \vdash_{\cap}^{\mathcal{T}} M : A \Rightarrow \Gamma \models_{\cap}^{\mathcal{T}} M : A.$$

PROOF. By induction on the derivation of $\Gamma \vdash_{\cap}^{\mathcal{T}} M : A$. Rules $(\rightarrow E)$, $(\cap I)$ and (U_{top}) are sound by the definition of type interpretation (Definition 17A.1).

As to the soundness of rule $(\rightarrow I)$, assume $\Gamma, x:A \vdash_{\cap}^{\mathcal{T}} M : B$ in order to show $\Gamma \models^{\mathcal{T}} (\lambda x. M) : (A \rightarrow B)$. Assuming $\mathcal{D}, \rho, \xi \models \Gamma$ we have to show

$$[\lambda x. M]_{\rho}^{\mathcal{D}} \in [A]_{\xi}^{\mathcal{D}} \rightarrow [B]_{\xi}^{\mathcal{D}}.$$

Let $d \in [A]_{\xi}^{\mathcal{D}}$. We are done if we can show

$$[M]_{\rho[x:=d]}^{\mathcal{D}} \in [B]_{\xi}^{\mathcal{D}},$$

because (\mathcal{D}, ξ) are \rightarrow -good. Now $\mathcal{D}, \rho[x := d], \xi \models \Gamma, x:A$, hence $[M]_{\rho[x:=d]}^{\mathcal{D}} \in [B]_{\xi}^{\mathcal{D}}$, by the induction hypothesis for $\Gamma, x:A \vdash_{\cap}^{\mathcal{T}} M : B$.

Rule (\leq) is sound, as we consider only (\mathcal{D}, ξ) that preserve $\leq_{\mathcal{T}}$. ■

Completeness

Now we characterize the complete theories.

17A.5. NOTATION. Let $\mathcal{T} \in \text{NTT}^U$ and $\mathcal{F}^\mathcal{T} = \langle \mathcal{F}^\mathcal{T}, \cdot, \llbracket \cdot \rrbracket^{\mathcal{F}^\mathcal{T}} \rangle$ its corresponding filter quasi λ -model, see Definition 16B.1.

(i) Let $\xi^\mathcal{T} : \mathbb{A}^\mathcal{T} \rightarrow \mathsf{P}(\mathcal{F}^\mathcal{T})$ be the type environment defined by

$$\xi^\mathcal{T}(\alpha) \triangleq \{X \in \mathcal{F}^\mathcal{T} \mid \alpha \in X\}.$$

(ii) Let $\llbracket \cdot \rrbracket^\mathcal{T} : \mathbb{T}^\mathcal{T} \rightarrow \mathsf{P}(\mathcal{F}^\mathcal{T})$ be the mapping $\llbracket \cdot \rrbracket_{\xi^\mathcal{T}}^{\mathcal{F}^\mathcal{T}}$.

The mapping $\llbracket \cdot \rrbracket^\mathcal{T} : \mathbb{T}^\mathcal{T} \rightarrow \mathsf{P}(\mathcal{F}^\mathcal{T})$ 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).

17A.6. PROPOSITION. *Let $\mathcal{T} \in \text{NTT}^U$. Then we have*

$$\llbracket A \rrbracket^\mathcal{T} = \{X \in \mathcal{F}^\mathcal{T} \mid A \in X\}.$$

PROOF. By induction on A . The only interesting case is when A is an arrow type. If $A \equiv B \rightarrow C$ we have

$$\begin{aligned} \llbracket B \rightarrow C \rrbracket^\mathcal{T} &= \{X \in \mathcal{F}^\mathcal{T} \mid \forall Y \in \llbracket B \rrbracket^\mathcal{T}. X \cdot Y \in \llbracket C \rrbracket^\mathcal{T}\}, && \text{by definition,} \\ &= \{X \in \mathcal{F}^\mathcal{T} \mid \forall Y. B \in Y \Rightarrow C \in X \cdot Y\}, && \text{by induction,} \\ &= \{X \in \mathcal{F}^\mathcal{T} \mid C \in X \cdot \uparrow B\}, && \text{by monotonicity,} \\ &= \{X \in \mathcal{F}^\mathcal{T} \mid C \in \uparrow\{C' \mid \exists B'. B' \rightarrow C' \in X\}\}, && \text{by the definition of} \\ &&& \text{filter application,} \\ &= \{X \in \mathcal{F}^\mathcal{T} \mid B \rightarrow C \in X\}, && \text{by } (\rightarrow) \text{ and } (U \rightarrow). \blacksquare \end{aligned}$$

17A.7. LEMMA. *Let $\mathcal{T} \in \text{NTT}^U$. Then $(\mathcal{F}^\mathcal{T}, \xi^\mathcal{T})$ is \rightarrow -good and preserves $\leq_\mathcal{T}$.*

PROOF. Suppose that $X \in \llbracket A \rrbracket^\mathcal{T}$ is such that

$$\llbracket M \rrbracket_{\rho[x:=X]}^\mathcal{T} \in \llbracket B \rrbracket^\mathcal{T},$$

in order to show $\llbracket \lambda x. M \rrbracket_\rho^\mathcal{T} \cdot X \in \llbracket B \rrbracket^\mathcal{T}$, which establishes that $(\mathcal{F}^\mathcal{T}, \xi^\mathcal{T})$ is \rightarrow -good. By Proposition 17A.6 we have $B \in \llbracket M \rrbracket_{\rho[x:=X]}^\mathcal{T}$, hence $B \in f(X)$, where we have put $f = \lambda d. \llbracket M \rrbracket_{\rho[x:=d]}^\mathcal{T}$. Since by Lemma 15B.8(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 $\llbracket \lambda x. M \rrbracket_\rho^\mathcal{T} \cdot X = F^\mathcal{T}(G^\mathcal{T}(f))(X) \in \llbracket B \rrbracket^\mathcal{T}$, by Definition 16A.5(i) and Proposition 17A.6.

As an immediate consequence of Proposition 17A.6 we get

$$A \leq_\mathcal{T} B \Leftrightarrow \forall X \in \mathcal{F}^\mathcal{T}. [A \in X \Rightarrow B \in X] \Leftrightarrow \llbracket A \rrbracket^\mathcal{T} \subseteq \llbracket B \rrbracket^\mathcal{T},$$

and therefore $(\mathcal{F}^\mathcal{T}, \xi^\mathcal{T})$ preserves $\leq_\mathcal{T}$. \blacksquare

Now we can prove the desired completeness result.

17A.8. THEOREM (Completeness). *Let $\mathcal{T} \in \text{TT}^U$.*

(i) $[\Gamma \vdash_\cap^\mathcal{T} M : A \Rightarrow \Gamma \vdash_\cap^\mathcal{T} M : A]$ iff $\mathcal{T} \in \text{NTT}^U$.

(ii) Let $\mathcal{T} \in \text{NTT}^U$. Then

$$\Gamma \vdash_\cap^\mathcal{T} M : A \Leftrightarrow \Gamma \vdash_\cap^\mathcal{T} M : A.$$

PROOF. (i) (\Rightarrow) It is easy to verify that all type interpretations validate rule (\rightarrow) and the axiom (U_{top}). As to axiom ($\rightarrow \cap$), consider the \mathcal{T} -basis $\Gamma = \{x:(A \rightarrow B) \cap (A \rightarrow C)\}$. From Definition 17A.1 we get

$$\Gamma \models_{\cap}^{\mathcal{T}} x : A \rightarrow (B \cap C).$$

Hence, by hypothesis, we have $\Gamma \vdash_{\cap}^{\mathcal{T}} x : A \rightarrow (B \cap C)$. Using Theorem 14A.9(i) it follows that $(A \rightarrow B) \cap (A \rightarrow C) \leq_{\mathcal{T}} A \rightarrow B \cap C$. Therefore axiom ($\rightarrow \cap$) holds.

As to axiom ($U \rightarrow$)

$$\begin{aligned} \models_{\cap}^{\mathcal{T}} x : U \rightarrow U &\Rightarrow \vdash_{\cap}^{\mathcal{T}} x : (U \rightarrow U) \\ &\Rightarrow x : U \vdash_{\cap}^{\mathcal{T}} x : (U \rightarrow U) \\ &\Rightarrow U \leq_{\mathcal{T}} (U \rightarrow U), \quad \text{by Theorem 14A.9(i).} \end{aligned}$$

This proves (\Rightarrow).

(\Leftarrow) Now suppose $\Gamma \models^{\mathcal{T}} M : A$ towards $\Gamma \vdash_{\cap}^{\mathcal{T}} M : A$. We use the filter quasi λ -model $\langle \mathcal{F}^{\mathcal{T}}, \cdot, \llbracket \cdot \rrbracket^{\mathcal{T}} \rangle$. By Lemma 17A.7 we have that $\Gamma \models_{\cap}^{\mathcal{T}} M : A$ implies $\llbracket M \rrbracket_{\rho_{\Gamma}}^{\mathcal{T}} \in \llbracket A \rrbracket^{\mathcal{T}}$, where

$$\rho_{\Gamma}(x) = \begin{cases} \uparrow A & \text{if } x : A \in \Gamma, \\ \uparrow U & \text{otherwise.} \end{cases}$$

We conclude $\Gamma \vdash_{\cap}^{\mathcal{T}} M : A$, using Proposition 17A.6 and Theorem 16B.7(i).

(ii) By Proposition 17A.4 and (i). ■

17A.9. COROLLARY. *For $\mathcal{T} \in \{\text{Scott, Park, CDZ, HR, DHM, BCD}\}$ one has for all $M \in \Lambda^{\phi}$*

$$\models_{\cap}^{\mathcal{T}} M : A \Rightarrow \vdash_{\cap}^{\mathcal{T}} M : A.$$

17A.10. REMARK. In [Barendregt, Coppo, and Dezani-Ciancaglini \[1983\]](#) the completeness of the type theory BCD was used to show the completeness of simple types via the following conservativity result

$$\forall A \in \mathbb{T}_{\rightarrow}^{\mathbb{A}} \forall M \in \Lambda [\vdash_{\cap}^{BCD} M : A \Rightarrow \vdash_{\lambda \rightarrow} M : A].$$

This solved an open problem of [Scott \[1975a\]](#). Independently a different completeness proof has been given in [Hindley \[1983\]](#).

17A.11. COROLLARY. *For $M \in \Lambda^{\phi}$ and $A \in \mathbb{T}_{\rightarrow}^{\mathbb{A}}$ one has*

$$\vdash_{\lambda \rightarrow} M : A \Leftrightarrow \models M : A.$$

PROOF. (\Rightarrow) This is soundness, Proposition 3A.37, part \Rightarrow .

$$\begin{aligned} (\Leftarrow) \models_{\lambda \rightarrow} M : A &\Rightarrow \vdash_{\cap}^{BCD} M : A \\ &\Rightarrow \vdash_{\cap}^{BCD} M : A, \quad \text{by Corollary 17A.9,} \\ &\Rightarrow \vdash_{\lambda \rightarrow} M : A, \quad \text{by Remark 17A.10. ■} \end{aligned}$$

Similar results for $\mathcal{T} \in \text{PTT}$ can be found in [Dezani-Ciancaglini, Honsell, and Alessi \[2003\]](#). See also Exercises 17F.3 and 17F.4.

17B. 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, it is only possible to characterize 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.

17B.1. DEFINITION. (i) A lambda term M is called *β -strongly normalizing* (SN) if there is no infinite β -reduction starting with M . This is equivalent to being $\beta\eta\text{-SN}$.

(ii) $\text{SN} \triangleq \{M \mid M \text{ is strongly normalizing}\}$.

For example $\text{SK} \in \text{SN}$, but $\Omega, \text{SK}\Omega \notin \text{SN}$, even if the last term has a normal form.

17B.2. LEMMA (van Raamsdonk, Severi, Sørensen, and Xi [1999]). *The set SN is the smallest set of terms closed under the following rules.*

$$\frac{M_1 \in \text{SN}, \dots, M_n \in \text{SN}}{xM_1 \cdots M_n \in \text{SN}} \quad n \geq 0$$

$$\frac{M \in \text{SN}}{\lambda x.M \in \text{SN}}$$

$$\frac{M[x := N]M_1 \cdots M_n \in \text{SN} \quad N \in \text{SN}}{(\lambda x.M)NM_1 \cdots M_n \in \text{SN}} \quad n \geq 0$$

PROOF. Let $\mathcal{S}N$ be the set defined by these rules. We show

$$M \in \mathcal{S}N \Leftrightarrow M \in \text{SN}.$$

(\Rightarrow) By induction on the generation of $\mathcal{S}N$.

(\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 $M \in \mathcal{S}N$ 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 $M \in \text{SN}$. In the case $\|M\| = n > 0$, we have three cases, being $x\vec{M}$, $\lambda x.N$ or $(\lambda x.P)N\vec{M}$. 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 $P[x := N]M_1 \cdots M_n$ and N , where the norm strictly decreases. ■

17B.3. DEFINITION. (i) A term M is *persistently head normalizing* if $M\vec{N}$ has a head normal form for all terms \vec{N} .

(ii) A term M is *persistently normalizing* if $M\vec{N}$ has a normal form for all normalizable terms \vec{N} .

(iii) A term M is *persistently strongly normalizing* if $M\vec{N}$ is strongly normalizing for all strongly normalizing terms \vec{N} .

The notion of persistently normalizing terms has been introduced in Böhm and Dezani-Ciancaglini [1975].

17B.4. NOTATION. Several classes of lambda terms are denoted by an acronym.

$$\mathbf{HN} \triangleq \{M \mid M \text{ has a head normal form}\}.$$

$$\mathbf{PHN} \triangleq \{M \mid M \text{ is persistently head normalizing}\}.$$

$$\mathbf{N} \triangleq \{M \mid M \text{ has a normal form}\}.$$

$$\mathbf{PN} \triangleq \{M \mid M \text{ is persistently normalizing}\}.$$

$$\mathbf{SN} \triangleq \{M \mid M \text{ is strongly normalizing}\}.$$

$$\mathbf{PSN} \triangleq \{M \mid M \text{ is persistently strongly normalizing}\}.$$

The following inclusions follow immediately by definition, except those of the last line below, namely $\mathbf{PSN} \subseteq \mathbf{PN} \subseteq \mathbf{PHN}$, which are proved in Exercise 17F.6.

The inclusion $\mathbf{PSN} \subseteq \mathbf{PN}$ follows also comparing Figs 34 and 43. It is easy to find examples to show that all these inclusions are strict.

$$\begin{array}{ccccccc} \mathbf{SN} & \subset & \mathbf{N} & \subset & \mathbf{HN} \\ \cup & & \cup & & \cup \\ \mathbf{PSN} & \subset & \mathbf{PN} & \subset & \mathbf{PHN} \end{array}$$

17B.5. EXAMPLE. (i) $Kx\Omega \in \mathbf{PN}$ but not in \mathbf{SN} , hence not in \mathbf{PSN} .

(ii) $(\lambda x.Kx\Omega) \in \mathbf{N}$ but not in \mathbf{SN} nor in \mathbf{PHN} , hence not in \mathbf{PN} .

(iii) $x\Omega \in \mathbf{PHN}$ but not in \mathbf{N} , hence not in \mathbf{PN} .

(iv) $x \in \mathbf{PSN}$.

(v) $\mathbb{I} \in \mathbf{SN}$, but not in \mathbf{PN} .

Stable sets

We will use the standard proof technique of type stable sets (Krivine [1990]).

17B.6. DEFINITION. The open term model of the lambda calculus consists of arbitrary λ -terms modulo β -conversion. That is $\mathcal{M}_{\Lambda(\beta)} \triangleq \langle \Lambda, \cdot, [\]^\Lambda \rangle$ and we have

$$\begin{aligned} \llbracket M \rrbracket_\rho^\Lambda &\triangleq \{N \mid N =_\beta M[\vec{x} := \rho(\vec{x})]\}, \quad \text{where } \vec{x} = FV(M), \\ \llbracket M \rrbracket_\rho^\Lambda \cdot \llbracket N \rrbracket_\rho^\Lambda &\triangleq \llbracket MN \rrbracket_\rho^\Lambda. \end{aligned}$$

The substitution $M[x := \rho(x)]$ is to be interpreted as follows. If $\rho(x) = [P]_{=\beta}$, then $M[x := \rho(x)] = M[x := P]$; this is independent of the choice of the representative P .

17B.7. REMARK. In $\mathcal{M}_{\Lambda(\beta)} = \langle \Lambda, \cdot, [\]^\Lambda \rangle$ one has for $X, Y \subseteq \mathcal{M}_{\Lambda(\beta)}$

$$(X \Rightarrow Y) = \{M \in \Lambda \mid \forall N \in X \ MN \in Y\}.$$

17B.8. DEFINITION. Let $X \subseteq \Lambda$.

(i) X is called *closed under head expansion* of redexes, notation $\text{h}\uparrow$ -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 $X \subseteq \mathbf{HN}$, it contains $x\vec{M}$ for all $\vec{M} \in \Lambda$ and is $\text{h}\uparrow$ -closed.

(iii) X is **N-stable** if $X \subseteq \mathbf{N}$, it contains $x\vec{M}$ for all $\vec{M} \in \mathbf{N}$ and is $\text{h}\uparrow$ -closed.

(iv) A set X is **SN-stable** if $X \subseteq \mathbf{SN}$, it contains $x\vec{M}$ for all $\vec{M} \in \mathbf{SN}$ and is closed under head expansion of redexes, whose arguments are in \mathbf{SN} .

From the above definition and Lemma 17B.2 we easily get the following.

17B.9. PROPOSITION. Let $S \in \{\text{HN}, \text{N}, \text{SN}\}$ 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 $(X \rightarrow Y)$ and $(X \cap Y)$ are S -stable.
- (iv) If Y is HN -stable and $X \neq \emptyset$, then $(X \rightarrow Y)$ is HN -stable. ■

17B.10. DEFINITION (Type environments). (i) The type environment

$\xi = \xi_{\text{BCD}}^1 : \mathbb{A}_\infty \rightarrow \mathcal{P}(\Lambda)$ in the open term model $\mathcal{M}_{\Lambda(\beta)}$ is defined as follows.

$$\xi(\alpha) \triangleq \text{HN}, \quad \text{if } \alpha \in \mathbb{A}_\infty.$$

- (ii) The type environment $\xi = \xi_{\text{BCD}}^2$ in $\mathcal{M}_{\Lambda(\beta)}$ is defined as follows.

$$\xi(\alpha) \triangleq \text{N}, \quad \text{if } \alpha \in \mathbb{A}_\infty.$$

- (iii) The type environment $\xi = \xi_{\text{DHM}}$ in $\mathcal{M}_{\Lambda(\beta)}$ is defined as follows.

$$\xi(0) \triangleq \text{PHN};$$

$$\xi(1) \triangleq \text{HN}.$$

- (iv) The type environment $\xi = \xi_{\text{CDZ}}$ in $\mathcal{M}_{\Lambda(\beta)}$ is defined as follows.

$$\xi(0) \triangleq \text{PN};$$

$$\xi(1) \triangleq \text{N}.$$

- (v) The type environment $\xi = \xi_{\text{CDV}}$ in $\mathcal{M}_{\Lambda(\beta)}$ is defined as follows.

$$\xi(\alpha) \triangleq \text{SN}, \quad \text{if } \alpha \in \mathbb{A}_\infty.$$

- (vi) The type environment $\xi = \xi_{\text{HL}}$ in $\mathcal{M}_{\Lambda(\beta)}$ is defined as follows.

$$\xi(0) \triangleq \text{PSN};$$

$$\xi(1) \triangleq \text{SN}.$$

17B.11. LEMMA. (i) $\llbracket A \rrbracket_{\xi_{\text{BCD}}^1}$ and $\llbracket A \rrbracket_{\xi_{\text{DHM}}}$ are HN -stable.

- (ii) $\llbracket A \rrbracket_{\xi_{\text{BCD}}^2}$ and $\llbracket A \rrbracket_{\xi_{\text{CDZ}}}$ are N -stable.

- (iii) $\llbracket A \rrbracket_{\xi_{\text{CDV}}}$ and $\llbracket A \rrbracket_{\xi_{\text{HL}}}$ are SN -stable.

PROOF. All points follow easily from Proposition 17B.9. ■

We shall show that for each type environment ξ_τ of Definition 17B.10 ($\mathcal{M}_{\Lambda(\beta)}, \xi_\tau$) are \rightarrow -good, see Definition 17A.2(i), and preserve \leq_τ . The proof occupies 17B.12-17B.14.

17B.12. LEMMA. (i) $M \in \text{SN}, N \in \text{PSN} \Rightarrow M[x := N] \in \text{SN}$.

- (ii) $M \in \text{SN}, N \in \text{PSN} \Rightarrow MN \in \text{SN}$.

- (iii) $M \in \text{N}, N \in \text{PN} \Rightarrow M[x := N] \in \text{N}$.

- (iv) $M \in \text{N}, N \in \text{PN} \Rightarrow MN \in \text{N}$.

- (v) $M \in \text{HN}, N \in \text{PHN} \Rightarrow M[x := N] \in \text{N}$.

- (vi) $M \in \text{HN}, N \in \text{PHN} \Rightarrow MN \in \text{HN}$.

PROOF. The first two statements follow using the inductive definition of SN given in 17B.2. The rest follow by an easy induction on the (head) normal form of M . ■

17B.13. PROPOSITION. (i) $\text{PSN} = (\text{N} \Rightarrow \text{PSN})$.

- (ii) $\text{SN} = (\text{PSN} \Rightarrow \text{SN})$.
- (iii) $\text{PN} = (\text{N} \Rightarrow \text{PN})$.
- (iv) $\text{N} = (\text{PN} \Rightarrow \text{N})$.
- (v) $\text{PHN} = (\text{HN} \Rightarrow \text{PHN})$.
- (vi) $\text{HN} = (\text{PHN} \Rightarrow \text{HN})$.

PROOF. All cases are immediate except the inclusions $\text{SN} \subseteq (\text{PSN} \Rightarrow \text{SN})$, $\text{N} \subseteq (\text{PN} \Rightarrow \text{N})$ and $\text{HN} \subseteq (\text{PHN} \Rightarrow \text{HN})$. These follow easily from Lemma 17B.12 (ii), (iv) and (vi). ■

17B.14. LEMMA. *For all $\xi_{\mathcal{T}}$ of Definition 17B.10 we have the following.*

- (i) $\forall N \in \llbracket B \rrbracket_{\xi_{\mathcal{T}}}, M[x := N] \in \llbracket A \rrbracket_{\xi_{\mathcal{T}}} \text{ implies } (\lambda x.M) \in \llbracket B \rightarrow A \rrbracket_{\xi_{\mathcal{T}}}$.
- (ii) $A \leq_{\mathcal{T}} B \Rightarrow \llbracket A \rrbracket_{\xi_{\mathcal{T}}} \subseteq \llbracket B \rrbracket_{\xi_{\mathcal{T}}}$.

I.e. for all $\xi_{\mathcal{T}}$ of Definition 17B.10 $(\mathcal{M}_{\Lambda(\beta)}, \xi_{\mathcal{T}})$ are \rightarrow -good and preserve $\leq_{\mathcal{T}}$.

PROOF. (i) If either $\mathcal{T} \neq \text{CDV}$ or $\mathcal{T} = \text{CDV}$ and $N \in \text{SN}$ one easily shows that $M[x := N] \in \llbracket A \rrbracket_{\xi_{\mathcal{T}}}$ implies $(\lambda x.M)N \in \llbracket A \rrbracket_{\xi_{\mathcal{T}}}$ by induction on A using Proposition 17B.9. The conclusion from the definition of \rightarrow .

(ii) By induction on the generation of $\leq_{\mathcal{T}}$, using Proposition 17B.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 $\Gamma_0^M = \{x_1:0, \dots, x_n:0\}$, where $\{x_1, \dots, x_n\} = \text{FV}(M)$.

17B.15. THEOREM (Characterization Theorems).

- (i) $M \in \text{N} \Leftrightarrow \forall \mathcal{T} \in \text{TT}^{\text{U}} \exists \Gamma, A. \Gamma \vdash_{\cap}^{\mathcal{T}} M : A \ \& \ \text{U} \notin \Gamma, A$
 - $\Leftrightarrow \exists \Gamma, A. \Gamma \vdash_{\cap}^{\text{BCD}} M : A \ \& \ \text{U} \notin \Gamma, A$.
 - $\Leftrightarrow \Gamma_0^M \vdash_{\cap}^{\text{CDZ}} M : 1$.
- (ii) $M \in \text{HN} \Leftrightarrow \forall \mathcal{T} \in \text{TT}^{\text{U}} \exists \Gamma \exists n, m \in \mathbb{N}. \Gamma \vdash_{\cap}^{\mathcal{T}} M : (\text{U}^m \rightarrow A)^n \rightarrow A$
 - $\Leftrightarrow \exists \Gamma, A. \Gamma \vdash_{\cap}^{\text{BCD}} M : A \ \& \ A \neq_{\text{BCD}} \text{U}$
 - $\Leftrightarrow \Gamma_0^M \vdash_{\cap}^{\text{DHM}} M : 1$.
- (iii) $M \in \text{SN} \Leftrightarrow \forall \mathcal{T} \in \text{TT} \exists \Gamma, A. \Gamma \vdash_{\cap}^{\mathcal{T}} M : A$.
 - $\Leftrightarrow \exists \Gamma, A. \Gamma \vdash_{\cap}^{\text{CDV}} M : A$.
 - $\Leftrightarrow \Gamma_0^M \vdash_{\cap}^{\text{HL}} M : 1$.

PROOF. We first prove (\Rightarrow) for (i)-(iii).

(i) By Corollary 14B.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 $\vec{M} \equiv M_1 \cdots M_m$. By the induction hypothesis we have $\Gamma_j \vdash_{\cap}^{\mathcal{T}} M_j : A_j$, for some Γ_j, A_j not containing U and for $j \leq m$. This implies that

$$\bigcup_{j \leq m} \Gamma_j \uplus \{x:A_1 \rightarrow \cdots \rightarrow A_m \rightarrow A\} \vdash_{\cap}^{\mathcal{T}} x\vec{M} : A.$$

Therefore, $\forall \mathcal{T} \in \text{TT}^{\text{U}} \exists \Gamma, A. \Gamma \vdash_{\cap}^{\mathcal{T}} M : A \ \& \ \text{U} \notin \Gamma, A$ in particular for $\mathcal{T} = \text{BCD}$.

For $\lambda_{\cap}^{\text{CDZ}}$ we also show by induction on M in normal form that $\Gamma_0^M \vdash M : 1$. If $M \equiv x\vec{M}$ then $\Gamma_0^M \vdash_{\cap}^{\text{CDZ}} M_j : 1$ by the induction hypothesis and weakening. As $0 = 1 \rightarrow 0$ in CDZ, this implies $\Gamma_0^M \vdash_{\cap}^{\text{CDZ}} x\vec{M} : 0$. By rule (\leq_{CDZ}) we conclude $\Gamma_0^M \vdash_{\cap}^{\text{CDZ}} M : 1$.

If $M \equiv \lambda y.N$ then by the induction hypothesis we have $\Gamma_0^M, y : 0 \vdash_{\cap}^{\text{CDZ}} N : 1$ and this implies $\Gamma_0^M \vdash_{\cap}^{\text{CDZ}} M : 0 \rightarrow 1$. Hence $\Gamma_0^M \vdash_{\cap}^{\text{CDZ}} M : 1$ by rule (\leq_{CDZ}) .

(ii) Again assume $M \equiv \lambda x_1 \cdots x_n.xM_1 \cdots M_m$ is in head normal form.

$$x:(U^m \rightarrow A) \vdash_{\cap}^{\mathcal{T}} xM_1 \cdots M_m : A, \quad \text{by } (\rightarrow E), \text{ hence}$$

$$x:(U^m \rightarrow A) \vdash_{\cap}^{\mathcal{T}} M : (U^m \rightarrow A)^n \rightarrow A, \quad \text{by } (\text{weakening}) \text{ and } (\rightarrow I).$$

As A is arbitrary we can get the type $\neq U$ in $\mathcal{T} = \text{BCD}$. We get

$$x:(U^m \rightarrow 0) \vdash_{\cap}^{\text{DHM}} M : (U^m \rightarrow 0)^n \rightarrow 0,$$

taking $\mathcal{T} = \text{DHM}$ and $A \equiv 0$. This implies

$$x:0 \vdash_{\cap}^{\text{DHM}} M : 1,$$

using $(U^m \rightarrow 0) =_{\text{DHM}} 0$, as $(U \rightarrow 0) =_{\text{DHM}} 0$, and $((U^m \rightarrow 0)^n \rightarrow 0) \leq_{\text{DHM}} 1$, as $0 \leq_{\text{DHM}} 1$ and $1 =_{\text{DHM}} 0 \rightarrow 1$.

(iii) By induction on the structure of strongly normalizing terms following Definition 17B.2. We only consider the case $M \equiv (\lambda x.R)N\vec{M}$ with $\vec{M} \equiv M_1 \cdots M_n$ and both $R[x := N]\vec{M}$ and N are strongly normalizing. By the induction hypothesis there are Γ, A, Γ', B such that $\Gamma \vdash_{\cap}^{\mathcal{T}} R[x := N]\vec{M} : A$ and $\Gamma' \vdash_{\cap}^{\mathcal{T}} N : B$. We get $\Gamma \uplus \Gamma' \vdash_{\cap}^{\mathcal{T}} R[x := N]\vec{M} : A$ and $\Gamma \uplus \Gamma' \vdash_{\cap}^{\mathcal{T}} N : B$, so if $n = 0$ we are done by Theorem 14B.4(i). If $n > 0$, then by iterated applications of Theorem 14A.1(ii) to $\Gamma \vdash_{\cap}^{\mathcal{T}} R[x := N]\vec{M} : A$ we obtain

$$\Gamma \vdash_{\cap}^{\mathcal{T}} R[x := N] : B_1^{(i)} \rightarrow \cdots \rightarrow B_n^{(i)} \rightarrow B^{(i)} \quad \Gamma \vdash_{\cap}^{\mathcal{T}} M_j : B_j^{(i)}, \quad (j \leq n)$$

and $\bigcap_{i \in I} B^{(i)} \leq_{\mathcal{T}} A$ for some $I, B_j^{(i)} (j \leq n), B^{(i)} \in \text{IT}^{\mathcal{T}}$. As in case $n = 0$ we obtain $\Gamma \uplus \Gamma' \vdash_{\cap}^{\mathcal{T}} (\lambda x.R)N : B_1^{(i)} \rightarrow \cdots \rightarrow B_m^{(i)} \rightarrow B^{(i)}$. So we can conclude $\Gamma \uplus \Gamma' \vdash_{\cap}^{\mathcal{T}} (\lambda x.R)N\vec{M} : A$. Finally, $\Gamma_0^M \vdash_{\cap}^{\text{HL}} M : 1$ follows from the observation that 1 is the top and 0 the bottom element in HL, see Lemma 13A.22(i).

(\Leftarrow) Now we show the converse implication. Let $\rho_0(x) = x$ for all $x \in V$.

(i) Suppose $\Gamma \vdash_{\cap}^{\text{BCD}} M : A$ and $U \notin A, \Gamma$. By soundness (Theorem 17A.4) it follows that $\Gamma \models_{\cap}^{\text{BCD}} M : A$. By Lemmas 17B.14 and 17B.11(ii) one has $\Lambda(\beta), \rho_0, \xi_{\text{BCD}}^2 \models \Gamma$. Hence, $M \in \llbracket A \rrbracket_{\xi_{\text{BCD}}^2} \subseteq \mathbb{N}$, again by Lemma 17B.11(ii).

Suppose $\Gamma_0^M \vdash_{\cap}^{\text{CDZ}} M : 1$. By soundness it follows that $\Gamma_0^M \models_{\cap}^{\text{CDZ}} M : 1$. By Lemmas 17B.14 and 17B.11(ii) one has $\Lambda(\beta), \rho_0, \xi_{\text{CDZ}} \models \Gamma$. Hence, $M \in \llbracket 1 \rrbracket_{\xi_{\text{CDZ}}} = \mathbb{N}$, by Definition 17B.10(iv).

(ii) Suppose $\Gamma \vdash_{\cap}^{\text{BCD}} M : A \neq U$. Then $\Gamma \models_{\cap}^{\text{BCD}} M : A$ by soundness. By Lemmas 17B.14 and 17B.11(i) one has $\Lambda(\beta), \rho_0, \xi_{\text{BCD}}^1 \models \Gamma$. Therefore we have $M \in \llbracket A \rrbracket_{\xi_{\text{BCD}}^1} \subseteq \mathbb{H}\mathbb{N}$, again by Lemma 17B.11(ii).

Suppose $\Gamma_0^M \vdash_{\cap}^{\text{DHM}} M : 1$. Again by soundness $\Gamma_0^M \models_{\cap}^{\text{DHM}} M : 1$. By Lemmas 17B.14 and 17B.11(i) one has $\Lambda(\beta), \rho_0, \xi_{\text{DHM}} \models \Gamma$. Hence, by Definition 17B.10(iii) one has $M \in \llbracket 1 \rrbracket_{\xi_{\text{DHM}}} = \mathbb{H}\mathbb{N}$.

(iii) Let $\Gamma \vdash_{\cap}^{\text{CDV}} M : A$. Again by soundness $\Gamma \models_{\cap}^{\text{CDV}} M : A$. By Lemmas 17B.14 and 17B.11(iii) one has $\Lambda(\beta), \rho_0, \xi_{\text{CDV}} \models \Gamma$. Hence $M \in \llbracket A \rrbracket_{\xi_{\text{CDV}}} \subseteq \text{SN}$, by Lemma 17B.11(iii). ■

17B.16. REMARK. (i) For $\mathcal{T} \in \text{TT}^U$ one has

$$\exists A, \Gamma, \Gamma \vdash_{\cap}^{\mathcal{T}} M : A \ \& \ U \neq_{\mathcal{T}} A, \Gamma \not\models M \in \text{HN}.$$

Take for example $\mathcal{T} = \text{Park}$, then $\vdash_{\cap}^{\text{Park}} \Omega : 0 \neq_{\text{Park}} U$, by Theorem 17D.3, but this term is unsolvable, hence without hnf, see B[1984].

(ii) There are many proofs of Theorem 17B.15(iii) in the literature: Pottinger [1980], 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, Dezani-Ciancaglini, and Zacchi [1987] persistently normalizing normal forms have been given a similar characterization using the notion of replaceable variable (Coppo, Dezani-Ciancaglini, and Zacchi [1987]). Other classes of terms are characterized in Dezani-Ciancaglini, Honsell, and Motohama [2005] and Tatsuta and Dezani-Ciancaglini [2006].

17C. 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], B[1984], Longo [1988], 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. B[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, Dezani-Ciancaglini, and Zacchi [1987], Honsell and Ronchi Della Rocca [1992], Dezani-Ciancaglini, Honsell, and Motohama [2005], Barendregt, Coppo, and Dezani-Ciancaglini [1983], Abramsky and Ong [1993] satisfy the Approximation Theorem. To

this end following [Dezani-Ciancaglini, Honsell, and Motohama \[2001\]](#) 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 \[2004\]](#) the approximation theorem is given also for the type assignment system $\lambda_{\cap}^{\text{EHR}}$ defined in Definition 13B.13.

For almost all the type theories of Fig. 33 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 $\lambda\perp$, see [B\[1984\]](#). The other extension is obtained by adding the constant Φ and is discussed in [Honsell and Ronchi Della Rocca \[1992\]](#).

17C.1. DEFINITION. (i) The set $\Lambda\perp$ of *$\lambda\perp$ -terms* is obtained by adding the constant *bottom*, notation \perp , 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 (\square_{\perp} and \square_L) from λ -terms to $\lambda\perp$ -terms and one mapping (\square_{Φ}) 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*.

17C.2. DEFINITION. The mappings $\square_{\perp} : \Lambda \rightarrow \Lambda\perp$, $\square_L : \Lambda \rightarrow \Lambda\perp$, $\square_{\Phi} : \Lambda \rightarrow \Lambda\Phi$ are inductively defined as follows.

$$\begin{aligned}\square(\lambda\vec{x}.y\vec{M}) &\triangleq \lambda\vec{x}.y\square(M_1)\cdots\square(M_m); \\ \square_{\perp}(\lambda\vec{x}.(\lambda y.R)N\vec{M}) &\triangleq \perp; \\ \square_L(\lambda\vec{x}.(\lambda y.R)N\vec{M}) &\triangleq \lambda\vec{x}.\perp; \\ \square_{\Phi}(\lambda\vec{x}.(\lambda y.R)N\vec{M}) &\triangleq \lambda\vec{x}.\Phi\square_{\Phi}(\lambda y.R)\square_{\Phi}(N)\square_{\Phi}(M_1)\cdots\square_{\Phi}(M_m),\end{aligned}$$

where $\square \in \{\square_{\perp}, \square_L, \square_{\Phi}\}$, $\vec{M} \equiv M_1 \cdots M_m$ and $m \geq 0$.

The mapping \square_{\perp} is related to the Böhm-tree of untyped lambda terms, whereas \square_L to the Lévy-Longo trees, see [van Bakel, Barbanera, Dezani-Ciancaglini, and de Vries \[2002\]](#), where these trees are related to intersection types.

In order to give the appropriate Approximation Theorem we will use the mapping \square_{\perp} for the type assignment systems $\lambda_{\cap}^{\text{Scott}}$, $\lambda_{\cap}^{\text{CDZ}}$, $\lambda_{\cap}^{\text{DHM}}$, $\lambda_{\cap}^{\text{BCD}}$, the mapping \square_L for the type assignment system $\lambda_{\cap}^{\text{AO}}$, and the mapping \square_{Φ} for the type assignment systems $\lambda_{\cap}^{\text{Park}}$,

$\lambda_{\cap}^{\text{HR}}$. Each one of the above mappings associates a set of approximants to each λ -term in the standard way.

17C.3. DEFINITION. Let $\mathcal{T} \in \{\text{Scott, Park, CDZ, HR, DHM, BCD, AO}\}$. The set $\mathcal{A}_{\mathcal{T}}(M)$ of \mathcal{T} -approximants of M is defined by

$$\mathcal{A}_{\mathcal{T}}(M) \triangleq \{P \mid \exists M'. M \rightarrow_{\beta} M' \text{ and } P \equiv \square(M')\},$$

where

$$\begin{aligned} \square &\triangleq \square_{\perp}, & \text{for } \mathcal{T} \in \{\text{Scott, CDZ, DHM, BCD}\}, \\ \square &\triangleq \square_L, & \text{for } \mathcal{T} \in \{\text{AO}\}, \\ \square &\triangleq \square_{\Phi}, & \text{for } \mathcal{T} \in \{\text{Park, HR}\}. \blacksquare \end{aligned}$$

We extend the typing to $\lambda\perp$ -terms and to $\lambda\Phi$ -terms by adding two different axioms for Φ and nothing for \perp .

17C.4. DEFINITION. (i) Let $\mathcal{T} \in \{\text{Scott, CDZ, DHM, BCD, AO}\}$. We extend the definition of type assignment $\Gamma \vdash_{\cap}^{\mathcal{T}} M : A$ to $\lambda\perp$ -terms by letting M, N in Definition 13B.3 range over $\Lambda\perp$.

(ii) We extend the type assignment $\lambda_{\cap}^{\text{Park}}$ to $\lambda\Phi$ -terms by adding the axiom
 $(\text{Ax-}\Phi\text{-Park}) \quad \Gamma \vdash_{\cap}^{\text{Park}} \Phi : 0.$

(iii) We extend the type assignment $\lambda_{\cap}^{\text{HR}}$ to $\lambda\Phi$ -terms by adding the axiom
 $(\text{Ax-}\Phi\text{-HR}) \quad \Gamma \vdash_{\cap}^{\text{HR}} \Phi : 1. \blacksquare$

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 Lemmas (Theorems 14A.1 and 14A.9) remain valid. In addition to these the following result is relevant.

17C.5. PROPOSITION. (i) Let $\mathcal{T} \in \{\text{Scott, CDZ, DHM, BCD, AO}\}$. Then

$$\Gamma \vdash_{\cap}^{\mathcal{T}} \perp : A \Leftrightarrow A =_{\mathcal{T}} \mathbb{U}.$$

- (ii) $\Gamma \vdash_{\cap}^{\text{Park}} \Phi : A \Leftrightarrow 0 \leq_{\text{Park}} A.$
- (iii) $\Gamma \vdash_{\cap}^{\text{HR}} \Phi : A \Leftrightarrow 1 \leq_{\text{HR}} A. \blacksquare$

17C.6. LEMMA. Let $\mathcal{T} \in \{\text{Scott, Park, CDZ, HR, DHM, BCD, AO}\}$.

- (i) $M_1 \rightarrow_{\beta} M_2 \& \Gamma \vdash_{\cap}^{\mathcal{T}} \square(M_1) : A \Rightarrow \Gamma \vdash_{\cap}^{\mathcal{T}} \square(M_2) : A.$
- (ii) If $P, P' \in \mathcal{A}_{\mathcal{T}}(M)$, $\Gamma \vdash_{\cap}^{\mathcal{T}} P : A$ and $\Gamma \vdash_{\cap}^{\mathcal{T}} P' : B$, then

$$\exists P'' \in \mathcal{A}_{\mathcal{T}}(M). \Gamma \vdash_{\cap}^{\mathcal{T}} P'' : A \cap B.$$

PROOF. (i). For $\mathcal{T} \in \{\text{Scott, CDZ, DHM, BCD, AO}\}$ the proof follows by induction on the structure of the term distinguishing cases (being or not in head normal form) and using the Inversion Lemmas.

For $\mathcal{T} \in \{\text{Park, HR}\}$ it suffices to consider the case $M_1 \equiv (\lambda x.M)N$ and $M_2 \equiv M[x := N]$. Notice that $\square_{\Phi}(M[x := N])$ is $\square_{\Phi}(M)$, where the occurrences of x have been replaced by $\Phi\square_{\Phi}(N)$ if they are functional and N is an abstraction, and by $\square_{\Phi}(N)$ otherwise. More formally, define the mapping $\overline{\square_{\Phi}} : \Lambda \rightarrow \Lambda\Phi$ by

$$\overline{\square_{\Phi}}(M) = \begin{cases} \Phi\square_{\Phi}(M) & \text{if } M \equiv \lambda x.M' \\ \square_{\Phi}(M) & \text{otherwise} \end{cases}$$

and the mapping $\{ \}_y^x : \Lambda \rightarrow \Lambda$ by

$$\begin{aligned}\{z\}_y^x &= z \\ \{M_1 M_2\}_y^x &= \begin{cases} y\{M_2\}_y^x & \text{if } M_1 \equiv x \\ \{M_1\}_y^x \{M_2\}_y^x & \text{otherwise} \end{cases} \\ \{\lambda z. M\}_y^x &= \lambda z. \{M\}_y^x.\end{aligned}$$

Then $\square_\Phi(M_1 M_2) = \overline{\square_\Phi}(M_1) \square_\Phi(M_2)$ and one can check, by induction on M , that $\square_\Phi(M[x := N]) \equiv \square_\Phi(\{M\}_y^x)[x := \square_\Phi(N)][y := \overline{\square_\Phi}(N)]$ for y fresh.

We may assume $A \neq \top \cup$. Then from $\Gamma \vdash_{\cap}^T \Phi(\lambda x. \square_\Phi(M)) \square_\Phi(N) : A$ we get $\Gamma \vdash_{\cap}^T \Phi(\lambda x. \square_\Phi(M)) : C \rightarrow A$, $\Gamma \vdash_{\cap}^T \square_\Phi(N) : C$ for some C , by Theorem 14A.9(ii). By Lemma 13A.24 we have $C \rightarrow A \neq \top \cup$, so again by Theorem 14A.9(ii) $\Gamma \vdash_{\cap}^T \Phi : B \rightarrow C \rightarrow A$, $\Gamma \vdash_{\cap}^T \lambda x. \square_\Phi(M) : B$, for some B .

For $T = \text{Park}$ we get $0 \leq_{\text{Park}} B \rightarrow C \rightarrow A$ from $\Gamma \vdash_{\cap}^{\text{Park}} \Phi : B \rightarrow C \rightarrow A$ by Proposition 17C.5(ii). This implies $B \leq_{\text{Park}} 0$, $C \leq_{\text{Park}} 0$, and $0 \leq_{\text{Park}} A$, since $0 =_{\tau} 0 \rightarrow 0$, since Park is β -sound by Theorem 14A.7, $(C \rightarrow A) \neq \top \cup$ and $A \neq \top \cup$. We obtain by rule (\leq) $\Gamma \vdash_{\cap}^{\text{Park}} \lambda x. \square_\Phi(M) : 0$ and $\Gamma \vdash_{\cap}^{\text{Park}} \square_\Phi(N) : 0$. We get $\Gamma, x:0 \vdash_{\cap}^{\text{Park}} \square_\Phi(M) : 0$ (by Theorem 14A.9 (iii)) and $\Gamma \vdash_{\cap}^{\text{Park}} \Phi \square_\Phi(N) : 0$ since $0 =_{\text{Park}} 0 \rightarrow 0$. Now $\square_\Phi(\{M\}_y^x)$ equals $\square_\Phi(M)$ with some occurrences of x replaced by the fresh variable y . Hence $\Gamma, y:0, x:0 \vdash_{\cap}^{\text{Park}} \square_\Phi(\{M\}_y^x) : 0$. So we conclude $\Gamma \vdash_{\cap}^{\text{Park}} \square_\Phi(\{M\}_y^x)[x := \square_\Phi(N)][y := \overline{\square_\Phi}(N)] : A$ by rules (*cut*) and (\leq) .

For $T = \text{HR}$ we get $1 \leq_{\text{HR}} B \rightarrow C \rightarrow A$ from $\Gamma \vdash_{\cap}^{\text{HR}} \Phi : B \rightarrow C \rightarrow A$ by Theorem 17C.5. This implies either $(B \leq_{\text{HR}} 1 \text{ and } 1 \leq_{\text{HR}} C \rightarrow A)$ or $(B \leq_{\text{HR}} 0 \text{ and } 0 \leq_{\text{HR}} C \rightarrow A)$ since $1 =_{\text{HR}} (1 \rightarrow 1) \cap (0 \rightarrow 0)$ and HR is β -sound by Theorem 14A.7, $C \rightarrow A \neq_{\text{HR}} \top \cup$ and $A \neq_{\text{HR}} \top \cup$ (notice that $1 \cap 0 = 0$). Similarly in the first case from $1 \leq_{\text{HR}} C \rightarrow A$ 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 $0 \leq_{\text{HR}} C \rightarrow A$ we get $C \leq_{\text{HR}} 1$ and $0 \leq_{\text{HR}} A$ since $0 =_{\text{HR}} 1 \rightarrow 0$.

To sum up, using rule (\leq) we have the following alternative cases.

- $\Gamma \vdash_{\cap}^{\text{HR}} \lambda x. \square_\Phi(M) : 1$, $\Gamma \vdash_{\cap}^{\text{HR}} \square_\Phi(N) : 1$, and $1 \leq_{\text{HR}} A$;
- $\Gamma \vdash_{\cap}^{\text{HR}} \lambda x. \square_\Phi(M) : 1$, $\Gamma \vdash_{\cap}^{\text{HR}} \square_\Phi(N) : 0$, and $0 \leq_{\text{HR}} A$;
- $\Gamma \vdash_{\cap}^{\text{HR}} \lambda x. \square_\Phi(M) : 0$, $\Gamma \vdash_{\cap}^{\text{HR}} \square_\Phi(N) : 1$, and $0 \leq_{\text{HR}} A$.

From Theorem 14A.9 (iii) we get alternatively:

- $\Gamma, x:1 \vdash_{\cap}^{\text{HR}} \square_\Phi(M) : 1$, and $\Gamma \vdash_{\cap}^{\text{HR}} \Phi \square_\Phi(N) : 1$;
- $\Gamma, x:0 \vdash_{\cap}^{\text{HR}} \square_\Phi(M) : 0$, and $\Gamma \vdash_{\cap}^{\text{HR}} \Phi \square_\Phi(N) : 0$;
- $\Gamma, x:1 \vdash_{\cap}^{\text{HR}} \square_\Phi(M) : 0$, and $\Gamma \vdash_{\cap}^{\text{HR}} \Phi \square_\Phi(N) : 1$,

so we can conclude as in the previous case.

(ii). By hypotheses there are M_1, M_2 such that $M \rightarrow_{\beta} M_1$, $M \rightarrow_{\beta} M_2$ and $P \equiv \square(M_1)$, $P' \equiv \square(M_2)$. By the Church-Rosser property of \rightarrow_{β} we can find M_3 such that $M_1 \rightarrow_{\beta} M_3$ and $M_2 \rightarrow_{\beta} M_3$. By (i) we can choose $P'' \equiv \square(M_3)$. ■

Approximation Theorem - Part 1

It is useful to introduce the following definition.

17C.7. DEFINITION. Let $T \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{BCD}, \text{AO}\}$. Write

$$[A]_{\Gamma}^{\mathcal{T}} = \{M \mid \exists P \in \mathcal{A}_{\mathcal{T}}(M). \Gamma \vdash_{\cap}^{\mathcal{T}} P : A\}.$$

By definition we get that $M \in [A]_{\Gamma}^{\mathcal{T}}$ and $N \rightarrow_{\beta} M$ imply $N \in [A]_{\Gamma}^{\mathcal{T}}$. Moreover $\Gamma \sqsubseteq \Gamma'$ implies $[A]_{\Gamma}^{\mathcal{T}} \subseteq [A]_{\Gamma'}^{\mathcal{T}}$ for all types $A \in \mathbb{T}^{\mathcal{T}}$.

In this subsection we prove that, if $M \in [A]_{\Gamma}^{\mathcal{T}}$, then there exists a derivation of $\Gamma \vdash_{\cap}^{\mathcal{T}} M : A$.

17C.8. PROPOSITION. *Let $\mathcal{T} \in \{\text{Scott, Park, CDZ, HR, DHM, BCD, AO}\}$.*

$$M \in [A]_{\Gamma}^{\mathcal{T}} \Rightarrow \Gamma \vdash_{\cap}^{\mathcal{T}} M : A.$$

PROOF. Write $P \equiv \square(M)$ with $\square = \square_{\perp}$ for $\mathcal{T} \in \{\text{Scott, CDZ, DHM, BCD}\}$,

$$\square = \square_L \quad \text{for } \mathcal{T} = \text{AO},$$

$$\square = \square_{\Phi} \quad \text{for } \mathcal{T} \in \{\text{Park, HR}\}.$$

By Corollary 14B.5 (ii) it is sufficient to show that for all mentioned \mathcal{T} one has

$$(7) \quad \Gamma \vdash_{\cap}^{\mathcal{T}} P : A \Rightarrow \Gamma \vdash_{\cap}^{\mathcal{T}} M : A.$$

For $\vdash_{\cap}^{\mathcal{T}}$ just write \vdash in this proof.

For $\mathcal{T} \in \{\text{Scott, CDZ, DHM, BCD, AO}\}$ the implication (7) follows from Proposition 17C.5(i) and the definition of the mappings \square_{\perp} and \square_L .

For $\mathcal{T} \in \{\text{Park, HR}\}$ we prove (7) by induction on M , assuming $A \neq_{\mathcal{T}} \mathbb{U}$.

Case $M \equiv x$. Trivial.

Case $M \equiv \lambda x.M'$. Then $P \equiv \lambda x.P'$ where $P' \equiv \square_{\Phi}(M')$. By Theorem 14A.1(iii) from $\Gamma \vdash P : A$ we get $\Gamma, x:B_i \vdash P' : C_i$ and $\bigcap_{i \in I} (B_i \rightarrow C_i) \leq A$ 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 $(\rightarrow I)$, $(\cap I)$ and (\leq) .

Case $M \equiv M_1 M_2$, where M_1 is not an abstraction. Then $P \equiv P_1 P_2$ where $P_1 \equiv \square_{\Phi}(M_1)$ and $P_2 \equiv \square_{\Phi}(M_2)$. By Theorem 14A.9(ii) from $\Gamma \vdash P : A$ we get $\Gamma \vdash P_1 : B \rightarrow A$, $\Gamma \vdash P_2 : B$ for some B . By induction this implies $\Gamma \vdash M_1 : B \rightarrow A$ 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 $P_1 \equiv \square_{\Phi}(M_1)$ and $P_2 \equiv \square_{\Phi}(M_2)$. As in the proof of Lemma 17C.6(i) from $\Gamma \vdash P : A$, where $A \neq_{\mathcal{T}} \mathbb{U}$, we get $\Gamma \vdash \Phi : B \rightarrow C \rightarrow A$, $\Gamma \vdash P_1 : B$, $\Gamma \vdash P_2 : C$ 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 17C.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_{Park}) and $(\rightarrow E)$ since $0 =_{\text{Park}} 0 \rightarrow 0$.

For $\mathcal{T} = \text{HR}$ as in the proof of Lemma 17C.6(i) we have the following alternative cases.

- $\Gamma \vdash_{\cap}^{\text{HR}} M_1 : 1$, $\Gamma \vdash_{\cap}^{\text{HR}} M_2 : 1$, and $1 \leq_{\text{HR}} A$;
- $\Gamma \vdash_{\cap}^{\text{HR}} M_1 : 1$, $\Gamma \vdash_{\cap}^{\text{HR}} M_2 : 0$, and $0 \leq_{\text{HR}} A$;
- $\Gamma \vdash_{\cap}^{\text{HR}} M_1 : 0$, $\Gamma \vdash_{\cap}^{\text{HR}} M_2 : 1$, 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→0) using rules (\leq_{HR}) and $(\rightarrow E)$. ■

Approximation Theorem - Part 2

In order to prove the converse of Proposition 17C.8 we will use a Kripke-like version of stable sets [Mitchell \[1996\]](#). First we need a technical result.

17C.9. LEMMA. Let $\mathcal{T} \in \{\text{Scott, Park, CDZ, HR, DHM, BCD, AO}\}$. Write $\Gamma' = \Gamma, z : B$, with $z \notin \text{FV}(M)$, and assume $A \neq_{\mathcal{T}} \mathbb{U}$ for $\mathcal{T} = \text{AO}$. Then

$$Mz \in [A]_{\Gamma'}^{\mathcal{T}} \Rightarrow M \in [B \rightarrow A]_{\Gamma}^{\mathcal{T}}.$$

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 $\Gamma \vdash \hat{P} : B \rightarrow A$.

There are two possibilities.

- $Mz \rightarrow_{\beta} M'z$ and $P \equiv \square(M'z)$;
- $Mz \rightarrow_{\beta} (\lambda x.M')z \rightarrow_{\beta} M'[x := z]$ and $P \in \mathcal{A}_{\mathcal{T}}(M'[x := z])$.

In the first case again there are two possibilities.

- $M' \equiv yM_1 \cdots M_m$, $m \geq 0$;
- $M' \equiv (\lambda y.M_0)M_1 \cdots M_m$, $m \geq 0$.

In total there are four cases:

- $P \equiv P'z$ and $P' \equiv y\square(M_1) \cdots \square(M_m) \in \mathcal{A}_{\mathcal{T}}(M)$;
- $P \equiv \perp$ and $\mathcal{T} \in \{\text{Scott, CDZ, DHM, BCD, AO}\}$;
- $P \equiv \Phi P'z$, $P' \equiv \square(\lambda y.M_0)\square(M_1) \cdots \square(M_m) \in \mathcal{A}_{\mathcal{T}}(M)$ and $\mathcal{T} \in \{\text{Park, HR}\}$;
- $M \rightarrow_{\beta} \lambda x.M'$ 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 $A =_{\mathcal{T}} \mathbb{U}$ because by assumption $\mathcal{T} \neq \text{AO}$, hence we have $(\mathbb{U} \rightarrow)$. Now let $A \neq_{\mathcal{T}} \mathbb{U}$. Then by Theorem 14A.9(ii) from $\Gamma' \vdash P : A$ we get $\Gamma' \vdash P' : C \rightarrow A$, $\Gamma' \vdash z : C$ for some C . By Theorem 14A.9(i) $B \leq C$ and we conclude using (\leq) and *(strengthening)* $\Gamma \vdash P' : B \rightarrow A$.

Case $P \equiv \perp$. By Proposition 17C.5(i) $A =_{\mathcal{T}} \mathbb{U}$. By assumption, $\mathcal{T} \neq \text{AO}$. Hence, we have rule $(\mathbb{U} \rightarrow)$.

Case $P \equiv \Phi P'z$, where $P' \in \mathcal{A}_{\mathcal{T}}(M)$ and $\mathcal{T} \in \{\text{Park, HR}\}$. Now we show that we can choose $\hat{P} \equiv P'$. Again let $A \neq_{\mathcal{T}} \mathbb{U}$. Then from $\Gamma' \vdash P : A$ we get by Theorem 14A.9(ii) and (i) $\Gamma' \vdash \Phi : C \rightarrow D \rightarrow A$, and $\Gamma' \vdash P' : C$, and $\Gamma' \vdash z : D$, for some C, D with $B \leq D$. For $\mathcal{T} = \text{Park}$, using Proposition 17C.5(ii) as in the proof of Lemma 17C.6(i), we get $C \leq_{\text{Park}} 0$, $D \leq_{\text{Park}} 0$, and $0 \leq_{\text{Park}} A$ (remember that $0 =_{\text{Park}} 0 \rightarrow 0$). Similarly for $\mathcal{T} = \text{HR}$, using Proposition 17C.5(iii), we get either $C \leq_{\text{HR}} 1$, $D \leq_{\text{HR}} 1$, and $1 \leq_{\text{HR}} A$ or $C \leq_{\text{HR}} 1, D \leq_{\text{HR}} 0$, and $0 \leq_{\text{HR}} A$ or $C \leq_{\text{HR}} 0, D \leq_{\text{HR}} 1$, and $0 \leq_{\text{Park}} A$ (remember that $1 =_{\text{HR}} (1 \rightarrow 1) \cap (0 \rightarrow 0)$ and $0 =_{\text{HR}} 1 \rightarrow 0$). In all cases we can conclude $C \leq D \rightarrow A \leq B \rightarrow A$ and therefore by (\leq) and *(strengthening)* $\Gamma \vdash P' : B \rightarrow A$.

Case $M \rightarrow_{\beta} \lambda x.M'$ and $P \in \mathcal{A}_{\mathcal{T}}(M'[x := z])$. If $\square = \square_{\perp}$ and $P \equiv \perp$, 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.⁸ 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,

⁸As already observed in Berline [2000] we cannot use here stable sets as we did in Section 17B since we need to take into account also the \mathcal{T} -bases, not only the λ -terms and their types.

since we have different cases for the type 0 and for arrow types according to the different type theories under consideration.

17C.10. DEFINITION (Kripke type interpretation).

Let $\mathcal{T} \in \{\text{Scott, Park, CDZ, HR, DHM, BCD, AO}\}$. Define $\llbracket A \rrbracket_{\Gamma}^{\mathcal{T}}$ for $A \in \mathbb{T}^{\mathcal{T}}$

$$\begin{aligned}\llbracket \alpha \rrbracket_{\Gamma}^{\mathcal{T}} &\triangleq \llbracket \alpha \rrbracket_{\Gamma}^{\mathcal{T}}, & \text{for } \alpha \in \mathbb{A}_{\infty} \cup \{\mathbf{U}, 1\}; \\ \llbracket 0 \rrbracket_{\Gamma}^{\mathcal{T}} &\triangleq \begin{cases} \{M \mid \forall \vec{N}. M\vec{N} \in [0]_{\Gamma}^{\mathcal{T}}\}, & \text{for } \mathcal{T} \in \{\text{Scott, DHM}\}; \\ \{M \mid \forall \Gamma' \not\sqsupseteq \Gamma \forall \vec{N} \in [1]_{\Gamma'}^{\mathcal{T}}. M\vec{N} \in [0]_{\Gamma'}^{\mathcal{T}}\}, & \text{for } \mathcal{T} \in \{\text{CDZ, HR}\}; \\ [0]_{\Gamma}^{\mathcal{T}}, & \text{for } \mathcal{T} = \text{Park}; \end{cases} \\ \llbracket A \rightarrow B \rrbracket_{\Gamma}^{\mathcal{T}} &\triangleq \begin{cases} \{M \mid \forall \Gamma' \not\sqsupseteq \Gamma \forall N \in \llbracket A \rrbracket_{\Gamma'}^{\mathcal{T}}. MN \in \llbracket B \rrbracket_{\Gamma'}^{\mathcal{T}}\}, & \text{if } \mathcal{T} \neq \text{AO or } B \neq_{\text{AO}} \mathbf{U}; \\ [A \rightarrow B]_{\Gamma}^{\mathcal{T}}, & \text{if } \mathcal{T} = \text{AO \& } B =_{\text{AO}} \mathbf{U}; \end{cases} \\ \llbracket A \cap B \rrbracket_{\Gamma}^{\mathcal{T}} &\triangleq \llbracket A \rrbracket_{\Gamma}^{\mathcal{T}} \cap \llbracket B \rrbracket_{\Gamma}^{\mathcal{T}}.\end{aligned}$$

The definition of $\llbracket A \rightarrow B \rrbracket_{\Gamma}^{\mathcal{T}}$ is somewhat involved in order to make Lemma 17C.12(ii) valid for $\mathcal{T} = \text{AO}$.

17C.11. PROPOSITION. (i) $M \in \llbracket A \rrbracket_{\Gamma}^{\mathcal{T}} \& N \rightarrow_B M \Rightarrow N \in \llbracket A \rrbracket_{\Gamma}^{\mathcal{T}}$.

(ii) $\Gamma \not\sqsubseteq \Gamma' \Rightarrow \llbracket A \rrbracket_{\Gamma}^{\mathcal{T}} \subseteq \llbracket A \rrbracket_{\Gamma'}^{\mathcal{T}}$, for all types $A \in \mathbb{T}^{\mathcal{T}}$.

PROOF. Easy. ■

The Lemmas 17C.12, 17C.15 and the final theorem are standard.

17C.12. LEMMA. Let $\mathcal{T} \in \{\text{Scott, Park, CDZ, HR, DHM, BCD, AO}\}$. Then

(i) $x\vec{M} \in [A]_{\Gamma}^{\mathcal{T}} \Rightarrow x\vec{M} \in \llbracket A \rrbracket_{\Gamma}^{\mathcal{T}}$.

(ii) $\llbracket A \rrbracket_{\Gamma}^{\mathcal{T}} \subseteq [A]_{\Gamma}^{\mathcal{T}}$.

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 $\Gamma' \not\sqsupseteq \Gamma$ and $\vec{N} \in [1]_{\Gamma'}^{\text{CDZ}}$.

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

$$\begin{aligned}x\vec{M} \in [0]_{\Gamma}^{\text{CDZ}} &\Rightarrow x\vec{M}\vec{N} \in [0]_{\Gamma'}^{\text{CDZ}} \quad \text{by rules } (\leq_{\text{CDZ}}) \text{ and } (\rightarrow E) \\ &\quad \text{since } 0 =_{\text{CDZ}} 1 \rightarrow 0, \\ &\Rightarrow x\vec{M} \in \llbracket 0 \rrbracket_{\Gamma}^{\text{CDZ}} \quad \text{by Definition 17C.10.}\end{aligned}$$

Case $A \equiv B \rightarrow C$. Let $\Gamma' \not\sqsupseteq \Gamma$ and $\mathcal{T} \neq \text{AO}$ or $C \neq_{\text{AO}} \mathbf{U}$ and let $N \in \llbracket B \rrbracket_{\Gamma'}^{\mathcal{T}}$.

$\llbracket B \rrbracket_{\Gamma'}^{\mathcal{T}} \subseteq [B]_{\Gamma'}^{\mathcal{T}}$ by induction on (ii). Hence

$$\begin{aligned}x\vec{M} \in [A]_{\Gamma}^{\mathcal{T}} &\Rightarrow x\vec{M}N \in [C]_{\Gamma'}^{\mathcal{T}} \quad \text{by rule } (\rightarrow E), \\ &\Rightarrow x\vec{M}N \in \llbracket C \rrbracket_{\Gamma'}^{\mathcal{T}} \quad \text{by induction on (i),} \\ &\Rightarrow x\vec{M} \in \llbracket B \rightarrow C \rrbracket_{\Gamma}^{\mathcal{T}} \quad \text{by Definition 17C.10.}\end{aligned}$$

(ii) Case $A \equiv B \rightarrow C$ and $\mathcal{T} \neq \text{AO}$ or $C \neq_{\text{AO}} \text{U}$. Let $\Gamma' = \Gamma, z:B$ with z fresh, and suppose $M \in \llbracket B \rightarrow C \rrbracket_{\Gamma}^{\mathcal{T}}$; as $z \in \llbracket B \rrbracket_{\Gamma, z:B}^{\mathcal{T}}$ by induction on (i), we have

$$\begin{aligned} M \in \llbracket B \rightarrow C \rrbracket_{\Gamma}^{\mathcal{T}} \& z \in \llbracket B \rrbracket_{\Gamma, z:B}^{\mathcal{T}} \Rightarrow Mz \in \llbracket C \rrbracket_{\Gamma'}^{\mathcal{T}} && \text{by Definition 17C.10,} \\ &\Rightarrow Mz \in \llbracket C \rrbracket_{\Gamma'}^{\mathcal{T}} && \text{by induction on (ii),} \\ &\Rightarrow M \in \llbracket B \rightarrow C \rrbracket_{\Gamma}^{\mathcal{T}} && \text{by Lemma 17C.9.} \end{aligned}$$

Case $A \equiv B \cap C$. This follows from $\llbracket B \cap C \rrbracket_{\Gamma}^{\mathcal{T}} = \llbracket B \rrbracket_{\Gamma}^{\mathcal{T}} \cap \llbracket C \rrbracket_{\Gamma}^{\mathcal{T}}$ and the induction hypothesis using Lemma 17C.6(ii). ■

The following lemma essentially states that the Kripke type interpretations agree with the corresponding type theories.

17C.13. LEMMA. (i) *Let $\mathcal{T} \in \{\text{CDZ}, \text{DHM}\}$. Then*

$$M \in [A]_{\Gamma, z:0}^{\mathcal{T}} \& N \in [0]_{\Gamma}^{\mathcal{T}} \Rightarrow M[z := N] \in [A]_{\Gamma}^{\mathcal{T}}.$$

(ii) *Let $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{BCD}, \text{AO}\}$. Then*

$$\forall A, B \in \mathbb{T}^{\mathcal{T}} [A \leq_{\mathcal{T}} B \Rightarrow \llbracket A \rrbracket_{\Gamma}^{\mathcal{T}} \subseteq \llbracket B \rrbracket_{\Gamma}^{\mathcal{T}}].$$

PROOF. (i) We may assume $A \neq_{\mathcal{T}} \text{U}$.

Let $\mathcal{T} = \text{CDZ}$. If $M \in [A]_{\Gamma, z:0}^{\text{CDZ}}$, then there is a $P \in \mathcal{A}_{\text{CDZ}}(M)$ such that $\Gamma, z:0 \vdash_{\cap}^{\text{CDZ}} P : A$. This is proved by induction on P .

Case $P \equiv \perp$. Trivial.

Case $P \equiv \lambda x.P'$. Then $M \rightsquigarrow_{\beta} \lambda x.M'$ and $P' \in \mathcal{A}_{\text{CDZ}}(M')$. From $\Gamma, z:0 \vdash_{\cap}^{\text{CDZ}} P : A$ we get $\Gamma, z:0, x:B_i \vdash_{\cap}^{\text{CDZ}} P' : C_i$ and $\bigcap_{i \in I} (B_i \rightarrow C_i) \leq_{\text{CDZ}} A$ for some I and $B_i, C_i \in \mathbb{T}^{\text{CDZ}}$ by Theorem 14A.1(iii). By induction for each $i \in I$ there is a $P_i \in \mathcal{A}_{\text{CDZ}}(M'[z := N])$ such that $\Gamma, x:B_i \vdash_{\cap}^{\text{CDZ}} P_i : C_i$. Let $P_i = \square(M_i)$, where $M'[z := N] \rightsquigarrow_{\beta} M_i$ and let M'' be a common reduct of the M_i and $P'' \equiv \square(M'')$. Then $P'' \in \mathcal{A}_{\text{CDZ}}(M'[z := N])$ and $\Gamma, x:B_i \vdash_{\cap}^{\text{CDZ}} P'' : C_i$, for all $i \in I$, by Lemma 17C.6(i). Clearly $\lambda x.P'' \in \mathcal{A}_{\text{CDZ}}(M[z := N])$ and by construction $\Gamma \vdash_{\cap}^{\text{CDZ}} \lambda x.P'' : A$.

Case $P \equiv x\vec{P}$, then $M \rightsquigarrow_{\beta} x\vec{M}$ and $\vec{P} \in \mathcal{A}_{\text{CDZ}}(\vec{M})$. From $\Gamma, z:0 \vdash_{\cap}^{\text{CDZ}} P : A$ we get $\Gamma, z:0 \vdash_{\cap}^{\text{CDZ}} x : \vec{B} \rightarrow A$ and $\Gamma, z:0 \vdash_{\cap}^{\text{CDZ}} \vec{P} : \vec{B}$ by Theorem 14A.9(ii) and Lemma 13A.24. By induction there are $\vec{P}' \in \mathcal{A}_{\text{CDZ}}(\vec{M}[z := N])$ such that $\Gamma \vdash_{\cap}^{\text{CDZ}} \vec{P}' : \vec{B}$. If $x \neq z$ we are done since $x\vec{P}' \in \mathcal{A}_{\text{CDZ}}(M[z := N])$ and we can derive $\Gamma \vdash_{\cap}^{\text{CDZ}} x\vec{P}' : A$ using $(\rightarrow E)$. Otherwise $\Gamma, z:0 \vdash_{\cap}^{\text{CDZ}} z : \vec{B} \rightarrow A$ implies $0 \leq_{\text{CDZ}} \vec{B} \rightarrow A$ by Theorem 14A.9(i). Being CDZ β -sound by Theorem 14A.7 from $0 =_{\text{CDZ}} \vec{1} \rightarrow 0$ we obtain $\vec{B} \leq_{\text{CDZ}} \vec{1}$ and $0 \leq_{\text{CDZ}} A$ by Lemma 13A.24. So we get $\Gamma \vdash_{\cap}^{\text{CDZ}} \vec{P}' : \vec{1}$, i.e. $\vec{M}[z := N] \in [1]_{\Gamma}^{\text{CDZ}}$. Now

$$N \in [0]_{\Gamma}^{\text{CDZ}} \& \vec{M}[z := N] \in [1]_{\Gamma}^{\text{CDZ}} \Rightarrow M[z := N] \in [0]_{\Gamma}^{\text{CDZ}},$$

by Definition 17C.10. Since $0 \leq_{\text{CDZ}} A$ we get $M[z := N] \in [A]_{\Gamma}^{\text{CDZ}}$.

Let $\mathcal{T} = \text{DHM}$. Then the proof is similar but easier. In the case $P \equiv z\vec{P}$ it follows from Definition 17C.10 that $N \in [0]_{\Gamma}^{\text{DHM}} \Rightarrow M[z := N] \in [A]_{\Gamma}^{\text{DHM}}$.

(ii) We treat the cases related to $A \rightarrow B \leq \text{U} \rightarrow \text{U}$ in AO, $(0 \rightarrow 1) = 1$, $0 = (1 \rightarrow 0)$ in CDZ, $(1 \rightarrow 1) \cap (0 \rightarrow 0) = 1$ in HR, and $(0 \rightarrow 0) = 0$ in Park.

Proof of $\llbracket A \rightarrow B \rrbracket_{\Gamma}^{\text{AO}} \subseteq \llbracket U \rightarrow U \rrbracket_{\Gamma}^{\text{AO}}$. If $B =_{\text{AO}} U$, then

$$\llbracket A \rightarrow B \rrbracket_{\Gamma}^{\text{AO}} = \llbracket A \rightarrow B \rrbracket_{\Gamma}^{\text{AO}} \subseteq \llbracket U \rightarrow U \rrbracket_{\Gamma}^{\text{AO}} = \llbracket U \rightarrow U \rrbracket_{\Gamma}^{\text{AO}}.$$

If, on the other hand, $B \neq_{\text{AO}} U$, then $M \in \llbracket A \rightarrow B \rrbracket_{\Gamma}^{\text{AO}}$. Write $\Gamma' = \Gamma, z:A$. Then $z \in [A]_{\Gamma'}^{\text{AO}}$, hence by Lemma 17C.12(i) $z \in \llbracket A \rrbracket_{\Gamma'}^{\text{AO}}$. So $Mz \in \llbracket B \rrbracket_{\Gamma'}^{\text{AO}} \subseteq [B]_{\Gamma'}^{\text{AO}}$, by Lemma 17C.12(ii), and therefore $M \in \llbracket A \rightarrow B \rrbracket_{\Gamma}^{\text{AO}} \subseteq \llbracket U \rightarrow U \rrbracket_{\Gamma}^{\text{AO}} = \llbracket U \rightarrow U \rrbracket_{\Gamma}^{\text{AO}}$, by Lemma 17C.9.

Proof of $\llbracket 0 \rightarrow 1 \rrbracket_{\Gamma}^{\text{CDZ}} \subseteq \llbracket 1 \rrbracket_{\Gamma}^{\text{CDZ}}$. We have

$$\begin{aligned} \llbracket 0 \rightarrow 1 \rrbracket_{\Gamma}^{\text{CDZ}} &\subseteq \llbracket 0 \rightarrow 1 \rrbracket_{\Gamma}^{\text{CDZ}}, && \text{by Lemma 17C.12(ii),} \\ &= \llbracket 1 \rrbracket_{\Gamma}^{\text{CDZ}}, && \text{since } 0 \rightarrow 1 =_{\text{CDZ}} 1, \\ &= \llbracket 1 \rrbracket_{\Gamma}^{\text{CDZ}}, && \text{by Definition 17C.10.} \end{aligned}$$

Proof of $\llbracket 1 \rrbracket_{\Gamma}^{\text{CDZ}} \subseteq \llbracket 0 \rightarrow 1 \rrbracket_{\Gamma}^{\text{CDZ}}$. Suppose $\Gamma' \not\supseteq \Gamma$, $M \in \llbracket 1 \rrbracket_{\Gamma}^{\text{CDZ}}$ and $N \in \llbracket 0 \rrbracket_{\Gamma'}^{\text{CDZ}}$, in order to show $MN \in \llbracket 1 \rrbracket_{\Gamma'}^{\text{CDZ}}$. By Definition 17C.10 $\llbracket 1 \rrbracket_{\Gamma}^{\text{CDZ}} = \llbracket 1 \rrbracket_{\Gamma}^{\text{CDZ}}$. If $M \in \llbracket 1 \rrbracket_{\Gamma}^{\text{CDZ}}$, then there is $P \in \mathcal{A}_{\text{CDZ}}(M)$ such that $\Gamma \vdash_{\cap}^{\text{CDZ}} P : 1$. We will show $MN \in \llbracket 1 \rrbracket_{\Gamma'}^{\text{CDZ}}$ by distinguishing cases of P .

Case $P \equiv \perp$. By Proposition 17C.5(i) one has $1 =_{\text{CDZ}} U$, contradicting Proposition 13A.28. So this case is impossible.

Case $P \equiv \lambda z.P'$. Then $M \rightarrow_{\beta} \lambda z.M'$ and $P' \in \mathcal{A}_{\text{CDZ}}(M')$. From $\Gamma \vdash_{\cap}^{\text{CDZ}} P : 1$ we get $\Gamma, z:0 \vdash_{\cap}^{\text{CDZ}} P' : 1$ by Theorem 14A.9(iii), since $1 =_{\text{CDZ}} 0 \rightarrow 1$. This implies $M' \in \llbracket 1 \rrbracket_{\Gamma, z:0}^{\text{CDZ}}$. We may assume that $z \notin \text{dom}(\Gamma')$. Then also $M' \in \llbracket 1 \rrbracket_{\Gamma', z:0}^{\text{CDZ}}$. Therefore

$$\begin{aligned} MN &\rightarrow_{\beta} (\lambda z.M')N \\ &\rightarrow_{\beta} M'[z := N] \\ &\in \llbracket 1 \rrbracket_{\Gamma'}^{\text{CDZ}}, && \text{by (i),} \\ &= \llbracket 1 \rrbracket_{\Gamma'}^{\text{CDZ}}. \end{aligned}$$

Case $P \equiv x\vec{P}$. Notice that $\Gamma \vdash_{\cap}^{\text{CDZ}} P : 1$ implies $\Gamma \vdash_{\cap}^{\text{CDZ}} P : 0 \rightarrow 1$, since $1 =_{\text{CDZ}} 0 \rightarrow 1$. By Lemma 17C.12(ii) $\llbracket 0 \rrbracket_{\Gamma'}^{\text{CDZ}} \subseteq \llbracket 0 \rrbracket_{\Gamma'}^{\text{CDZ}}$, hence there is $P' \in \mathcal{A}_{\text{CDZ}}(N)$ such that $\Gamma' \vdash_{\cap}^{\text{CDZ}} P' : 0$. We get $\Gamma' \vdash_{\cap}^{\text{CDZ}} PP' : 1$. As $PP' \in \mathcal{A}_{\text{CDZ}}(MN)$ we conclude that $MN \in \llbracket 1 \rrbracket_{\Gamma'}^{\text{CDZ}}$.

Proof of $\llbracket 1 \rightarrow 0 \rrbracket_{\Gamma}^{\text{CDZ}} \subseteq \llbracket 0 \rrbracket_{\Gamma}^{\text{CDZ}}$. We have $\llbracket 1 \rightarrow 0 \rrbracket_{\Gamma}^{\text{CDZ}} \subseteq \llbracket 1 \rightarrow 0 \rrbracket_{\Gamma}^{\text{CDZ}}$, by Lemma 17C.12(ii) and $\llbracket 1 \rightarrow 0 \rrbracket_{\Gamma}^{\text{CDZ}} = \llbracket 0 \rrbracket_{\Gamma}^{\text{CDZ}}$, as $1 \rightarrow 0 =_{\text{CDZ}} 0$, using Definition 17C.7. Moreover using Definition 17C.10 it follows that

$$\begin{aligned} \llbracket 1 \rightarrow 0 \rrbracket_{\Gamma}^{\text{CDZ}} &= \{M \mid \forall \Gamma' \not\supseteq \Gamma, \forall N \in \llbracket 1 \rrbracket_{\Gamma'}^{\text{CDZ}}. MN \in \llbracket 0 \rrbracket_{\Gamma'}^{\text{CDZ}}\} \\ &= \{M \mid \forall \Gamma' \not\supseteq \Gamma, \forall N \in \llbracket 1 \rrbracket_{\Gamma'}^{\text{CDZ}}. MN \in \llbracket 0 \rrbracket_{\Gamma'}^{\text{CDZ}}\} \\ &\subseteq \{M \mid \forall \Gamma' \not\supseteq \Gamma, \forall N, \vec{N} \in \llbracket 1 \rrbracket_{\Gamma'}^{\text{CDZ}}. MN\vec{N} \in \llbracket 0 \rrbracket_{\Gamma'}^{\text{CDZ}}\}. \end{aligned}$$

From $\llbracket 1 \rightarrow 0 \rrbracket_{\Gamma}^{\text{CDZ}} \subseteq \llbracket 0 \rrbracket_{\Gamma}^{\text{CDZ}}$ and

$$\llbracket 1 \rightarrow 0 \rrbracket_{\Gamma}^{\text{CDZ}} \subseteq \{M \mid \forall \Gamma' \not\supseteq \Gamma, \forall N, \vec{N} \in [1]_{\Gamma'}^{\text{CDZ}}. MN\vec{N} \in [0]_{\Gamma'}^{\text{CDZ}}\}$$

we can conclude

$$\llbracket 1 \rightarrow 0 \rrbracket_{\Gamma}^{\text{CDZ}} \subseteq \{M \mid \forall \Gamma' \not\supseteq \Gamma, \vec{N} \in [1]_{\Gamma'}^{\text{CDZ}}. MN\vec{N} \in [0]_{\Gamma'}^{\text{CDZ}}\} = \llbracket 0 \rrbracket_{\Gamma}^{\text{CDZ}}.$$

Proof of $\llbracket 0 \rrbracket_{\Gamma}^{\text{CDZ}} \subseteq \llbracket 1 \rightarrow 0 \rrbracket_{\Gamma}^{\text{CDZ}}$. Again using Definition 17C.10 one has

$$\begin{aligned} M \in \llbracket 0 \rrbracket_{\Gamma}^{\text{CDZ}} &\Rightarrow \forall \Gamma' \not\supseteq \Gamma, \forall N, \vec{N} \in [1]_{\Gamma'}^{\text{CDZ}}. MN\vec{N} \in [0]_{\Gamma'}^{\text{CDZ}} \\ &\Rightarrow \forall \Gamma' \not\supseteq \Gamma, \forall N \in [1]_{\Gamma'}^{\text{CDZ}}. MN \in \llbracket 0 \rrbracket_{\Gamma'}^{\text{CDZ}} \\ &\Rightarrow M \in \llbracket 1 \rightarrow 0 \rrbracket_{\Gamma}^{\text{CDZ}}. \end{aligned}$$

Proof of $\llbracket (1 \rightarrow 1) \cap (0 \rightarrow 0) \rrbracket_{\Gamma}^{\text{HR}} \subseteq \llbracket 1 \rrbracket_{\Gamma}^{\text{HR}}$. By Lemma 17C.12(ii) one has

$$\begin{aligned} \llbracket (1 \rightarrow 1) \cap (0 \rightarrow 0) \rrbracket_{\Gamma}^{\text{HR}} &\subseteq \llbracket (1 \rightarrow 1) \cap (0 \rightarrow 0) \rrbracket_{\Gamma}^{\text{HR}} \\ &= [1]_{\Gamma}^{\text{HR}} \\ &= \llbracket 1 \rrbracket_{\Gamma}^{\text{HR}}, \end{aligned}$$

by Definition 17C.7, $(1 \rightarrow 1) \cap (0 \rightarrow 0) = 1$ and Definition 17C.10.

Proof of $\llbracket 1 \rrbracket_{\Gamma}^{\text{HR}} \subseteq \llbracket (1 \rightarrow 1) \cap (0 \rightarrow 0) \rrbracket_{\Gamma}^{\text{HR}}$. Let $\Gamma' \not\supseteq \Gamma$.

$$\begin{aligned} M \in \llbracket 1 \rrbracket_{\Gamma}^{\text{HR}} &\Rightarrow M \in [1]_{\Gamma}^{\text{HR}} \\ (8) \quad &\Rightarrow \exists P \in \mathcal{A}_{\text{HR}}(M) \Gamma \vdash_{\cap}^{\text{HR}} P : 1, \quad \text{by Definition 17C.7.} \end{aligned}$$

$$\begin{aligned} N \in \llbracket 1 \rrbracket_{\Gamma'}^{\text{HR}} &\Rightarrow N \in [1]_{\Gamma'}^{\text{HR}} \\ (9) \quad &\Rightarrow \exists P' \in \mathcal{A}_{\text{HR}}(N) \Gamma' \vdash_{\cap}^{\text{HR}} P' : 1, \quad \text{by Definition 17C.7.} \end{aligned}$$

Let $\hat{P} \equiv \Phi PP'$ if P is a lambda abstraction and $\hat{P} \equiv PP'$ otherwise.

$$\begin{aligned} (8) \text{ and } (9) &\Rightarrow \Gamma' \vdash^{\text{HR}} \hat{P} : 1, \quad \text{by (Ax-}\Phi\text{-HR), } (\leq_{\mathcal{T}}), (\rightarrow\text{E}), \\ &\Rightarrow MN \in [1]_{\Gamma'}^{\text{HR}}, \quad \text{since } \hat{P} \in \mathcal{A}_{\text{HR}}(MN), \\ &\Rightarrow MN \in \llbracket 1 \rrbracket_{\Gamma'}^{\text{HR}} \\ &\Rightarrow M \in \llbracket 1 \rightarrow 1 \rrbracket_{\Gamma}^{\text{HR}}. \end{aligned}$$

$$\begin{aligned} N \in \llbracket 0 \rrbracket_{\Gamma'}^{\text{HR}} &\Rightarrow N \in [0]_{\Gamma'}^{\text{HR}} \\ (10) \quad &\Rightarrow \exists P' \in \mathcal{A}_{\text{HR}}(N) \Gamma' \vdash_{\cap}^{\text{HR}} P' : 0, \quad \text{by Definition 17C.7.} \end{aligned}$$

$$\begin{aligned} \vec{N} \in \llbracket 1 \rrbracket_{\Gamma'}^{\text{HR}} &\Rightarrow \vec{N} \in [1]_{\Gamma'}^{\text{HR}} \\ (11) \quad &\Rightarrow \exists \vec{P} \in \mathcal{A}_{\text{HR}}(\vec{N}) \Gamma' \vdash_{\cap}^{\text{HR}} \vec{P} : \vec{1}, \quad \text{by Definition 17C.7.} \end{aligned}$$

Let $\hat{P} \equiv \Phi PP' \vec{P}$ if P is a lambda-abstraction and $\hat{P} \equiv PP' \vec{P}$ otherwise.

$$\begin{aligned}
(8), (10) \text{ and } (11) &\Rightarrow \Gamma' \vdash^{\text{HR}} \hat{P} : 0, \quad \text{by (Ax-}\Phi\text{-HR), } (\leq_{\mathcal{T}}), (\rightarrow\text{E}) \\
&\Rightarrow MN\vec{N} \in [0]_{\Gamma'}^{\text{HR}} \quad \text{since } \hat{P} \in \mathcal{A}_{\text{HR}}(MN\vec{N}) \\
&\Rightarrow MN \in [0]_{\Gamma'}^{\text{HR}} \\
&\Rightarrow M \in [0 \rightarrow 0]_{\Gamma}^{\text{HR}}.
\end{aligned}$$

Proof of $[0 \rightarrow 0]_{\Gamma}^{\text{Park}} \subseteq [0]_{\Gamma}^{\text{Park}}$. Let $M \in [0 \rightarrow 0]_{\Gamma}^{\text{Park}}$ and $\Gamma' = \Gamma, z : 0$, where $z \notin \text{FV}(M)$.

$$\begin{aligned}
z \in [0]_{\{z:0\}}^{\text{Park}} &\Rightarrow z \in [0]_{\{z:0\}}^{\text{Park}} \\
&\Rightarrow Mz \in [0]_{\Gamma'}^{\text{Park}} \\
&\Rightarrow Mz \in [0]_{\Gamma'}^{\text{Park}} \\
&\Rightarrow M \in [0]_{\Gamma'}^{\text{Park}}, \quad \text{by Lemma 17C.9 and } (0 \rightarrow 0) \leq_{\text{Park}} 0, \\
&\Rightarrow M \in [0]_{\Gamma}^{\text{Park}}.
\end{aligned}$$

Proof of $[0]_{\Gamma}^{\text{Park}} \subseteq [0 \rightarrow 0]_{\Gamma}^{\text{Park}}$. Let $\Gamma' \sqsupseteq \Gamma$. Then we have

$$\begin{aligned}
(12) \quad M \in [0]_{\Gamma}^{\text{Park}} &\Rightarrow M \in [0]_{\Gamma}^{\text{Park}} \\
&\Rightarrow \exists P \in \mathcal{A}_{\text{Park}}(M) \Gamma \vdash_{\cap}^{\text{Park}} P : 0, \quad \text{by Definition 17C.7.}
\end{aligned}$$

$$\begin{aligned}
(13) \quad N \in [0]_{\Gamma'}^{\text{Park}} &\Rightarrow N \in [0]_{\Gamma'}^{\text{Park}} \\
&\Rightarrow \exists P' \in \mathcal{A}_{\text{Park}}(N) \Gamma' \vdash_{\cap}^{\text{Park}} P' : 0, \quad \text{by Definition 17C.7.}
\end{aligned}$$

Let $\hat{P} \equiv \Phi PP'$ if P is a lambda-abstraction and $\hat{P} \equiv PP'$ otherwise.

$$\begin{aligned}
(12) \text{ and } (13) &\Rightarrow \Gamma' \vdash_{\cap}^{\text{Park}} \hat{P} : 0, \quad \text{by (Ax-}\Phi\text{), } (\leq_{\text{Park}}), (\rightarrow\text{E}) \\
&\Rightarrow MN \in [0]_{\Gamma'}^{\text{Park}}, \quad \text{since } \hat{P} \in \mathcal{A}_{\text{Park}}(MN) \\
&\Rightarrow MN \in [0]_{\Gamma'}^{\text{Park}} \\
&\Rightarrow M \in [0 \rightarrow 0]_{\Gamma}^{\text{Park}}. \blacksquare
\end{aligned}$$

17C.14. DEFINITION (Semantic Satisfiability). Let ρ be a mapping from term variables to terms and write $\llbracket M \rrbracket_{\rho} \triangleq M[\vec{x} := \rho(\vec{x})]$, where $\vec{x} = \text{FV}(M)$. Define

- (i) $\mathcal{T}, \rho, \Gamma \models M : A \iff \llbracket M \rrbracket_{\rho} \in [A]_{\Gamma}^{\mathcal{T}}$.
- (ii) $\mathcal{T}, \rho, \Gamma' \models \Gamma \iff \mathcal{T}, \rho, \Gamma' \models x : B$, for all $(x:B) \in \Gamma$;
- (iii) $\Gamma \models_{\cap}^{\mathcal{T}} M : A \iff \mathcal{T}, \rho, \Gamma' \models \Gamma \Rightarrow \mathcal{T}, \rho, \Gamma' \models M : A$, 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.

17C.15. LEMMA. Let $\mathcal{T} \in \{\text{Scott, Park, CDZ, HR, DHM, BCD, AO}\}$. Then

$$\Gamma \vdash_{\cap}^{\mathcal{T}} M : A \Rightarrow \Gamma \models^{\mathcal{T}} M : A.$$

PROOF. The proof is by induction on the derivation of $\Gamma \vdash_{\cap}^{\mathcal{T}} M : A$.

Cases (Ax), (Ax-U). Immediate.

Cases (\rightarrow E), (\cap I). By induction.

Case (\leq). By Lemma 17C.13(ii).

Case (\rightarrow I). Suppose $M \equiv \lambda y.R$, $A \equiv B \rightarrow C$ and $\Gamma, y : B \vdash_{\cap}^{\mathcal{T}} R : C$.

Subcase $\mathcal{T} \neq \text{AO}$ or $C \neq_{\text{AO}} \text{U}$. Suppose $\mathcal{T}, \rho, \Gamma' \models \Gamma$ in order to show $\llbracket \lambda y.R \rrbracket_{\rho} \in \llbracket B \rightarrow C \rrbracket_{\Gamma'}^{\mathcal{T}}$. Let $\Gamma'' \not\models \Gamma'$ and $T \in \llbracket B \rrbracket_{\Gamma''}^{\mathcal{T}}$. Then by the induction hypothesis $\llbracket R \rrbracket_{\rho[y:=T]} \in \llbracket C \rrbracket_{\Gamma''}^{\mathcal{T}}$. We may assume $y \notin \rho(x)$ for all $x \in \text{dom}(\Gamma)$. Then one has $\llbracket \lambda y.R \rrbracket_{\rho} T \rightarrow_{\beta} \llbracket R \rrbracket_{\rho[y:=T]}$ and hence $\llbracket \lambda y.R \rrbracket_{\rho} T \in \llbracket C \rrbracket_{\Gamma''}^{\mathcal{T}}$, by Proposition 17C.11. Therefore $\llbracket \lambda y.R \rrbracket_{\rho} \in \llbracket B \rightarrow C \rrbracket_{\Gamma'}^{\mathcal{T}}$.

Subcase $\mathcal{T} = \text{AO}$ and $C =_{\text{AO}} \text{U}$. The result follows easily from $\lambda x. \perp \in \mathcal{A}_{\text{AO}}(\llbracket \lambda y.R \rrbracket_{\rho})$ for all ρ , and we can derive $\vdash_{\cap}^{\text{AO}} \lambda x. \perp : B \rightarrow C$, using (Ax-U), (\rightarrow I) and (\leq_{AO}). Therefore $\llbracket M \rrbracket_{\rho} \in \llbracket A \rrbracket_{\Gamma}^{\mathcal{T}}$, implying $\llbracket M \rrbracket_{\rho} \in \llbracket A \rrbracket_{\Gamma}^{\mathcal{T}}$ by Definition 17C.10. ■

Now we can prove the converse of Proposition 17C.8.

17C.16. PROPOSITION. *Let $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{BCD}, \text{AO}\}$. Then*

$$\Gamma \vdash_{\cap}^{\mathcal{T}} M : A \Rightarrow M \in \llbracket A \rrbracket_{\Gamma}^{\mathcal{T}}.$$

PROOF. Let $\rho_0(x) = x$. By Lemma 17C.12(i) $\mathcal{T}, \rho_0, \Gamma \models \Gamma$. Then $\Gamma \vdash_{\cap}^{\mathcal{T}} M : A$ implies $M = \llbracket M \rrbracket_{\rho_0} \in \llbracket A \rrbracket_{\Gamma}^{\mathcal{T}}$ by Lemma 17C.15. So we conclude $M \in \llbracket A \rrbracket_{\Gamma}^{\mathcal{T}}$ by Lemma 17C.12(ii). ■

17C.17. THEOREM (Approximation Theorem). *Let $\mathcal{T} \in \{\text{Scott}, \text{Park}, \text{CDZ}, \text{HR}, \text{DHM}, \text{BCD}, \text{AO}\}$. Then*

$$\Gamma \vdash_{\cap}^{\mathcal{T}} M : A \Leftrightarrow \exists P \in \mathcal{A}_{\mathcal{T}}(M). \Gamma \vdash_{\cap}^{\mathcal{T}} P : A.$$

PROOF. By Propositions 17C.8 and 17C.16. ■

17C.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*

$$\llbracket M \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}} = \{A \mid \Gamma \vdash_{\cap}^{\mathcal{T}} P : A \text{ for some } P \in \mathcal{A}_{\mathcal{T}}(M) \text{ and some } \Gamma \models \rho\}.$$

PROOF. By Theorem 16B.7 and the Approximation Theorem. ■

Another way of writing this is

$$\begin{aligned} \llbracket M \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}} &= \bigcup_{P \in \mathcal{A}_{\mathcal{T}}(M)} \llbracket P \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}} \\ &= \bigcup_{P \in \mathcal{A}_{\mathcal{T}}(M)} \{A \mid \Gamma \vdash_{\cap}^{\mathcal{T}} P : A \text{ for some } \Gamma \models \rho\}. \end{aligned}$$

This gives the motivation for the name ‘Approximation Theorem’. Theorem 17C.17 was first proved for $\mathcal{T} = \text{BCD}$ in [Barendregt, Coppo, and Dezani-Ciancaglini \[1983\]](#), for $\mathcal{T} = \text{Scott}$ in [Ronchi Della Rocca \[1988\]](#), for $\mathcal{T} = \text{CDZ}$ in [Coppo, Dezani-Ciancaglini, and Zacchi \[1987\]](#), 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 \[1992\]](#).

17D. Applications of the approximation theorem

As discussed in Section 16B type theories give rise in a natural way to *filter λ-models*. Properties of $\mathcal{F}^{\mathcal{T}}$ with $\mathcal{T} \in \{\text{Scott, CDZ, AO, BCD, Park, HR}\}$ can be easily derived using Theorem 17C.17. For instance, one can check the following.

- The models $\mathcal{F}^{\text{Scott}}$, \mathcal{F}^{CDZ} , \mathcal{F}^{DHM} , \mathcal{F}^{BCD} are sensible.
- The top element in \mathcal{F}^{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}^{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, Coppo, and Dezani-Ciancaglini \[1983\]](#), [Coppo, Dezani-Ciancaglini, and Zacchi \[1987\]](#), [Ronchi Della Rocca \[1988\]](#), [Abramsky and Ong \[1993\]](#), [Honsell and Ronchi Della Rocca \[1992\]](#).

17D.1. THEOREM. *The models $\mathcal{F}^{\mathcal{T}}$ with $\mathcal{T} \in \{\text{Scott, CDZ, DHM, BCD}\}$ are sensible, i.e. for all unsolvable terms M, N one has*

$$\llbracket M \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}} = \llbracket N \rrbracket_{\rho}^{\mathcal{F}^{\mathcal{T}}}.$$

PROOF. It follows immediately from Corollary 17C.18 of the Approximation Theorem, the fact that \perp is the only approximant of an unsolvable term for the mapping \square_{\perp} , and Proposition 17C.5(i). ■

Let us recall that according to Definition 16D.1(i) an untyped lambda term M is of order ∞ if

$$\forall n \exists N_n. M \rightarrow_{\beta} \lambda x_1 \cdots \lambda x_n. N_n.$$

17D.2. 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) $\llbracket M \rrbracket_{\rho}^{\mathcal{F}^{\text{AO}}} = \top = \top^{\text{AO}} \in \mathcal{F}^{\text{AO}}$ for all valuations ρ .

PROOF. Write \vdash, \leq for $\vdash_{\cap}^{\text{AO}}, \leq_{\text{AO}}$, respectively. ((i) \Leftrightarrow (ii)). It is easy to check by structural induction on types (see Exercise 13E.4) that

$$\forall A \in \mathbb{T}^{\text{AO}} \exists n \in \mathbb{N}. (\mathbf{U}^n \rightarrow \mathbf{U}) \leq A.$$

So by the Approximation Theorem it suffices to show that, if $P \in \Lambda \perp$ is an approximate normal form, then we have

$$\vdash P : (\mathbf{U}^n \rightarrow \mathbf{U}) \Leftrightarrow P \equiv \lambda x_1 \cdots \lambda x_n. P' \text{ for some } P'.$$

(\Leftarrow) By axiom $(\mathbf{U}_{\text{top}})$ and rule $(\rightarrow I)$. (\Rightarrow) Assume towards a contradiction that $P \equiv \lambda x_1 \cdots \lambda x_m. P'$ for $m < n$ and P' is of the form \perp or $x \vec{P}$. Then by Theorem 14A.9(iii)

$$\vdash P : (\mathbf{U}^n \rightarrow \mathbf{U}) \Rightarrow \{x_1 : \mathbf{U}, \dots, x_m : \mathbf{U}\} \vdash P' : (\mathbf{U}^{n-m} \rightarrow \mathbf{U}).$$

But this latter judgment can neither be derived if $P' \equiv \perp$, by Proposition 17C.5(i) and Lemma 13A.25, nor if $P' \equiv x \vec{P}$, by Theorem 14A.1(i) and (ii) and Lemma 13A.25.

((ii) \Leftrightarrow (iii)). Suppose $\vdash M : A$ for all A . Then by Theorem 16B.7

$$\llbracket M \rrbracket_{\rho}^{\mathcal{F}^{\text{AO}}} = \{A \mid \Gamma \vdash M : A \text{ for some } \Gamma \models \rho\} = \mathbb{T}^{\text{AO}},$$

for all ρ . This is the top element in \mathcal{F}^{AO} . Conversely, if $\llbracket M \rrbracket_{\rho}^{\mathcal{F}^{\text{AO}}} = \top = \mathbb{T}^{\text{AO}}$, for all ρ , then take $\rho_0(x) = \uparrow \mathbf{U}$. Then $\Gamma_{\rho_0} = \{x:\mathbf{U} \mid x \in \text{Var}\}$. Hence

$$\begin{aligned} \mathbb{T}^{\text{AO}} &= \llbracket M \rrbracket_{\rho_0}^{\mathcal{F}^{\text{AO}}} \\ &= \{A \mid \Gamma \vdash M : A \text{ for some } \Gamma \models \rho_0\}, \quad \text{by Theorem 16B.7,} \\ &= \{A \mid \vdash M : A\}, \quad \text{by Exercise 13E.5.} \end{aligned}$$

Therefore $\vdash M : A$ for all $A \in \mathbb{T}^{\text{AO}}$. ■

We denote by $\Lambda_{\downarrow \Lambda^0}$ the set of terms which reduce to a closed term.

17D.3. THEOREM. A term $M \in \Lambda_{\downarrow \Lambda^0}$ iff $\vdash_{\cap}^{\text{Park}} M : 0$.

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:

$$\{x:0 \mid x \in \mathcal{V}\} \vdash_{\cap}^{\text{Park}} P : 0 \text{ iff } \text{FV}(P) \subseteq \mathcal{V}.$$

(\Leftarrow) By an easy induction on P , using that $0 = 0 \rightarrow 0$.

(\Rightarrow) By induction on P . Lemma 13A.24 shows $\vec{B} \rightarrow 0 \neq \mathbf{U}$.

Case $P \equiv \lambda y.Q$. By Theorem 14A.9(iii) and the induction hypothesis for Q .

Case $P \equiv y\vec{P}$. By Theorem 14A.9(ii) we have $\Gamma \vdash_{\cap}^{\text{Park}} y : \vec{B} \rightarrow 0$ and $\Gamma \vdash_{\cap}^{\text{Park}} \vec{P} : \vec{B}$. Hence by Theorem 14A.9(i) one has $y \in \mathcal{V}$ and $0 \leq \vec{B} \rightarrow 0$. By β -soundness and $0 = 0 \rightarrow 0$ we get $B_i \leq 0$. Thus $\Gamma \vdash_{\cap}^{\text{Park}} P_i : 0$ and hence $\text{FV}(P_i) \subseteq \mathcal{V}$, 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 λl -terms.

17D.4. LEMMA. Let $m > 0$. Then

$$1 \leq_{\text{HR}} A_1 \rightarrow \cdots A_m \rightarrow 0 \Rightarrow [A_i = 0, \text{ for some } i].$$

PROOF. By induction on m .

Case $m = 1$. Then $1 = (1 \rightarrow 1) \cap (0 \rightarrow 0) \leq A_1 \rightarrow 0$. 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. ■

17D.5. NOTATION. Write $\Gamma_1^P \triangleq \{x:1 \mid x \in \text{FV}(P)\}$.

17D.6. LEMMA. There is no anf P such that $\Gamma_1^P \vdash_{\cap}^{\text{HR}} P : 0$.

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 $\Gamma_1^P \vdash P : 0$ implies $\Gamma_1^P, z:1 \vdash P' : 0$ and the induction hypothesis applies.

Case $P \equiv P_0 P_1 \dots P_m$, where P_0 is either a variable z or Φ . Then

$$(14) \quad \Gamma_1^P \vdash P_0 : A_1 \rightarrow \cdots A_m \rightarrow 0$$

and $\Gamma_1^{P_i} \vdash P_i : A_i$ for some A_i , by the Inversion Lemma 14A.9(ii) and (*strengthening*). Then $1 \leq A_1 \rightarrow \dots \rightarrow A_m \rightarrow 0$, by (14). By Lemma 17D.4, there exists an i such that $A_i = 0$. Then $\Gamma \vdash P_i : 0$. By the induction hypothesis, this is impossible. ■

17D.7. LEMMA. *Let $P \in \Lambda\Phi$ be an anf. If $\Gamma_1^P \vdash^{\text{HR}} P : 1$, then P is a λl -term.*

PROOF. By induction on P .

Case $P \equiv \lambda z.P'$. By Inversion Lemma 14A.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 λl -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 17D.6.

Case $P \equiv P_0 P_1 \dots P_n$, where $P_0 = x$ or $P_0 = \Phi$. By the Inversion Lemma 14A.9 (ii), $\Gamma_1^P \vdash P_0 : A_1 \rightarrow \dots \rightarrow A_n \rightarrow 1$ and $\Gamma_1^P \vdash P_i : A_i$ for all $i > 0$. By Inversion Lemma 14A.9(i) for $P_0 = x$ or Proposition 17C.5(iii) for $P_0 = \Phi$, we get $1 \leq A_1 \rightarrow \dots \rightarrow A_n \rightarrow 1$. Since $0 \leq 1$ and $0 = 1 \rightarrow 0$, we get that $1 \rightarrow \dots \rightarrow 1 \rightarrow 0 \leq A_1 \rightarrow \dots \rightarrow A_n \rightarrow 1$. 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 λl -terms. ■

17D.8. LEMMA. *Let $P \in \Lambda\Phi$ be an anf.*

- (i) *If P is a λl -term, then $\Gamma_1^P \vdash^{\text{HR}} P : 1$.*
- (ii) *If P is a λl -term and $x \in FV(P)$, then $\Gamma_1^P \uplus \{x : 0\} \vdash^{\text{HR}} P : 0$.*

PROOF. By simultaneous induction on P .

- (i) Case $P \equiv y$. Trivial.

Case $P \equiv \lambda z.P'$. If P is a λl -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 $1 = (1 \rightarrow 1) \cap (0 \rightarrow 0)$, we get $\Gamma_1^P \vdash P : 1$ by rules ($\rightarrow\text{I}$), ($\cap\text{I}$), (\leq)).

Case $P = P'P''$. Then $\Gamma_1^{P'} \vdash P' : 1 \leq 1 \rightarrow 1$ and $\Gamma_1^{P''} \vdash P'' : 1$, by the induction hypothesis (i). Hence $\Gamma_1^P \vdash P : 1$ by rules (*weakening*), (\leq) and ($\rightarrow\text{E}$).

- (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 ($\rightarrow\text{I}$), (\leq) with $1 \rightarrow 0 = 0$ 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

$$\Gamma_1^{P'} \uplus \{x : 0\} \vdash P' : 0 = 1 \rightarrow 0.$$

By the induction hypothesis (i) $\Gamma_1^{P''} \vdash P'' : 1$. Hence by (*weakening*) and ($\rightarrow\text{E}$) 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 $\Gamma_1^{P'} \vdash P' : 1 \leq 0 \rightarrow 0$. Hence by (*weakening*) and ($\rightarrow\text{E}$) we conclude $\Gamma_1^P \uplus \{x : 0\} \vdash P : 0$. ■

Now we can characterize the set $\Lambda_{\downarrow\Lambda^l}$ of terms which reduce to λl -terms in a type theoretic way.

17D.9. THEOREM. Let M be a lambda term. Then

$$M \in \Lambda_{\downarrow \Lambda^1} \Leftrightarrow \Gamma_1^M \vdash_{\cap}^{\text{HR}} M : 1.$$

PROOF. By Theorem 17C.17 the types of M are all and only the types of its approximants. By Lemmas 17D.7 and 17D.8(i) an anf is a λI -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 $\square_{\Phi}(M)$ is a λI -term, then M is a λI -term as well. ■

Theorem 17D.9 was first proved in [Honsell and Ronchi Della Rocca \[1992\]](#) by purely semantic means.

The following results are translations of the results in Theorems 17D.3, 17D.9 and 17B.15 to filter models.

17D.10. PROPOSITION. Let $M \in \Lambda^\varnothing$ be a closed lambda term. Then

- (i) M has a normal form $\Leftrightarrow \llbracket M \rrbracket^{\mathcal{D}_\infty^{\text{CDZ}}} \sqsupseteq 1$.
- (ii) M is solvable $\Leftrightarrow \llbracket M \rrbracket^{\mathcal{D}_\infty^{\text{DHM}}} \sqsupseteq 1$.
- (iii) M reduces to a λI -term $\Leftrightarrow \llbracket M \rrbracket^{\mathcal{D}_\infty^{\text{HR}}} \sqsupseteq 1$.
- (iv) M is strongly normalizing $\Leftrightarrow \llbracket M \rrbracket^{\mathcal{F}^{\text{CDV}}} \neq \emptyset$.

Let $M \in \Lambda$ be an (open) lambda term. Then

- (v) M reduces to a closed term $\Leftrightarrow \llbracket M \rrbracket_\rho^{\mathcal{D}_\infty^{\text{Park}}} \sqsupseteq 0$, for all ρ .

PROOF. (i)-(ii) We explain the situation for (i), the other case being similar.

$$\begin{aligned} M \text{ has a nf} &\Leftrightarrow \vdash_{\cap}^{\text{CDZ}} M : 1, && \text{by Theorem 17B.15(i),} \\ &\Leftrightarrow \llbracket M \rrbracket^{\mathcal{F}^{\text{CDZ}}} \ni 1, && \text{by 16B.7,} \\ &\Leftrightarrow \llbracket M \rrbracket^{\mathcal{F}^{\text{CDZ}}} \sqsupseteq \uparrow 1, && \text{by the definition of filters,} \\ &\Leftrightarrow \llbracket M \rrbracket^{\mathcal{D}_\infty^{\text{CDZ}}} \sqsupseteq \hat{f}(\uparrow 1) = f(1), && \text{by Theorem 16C.22,} \\ &\Leftrightarrow \llbracket M \rrbracket^{\mathcal{D}_\infty^{\text{CDZ}}} \sqsupseteq \Phi_{0,\infty}(1), && \text{since } 1 \in \mathcal{D}_0, \\ &\Leftrightarrow \llbracket M \rrbracket^{\mathcal{D}_\infty^{\text{CDZ}}} \sqsupseteq 1, \end{aligned}$$

by the identification of \mathcal{D}_0 as a subset of \mathcal{D}_∞ .

(iii) Similarly, using Theorem 17D.9.

(iv) As (i), but simpler as the step towards \mathcal{D}_∞ is not made. Notice that \mathcal{F}^{CDV} gives rise to a λI -model, not to a λ -model.

(v) Similarly, using Theorem 17D.3. ■

We do not have a characterization like (i) in the Proposition for $\mathcal{D}_\infty^{\text{Scott}}$, as it was shown in [Wadsworth \[1976\]](#) that there is a closed term J without a normal form such that $\mathcal{D}_\infty^{\text{Scott}} \models I = J$.

17E. Undecidability of inhabitation

In this section we consider type theories with infinitely many type atoms, as described in Section 13A. 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_{CDV} , respectively. Moreover $\text{TI} = \text{TI}^{\text{CDV}}$.

We investigate the *inhabitation problem* for this type theory, which is to determine, for a given type A , if there exists a closed term of type A (an 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.

17E.1. LEMMA. *Let $\Gamma = \{x_1:A_1, \dots, x_n:A_n\}$. 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 $\vdash N : A_1 \rightarrow \dots \rightarrow A_n \rightarrow A$.*

PROOF. (1) \Rightarrow (2) Define $N \equiv \lambda x_1 \dots x_n. M$. Apply n times rule $(\rightarrow I)$.

(2) \Rightarrow (1) Take $M \equiv Nx_1 \dots x_n$ and apply n times $(\rightarrow E)$ and *(weakening)*. ■

The main result of the present section (Theorem 17E.31) is that type inhabitation is undecidable for $\mathcal{T} = \text{CDV}$. Compare this to Proposition 2D.15 and Statman [1979a], stating that for simple types the problem is decidable in polynomial space.

By Theorem 17B.15 and Corollary 14B.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 (“typewriter automaton”). Thus, the global strategy of the proof is as follows. We make the following abbreviations.

17E.2. DEFINITION. We introduce the following decision problems.

- 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 $\lambda_{\cap}^{\text{CDV}}$.

These are problems in the following sense. In each case there is a set involved: the set of natural number codes for the description of the automata or types involved; the problem consists in determining whether a candidate element of the set actually belongs to it. It is well known that EQA is undecidable, see e.g. Kozen [1997]. The following inequalities, show that IHP is undecidable.

- $\text{EQA} \leq_m \text{ETW}$ (Lemma 17E.26);
- $\text{ETW} \leq_m \text{WTG}$ (Proposition 17E.30);
- $\text{WTG} \leq_m \text{IHP}$ (Corollary 17E.24).

Basic properties

We begin with some basic observations concerning the relation \leq .

17E.3. LEMMA. Let $n > 0$.

- (i) Let $\alpha \in \mathbb{A}_\infty$, and none of the $A_1, \dots, A_n \in \mathbb{T}$ be an intersection. Then

$$A_1 \cap \dots \cap A_n \leq \alpha \Rightarrow \exists i. \alpha \equiv A_i.$$

- (ii) Let $\alpha_1, \dots, \alpha_n \in \mathbb{A}_\infty$ and $A \in \mathbb{T}$ be not an intersection. Then

$$\alpha_1 \cap \dots \cap \alpha_n \leq A \Rightarrow \exists i. A \equiv \alpha_i.$$

- (iii) Let $\alpha_1, \dots, \alpha_n \in \mathbb{A}_\infty$ and $A \in \mathbb{T}$. Then

$$\alpha_1 \cap \dots \cap \alpha_n \leq A \Rightarrow A \equiv \alpha_{i_1} \cap \dots \cap \alpha_{i_k},$$

for some $k \geq 0$ and $1 \leq i_1 < \dots < i_k \leq n$.

PROOF. (i), (ii) Exercise 17F.18.

(iii) Let $A \equiv B_1 \cap \dots \cap B_k$ with $k > 0$ and the B_j not intersections. Then for each j one has $\alpha_1 \cap \dots \cap \alpha_n \leq B_j$ and can apply (ii) to show that $B_j \equiv \alpha_{i_j}$. ■

17E.4. LEMMA. Let $A_1, \dots, A_n, B \in \mathbb{T}$ and $\alpha_1, \dots, \alpha_n, \beta \in \mathbb{A}_\infty$, with $n > 0$. Then

$$(A_1 \rightarrow \alpha_1) \cap \dots \cap (A_n \rightarrow \alpha_n) \leq (B \rightarrow \beta) \Rightarrow \exists i [\beta \equiv \alpha_i \& B \leq A_i].$$

PROOF. By Theorem 14A.7 the type theory CDV is β -sound. Hence by the assumption it follows that $B \leq (A_{i_1} \cap \dots \cap A_{i_k})$ and $(\alpha_{i_1} \cap \dots \cap \alpha_{i_k}) \leq \beta$. By Lemma 17E.3(ii) one has $\beta \equiv \alpha_{i_p}$, for some $1 \leq p \leq k$, and the conclusion follows. ■

17E.5. LEMMA. If $\Gamma \vdash \lambda x.M : A$ then $A \notin \mathbb{A}_\infty$.

PROOF. Suppose $\Gamma \vdash \lambda x.M : \alpha$. By Lemma 14A.1(iii) it follows that there are $n > 0$ and $B_1, \dots, B_n, C_1, \dots, C_n$ such that $\Gamma, x : B_i \vdash M : C_i$, for $1 \leq i \leq n$, and $(B_1 \rightarrow C_1) \cap \dots \cap (B_n \rightarrow C_n) \leq \alpha$. This is impossible by Lemma 17E.3(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.

17E.6. DEFINITION. (i) Let $\mathcal{X}, \mathcal{Y} \subseteq \mathbb{T}$ be sets of types. Write

$$\mathcal{X} \rightarrow \mathcal{Y} \triangleq \{X \rightarrow Y \mid X \in \mathcal{X}, Y \in \mathcal{Y}\}.$$

$$\mathcal{X} \sqcap \mathcal{Y} \triangleq \{X \cap Y \mid X \in \mathcal{X}, Y \in \mathcal{Y}\}.$$

$$\mathcal{X}^\cap \triangleq \{A_1 \cap \dots \cap A_n \mid n \geq 1 \& A_1, \dots, A_n \in \mathcal{X}\}.$$

If $A \equiv A_1 \cap \dots \cap A_n$, 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.

$$(1) \quad \mathcal{A} \triangleq \mathbb{A}_\infty^\cap.$$

$$(2) \quad \mathcal{B} \triangleq (\mathbb{A}_\infty \rightarrow \mathbb{A}_\infty)^\cap.$$

$$(3) \quad \mathcal{C} \triangleq (\mathcal{D} \rightarrow \mathbb{A}_\infty)^\cap, \text{ where}$$

$$(4) \quad \mathcal{D} \triangleq (\mathcal{B} \rightarrow \mathbb{A}_\infty) \sqcap (\mathcal{B} \rightarrow \mathbb{A}_\infty).$$

(iii) Types in $\mathcal{A} \cup \mathcal{B} \cup \mathcal{C}$ are called *game types*.

(iv) A type context Γ is a *game context* if all types in Γ are game types.

We show some properties of type judgements involving game types.

17E.7. LEMMA. *For a game context Γ the following three properties hold.*

$$(i) \quad \Gamma \vdash xM : A \Rightarrow A \in \mathcal{A} \& \exists n > 0. [\Gamma(x) \equiv (E_1 \rightarrow \alpha_1) \cap \dots \cap (E_n \rightarrow \alpha_n) \&$$

$$\exists k > 0 \exists i_1, \dots, i_k. [1 \leq i_1 < \dots < i_k \leq n \&$$

$$A \equiv \alpha_{i_1} \cap \dots \cap \alpha_{i_k} \& \Gamma \vdash M : (E_{i_1} \cap \dots \cap E_{i_k})].$$

$$(ii) \quad \Gamma \vdash xM : \alpha \Rightarrow \exists n > 0. [\Gamma(x) \equiv (E_1 \rightarrow \alpha_1) \cap \dots \cap (E_n \rightarrow \alpha_n) \&$$

$$\exists i. [1 \leq i \leq n \& \alpha \equiv \alpha_i \& \Gamma \vdash M : E_i]].$$

$$(iii) \quad \Gamma \not\vdash xMN : A.$$

PROOF. (i) Suppose $\Gamma \vdash xM : A$. By Lemma 14A.9(ii) we have for some type B that $\Gamma \vdash x : (B \rightarrow A)$ and $\Gamma \vdash M : B$. Then $\Gamma(x) \leq B \rightarrow A$, by Lemma 14A.1(i). By Lemma 17E.3(ii), this cannot happen if $\Gamma(x)$ is in \mathcal{A} . Thus $\Gamma(x)$, being a game type, is of the form $(E_1 \rightarrow \alpha_1) \cap \dots \cap (E_n \rightarrow \alpha_n)$. Then,

$$(E_1 \rightarrow \alpha_1) \cap \dots \cap (E_n \rightarrow \alpha_n) \equiv \Gamma(x) \leq (B \rightarrow A),$$

Since CDV is β -sound and by Lemma 17E.3(iii), we have $B \leq E_{i_1} \cap \dots \cap E_{i_k}$ and $\alpha_{i_1} \cap \dots \cap \alpha_{i_k} \equiv A$, for some $k > 0$ and i_j such that $1 \leq i_j \leq n$.

(ii) By (i) and Lemma 17E.3(ii).

(iii) By (i), using that $B \rightarrow A \neq \alpha_{i_1} \cap \dots \cap \alpha_{i_k}$, by Lemma 17E.3(ii). ■

17E.8. LEMMA. *If A is a game type and $D \in \mathcal{D}$, then $A \not\leq D$.*

PROOF. Suppose $A \leq D \leq (B \rightarrow \alpha)$, with $B \in \mathcal{B}$. The case $A \in \mathcal{A}$ is impossible by Lemma 17E.3(ii). If $A \in \mathcal{B}$, then $(\alpha_1 \rightarrow \beta_1) \cap \dots \cap (\alpha_n \rightarrow \beta_n) \leq B \rightarrow \alpha$ and hence $B \leq \alpha_i$ for some i , by Lemma 17E.4. By Lemma 17E.3(i) this is also impossible. If $A \in \mathcal{C}$, then $(D_1 \rightarrow \beta_1) \cap \dots \cap (D_n \rightarrow \beta_n) \leq B \rightarrow \alpha$ and hence $B \leq D_i \in \mathcal{D}$ for some i , by Lemma 17E.4. We have already shown that this is impossible. ■

For game contexts the Inversion Lemma 14A.9 can be extended as follows.

17E.9. LEMMA. *Let Γ be a game context, and let M be in normal form.*

(i) *If $\Gamma \vdash M : (B_1 \rightarrow \alpha_1) \cap (B_2 \rightarrow \alpha_2) \in \mathcal{D}$, 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 $\Gamma(x) \equiv (E_1 \rightarrow \beta_1) \cap \dots \cap (E_n \rightarrow \beta_n)$, 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 17E.7 the term M cannot be an application. If it is a variable x , then $\Gamma(x) \leq (B_1 \rightarrow \alpha_1) \cap (B_2 \rightarrow \alpha_2)$, by Lemma 14A.1(i). This

contradicts Lemma 17E.8, because $\Gamma(x)$ is a game type. It follows that $M = \lambda y.N$ and $\Gamma \vdash \lambda y.N : (B_i \rightarrow \alpha_i)$. Then for $i = 1, 2$ one has $\Gamma, y:B_i \vdash N_i : \alpha_i$, by Lemma 14A.9(iii).

(ii) M is not an abstraction by Lemma 17E.5. 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 17E.3(i) one has $\Gamma(z) \in \mathcal{A}$, with α as one of the components. If M is an application, $M \equiv zM_1 \cdots M_m$, $m > 0$, write

$$\Gamma(z) = (E_1 \rightarrow \beta_1) \cap \cdots \cap (E_n \rightarrow \beta_n).$$

As it is a game type $\Gamma(z) \notin \mathcal{A}$, by Theorem 14A.9(ii). Then $M \equiv zN$, by Lemma 17E.7(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. Within a game one can have a particular ‘*play*’. (Chess is a game, but the match between Kasparov and Karpov consisted of several plays). These form an intermediate step in our construction. The idea of a tree game is to represent, in an abstract way, the crucial combinatorial behavior of proof search in CDV. We now focus on establishing $\text{WTG} \leq_m \text{IHP}$.

17E.10. 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

$$G = \langle a, A, \{C_1, \dots, C_n\} \rangle,$$

where $a \in \Sigma$, $A \subseteq \Sigma$ and C_1, \dots, 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.

17E.11. DEFINITION. Let Σ be a finite subset of \mathbb{A}_∞ , 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 $A = \{a_1, \dots, a_n\}$, then $\tilde{A} \triangleq a_1 \cap \cdots \cap a_n$.
2. If $B = \{\langle a_1, b_1 \rangle, \dots, \langle a_n, b_n \rangle\}$, then $\tilde{B} \triangleq (a_1 \rightarrow b_1) \cap \cdots \cap (a_n \rightarrow b_n)$.
3. If $C = \{\langle \langle B_1, b_1 \rangle, \langle B'_1, b'_1 \rangle, c_1 \rangle, \dots, \langle \langle B_n, b_n \rangle, \langle B'_n, b'_n \rangle, c_n \rangle\}$, then
 $\tilde{C} \triangleq (((\tilde{B}_1 \rightarrow b_1) \cap (\tilde{B}'_1 \rightarrow b'_1)) \rightarrow c_1) \cap \cdots \cap (((\tilde{B}_n \rightarrow b_n) \cap (\tilde{B}'_n \rightarrow b'_n)) \rightarrow c_n)$.

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 non-deterministically 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.

17E.12. DEFINITION. Let $G = \langle a, A, \{C_1, \dots, C_n\} \rangle$ be a tree game over Σ . A *position* T of G is a finite labelled tree, satisfying the following conditions.

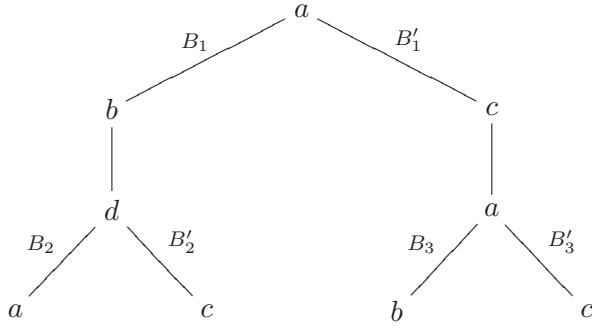


FIGURE 44. An example position

- 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.

17E.13. DEFINITION. Let $G = \langle a, A, \{C_1, \dots, C_n\} \rangle$ 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* if all labels of the leaves of T are in A .

17E.14. DEFINITION. Let T be a position in a game $G = \langle a, A, \{C_1, \dots, C_n\} \rangle$, 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 Fig. 45. 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*.

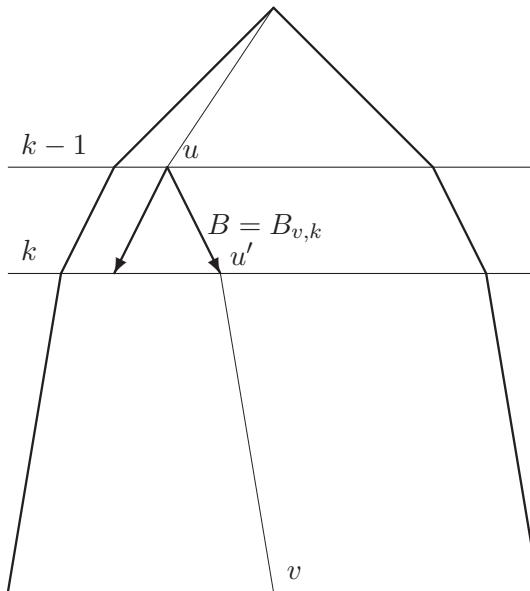
17E.15. DEFINITION. (i) Let $G = \langle a, A, \{C_1, \dots, C_n\} \rangle$ and let T be a (current) position in G . There are two possibilities to obtain a *next position*.

- (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 appropriate actions can be performed at every leaf—otherwise the resulting tree is not a position.

- (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 ;

FIGURE 45. Local move X associated to node v

(b) Create a single child of v , labelled a .

Again the step is only allowed if the appropriate actions can be performed at every leaf—otherwise the resulting tree is not a position.

(ii) If a position T' is reachable from T with help of one global step C_i , we write

$$T \xrightarrow{C_i} T'.$$

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* if 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 is *solvable* (i.e. the player can win) if the initial position is favorable.

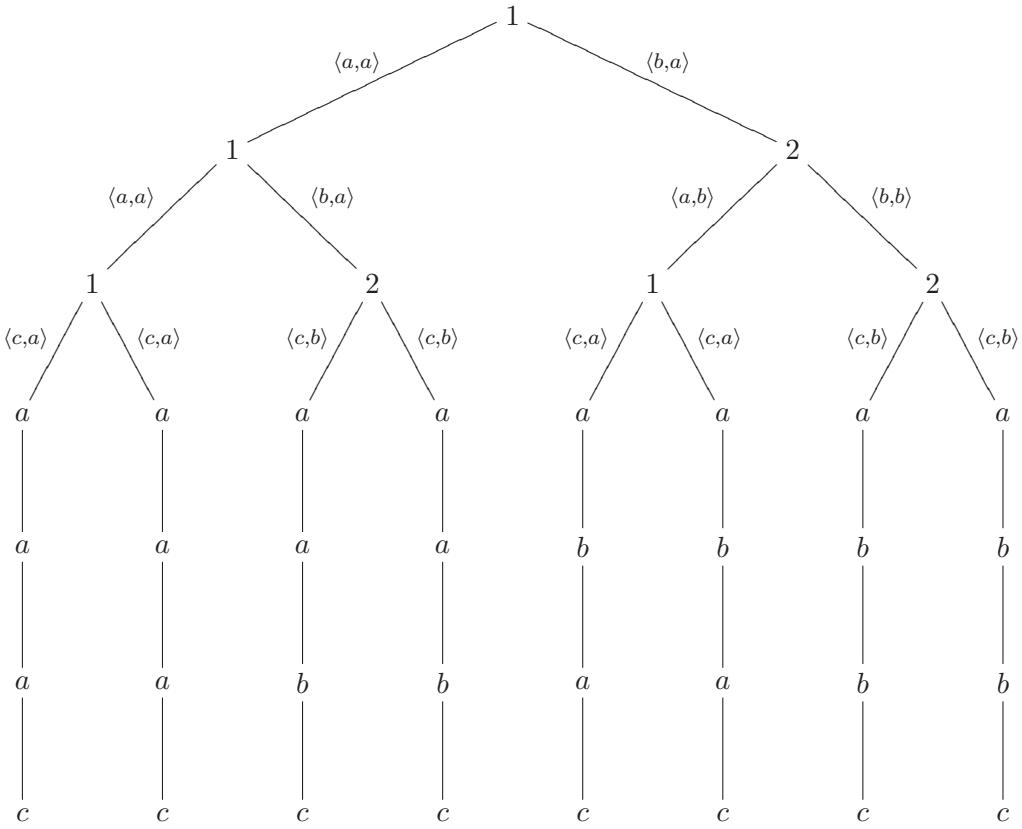
The following example gives an idea of the tree games and moreover it is important for our principal construction.

17E.16. 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

- $C_1 = \{\langle \langle \langle a, a \rangle \rangle, 1 \rangle, \langle \langle b, a \rangle \rangle, 2 \rangle, 1 \rangle, \langle \langle \langle a, b \rangle \rangle, 1 \rangle, \langle \langle b, b \rangle \rangle, 2 \rangle, 2 \rangle\};$
- $C_2 = \{\langle \langle \langle c, a \rangle \rangle, a \rangle, \langle \langle c, a \rangle \rangle, a \rangle, 1 \rangle, \langle \langle \langle c, b \rangle \rangle, a \rangle, \langle \langle c, b \rangle \rangle, a \rangle, 2 \rangle\}.$

Fig. 46 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

$$T_0 \Rightarrow^{C_1} T_1 \Rightarrow^{C_1} T_2 \Rightarrow^{C_2} T_3 \Rightarrow^1 T_4 \Rightarrow^2 T_5 \Rightarrow^3 T_6 = T,$$

FIGURE 46. A winning position in G_0 of Example 17E.16.

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:

$$T_0 \Rightarrow^{C_1} T_1 \Rightarrow^{C_1} \dots \Rightarrow^{C_1} T_{n-1} \Rightarrow^{C_2} T_n \Rightarrow^1 T_{n+1} \Rightarrow^2 \dots \Rightarrow^n T_{2n},$$

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 $n+k-1$). 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 nondeterministic 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 non-determinism 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.

17E.17. DEFINITION. Let $k, n > 0$. Let $\vec{A}_1, \dots, \vec{A}_n$ be n sequences of k types:

$$\vec{A}_i = A_{i1}, \dots, A_{ik}.$$

Let $\Gamma_i = \{x_1:A_{i1}, \dots, x_k:A_{ik}\}$ and let $\alpha_i \in \mathbb{A}_\infty$, for $1 \leq i \leq n$. An instance of a *generalized inhabitation problem* (*gip*) is a finite set of pairs $P = \{\langle \Gamma_i, \alpha_i \rangle \mid i = 1, \dots, n\}$, 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 . Then we say that M solves P . This is equivalent to requiring that

$$\vdash \lambda \vec{x}.M : (\vec{A}_1 \rightarrow \alpha_1) \cap \dots \cap (\vec{A}_n \rightarrow \alpha_n).$$

17E.18. DEFINITION. Let $G = \langle a, A, \{C_1, \dots, C_n\} \rangle$ be a tree game and let T be a position in G .

(i) Define $\Gamma_G \triangleq \{x_0:\tilde{A}, x_1:\tilde{C}_1, \dots, x_n:\tilde{C}_n\}$.

(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

$$\Gamma_v \triangleq \{y_k:\tilde{B}_{v,k} \mid k \in J\}.$$

(iii) Define the gip $P_T^G \triangleq \{\langle \Gamma_G \cup \Gamma_v, a_v \rangle \mid v \text{ is a leaf of } T \text{ with label } a_v\}$.

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 17F.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 of the nature of intersection types and the need to solve various inhabitation problems uniformly (i.e., to solve a *gip*).

17E.19. LEMMA. Let $G = \langle a, A, \{C_1, \dots, C_n\} \rangle$ 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_2 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 $P_{T_2}^G$ one has

$$\begin{aligned} \Gamma_G \cup \Gamma_v, y_k:\tilde{B} &\vdash N : a_{v'}; \\ \Gamma_G \cup \Gamma_v, y_k:\tilde{B}' &\vdash N : a_{v''}. \end{aligned}$$

So $\Gamma_G \cup \Gamma_v \vdash \lambda y_k.N : (\tilde{B} \rightarrow a_{v'}) \cap (\tilde{B}' \rightarrow a_{v''})$. 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 $y_k : \tilde{B}_{v,k} \vdash y_k : a'_v \rightarrow a_v$ and $\Gamma_G \cup \Gamma_v \vdash N : a_{v'}$, by assumption. Therefore

$$\Gamma_G \cup \Gamma_v \vdash y_k N : a_v. \blacksquare$$

17E.20. COROLLARY. Let G be a tree-game. For positions T one has

$$T \text{ is favorable } \Rightarrow P_T^G \text{ has a solution.}$$

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.

17E.21. LEMMA. Let T_1 be a position in a tree game G and let M be a solution in β -nf of $P_{T_1}^G$. 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 $P_{T_2}^G$, 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 $P_{T_2}^G$, for some T_2 with $T_1 \Rightarrow^{C_i} T_2$.

PROOF. Case $M \equiv z$. As M is a solution of $P_{T_1}^G$, one has

$$\Delta_v = \Gamma_G \cup \Gamma_v \vdash z : a_v,$$

for all leaves v of T_1 . Then by Lemma 14A.1(i) one has $\Delta_v(z) \leq a_v$. Hence by Lemma 17E.3(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 17E.9(ii) we have $M \equiv zN$ and $\Delta_v(z) \equiv (E_1 \rightarrow \beta_1) \cap \dots \cap (E_n \rightarrow \beta_n)$, 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

$$z : (\alpha_1 \rightarrow \beta_1) \cap \dots \cap (\alpha_n \rightarrow \beta_n)$$

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 $y_k : \tilde{B}_{v,k} \vdash y_k : \alpha_j \rightarrow \beta_j$. Hence $\Delta_{v'} \vdash M : a_{v'}$.

Subcase $E_j \in \mathcal{D}$. Then $\Delta_v(z) \equiv (E_1 \rightarrow \beta_1) \cap \dots \cap (E_n \rightarrow \beta_n)$. Hence $z : \Delta_v(z)$ is $x_i : C_i$ for some i . So $z \in \text{dom}(\Gamma_G)$ and therefore $\Delta_w(z) = \Delta_v(z)$ for all leaves w of T_1 . Let $C_i = \{\dots, \langle \langle B_j, \alpha_j \rangle, \langle B'_j \alpha'_j \rangle, \beta_j \rangle, \dots\}$ and $E_j = ((\tilde{B}_j \rightarrow \alpha_j) \cap (\tilde{B}'_j \rightarrow \alpha'_j))$. 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 $k = \text{depth}(T_1)$. One has $\Delta_v \vdash N : E_j$, hence by Lemma 17E.9(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 17E.5. ■

17E.22. COROLLARY. Let G be a tree game. For positions T one has

$$T \text{ is favorable} \Leftrightarrow P_T^G \text{ has a solution.}$$

PROOF. (\Rightarrow) This was Corollary 17E.20. (\Leftarrow) Let M be a solution of P_T^G . By Theorem 17B.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 . ■

17E.23. THEOREM. Let G be a tree game with initial position $\{a\}$. Then

$$G \text{ is solvable} \Leftrightarrow P_{\{a\}}^G \text{ has a solution.}$$

PROOF. Immediate from the previous Corollary. ■

17E.24. COROLLARY. $\text{WTG} \leq_m \text{IHP}$, i.e. winning a tree-game is many-one reducible 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 kind of a linear-bounded automaton, see Kozen [1997].

17E.25. DEFINITION. (i) A (deterministic) *typewriter automaton* \mathcal{A} is a tuple of the form

$$\mathcal{A} \triangleq \langle \Sigma, Q, q_0, F, \varrho \rangle,$$

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

$$\varrho : (Q - F) \times (\Sigma \cup \{\epsilon\}) \rightarrow Q \times (\Sigma \cup \{\epsilon\}),$$

which must satisfy the following condition: whenever $\varrho(q, a) = (p, b)$, then either $a, b \in \Sigma$ or $a = b = \epsilon$.

(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* $\bar{\varrho}$ is defined as follows:

- $\bar{\varrho}(\langle w, q, av \rangle) \triangleq \langle wb, p, v \rangle$, if $a \neq \epsilon$ and $\varrho(q, a) = (p, b)$;
- $\bar{\varrho}(\langle w, q, \epsilon \rangle) \triangleq \langle \epsilon, p, w \rangle$, if $\varrho(q, \epsilon) = (p, \epsilon)$.

(iv) The language $L^{\mathcal{A}}$ accepted by \mathcal{A} is the set of all $w \in \Sigma^*$, such that

$$\bar{\varrho}^k(\langle \epsilon, q_0, w \rangle) = \langle u, q, v \rangle, \text{ for some } k \text{ and } q \in F, \text{ and } uv \in \Sigma^*.$$

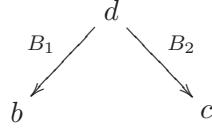
Recall that ETW is the emptiness problem for typewriter automata and EQA is the emptiness problem for queue automata. The latter is undecidable, see Kozen [1997]. We need the following.

17E.26. LEMMA. $\text{EQA} \leq_m \text{ETW}$.

PROOF. Exercise 17F.23. ■

It follows that also ETW is undecidable.

Our goal is now to represent typewriters as games, in order to establish $\text{ETW} \leq_m \text{WTG}$. We begin with a refinement of Example 17E.16. 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



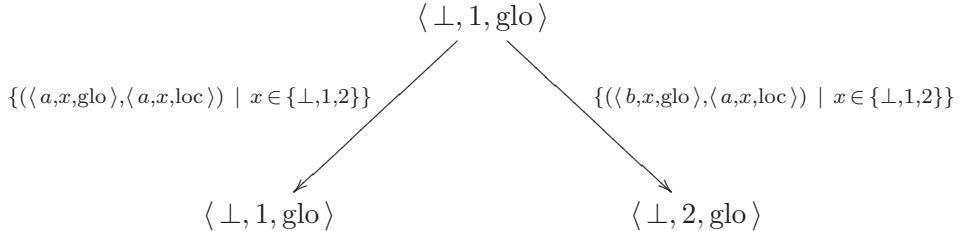
in order to enhance readability.

17E.27. DEFINITION. The alphabet Σ_1 is the following Cartesian product.

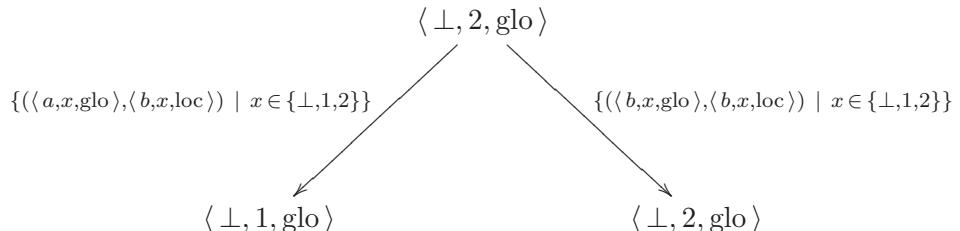
$$\Sigma_1 \triangleq \{\perp, a, b\} \times \{\perp, 1, 2\} \times \{\text{loc}, \text{glo}\}.$$

We define a *tree game* $G_1 \triangleq \langle \langle \perp, 1, \text{glo} \rangle, \Sigma_1, \{C_1, C_2, C_3\} \rangle$. The set of accepting labels is Σ_1 , because we are interested in all possible ‘plays’ (i.e. instances) of the game. The moves are defined as follows.

(i) *global move* C_1 consists of the following two triples:

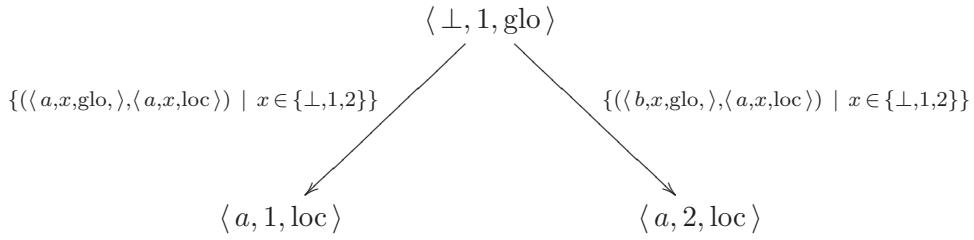


and

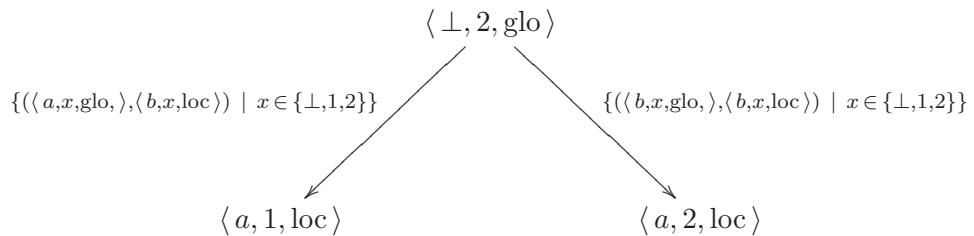


Before we define other global moves, let us point out that C_1 is very similar to rule C_1 of the game G_0 in Example 17E.16 (observe the a and b in the first component and 1 and 2 in the second one).

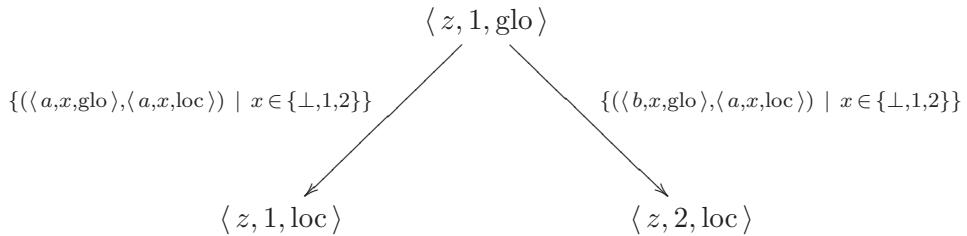
(ii) *global move* C_2 consists again of two triples:



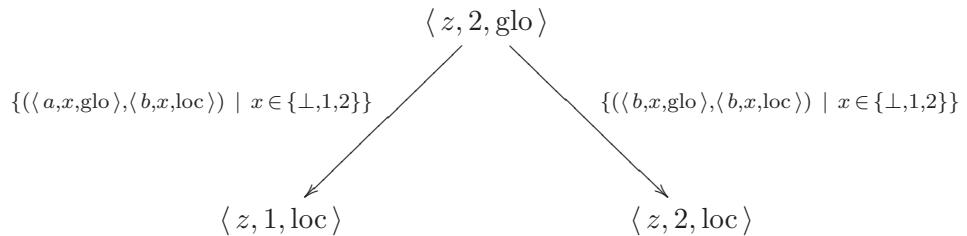
and



(iii) *global move C₃* consists of the triples (where $z \in \{a, b\}$, i.e., $z \neq \perp$)



and



17E.28. LEMMA. *Every play of G_1 must have the following sequence of moves.*

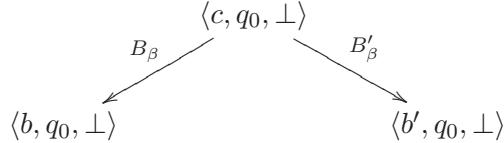
	C_1		C_1		C_1		C_1	\dots	C_1		C_2	
1	C_3	2	C_3	3	C_3	4	C_3	\dots	m	C_3	$m+1$	C_3
$m+2$	C_3	$m+3$	C_3	$m+4$	C_3	\dots	\dots	\dots	\dots	\dots	\dots	

That is, the game starts with m times a C_1 move (possibly $m = 0$ or $m = \infty$), followed by a single C_2 move. Then the local and global C_3 moves alternate. It is convenient to see a play as divided into phases. Each phase consists of $m+1$ “rounds”; in the first phase a “round” is a single global move, in the other phases it consists of two steps: one local and one global. The local move declared at any global step is executed $m+1$ phases later, after exactly $2m+1$ steps.

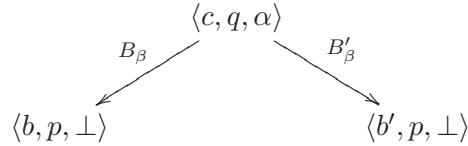
PROOF. Exercise 17F.22. ■

17E.29. 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 $\Sigma_1 \times Q \times (\Sigma^{\mathcal{A}} \cup \{\perp, \epsilon\})$.
- (ii) For each local move B of G_1 and each $\beta \in \Sigma^{\mathcal{A}} \cup \{\epsilon\}$, we define a local move $B_{\beta} = \{(\langle a, q, \beta \rangle, \langle b, q, \perp \rangle) \mid q \in Q \text{ and } \langle a, b \rangle \in B\}$.
- (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 $C_{\Delta}^{\mathcal{A}}$ consisting of all triples of the form



where $\varrho(q, \alpha) = (p, \beta)$.

- (v) Define $C_1^{\mathcal{A}}(\beta) \triangleq \{\Delta_{\beta} \mid \Delta \in C_1\}$, for $\beta \in \Sigma^{\mathcal{A}}$;
- $C_2^{\mathcal{A}} \triangleq \{\Delta_{\epsilon} \mid \Delta \in C_2\}$;
- $C_3^{\mathcal{A}} \triangleq \bigcup \{C_{\Delta}^{\mathcal{A}} \mid \Delta \in C_3\}$.
- (vi) The initial symbol of $G^{\mathcal{A}}$ is $a = \langle a_1, q_0, \perp \rangle$, where we have $a_1 = \langle \perp, 1, G \rangle$, the initial symbol of G_1 .
- (vii) The set of final symbols is $A = \Sigma_1 \times (\Sigma^{\mathcal{A}} \cup \{\perp, \epsilon\}) \times F^{\mathcal{A}}$;
- (viii) Finally, we take $G^{\mathcal{A}} \triangleq \langle a, A, \{C_1^{\mathcal{A}}(\beta) \mid \beta \in \Sigma^{\mathcal{A}}\} \cup \{C_2^{\mathcal{A}}, C_3^{\mathcal{A}}\} \rangle$.

17E.30. PROPOSITION. Let \mathcal{A} be a typewriter that accepts the language $L_{\mathcal{A}}$. Then

$$L_{\mathcal{A}} \neq \emptyset \Leftrightarrow G^{\mathcal{A}} \text{ is solvable.}$$

Hence ETW \leq_m WTG.

PROOF. Our game G^A behaves as a “Cartesian product” of G_1 and A . Informally speaking, there is no communication between the first component and the other two. In particular we have the following.

1. Lemma 17E.28 remains true with G_1 replaced by G^A and C_1, C_2, C_3 replaced respectively by $C_1^A(\beta), C_2^A, C_3^A$. That is, a legitimate play of G^A must look as follows.

	$C_1^A(\beta_1)$		$C_1^A(\beta_2)$		\cdots	$C_1^A(\beta_m)$		C_2^A
1	C_3^A	2	C_3^A		\cdots	m	C_3^A	$m+1$ C_3^A
$m+2$	C_3^A	$m+3$	C_3^A		\cdots			

2. If a position T of G^A can be reached from the initial position, then the second and third components of labels are always the same for all nodes at every fixed level of T .

Consider a play of the game G^A as in 1 above. Note that this sequence is fully determined by the choice of m and β_1, \dots, β_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 $O^A(w)$ the ‘opening’ $C_1^A(\beta_1), \dots, C_1^A(\beta_m)$ in the game G^A .

Claim. Typewriter A accepts w iff $O^A(w)$ leads to a winning position in G^A .

We shall now prove the claim. Let β_j^k denote the symbol contained in the j -th cell of the tape of 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 $\beta_{m+1}^k = \epsilon$. 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 statements hold.

- (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 .

In particular, the second and third component of these labels are solely determined by the depth of the appropriate node. That is, the computation-related information is the same in every node at any given level; the tree shape is only needed to ensure the pattern of alternating global and local moves.

To have a closer look at the simulation, assume that $\varrho(\beta, q) = (\alpha, p)$. Then a global move changes labels of the form (\cdot, β, q) (where we ignore the first coordinate) into (\cdot, \perp, p) and creates a local move containing pairs of the form $(\cdot, \alpha, q') \rightarrow (\cdot, \perp, q')$. The role of the local move is to pass the information about α (the symbol to be written in place of β) to the next phase of the game.

The next local move brings the information from the previous phase about the symbol in the next tape cell, and then we have again a global move, and so on. Consider for example an initial word $\beta_1\beta_2\beta_3$ and let the automaton take on the following sequence of

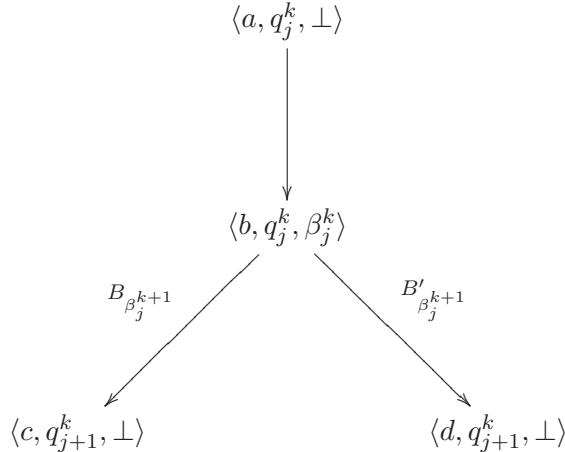


FIGURE 47. Simulation of a single machine step

configurations:

$$\begin{aligned}
 (\varepsilon, q_0, \beta_1 \beta_2 \beta_3) &\rightarrow (\gamma_1, q_1, \beta_2 \beta_3) \rightarrow (\gamma_1 \gamma_2, q_2, \beta_3) \rightarrow (\gamma_1 \gamma_2 \gamma_3, q_3, \varepsilon) \rightarrow \\
 &\rightarrow (\varepsilon, p_0, \gamma_1 \gamma_2 \gamma_3) \rightarrow (\delta_1, p_1, \gamma_2 \gamma_3) \rightarrow (\delta_1 \delta_2, p_2, \gamma_3) \rightarrow (\delta_1 \delta_2 \delta_3, p_3, \varepsilon) \rightarrow \\
 &\rightarrow (\varepsilon, r_0, \delta_1 \delta_2 \delta_3) \rightarrow \dots
 \end{aligned}$$

This corresponds to a play with these labels on the consecutive levels of the tree:

$$\begin{aligned}
 (\cdot, \perp, q_0) &\xrightarrow{\beta_1} (\cdot, \perp, q_0) \xrightarrow{\beta_2} (\cdot, \perp, q_0) \xrightarrow{\beta_3} (\cdot, \perp, q_0) \xrightarrow{\varepsilon} \\
 (\cdot, \beta_1, q_0) &\xrightarrow{\gamma_1} (\cdot, \perp, q_1) \rightarrow (\cdot, \beta_2, q_1) \xrightarrow{\gamma_2} (\cdot, \perp, q_2) \rightarrow (\cdot, \beta_3, q_2) \xrightarrow{\gamma_3} (\cdot, \perp, q_3) \rightarrow (\cdot, \varepsilon, q_3) \xrightarrow{\varepsilon} \\
 (\cdot, \perp, p_0) &\rightarrow (\cdot, \gamma_1, p_0) \xrightarrow{\delta_1} (\cdot, \perp, p_1) \rightarrow (\cdot, \gamma_2, p_1) \xrightarrow{\delta_2} (\cdot, \perp, p_2) \rightarrow (\cdot, \gamma_3, p_2) \xrightarrow{\delta_3} (\cdot, \perp, p_3) \rightarrow \\
 (\cdot, \varepsilon, p_3) &\xrightarrow{\varepsilon} (\cdot, \perp, r_0) \rightarrow (\cdot, \delta_1, r_0) \rightarrow \dots
 \end{aligned}$$

Fig. 47 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 $L_A \neq \emptyset$ iff there is a strategy to win G^A . Hence the emptiness problem for typewriter automata can be reduced to the problem of winning tree games. ■

17E.31. THEOREM. *The inhabitation problem for $\lambda_{\cap}^{\text{CDV}}$ is undecidable.*

PROOF. By Corollary 17E.24, Lemma 17E.26 and Proposition 17E.30. ■

17E.32. COROLLARY. *The inhabitation problem for $\lambda_{\cap}^{\text{BCD}}$ is undecidable.*

PROOF. Do Exercise 17F.26. ■

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 \[1983a\]](#). We denote it by *i-rank* in order to distinguish it from the rank in Definition 1A.21.

$$\begin{aligned} \text{i-rank}(A) &\triangleq 0, && \text{for simple types } A; \\ \text{i-rank}(A \cap B) &\triangleq \max(1, \text{i-rank}(A), \text{i-rank}(B)); \\ \text{i-rank}(A \rightarrow B) &\triangleq \max(1 + \text{i-rank}(A), \text{i-rank}(B)), && \text{if } \cap \text{ occurs in } A \rightarrow B. \end{aligned}$$

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 decidable for i-rank 3, see [Urzyczyn \[2009\]](#). Decidability for i-rank 2 is done in Exercise 17F.24. Some other decidable cases are discussed in [Kurata and Takahashi \[1995\]](#).

From the point of view of 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 \[1983\]](#), [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, Ghilezan, and Venneri \[1997\]](#), [Wells, Dimock, Muller, and Turbak \[1997\]](#), [Capitani, Loret, and Venneri \[2001\]](#), [Ronchi Della Rocca \[2002\]](#), and [Pimentel, Ronchi Della Rocca, and Roversi \[In preparation\]](#), [Liquori and Ronchi Della Rocca \[2007\]](#), [Bono, Venneri, and Bettini \[2008\]](#) for this approach.

Undecidability of inhabitation in $\lambda_{\cap}^{\text{CDV}}$ vs. λ -definability in \mathcal{M}_X

During proofchecking this book [Salvati \[2009\]](#) showed that the undecidability of inhabitation follows directly from the undecidability of λ -definability in full type structures over a finite set, Theorem 4A.21. The main idea is a simple embedding of the elements of \mathcal{M}_X into the intersection types in \mathbb{T}_{\cap}^X .

17E.33. DEFINITION. Let X be a finite set. Consider the full type structure \mathcal{M}_X over X , see Definition 2D.17 Elements of $\mathcal{M}_X = \bigcup_{A \in \mathbb{T}_{\rightarrow}^0} X(A)$ can be encoded as elements of \mathbb{T}_{\cap}^X . To avoid confusion between simple and intersection types, we write $A, B, \dots \in \mathbb{T}_{\rightarrow}^0$ and $\sigma, \xi, \dots \in \mathbb{T}_{\cap}^X$. This helps to disambiguate \vdash as being either $\vdash_{\lambda_{\cap}^0}^{\text{Cu}}$ or $\vdash_{\lambda_{\cap}}^{\text{CDV}}$. For $d \in X(A)$ define $\xi_d \in \mathbb{T}_{\cap}^X$ by induction on the structure of $A \in \mathbb{T}_{\rightarrow}^0$.

$$\begin{aligned} \xi_d &\triangleq d, && \text{if } A = 0; \\ \xi_d &\triangleq \bigcap_{e \in X(B)} (\xi_e \rightarrow \xi_{de}), && \text{if } A = B \rightarrow C. \end{aligned}$$

17E.34. LEMMA. Let $d \in X(A \rightarrow B)$, $e \in X(A)$. Then we have

- (i) $\Gamma \vdash M : \xi_d \& \Gamma \vdash N : \xi_e \Rightarrow \Gamma \vdash (MN) : \xi_{de}$.
- (ii) $\Gamma \vdash (\lambda x.N) : \xi_d \Leftrightarrow \forall e \in X(A) [\Gamma, x : \xi_e \vdash N : \xi_{de}]$.

PROOF. (i) Since $\xi_d \leq \xi_e \rightarrow \xi_{de}$.

(ii) (\Rightarrow) By (i) and the subject reduction property for \vdash , being $\vdash_{\cap}^{\text{CDV}}$.

(\Leftarrow) By the rules $(\rightarrow\text{I})$ and $(\cap\text{I})$. ■

17E.35. LEMMA. (i) Let $\alpha \in X$ be a type atom and $\sigma \in \mathbb{T}_{\cap}^X$. Then

$$\alpha \leq_{\text{CDV}} \sigma \text{ or } \sigma \leq_{\text{CDV}} \alpha \Rightarrow \sigma \equiv \alpha.$$

(ii) Let $d, e \in \mathcal{M}_X$. Then

$$\xi_d \leq_{\text{CDV}} \xi_e \Rightarrow d = e.$$

PROOF. (i) By induction on the derivation of say $\alpha \leq \sigma$ in \mathcal{T}^{CDV} .

(ii) By induction on the joint size of ξ_d and ξ_e . If either of the two types is an atom, the result follows by (i). Otherwise, both types are intersections of arrows and we have $d \in X(A \rightarrow B)$ and $e \in X(C \rightarrow D)$, for some $A, B, C, D \in \mathbb{T}^0$. Thus for all $c \in X(C)$

$$\xi_d = (\xi_{a_1} \rightarrow \xi_{da_1}) \cap \cdots \cap (\xi_{a_k} \rightarrow \xi_{da_k}) \leq \xi_c \rightarrow \xi_{ec},$$

where $X(A) = \{a_1, \dots, a_k\}$. By Theorem 14A.7 (β -soundness) it follows that

$$\begin{aligned} \xi_c &\leq \xi_{a_{i_1}} \cap \cdots \cap \xi_{a_{i_p}} & (\leq a_{i_j}) & (1) \\ \xi_{da_{i_1}} \cap \cdots \cap \xi_{da_{i_p}} &\leq \xi_{ec} & (2), \end{aligned}$$

for a non-empty subset $\{a_{i_1}, \dots, a_{i_p}\} \subseteq X(A)$. By the induction hypothesis, we have $c = a_{i_1} = \dots = a_{i_p} = a$, say, and $A = C$. Hence (2) boils down to $\xi_{dc} \leq \xi_{ec}$. Again by the induction hypothesis one has $dc = ec$ and $B = D$. As c was arbitrary, it follows that the functions d and e are equal. ■

17E.36. DEFINITION. An *X -basis* is of the form

$$\Gamma = \{x_1 : \xi_{d_1}, \dots, x_n : \xi_{d_n}\},$$

with $d_1, \dots, d_n \in \mathcal{M}_X$.

17E.37. LEMMA. Define sets $\mathcal{N}, \mathcal{V} \subseteq \Lambda$ as the smallest sets such that

$$\begin{aligned} N_1, \dots, N_n \in \mathcal{N}, n \geq 0 &\Rightarrow xN_1 \cdots N_n \in \mathcal{V} \\ M \in \mathcal{V} &\Rightarrow M \in \mathcal{N} \\ M \in \mathcal{N} &\Rightarrow (\lambda x.M) \in \mathcal{N}. \end{aligned}$$

Then

$$\begin{aligned} \mathcal{N} &= \{M \in \Lambda \mid M \text{ in } \beta\text{-nf}\} \\ \mathcal{V} &= \{M \in \mathcal{N} \mid M \equiv x\vec{N}, \text{ for some } \vec{N} = N_1 \cdots N_n\}. \end{aligned}$$

17E.38. LEMMA. Consider two predicates P, Q on Λ :

$P(M) :=$ for all X -bases Γ and $d_1, d_2 \in X(A)$ one has

$$\Gamma \vdash M : \xi_d \& \Gamma \vdash M : \xi_{d'} \Rightarrow d = d'.$$

$Q(M) :=$ M is of the form $M \equiv xN_1 \cdots N_n$, $n \geq 0$, and for all X -bases Γ and $\sigma \in \mathbb{T}_{\cap}^X$ one has

$$\begin{aligned} \Gamma \vdash M : \sigma &\Rightarrow \exists e_1, \dots, e_n \in \mathcal{M}_X. [\Gamma \vdash N_i : \xi_{e_i}, 1 \leq i \leq n, \\ &\quad \& \Gamma \vdash M : \xi_{ce_1 \cdots e_n} \leq \sigma], \text{ where } \Gamma(x) = \xi_c. \end{aligned}$$

Then for all $M \in \Lambda$ one has

$$\begin{aligned} M \in \mathcal{N} &\Rightarrow P(M); \\ M \in \mathcal{V} &\Rightarrow Q(M). \end{aligned}$$

PROOF. By ‘induction on the simultaneous inductive definition’ of \mathcal{V} and \mathcal{N} in Lemma 17E.37. It suffices to show

$$\begin{aligned} (1) \quad P(N_1), \dots, P(N_n) &\Rightarrow Q(xN_1 \dots N_n) \\ (2) \quad P(N_1), \dots, P(N_n) &\Rightarrow P(xN_1 \dots N_n) \\ (3) \quad P(M) &\Rightarrow P(\lambda x.M). \end{aligned}$$

As to (1), suppose $P(N_1), \dots, P(N_n)$ towards $Q(xN_1 \dots N_n)$. We do this by induction on n . For $n = 0$ the assumption is $\Gamma \vdash x : \sigma$. By Theorem 14A.1(i) one has $\Gamma(x) \leq \sigma$, i.e. $\Gamma \vdash x : \xi_c \leq \sigma$, with $\Gamma(x) = \xi_c$. Therefore $Q(x)$. Now suppose $P(N_1), \dots, P(N_{n+1})$ towards $Q(xN_1 \dots N_{n+1})$. Assume $\Gamma \vdash xN_1 \dots N_{n+1} : \sigma$. Then by Theorem 14A.9(ii) one has

$$\exists \nu \in \mathbb{T}_{\cap}^X. [\Gamma \vdash xN_1 \dots N_n : \nu \rightarrow \sigma \ \& \ \Gamma \vdash N_{n+1} : \nu].$$

By the induction hypothesis one has $Q(xN_1 \dots N_n)$, hence for some $e_1, \dots, e_n \in \mathcal{M}_X$

$$\Gamma \vdash N_i : \xi_{e_i} \ (1 \leq i \leq n) \ \& \ \Gamma \vdash xN_1 \dots N_n : \xi_{ce_1 \dots e_n} \leq \nu \rightarrow \sigma.$$

Write $b = ce_1 \dots e_n$. By Lemma 17E.34(i) the intersection type ξ_b cannot be a type atom (in X), hence writing $X(\nu) = \{a_1, \dots, a_k\}$ we have

$$\xi_b = (\xi_{a_1} \rightarrow \xi_{ba_1}) \cap \dots \cap (\xi_{a_k} \rightarrow \xi_{ba_k}) \leq \nu \rightarrow \sigma.$$

By Theorem 14A.7 one has for some non-empty $\{i_1, \dots, i_p\} \subseteq \{1, \dots, k\}$

$$\begin{aligned} \nu &\leq \xi_{a_{i_1}} \cap \dots \cap \xi_{a_{i_p}} \\ &\quad \xi_{ba_{i_1}} \cap \dots \cap \xi_{ba_{i_p}} \leq \sigma. \end{aligned}$$

Since $P(N_{n+1})$ and $\Gamma \vdash N_{n+1} : \nu \leq \xi_{a_i}$ we have $a_{i_1} = \dots = a_{i_k} = a$, say. Noting that

$$\begin{aligned} \Gamma \vdash xN_1 \dots N_n : \xi_b &\leq \xi_a \rightarrow \xi_{ba} \\ \Gamma \vdash N_{k+1} : \nu &\leq \xi_a, \end{aligned}$$

taking $e_{n+1} = a$ it follows that

$$\Gamma \vdash xN_1 \dots N_{n+1} : \xi_{ba} = \xi_{ce_1 \dots e_{n+1}} \leq \sigma.$$

Therefore indeed $Q(xN_1 \dots N_{n+1})$.

As to (2), writing $M \equiv xN_1 \dots N_n$ suppose $P(N_i)$ towards $P(M)$. Assume

$$\Gamma \vdash M : \xi_d \ \& \ \Gamma \vdash M : \xi_{d'}$$

towards $d = d'$. As $Q(M)$, by (1), we have

$$\begin{aligned} \exists e_1, \dots, e_n. \Gamma \vdash M : \xi_{ce_1 \dots e_n} &\leq \xi_d \ \& \ \Gamma \vdash N_i : \xi_{e_i}, \\ \exists e'_1, \dots, e'_n. \Gamma \vdash M : \xi_{ce'_1 \dots e'_n} &\leq \xi_{d'} \ \& \ \Gamma \vdash N_i : \xi_{e'_i}. \end{aligned}$$

Since $P(N_i)$ we have $e_i = e'_i$. Therefore by Lemma 17E.34(ii) we have $d = c\vec{e} = c\vec{e}' = d'$.

As to (3), suppose $P(M)$ towards $P(\lambda x.M)$. Assume $\Gamma \vdash \lambda x.M : \xi_d$ and $\Gamma \vdash \lambda x.M : \xi_e$, in order to show $d = e$. By Theorem 14A.1(iii) and Lemma 17E.35(ii) it follows that $d \in X(A \rightarrow B)$. So $\xi_d \leq (\xi_c \rightarrow \xi_{dc})$ for all $c \in X(A)$. Then for all $c \in X(A)$

$$\begin{aligned} \Gamma &\vdash \lambda x.M : \xi_c \rightarrow \xi_{dc}, \\ \Gamma, x : \xi_c &\vdash M : \xi_{dc}, \quad \text{by Theorem 14A.9(iii).} \end{aligned}$$

Similarly $\Gamma, x : \xi_c \vdash M : \xi_{ec}$. By $P(M)$ we have $\forall c \in X(A).dc = ec$. Therefore $d = e$. ■

17E.39. PROPOSITION. Let $A \in \mathbb{T}^0$, $M \in \Lambda_{\rightarrow}(A)$, and $d \in X(A)$. For a valuation ρ in \mathcal{M}_X , write $\Gamma_\rho(x) = \xi_{\rho(x)}$. Then

- (i) $\llbracket M \rrbracket_\rho = d \Leftrightarrow \Gamma_\rho \upharpoonright \text{FV}(M) \vdash |M| : \xi_d$.
- (ii) If $M \in \Lambda_{\rightarrow}^\circ(A)$, then

$$\llbracket M \rrbracket = d \Leftrightarrow \vdash |M| : \xi_d.$$

PROOF. (i) (\Rightarrow) By induction on M one shows

$$\Gamma_\rho \upharpoonright \text{FV}(M) \vdash |M| : \xi_{\llbracket M \rrbracket_\rho}.$$

We write Γ_ρ for $\Gamma_\rho \upharpoonright \text{FV}(M)$.

Case $M \equiv x^A$. Then $\Gamma_\rho \vdash x : \xi_{\rho(X)}$, as $\Gamma(x) = \xi_{\rho(x)}$.

Case $M \equiv NL$. By the induction hypothesis one has

$$\Gamma_\rho \vdash |N| : \xi_{\llbracket N \rrbracket_\rho} \& \Gamma_\rho \vdash |L| : \xi_{\llbracket L \rrbracket_\rho}.$$

Therefore by Lemma 17E.34(i)

$$\Gamma_\rho \vdash |NL| : \xi_{\llbracket N \rrbracket_\rho \llbracket L \rrbracket_\rho} = \xi_{\llbracket NL \rrbracket_\rho}.$$

Case $M \equiv \lambda x^B.N$. By the induction hypothesis one has for all $b \in X(B)$

$$\Gamma_{\rho[x:=b]} \vdash |N| : \xi_{\llbracket N \rrbracket_{\rho[x:=b]}}.$$

Hence

$$\begin{aligned} \Gamma_\rho, x : \xi_b \vdash |N| : \xi_{\llbracket N \rrbracket_{\rho[x:=b]}} &\Rightarrow \Gamma_\rho \vdash \lambda x.|N| : \xi_b \rightarrow \xi_{\llbracket N \rrbracket_{\rho[x:=b]}} \\ &\Rightarrow \Gamma_\rho \vdash |\lambda x^B.N| : \bigcap_{b \in X(B)} \cdot \xi_b \rightarrow \xi_{\llbracket N \rrbracket_{\rho[x:=b]}} = \xi_{\llbracket \lambda x^B.N \rrbracket_\rho}. \end{aligned}$$

(\Leftarrow) Assume $\Gamma_\rho \vdash |M| : \xi_d$. Then by normalization and subject reduction one has

$$\Gamma_\rho \vdash |M^{\text{nf}}| : \xi_d,$$

where M^{nf} is the β -nf of M . By (\Rightarrow) one has

$$\Gamma_\rho \vdash |M^{\text{nf}}| : \xi_{\llbracket M^{\text{nf}} \rrbracket_\rho}.$$

By Lemmas 17E.37 and 17E.38 one has $P(M^{\text{nf}})$. Therefore

$$d = \llbracket M^{\text{nf}} \rrbracket_\rho = \llbracket M \rrbracket_\rho.$$

(ii) By (i). ■

17E.40. LEMMA. For $d \in X(A)$ with $A \in \mathbb{T}^0$, define $\xi_d^\sim = A$. For $\Gamma = \{x_1:\xi_{d_1}, \dots, x_n:\xi_{d_n}\}$, an X -basis, define $\Gamma^\sim = \{x_1:\xi_{d_1}^\sim, \dots, x_n:\xi_{d_n}^\sim\}$. Then for $M \in \Lambda$ in β -nf one has

$$\Gamma \vdash_{\lambda_\cap} M : \xi_d \Rightarrow \Gamma^\sim \vdash_{\lambda_\cap^0}^{Cu} M : \xi_d^\sim.$$

PROOF. By induction on the generation of M .

Case $M \equiv x$. Then $\Gamma \vdash_{\lambda_\cap} x : \xi_d$ implies $\Gamma(x) \leq \xi_d$, so $\Gamma(x) = \xi_d$, by Lemma 17E.35(ii). Then $\Gamma^\sim \vdash_{\lambda_\cap^0}^{Cu} x : \xi_d^\sim$.

Case $M \equiv \lambda x.N$. Then $\Gamma \vdash_{\lambda_\cap} \lambda x.N : \xi_e = \bigcap_{a \in X(A)} (\xi_a \rightarrow \xi_{ea})$, with $e \in X(A \rightarrow B)$ implies $\Gamma, x:\xi_a \vdash_{\lambda_\cap} N : \xi_{ea}$, for all $a \in X(A)$. Therefore by Lemma 17E.34(ii) one has

$$\Gamma^\sim \vdash_{\lambda_\cap^0}^{Cu} (\lambda x.N) : (A \rightarrow B).$$

Case $M \equiv xN_1 \cdots N_n$. Suppose $\Gamma \vdash_{\lambda_\cap} xN_1 \cdots N_n : \xi_d$. Then by Lemmas 17E.38 and 17E.35(ii) one has $d = ca_1 \cdots a_n$ with $\Gamma(x) = \xi_c$, $\Gamma \vdash_{\lambda_\cap} N_i : \xi_{a_i}$, and $a_i \in X(A_i)$, and $c \in X(A_1 \rightarrow \cdots \rightarrow A_n \rightarrow B)$. By the induction hypothesis one has

$$\Gamma^\sim \vdash_{\lambda_\cap^0}^{Cu} N_i : A_i \ \& \ \Gamma^\sim \vdash x : (A_1 \rightarrow \cdots \rightarrow A_n \rightarrow B).$$

Therefore

$$\Gamma^\sim \vdash_{\lambda_\cap^0}^{Cu} xN_1 \cdots N_n : B = \xi_d^\sim,$$

as $d = ca_1 \cdots a_n \in X(B)$ by Lemma 17E.34(i). ■

17E.41. THEOREM (Salvati). The undecidability of inhabitation in \mathcal{T}^{CDV} follows by a reduction from the undecidability of λ -definability, Theorem 4A.21.

PROOF. Given $A \in \mathbb{T}^0$ and $d \in X(A)$ we claim that

$$\exists M \in \Lambda_{\rightarrow}^0(A) \llbracket M \rrbracket = d \Leftrightarrow \exists M \in \Lambda^0 \vdash_{\lambda_\cap} M : \xi_d.$$

(\Rightarrow) By Proposition 17E.39(ii).

$$\begin{aligned} (\Leftarrow) \vdash_{\lambda_\cap} M \in \xi_d &\Rightarrow \vdash_{\lambda_\cap^0}^{Cu} M : A, && \text{by Proposition 17E.40,} \\ &\Rightarrow M^+ \in \Lambda_{\rightarrow}^0(A) \ \& \ |M^+| = M, && \text{by Proposition 1B.19(ii),} \\ &\Rightarrow \llbracket M^+ \rrbracket = d, && \text{by Proposition 17E.39(ii).} \end{aligned}$$

Therefore d is definable iff ξ_d is inhabited. This yields a reduction of the definability problem to the inhabitation problem. ■

In a personal communication Sylvain Salvati mentioned that the proof (not the statement) of the undecidability of inhabitation in \mathcal{T}^{CDV} also implies undecidability of λ -definability in $\mathcal{M}_{\mathcal{P}(X)}$. In the reduction of inhabitation in λ_\cap^{CDV} to λ -definability of elements in the full λ -structure over a finite set, the intersection types are interpreted as non-deterministic finite automata, while the elements of the full type structure correspond to deterministic finite automata. This explains the need of the power-set construction in $\mathcal{M}_{\mathcal{P}(X)}$, which is similar to the one for making a finite automaton deterministic: states of the new automaton are sets of states of the former one.

17F. Exercises

17F.1. Show by means of examples that the type theories Plotkin and Engeler are not complete, i.e. we do not have $\Gamma \models^{\mathcal{T}} M : A \Leftrightarrow \Gamma \vdash_{\cap}^{\mathcal{T}} M : A$, for $\mathcal{T} = \text{Plotkin}$ or $\mathcal{T} = \text{Engeler}$.

17F.2. Show that $\Sigma(\{0\}, \text{CDV})$ is the smallest complete type theory (w.r.t. the order of Fig. 34).

17F.3. Show that for all $\mathcal{T} \in \text{PTT}$ one has

$$\Gamma \vdash_{\cap}^{\mathcal{T}} M : A \Rightarrow \Gamma \models_{\cap}^{\mathcal{T}} M : A.$$

[Hint. Adapting Definition 17A.1 to proper intersection type theories in the obvious way.]

17F.4. Let $\mathcal{T} \in \text{TT}^{-\text{U}}$. Take $\mathcal{F}^{\mathcal{T}} = \langle \mathcal{F}^{\mathcal{T}}, \cdot, \llbracket \] \rrbracket^{\mathcal{F}^{\mathcal{T}}} \rangle$ as in 16B.1; define the type environment $\xi^{\mathcal{T}} : \mathbb{A}^{\mathcal{T}} \rightarrow \mathsf{P}(\mathcal{F}^{\mathcal{T}})$ by

$$\xi^{\mathcal{T}}(\alpha) \triangleq \{X \in \mathcal{F}^{\mathcal{T}} \mid \alpha \in X\};$$

and let $\llbracket \] \rrbracket^{\mathcal{T}} : \mathbb{A}^{\mathcal{T}} \rightarrow \mathsf{P}(\mathcal{F}^{\mathcal{T}})$ be the map $\llbracket \] \rrbracket_{\xi^{\mathcal{T}}}^{\mathcal{F}^{\mathcal{T}}}$ defined by deleting the first clause from Definition 17A.1. Show the following.

- (i) $\llbracket A \rrbracket^{\mathcal{T}} = \{X \in \mathcal{F}^{\mathcal{T}} \mid A \in X\}$.
- (ii) $\mathcal{F}^{\mathcal{T}}, \xi^{\mathcal{T}}$ are \rightarrow -good and preserve $\leq_{\mathcal{T}}$.
- (iii) $[\Gamma \models_{\cap}^{\mathcal{T}} M : A \Rightarrow \Gamma \vdash_{\cap}^{\mathcal{T}} M : A] \Leftrightarrow \mathcal{T} \in \text{PTT}$.
- (iv) $\Gamma \models_{\cap}^{\mathcal{T}} M : A \Leftrightarrow \Gamma \vdash_{\cap}^{\mathcal{T}} M : A$.

17F.5. Show that all quasi λ -models and all type environments preserve \leq_{BCD} .

17F.6. [Dezani-Ciancaglini, Honsell, and Motohama \[2005\]](#), [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 $\vec{M} \in \mathbb{N}$. 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 $\vec{M} \in \mathbb{N}$. Are these all of them? Is this enough to characterize them?
4. Conclude that $\text{PSN} \subset \text{PN} \subset \text{PHN}$.

17F.7. Show that PHN is HN -stable and PN is N -stable.

17F.8. Let $\mathcal{M}_{\Lambda(\beta)} = \langle \Lambda, \cdot, \llbracket \] \rrbracket^{\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 if $M \vec{N}$ has a head normal form for all terms \vec{N} (see Definition 17B.3). Define the type environment

$$\xi_{\text{Scott}}(0) \triangleq \{[M] \mid M \text{ is persistently head normalizing}\}.$$

Prove that $(\mathcal{M}_{\Lambda(\beta)}, \xi_{\text{Scott}})$ preserves \leq_{Scott} .

17F.9. Let $\mathcal{M}_{\Lambda(\beta)}$ and $[M]$ be as in Exercise 17F.8. Define the type environment

$$\xi_{\text{Park}}(0) \triangleq \{[M] \mid M \text{ reduces to a closed term}\}.$$

Prove that $(\mathcal{M}_{\Lambda(\beta)}, \xi_{\text{Park}})$ preserves \leq_{Park} .

17F.10. A term $(\lambda x.M)N$ is a [\$\beta\text{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 $\rightarrow_{\beta N}$ the induced reduction. Show that if Γ assigns types to all free variables in N , i.e. $x \in \text{dom}(\Gamma)$ for all $x \in \text{FV}(N)$, then

$$\Gamma \vdash_{\cap}^{\text{HL}} M : A \ \& \ N \rightarrow_{\beta N} M \Rightarrow \Gamma \vdash_{\cap}^{\text{HL}} N : A.$$

[Hint: use Theorem 17B.15(iii).]

- 17F.11. Show that in the system induced by the type theory AO, the terms typable with type $U \rightarrow U$ 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.
- 17F.12. Show that in the system induced by the type theory EHR, as defined in Definition 13B.13, the terms typable with type V in the context all whose predicates are V are precisely the terms which reduce either to an abstraction or to a variable using the call-by-value β -reduction rule.
- 17F.13. Show that all type interpretation domains and all type environments agree with AO.
- 17F.14. Using the Approximation Theorem, show that
- there is no type deducible for Ω in the system $\vdash_{\cap}^{\text{CDV}}$;
 - $\vdash_{\cap}^{\text{BCD}} \Omega : A$ iff $A =_{\text{BCD}} U$;
 - $\vdash_{\cap}^{\text{Park}} \Omega : A$ iff $0 \leq_{\text{Park}} A$.
- 17F.15. Using the Approximation Theorem, show that in the system $\lambda_{\cap}^{\text{AO}}$ the set of types deducible for $\omega\omega$ is strictly included in the set of types deducible for $K(\Omega)$.
- 17F.16. Using the Approximation Theorem, show that in the system $\lambda_{\cap}^{\text{Scott}}$ the set of types deducible for J and I coincide.
- 17F.17. Prove that typability in $\lambda_{\cap}^{\text{CDV}}$ is undecidable.
- 17F.18. Prove Lemma 17E.3(i) and (ii). [Hint. Similar to the proof of Lemma 13A.22.]
- 17F.19. Consider the type assignment systems $\lambda_{\cap}^{\text{Krivine}}$ and $\lambda_{\cap}^{\text{Krivine}^U}$ as defined in Exercise 13E.10.
- (i) Prove an analogue of Lemma 17E.9.
 - (ii) Prove that if Γ is a game context then $\Gamma \vdash^{\text{Krivine}} M : \alpha$ and $\Gamma \vdash^{\text{Krivine}^U} M : \alpha$ 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 $\delta \cap (\alpha \rightarrow \beta) \cap (\alpha \rightarrow \gamma) \rightarrow \delta \cap (\alpha \rightarrow \beta \cap \gamma)$ is inhabited in $\lambda_{\cap}^{\text{BCD}}$ but is not inhabited in $\lambda_{\cap}^{\text{Krivine}^U}$.
- 17F.20. Let Γ be a game context and $\alpha \in A_\infty$. Prove that if $\Gamma \vdash M : \alpha$ then every node in the Böhm tree of M (see B[1984], Ch. 10) has at most one branch.
- 17F.21. Complete the proofs of Proposition 17E.23 and Lemma 17E.24.
- 17F.22. Prove Lemma 17E.28. [Hint. Compare G_1 to the game G_0 of Example 17E.16. Observe that in each sequence of positions

$$T_{(2k+1)n}, \dots, T_{(2k+3)n},$$

the odd steps behave as an initial phase of G_0 , while the even steps behave as a final phase of G_0 . Writing

$$\begin{aligned} \perp_i &\triangleq \langle \perp, i, G \rangle; \\ A &\triangleq \langle a, 1, \{G, L\} \rangle \langle a, 2, \{G, L\} \rangle; \\ B &\triangleq \langle b, 1, \{G, L\} \rangle \langle b, 2, \{G, L\} \rangle \end{aligned}$$

we have the following (the canopy of a tree is the collection of its leaves) for the case of two initial C_1 steps.

position #	via move	canopy of position
0		\perp_1
1	C_1	$\perp_1 \perp_2$
2	C_1	$(\perp_1 \perp_2)^2$
3	C_2	$A^2 A^2$
4	1	$A^2 B^2$
5	C_3	$A^4 B^4$
6	2	$(A^2 B^2)^2$
7	C_3	$(A^4 B^4)^2$
8	3	$(A^2 B^2)^4$
9	C_3	$(A^4 B^4)^4$
10	5	$(A^2 B^2)^8$
11	C_3	$(A^4 B^4)^8$
12	7	$(A^2 B^2)^{16}$
...
$4 + 2k$		$(A^2 B^2)^{2^k}$

If one starts with m moves of C_1 ($0 < m < \infty$), then the canopy 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.]

- 17F.23. Prove Lemma 17E.26. [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

$$\$ \$ \cdots \$ <011100010> \# \cdots \# \#,$$

with a certain number of the \$’s and #’s. The initial empty queue is just “<># · · · #”. Now an *insert* instruction means: *replace “>” with a digit and replace the first “#” with “>”, and similarly for a remove*. The number of \$’s increases after each *remove*, while the suffix of #’s shrinks after each *insert*, so that the queue “moves to the right”. If the number of the initial suffix of #’s is sufficiently large, then a typewriter automaton can verify the queue computation.]

- 17F.24. [Kuśmierek \[2007\]](#). Prove that in $\lambda_{\cap}^{\text{CDV}}$ the inhabitation problem for types of i-rank at most 2 is decidable. See definition of i-rank after Theorem 17E.31. Note that if the i-rank of $B \rightarrow C$ is at most 2 then the i-rank of B is at most 1.

- 17F.25. [Kuśmierek \[2007\]](#).

- (i) Let $\iota \triangleq (\alpha \rightarrow \alpha) \cap (\beta \rightarrow \beta)$ and define for $k = 0, \dots, n$ the type

$$A_k \triangleq \alpha \rightarrow \iota^k \rightarrow (\alpha \rightarrow \beta) \rightarrow (\beta \rightarrow \alpha)^{n-k} \rightarrow \beta.$$

Prove that the shortest inhabitant of A_1, \dots, 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?

17F.26. Show that inhabitation in λ_n^{BCD} is undecidable. [Hint. Show that λ_n^{BCD} is conservative over λ_n^{CDV} for β -normal forms, that is

$$\Gamma \vdash^{BCD} M : A \Leftrightarrow \Gamma \vdash^{CDV} M : A,$$

for all $A \in \mathbb{T}^{CDV}$, Γ with types from \mathbb{T}^{CDV} , and $M \in \Lambda$ in β -normal form. Use Theorem 17B.15(i).]

BIBLIOGRAPHY

M. ABADI AND L. CARDELLI

[1996] *A Theory of Objects*, Springer.

M. ABADI AND M. P. FIORE

[1996] *Syntactic considerations on recursive types*, *Logic in Computer Science*, IEEE Computer Society Press, pp. 242–252.

M. ABADI AND G. D. PLOTKIN

[1990] *A PER model of polymorphism and recursive types*, *Logic in Computer Science*, IEEE Computer Society Press, pp. 355–365.

H. ABELSON, R. K. DYBVIK, C. T. HAYNES, G. J. ROZAS, N. I. ADAMS IV, D. P. FRIEDMAN, E. KOHL

[1991] *Revised Report on the Algorithmic Language Scheme*, *ACM SIGPLAN Lisp Pointers*, vol. IV, no. 3, pp. 1–55.

S. ABRAMSKY

[1990] *The lazy lambda calculus*, *Research Topics in Functional Programming* (D. A. Turner, editor), Addison-Wesley, pp. 65–116.

[1991] *Domain theory in logical form*, *Annals of Pure and Applied Logic*, vol. 51, no. 1-2, pp. 1–77.

S. ABRAMSKY AND A. JUNG

[1994] *Domain theory*, *Handbook for logic in computer science* (S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors), vol. 3, Clarendon Press, pp. 1–168.

S. ABRAMSKY AND C.-H. L. ONG

[1993] *Full abstraction in the lazy lambda calculus*, *Information and Computation*, vol. 105, no. 2, pp. 159–267.

W. ACKERMANN

[1928] *Zum Hilbertschen Aufbau der reellen Zahlen*, *Mathematische Annalen*, vol. 99, pp. 118–133.

P. ACZEL

[1988] *Non-Well-Founded Sets*, Center for the Study of Language and Information, Stanford.

- A. V. AHO, R. SETHI, AND J. D. ULLMAN
[1986] *Compilers*, Addison-Wesley.
- F. ALESSI
[1991] *Strutture di tipi, teoria dei domini e modelli del lambda calcolo*, *Ph.D. thesis*, Torino University.
[1993] *The p model*, Internal Report, Udine University.
- F. ALESSI AND F. BARBANERA
[1991] *Strong conjunction and intersection types*, *Mathematical foundations of computer science*, Lecture Notes in Computer Science, Springer, pp. 64–73.
- F. ALESSI, F. BARBANERA, AND M. DEZANI-CIANCAGLINI
[2003] *Intersection types and computational rules*, *Workshop on logic, language, information and computation*, Electronic Notes in Theoretical Computer Science, vol. 84, Elsevier, pp. 1–15.
[2004] *Tailoring filter models*, *Types*, Lecture Notes in Computer Science, vol. 3085, Springer, pp. 17–33.
[2006] *Intersection types and lambda models*, *Theoretical Computer Science*, vol. 355, no. 2, pp. 108–126.
- F. ALESSI, M. DEZANI-CIANCAGLINI, AND F. HONSELL
[2001] *Filter models and easy terms*, *Italian conference on theoretical computer science*, Lecture Notes in Computer Science, vol. 2202, Springer, pp. 17–37.
[2004] *Inverse limit models as filter models*, *International workshop on higher-order rewriting*, RWTH Aachen, pp. 3–25.
- J.-P. ALLOUCHE AND J. SHALLIT
[2003] *Automatic sequences*, Cambridge University Press.
- R. M. AMADIO
[1991] *Recursion over realizability structures*, *Information and Computation*, vol. 91, no. 1, pp. 55–85.
- R. M. AMADIO AND L. CARDELLI
[1993] *Subtyping recursive types*, *ACM Transactions on Programming Languages and Systems*, vol. 15, no. 4, pp. 575–631.
- R. M. AMADIO AND P.-L. CURIEN
[1998] *Domains and lambda-calculi*, Cambridge University Press.
- PETER B. ANDREWS
[2002] *An introduction to mathematical logic and type theory: To truth through proof*, Applied Logic, vol. 27, Springer.
- A. W. APPEL
[1992] *Compiling with continuations*, Cambridge University Press.

- A. W. APPEL AND D. MCALLESTER
[2001] *An indexed model of recursive types for foundational proof-carrying code*, *ACM Transactions on Programming Languages and Systems*, vol. 23, no. 5, pp. 657–683.
- Z. M. ARIOLA AND J. W. KLOP
[1994] *Cyclic lambda graph rewriting*, *Logic in Computer Science*, IEEE Computer Society Press, pp. 416–425.
[1996] *Equational term graph rewriting*, *Fundamenta Informaticae*, vol. 26, no. 3–4, pp. 207–240.
- L. AUGUSTSON
[1999] *Cayenne a language with dependent types*, *Advanced functional programming*, Lecture Notes in Computer Science, vol. 1608, Springer, pp. 240–267.
- J. AVIGAD, K. DONNELLY, D. GRAY, AND PAUL RAFF
[2007] *A formally verified proof of the prime number theorem*, *ACM Transactions on Computational Logic*, vol. 9, no. 1-2, pp. 1–23.
- F. BAADER AND T. NIPKOW
[1998] *Term rewriting and all that*, Cambridge University Press.
- F. BAADER AND W. SNYDER
[2001] *Unification theory*, *Handbook of automated reasoning* (J.A. Robinson and A. Voronkov, editors), vol. I, Elsevier, pp. 447–533.
- J. W. BACKUS
[1978] *Can programming be liberated from the von Neumann style?*, *Communication of the ACM*, vol. 21, pp. 613–641.
- J. BAETEN AND B. BOERBOOM
[1979] ω can be anything it shouldn't be, *Indagationes Mathematicae*, vol. 41, pp. 111–120.
- S. VAN BAKEL
[1992] *Complete restrictions of the intersection type discipline*, *Theoretical Computer Science*, vol. 102, no. 1, pp. 135–163.
[1993] *Principal type schemes for the strict type assignment system*, *Journal of Logic and Computation*, vol. 3, no. 6, pp. 643–670.
- S. VAN BAKEL, F. BARBANERA, M. DEZANI-CIANCAGLINI, AND F.-J. DE VRIES
[2002] *Intersection types for lambda-trees*, *Theoretical Computer Science*, vol. 272, no. 1-2, pp. 3–40.
- J. BALDRIDGE
[2002] *Lexically specified derivational control in Combinatory Categorial Grammar*, *Ph.D. thesis*, University of Edinburgh.

- F. BARBANERA, M. DEZANI-CIANCAGLINI, AND U. DE'LIGUORO
[1995] *Intersection and union types: syntax and semantics*, *Information and Computation*, vol. 119, no. 2, pp. 202–230.
- H. P. BARENDREGT
[1974] *Pairing without conventional restraints*, *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, vol. 20, pp. 289–306.
[1984] *The Lambda Calculus: its Syntax and Semantics*, revised ed., North-Holland.
[1991] *Theoretical pearls: Self-interpretation in lambda calculus*, *Journal of Functional Programming*, vol. 1, no. 2, pp. 229–233.
[1992] *Lambda calculi with types*, *Handbook for logic in computer science* (S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors), Oxford University Press, pp. 117–309.
[1994] *Discriminating coded lambda terms*, *From universal morphisms to megabytes: A Baayen Space-Odyssey* (K.R. Apt, A.A. Schrijver, and N.M. Temme, editors), CWI, pp. 141–151.
[1995] *Enumerators of lambda terms are reducing constructively*, *Annals of Pure and Applied Logic*, vol. 73, pp. 3–9.
- H. P. BARENDREGT AND E. BARENSEN
[1997] *Efficient computations in formal proofs*, *Journal of Automated Reasoning*, pp. 321–336.
- H. P. BARENDREGT, M. BUNDER, AND W. DEKKERS
[1993] *Systems of illative combinatory logic complete for first order propositional and predicate calculus*, *The Journal of Symbolic Logic*, vol. 58, no. 3, pp. 89–108.
- H. P. BARENDREGT, M. COPPO, AND M. DEZANI-CIANCAGLINI
[1983] *A filter lambda model and the completeness of type assignment*, *The Journal of Symbolic Logic*, vol. 48, no. 4, pp. 931–940.
- H. P. BARENDREGT AND S. GHILEZAN
[2000] *Lambda terms for natural deduction, sequent calculus and cut-elimination*, *Journal of Functional Programming*, vol. 10, pp. 121–134.
- H. P. BARENDREGT AND A. REZUS
[1983] *Semantics for classical Automath and related systems*, *Information and Control*, vol. 59, pp. 127–147.
- E. BARENSEN AND J. E. W. SMETSERS
[1993] *Conventional and uniqueness typing in graph rewrite systems (extended abstract)*, *Foundations of software technology and theoretical computer science*, Lecture Notes in Computer Science, vol. 761, Springer, pp. 41–51.
[1996] *Uniqueness typing for functional languages with graph rewriting semantics*, *Mathematical Structures in Computer Science*, vol. 6, no. 6, pp. 579–612.

H. BEKIĆ

[1984] *Programming languages and their definition*, Lecture Notes in Computer Science, vol. 177, Springer-Verlag, Berlin, Selected papers edited by C. B. Jones.

J. F. A. K. VAN BENTHEM

[1995] *Language in action: Categories, lambdas, and dynamic logic*, The MIT Press.

J. F. A. K. VAN BENTHEM AND A. TER MEULEN

[1997] *Handbook of logic and language*, Elsevier and MIT Press.

L. S. VAN BENTHEM JUTTING

[1977] *Checking Landau's "Grundlagen" in the Automath System*, *Ph.D. thesis*, Eindhoven University of Technology.

A. BERARDUCCI AND C. BÖHM

[1993] *A self-interpreter of lambda calculus having a normal form*, *Computer science logic*, Lecture Notes in Computer Science, vol. 702, Springer, pp. 85–99.

C. BERLINE

[2000] *From computation to foundations via functions and application : the lambda-calculus and its webbed models*, *Theoretical Computer Science*, vol. 249, pp. 81–161.

R. BERNARDI

[2002] *Reasoning with polarity in categorial type logic*, *Ph.D. thesis*, Utrecht Institute of Linguistics OTS.

Y. BERTOT AND P. CASTÉRAN

[2004] *Interactive theorem proving and program development*, Texts in Theoretical Computer Science, Springer.

M. A. BEZEM

[1985a] *Strong normalization of bar recursive terms without using infinite terms*, *Archiv für Mathematische Logik und Grundlagenforschung*, vol. 25, pp. 175–181.

[1985b] *Strongly majorizable functionals of finite type: A model for barrecursion containing discontinuous functionals*, *The Journal of Symbolic Logic*, vol. 50, no. 3, pp. 652–660.

[1989] *Compact and majorizable functionals of finite type*, *The Journal of Symbolic Logic*, vol. 54, no. 1, pp. 271–280.

L. BIRKEDAL AND R. HARPER

[1999] *Constructing interpretations of recursive types in an operational setting*, *Information and Computation*, vol. 155, pp. 3–63.

C. BÖHM

[1966] *The CUCH as a Formal and Description Language*, *Formal languages description languages for computer programming*, North-Holland, pp. 179–197.

- [1975] **λ -calculus and Computer Science Theory**, Lecture Notes in Computer Science, vol. 37, Springer.
- C. BÖHM AND A. BERARDUCCI
 [1985] *Automatic synthesis of typed Λ -programs on term algebras*, **Theoretical Computer Science**, vol. 39, pp. 135–154.
- C. BÖHM AND M. DEZANI-CIANCAGLINI
 [1975] *λ -terms as total or partial functions on normal forms*, **λ -calculus and Computer Science Theory**, Lecture Notes in Computer Science, vol. 37, Springer, pp. 96–121.
- C. BÖHM AND W. GROSS
 [1966] *Introduction to the CUCH, Automata theory* (E.R. Caianiello, editor), Academic Press, pp. 35–65.
- C. BÖHM, A. PIPERNO, AND S. GUERRINI
 [1994] *Lambda-definition of function(al)s by normal forms*, **European Symposium on Programming**, vol. 788, Springer, pp. 135–154.
- V. BONO, B. VENNERI, AND L. BETTINI
 [2008] *A typed lambda calculus with intersection types*, **Theoretical Computer Science**, vol. 398, no. 1-3, pp. 95–113.
- A. BOVE, P. DYBJER, AND U. NORELL
 [2009] *A Brief Overview of Agda A Functional Language with Dependent Types*, **Theorem proving in higher order logics**, Lecture Notes in Computer Science, vol. 5674, Springer, pp. 73–78.
- M. BRANDT AND F. HENGLEIN
 [1998] *Coinductive axiomatization of recursive type equality and subtyping*, **Fundamenta Informaticæ**, vol. 33, pp. 309–338.
- V. BREAZU-TANNEN AND A. R. MEYER
 [1985] *Lambda calculus with constrained types*, **Logics of programs**, Lecture Notes in Computer Science, vol. 193, Springer, pp. 23–40.
- K. B. BRUCE, A. R. MEYER, AND J. C. MITCHELL
 [1990] *The semantics of second-order lambda calculus*, **Logical Foundations of Functional Programming**, University of Texas at Austin Year of Programming Series, Addison Wesley, pp. 213–272.
- N. G. DE BRUIJN
 [1970] *The mathematical language AUTOMATH, its usage and some of its extensions*, **Symposium on Automatic Demonstration**, Lecture Notes in Mathematics, no. 125, Springer, pp. 29–61.

[1972] *Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem*, *Indagationes Mathematicae*, vol. 34, pp. 381–392.

[1994] *Reflections on Automath*, *Selected papers on Automath* (R. P. Nederpelt, J. H. Geuvers, and R. C. de Vrijer, editors), Studies in Logic and the Foundations of Mathematics, no. 133, North-Holland, pp. 201–228.

R. M. BURSTALL

[1969] *Proving properties of programs by structural induction*, *Computer Journal*, vol. 12, no. 1, pp. 41–48.

W. BUSZKOWSKI, W. MARCISZEWSKI, AND J. F. A. K. VAN BENTHEM

[1988] *Categorial grammar*, Linguistic & Literary Studies in Eastern Europe **25**, John Benjamins.

W. BUSZKOWSKI AND G. PENN

[1990] *Categorial grammars determined from linguistic data by unification*, *Studia Logica*, vol. 49, no. 4, pp. 431–454.

B. CAPITANI, M. LORETI, AND B. VENNERI

[2001] *Hyperformulae, parallel deductions and intersection types*, *Electronic Notes in Theoretical Computer Science*, vol. 50, no. 2, pp. 1–18.

V. CAPRETTA 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*, vol. 9, no. 6, pp. 719–739.

L. CARDELLI

[1988] *A semantics of multiple inheritance*, *Information and Computation*, vol. 76, no. 2-3, pp. 138–164.

F. CARDONE

[1989] *Relational semantics for recursive types and bounded quantification*, *Automata, Languages and Programming*, Lecture Notes in Computer Science, vol. 372, Springer, pp. 164–178.

[1991] *Recursive types for Fun*, *Theoretical Computer Science*, vol. 83, pp. 29–56.

[2002] *A coinductive completeness proof for the equivalence of recursive types*, *Theoretical Computer Science*, vol. 275, pp. 575–587.

F. CARDONE AND M. COPPO

[1990] *Two extensions of Curry's type inference system*, *Logic and computer science* (P. Odifreddi, editor), APIC Studies in Data Processing, vol. 31, Academic Press, pp. 19–76.

[1991] *Type inference with recursive types. Syntax and Semantics*, *Information and Computation*, vol. 92, no. 1, pp. 48–80.

- [2003] *Decidability properties of recursive types*, ***Italian Conference on theoretical Computer Science***, Lecture Notes in Computer Science, vol. 2841, Springer, pp. 242–255.
- F. CARDONE, M. DEZANI-CIANCAGLINI, AND U. DE'LGUORO
 [1994] *Combining type disciplines*, ***Annals of Pure and Applied Logic***, vol. 66, no. 3, pp. 197–230.
- F. CARDONE AND J. R. HINDLEY
 [2009] *Lambda-calculus and combinators in the 20th century*, ***Handbook of the history of logic*** (D. M. Gabbay and J. Woods, editors), vol. 5: Logic from Russell to Church, Elsevier, pp. 723–817.
- A. CHURCH
 [1932] *A set of postulates for the foundation of logic (1)*, ***Annals of Mathematics***, vol. 33, pp. 346–366.
 [1933] *A set of postulates for the foundation of logic (2)*, ***Annals of Mathematics***, vol. 34, pp. 839–864.
 [1936] *An unsolvable problem of elementary number theory*, ***American Journal of Mathematics***, vol. 58, pp. 354–363.
 [1940] *A formulation of the simple theory of types*, ***The Journal of Symbolic Logic***, vol. 5, pp. 56–68.
 [1941] *The Calculi of Lambda-Conversion*, Princeton University Press, Annals of Mathematics Studies, no. 6.
- A. CHURCH AND J. B. ROSSER
 [1936] *Some properties of conversion*, ***Transactions of the American Mathematical Society***, vol. 39, pp. 472–482.
- H. COMON AND Y. JURSKI
 [1998] *Higher-order matching and tree automata*, ***Computer Science Logic***, Lecture Notes in Computer Science, vol. 1414, Springer, pp. 157–176.
- M. COPPO
 [1985] *A completeness theorem for recursively defined types*, ***Automata, Languages and Programming***, Lecture Notes in Computer Science, vol. 194, Springer, pp. 120–129.
- M. COPPO AND M. DEZANI-CIANCAGLINI
 [1980] *An extension of the basic functionality theory for the λ -calculus*, ***Notre Dame Journal of Formal Logic***, vol. 21, no. 4, pp. 685–693.
- M. COPPO, M. DEZANI-CIANCAGLINI, F. HONSELL, AND G. LONGO
 [1984] *Extended type structures and filter lambda models*, ***Logic Colloquium***, North-Holland, pp. 241–262.
- M. COPPO, M. DEZANI-CIANCAGLINI, AND G. LONGO

- [1983] *Applicative information systems, Colloquium on trees in algebra and programming*, Lecture Notes in Computer Science, vol. 159, Springer, pp. 35–64.
- M. COPPO, M. DEZANI-CIANCAGLINI, AND P. SALLÉ
 [1979] *Functional characterization of some semantic equalities inside lambda-calculus., Automata, Languages and Programming*, pp. 133–146.
- M. COPPO, M. DEZANI-CIANCAGLINI, AND B. VENNERI
 [1981] *Functional characters of solvable terms, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, vol. 27, no. 1, pp. 45–58.
- M. COPPO, M. DEZANI-CIANCAGLINI, AND M. ZACCHI
 [1987] *Type theories, normal forms, and D_∞ -lambda-models, Information and Computation*, vol. 72, no. 2, pp. 85–116.
- S. COSMADAKIS
 [1989] *Computing with recursive types (extended abstract), Logic in Computer Science*, IEEE Computer Society Press, pp. 24–38.
- B. COURCELLE
 [1983] *Fundamental properties of infinite trees, Theoretical Computer Science*, vol. 25, pp. 95–169.
- B. COURCELLE, G. KAHN, AND J. VUILLEMIN
 [1974] *Algorithmes d'équivalence et de réduction à des expressions minimales, dans une classe d'équations récursives simples, Automata, Languages and Programming*, Lecture Notes in Computer Science, vol. 14, Springer, pp. 200–213.
- G. COUSINEAU, P.-L. CURIEN, AND M. MAUNY
 [1987] *The categorical abstract machine, Science of Computer Programming*, vol. 8(2), pp. 173–202.
- J. N. CROSSLEY
 [1975] *Reminiscences of logicians, Algebra and logic* (J. N. Crossley, editor), Lecture Notes in Mathematics, vol. 450, Springer, pp. 1–62.
- P.-L. CURIEN
 [1993] *Categorical combinators, sequential algorithms, and functional programming*, second ed., Progress in Theoretical Computer Science, Birkhäuser.
- H. B. CURRY
 [1934] *Functionality in combinatory logic, Proceedings of the National Academy of Science of the USA*, vol. 20, pp. 584–590.
 [1969] *Modified basic functionality in combinatory logic, Dialectica*, vol. 23, pp. 83–92.
- H. B. CURRY AND R. FEYS
 [1958] *Combinatory logic*, vol. I, North-Holland.

R. DAVID AND K. NOUR

[2007] *An arithmetical proof of the strong normalization for the lambda-calculus with recursive equations on types*, *Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science, vol. 4583, Springer, pp. 84–101.

M. DAVIS

[1973] *Hilbert's tenth problem is unsolvable*, *American Mathematical OPmonthly*, vol. 80, pp. 233–269.

M. DAVIS, J. ROBINSON, AND H. PUTNAM

[1961] *The decision problem for exponential Diophantine equations*, *Annals of Mathematics, second series*, vol. 74, no. 3, pp. 425–436.

R. DEDEKIND

[1901] *Essays on the theory of numbers*, Open Court Publishing Company, Translation by W.W. Beman of *Stetigkeit und irrationale Zahlen* (1872) and *Was sind und was sollen die Zahlen?* (1888), reprinted 1963 by Dover Press.

W. DEKKERS

[1988] *Reducibility of types in typed lambda calculus. Comment on: “On the existence of closed terms in the typed λ -calculus, I”* ([Statman \[1980a\]](#)), *Information and Computation*, vol. 77, no. 2, pp. 131–137.

W. DEKKERS, M. BUNDER, AND H. P. BARENDRGHT

[1998] *Completeness of the propositions-as-types interpretation of intuitionistic logic into illative combinatory logic*, *The Journal of Symbolic Logic*, vol. 63, no. 3, pp. 869–890.

M. DEZANI-CIANCAGLINI, S. GHILEZAN, AND S. LIKAVEC

[2004] *Behavioural Inverse Limit lambda-Models*, *Theoretical Computer Science*, vol. 316, no. 1-3, pp. 49–74.

M. DEZANI-CIANCAGLINI, S. GHILEZAN, AND B. VENNERI

[1997] *The “relevance” of intersection and union types*, *Notre Dame Journal of Formal Logic*, vol. 38, no. 2, pp. 246–269.

M. DEZANI-CIANCAGLINI, F. HONSELL, AND F. ALESSI

[2003] *A complete characterization of complete intersection-type preorders*, *ACM Transactions on Computational Logic*, vol. 4, no. 1, pp. 120–147.

M. DEZANI-CIANCAGLINI, F. HONSELL, AND Y. MOTOHAMA

[2005] *Compositional characterization of λ -terms using intersection types*, *Theoretical Computer Science*, vol. 340, no. 3, pp. 459–495.

[2001] *Approximation theorems for intersection type systems*, *Journal of Logic and Computation*, vol. 11, no. 3, pp. 395–417.

M. DEZANI-CIANCAGLINI AND I. MARGARIA

- [1986] *A characterization of F-complete type assignments*, *Theoretical Computer Science*, vol. 45, no. 2, pp. 121–157.
- P. DI GIANANTONIO AND F. HONSELL
 [1993] *An abstract notion of application*, *Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science, vol. 664, Springer, pp. 124–138.
- K. DOŠEN
 [1992] *A brief survey of frames for the Lambek calculus*, *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, vol. 38, pp. 179–187.
- G. DOWEK
 [1994] *Third order matching is decidable*, *Annals of Pure and Applied Logic*, vol. 69, no. 2-3, pp. 135–155.
- J.-P. VAN DRAANEN
 [1995] *Models for simply typed lambda-calculi with fixed point combinators and enumerators*, *Ph.D. thesis*, Catholic University of Nijmegen.
- R. DYCKHOFF AND L. PINTO
 [1999] *Permutability of proofs in intuitionistic sequent calculi*, *Theoretical Computer Science*, vol. 212, pp. 141–155.
- M. C. J. D. VAN EKELEN AND M. J. PLASMEIJER
 [1993] *Functional programming and parallel graph rewriting*, Addison-Wesley.
- H. M. M. TEN EIKELDER
 [1991] *Some algorithms to decide the equivalence of recursive types*, URL: <alexandria.tue.nl/extra1/wskrap/publichtml/9211264.pdf>.
- H. ELBERS
 [1996] *Personal communication*.
- J. ENDRULLIS, C. GRABMAYER, J. W. KLOP, AND V. VAN OOSTROM
 [2010] *Decidability of weak μ -equality*, Manuscript, Free University, Amsterdam.
- J. ENDRULLIS, C. GRABMAYER, J.-W. KLOP, AND V. VAN OOSTROM
 [2011] *On equal μ -terms*, *Theoretical Computer Science*, vol. 412, pp. 3175–3202, Festschrift in Honour of Jan Bergstra.
- E. ENGELER
 [1981] *Algebras and combinators*, *Algebra Universalis*, vol. 13, no. 3, pp. 389–392.
- J. ESPIRITO SANTO
 [2000] *Revisiting the correspondence between cut-elimination and normalisation*, *Automata, Languages and Programming*, Lecture Notes in Computer Science, vol. 1853, Springer, pp. 600–611.
 [2007] *Completing Herbelin’s programme*, *Typed Lambda Calculi and Applications* (S. Ronchi Della Rocca, editor), Lecture Notes in Computer Science, vol. 4583,

Springer, pp. 118–132.

J. ESPIRITO SANTO, S. GHILEZAN, AND J. IVETIĆ
[2008] *Characterising strongly normalising intuitionistic sequent terms*, ***Types for proofs and programs***, Lecture Notes in Computer Science, vol. 4941, Springer, pp. 85–99.

EUCLID OF ALEXANDRIA

[-300] *The Elements*, English translation in Heath [1956].

M. FIORE

[1996] *A coinduction principle for recursive data types based on bisimulation*, ***Information and Computation***, vol. 127, no. 2, pp. 186–198.
[2004] *Isomorphisms of generic recursive polynomial types*, ***SIGPLAN Notices***, vol. 39, no. 1, pp. 77–88.

P. FLAJOLET AND R. SEDGEWICK

[1993] *The average case analysis of algorithms: counting and generating functions*, ***Technical Report 1888***, INRIA.

S. FORTUNE, D. LEIVANT, AND M. O'DONNEL

[1983] *The expressiveness of simple and second-order type structures*, ***Journal of the ACM***, vol. 30, no. 1, pp. 151–185.

A. FOX

[2003] *Formal Specification and Verification of ARM6*, ***Theorem proving in higher order logics 2003*** (D. A. Basin and B. Wolff, editors), Lecture Notes in Computer Science, vol. 2758, Springer.

P. J. FREYD

[1990] *Recursive types reduced to inductive types*, ***Logic in Computer Science***, IEEE Computer Society Press, pp. 498–507.
[1991] *Algebraically complete categories*, ***Como Category Theory Conference***, Lecture Notes in Mathematics, vol. 1488, Springer, pp. 131–156.
[1992] *Remarks on algebraically compact categories*, ***Applications of Categories in Computer Science***, London Mathematical Society Lecture Notes Series, vol. 177, Cambridge University Press, pp. 95–106.

H. M. FRIEDMAN

[1975] *Equality between functionals*, ***Logic Colloquium***, Lecture Notes in Mathematics, vol. 453, Springer.

R. O. GANDY

[1980a] *Church's Thesis and principles for mechanisms*, ***The Kleene Symposium***, North-Holland, pp. 123–148.
[1980b] *An early proof of normalization by A. M. Turing*, in **Hindley and Seldin [1980]** (J. P. Seldin and J. R. Hindley, editors), pp. 453–457.

- V. GAPEYEV, M. Y. LEVIN, AND B. C. PIERCE
[2002] *Recursive subtyping revealed*, *Journal of Functional Programming*, vol. 12, no. 6, pp. 511–548.
- G. GENTZEN
[1936] *Untersuchungen über das logischen Schliessen*, *Mathematische Zeitschrift*, vol. 39, pp. 405–431, Translation in: *Collected papers of Gerhard Gentzen*, ed. M. E. Szabo, North-Holland [1969], 68–131.
[1969] *Investigations into logical deduction*, *The collected papers of Gerhard Gentzen* (M. E. Szabo, editor), North-Holland, pp. 68–131.
[2008] *The normalization of derivations*, *The Bulletin of Symbolic Logic*, vol. 14, pp. 245–257.
- S. GHILEZAN
[1996] *Strong normalization and typability with intersection types*, *Notre Dame Journal of Formal Logic*, vol. 37, no. 1, pp. 44–52.
- G. K. GIERZ, K. H. HOFMANN, K. KEIMEL, J. D. LAWSON, M. W. MISLOVE, AND D. S. SCOTT
[1980] *A compendium of continuous lattices*, Springer.
- J.-Y. GIRARD
[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*, *Scandinavian Logic Symposium*, Studies in Logic and the Foundations of Mathematics, vol. 63, North-Holland, pp. 63–92.
[1995] *Linear logic: its syntax and semantics*, *Advances in linear logic* (J.-Y. Girard, Y. Lafont, and L. Regnier, editors), London Mathematical Society Lecture Note Series, Cambridge University Press.
- J.-Y. GIRARD, Y. G. A. LAFONT, AND P. TAYLOR
[1989] *Proofs and types*, Cambridge Tracts in Theoretical Computer Science, vol. 7, Cambridge University Press.
- K. GÖDEL
[1931] *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I*, *Monatshefte für Mathematik und Physik*, vol. 38, pp. 173–198, German; English translation in Heijenoort [1967], pages 592–618.
[1958] *Ueber eine bisher noch nicht benützte Erweiterung des finiten Standpunktes*, *Dialectica*, vol. 12, pp. 280–287.
- J. GOGUEN, J. W. THATCHER, E. G. WAGNER, AND J. B. WRIGHT
[1977] *Initial algebra semantics and continuous algebras*, *Journal of the ACM*, vol. 24, pp. 68–95.
- W. D. GOLDFARB

- [1981] *The undecidability of the second-order unification problem*, *Theoretical Computer Science*, vol. 13, no. 2, pp. 225–230.
- G. GONTHIER
 [2008] *Formal Proof—The Four-Color Theorem*, *Notices of the American Mathematical Society*, vol. 55, no. 11, pp. 1382–1393.
- A. D. GORDON
 [1994] *Functional programming and Input/Output*, Distinguished Dissertations in Computer Science, Cambridge University Press.
- M. GORDON, R. MILNER, AND C. P. WADSWORTH
 [1979] *Edinburgh LCF. A Mechanical Logic of Computation*, Lecture Notes in Computer Science, vol. 78, Springer.
- M.J.C. GORDON AND T.F. MELHAM
 [1993] *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*, Cambridge University Press.
- C. GRABMAYER
 [2005] *Relating proof systems for recursive types*, *Ph.D. thesis*, Vrije Universiteit Amsterdam.
 [2007] *A duality between proof systems for cyclic term graphs*, *Mathematical Structures in Computer Science*, vol. 17, pp. 439–484.
- NIKHIL, R. S.
 [2008] Bluespec: a general-purpose approach to High-Level Synthesis based on parallel atomic transactions, pp. 129–146, Springer, June, pp. 129–146.
- P. DE GROOTE
 [1995] *The Curry-Howard isomorphism*, Cahiers du Centre de Logique, vol. 8, Academia-Bruylant.
- P. DE GROOTE AND S. POGODALLA
 [2004] *On the expressive power of abstract categorial grammars: Representing context-free formalisms*, *Journal of Logic, Language and Information*, vol. 13, no. 4, pp. 421–438.
- A. GRZEGORCZYK
 [1964] *Recursive objects in all finite types*, *Fundamenta Mathematicae*, vol. 54, pp. 73–93.
- C. A. GUNTER
 [1992] *Semantics of programming languages: Structures and techniques*, MIT Press.
- C. A. GUNTER AND D. S. SCOTT

- [1990] *Semantic domains*, **Handbook of Theoretical Computer Science** (J. Van Leeuwen, editor), vol. B, North-Holland, MIT-Press, pp. 633–674.
- T. C. HALES
[2005] *A proof of the Kepler conjecture*, **Annals of Mathematics**, vol. 162, no. 3, p. 10651185.
- J. HARRISON
[2009a] *HOL Light: An Overview*, **Theorem proving in higher order logics**, Springer, pp. 60–66.
[2009b] *Formalizing an Analytic Proof of the Prime Number Theorem*, **Journal of Automated Reasoning**, vol. 43, no. 3, pp. 243–261.
- R. HARROP
[1958] *On the existence of finite models and decision procedures for propositional calculi*, **Proceedings of the Cambridge Philosophical Society**, vol. 54, pp. 1–13.
- T. L. HEATH
[1956] *The thirteen books of Euclid's Elements*, Dover Publications.
- J. VAN HEIJENOORT
[1967] *From Frege to Gödel: A source book in Mathematical Logic, 1879–1931*, Harvard University Press.
- P. HENDERSON
[1980] *Functional programming: Application and implementation*, Prentice-Hall.
- L. HENKIN
[1950] *Completeness in the theory of types*, **The Journal of Symbolic Logic**, vol. 15, pp. 81–91.
- H. HERBELIN
[1995] *A lambda calculus structure isomorphic to Gentzen-style sequent calculus structure*, **Computer Science Logic**, Lecture Notes in Computer Science, vol. 933, Springer, pp. 61–75.
- D. HILBERT AND W. ACKERMANN
[1928] *Grundzüge der Theoretischen Logik*, first ed., Die Grundlehren der Mathematischen Wissenschaften in Einzeldarstellungen, Band XXVII, Springer.
- J. R. HINDLEY
[1969] *The principal type-scheme of an object in combinatory logic*, **Transactions of the American Mathematical Society**, vol. 146, pp. 29–60.
[1983] *The completeness theorem for typing λ -terms*, **Theoretical Computer Science**, vol. 22, pp. 127–133.
[1992] *Types with intersection: An introduction*, **Formal Aspects of Computing**, vol. 4, no. 5, pp. 470–486.

- [1997] *Basic simple type theory*, Cambridge University Press.
- J. R. HINDLEY AND J. P. SELDIN
 [1980] *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press.
- RALF HINZE, J. JEURING, AND A. LÖH
 [2007] *Comparing approaches to generic programming, Datatype-generic programming 2006* (R. Backhouse, J. Gibbons, R. Hinze, and J. Jeuring, editors), Lecture Notes in Computer Science, vol. 4719, Springer.
- A. HODGES
 [1983] *The enigma of intelligence*, Unwin paperbacks.
- F. HONSELL AND M. LENISA
 [1993] *Some results on the full abstraction problem for restricted lambda calculi, Mathematical Foundations of Computer Science*, Springer, pp. 84–104.
 [1999] *Semantical analysis of perpetual strategies in λ -calculus*, *Theoretical Computer Science*, vol. 212, no. 1-2, pp. 183–209.
- F. HONSELL AND S. RONCHI DELLA ROCCA
 [1992] *An approximation theorem for topological lambda models and the topological incompleteness of lambda calculus*, *Journal of Computer and System Sciences*, vol. 45, no. 1, pp. 49–75.
- W. A. HOWARD
 [1970] *Assignment of ordinals to terms for primitive recursive functionals of finite type, Intuitionism and proof theory* (J. Myhill A. Kino and R. E. Vesley, editors), Studies in Logic and the Foundations of Mathematics, North-Holland, pp. 443–458.
 [1980] *The formulas-as-types notion of construction*, in [Hindley and Seldin \[1980\]](#), pp. 479–490.
- P. HUDAK, J. PETERSON, AND J. H. FASEL
 [1999] *A gentle introduction to Haskell 98*, *Technical report*, Dept. of Computer Science, Yale University, USA.
- P. HUDAK, S. PEYTON JONES, P. WADLER, B. BOUTEL, J. FAIRBAIRN, J. FASEL, M. M. GUZMAN
 [1992] *Report on the programming language Haskell: a non-strict, purely functional language (version 1.2)*, *ACM SIGPLAN Notices*, vol. 27, no. 5, pp. 1–164.
- G. P. HUET
 [1975] *A unification algorithm for typed lambda-calculus*, *Theoretical Computer Science*, vol. 1, pp. 27–57.
- R. J. M. HUGHES
 [1984] *The design and implementation of programming languages*, *Ph.D. thesis*, University of Oxford.

[1989] *Why functional programming matters*, *The Computer Journal*, vol. 32, no. 2, pp. 98–107.

G. HUTTON

[2007] *Programming in Haskell*, Cambridge University Press.

M. HYLAND

[1975/76] *A syntactic characterization of the equality in some models for the lambda calculus*, *Proceedings of the London Mathematical Society (2)*, vol. 12, no. 3, pp. 361–370.

K. E. IVERSON

[1962] *A Programming Language*, Wiley.

G. JACOPINI

[1975] *A condition for identifying two elements of whatever model of combinatory logic, λ -calculus and computer science theory*, Lecture Notes in Computer Science, vol. 37, Springer, pp. 213–219.

B. JAY

[2009] *Pattern Calculus*, Springer.

FELIX JOACHIMSKI AND RALPH MATTHES

[2003] *Short proofs of normalization for the simply-typed lambda-calculus, permutative conversions and go”del’s t*, *Archiv für Mathematische Logik und Grundlagenforschung*, vol. 42, no. 1, pp. 59–87.

T. JOHNSSON

[1984] *Efficient compilation of lazy evaluation*, *SIGPLAN Notices*, vol. 19, no. 6, pp. 58–69.

P. T. JOHNSTONE

[1986] *Stone spaces*, Cambridge University Press.

T. JOLY

[2001a] *Constant time parallel computations in lambda-calculus*, *Theoretical Computer Science*, vol. 266, no. 1, pp. 975–985.

[2001b] *The finitely generated types of the lambda-calculus*, *Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science, vol. 2044, Springer, pp. 240–252.

[2002] *On λ -definability II, the Fixed Type Problem and Finite Generation of Types*, Unpublished. Author’s email: <Thierry.Joly@pps.jussieu.fr>.

[2003] *Encoding of the halting problem into the monster type & applications*, *Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science, vol. 2701, Springer, pp. 153–166.

[2005] *On lambda-definability I: the fixed model problem and generalizations of the matching problem*, *Fundamenta Informaticae*, vol. 65, no. 1-2, pp. 135–151.

- J. P. JONES
[1982] *Universal Diophantine equation*, *The Journal of Symbolic Logic*, vol. 47, no. 3, pp. 549–571.
- M. P. JONES
[1993] *A system of constructor classes: overloading and implicit higher-order polymorphism*, *Functional programming languages and computer architecture*, ACM Press, pp. 52–61.
- J.W. KLOP, V. VAN OOSTROM, AND F. VAN RAAMSDONK
[1993] *Combinatory reduction systems: Introduction and survey*, *Theoretical Computer Science*, vol. 121, pp. 279–308.
- F. KAMAREDDINE, T. LAAN, AND R. NEDERPELT
[2004] *A modern perspective on type theory: From its origins until today*, Applied Logic Series, vol. 29, Kluwer Academic Publishers.
- M. KANAZAWA
[1998] *Learnable classes of categorial grammars*, Cambridge University Press.
- M. KAUFMANN, P. MANOLIOS, AND J. S. MOORE
[2000] *Computer-Aided Reasoning: An Approach*, Kluwer.
- A.J. KFOURY AND J. WELLS
[1995] *New notions of reduction and non-semantic proofs of strong β -normalization in typed λ -calculi*, *Logic in Computer Science*, IEEE Computer Society Press, pp. 311–321.
- S. C. KLEENE
[1936] *Lambda-definability and recursiveness*, *Duke Mathematical Journal*, vol. 2, pp. 340–353.
[1952] *Introduction to metamathematics*, The University Series in Higher Mathematics, van Nostrand.
[1959a] *Countable functionals*, *Constructivity in mathematics* (A. Heyting, editor), Studies in Logic and the Foundations of Mathematics, North-Holland, pp. 81–100.
[1959b] *Recursive functionals and quantifiers of finite types. I*, *Transactions of the American Mathematical Society*, vol. 91, pp. 1–52.
[1975] *Reminiscences of logicians. reported by J. N. Crossley*, *Algebra and logic*, Lecture Notes in Mathematics, vol. 450, Springer, pp. 1–62.
- G. KLEIN, K. ELPHINSTONE, G. HEISER, J. ANDRONICK, D. COCK, P. DERRIN, D. ELKADUWE, K. HAN, AND J. REED
[2009] *seL4: formal verification of an OS kernel*, *ACM Symposium on Principles of Operating Systems* (J.N. Matthews and Th. Anderson, editors), Big Sky, pp. 207–220.
- J. W. KLOP
[1980] *Combinatory reduction systems*, *Ph.D. thesis*, Utrecht University.

[1992] *Term rewriting systems*, ***Handbook of Logic in Computer Science*** (S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors), Oxford University Press, pp. 1–116.

J. W. KLOP AND R. C. DE VRIJER

[1989] *Unique normal forms for lambda calculus with surjective pairing*, ***Information and Computation***, vol. 80, no. 2, pp. 97–113.

J. W. KLOP, V. VAN OOSTROM, AND F. VAN RAAMSDONK

[1993] *Combinatory reduction systems: introduction and survey*, ***Theoretical Computer Science***, vol. 121, pp. 279–308.

P. KOOPMAN AND M. J. PLASMEIJER

[1999] *Efficient combinator parsers*, ***Implementation of functional languages*** (K. Hammond, A. J. T. Davie, and C. Clack, editors), Lecture Notes in Computer Science, vol. 1595, Springer, pp. 120–136.

[2006] *Fully automatic testing with functions as specifications*, ***Selected lectures of the 1st central european functional programming school***, Lecture Notes in Computer Science, vol. 4164, Springer, pp. 35–61.

C. H. A. KOSTER

[1969] *On infinite modes*, ***ALGOL Bulletin***, vol. 30, pp. 86–89.

C. P. J. KOYMANS

[1982] *Models of the lambda calculus*, ***Information and Control***, vol. 52, no. 3, pp. 306–323.

D. KOZEN

[1977] *Complexity of finitely presented algebras*, ***ACM Symposium on Theory of Computing***, ACM Press, pp. 164–177.

[1997] *Automata and computability*, Springer.

D. KOZEN, J. PALSBERG, AND M. I. SCHWARTZBACH

[1995] *Efficient recursive subtyping*, ***Mathematical Structures in Computer Science***, vol. 5, no. 1, pp. 113–125.

G. KREISEL

[1959] *Interpretation of analysis by means of constructive functionals of finite types*, ***Constructivity in mathematics*** (A. Heyting, editor), North-Holland, pp. 101–128.

S. A. Kripke

[1965] *Semantical analysis of intuitionistic logic I*, ***Formal systems and recursive functions*** (M. Dummett and J. N. Crossley, editors), North-Holland, pp. 92–130.

J.-L. KRIVINE

[1990] *Lambda-calcul types et modèles*, Masson, English translation [Krivine \[1993\]](#).

- [1993] *Lambda-calculus, types and models*, Ellis Horwood, Translated from the 1990 French original by René Cori.
- T. KURATA AND M. TAKAHASHI
 [1995] *Decidable properties of intersection type systems, Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science, vol. 902, Springer, pp. 297–311.
- N. KURTONINA
 [1995] *Frames and labels. A modal analysis of categorial inference*, *Ph.D. thesis*, OTS Utrecht, ILLC Amsterdam.
- N. KURTONINA AND M. MOORTGAT
 [1997] *Structural control, Specifying syntactic structures* (Patrick Blackburn and Maarten de Rijke, editors), Center for the Study of Language and Information, Stanford, pp. 75–113.
- D. KUŚMIEREK
 [2007] *The inhabitation problem for rank two intersection types*, *Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science, vol. 4583, pp. 240–254.
- J. LAMBEK
 [1958] *The mathematics of sentence structure*, *American Mathematical Monthly*, vol. 65, pp. 154–170, Also in Buszkowski, Marciszewski, and Benthem [1988].
 [1961] *On the calculus of syntactic types*, *Structure of language and its mathematical aspects* (R. Jacobson, editor), Proceedings of the Symposia in Applied Mathematics, vol. XII, American Mathematical Society, pp. 166–178.
 [1980] *From λ -calculus to Cartesian closed categories*, In: [Hindley and Seldin \[1980\]](#) (J. R. Hindley and J. P. Seldin, editors), Academic Press, pp. 375–402.
- J. LAMBEK AND P. J. SCOTT
 [1981] *Intuitionist type theory and foundations*, *Journal of Philosophical Logic*, vol. 10, pp. 101–115.
- E. LANDAU
 [1960] *Grundlagen der analysis*, 3-rd edition ed., Chelsea Publishing Company.
- P. J. LANDIN
 [1964] *The mechanical evaluation of expressions*, *The Computer Journal*, vol. 6, no. 4, pp. 308–320.
- P. D. LAX
 [2002] *Functional analysis*, Pure and Applied Mathematics, Wiley.
- D. J. LEHMANN AND M. B. SMYTH
 [1981] *Algebraic specification of data types: A synthetic approach*, *Mathematical Systems Theory*, vol. 14, pp. 97–139.
- D. LEIVANT

- [1983a] *Polymorphic type inference*, *Symposium on principles of programming languages*, ACM Press, pp. 88–98.
- [1983b] *Reasoning about functional programs and complexity classes associated with type disciplines*, *Symposium on foundations of computer science*, IEEE Computer Society Press, pp. 460–469.
- [1986] *Typing and computational properties of lambda expressions*, *Theoretical Computer Science*, vol. 44, no. 1, pp. 51–68.
- [1990] *Discrete Polymorphism*, *ACM conference on LISP and functional programming*, ACM Press, pp. 288–297.

X. LEROY

- [2009] *A formally verified compiler back-end*, *Journal of Automated Reasoning*, vol. 43, no. 4, pp. 363–446.

J.-J. LÉVY

- [1978] *Réductions correctes et optimales dans le lambda-calcul*, *Ph.D. thesis*, Université Paris VII.

L. LIQUORI AND S. RONCHI DELLA ROCCA

- [2007] *Intersection types à la Church*, *Information and Computation*, vol. 205, no. 9, pp. 1371–1386.

R. LOADER

- [1997] *An algorithm for the minimal model*, Unpublished manuscript, obtainable from <homepages.ihug.co.nz/~suckfish/papers/papers.html>..
- [2001a] *Finitary PCF is not decidable*, *Theoretical Computer Science*, vol. 266, no. 1-2, pp. 341–364.
- [2001b] *The undecidability of lambda definability*, In: *Zeleny and Anderson [2001]*, pp. 331–342.
- [2003] *Higher order β matching is undecidable*, *Logic Journal of the Interest Group in Pure and Applied Logics*, vol. 11, no. 1, pp. 51–68.

G. LONGO

- [1988] *The lambda-calculus: connections to higher type recursion theory, proof-theory, category theory*, *Annals of Pure and Applied Logic*, vol. 40, pp. 93–133.

E. G. K. LOPEZ-ESCOBAR

- [1983] *Proof functional connectives*, *Methods in mathematical logic*, Lecture Notes in Mathematics, vol. 1130, pp. 208–221.

D. MACQUEEN, G. D. PLOTKIN, AND R. SETHI

- [1986] *An ideal model for recursive polymorphic types*, *Information and Control*, vol. 71, no. (1/2), pp. 95–130.

H. G. MAIRSON

- [1992] *A simple proof of a theorem of Statman*, *Theoretical Computer Science*, vol. 103, no. 2, pp. 387–394.

- G. S. MAKANIN
 [1977] *The problem of solvability of equations in a free semigroup*, ***Mathematics of the USSR-Sbornik***, vol. 32, no. 2, pp. 129–198.
- P. MARTIN-LÖF
 [1984] *Intuitionistic type theory*, Studies in Proof Theory, Bibliopolis.
- M. MARZ
 [1999] *An algebraic view on recursive types*, ***Applied Categorical Structures***, vol. 7, no. 1-2, pp. 147–157.
- Y. V. MATIYASEVIČ
 [1972] *Diophantine representation of recursively enumerable predicates*, ***Mathematical Notes***, vol. 12, no. 1, pp. 501–504.
 [1993] *Hilbert's tenth problem*, Foundations of Computing Series, MIT Press.
- R. MAYR AND T. NIPKOW
 [1998] *Higher-order rewrite systems and their confluence*, ***Theoretical Computer Science***, vol. 192, no. 1, pp. 3–29.
- S. PEYTON JONES
 [2003] *Haskell 98 language and libraries: the revised report*, Cambridge University Press.
- J. MCCARTHY
 [1963] *A basis for a mathematical theory of computation*, ***Computer Programming and Formal Systems*** (P. Braffort and D. Hirschberg, editors), North-Holland, pp. 33–70.
- J. MCCARTHY, P.W. ABRAHAMS, D. J. EDWARDS, T. P. HART, AND M. I. LEVIN
 [1962] *Lisp 1.5 programmer's manual*, MIT Press.
- N. J. McCracken
 [1979] *An investigation of a programming language with a polymorphic type structure*, ***Ph.D. thesis***, Syracuse University.
- P.-A. MELLIES
 [1996] *Description Abstraite des Systèmes de Réécriture*, ***Ph.D. thesis***, Université de Paris.
- N. P. MENDLER
 [1987] *Inductive definitions in type theory*, ***Ph.D. thesis***, Department of Computer Science, Cornell University.
 [1991] *Inductive types and type constraints in the second-order lambda calculus*, ***Annals of Pure and Applied Logic***, vol. 51, pp. 159–172.
- A. R. MEYER

[1982] *What is a model of the lambda calculus?*, *Information and Control*, vol. 52, no. 1, pp. 87–122.

R. MILNER

[1978] *A theory of type polymorphism in programming*, *Journal of Computer and System Sciences*, vol. 17, pp. 348–375.

R. MILNER, M. TOFTE, R. HARPER, AND D. MCQUEEN

[1997] *The Definition of Standard ML*, The MIT Press.

G. E. MINTS

[1989] *The completeness of provable realizability*, *Notre Dame Journal Formal Logic*, vol. 30, pp. 420–441.

[1996] *Normal forms for sequent derivations*, *Kreiseliana. About and Around Georg Kreisel* (P. Odifreddi, editor), A.K. Peters, pp. 469–492.

J. MITCHELL

[1996] *Foundation for Programming Languages*, MIT Press.

G. MITSCHKE

[1976] λ -Kalkül, δ -Konversion und axiomatische Rekursionstheorie, *Technical Report Preprint 274*, Technische Hochschule, Darmstadt.

T. Æ. MOGENSEN

[1992] *Theoretical pearls: Efficient self-interpretation in lambda calculus*, *Journal of Functional Programming*, vol. 2, no. 3, pp. 345–364.

F. MOLLER AND S. A. SMOLKA

[1995] *On the computational complexity of bisimulation*, *ACM Computing Surveys*, vol. 27, no. 2, pp. 287–289.

R. MONTAGUE

[1973] *The proper treatment of quantification in ordinary English*, *Approaches to natural language* (J. Hintikka, J. M. E. Moravcsik, and P. Suppes, editors), Dordrecht.

R. MOOT

[2002] *Proof nets for linguistic analysis*, *Ph.D. thesis*, Utrecht University.

G. MORRILL

[1994] *Type logical grammar*, Kluwer.

J. H. MORRIS

[1968] *Lambda-calculus models of programming languages*, *Ph.D. thesis*, Massachusetts Institute of Technology.

M. MUZALEWSKI

[1993] *An Outline of PC Mizar*, Fondation Philippe le Hodey, Brussels.

G. NADATHUR AND D. MILLER

- [1988] *An overview of λ Prolog*, *Logic programming conference*, MIT Press, pp. 810–827.
- R. P. NEDERPELT
 [1973] *Strong normalisation in a typed lambda calculus with lambda structured types*, *Ph.D. thesis*, Eindhoven University.
- R. P. NEDERPELT, J. H. GEUVERS, AND R. C. DE VRIJER
 [1994] *Selected Papers on Automath*, Studies in Logic and the Foundations of Mathematics, no. 133, North-Holland.
- A. NERODE, P. ODIFREDDI, AND R. PLATEK
 [In preparation] *The four noble truths of logic*, To appear.
- T. NIPKOW, L. C. PAULSON, AND M. WENZEL
 [2002a] *Isabelle/HOL*, Lecture Notes in Computer Science, vol. 2283, Springer-Verlag, Berlin, A proof assistant for higher-order logic.
 [2002b] *Isabelle/HOL, a proof assistant for higher-order logic*, Lecture Notes in Computer Science, vol. 2283, Springer.
- M. OOSTDIJK
 [1996] *Proof by Calculation*, *Master's thesis*, Nijmegen University.
- V. VAN OOSTROM
 [2007] α -free- μ , Unpublished manuscript, Vincent.vanOostrom@phil.uu.nl.
- V. PADOVANI
 [1996] *Filtrage d'ordre supérieur*, *Ph.D. thesis*, Université de Paris VII.
 [2000] *Decidability of fourth order matching*, *Mathematical Structures in Computer Science*, vol. 3, no. 10, pp. 361 – 372.
- C. PAIR
 [1970] *Concerning the syntax of ALGOL 68*, *ALGOL Bulletin*, vol. 31, pp. 16–27.
- R. PARikh
 [1973] *On the length of proofs*, *Transactions of the American Mathematical Society*, vol. 177, pp. 29–36.
- D. PARK
 [1976] *The Y combinator in Scott's lambda calculus models*, *Theory of Computation Report 13*, University of Warwick, Department of Computer Science.
- M. PENTUS
 [1993] *Lambek grammars are context free*, *Logic in Computer Science*, IEEE Computer Society Press, pp. 429–433.
 [2006] *Lambek calculus is NP-complete*, *Theoretical Computer Science*, vol. 357, no. 1-3, pp. 186–201.
- R. PÉTER

- [1967] *Recursive functions*, third revised ed., Academic Press.
- S. PEYTON-JONES
 [1987] *The Implementation of Functional Programming Languages*, Prentice Hall.
- S. PEYTON JONES, D. VYTINIOTIS, S. WEIRICH, AND G. WASHBURN
 [2006] *Simple unification-based type inference for GADTs*, *International conference on functional programming*, pp. 50–61.
- S. PEYTON JONES [EDITOR], J. HUGHES [EDITOR], L. AUGUSTSSON, D. BARTON, B. BOUTEL, W. BURTON
 [1999] *Haskell 98 — A non-strict, purely functional language*, URL <www.haskell.org/definition/>.
- S. L. PEYTON JONES AND P. WADLER
 [1993] *Imperative functional programming*, *Principles of Programming Languages*, ACM Press, pp. 71–84.
- B. C. PIERCE
 [2002] *Types and programming languages*, MIT Press.
- M. R. C. PIL
 [1999] *Dynamic types and type dependent functions*, *Implementation of functional languages* (H. Hammond, T. Davie, and C. Clack, editors), Lecture Notes in Computer Science, vol. 1595, Springer, pp. 169–185.
- E. PIMENTEL, S. RONCHI DELLA ROCCA, AND L. ROVERSI
 [In preparation] *Intersection types from a proof-theoretic perspective*.
- M. J. PLASMEIJER AND M. VAN EKELEN
 [2002] *Concurrent Clean language report (version 2.1)*, <www.cs.ru.nl/~clean>.
- R. PLASMEIJER, P. ACHTEM, AND P. KOOPMAN
 [2007] *iTasks: Executable Specifications of Interactive Work Flow Systems for the Web*, *International Conference on Functional Programming*, ACM Press, pp. 141–152.
- R. A. PLATEK
 [1966] *Foundations of recursions theory*, *Ph.D. thesis*, Stanford University.
- J. VON PLATO
 [2001a] *A proof of Gentzen's Hauptsatz without multicut*, *Archiv für Mathematische Logik und Grundlagenforschung*, vol. 40, no. 1, pp. 9–18.
 [2001b] *Natural deduction with general elimination rules*, *Archiv für Mathematische Logik und Grundlagenforschung*, vol. 40, no. 7, pp. 541–567.
 [2008] *Gentzen's proof of normalization for natural deduction*, *The Bulletin of Symbolic Logic*, vol. 14, no. 2, pp. 240–257.

- G. D. PLOTKIN
- [1975] *Call-by-name, call-by-value and the λ -calculus*, *Theoretical Computer Science*, vol. 1, no. 2, pp. 125–159.
 - [1977] *LCF considered as a programming language*, *Theoretical Computer Science*, vol. 5, pp. 225–255.
 - [1980] *Lambda-definability in the full type hierarchy*, in **Hindley and Seldin [1980]** (J. R. Hindley and J. P. Seldin, editors), Academic Press, pp. 363–373.
 - [1982] *The category of complete partial orders: a tool for making meanings*, Postgraduate lecture notes, Edinburgh University.
 - [1985] *Lectures on predomains and partial functions*, Center for the Study of Language and Information, Stanford.
 - [1993] *Set-theoretical and other elementary models of the λ -calculus*, *Theoretical Computer Science*, vol. 121, no. 1-2, pp. 351–409.
- H. POINCARÉ
- [1902] *La science et l'hypothèse*, Flammarion.
- A. POLONSKY
- [2010] Personal communication.
- E. POST
- [1947] *Recursive unsolvability of a problem of Thue*, *The Journal of Symbolic Logic*, vol. 12, no. 1, pp. 1–11.
- G. POTTINGER
- [1977] *Normalization as a homomorphic image of cut-elimination*, *Annals of Mathematical Logic*, vol. 12, pp. 323–357.
 - [1980] *A type assignment for the strongly normalizable λ -terms*, in **Hindley and Seldin [1980]**, pp. 561–77.
 - [1981] *The Church-Rosser theorem for the typed λ -calculus with surjective pairing*, *Notre Dame Journal Formal Logic*, vol. 22, no. 3, pp. 264–268.
- D. PRAWITZ
- [1965] *Natural deduction*, Almqvist & Wiksell.
 - [1971] *Ideas and results in proof theory*, *Scandinavian logic symposium* (J. E. Fenstad, editor), North-Holland, pp. 235–307.
- F. VAN RAAMSDONK
- [1996] *Confluence and normalisation for higher-order rewriting*, *Ph.D. thesis*, Vrije Universiteit.
- F. VAN RAAMSDONK, P. SEVERI, M.H. SØRENSEN, AND H. XI
- [1999] *Perpetual reductions in lambda calculus*, *Information and Computation*, vol. 149, no. 2, pp. 173–225.
- J. C. REYNOLDS

- [1972] *Definitional interpreters for higher-order programming languages*, **ACM national conference**, ACM Press, pp. 717–740.
- [1993] *The discoveries of continuations*, **LISP and Symbolic Computation**, vol. 6, no. 3/4, pp. 233–247.
- N. ROBERTSON, D. SANDERS, P. SEYMOUR, AND R. THOMAS
[1997] *The four-colour theorem*, **Journal of Combinatorial Theory. Series B**, vol. 70, no. 1, pp. 2–44.
- J. A. ROBINSON
[1965] *A machine-oriented logic based on the resolution principle*, **Journal of the ACM**, vol. 12, no. 1, pp. 23–41.
- H. ROGERS JR.
[1967] *Theory of recursive functions and effective computability*, McGraw-Hill.
- S. RONCHI DELLA ROCCA
[1988] *Lecture notes on semantics and types*, Technical report, Torino University.
[2002] *Intersection typed lambda-calculus*, **Intersection Types and Related Systems**, vol. 70, Electronic Notes in Theoretical Computer Science, no. 1, Elsevier, pp. 163–181.
- S. RONCHI DELLA ROCCA AND L. PAOLINI
[2004] *The parametric lambda calculus a metamodel for computation*, Texts in Theoretical Computer Science, vol. XIII, Springer.
- J. B. ROSSER
[1984] *Highlights of the history of the lambda-calculus*, **Annals of the History of Computing**, vol. 6, no. 4, pp. 337–349.
- J. J. M. M. RUTTEN
[2000] *Universal coalgebra: a theory of systems*, **Theoretical Computer Science**, vol. 249, no. 1, pp. 3–80.
[2005] *A coinductive calculus of streams*, **Mathematical Structures in Computer Science**, vol. 15, no. 1, pp. 93–147.
- S. SALVATI
[2009] *Recognizability in the Simply Typed Lambda-Calculus*, **Workshop on logic, language, information and computation**, Lecture Notes in Computer Science, vol. 5514, Springer, pp. 48–60.
- M. SCHMIDT-SCHAUSS
[1999] *Decidability of behavioural equivalence in unary PCF*, **Theoretical Computer Science**, vol. 216, no. 1-2, pp. 363–373.
- A. SCHUBERT

- [1998] Second-order unification and type inference for Church-style polymorphism, *Symposium on principles of programming languages*, pp. 279–288.
- H. SCHWICHTENBERG
- [1975] Elimination of higher type levels in definitions of primitive recursive functionals by means of transfinite recursion, *Logic colloquium*, Studies in Logic and the Foundations of Mathematics, vol. 80, North-Holland, pp. 279–303.
 - [1976] Definierbare Funktionen im λ -Kalkül mit Typen, *Archief für Mathematische Logik*, vol. 25, pp. 113–114.
 - [1999] Termination of permutative conversion in intuitionistic Gentzen calculi, *Theoretical Computer Science*, vol. 212, no. 1-2, pp. 247–260.
- H. SCHWICHTENBERG AND U. BERGER
- [1991] An inverse of the evaluation functional for typed λ -calculus, *Logic in Computer Science*, IEEE Computer Society Press, pp. 203–211.
- D. S. SCOTT
- [1970] Constructive validity, *Symposium on automated demonstration*, Lecture Notes in Mathematics, vol. 125, Springer, pp. 237–275.
 - [1972] Continuous lattices, *Toposes, algebraic geometry and logic*, Lecture Notes in Mathematics, vol. 274, Springer, pp. 97–136.
 - [1975a] Open problem 4, in **Böhm [1975]**, p. 369.
 - [1975b] Some philosophical issues concerning theories of combinators, in **Böhm [1975]**, pp. 346–366.
 - [1976] Data types as lattices, *SIAM Journal on Computing*, vol. 5, pp. 522–587.
 - [1980] Relating theories of the λ -calculus, in **Hindley and Seldin [1980]**, pp. 403–450.
- M. SHEERAN
- [2005] Hardware design and functional programming: a perfect match, *Journal of Universal Computer Systems*, vol. 11, no. 7, pp. 1135–1158.
- M. B. SMYTH AND G. D. PLOTKIN
- [1982] The category-theoretic solution of recursive domain equations, *SIAM Journal on Computing*, vol. 11, no. 4, pp. 761–783.
- M. H. SØRENSEN
- [1997] Strong normalization from weak normalization in typed λ -calculi, *Information and Computation*, vol. 133, pp. 35–71.
- M. H. SØRENSEN AND P. URZYCZYN
- [2006] *Lectures on the Curry-Howard Isomorphism*, Elsevier.
- C. SPECTOR
- [1962] Provable recursive functionals of analysis, *Recursive function theory* (J. C. E. Dekker, editor), American Mathematical Society, pp. 1–27.
- R. STATMAN

- [1979] *The typed λ -calculus is not elementary recursive*, *Theoretical Computer Science*, vol. 9, no. 1, pp. 73–81.
- [1979a] *Intuitionistic propositional logic is polynomial-space complete*, *Theoretical Computer Science*, vol. 9, no. 1, pp. 67–72.
- [1980a] *On the existence of closed terms in the typed λ -calculus. I*, *To H. B. Curry: Essays on combinatory logic, lambda calculus and formalism* (J. R. Hindley and J. P. Seldin, editors), Academic Press, pp. 511–534.
- [1980b] *On the existence of closed terms in the typed λ -calculus. III*, Carnegie Mellon University.
- [1981] *On the existence of closed terms in the typed λ -calculus II: Transformations of unification problems*, *Theoretical Computer Science*, vol. 15, no. 3, pp. 329–338.
- [1982] *Completeness, invariance and λ -definability*, *The Journal of Symbolic Logic*, vol. 47, no. 1, pp. 17–26.
- [1985] *Equality between functionals revisited*, *Harvey Friedman's research on the foundations of mathematics* (L. A. Harrington, M. D. Morley, A. Ščedrov, and S. G. Simpson, editors), Studies in Logic and the Foundations of Mathematics, vol. 117, North-Holland, pp. 331–338.
- [1994] *Recursive types and the subject reduction theorem*, *Technical Report 94-164*, Carnegie Mellon University.
- [2000] *Church's lambda delta calculus*, *Logic for programming and automated reasoning*, Lecture Notes in Computer Science, vol. 1955, Springer, pp. 293–307.
- [2007] *On the complexity of α -conversion*, *The Journal of Symbolic Logic*, vol. 72, no. 4, pp. 1197–1203.

M. STEEDMAN

- [2000] *The syntactic process*, MIT Press.

G. L. STEELE JR.

- [1978] *Rabbit: A compiler for Scheme*, *Technical Report AI-TR-474*, Artificial Intelligence Laboratory, MIT.

- [1984] *Common Lisp: The language*, Digital Press.

S. STENLUND

- [1972] *Combinators, λ -terms and proof theory*, Synthese Library, D. Reidel.

C. STIRLING

- [2009] *Decidability of higher-order matching*, *Logical Methods in Computer Science*, vol. 5, no. 3, pp. 1–52.

K. STØVRING

- [2006] *Extending the extensional lambda calculus with surjective pairing is conservative*, *Logical Methods in Computer Science*, vol. 2, no. 2:1, pp. 1–14.

G. SUDAN

- [1927] *Sur le nombre transfini ω^ω* , *Bulletin mathématique de la Société Roumaine des Sciences*, vol. 30, pp. 11–30.

W. W. TAIT

- [1965] *Infinitely long terms of transfinite type I*, *Formal systems and recursive functions* (J. Crossley and M. Dummett, editors), North-Holland, pp. 176–185.
- [1967] *Intensional interpretations of functionals of finite type. I*, *The Journal of Symbolic Logic*, vol. 32, pp. 198–212.
- [1971] *Normal form theorem for barrecursive functions of finite type*, *Scandinavian logic symposium*, North-Holland, pp. 353–367.

M. TATSUTA AND M. DEZANI-CIANCAGLINI

- [2006] *Normalisation is Insensible to Lambda-term Identity or Difference*, *Logic in Computer Science*, IEEE Computer Society Press, pp. 327–336.

TERESE

- [2003] *Term rewriting systems*, Cambridge University Press.

J. TERLOUW

- [1982] *On definition trees of ordinal recursive functionals: reduction of the recursion orders by means of type level raising*, *The Journal of Symbolic Logic*, vol. 47, no. 2, pp. 395–402.

J. W. THATCHER

- [1973] *Tree automata: an informal survey*, *Currents in the Theory of Computing* (A. V. Aho, editor), Prentice-Hall, pp. 143–172.

S. THOMPSON

- [1995] *Miranda, The Craft of Functional Programming*, Addison-Wesley.

H. J. TIEDE

- [2001] *Lambek calculus proofs and tree automata*, *Logical Aspects of Computational Linguistics*, Lecture Notes in Artificial Intelligence, vol. 2014, Springer, pp. 251–265.

- [2002] *Proof Tree Automata, Words, proofs, and diagrams* (D. Barker-Plummer, D. Beaver, J. F. A. K. van Benthem, and P. Scotto di Luzio, editors), Center for the Study of Language and Information, Stanford, pp. 143–162.

A. S. TROELSTRA

- [1973] *Metamathematical investigation of intuitionistic arithmetic and analysis*, Lecture Notes in Mathematics, no. 344, Springer.

- [1999] *Marginalia on sequent calculi*, *Studia Logica*, vol. 62, no. 2, pp. 291–303.

A. S. TROELSTRA AND H. SCHWICHTENBERG

- [1996] *Basic proof theory*, Cambridge University Press.

D. A. TURNER

- [1976] *The SASL language manual*, <http://www.eis.mdx.ac.uk/staffpages/dat/saslname.pdf>.

- [1979] *A new implementation technique for applicative languages*, *Software—Practice and Experience*, vol. 9, pp. 31–49.

- [1981] *The semantic elegance of functional languages*, ***Functional programming languages and computer architecture***, ACM Press, pp. 85–92.
- [1985] *Miranda, a non-strict functional language with polymorphic types*, ***Functional programming languages and computer architectures***, Lecture Notes in Computer Science, vol. 201, Springer, pp. 1–16.
- C. URBAN, S. BERGHOFER, AND M. NORRISH
- [2007] *Barendregt's Variable Convention in Rule Inductions*, ***Proc. of the 21th international conference on automated deduction (cade)***, LNAI, vol. 4603, pp. 35–50.
- C. URBAN AND C. TASSON
- [2005] *Nominal techniques in Isabelle/HOL*, ***Automated deduction—CADE-20***, Lecture Notes in Computer Science, vol. 3632, Springer, pp. 38–53.
- P. URZYCZYN
- [1999] *The emptiness problem for intersection types*, ***The Journal of Symbolic Logic***, vol. 64, no. 3, pp. 1195–1215.
- [2009] *Inhabitation of low-rank intersection types*, ***Typed Lambda Calculi and Applications***, Lecture Notes in Computer Science, vol. 5608, Springer, pp. 356–370.
- B. VENNERI
- [1994] *Intersection types as logical formulae*, ***Journal of Logic and Computation***, vol. 4, no. 2, pp. 109–124.
- [1996] *Private communication*, Florence University.
- W. VERMAAT
- [2006] *The logic of variation. A cross-linguistic account of wh-question formation*, ***Ph.D. thesis***, Utrecht University.
- H. VOGEL
- [1976] *Ein starker Normalisationssatz für die barrekursiven Funktionale*, ***Archiv für Mathematische Logik und Grundlagenforschung***, vol. 18, pp. 81–84.
- J. VOIGTLÄNDER
- [2009] *Free theorems involving type constructor classes: functional pearl*, ***ACM SIGPLAN international conference on Functional programming***, ACM Press, pp. 173–184.
- J. VOUILLON AND P.-A. MELLIES
- [2004] *Semantic types: A fresh look at the ideal model for types*, ***Principles of Programming Languages***, ACM Press, pp. 24–38.
- R. C. DE VRIJER
- [1989] *Extending the lambda calculus with surjective pairing is conservative*, ***Logic in computer science***, pp. 204–215.
- [1987] *Surjective pairing and strong normalization: two themes in lambda calculus*, ***Ph.D. thesis***, University of Amsterdam.

- C. P. WADSWORTH
 [1971] *Semantics and pragmatics of the lambda-calculus*, **Ph.D. thesis**, University of Oxford.
 [1976] *The relation between computational and denotational properties for Scott's D_∞ -models of the lambda-calculus*, **SIAM Journal of Computing**, vol. 5, no. 3, pp. 488–521.
- M. WAND
 [1987] *A simple algorithm and proof for type inference*, **Fundamenta Informaticae**, vol. 10, pp. 115–122.
- H. WANSING
 [2002] *Sequent systems for modal logics*, **Handbook of philosophical logic** (D. Gabbay and F. Guenther, editors), vol. 8, Kluwer, pp. 61–145.
- J. B. WELLS
 [1999] *Typability and type checking in system F are equivalent and undecidable*, **Annals of Pure and Applied Logic**, vol. 98, no. 1-3, pp. 111–156.
- J. B. WELLS, A. DIMOCK, R. MULLER, AND F. TURBAK
 [1997] *A typed intermediate language for flow-directed compilation*, **Theory and practice of software development**, Lecture Notes in Computer Science, no. 1214, pp. 757–771.
- A.N. WHITEHEAD AND B. RUSSELL
 [1910-1913] *Principia mathematica*, Cambridge University Press.
- F. WIEDIJK
 [2006] *The seventeen provers of the world*, Lecture Notes in Computer Science, vol. 3600, Springer.
- A. VAN WIJNGAARDEN
 [1981] *Revised report of the algorithmic language algol 68*, **ALGOL Bulletin**, no. Sup 47, pp. 1–119.
- H. XI
 [1997] *Weak and strong beta normalisations in typed λ -calculi*, **Typed Lambda Calculi and Applications**, Lecture Notes in Computer Science, vol. 1210, Springer, pp. 390–404.
- M. ZELENY AND C.A. ANDERSON
 [2001] *Church memorial volume: Logic, Language, and Computation*, Kluwer.
- J. ZUCKER
 [1974] *Cut-elimination and normalization*, **Annals of Mathematical Logic**, vol. 7, pp. 1–112.

C. ZYLBERAJCH

[1991] *Syntaxe et semantique de la facilité en lambda-calcul.*, **Ph.D. thesis**, Université de Paris VII.

INDICES

There is an index of definitions, of names (authors of cited references), and of symbols. The index of symbols is subdivided as follows.

- Terms: general, operations, classes, relations, theories.
- Types/Propositions: general, operations, classes, relations, theories, classes of theories.
- Assignment: theories, derivation.
- Models: general, operations on, classes, relations, interpretations in.
- Miscellaneous: general, operations, classes, relations.
- Categories: general, functors.

The meaning of the main division is as follows. There are objects and relations between these. Using the operations one constructs new objects. By collecting objects one obtains classes, by collecting valid relations one obtains theories. Categories consist of classes of objects together with operations.

Index of definitions

- α -avoiding, 317
- α -flexible, 317
- $\dot{\mu}$ -redex, 315
- λ -definable, 32
 - slantwise, 148
- μ -redex, 323
- ∇ -uniform, 408
- safe
 - $\dot{\mu}$ -type, 317
- abstract reduction system, 45
- Ackermann function, 217
- added, 311
- additive, 513
- adfluent, 231
- adjunction, 309
- admissible rule, 455
- agree
 - with ρ , 541
 - with constraints, 375
- algebra, 35, 338
- algebraic, 35, 35
 - ω -lattice, 405
 - lattice, 405
 - map, 337
- ambiguous, 218
- application operators, 76
- applicative structure, 84
- approximable maps, 509
- approximant
 - \mathcal{T} -, 584
 - direct, 418
 - of M , 418
 - of a λ -term, 418, 456
- approximate
 - interpretation, 415
- approximation, 407
 - of a regular tree, 415
 - of a type, 411
 - theorem, 419
- argument, 578
- arity, 12
- assertion
- \mathcal{S} -, 478
- \mathcal{T} -, 473
- association
 - to the left, 5
 - to the right, 5
- atom
 - type -, 291, 463
- atomic
 - type inequality, 433
- bar condition, 229
- bar recursion, 230
- basis, 8, 454
 - X -, 614
 - \mathcal{S} -, 478
 - \mathcal{T} -, 473
 - á la de Bruijn*, 15
- binary trees, 34
- binds, 319
- bisimulation, 328, 346
- Boolean
 - interpretation, 62
 - valuation, 62
- bottom, 432, 583
- bound variable
 - of a λ -term, 5
 - of a μ -type, 314
- canonical form, 488
- captures, 319
- carrier, 338
- Cartesian
 - monoid, 197
 - free -, 200
 - product, 81
- category
 - of algebraic lattices, 405
 - of complete lattices, 405
- characteristic type, 126
- Church
 - numeral, 32
 - version of λ_{\equiv}^A , 298
 - version of $\lambda_{\rightarrow}^{\mathbb{A}}$, 13

- Church-Rosser, 309
 Church-Rosser property, 46
 weak -, 46
 Church-Rosser theorem
 for \Rightarrow_μ , 309
 for $\lambda_{\rightarrow}^{\text{Ch}}$, 23
 for $\lambda_{\rightarrow}^{\text{dB}}$, 19
 for $\lambda_{\rightarrow}^{\text{Cu}}$, 18
 for $\lambda\beta\eta$, 6
 circular, 324
 class
 of a set of constants, 126
 of a type, 119
 classically valid, 62
 closed, 313
 term, 5
 term model, 242
 under application, 77
 under head expansion, 578
 under reduction, 395
 closure
 under reduction, 18
 closure ordinal, 221
 coalgebra
 T -, 339
 morphism, 339
 code, 214
 computationally adequate, 563
 combinator, 5
 Combinatory Reduction System, 324
 compact
 element, 405
 compatible, 466
 complete, 121
 A -, 139
 for pure terms, 139
 lattice, 404
 partial order, 404
 subset, 408
 completeness, 459, 575
 components, 12, 41, 600
 computable, 52
 under substitution, 52
 Computer Mathematics, 261
 configuration, 607
 confluent, 6, 46
 from M , 92
 congruence, 293, 304
 consistent, 7, 86
 constant, 86, 178
 constraints, 375
 constructor, 35
 continuous, 238, 405
 contraction
 of $\dot{\mu}$ -redex, 315
 of μ -redex, 323
 contractive
 map, 337
 type - in a variable, 324
 conversion
 α - - on types, 315
 $\dot{\mu}$ - - on types, 315
 μ - - on types, 324
 Coq, 15
 corecursion, 346
 CPO: complete partial order, 404
 Curry
 version of $\lambda_{\underline{\rightarrow}}^A$, 292
 version of λ_{\rightarrow}^A , 13
 Curry-Howard isomorphism, 271
 cyclic, 375
 de Bruijn
 version of λ_{\rightarrow}^A , 15
 decidable, 152
 decision problem, 152
 declaration, 8
 S -, 478
 T -, 473
 deductive closure, 362
 definable
 λ -, 151
 functional - by TR_α , 226
 defined
 for partial function, xx
 partial semantics, 78
 deflationary
 set of type inequalities, 433
 type inequality, 433
 type theory, 433
 degree, 178

- dependent
 - on a variable, 83
- depth
 - of a basis, 11
 - of a type, 11
- derivability
 - for subtyping, 432
- derivable
 - R - word, 152
 - from Γ , 8
 - proposition, 64
- derivable rule, 455
- derivative, 346
- descendants, 375
 - direct -, 375
- Dialectica interpretation, 241
- Diophantine, 209, 210
 - uniformly in n , 211
- directed, 238
 - subset, 404
- discharged, 9
- domain, 8, 308
- elementary equivalent, 79
- elementary equivalent at A , 79
- embed, 121
- embeddable, 294
- empty tree, 34
- empty word, 152
- encoded, 153
- enriched, 304
 - set of types, 304
 - syntactic type algebra, 304
- enumerates
 - slantwise, 149
- enumeration
 - problem, 55
 - for λ_{\equiv}^A , 385
- environment
 - term, 77
- environment in \mathcal{D} , 535
- equality
 - strong - on types, 329
 - weak - on types, 329
- equation
 - over \mathcal{D} , 86
- equational, 209
- version, 189
- equational logic
 - extended by \mathcal{E} , 294
- equi-recursive, 300
- equivalence of recursive types
 - completeness, 419
- equivalent
 - sets of constraints, 375
 - sr, 310
 - weakly, 322
- evaluation
 - of a term, 396
- expansive, 92
- extensional, 77, 537
 - equality, 6
 - hereditary, 235
 - equivalence, 124
- extensionality, 76
- F-semantics, 439, 562
- favorable, 603
- filter, 457, 480, 480
- filter model
 - for λ , 541
 - for λI , 541
- filter quasi λ -model, 540
- filter structure, 481
- final
 - coalgebra, 339
- final labels, 601
- finite
 - element, 405
 - tree, 326
- finite completeness theorem, 123
- finitely generated, 149, 160
- fixed point
 - combinator, 68, 236
 - least, 221
 - recursion, 236
- flat, 371
- fold-unfold, 336
- folding, 310, 323
- forcing
 - $\Gamma \Vdash A$, 66
 - $k \Vdash A$, 65

- $\mathcal{K} \Vdash A$, 65
- $\mathcal{K} \Vdash \Gamma$, 66
- $\Vdash A$, 66
- forgetful map, 21
- formulæ-as-types, 613
- formulas-as-types, 271
- free variable
 - lemma
 - for $\lambda_{\rightarrow}^{\text{Cu}}$, 16
 - of a λ -term, 5
 - of a type, 314
- free(ly generated), 35
- fully abstract, 139
- fully eta (*f.e.*) expanded, 41
- function
 - partial, xx
- functional
 - continuous, 243
 - hereditarily finite, 235
- functional programming, 250
- fundamental theorem
 - for semantic logical relations, 99
 - for syntactic logical relations, 94
- Gödel's \mathcal{T} , 217
- Gödel-number, 71
- Galois connection, 513
- game
 - rules, 602
- game context, 600
- game types, 600
- Gandy-hull, 100
- generated by, 457
- gip: inhabitation problem, 605
- global move, 601
 - C_2 , 608
 - C_3 , 609
- good
 - \rightarrow -, 574
- grammar
 - 2-level, 26
- graph, 470
- greatest lower bound, 454
- group
 - Thompson-Freyd-Heller, 204, 240
- h \uparrow -closed, 578
- has a lnf, 26
- has index i , 178
- has type, 10
- has type A relative to Γ , 10
- head-reducible to, 119
- height, 570
- hereditarily
 - adfluent, 231
 - extensional operators, 242
 - finite, 231
 - monotone functions, 77
 - recursive operators, 242
- hereditarily non-trivial, 424
- hierarchy theorem, 118
 - revisited, 121
- higher order logic, 186
- Hindley-Milner algorithm, 59
- holds (validity)
 - for equations, 294
 - for existential statements, 307
- homogeneous, 433
 - in α , 433
 - type theory, 433
- ideal, 408, 429, 440
- implicational propositional logic, 64
- inconsistent
 - equation, 7
 - terms, 90
 - theory, 7
- indeterminate, 306, 308
- induction loading, 53
- inductive, 220, 225
 - sr, 399
 - types, 395
- inductively, 400
- inequality, 174
- inflationary
 - set of type inequalities, 433
 - type inequality, 433
 - type theory, 433
- inhabitation
 - for λ_{\leq}^A , 385
- inhabitation problem, 55, 598
 - generalized, 605

- inhabited relative to Γ , 10
 initial
 algebra, 339
 initial label, 601
 initial position, 602
 instance
 of a problem, 152
 instantaneous description, 607
 interpolation
 dual - problem, 166
 equation, 166
 problem, 166
 interpretation, 85, 331
 of a type, 396
 of terms, 457
 of types, 458
 under ρ , 67
 intersection type
 language, 452
 preorder, 453
 structure, 477
 theory, 453
 intersection type theory
 with universe, 464
 invariant, 105
 μ -, 317
 inversion, 485
 Inversion Lemma, 384
 for λ_{\cap}^T , 485
 for $\lambda_{\rightarrow}^{\text{Ch}}$, 18
 for $\lambda_{\rightarrow}^{\text{dB}}$, 19
 for $\lambda_{\rightarrow}^{\text{Cu}}$, 16
 for syntactic type algebras, 384
 for type algebras, 383
 Inversion lemma
 for λ_{\leqslant}^S , 435
 for invertible type algebras, 435
 invertible, 302, 302, 434
 i-rank, 613
 irrelevant, 167
 is inhabited by, 10
 iso-recursive, 300
 isomorphic, 294
 isomorphic λ -models, 539
 isomorphism, 294
 Jacopini tableau, 89
 justifies
 set of equations -, 296
 type algebra -, 296
 kernel, 341
 Kleene application, 242
 Kripke model, 65
 Kripke type interpretation, 588
 labels, 601
 lambda Galois connection, 515
 lambda structure, 514
 natural, 515
 lambda structures, 515
 large, 117
 lattice
 ω -algebraic -, 405
 algebraic -, 405
 complete -, 404
 layered non-empty subfamily, 77
 lazy, 470, 524, 562
 lazy lists, 347
 lazy zip structure, 518
 lead symbol, 325
 least congruence extending \mathcal{E} , 294
 left, 153
 left adjoint, 513
 legal, 15
 length, 71
 lifting of S , 100
 limit, 220
 lists, 222
 lnf: long normal form, 26
 local move, 601
 k -th, 602
 logical framework, 265
 logical relation
 semantic, 98
 syntactic, 91
 logically equivalent
 sr, 310
 lower
 approximations, 436
 matching

- problem
 - with unknown X , 160
 - pure - problem, 160
- matrix, 41
- maximal theory, 140
- maximally trivial, 424
- meet semi-lattice, 477
- meet semi-lattice with universe, 477
- minimal model, 140
- model, 87
 - λ , 536, 536
 - η , 537
 - λ^0_{\rightarrow} , 78
 - of λ_T , 219
 - of λ_Y , 237
 - quasi λ -, 536
 - typed λ -, 78
 - untyped λ -, 84
- monotonic, 405
- monster, 30
- morphism, 76, 294, 339
 - for Cartesian monoids, 198
 - of type structures, 432
- most general unifier, 57
- move
 - global, 608
- multi head-reducible to, 119
- multi-sorted, 37
- multiplication of \mathcal{T} -bases, 475
- multiset, 46
 - order, 47
- natural, 470, 514
- natural deduction, 9
- natural lambda structures, 524
- natural zip structure, 518
- negative
 - α - in A , 394
 - occurrence, 394
- next ID function, 607
- nf: normal form, 45
- no, 153
- non-empty
 - logical relation, 98
- norm, 577
- normal form
 - R -, 45
 - Φ -, 42
 - long, 26
- normalization theorem
 - strong -
 - for λ^A_{\rightarrow} , 52
 - weak -
 - for λ^A_{\rightarrow} , 49
- normalizing
 - strongly R -, 45
 - weakly R -, 45
- numeral
 - Church, 32
 - in \mathcal{F} , 211
 - in \mathcal{T} , 217
- observable type, 423
- observational
 - equivalence, 124
- observationally equivalent, 423
- ogre, 568
- open set
 - Scott -, 405
- order, 562
 - of a basis, 11
 - of a matching problem, 160
 - of a type, 11
- ordering on \mathcal{D}_∞ , 550
- ordinals, 220
- orphan, 371
- output-type, 160
- pairing, 194
 - $A \times A \rightarrow A$, 67
 - R -, 38, 39
 - surjective, 39
- partial
 - CPO of - functions, 405
 - function, xx
 - semantics
 - $\llbracket M \rrbracket_\rho$ defined, 78
 - $\llbracket M \rrbracket_\rho$ undefined, 78
 - surjective homomorphism, 98
- partially ordered set, 404
- pattern, 169
- pattern matching, 60

- permutation of arguments, 107
- perpetual, 563
- persistently head normalizing, 577
- persistently normalizing, 577
- persistently strongly normalizing, 577
- Platek's \mathcal{Y} , 237
- play
 - in a game, 601
- Plotkin's Model, 568
- polynomial, 339
- polynomial structure, 82
- poor type, 149
- poset: partially ordered set, 404
- position, 601
 - next, 602
- positive
 - α - in A , 394
 - occurrence, 394
 - type, 394
- predicate, 8, 473
- preorder
 - intersection type -, 453
- preserves, 574
- prime, 302
- principal, 457
 - recursive basis
 - of a term, 389, 441
 - recursive triple
 - of a term, 389, 441
 - recursive type
 - of a term, 389, 441
 - type algebra
 - of a term, 389, 441
 - type theory
 - of a term, 441
- principal pair, 59
- principal type, 59
- principal type theorem, 59
 - second, 60
- problem
 - EQA, 598
 - ETW, 598
 - IHP, 598
 - WTG, 598
- product, 38, 39
- Cartesian
 - R -, 39
- programming
 - functional, xv
 - imperative, xv
- projection, 95, 209
- proper, 41, 308, 470, 524
- proper strict zip structure, 519
- property
 - λK^o -, 54
- propositions-as-types, 64
- protective, 317
- pseudo-negation, 41
- pseudo-terms, 15
- quasi λ -model, 537
- $R\text{-exp}$, 456
- $R\text{-red}$, 456
- rank
 - of a basis, 11
 - of a formula, 193
 - of a matching problem, 160
 - of a type, 11
- realizability interpretation, 468
- recursion
 - bar, 229, 230
 - nested, 241
 - primitive
 - simultaneous, 241
- recursive
 - primitive
 - functionals, 216
 - functions, 215
- recursor, 216
 - of type A , 216
- redex
 - βN , 618
 - I -, 53
 - K -, 53
 - K^+ -, 53
 - K^o -, 53
- reduced form, 325
 - principal, 325
- Reducibility Theorem, 116
- reducible

- decision problems
 - many one, 152
 - Turing, 152
- types, 107
- reduction
 - T , 217
 - α - - on types, 315
 - $\dot{\mu}$ - - on types, 315
 - μ - - on types, 323
 - β -, 6
 - η -, 6
 - \mathcal{R} - - on types, 308
- reflexive structure, 406
- regular, 327, 327
- relation
 - semantic, 98
 - syntactic, 91
- relativized, 598
- relevant, 167
- representable, 458, 544
- represented in, 126
- retract
 - SP , 197
- retraction pair
 - SP , 197
- rewrite rule, 152
- rich type, 149
- right, 153
- right adjoint, 513
- rightmost redex, 49
- rule
 - admissible, 455
 - derivable, 455
- safe
 - forever - $\dot{\mu}$ -type, 317
- satisfaction, 403, 459
 - $T \models M = N$, 85
 - $\rho \models A$, 62
 - $\rho \models \Gamma$, 62
 - $\mathcal{D}, \rho, \xi \models M : A$, 85
 - $\mathcal{D}, \rho \models M = N$, 84
 - $\mathcal{D}, \rho \models T$, 84
 - $\mathcal{D} \models T$, 85
 - $\mathcal{M}, \rho \models M = N$, 78
 - $\mathcal{M} \models M = N$, 78
- $\mathcal{M} \models \mathcal{E}$, 87
- $\Gamma \models M : A$
 - for λ_{\cap}^S , 574
 - in term model, 396
- satisfies, 343
 - existential statement, 307
- saturated, 395
- search-type, 160
- section
 - A -, 139
- self-capturing, 319
- Semantic Satisfiability, 592
- semantics
 - partial, 77
 - separable, 204
 - separable from, 90
 - signature functor, 338
 - simple, 308
 - simple semantics, 573
 - simplified syntax, 5
 - simulation, 438
 - simultaneous recursion
 - over $\mathcal{A}(\vec{X})$, 308
 - over \mathcal{A} , 308
 - slantwise
 - λ -defineble, 148
 - enumerates, 149
 - small, 117
 - SN: strongly normalizing, 45, 577
 - solitary game, 601
 - solution, 307, 605
 - solvable
 - game, 603
 - solves, 307
 - M solves P , 605
 - sound
 - β , 487
 - η -, 493
 - η^U -, 493
 - soundness, 459, 574
 - Spector's \mathcal{B} , 230
 - sr: simultaneous recursion, 308
 - stable
 - $\mathbf{H}\mathbf{N}$, 578
 - \mathbf{N} , 578

- SN, 578
- standard \Rightarrow_μ -reduction, 353
- statement, 8
 - \mathcal{S} -, 478
 - \mathcal{T} -, 473
- step function, 417, 505
- streams, 345
- strict, 405, 537
 - function, 504
- strict lambda structure, 524
- strict lambda structures, 524
- strict zip structure, 519
- string, 201
- strongly normalizing, 45
 - β -, 577
- structure
 - full type, 75
 - type, 75
- sub, 464
- subject, 8, 433, 473
- subject reduction
 - for $\lambda^A_=\$, 384
- subject reduction property
 - for $\lambda^{\text{dB}}_\rightarrow$, 19
 - for $\lambda^{\text{Cu}}_\rightarrow$, 17
- substitution, 328
 - naive, 314
 - smart, 314
 - type
 - in term, 14
 - in type, 8
- substitution lemma
 - for $\lambda^{\text{Ch}}_\rightarrow$, 18
 - for $\lambda^{\text{dB}}_\rightarrow$, 19
 - for $\lambda^{\text{Cu}}_\rightarrow$, 17
- substitutive, 92
 - semantic logical relation, 100
- substitutor, 57
- subterm closure, 358
- subtree, 327
 - subtree of t at w , 327
- subtyping relation, 436
- successor, 220, 220
- sufficient
 - set of constants, 123
- sup: supremum, 404
- support, 46
- supremum, 220, 404
- surjective, 149, 195
- surjective pairing, 40
- syntactic
 - morphism, 302
 - type algebra, 293
- syntactical type structure, 478
- target, 12
- term
 - $\lambda\Phi$, 583
 - $\lambda\perp$, 583
 - pure, 86
- term environment, 396, 457
- term model, 396
 - closed
 - for λ_B , 235
 - open, 88
 - terms of λ_{SP} , 195
 - terms of $\lambda^{\text{Ch}}_\rightarrow$, 14
 - terms of type A , 14
 - terms-as-proofs, 271
- theory
 - consistent, 424
- theory of \mathcal{M} , 78
- to the right, 49
- top, 432, 454, 464
- topology
 - Scott -, 405
- trace
 - of a under f , 142
- transfinite
 - induction, 225
 - iteration, 221
 - recursion, 225
- tree, 326
 - tree functor, 340
 - tree game, 601
 - G_0 , 603
 - G_1 , 608
 - tree-unfolding, 329
 - trivial, 76, 198
 - trivial type, 423
 - true in \mathcal{D} , 458, 536

- truncation, 327
- typability, 55
 - for λ_{\equiv}^A , 385
- typable, 10, 454
- typable relative to, 10
- type
 - assignment
 - à la* Curry, 8
 - à la* de Bruijn, 16
 - atom, 7
 - characteristic, 126
 - checking, 55
 - depth, 11
 - for words, 33
 - function space, 7
 - level, 11
 - natural number, 217
 - of λ_{SP} , 195
 - order, 11
 - poor, 149
 - rank, 11
 - reconstruction, 55
 - reduction
 - $\beta\eta$, 107
 - $\beta\eta_{SP}$, 208
 - rich, 149
 - simple, 7
 - trivial, 423
- type algebra, 291
 - (equation over, 294
 - free -, 291, 300
- interpretation of a -, 403
- quotient of, 295
- satisfying a set of equations, 294
- satisfying an equation, 294
- subalgebra, 300
- type assignment
 - λ_{\cap}^{BCD} , 454
 - λ_{\cap}^T , 473
- type checking
 - for λ_{\equiv}^A , 385
- type environment, 458
- type interpretation, 413, 573
 - approximate -, 411
- type intersection theory
 - without universe, 464
- type nesting, 564
- type reconstruction
 - for λ_{\equiv}^A , 385
- type structure, 431, 477
 - determined by \mathcal{T} , 433
 - full -, 67
 - graph -, 479
 - intersection -
 - with universe, 477
 - without universe, 477
 - lazy -, 479
 - natural -, 479
 - proper -, 479
- type theory, 464
 - axiomatized by \mathcal{H} , 433
 - for subtyping, 433
 - graph -, 470
 - intersection -, 453, 464
 - with universe, 464
 - without universe, 464
 - lazy -, 470
 - natural -, 470
 - proper -, 470
- type-semantics theorem, 541
- type-tree, 326
- typed applicative structure, 76
- typed lambda theory, 86
- typed terms
 - of $\lambda_{\equiv}^{A,\text{Ch}_0}$, 299
 - of $\lambda_{\equiv}^{A,\text{Ch}}$, 298
- typewriter automaton, 607
- ultrametric space, 337
- undefined
 - for partial function, xx
 - partial semantics, 78
- unfolding, 323
- unfolding (modulo \mathcal{R}), 310
- unicity of types
 - for $\lambda_{\rightarrow}^{\text{Ch}}$, 19
 - for $\lambda_{\rightarrow}^{\text{dB}}$, 19
- unification
 - first-order, 58
 - problem
 - with unknown X , 160

- pure - problem, 160
- system of - problems, 162
- theorem, 58
- with constants from Γ , 162
- with several unknowns, 161
- unifier, 57
- uniform
 - set, 408
- uniquely closed, 313
- universal n -ary relation, 95
- universal top, 452
- universe, 452, 464
- upper
 - approximations, 436
- upperbound, 404
- useless, 424
- valid, 364
 - k -, 364
- valuation, 67
 - partial, 77
- variable
 - bound, 5
 - bound type -, 314
 - free, 5
 - free type -, 314
- variant, 58
- weak head expansion, 92
- weakening lemma
 - for $\lambda \rightarrow^{\text{Cu}}$, 16
- weakly confluent, 46
- weakly encoded, 153
- well-founded, 303
- winning, 602
- witnesses, 89
- WN: weakly normalizing, 45
- word, 152
- yes, 153
- zero, 220
- zip structure, 518

Index of names

- Abadi and Cardelli [1996], 446
 Abadi and Fiore [1996], 300
 Abadi and Plotkin [1990], 408, 420, 447
 Abelson, Dybvig, Haynes, Rozas, IV, Friedman, Kohlbecker, Jr., Bartley, Halstead, [1991], xvii, 252
 Abramsky [1991], 450, 509
 Abramsky and Jung [1994], 237
 Abramsky [1990], 444
 Abramsky and Ong [1993], 462, 469, 535, 562, 562, 562, 582, 582, 582, 593, 594
 Ackermann [1928], 215, 304
 Aczel [1988], 334
 Aho, Sethi, and Ullman [1986], 446
 Alessi [1991], 535, 535, 544, 553, 553, 567
 Alessi [1993], 548, 569
 Alessi, Barbanera, and Dezani-Ciancaglini [2003], 485, 490
 Alessi, Barbanera, and Dezani-Ciancaglini [2004], 534, 544, 548, 569
 Alessi, Barbanera, and Dezani-Ciancaglini [2006], 485, 490
 Alessi, Dezani-Ciancaglini, and Honsell [2001], 535, 563
 Alessi, Dezani-Ciancaglini, and Honsell [2004], 534, 553
 Alessi and Barbanera [1991], 613
 Allouche and Shallit [2003], 394
 Amadio [1991], 420, 426
 Amadio and Cardelli [1993], 360, 367, 431, 436, 439, 439, 447
 Amadio and Curien [1998], 420, 582, 582
 Andrews [2002], 265
 Appel and McAllester [2001], 420
 Appel [1992], 253
 Ariola and Klop [1994], 381
 Ariola and Klop [1996], 324, 351, 360, 362, 367, 367
 Augustson [1999], 259
 Avigad, Donnelly, Gray, and Raff [2007], 266
 Baader and Nipkow [1998], 59
 Baader and Snyder [2001], 59
 Backus [1978], 251
 Baeten and Boerboom [1979], 535, 563
 van Bakel [1992], 582
 van Bakel [1993], 474
 van Bakel, Barbanera, Dezani-Ciancaglini, and de Vries [2002], 583
 Baldridge [2002], 286
 Barbanera, Dezani-Ciancaglini, and de'Liguoro [1995], 268, 275, 475
 Barendregt [1974], 39, 40, 162, 162, 195
 Barendregt [1984], ii, iii, 5, 5, 6, 6, 32, 42, 46, 69, 85, 90, 90, 108, 111, 124, 142, 202, 219, 249, 250, 251, 251, 252, 252, 267, 317, 329, 350, 406, 408, 408, 408, 416, 416, 416, 418, 423, 427, 503, 534, 534, 535, 538, 544, 549, 549, 549, 552, 560, 561, 563, 582, 582, 582, 583, 619
 Barendregt [1992], 13, 20, 246, 262, 265
 Barendregt and Barendsen [1997], 263
 Barendregt, Bunder, and Dekkers [1993], 267
 Barendregt, Coppo, and Dezani-Ciancaglini [1983], 85, 408, 450, 452, 452, 462, 469, 469, 469, 481, 497, 534, 534, 571, 572, 573, 576, 582, 593, 594
 Barendregt and Ghilezan [2000], 267, 475
 Barendregt and Rezus [1983], 420
 Barendsen and Smetsers [1993], [1996], 257
 Bekić [1984], 349
 van Benthem Jutting [1977], 263
 van Benthem [1995], xviii, 285, 286
 van Benthem and ter Meulen [1997], 286
 Berarducci and Böhm [1993], 245, 248
 Berline [2000], 562, 587
 Bernardi [2002], 286

- Bertot and Castéran [2004], 15, 260, 263, 265
 Bezem [1985a], 234
 Birkedal and Harper [1999], 420
 Böhm [1966], 245
 Böhm [1975], 650, 650
 Böhm and Berarducci [1985], 245, 246
 Böhm and Dezani-Ciancaglini [1975], 577
 Böhm and Gross [1966], 245
 Böhm, Piperno, and Guerrini [1994], 245, 248, 249
 Bono, Venneri, and Bettini [2008], 613
 Bove, Dybjer, and Norell [2009], 265
 Brandt and Henglein [1998], 360, 360, 361, 431, 436, 437, 437, 438, 438, 438, 444, 447
 Breazu-Tannen and Meyer [1985], 291
 Bruce, Meyer, and Mitchell [1990], 420, 427
 de Bruijn [1970], [1994], 262
 de Bruijn [1972], 115, 317
 Burstall [1969], 445
 Buszkowski and Penn [1990], 279
 Capitani, Loretto, and Venneri [2001], 613
 Capretta and Valentini [1998], 268
 Cardelli [1988], 445
 Cardone [1989], 420
 Cardone and Hindley [2009], 1
 Cardone [2002], 429
 Cardone [1991], 447
 Cardone and Coppo [2003], 335, 352, 352, 353, 355
 Cardone and Coppo [1990], 450
 Cardone and Coppo [1991], 408, 414, 418, 419, 419, 447
 Cardone, Dezani-Ciancaglini, and de'Liguoro [1994], 447
 Church [1932], v
 Church [1933], v
 Church [1936], 245
 Church [1940], xvii, 185, 265
 Church [1941], 1, 43, 252
 Church and Rosser [1936], 252
 Coppo and Dezani-Ciancaglini [1980], 452, 462
 Coppo, Dezani-Ciancaglini, Honsell, and Longo [1984], 485, 534, 534, 535, 548, 553, 562
 Coppo, Dezani-Ciancaglini, and Longo [1983], 534, 561
 Coppo, Dezani-Ciancaglini, and Sallé [1979], 462, 464
 Coppo, Dezani-Ciancaglini, and Venneri [1981], 462
 Coppo, Dezani-Ciancaglini, and Zacchi [1987], 462, 534, 553, 582, 582, 582, 593, 594
 Coppo [1985], 408, 418, 418, 419, 447
 Cosmadakis [1989], 423, 443
 Courcelle, Kahn, and Vuillemin [1974], 444
 Courcelle [1983], 326, 326, 334, 334, 334, 335, 336, 337, 338, 428, 447
 Cousineau, Curien, and Mauny [1987], 253
 Crossley [1975], 1
 Curien [1993], 199, 253
 Curry [1934], v, xvii
 Curry [1969], 59, 251, 387, 446
 Curry and Feys [1958], xvii, xviii, 265, 490
 Davis [1973], 164
 David and Nour [2007], 395
 Davis, Robinson, and Putnam [1961], 164
 Dedekind [1901], 339
 Dekkers [1988], 142, 148
 Dekkers, Bunder, and Barendregt [1998], 267
 Dezani-Ciancaglini, Ghilezan, and Likavec [2004], 535
 Dezani-Ciancaglini, Ghilezan, and Venneri [1997], 613
 Dezani-Ciancaglini, Honsell, and Alessi [2003], 464, 573, 576
 Dezani-Ciancaglini, Honsell, and Motohama [2005], 462, 572, 582, 582, 618
 Dezani-Ciancaglini, Honsell, and Motohama [2001], 573, 583

- Dezani-Ciancaglini and Margaria [1986], 562
- Di Gianantonio and Honsell [1993], 476, 553
- Došen [1992], 278
- van Draanen [1995], 238
- Dyckhoff and Pinto [1999], 276
- van Eekelen and Plasmeijer [1993], xviii, 255
- ten Eikelder [1991], 360
- Elbers [1996], 265
- Endrullis, Grabmayer, Klop, and van Oostrom [2010], 314
- Endrullis, Grabmayer, Klop, and van Oostrom [2011], 352, 355, 359, 361, 381
- Engeler [1981], 408, 462, 469, 566, 566
- Espirito Santo [2000], 268
- Espirito Santo [2007], 268
- Espirito Santo, Ghilezan, and Ivetić [2008], 268
- Euclid of Alexandria [-300], 261
- Fiore [1996], 447
- Fiore [2004], 446
- Flajolet and Sedgewick [1993], 181
- Fortune, Leivant, and O'Donnell [1983], 148
- Fox [2003], 266
- Freyd [1990], 446
- Freyd [1991], 446
- Freyd [1992], 446
- Friedman [1975], 98, 99, 113, 122
- Gandy [1980a], 50
- Gandy [1980b], 45, 53, 71
- Gapeyev, Levin, and Pierce [2002], 300, 436, 448
- Gentzen [2008], 268
- Gentzen [1969], 262
- Gentzen [1936], 275
- Ghilezan [1996], 582
- Gierz, Hofmann, Keimel, Lawson, Mislove, and Scott [1980], 504, 549
- Girard [1971], 53, 395
- Girard [1995], 257
- Girard, Lafont, and Taylor [1989], xx, 1, 13, 246, 258, 476
- Gödel [1931], 194
- Gödel [1958], 216, 241, 242
- Goguen, Thatcher, Wagner, and Wright [1977], 300, 428
- Goldfarb [1981], 161, 176, 182
- Gonthier [2008], 266
- Gordon, Milner, and Wadsworth [1979], 445
- Gordon [1994], 255, 256
- Gordon and Melham [1993], 265
- Grabmayer [2005], [2007], 447
- Grabmayer [2005], 360, 367
- Grabmayer [2007], 325, 382
- de Groote [1995], 265, 271
- de Groote and Pogodalla [2004], 286
- Grzegorczyk [1964], 39
- Gunter [1992], 237
- Gunter and Scott [1990], 253, 446
- Hales [2005], 266
- Harrison [2009a], 265
- Harrison [2009b], 266
- Harrop [1958], 67
- Henderson [1980], 252
- Henkin [1950], 75
- Herbelin [1995], 268, 276
- Hilbert and Ackermann [1928], 252
- Hindley [1969], 59, 60, 251, 390, 446
- Hindley [1983], 85, 573, 576
- Hindley [1992], 450
- Hindley [1997], 1, 60, 273
- Hindley and Seldin [1980], 634, 638, 642, 648, 648, 650
- Hinze, Jeuring, and Löh [2007], 258
- Hodges [1983], 250
- Honsell and Lenisa [1993], [1999], 562
- Honsell and Lenisa [1999], 462, 563, 563, 618
- Honsell and Ronchi Della Rocca [1992], 462, 476, 476, 553, 582, 582, 582, 582, 583, 593, 594, 597
- Howard [1970], 50, 228
- Howard [1980], 265, 271
- Hudak, Peterson, and Fasel [1999], 445

- Hudak, Peyton Jones, Wadler, Boutel,
Fairbairn, Fasel, Guzman, Ham-
mond, Hughes, Johnsson, [1992],
255
- Huet [1975], 160
- Hughes [1984], 254
- Hughes [1989], 250, 254
- Hutton [2007], xviii
- Hyland [1975/76], 582, 582
- Iverson [1962], 251
- Jacopini [1975], 89, 535, 563
- Jay [2009], 448
- Joachimski and Matthes [2003], 268
- Johnsson [1984], 254
- Johnstone [1986], 512
- Joly [2001a], 144
- Joly [2002], 149, 160
- Joly [2005], 159, 159
- Jones [1982], 164
- Jones [1993], 258
- Kamareddine, Laan, and Nederpelt [2004],
1, 1
- Kanazawa [1998], 279
- Kaufmann, Manolios, and Moore [2000],
265
- Kfoury and Wells [1995], 55
- Kleene [1936], 249, 249
- Kleene [1952], xx
- Kleene [1959a], 235, 243
- Kleene [1959b], 235
- Kleene [1975], 248
- Klein, Elphinstone, Heiser, Andronick,
Cock, Derrin, Elkaduwe, Engel-
hardt, Kolanski, Norrish, [2009],
266
- Klop [1980], 53, 111, 196
- Klop [1992], 308, 324
- Klop and de Vrijer [1989], 195
- Klop, van Oostrom, and van Raamsdonk
[1993], 324
- J.W. Klop, V. van Oostrom, and F. van Raamsdonk
[1993], 354
- Koopman and Plasmeijer [1999], xviii
- Koopman and Plasmeijer [2006], xviii
- Koster [1969], 360, 362, 367, 444
- Koymans [1982], 253
- Kozen [1977], 380
- Kozen [1997], 598, 607, 607
- Kozen, Palsberg, and Schwartzbach [1995],
360, 447
- Kreisel [1959], 235
- Kripke [1965], 67
- Krivine [1990], 482, 578, 582, 582
- Krivine [1993], 641
- Kurata and Takahashi [1995], 613
- Kurtonina [1995], 278
- Kurtonina and Moortgat [1997], 284, 287
- Kuśmierek [2007], 620, 620
- Lambek [1958], 278, 282
- Lambek [1961], 278, 282
- Lambek [1980], 197
- Lambek and Scott [1981], 1
- Landau [1960], 263
- Landin [1964], 252
- Lax [2002], xvii
- Leivant [1983b], 247
- Leivant [1983a], 613
- Leivant [1986], 582
- Leivant [1990], 148
- Leroy [2009], 266
- Lévy [1978], 49, 49, 71
- Liquori and Ronchi Della Rocca [2007],
613
- Loader [2001a], 159
- Loader [2001b], 152, 159
- Loader [2003], 165
- Loader [1997], 177
- Longo [1988], 582
- Lopez-Escobar [1983], 613
- MacQueen, Plotkin, and Sethi [1986], 408,
426, 447
- Mairson [1992], 62
- Makanin [1977], 182
- Martin-Löf [1984], 265, 265, 271
- Marz [1999], 371
- Matiyasevič [1972], 164, 212
- Matiyasevič [1993], 164
- Mayr and Nipkow [1998], 197
- McCarthy, Abrahams, Edwards, Hart,
and Levin [1962], xvii, 252

- McCarthy [1963], 446
 McCracken [1979], 420
 Mellies [1996], 318
 Mendler [1987], 393
 Mendler [1991], 395, 397
 Meyer [1982], 544
 Milner [1978], 59, 251, 253, 444, 446
 Milner, Tofte, Harper, and McQueen [1997], xvii
 Mints [1989], 613
 Mints [1996], 268, 268, 275, 276
 Mitchell [1996], 586
 Mitschke [1976], 563
 Mogensen [1992], 249, 249, 249
 Moller and Smolka [1995], 360
 Montague [1973], xviii
 Moot [2002], 285, 286
 Morrill [1994], 286
 Morris [1968], 446
 Muzalewski [1993], 265
 Nadathur and Miller [1988], 250
 Nederpelt [1973], 53
 Nederpelt, Gevers, and de Vrijer [1994], xix, 15, 262
 Nerode, Odifreddi, and Platek [In preparation], 1
Nikhil, R. S. [2008], 259
 Nipkow, Paulson, and Wenzel [2002a], 265
 Nipkow, Paulson, and Wenzel [2002b], 260, 263
 van Oostrom [2007], 320
 Oostdijk [1996], 265
 Padovani [2000], 161, 165
 Padovani [1996], 177
 Pair [1970], 444
 Parikh [1973], 59
 Park [1976], 423, 462, 534, 582
 Pentus [1993], [2006], 285
 Péter [1967], 71, 216, 241
 Peyton Jones [2003], xviii
 Peyton-Jones [1987], 445
 Peyton Jones [editor], Hughes [editor], Augustsson, Barton, Boutel, Burton, Fraser, Fasel, Hammond, Hinze, [1999], 445
 Peyton Jones, Vytiniotis, Weirich, and Washburn [2006], 259
 Peyton Jones and Wadler [1993], 255
 Pierce [2002], 290
 Pil [1999], 259
 Pimentel, Ronchi Della Rocca, and Roversi [In preparation], 613
 Plasmeijer, Achten, and Koopman [2007], 258
 Plasmeijer and van Eekelen [2002], xviii
 von Plato [2001b], 268
 von Plato [2001a], 268
 von Plato [2008], 268
 Platek [1966], 236
 Plotkin [1975], 253, 469
 Plotkin [1977], 236, 237
 Plotkin [1982], 422, 552
 Plotkin [1980], 122
 Plotkin [1985], 300
 Plotkin [1993], 462, 523, 553, 567, 568, 568, 568
 Poincaré [1902], 263
 Polonsky [2010], 441
 Post [1947], 152
 Pottinger [1977], 268, 276
 Pottinger [1980], 582
 Pottinger [1981], 111
 Prawitz [1965], 95, 193, 193, 268
 Prawitz [1971], 53
 van Raamsdonk [1996], 237
 van Raamsdonk, Severi, Sørensen, and Xi [1999], 577
 Reynolds [1972], 253
 Reynolds [1993], 253
 Robertson, Sanders, Seymour, and Thomas [1997], 266
 Robinson [1965], 58
 Rogers Jr. [1967], 152
 Ronchi Della Rocca [2002], 613
 Ronchi Della Rocca [1988], 582, 593, 594

- Ronchi Della Rocca and Paolini [2004], 476, 583
 Rosser [1984], 1
 Rutten [2000], 339, 341, 447
 Rutten [2005], 347
 Salvati [2009], 613
 Schmidt-Schauß [1999], 177
 Schubert [1998], 164, 385
 Schwichtenberg [1975], 228
 Schwichtenberg [1976], 33
 Schwichtenberg [1999], 276
 Schwichtenberg and Berger [1991], 239
 Scott [1970], 265
 Scott [1972], 253, 420, 447, 450, 462, 534, 549, 549, 552, 567, 567, 582
 Scott [1975b], 290, 291, 291, 336
 Scott [1975a], 84, 85, 403, 502, 571, 573, 573, 576
 Scott [1980], 197
 Scott [1976], 420, 427, 427
 Sheeran [2005], xviii, 259
 Smyth and Plotkin [1982], 253, 420, 446
 Sørensen [1997], 55
 Sørensen and Urzyczyn [2006], xx, 13, 64, 265, 271
 Spector [1962], 229, 229, 230
 Statman [1979a], 67, 598
 Statman [1979], 62, 275
 Statman [1980b], 121, 136
 Statman [1980a], 108, 108, 115, 116, 116, 118, 118, 118, 119, 632
 Statman [1981], 163
 Statman [1982], 33, 123
 Statman [1985], 122
 Statman [1994], 291, 368, 374, 378, 385
 Statman [2000], 194, 194
 Statman [2007], 43
 Steedman [2000], 286
 Steele Jr. [1978], 253
 Steele Jr. [1984], 252
 Stenlund [1972], 53
 Stirling [2009], 161, 165
 Støvring [2006], 195
 Sudan [1927], 215
 Tait [1965], 228
 Tait [1967], 52, 53
 Tait [1971], 234
 Tatsuta and Dezani-Ciancaglini [2006], 572, 582, 618
Terese [2003], 196, 199, 308, 309, 321, 324, 324, 368, 369, 369
 Terlouw [1982], 226
 Thatcher [1973], 73
 Thompson [1995], xviii
 Tiede [2001], [2002], 285
 Troelstra [1973], 103, 234, 235, 241
 Troelstra [1999], 276
 Troelstra and Schwichtenberg [1996], 273
 Turner [1976], [1979], 254
 Turner [1981], 250, 254
 Turner [1985], 255
 Urban, Berghofer, and Norrish [2007], 317
 Urban and Tasson [2005], 317
 Urzyczyn [2009], 613
 Urzyczyn [1999], 573, 612
 Venneri [1994], 613
 Venneri [1996], 582
 Vermaat [2006], 286
 Vogel [1976], 234
 Voigtländer [2009], 258
 Vouillon and Mellies [2004], 420
 de Vrijer [1987], 50, 55, 71, 195
 de Vrijer [1989], 195
 Wadsworth [1971], 254, 254
 Wadsworth [1976], 419, 582, 582, 582, 597
 Wand [1987], 59, 387, 446
 Wansing [2002], 284
 Wells [1999], 13, 20
 Wells, Dimock, Muller, and Turbak [1997], 613
 Whitehead and Russell [1910-1913], xvii
 Wiedijk [2006], 266
 van Wijngaarden [1981], 26, 127, 290, 307, 444
 Xi [1997], 55
 Zeleny and Anderson [2001], 643
 Zucker [1974], 268, 276
 Zylberajch [1991], 535

Index of symbols

1. Terms

A. general

- B_n , 204
- C_0 , 204
- C_{n+1} , 204
- $E := \epsilon$, 34
- M^- , 24
- M^Γ , 24
- $N \cdot k$, 40
- $Pts :=_\beta p(t, s)$, 34
- R_A , 216
- Φ , 583
- \perp , 583
- ϵ , 34
- $\langle M_1, \dots, M_n \rangle$, 40
- K^∞ , 568
- K_* , 40
- $\text{left}_0^{A,B}$, 39
- $\text{pair}_0^{A,B}$, 39
- proj_k^n , 40
- $\text{right}_0^{A,B}$, 39
- tuple^n , 40
- B , 40, 230
- B^c , 230
- $B_{A,B}^c$, 230
- $B_{A,B}$, 230
- CR , 46
- C , 40
- C_* , 40
- WCR , 46
- W , 40
- Y_A , 237
- $false$, 31
- $plus$, 32
- $test$, 36
- $times$, 33
- $true$, 31
- c_n , 32
- $empty?$, 34
- $concat$, 33

B. operations

- $M(\mathcal{X}_1, \dots, \mathcal{X}_k)$, 230

- S_M , 49
 - $\#M$, 71
 - $\Gamma \upharpoonright FV(M)$, 16
 - $\omega(M)$, 418
 - $\{\}_{y'}^x$, 585
 - \square , 583
 - \square_L , 583
 - \square_Φ , 583
 - \square_\perp , 583
 - $\text{order}(M)$, 562
 - p , 34
 - \mathcal{XY} , 230
 - $\text{pp}(M)$, 59
 - $\text{pt}(M)$, 59
 - $\text{1th}(M)$, 71
- C. classes**
- T_{CM}^n , 199
 - T_{CM} , 199
 - $[A]_\Gamma^T$, 586
 - $\Lambda \rightarrow [D]$, 86
 - Λ , 5
 - $\Lambda\Phi$, 583
 - $\Lambda\perp$, 418, 583
 - $\Lambda \rightarrow [D]$, 86
 - Λ_{SP} , 195
 - $\Lambda \rightarrow^{\text{Ch}}$, 14
 - $\Lambda \rightarrow^{\text{Ch}}(A)$, 14
 - $\nabla(D)$, 126, 126
 - H_N , 578
 - N , 578
 - PHN , 578
 - PN , 578
 - $X \in \mathcal{HF}_A$, 231
 - $\Lambda \rightarrow^{\text{dB}}$, 15
 - $\mathcal{A}(M)$, 456
 - $\mathcal{A}_T(M)$, 584
 - \mathcal{C}_0 , 127
 - \mathcal{C}_1 , 127
 - \mathcal{C}_2 , 127
 - \mathcal{C}_3 , 127
 - \mathcal{C}_4 , 127
 - \mathcal{C}_5 , 127

\mathcal{C}_{-1} , 127	\mathbb{T}_{\approx}^* , 330
$\mathcal{M}_{\Lambda(\beta)}$, 578	\mathbb{T}_{μ}^* , 331
HF , 231	\mathcal{A}/\mathcal{E} , 295
NF , 22	\mathcal{S}_M , 441
$\text{nf}(\mathcal{X})$, 230	$(-)^*_{\mu}$, 331
vNF , 22	$(-)^*_{\approx}$, 330
$\vee^{\mathbb{T}}$, 13	$(-)^*_{\mathcal{R}}$, 330
WN , 45	B. special types
$\mathcal{A}(M)$, 418	\top^2 , 34
PSN , 578	U , 452, 464
SN , 577, 578	Bool , 31
D. relations	Bool_A , 31
$M \approx_D^{\text{ext}} N$, 124	Nat , 32
$M \approx_D^{\text{obs}} N$, 124	Sigma^* , 33
$P \# Q$, 7	C. operations
$T \vdash M = N$, 7	$((\mathcal{R})) \rightarrow A$, 391
$T \vdash_{\lambda\beta\eta} M = N$, 7	$A(i, j)$, 12
$\twoheadrightarrow_{\eta}$, 6	$A[\alpha := C]$, 8
$\twoheadrightarrow_{\beta\eta}$, 6	$A[\beta := B]$, 314
$\twoheadrightarrow_{\beta}$, 6	$A^k \rightarrow B$, 11
\rightarrow_{η} , 6	A^* , 57
$\rightarrow_{\beta\eta}$, 6	$A_1 \cap \dots \cap A_n$, 465
\rightarrow_{β} , 6	A_i , 12
\rightarrow_T , 217	$A[\beta := B]_{\not\in}$, 314
\twoheadrightarrow_{CM} , 199	$L(A; \Gamma)$, 27
$\ _N$, 171	$M[\alpha := B]$, 14
\simeq , xx	$N(A; \Gamma)$, 27
\rightarrow_{CM} , 199	$[A]$, 64
E. theories	$[\Gamma]$, 64
$\mathcal{E}(A)$, 87	$\llbracket A \rrbracket_{\Gamma}^T$, 588
\mathcal{E}^+ , 87	$\Gamma \uplus \Gamma'$, 475
$\mathcal{E}_{\beta\eta}$, 87	Γ_A , 64
$\lambda\beta\eta$, 6	$\Box(A)$, 564
Th_{\max} , 140	$\uparrow A$, 457, 480
2. Types/Propositions	$\uparrow X$, 457, 480
A. algebras, structures	φ^+ , 189
(U_{top}) , 464	n_k , 12
$[a]_{\approx}$, 293	$\llbracket A \rrbracket_{\xi}$, 85
$\mathbb{T}_{\mu}^{\mathbb{A}}$, 313	$\llbracket A \rrbracket_{\rho}$, 62
\mathbf{X} , 309	$\text{cf}(A)$, 488
\mathcal{A}/\approx , 293	$\text{class}(A)$, 119
$\mathcal{A}[\mathcal{R}]$, 309	depth , 554
\mathcal{A}_M , 389	$\text{dom}(\Gamma)$, 8
\mathbb{T}/\approx , 293	$\text{dpt}(A)$, 11
$\mathbb{T}[\mathcal{R}]^*$, 330	$\text{ls}(A)$, 325

$\text{ord}(A)$, 11	$\textcolor{red}{\mathbb{T}}_2$, 117
$\text{rk}(A)$, 11	$\mathcal{SC}^+(A)$, 397
$\sim A$, 41	E. relations
f , 557	$=_\mu$, 324
$\llbracket A \rrbracket_\tau$, 396	$=_{F\mu}$, 315
$\mathcal{I}^n \llbracket A \rrbracket_\eta$, 411	$=_{\mathcal{E}}$, 295
$\mathcal{I} \llbracket A \rrbracket_\eta$, 413	$=_{\mathcal{R}}^{-1}$, 369
$\mathcal{SR}(A)$, 352	$A =_\mu B$, 322, 353
$\mathcal{T}(A)$, 352	$A <_{\mathcal{T}} B$, 465
$\mathcal{T}^n \llbracket t \rrbracket_\eta$, 415	$A = B$, 453
and, 31	$A =_{\mathcal{T}} B$, 465
iff, 31	$A \equiv B$, 453, 465
imp, 31	$A \leq B$, 453, 464
not, 31	$A \leq_{\mathcal{T}} B$, 465
or, 31	$A \leq_{BCD} B$, 453
t, 568	$A \leq_{\beta\eta SP} B$, 208
D. classes	$A \leq_{\beta\eta} B$, 107
\mathcal{SC} , 358	$A \sim_{\beta\eta} B$, 107
$\mathcal{SC}_r^w(a)$, 358	$A \triangleleft_{SP} B$, 197
$[\mathbb{T}]$, 453	$A =_{BCD} B$, 453
$\mathcal{SC}_r^s(A)$, 361	R_n , 382
$\mathcal{SC}^w(\mathbf{a})$, 380	\rightarrow_α , 315
$\mathcal{SC}^s(A)$, 361	\rightarrow_μ , 323
$\mathbb{A}^\mathcal{T}$, 465	$\Gamma \vdash A$, 64
\mathbb{A}^{ABD} , 548	$\Gamma \vdash_{\text{PROP}} A$, 64
\mathbb{A}^{BCD} , 452	\Rightarrow_α^* , 315
\mathbb{A}_0 , 7	\Rightarrow_α , 315
\mathbb{A}_∞ , 7, 452	\Rightarrow_μ^n , 323
\mathbb{A}_∞^U , 452	\leq , 598
$\text{Typable}(\mathcal{X})$, 397	\leq_μ^* , 437
\mathbb{T} , 7, 452, 463, 598	\leq_{fin} , 436
\mathbb{T}^0 , 7	\approx , 293
\mathbb{T}^∞ , 7	\equiv_α , 315
\mathbb{T}_\cap^A , 463	\Rightarrow_μ^* , 323
$\mathbb{T}_\cap^\mathcal{T}$, 465	\Rightarrow_μ , 323
\mathbb{T}_\cap^A , 463	$\Rightarrow_{\mathcal{R}}^*$, 308
$\mathbb{T}_{CDHL(t)}$, 553	$\vdash ? : A$, 598
\mathbb{T}_\cap , 463	\lesssim , 442
\mathbb{T}_\cap^{BCD} , 452	$\vdash A$, 64
$\textcolor{red}{\mathbb{T}}_1$, 117	$\vdash_{BCD} A \leq B$, 453
$\textcolor{red}{\mathbb{T}}_3$, 117	$\rightarrow_{\dot{\mu}}$, 315
$\textcolor{red}{\mathbb{T}}_4$, 117	$\mathcal{E} \vdash^{\text{inv}} a = b$, 378
$\textcolor{red}{\mathbb{T}}_5$, 117	$\mathcal{S}(\mathcal{H}, A, B)$, 365
$\textcolor{red}{\mathbb{T}}_{-1}$, 117	$\mathcal{H} \vdash_{BH\leq} A \leq B$, 437
$\textcolor{red}{\mathbb{T}}_0$, 117	$\mathcal{H} \vdash_{BH} A = B$, 363

$=_{\mathcal{A}}^*$, 329	LTT ^U , 470
$\Gamma \Subset \Gamma'$, 475	NTT ^U , 470
F. theories	PTT, 470
(BH_{\leqslant}) , 437	3. Assignment
(μ) , 322	A. theories
(\mathcal{E}) , 294	Γ_M , 389, 441
(\mathcal{R}) , 308	Γ_M^μ , 393
(\mathcal{R}^*) , 373	$\Gamma_{\mathcal{R}}$, 391
Δ , 186	$\lambda\mu^+$, 394
$\Delta(n)$, 193	$\lambda\mu^*$, 335
Δ^- , 193	$\lambda_{\rightarrow}^{\mathbb{A}, \text{dB}}$, 15
δ , 189	$\lambda\mathcal{R}^*$, 335
$\delta(n)$, 193	$\langle \Gamma_M, \mathcal{A}_M, a_M \rangle$, 389, 441
TT^{-U} , 464	a_M , 389, 441
TT^U , 464	a_M^μ , 393
$\lambda\mu^{*\infty}$, 419	$\Lambda_{\underline{\underline{\mathcal{A}}}}^{\mathcal{A}, \text{Ch}_0}(A)$, 299
\mathcal{D}_n , 370, 378	$\Lambda_{\underline{\underline{\mathcal{A}}}}^{\mathcal{A}, \text{Ch}}(A)$, 298
(BH) , 363, 363	$\lambda\mu$, 336
ABD, 548	λ_{\rightarrow}^0 , 11
Alessi, 568	$\lambda_{\underline{\underline{\mathcal{A}}}}^{\mathcal{A}, \text{Cu}}$, 292
CDHL(t), 553	$\lambda_{\underline{\underline{\mathcal{A}}}}^{\mathcal{A}, \text{dB}}$, 298
TRS(\mathcal{R}), 308	$\lambda_{\cap}^{\text{Krivine}}$, 482
TRS ⁻¹ (\mathcal{R}), 369	$\lambda_{\cap}^{\text{BCD}}$, 454
TT, 464	λ_B , 229, 230
Krivine, 482	λ_T , 217
Krivine ^U , 482	λ_Y , 237
AO, 468	$\lambda_{\cap}^{\text{Krivine}^U}$, 482
BCD, 453, 468	λ_{\cap} , 473
CDS, 468	λ_{\cap}^T , 473
CDV, 468	$\lambda_{\cap}^{\text{HR}}$, 584
CDZ, 468	$\lambda_{\cap}^{\text{Park}}$, 584
CD, 468	$\lambda_{\rightarrow}^{\text{Ch}}$, 13, 13
DHM, 468	$\lambda_{\rightarrow}^{\text{dB}}$, 15
EHR, 476	$\lambda_{\rightarrow}^{\mathcal{A}, \text{Ch}}$, 13
Engeler, 468	$\lambda\mathcal{R}$, 336
HL, 468	$\lambda\mathcal{R}^*$, 336
HR, 468	$\lambda\approx$, 336
Park, 468	$\lambda_{\underline{\underline{\mathcal{A}}}}^{\mathcal{A}}$, 292
Plotkin, 468	$\lambda_{\underline{\underline{\mathcal{A}}}}^{\mathcal{A}, \text{Ch}_0}$, 299
Scott, 468	$\lambda_{\underline{\underline{\mathcal{A}}}}^{\mathcal{A}, \text{Ch}}$, 298
\mathcal{I}_n , 370, 378	$\lambda_{\rightarrow}^{\mathbb{T}/\approx}$, 297
ADH(M), 565	$\lambda_{\rightarrow}^{\mathbb{A}, \text{Ch}}$, 13
G. theories (classes of)	$\lambda_{\rightarrow}^{\mathbb{A}, \text{Cu}}$, 13
$\text{ADH}_n(M)$, 564	$\lambda_{\rightarrow}^{\text{Cu}}$, 13
GTT ^U , 470	$\lambda_{\mathcal{A}}$, 336

$\lambda\mathcal{E}$, 336	G_{Em} , 566
B. derivation	$X \Rightarrow Y$, 85
$\Gamma \vdash M : A$, 454	$\Phi_{\infty m}$, 550, 550
$\Gamma \vdash M : A,$, 473	$\Phi_{m\infty}$, 550
$\Gamma \vdash M : A$, 8	Φ_{mn} , 550
$\Gamma \vdash ? : A$, 598	$\Phi_{m\infty}$, 550
$\Gamma \vdash_{\lambda}^{\text{Cu}} M : A$, 8	\cap , 505
$\Gamma \vdash_{\cap}^{\text{BCD}} M : A$, 454	\cap_{\leq} , 505
$\Gamma \vdash_{\cap}^{\mathcal{T}} M : A$, 584	$\cdot_{A,B}$, 76
$\Gamma \vdash_{\cap}^{\mathcal{S}} M : A$, 478	\cdot_Z , 522
$\Gamma \vdash_{\cap}^{\mathcal{T}} M : A$, 473	$\cdot_{\mathcal{P}\omega}$, 567
$\Gamma \vdash_{\lambda \rightarrow} M : A$, 8	$e \mapsto e'$, 505
Γ_1^P , 595	$f \cdot_{A,B} x$, 76
\vdash_A , 292	$f \cdot x$, 76
\vdash , 473, 598	w_n , 554
$\vdash_{\cap}^{\mathcal{T}}$, 473	$Z_{F,G}(a,b)$, 526
\vdash_{\cap} , 473	$\mathbb{K}^{\mathcal{T}}$, 545
$\vdash_{\lambda \underline{A}}$, 292	$\mathcal{G}_{\Delta}(\mathcal{M})$, 100
4. Models	$\mathcal{R}(\mathcal{D})$, 526
A. general	$\text{Th}(\mathcal{M})$, 78
$(\mathcal{M}_{\mathcal{E}}, \cdot)$, 87	$i_0^{\mathcal{T}}$, 559
HEO_B , 242	i_n , 550
HRO_B , 242	$j_0^{\mathcal{T}}$, 559
d^n , 550	j_n , 550
d_n , 550	m^R , 513
\mathcal{C}^n , 207	n^L , 513
$\mathcal{D}_0^{\mathcal{T}}$, 559	$\sqcup_{\mathcal{X}}$, 480
\mathcal{D}_{∞}^t , 553	\sqcup_X , 550
\mathcal{D}_{n+1} , 550	$\mathcal{M}(\langle \mathcal{D}, Z \rangle)$, 520
$\mathcal{F}^{\mathcal{T}}$, 480, 540, 541	C. classes
\mathcal{M}_n , 67	$\Theta_{\mathcal{D}_Z}(x, y)$, 522
\mathcal{M}_{\min} , 140	\mathcal{D}_{∞} , 550
$\mathcal{Q}(\mathcal{S})$, 520	\mathcal{F} , 457
\mathcal{M}_X , 67	$\mathcal{F}^{\mathcal{T}}$, 481
$\langle \mathcal{D}, \cdot_F, [[\]^{F,G}] \rangle$, 537	\mathcal{F}^{BCD} , 457
B. operations on	$\mathcal{K}(\mathcal{F}^{\mathcal{T}})$, 480
$\text{init}(\mathcal{T})$, 559	MSL , 477
$(X \Rightarrow Y)$, 458, 573, 578	MSL^U , 477
$F^{\mathcal{T}}$, 481	TS , 477
F_{∞} , 552	TS^U , 477
F_{Pm} , 568	$\text{TS}^{\mathbb{U}}$, 477
F_{Em} , 566	Pm , 568
$G^{\mathcal{T}}$, 481	PLS^s , 524
G_{∞} , 552	Em , 566
G_{Pm} , 568	D. relations

$(\mathbf{m} :) D \cong D'$, 539	$\rho[x := d]$, 77, 457, 536
$A =_k^* B$, 364	$\rho \models A$, 62
$E_1 \succsim E_2$, 78	$\rho \models \Gamma$, 62
$E_1 \simeq E_2$, 78	$k \Vdash_{\mathcal{K}} A$, 65
$\Gamma \models M : A$, 396, 459	$\llbracket M \rrbracket_{\rho}^{\mathcal{M}}$, 67, 77, 406
$\Gamma \models \rho$, 541	$\llbracket M \rrbracket_{\rho}^{\mathcal{M}}$, 77
$\Gamma \models_{\cap}^{\mathcal{T}} M : A$, 574, 592	$\llbracket \] \rrbracket^{\mathcal{D}}$, 536
$\Gamma \models_{\mathcal{A}} M : a$, 404	$\llbracket \] \rrbracket_{\rho}^D$, 457
$\Gamma \models_{\mathcal{A}, \mathcal{D}} M : a$, 404	$\llbracket \] \rrbracket_{\xi}$, 458
$\Gamma \models_{\mathcal{D}, \rho, h} M : a$, 404	$\llbracket \] \rrbracket^{\mathcal{F}^{\mathcal{T}}}$, 540
$\eta \models_k A \leqslant B$, 439	$\llbracket \] \rrbracket^{\mathcal{T}}$, 575
$\eta \models_k \mathcal{H}$, 439	$\llbracket \] \rrbracket_{\rho}$, 536
\leq , 504	$\beta\mathbf{N}$, 618
$\mathcal{H} \models A \leqslant B$, 439	$\mathcal{D}, \rho, \xi \models M : A$, 85
$\mathcal{H} \models_k A \leqslant B$, 439	$\mathcal{D}, \rho, \xi \models \Gamma$, 85
$\models_{\mathcal{D}, \rho, h} M : a$, 403	$\mathcal{D}, \rho \models M = N$, 84
$\models_{\mathcal{D}, \rho, h} \Gamma$, 403	$\mathcal{D}, \rho \models T$, 84
$\sqsubseteq_{\mathcal{D}}$, 504	$\mathcal{D} \models T$, 85
$\tau, \rho \models \Gamma$, 396	$\mathcal{K} \Vdash \Gamma$, 66
$\llbracket \] \rrbracket_{\xi}^{\mathcal{D}}$, 573	$\mathcal{K} \Vdash A$, 65
$\mathcal{A} \models \exists X.a = b \rightarrow X$, 306	$\mathcal{M}, \rho \models M = N$, 78
$\mathcal{A} \models \mathcal{E}$, 294	$\mathcal{M} \models M = N$, 78
$\mathcal{D}, \rho, \xi \models M : A$, 459, 574	$\text{Env}_{\mathcal{D}}$, 77, 536
$\mathcal{D}, \rho, \xi \models \Gamma$, 459, 574	ξ_{BCD}^1 , 579
$\mathcal{D} \models M = N$, 458, 536	ξ_{BCD}^2 , 579
$\mathcal{M} \equiv_A \mathcal{N}$, 79	$\xi^{\mathcal{T}}$, 575, 618
$\mathcal{M} \equiv \mathcal{N}$, 79	ξ_{CDV} , 579
$\mathcal{M} \times \mathcal{N}$, 81	ξ_{CDZ} , 579
$\mathcal{T}, \rho, \Gamma' \models \Gamma$, 592	ξ_{DHM} , 579
$\mathcal{T}, \rho, \Gamma \models M : A$, 592	ξ_{HL} , 579
E. interpretations in	ξ_{Park} , 618
$((-))_{\rho}^{\mathcal{M}}$, 82	ξ_{Scott} , 618
$T \models M = N$, 85	
$\llbracket M \rrbracket_{\rho}$, 592	
$\llbracket \] \rrbracket^{F,G}$, 537	
$\llbracket \] \rrbracket_{\xi^{\mathcal{T}}}$, 618	
$\llbracket \] \rrbracket^{\Lambda}$, 578	
$\Gamma \Vdash A$, 66	
$\Gamma \models M : A$, 85	
$\Vdash A$, 66	
$\llbracket M \rrbracket_{\rho} \downarrow$, 78	
$\llbracket M \rrbracket_{\rho} \uparrow$, 78	
ι , 106	
$\models A$, 62	
	5. Miscellaneous
	A. general
	(\Diamond) , 548
	B_{β} , 610
	$B_{v,k}$, 602
	C_1 , 603
	C_2 , 603
	$G^{\mathcal{A}}$, 610
	ID , 607
	P_T^G , 605
	Δ_{β} , 610
	Γ_G , 605

Γ_v , 605	L^A , 607
\equiv , xx	T^2 , 34
$\overset{\Delta}{\iff}$, xx	X^Y , 75
\triangleq , xx	\mathcal{F} , 200
\blacksquare , xx	$\mathcal{S}(\mathcal{D})$, 411
TI , 225	\mathcal{I} , 202
TI_α , 225	$\mathcal{K}(\mathcal{D})$, 405
TR , 225	\mathcal{L} , 202
TR_α , 225, 226	\mathcal{N} , 211
d_n , 407	\mathcal{R} , 202
$::=$, xx	$\mathcal{F}[\vec{x}]$, 200
\mathbb{N} , xx	$\mathcal{S}_{\mathcal{T}}$, 433
EQA , 598	\mathcal{T}_{BH} , 437
ETW , 598	$\text{form}(\text{PROP})$, 64
IHP , 598	D. relations
WTG , 598	$(X, R) \models R\text{-SN}$, 45
Tr_{fin} , 326	$(X, R) \models \text{SN}$, 45
Tr_{fin}^A , 326	$(X, R) \models \text{WN}$, 45
Tr_{inf} , 326	$M \in \text{HF}^0$, 231
Tr_{inf}^A , 326	$P \leq_T Q$, 152
Tr_{reg} , 327	$P \leq_m Q$, 152
Tr_{reg}^A , 327	$R(*_1, \dots, *_n)$, 92
$t \mid w$, 327	R_A^U , 95
B. operations	$T \Rightarrow T'$, 603
2_n , 71	$T \Rightarrow^k T'$, 603
$E.L$, 189	$T \Rightarrow^{C_i} T'$, 603
$E.R$, 189	\implies , 27
R^* , 92	\Rightarrow_T^* , 603
S^\wedge , 105	$\vdash_{\mathbb{T}/\approx}$, 296
$X \Rightarrow Y$, 403	\rightarrow_S , 46
$\#X$, 143	$\vdash_{\mathcal{T}}$, 433
$\exists R$, 95	$f(x) = y$, xx
\nexists , xx	$u \vdash_R s$, 152
$\sqcup X$, 404	$x \models R\text{-SN}$, 45
f^\natural , 301	$x \models R\text{-WN}$, 45
f^\sharp , 301	$x \models \text{SN}$, 45
$h(a_1 = a_2)$, 295	$x \models \text{WN}$, 45
$h(\mathcal{E})$, 295	$f(x)\downarrow$, xx
$h : \mathcal{A} \rightarrow \mathcal{B}$, 294	$f(x)\uparrow$, xx
$\llbracket \]_\rho$, 77	6. Categories
$\text{Cp}_n(f)$, 212	A. general
$\text{dom}(f)$, xx	ALG_a^s , 506
C. classes	ALG_a , 506
ARS , 45	CPO , 405
C_Δ^A , 610	GTS^U , 479

- LLS**, 524
- LS**, 515, 524
- MSL^{U} , 479
- MSL^{U} , 479
- NLS**, 515, 524
- NTS^{U} , 479
- $\text{TS}^{-\text{U}}$, 479
- TS^{U} , 479
- LS^s , 524
- LTS^{U} , 479
- LZS**, 518
- PLS^s , 524
- $\text{PTS}^{-\text{U}}$, 479
- PZS^s , 519
- ZS^s , 519
- ZS**, 518
- ALG**, 405
- B. functors**
- Cmp , 509
- F_{ZS} , 522, 523
- $\text{Flt}_{\text{NTS}^{\text{U}}}$, 515, 516
- \mathcal{R} , 526
- Cmp_s , 512
- Flt_{NLS} , 516, 516
- $\text{Flt}_{\text{TS}^{\text{U}}}$, 528
- Flt , 508

