ELSEVIER

Theoretical Computer Science 249 (2000) 81–161

Theoretical Computer Science

www.elsevier.com/locate/tcs

From computation to foundations via functions and application: The λ-calculus and its webbed models

Chantal Berline

CNRS et Université Paris 7, Equipe de Logique Mathématique (case 7012), 2, place Jussieu, 72251 Paris Cedex 05, France

Received April 1997; revised 18 January 1999

Abstract

Church's λ-calculus is an enthralling object of mathematical and logical study, born in 1930 as the mathematical theory of functions as rules, and invented for foundational purposes.

λ-calculus gave rise to the first (and the most elegant) mathematical definition of computable functions and inspired the main theorems of recursion theory. It came back on the scene in the 1960s with the development of programming theory. A capital contribution of λ-calculus to this subject is that it allows the mathematical expression and development of the Curry–Howard correspondence between proofs and programs, which generates deep and active research.

The first aim of this paper is to introduce λ-calculus to a mathematical audience with no previous knowledge of it. After giving a brief insight to the conceptual and practical importance of typed calculi we will concentrate on *untyped λ-calculus*, which is logic free, has the most powerful expressive power and can be more easily described.

In the second and main part of the paper we give an elementary, algebraic, and bottom-up presentation of its useful classes of models, which are powersets built from adequate "webs". We focus on two methods: completion of partial webs (for building models) and reducibility (for studying them).

In the third part we try to give evidence that the study of models is interesting per se. We survey or raise a lot of natural questions which arise when one tries to develop a model theory for untyped λ-calculus, in the sense of a general study of the relations between its models and its equational extensions, and we illustrate them with many recent results.

Finally, we give a sketchy presentation of Grue's map theory, which is a common foundation for Mathematics, Logic and Computer Science, based on λ-calculus and, hence, on the notion of function and application (instead of sets and membership). *MT* fulfills Church's original aim and its consistency can be proved by exhibiting webbed models for it.

Keywords: λ-calculus; Models of untyped λ-calculus; Webbed models; Equational extensions of λ-calculus; Map theory; Graph models

E-mail address: berline@logique.jussieu.fr (C. Berline).

0304-3975/00/$ - see front matter
PII: S0304-3975(00)00057-8

Contents

1. Introduction . . . 83
1.1. Plan of the paper . . . 85
2. Basic motivations and intuitions . . . 85
2.1. λ-calculus as a foundation for programming theory . . . 86
2.2. A word on the Curry–Howard correspondence . . . 86
2.3. But what is λ-calculus? . . . 87
2.4. λ-terms vs. integers . . . 88
2.5. Is λ-calculus algebraic? . . . 88
2.6. λ-calculus for general foundations . . . 89
2.7. Further readings . . . 90
3. λ-calculus . . . 91
3.1. λ-terms . . . 91
3.2. Substitutions . . . 92
3.3. β-reduction and β-congruence . . . 94
3.4. Left- and head-normalizations . . . 95
3.5. Church's definition of computable functions . . . 97
3.6. Solvable and unsolvable terms . . . 98
3.7. The axiomatization of β-congruence . . . 98
3.8. The consistency of λ-calculus . . . 99
3.9. *$\beta\eta$-reduction and $\beta\eta$-congruence* . . . 100
3.10. Equational theories and λ-theories . . . 100
3.11. Basic consistencies and inconsistencies . . . 101
3.12. Strategies and observational equivalences . . . 103
4. Models . . . 105
4.1. Introduction . . . 105
4.2. Applicative structures and reflexive models of λ-calculus . . . 106
4.3. Equational theories of models . . . 107
4.4. Ordered applicative structures and monotonicity . . . 108
4.5. Refining monotonicity . . . 108
4.6. The basic example: continuity over full powerset domains . . . 110
4.7. A basic tool: fixed point operators on p.o. sets . . . 111
5. Webbed models . . . 112
5.1. Introduction . . . 112
5.2. Intended applications of webbed models . . . 114
5.3. A paradigm: Engeler's model $\mathscr{E}$ . . . 114
5.3.1. Definition of $\mathscr{E}$ . . . 114
5.3.2. Applications of Engeler's model to λ-calculus . . . 116
5.3.3. Proving the head-normalization theorem via Engeler's model . . . 117
5.3.4. Proving the left-normalization theorem via Engeler's model . . . 119
5.4. Methods for constructing and studying webbed models . . . 120
5.5. Graph models . . . 121
5.6. Other models built from webs . . . 127
5.6.1. Common features . . . 127
5.6.2. K-models . . . 130
5.6.3. G-models . . . 137
5.6.4. pcs-models . . . 139
5.6.5. κ-pcs . . . 140
5.6.6. *H-models . . . 141
5.6.7. *Prime algebraic domains vs. algebraic domains . . . 142
5.6.8. *Scott's information systems (SIS) . . . 143
5.6.9. *Filter models . . . 143
5.6.10. *Related work . . . 144
5.7. Webbed models for foundations . . . 144

6. Recent results and open questions 147
6.1. Incompleteness of the usual semantics 147
6.2. Do there exist semantic models of λ_β, $\lambda_{\beta\eta}$, $\mathscr{H}$, $\lambda_{\beta\infty}$? 149
6.3. Global representativeness of the classes of models w.r.t. λ-theories 150
6.4. Can the shape of the model be influential? 152
6.5. Towards a general approximation theorem for comparing theories of models? 152
6.6. Orderability and orders 154
6.7. Correctness and completeness w.r.t. strategies 154
6.8. Easiness and forcing revisited 156
Acknowledgements 157
References 157

1. Introduction

Though the title could cover the work done for 60 years in λ-calculus, the present paper is by no means a general or historic survey. Not only do we deal mainly with *untyped* λ-calculus, but we have clearly favored its equational aspects, even if computational and logical motivations are presented. Concerning motivations we have focused on the programming with proofs paradigm, which we esteem, as do many people, to be of first importance for the mathematical, logical, and computer science communities.

Our personal interests and choices also show up in the presentation of models we give here, in the kind of problems we survey in Section 6, and in the fact that we have included a presentation of Grue's map theory even if the system is very recent and not that well known yet among λ-calculus practitioners. This system is indeed an impressive illustration of the expressive power of λ-calculus. It also allows us to show that we can go very far with the construction methods that are presented in this paper, since we can even build models which are, in some sense, equivalent to models of ZFC.

The paper evolved from the lecture I gave at the ICMAA Conference and which was hence intended for a general mathematical audience. This explains why it starts from zero. The paper can in particular also be read by set-theorists with no prior knowledge of λ-calculus.

The paper is also intended for students in λ-calculus, and even for researchers, since it is, to our knowledge, the first place where one can find a global study of models and of their equational properties. Furthermore, the presentation is more systematic, direct and simple than is usual in the area.

In the last section we survey a lot of questions, from long-standing problems to recent ones, illustrating them with the most recent results. Some of them should be accessible, especially those which arose as I was writing this paper.

Finally, the paper contains many pointers to the literature which make it useful as a starting point for a literature search.

To protect the naive reader from an excessive amount of information *stars* decorate the most technical remarks or sections. A natural point to stop reading could be the end of Section 5.4, or the end of Section 5.5 where the simplest class of webbed models is presented; he could also have a look at section 6.

A reader interested in set-theory should go to the end of Section 5.7, but could omit all starred sections, as well as Section 5.6.3.

On the other hand, a reader familiar with elementary λ-calculus should start from Section 3.10.

Untyped λ-calculus is the main concern of this paper, and from now on will be referred to simply as *λ-calculus*. *λ-calculus* is logic-free, nearly algebraic,[1] and is the common basis of all *typed λ-calculi*, where λ-calculus is limited by logical systems, which in turn allow a better control of its operational behaviors. We will not be concerned here with typed calculi, except for giving motivations and for some comments. One of the most important motivations at the moment is the *programming with proofs paradigm*, which is based on the *Curry–Howard correspondence* between proofs and programs, both viewed as λ-terms.

λ-calculus has more *expressive power* than any typed calculus, not being limited by specific rules. In particular, λ-calculus admits *unrestricted self-application* (say of a function to itself), and *unrestricted existence of fixed points*. Being logic-free, untyped λ-calculus can also be more directly described. As the counterpart of its huge expressive power λ-calculus has puzzling features, and a much more difficult and interesting model theory. For example the first model of λ-calculus (if one excepts the term model), namely Scott's $\mathscr{D}_\infty$ was built in 1969, more than 30 years after the consistency of λ-calculus was established.

As a matter of fact, before creating $\mathscr{D}_\infty$ Scott had settled the foundations of a general framework for such construction [103]. This framework, which has given rise to a lot of other models, is now called *Scott's continuous semantics* and it is the basis of all the other interesting "functional" semantics invented so far, namely G. Berry's stable semantics and the recent strongly stable semantics of A. Bucciarelli and T. Ehrhard.

Models are significant on their own, or for proving consistencies of extensions of λ-calculus (designed either with concrete, theoretical, or foundational purposes), or for studying the operational features of λ-calculus.

There are intrinsic difficulties in studying models of λ-calculus, but our view is that the lack of general methods is partly due to the use of overgeneral frameworks in the usual presentations of models, and to the fact that the few models whose theory is known, for example the most famous, $\mathscr{D}_\infty$ and P_ω, are still frequently presented under overcomplicated presentations, which render all computations heavy or/and specific.[2,3] Meanwhile, the most simple of all models, namely Engeler and Plotkin's model $\mathscr{E}$ is often simply ignored. On the other hand, a lot of models and families of models have now been produced, and the result of this is that the situation is rather confused, that few models have really been investigated, always individually, and worst of all, that a lot of λ-calculists are unfamiliar with them, or think that they are unusable.

[1] See Section 2.5 for a precise meaning.

[2] See the introduction of Section 5.

[3] A very recent illustration of this point is [37], which studies a particular property of Scott's $\mathscr{D}_\infty$ using the worst possible description of this model (for this purpose). This furthermore hides the real scope of the result.

The starting point of our program is to show that it can be a significant gain in some cases to work directly[4] with models. A few examples are given here but we think that this point of view deserves to be further developed.

The next aim is to get a global and clearer view of the meaningful subclasses of models which have already been isolated as significant, and to present at this rather general level the methods which have been used for individual models (like the reducibility method). This allows us to get a few transfer theorems (and we hope for better ones). Besides doing some necessary cleaning, the intention behind this work is to initiate a real model theory of λ-calculus, in the sense of a general study of the relations between its models and its equational extensions. Such a study is lacking today in spite of a number of involved and tricky, but not integrated, results.

1.1. Plan of the paper

Section 2 provides some basic intuitions and modern motivations behind the λ-calculus. Section 3 describes the syntax of λ-calculus, states the main syntactical theorems, and settles the terminology about equational theories which will be used in the sequel. In Section 4 we show how one was naturally led to model λ-calculus, and present the diverse semantics. In Section 5 we make a systematic presentation of the classes of models that we wish to promote and a systematic (partial) study of them via reducibility. We essentially follow a bottom-up approach, starting with the simplest of all models, namely Engeler's model $\mathscr{E}$ (also due to Plotkin), which we use to give concise proofs of the classical normalization theorems, and ending with a sketchy presentation of the model designed in [19] to model Map Theory. Two general sections are however inserted, as soon as we have met enough concrete examples: Section 5.4, where we discuss the methods known for building and studying models, and Section 5.6.1 which gives the common features of all webbed models. Finally, the connections with more traditional classes of models of the continuous semantics are established.

In Section 6 we survey recent results and open questions, mainly on the equational theories of models and on the representativeness of the diverse classes.

We hope that these questions will stimulate interest and research in models of (untyped) λ-calculus, and will in particular revive the interest for looking at non-computationally-driven problems in this area.

2. Basic motivations and intuitions

λ-calculus is a mathematical theory of functions "*viewed as rules*" (prescribing how to compute a function from its arguments). It issues from a logical system invented by Church in 1930 and which was intended to provide a foundation for computation, mathematics, and predicative reasoning [26, 27]. Church's original system was shown

[4] By "directly" we mean without a useless "detour" via a proof theoretic translation.

to be inconsistent by Kleene and Rosser in 1935 [75], but the purely mathematical kernel was consistent [28, 29]; it gave rise to the first, and most elegant, mathematical formalization of the intuitive notion of computable function (Section 3.5) and inspired the main theorems of recursion theory [74]. Church's thesis conjectures that every mathematical notion of computable function is equivalent to this one, and the various definitions of recursive functions which were given later on, by Gödel, Kleene, Wang, Türing, Peter, Ersov, and others, all confirmed this (cf. [88]).

2.1. λ-calculus as a foundation for programming theory

This simple and conceptual model of *computability* returned to the front of the scene in the 1960s with the development of Computer Science, under the impulse of Backus (cf. [5]) and Landin [83], and generated the family of *functional languages* (Lisp [McCarthy $\sim$ 1960], Haskell, Miranda, ML, Caml,...). In functional languages functions and functionals (i.e. functions of functions) may be passed as arguments to a program as easily as concrete data, which is not the case with imperative languages (Fortran, Pascal, C) (the other conceptual differences between imperative programming (founded by Von Neumann) and functional programming are clearly explained, e.g. in the first pages of [5]).

The other main conceptual contribution of λ-calculus to programming theory comes from the fact that its terms can be viewed as proofs as well as programs: λ-calculus allows the mathematical expression of the *Curry Howard* "*Isomorphism*", which is a very deep correspondence between *proof theory* (i.e. the mathematical analysis of mathematical reasoning), and concrete programming theory. The *programming with proofs paradigm* is a topic in active progress, and the present expansions of the Curry Howard Correspondence generate deep and beautiful exchanges of intuitions and problems between Computer Science, Proof Theory and Mathematics.

2.2. A word on the Curry–Howard correspondence

The Curry–Howard correspondence between proofs and programs is mainly expressed via typed λ-calculi. What is a typed-calculus? Well, just the fragment of λ-calculus which encodes the set of proofs derivable in some given logical system. In this logical viewpoint *type* is just a synonym for "formula", and "A is a type for the term t" is equivalent to "t encodes a proof of A in the given logic.

Typed λ-calculi are "enriched restrictions" of λ-calculus. "*Enriched*" because types are systematically assigned to λ-terms. "*Restrictions*" because, given a logic, the associated typed λ-calculus contains only those λ-terms which encode proofs in this logic.

Thus, the *expressive power* of a typed λ-calculus is limited by its logic, but the positive counterpart is that the computation can be *controlled* by the logic. This shows up globally since, most often, the restricted set of terms will then consist only of normalizable terms (i.e. terms whose computation ends). This also shows up locally, since the type of a term can become a very precise specification of what the term computes.

Functional languages are not necessarily typed (Lisp is untyped), and when they are typed then types are used mainly for insuring some compatibility checkings. This corresponds to the use of weak but decidable typing systems.

By contrast the "full" programming with proofs paradigm is based on very expressive logics, which enjoy the most interesting mathematical and computational properties. In particular, provided that the underlying logic is sufficiently powerful, say second-order predicate logic(s), it necessarily produces programs which do what you expect, and for general mathematical reasons [85, 77, p. 154].

2.3. But what is λ-calculus?

Intuitively, *any term of lambda-calculus is a function* where the notion of function used is a very general one (since every term can apply to every term, and in particular to itself) and has a computational (and intentional) flavor: once a reduction strategy has been chosen, each term represents an algorithm, i.e. a precise way to compute the result of applying this term to an argument. In other words, we have an intensional notion of function as opposed to the usual extensional notion of a function as a graph. Here *extensional* just means that the function is determined by the results it gives on all possible arguments, while *intensional* expresses that we take into consideration the way they are calculated.

The syntax of λ-terms is very simple: *λ-terms* are built using only variables, application, and the abstraction operator λ. *Application* is a binary operator or "constructor" (just so as not to use the word "function" once more), and λ is a binding operator. For any term t and variable x, $\lambda x.t$ is a term which can be thought of as "a name for the function which associates $t[x]$ to x"; here $t[x]$ just means that t possibly contains x as a free variable. Thus $\lambda x.x$ appears as the identity, $\lambda x.(x)x$ as self-application, and $\lambda x\lambda y\lambda z.y$ as a projection; all are "universal" in the sense that there is no specified domain or range.

In fact, there is no way of expressing domain or range in the syntax of untyped λ-calculus. These notions can be reintroduced at the level of interpretations, via "functional semantics" (a kind of semantics which includes all the models we will present later on), or, on an abstract syntactical level, via typing.

The calculus (called *β-reduction*) is defined by a very simple set of rewriting rules on the terms of the language. The heart of the calculus is *substitution*: the result of applying $\lambda x.t$ to u is the term $t[x:=u]$ obtained by replacing all free occurrences of x in t by u; it is important however to take care of avoiding variable clashes, as we will see. At this stage we note that, although every term has an intended functional behavior (it can be applied to arguments), only for *abstractions*, namely terms of the form $\lambda x.t$, does this intention gives rise to a possible computation.

Read through the Curry–Howard correspondence, β-reduction corresponds to (possibly) multiple steps in some *cut-elimination* process (in proofs), which is nothing less than the successive replacement of lemmas by their proofs, within a given proof.

Remark. The control given by the logic in a typed calculus amounts to a restriction of application, and in particular of self-application. The price is a loss of terms, and

hence of general tools, like the "fixed point operators" that we shall meet later on, while we gain global properties of the remaining terms, like normalization properties (i.e. all relevant computations end).

2.4. λ-terms vs. integers

Though λ-calculus is very elementary, its study is very difficult; in this respect, the term model Λ can be compared to the set of integers N. Like N, also, Λ is able to represent all finitary objects (or *data*) (integers, lists, words, trees, formulas), as well as all the computable functions on them. The major difference is that λ-calculus allows meaningful encodings, which not only respect the structure of objects but also the functional intuitions behind them, while the encodings in N largely depend on arbitrary choices. Finally, a single term may as well represent several concepts at a time, which is extremely valuable either when dealing with computational motivations or with foundational ones. This pluri-intentionality of terms shows up globally (terms may be considered as functions, as programs, as proofs, and also as classes (a traditional view which goes back to Frege (cf. [55, p. 3; 105])), or as sets [54] (cf. Section 2.6), and shows up locally: for different possible meanings of $\lambda x\lambda y.x$ see Section 3.1.

To summarize: owing to Church's original intentions, λ-calculus has a considerable expressive power and is able to represent functional ideas as well as concrete data. This is of definite interest for computation, and also strongly motivates the desire to achieve Church's foundational aim.

2.5. Is λ-calculus algebraic?

Untyped λ-calculus is more algebraic in spirit than the typed calculi, since no logic or proof theory has to be specified, and its syntax is simpler. However the λ-operator, and also substitutions, are at the core of the syntax and prevent a direct first-order formalization of the theory.

As a matter of fact it is possible to get rid of λ and to push substitution to its usual metalevel just by adjoining to application two constant symbols k and s satisfying $kxy = x$ and $sxyz = (xz)(yz)$ and two rather simple $\forall\exists$ axioms (see [77, p. 77]). We are then faced with Schönfinkel and Curry's "*combinatory logic*" (CL).

There is a syntactic translation between the two systems, and behind this, once more, a meaningful correspondence in proof theory: λ-calculus encodes proofs written in Gentzen's natural deduction system, while CL corresponds to Hilbert's deductive system.

Both systems are equivalent at the equational level [77, p. 87] but, though *CL* underlies a conversion whose basic rewriting rules are: $kuv \to u$ and $suvw \to (uv)(uw)$, the computational behavior of both systems is different [8, p. 155].

For *CL*, which is a first order theory, the notion of models is the standard one. But for λ-calculus one had to develop specific notions of models.

Finally, all the functional intuitions are lost in *CL*; for example the identity $\lambda x.x$ is translated by *skk*! This point largely justifies the effort which is required to enter in the world of λ-calculus, even if it looks less familiar at first sight.

2.6. λ-calculus for general foundations

We shall now explore another recent direction with respect to the role of λ-calculus for foundations.

The idea of taking as primitive the notions of function and application also goes back to G. Frege, the founder of Mathematical Logic (1879, [55, pp. 1–5]), and is also the basis of Schönfinkel's logical system (1924, [55, pp. 355–366]), whose mathematical part is CL.

Church's system, as well as the many systems proposed thereafter (at least the ones I know), relied on the following translation between λ-calculus and naive set theory; the key points of this (naive) correspondence are: set $\rightarrow$ function, membership $\rightarrow$ application (in the sense that $x \in y$ corresponds to "*yx is true*"), class-formation $\rightarrow$ abstraction (in the sense that $\{x/A \text{ is true}\}$ is denoted by $\lambda x.A$. Such a system is presented in [45]; others are mentioned in [44].

The view in Grue's system *Map Theory* (*MT*) that we present below differs radically, and is in fact dual: in *MT* the idea is that $x \in y$ if there is a u such that $x = yu$. We are interested in Grue's system since it is directly based on λ-calculus, on strong computational intuitions, and on functions. In particular *MT* does not inject at the level of language complicated concepts directly inspired by set theory. The language is simple, most of the axioms and rules are natural, even if the axiomatization has still to be matured (and we hope that webbed models like those built in [19] and sketched in Section 5.7, besides providing a clear consistency proof, will help in this). Another interest is that λ-calculus plays in this system its full computational role, and transfers it to set-theoretic constructions. This idea is developed in the introduction of [19] where, in particular, *MT* is compared with Flagg and Myhill's system [45].

As already mentioned, *MT* is designed as a common foundation for Mathematics, Logic and Computer Science. It is an equational extension of λ-calculus: its axioms are equations, its rules only permit to infer equations from equations, and they include the axioms and rules of λ-calculus (cf. Section 3.7). Like λ-calculus, *MT* underlies a rewriting system which is the core of computation, and is in fact essentially that of λ-calculus.

Grue's language endows λ-calculus with five constants $\top, \bot, if, \varepsilon$, and φ which all have at least two of the three following intuitive meanings: computational, logical and set theoretical. To begin with, $\top$ is intended for "truth", "termination of calculi", and "the empty set" and $\bot$ represents "indeterminacy" and "looping" (calculi which never end and give no information in the meantime). In the computational world *if* represents Mc Carthy's conditional (*if ... then ... else*) and also the pairing construct w.r.t. finitary data; in the set theoretical world *if* is the pair set operator, and in the logical world *if*, together with $\top$ and $\bot$, allows the interpretation of propositional calculus. Now φ

can be viewed as the characteristic function of the class Φ of all maps which represent sets. Finally, ε is a strict version of *Hilbert's operator*, which is a choice operator already present among Church's constants. Modulo a correct axiomatization, ε allows one to define the usual quantifiers (ranging over Φ) and gives furthermore the Axiom of Choice for free when maps are interpreted as sets; and in a computational world ε (as well as the other quantifiers it generates) can be interpreted in terms of an infinite number of parallel computations.

As a further example let us mention that $\lambda x.\top$ represents $\{\emptyset\}$ in the set theoretical world, is the canonical representative for "false" in the logical world, and, in the computational world, is the "black box" which applied to every argument replies $\top$ without even looking at the argument.

It is worth noting that in *MT* the representation of integers and truth values is based upon $\top$ and *if*, and not on Church's integers or booleans.

An intuitive description of the axiomatization of *MT* and of its "canonical models" is given in Section 5.7.

There exist syntactical translations between *MT* and (*ZFC*+ Predicate calculus) [54], which are highly non-trivial, since both systems are really designed on different basis. There exists also a semantic correspondence between canonical models of *MT* and universes of Set Theory, which is described in [19, Appendix A] and sketched at the end of Section 5.7.

2.7. Further readings

This introduction can be complemented with the references below, which are of course not exhaustive.

Let us first mention that a very recent issue of the Bulletin of Symbolic Logic is dedicated to Alonzo Church. The reader will find there in particular an interesting and historical *survey* of Barendregt on the impact of λ-calculus in logic and computer science [9].

Classical motivations (up to the 1980s) *behind λ-calculus and its continuous semantics* can be found in Scott's papers. [104, 108, 107] and in the introductory parts of Barendregt [8], and Stoy [111]. Odifreddi's introduction [89, pp. 4–13] covers also more recent programming motivations. For an interesting account of the first years of λ-calculus and of recursive function theory see Kleene's [74].

Concerning the links with programming theory: Krivine's brief survey [78], and Backus [5], which is a 20-year old call for functional programming, are intended for a general audience. The Curry–Howard correspondence for intuitionist second-order predicative logic is well explained in [81]. For the classical case see [80]. For the expressiveness of intuitionist second-order typed-calculi, for their interest from mathematical, computational and metamathematical points of view, and for the connections with existing programming languages (at that moment) see [46]. The mathematical justifications of the (intuitionist) programming with proofs paradigm can be found (with no comments) in Krivine's book below.

Books: We will essentially refer here to Barendregt's book [8] and to Krivine's book [76, 77]. In particular, it is worth noting that the *systematic* study of webbed models of (untyped) λ-calculus (in the sense of the present paper) was initiated in [76]. *Other classical text-books* are Stoy [111], and Hindley–Seldin [60]. For the connections of λ-calculus with *recursion theory*, and of continuity with *computability*, see Odifreddi's book [88]. Girard's proofs and types [50] is a *proof-theoretic* approach to typed calculi.

3. λ-calculus

Throughout the paper, $\equiv$ will denote definitional identity.

3.1. λ-terms

We start from an infinite set of variables V, a binary function symbol (which will be omitted), and an "abstraction operator" λ.

Since it is more convenient for several purposes, and also for compatibility reasons, we will adopt Krivine's notation for application, which is inverse to the usual one: t applied to t' will be denoted $(t)t'$!

The set T of λ-terms is defined as the smallest set which contains V and is closed under application and under the abstraction operator: if t and t' are terms and x is a variable, then $(t)t'$ and $\lambda x.t$ are terms.

λ is a binding operator (in this respect the status of $\lambda x.$ is similar to that of $\forall x$ or $\delta/\delta x$). As a matter of fact, one very soon works "up to α-renaming", that is we identify two terms which differ only by a (correct) renaming of bounded variables.

For the formal definition of *free* and *bound* variables, *subterms* and of α-*renaming*, we refer to [77], but examples are given below. *Closed terms*, also called *combinators*, are those terms without free variables.

Notations. Λ will denote the set of λ-terms up to α-renaming. Λ^0 the set of closed λ-terms, up to α-renaming. $\Lambda^{<\omega}$ the set of finite sequences of elements of Λ. $t, u, v \ldots$ range over all terms, and $f, x, y \ldots$ over variables. $\bar{t}$, $\bar{u}, \ldots$ range over finite sequences of elements of Λ. $FV(t)$ will denote the set of all free variables of t.

Conventions. Except when reasoning by induction on the structure or length of terms (or when giving a term to a computer !), one takes the freedom of adding pairs of parentheses when necessary for readability, and one uses the following *abbreviations*:

$\lambda x \lambda y.t$ for $\lambda x.\lambda y.t.$

$\lambda \bar{x}.t$ for $\lambda x_1 \ldots \lambda x_n.t$, with $n = l(\bar{x}) \geqslant 0.$

$u\bar{v}$ for $(..((u)v_1)v_2...)v_n$, with $n = l(\bar{v}) \geqslant 0.$

For $n = 0$, $\lambda \bar{x}.t$ and $u\bar{v}$ denote respectively t and u.

Example 1 (*Free and bound variables*). In $(\lambda x.x)x$ the first occurrence of x is bounded, and the second is free. Also, this term is α-equivalent to, and hence identified with, $(\lambda y.y)x$.

$\lambda x.(\lambda x.x)x$ is closed: the first occurrence of x is bounded by the second λ, and conversely. But we will in general avoid such an unreadable choice of variable names.

Example 2 (*Church integers*). $\hat{n} \equiv \lambda f.\lambda x.\underbrace{(f)(f)\ldots(f)}_{n \text{ times}}x$ which represents the integer n as the nth *iterator* which, given two arguments, applies n-times the first to the second.

Example 3 (*The Booleans True and False*). $T \equiv \lambda x.\lambda y.x$ and $F \equiv \lambda x.\lambda y.y$.

Example 4. $I \equiv \lambda x.x$, $K \equiv \lambda x.\lambda y.x$ and $S \equiv \lambda x.\lambda y.\lambda z.((x)z)(y)z$. K and S are basic combinators, which are furthermore essential for the translation of combinatory logic into λ-calculus. Using the abbreviations above, we have $S = \lambda x\lambda y\lambda z.(xz)(yz)$.

Example 5. $\Omega \equiv \delta\delta$ where $\delta \equiv \lambda x.xx$ is self-application. Ω is frequently used for representing looping (any computation which lasts indefinitely, without even giving partial results) and indeterminacy. This will be justified later on.

Example 6 (*Curry's fixed point operator*). $Y \equiv \lambda f.(\lambda x.f(xx))\lambda x.f(xx)$.

It will be justified later on that Y is indeed a fixed point operator. Fixed point combinators are crucial for representing functions defined by recursive equations within λ-calculus. [5]

Example 7 (λI-*terms*). A λI-*term* is a term which, like I (and S, $\hat{n}$ for $n \neq 0$, Ω, Y), but unlike K, has no subterm of the shape $\lambda y.u$ with $y \notin FV(u)$.

λI-terms have a nice normalization property (Proposition 25) and were favored by Church, in spite of complications due to the fact that F and $\hat{0}$ are not λI-terms.

The simple example of F above illustrates how several meanings can be given to the same term, since $\lambda x.\lambda y.y$ represents simultaneously a second projection, the Church integer $\hat{0}$, and the boolean value $F(alse)$; it also represents a proof of $A \Rightarrow (B \Rightarrow B)$, but that is another (typed!) story. This does not come from an arbitrary encoding of the different notions: the operational behavior of $\lambda x.\lambda y.y$ will indeed be compatible with all these interpretations.

3.2. Substitutions

The treatment of substitution is not the most exciting part in the presentation of λ-calculus but, since it is the core of the calculus, one has to put up with it.

[5] Curry called Y the *paradoxical combinator* since Y allows a standard translation of Russell's Paradox into a term of λ-calculus (where it no longer gives rise to a paradox, essentially because translations of sentences into λ-calculus are not necessarily 2-valued).

The "variable clashes" phenomenon which will be pointed out below for λ-calculus also occurs in usual mathematics, however it is never given an explicit treatment, since the real interest is elsewhere. Let us give an example:

From the intuitive meaning of the formula (F): $\forall x \exists y((x+y)=0)$ it should be clear that if (F) is true then all its instances $(Ft): \exists y((t+y)=0)$, where t is any term of the language, should be also true; however this is not the case: we cannot substitute y for x since (F) clearly does not imply $\exists y((y+y)=0)$. What happens is that the new "y" has been captured by $\exists y$. Since y should indeed be substitutable, and since the y bound by $\exists$ has no real existence, we change its name in (F) and work, when treating the "y" case, with the "α-equivalent" formula $\forall x \exists z\,((x+z)=0)$.

In usual mathematics, as well indeed as in CL, this treatment, that is the correct use of the first-order predicate calculus rules, is done implicitly. But when studying λ-calculus we cannot skirt this point, since it is the real heart of our topic; thus we are led to consider two notions of substitution.

Definition 8 (*Simple* (*or contextual*) *substitution*). $u\langle x := v\rangle$ is obtained by replacing all the *free* occurrences of x in u by v.

Simple substitution is useful for expressing congruences and more generally for "putting terms into contexts", which happens to be necessary when studying their full operational behavior. However, this substitution cannot be the base of λ-computation, since we wish to keep the meaning of a term during its computation, whereas in $u\langle x := v\rangle$ the free variables of v may be captured by some λ's of u (*variable clashes*), which is a bad feature from this point of view:

Example 9. $(\lambda y.x)\ \langle x := y\rangle \equiv \lambda y.\, y$, where a "constant function" becomes the identity.

Thus, for λ-computation, we have to take care of working with the following notion.

Definition 10. *Correct substitution* (no variable clashes) is defined as follows: $u[x := v] \equiv u'\langle x := v\rangle$ where u' is any term obtained by renaming the bound variables of u to avoid the free variables of v. The justification that this is a good definition (up to α-renaming) is worked out in [77].

Example 11. $(\lambda y.x)\ [x := y] \equiv (\lambda z.x)\langle x := y\rangle \equiv \lambda z.y$.

Of course, closed terms are not affected by substitutions: if t is closed then $t = t[x := v] = t\langle x := v\rangle$.

Definition 12. A relation on Λ is *contextual* if it is compatible with application and λ-abstraction, or equivalently with simple substitution. A *congruence* is a contextual equivalence relation.

3.3. β-reduction and β-congruence

Definition 13. A term of the form $(\lambda x.u)v,\ u,v \in \Lambda$ is called *a redex*, and $u[x := v]$, the term obtained by correct substitution of v to all free occurrences of x in u, is called *its reduct*.

This "reduct" can be much longer than the redex itself, since x may have several free occurrences in u. A term may contain no, one, or several redexes as subterms. And these redexes may be nested, like in $I(I)K \equiv (\lambda x.x)((\lambda y.y)K)$, or disjoint, like in $IK(I)K \equiv ((\lambda x.x)K)((\lambda y.y)K)$.

One step of calculus inside a term t consists in replacing an arbitrary redex by its reduct (a choice has to be made). One writes $t \to_{\beta_0} t'$ if t' is obtained from t by a one-step reduction. A *reduction sequence* is a sequence of one-step reductions, for example: $IK(I)K \to_{\beta_0} K(I)K \to_{\beta_0} KK$.

Definition 14. *β-reduction* is the reflexive and transitive closure of $\to_{\beta_0}$, and is denoted by $\to_\beta$. It is a contextual relation because $\to_{\beta_0}$ is.

Definition 15. *β-congruence* or *β-equivalence*, denoted by $=_\beta$, is the symmetric and transitive closure of $\to_\beta$, and is also the smallest equivalence relation on Λ which contains $\to_{\beta_0}$. It is a *congruence* over Λ.

Example 16 (*Curry's fixed point combinator*). Let $Y \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ then, for any term t, $Yt =_\beta t(Yt)$[6] hence, modulo $=_\beta$, any t has a fixed point in Λ, namely Yt. Note also that $\Omega =_\beta YI$.

Definition 17. A term is *normal* if it contains no redex, *normalizable* if it can be reduced to a normal term, and *strongly normalizable* if there is no infinite reduction sequence starting from t. A *normalizing sequence* is a reduction sequence ending with a normal term.

One of the important properties of β-conversion is the *Church–Rosser property* (*CR*), also called the *confluence property*:

Theorem 18 (CR). *If t β-reduces to t and to t', then there is a common t'' to which t and t' β-reduce.*

Two more or less immediate consequences of *CR* are:

Corollary 19. *If t is normalizable, then all its normalizing sequences end on the same normal term.*

Corollary 20. *Two terms are β-equivalent iff they have a common β-reduct.*

[6] Hint: try first to reduce Yt.

3.4. Left- and head-normalizations

It is easy to prove that the set N of *normal terms* is the smallest set of terms which contains variables, and is such that $\lambda\vec{x}.yt_1\dots t_n$ is in N whenever all the t_i's are in N (for $\vec{x}$ and y any variables, $n\geqslant 0$, $l(\vec{x})\geqslant 0$). A more general class of terms is that of *head-normal terms.*

Definition 21. A term is *head-normal*[7] if it is of the form $\lambda\vec{x}.y\vec{t}$ (with no condition on the t_i's).

It is very easy to prove, by induction on the structure (or length) of terms, that any term t is either head-normal or has the form $\lambda\vec{x}.R\vec{t}$, where R is a redex; in this case R is called the *head redex* of t.

One step of head-reduction consists in reducing the head redex of the term (if any). Then, $t\to_h t'$ means that one can reach t' from t by a succession of one-step head reductions.

The *left reduction strategy* consists in always reducing the leftmost redex of a (non-normal) term, that is the redex (if there is one) which has the leftmost "(λ". The *head reduction strategy* always reduces the head-redex of a term (if any), and stops when there is none. Since the head redex of a term (if any) is its leftmost redex, the maximal sequence of the successive left reducts of a term begins with the maximal sequence of its successive head reducts (and coincides with it if one never reaches a head-normal term).

Definition 22. A term is *left-normalizable* (resp. *head-normalizable*) if the left-reduction sequence (resp. the head-sequence) is finite.

Obviously, left-normalizable terms are normalizable and are head-normalizable.

Example 23. $\Omega\equiv\delta\delta$, with $\delta\equiv\lambda x.xx$ is the most famous example of a non-head-normalizable term (and hence non-left-normalizable), indeed

$$\Omega\equiv\delta\delta\to_h\delta\delta\to_h\delta\delta\to_h\cdots.$$

$F\Omega$ is normalizable (and reduces to I) but is not strongly normalizable, $x\Omega$ is head normal but not left-normalizable. YK and $K^\infty\equiv(\lambda x.K(xx))(\lambda x.K(xx))$ are other interesting examples of non-head-normalizable closed terms: this time the head-reduction produces infinitely many head λ's, since

$$YK\to_h K^\infty\to_h\lambda x_1.K^\infty\to_h\lambda x_1\lambda x_2.K^\infty\to_h\cdots.$$

The relationship between the diverse normalizations is expressed by the two propositions below:

[7] For a reason I do not understand these terms are generally called "*in head normal form*".

Proposition 24. *Completeness of the left-reduction strategy*: *normalizable* $\Longleftrightarrow$ *left-normalizable*($\Longrightarrow$ *head-normalizable*).

The only non-trivial implication is the first left-to-right implication, which can be understood as a completeness property of the left-reduction w.r.t normalization. A semantic proof will be given in Section 5.3.4 and a variant in Section 5.6.2.

Proposition 25. *For* λI*-terms*: *normalizable* $\Longleftrightarrow$ *strongly normalizable.*

Left reduction has the best mathematical properties, to begin with the completeness result above, but it is not very efficient, since it multiplies the computation of arguments. For example, the left reduction of δv, with $\delta = \lambda x.xx$ and $v \equiv Ix$, will have to calculate v twice.

Call-by-name, *call-by-value*. Left-reduction is also called the "*call-by-name strategy*", by contrast to the "*call-by-value strategy*" [94], which always computes an argument before consuming it. In this latter case only those redexes $(\lambda x.t)u$ where u belongs to the subset of terms considered as "values" are reducible. The call-by-value strategy can be simulated within left reduction [94], and a more recent result shows that this can be done in a meaningful and efficient way by means of closed terms, the "*storage operators*", which also happen to play a crucial role in the extension of the Curry–Howard correspondence to classical logic [79].

Laziness. Each strategy has a *lazy* version where one stops computation as soon as one reaches an abstraction (the idea being that it is not worth computing a function which will never be applied to an argument). Laziness is one of the features of most concrete programming functional languages, and is also part of the notion of reduction underlying Map theory. The following useful definition makes particular sense when dealing with lazy versions of λ-calculus.

Definition 26. *The functionality order* $o(t)$ of a term t is the largest $n \geqslant 0$ such that t β-reduces to some $\lambda\vec{x}.u$ with $l(\vec{x}) = n$, if such a largest integer exists, otherwise $o(t) \equiv \infty$.

For example, $o(II) = 1$, $o(K) = 2$, $o(K^\infty) = \infty$ and, finally, for t closed, $o(t) = 0$ implies t unsolvable.

Comments on the operational behavior of head- and left-reductions. Suppose that the term t head-reduces to a head-normal term $\lambda\vec{x}.y\vec{t}$. Then either this term is normal, and we have finished, or the left reduction of t continues inside the subterms t_i'*s* (from left to right), and, whatever will happen next, all the successive reducts will be of the form $\lambda\vec{x}.y\vec{u}$ with $l(\vec{t}) = l(\vec{u})$ $(= n)$; thus at the end of the head-reduction of t we have got as a partial result the invariant $(\lambda\vec{x}.y, n)$; as the left reduction of t keeps on we will maybe get further such partial informations, that we can gather. If the term is (left-) normalizable, then a complete result will be reached at the hand: the normal form of t. At the other end, if t is not even head-normalizable, then the head or left reduction

of t will continue indefinitely, without ever giving any information on t.[8] This is the reason why *non-head-normalizable terms* are considered as representing *looping*, that is: any calculus which lasts indefinitely without giving any useful information back. For the same reason they are also considered, in different contexts, as representing *indeterminacy*, or *total lack of information* (though one can still be more drastic).

Later on, we will give an *algebraic characterization* of the head-normalizable terms as *solvable terms*. The equivalence between the operational and the algebraic definitions is not at all trivial, in one direction, and the most concise way I know of presenting the proof is done via a third characterization: both classes of terms contain exactly those terms which can be given a value different from "bottom" in Engeler's model (Section 5.3.3). Variations of the proof give the completeness of left-normalization, and other normalization results.

3.5. Church's definition of computable functions

Definition 27. A partial function $f : N \to N$ is *λ-representable* (or *λ-computable*) if there exists $t \in \Lambda$ such that

$t\hat{n} \to_\beta \widehat{f(n)}$ if $f(n)$ is defined,
$t\hat{n}$ is not normalizable otherwise

Church's definition is very elegant. The only arbitrary choice here is the way we chose to represent the integers (here by Church's integers), and it corresponds to a precise understanding of integers: as iterators. Other meaningful choices can be made, which have a different computational behavior.[9]

**Variations of the definition.* We mention different ways (two operational and one algebraic) of giving an equivalent definition. Firstly, because of the completeness of left-normalization and the fact that Church's integers $\hat{n}$ are normal terms, $\to_\beta$ can be strengthened to "*left−reducts to*..." and "not normalizable" to "*the left-normalization procedure never ends*", which gives the full computational flavor. Secondly, an (a priori) stronger definition asks for "not head-normalizable" instead of "not normalizable" (Barendregt). Finally, $\to_\beta$ can obviously be replaced by $=_\beta$ (because of Corollary 20).

Church's Thesis states that this definition is equivalent to any other mathematical definition of a computable function.

All existing mathematical formalizations of the notion of computable (or recursive) functions, no matter how different they appear at first sight, can be proved to be equivalent (and equivalent to this one !). For an enlightening presentation of the various possible definitions and of their equivalences, as well as for interesting intensional extensions of Church's Thesis, see [88].

[8] Except maybe for a lower bound on its functionality order.

[9] For example, there is no λ-term which computes the predecessor of a Church integer *in a constant number of reduction steps* [91]. There are other representations of integers which do not have this drawback (but have others) [90].

A few basic examples. The *addition* can be represented by $\lambda y.\lambda z.\lambda f.\lambda x.(yf)(zfx)$; the *multiplication* by $\lambda y.\lambda z.\lambda f.\lambda x.(y(zf))x$ or $\lambda y.\lambda z.\lambda f.y(zf)$; and *exponentiation* $(m,n)\mapsto m^n$ by $\lambda y.\lambda z.zy$. It is less direct to produce a term for the predecessor.

3.6. Solvable and unsolvable terms

We now turn to the algebraic view of head-normalizable terms as *solvable terms*. That both notions are equivalent is by no way obvious. But once the result is proved we have the fruitful option of switching between an algebraic and an operational view of the same objects.

Definition 28. A *closed* term t is *solvable* if there is a finite sequence $\bar{u}\in\Lambda^{<\omega}$ such that $t\bar{u}=_\beta I$. More generally, a term t is *solvable* if some (or any) of its closures $\lambda\bar{x}.t$ is solvable (*closure* means here that $\bar{x}$ contains all the free variables of t).

Theorem 29. *solvable* $\Longleftrightarrow$ *head-normalizable.*

As a consequence, Ω and K^∞ are unsolvable.

Another trivial consequence is that the class of head-normalizable terms is closed under β-equivalence (since the class of solvable terms is, obviously).

Proof of the theorem. $\Rightarrow$ will be proved in Section 5.3.3.

$\Leftarrow$: It is clearly enough to check it for closed head-normal terms. Now, if $t=\lambda x_1\ldots x_m.yu_1\ldots u_n$ is such a term, with $m,n\geqslant 0$ and $y=x_j$, then $tv_1\ldots v_m\rightarrow_\beta I$, for $v_j=\lambda z_1\ldots z_n.I$ and any choice of the other v_i's.

Remark 1. Although this requires a small familiarity with λ-calculus, it is not difficult to prove the completeness of left-reduction (Theorem 24) as a corollary of Theorem 29. A sketch of the proof is as follows: first notice that all normal terms, and hence all normalizable terms, are solvable; hence they are head-normalizable; afterwards the proof goes by induction on the length of the normal term v such that $t\rightarrow_\beta v$ (hint: factorize via the head-normal form t_h of t).

3.7. The axiomatization of β-congruence

Suppose we are interested in developing tools for reasoning about equations $t=u$ between terms. Since λ-terms are *not* first-order terms, we cannot use predicate calculus as the formal way of reasoning about these equations, and have to propose an alternative. Since furthermore we want to stay in a world of equations it will be "*equational reasoning*" in the following sense. For axiomatizing an (equational) theory, which is a set E of equations between terms, we specify a (recursive) subset of E, the *axioms*, and a recursive set of inference rules which allow to infer from the axioms all the equations of E, and nothing else.

A *rule* has the following form: $e_1;\ldots;e_n \vdash e$ where e and the e_i's are equations and $n \geqslant 0$ (using $n=0$ allows us to include the axioms among the rules); The sign $\vdash$ is to be understood as follows: we are able to derive the equation on its right-hand side only if we are able to derive all the equations on its left-hand side. In the case of β-congruence it is nearly obvious that the axiomatization below works, where, as usual in this paper, the metavariables $t,u,\ldots$ range over terms, and x,y over variables:

1. (redex = reduct) $\vdash (\lambda x.u)\ v =_\beta u[x:=v]$
2. (reflexivity) $\vdash t =_\beta t$
3. (symmetry) $t =_\beta t' \vdash t' =_\beta t$
4. (transitivity) $t =_\beta t'; t' =_\beta t'' \vdash t =_\beta t''$
5. (app) $t =_\beta t'; u =_\beta u' \vdash tu =_\beta t'u'$
6. (abs) $t =_\beta t' \vdash \lambda x.t =_\beta \lambda x.t'$

Rules 5 and 6 express that $=_\beta$ commutes with the operators, and are equivalent to the single rule:

5-6. (congr) $t =_\beta t' \vdash u\langle x:=t\rangle =_\beta u\langle x:=t'\rangle$

Axiomatizations in the same style can be done more generally for any recursively presentable binary relation on Λ. For example an axiomatization of $\rightarrow_\beta$ can be given in the same style: just suppress the symmetry rule and replace $=_\beta$ with $\rightarrow_\beta$.

Remark 2. We could not replace the "proof-theoretic" sign $\vdash$ for logical inference by a classical implication, adding universal quantifiers where they seem to be needed, in order to present the system above as a classical $\forall\exists$ first-order theory. Rules 1 and 6 would clearly be problematic and one has first to get rid of the λ-operator and of substitution, in other words to turn to combinatory logic.

3.8. The consistency of λ-calculus

An immediate consequence of Corollary 20 is that the *term model of β-equality*, namely $\Lambda/=_\beta$, is infinite, since there are infinitely many distinct normal terms. This proves in particular that *λ-calculus is consistent*, in the sense that it is impossible to derive all equalities from the system above (or equivalently that β-congruence differs from $\Lambda\times\Lambda$).

This argument relies on *CR*, and hence is elementary and purely syntactical.

A more conceptual way for proving the consistency would be to exhibit a meaningful model, that is, here, a mathematical structure whose construction is based on non-operational intuitions, and in which one can interpret terms in such a way that two β-equivalent terms have the same interpretation (and not all terms are equated). As already mentioned the first model only came 30 years after the syntactical proof.

Remark 3. To prove the consistency of λ-calculus, it is enough to prove that the booleans T and F are β-distinct (which follows from CR). Indeed, for any u,t we have: $Ftu =_\beta u$ and $Ttu =_\beta t$; so $F =_\beta T$ implies $u =_\beta t$.

3.9. $\beta\eta$-reduction and $\beta\eta$-congruence

Definition 30. *η-equivalence* is the smallest congruence on Λ containing all pairs $(\lambda x.tx, t)$ with x not free in t. It underlies a notion of reduction called *η-reduction* (where $\lambda x.tx \to_\eta t$).

η-equivalence plays a secondary but pleasant and useful role in the theory of λ-calculus since it allows one to see each term as an abstraction. Both η-reduction and η-equivalence can be consistently added to λ-calculus. The resulting *$\beta\eta$-reduction* and *$\beta\eta$-congruence* share the main properties of their β-analogues, to begin with the Church–Rosser property. The new calculus is called *extensional λ-calculus* (or $\lambda\eta$-calculus). The semantic consistency proofs for the λ- and $\lambda\eta$-calculus were obtained simultaneously, since Scott's first model is indeed extensional.

To axiomatize $\beta\eta$-equivalence it is enough to add to the system given in Section 3.7 one of the two axioms below (and to replace $=_\beta$ by $=_{\beta\eta}$).
7. $\vdash \lambda x.tx =_{\beta\eta} t$ (for all t,x such that x is not free in t)
7′. $\vdash \varepsilon =_{\beta\eta} I$ (where $\varepsilon \equiv \lambda x\lambda y.xy$).

Proposition 31. *It is equivalent for a term to be β- or $\beta\eta$-normalizable.*

Proof. β-normalizable terms are $\beta\eta$-normalizable because, first, η-reduction does not create redexes and, second, η-reduction is always terminating since it decreases the length of terms. The converse can rather easily be proved syntactically ([8] or [77]). A semantic argument will be given in Section 5.6.2 (cf. Example 140). □

3.10. Equational theories and λ-theories

We will use the following terminology, and its obvious analogues relative to the $\lambda\eta$-calculus.

Definition 32. A *λ-congruence* is any congruence relation on Λ which extends $=_\beta$. Any set E of equations between terms of Λ generates a λ-congruence $=_E$, namely the smallest λ-congruence which contains all pairs (t,u) such that $t=u$ is in E.

Definition 33. A *λ-theory* T is a set of equations such that $\{(t,u) \mid t=u \in T\}$ is a λ-congruence.

Thus λ-congruences and λ-theories are essentially the same thing, either viewed as a binary relation or as a set of equations. More generally:

Definition 34. A *theory* will be in this paper any set of equations between terms of Λ which "is" an equivalence relation on Λ.

Example 35 (λ_β *and* $\lambda_{\beta\eta}$). The λ-theories corresponding to $=_\beta$ and $=_{\beta\eta}$ are traditionally called λ_β and $\lambda_{\beta\eta}$.

Definition 36. A theory is *sensible* if it equates all unsolvable terms, and is *semi-sensible* if it does not equate a solvable term with an unsolvable term.

The existence of (consistent) sensible theories and the fact that such theories are necessarily semi-sensible follows from Proposition 45 and Remark 4 below.

Some λ-theories are axiomatizable (in the sense of Section 3.7), most are not. As a matter of fact, no sensible λ-theory is axiomatizable; on the other hand axiomatizable λ-theories are dense (for $\subseteq$) (cf. Barendregt's book [8, p. 431]).

Some λ-theories can appear as the symmetric closure of a binary reduction extending β-conversion, most do not (cf. [8, Chapters 16,17]).

λ-theories may appear either in a syntactical context (cf. Section 3.12), or in a semantic one, since the set of equations satisfied in a model is a λ-theory (cf. Section 4.2).

Definition 37. A theory is *lazy* if it never equates two terms with different (functionality) orders. It is strongly lazy if it is lazy and, for all $n \in N \cup \{\infty\}$ it gives the same value to all closed unsolvables of order n.

Lazy theories are semi-sensible; this is left as an exercise (using Remark 4).

Definition 38. We say that a set E of equations *is consistent with λ-calculus* if the congruence that it generates is non-trivial. Using Remark 3 it is easy to see that this amounts to say that $T = F$ cannot be derived from the inference system obtained by adding the elements of E as new axioms to the system defining $=_\beta$ in Section 3.7.

Of course, the problem of the consistency of sets of equations in an extended language (with constants added), is also important, as well for practical applications ("sugared" programming languages), as for theoretical investigations, or for foundational purposes, but we restrict the scope here to "pure" λ-calculus.

3.11. Basic consistencies and inconsistencies

We give here some non-trivial examples of consistency and inconsistency results, which begin with a simple observation concerning *separable terms*.

Definition 39. Two closed terms t, u are *separable* if there is a $\vec{v}$ such that $t\vec{v} =_\beta T$ and $u\vec{v} =_\beta F$; more generally two terms t, u are separable if it is the case for some pair of closures $\lambda\vec{x}.t$ and $\lambda\vec{x}.u$.

Obviously:

Lemma 40. *It is inconsistent to equate two separable terms.*

Definition 41. Two head-normal terms are *similar* if, once written as $\lambda x_1 \dots x_n.xt_1 \dots t_m$ and $\lambda x_1 \dots x_{n'}.x't_1 \dots t_{m'}$ (i.e. we take the same x_i's for both, as long as possible), we have that x and x' are the same variable and $m - n = m' - n'$.

It is an easy exercise to show that non-similar head-normal terms are separable, so we have:

Corollary 42. *It is inconsistent to equate two non-similar head-normal terms.*

By contrast the following result, due to Böhm [22], is far from obvious.

Theorem 43 (Böhm). *Any two non-η-equivalent normal terms are separable.*

Corollary 44. *It is inconsistent to equate two non-η-equivalent normal terms.*

Proposition 45. *It is consistent to equate* (*simultaneously*) *all unsolvable terms.*

Proof of Proposition 45. We will prove in Section 5.3.3 that Engeler's model $\mathscr{E}$ equates all unsolvables and we will see that the method is general enough to apply (easily) to many models.

In fact, there exists 2^{ω} different sensible λ-theories ([8, p. 421] and Section 6.3).

Remark 4. It is however inconsistent to equate simultaneously all unsolvable terms to the same solvable term t. It is indeed easy to prove that $K^{\infty} = t$ is consistent iff t is unsolvable (hint: first try $K^{\infty} = I$).

Remark 5. It is inconsistent to equate all non-normalizable terms.

Proof. $\lambda x.F\Omega = \lambda x.T\Omega$ implies $KF\Omega = KT\Omega$, and hence implies $F = T$.

*Proposition 45 can be seen as a consequence of Theorem 47 below, which needs more work than will be presented here, and for which we need the following definition:

Definition 46. * The *Böhm tree* of a term t is a finitely branching labelled tree which is defined as follows: if t is not head-normalizable, then $BT(t)$ has just one node, labelled $\perp$, if $t \to_h \lambda\vec{x}.yt_1 \dots t_n$, with possibly $l(\vec{x}) = 0$ or $n = 0$, then $BT(t)$ has a root labelled $\lambda\vec{x}.y$ to which n-subtrees are attached, which are, from left to right, $BT(t_1), \dots, BT(t_n)$.

Let $BT \equiv \{t = u \mid BT(t) = BT(u)\}$. It is not difficult to prove that β-equivalent terms have the same Böhm tree (by induction on the depth of the nodes [8, p. 219]).

Theorem 47. * *BT is a consistent λ-theory.*

As a matter of fact $\mathscr{D}_{\infty}, P_{\omega}$, and $\mathscr{E}$ all satisfy BT. Furthermore $Th(\mathscr{E}) = Th(P_{\omega}) = BT$, hence BT is a λ-theory (and in particular it is contextual); proofs for P_{ω} and $\mathscr{D}_{\infty}$ are given in [61] or [112, 113].

Corollary 48. * *It is consistent to equate all fixed point operators.*

Proof. Suppose Z is a fixed point operator, i.e. a closed term such that $Zt =_\beta t(Zt)$ for all $t \in \Lambda$. Applying this to $t \equiv \lambda y.z$ we get $Z(\lambda y.z) =_\beta z$, hence $Z(\lambda y.z)$ left-normalizes to z (by CR plus the completeness of left-normalization), which is possible only if the head-normalization of Z reaches an abstraction. Hence there is W such that $Z \to_h \lambda x.W$. Now, from $Zx =_\beta x(Zx)$ we get $W =_\beta xW$. From this equation and from the fact that β-equivalent terms have the same tree it follows easily that $BT(W)$ consists of just one infinite branch, all nodes being labelled by x. Thus, $BT(Z)$ which is obtained from $BT(W)$ just by adding $\lambda x.$ in front of the label of the root, is (infinite and) independent of Z, and we can apply Theorem 47. □

The next proposition argues in favor of the "total undefinedness" of Ω.

Definition 49. A closed term is *easy* if it can be consistently equated to any other closed term t.

Easy terms are necessarily unsolvable (this is a direct consequence of Remark 4), but the converse is false: $\Omega_3 \equiv \delta_3\delta_3$, where $\delta_3 \equiv \lambda x.xx$ is unsolvable but is not an easy term: it is indeed nearly trivial to derive $\delta_3 = I$ from $\Omega_3 = I$ (hint: start by reducing Ω_3), which contradicts Corollary 42.

Proposition 50. *Ω is easy.*

Proof. The proposition was first proved syntactically by Jacopini [62] (cf. [8, p. 402]). A semantic proof was then given by Baeten and Boerbom [6] (cf. Section 5.5).

Many other interesting examples of consistent λ-theories are studied in [8, Chapters 16,17], most of the time syntactically. We will present more recent consistency results in Section 6, which all are obtained by building adequate webbed models. Also easiness will be revisited in Section 6.8.

3.12. Strategies and observational equivalences

We are interested here in theories which arise from syntactical considerations. They can be motivated as well by computational considerations as by theoretical ones.

These theories are generated by strategies, with possible variations on the meaning of "generated"; *strategy* has here a rather general meaning since total freedom (β-conversion) will be considered as a strategy.

A *strategy* S has two components: the first tells us what kind of reductions we are allowed to perform, and the second tells us which terms are considered as admissible results (or *values*). We write $t \to_S u$ if u is obtained from t by performing S, and $t \Downarrow_S$ if there is a value u such that $t \to_S u$.

Example 51.[10]

1. S^n: left-reduction and normal terms (*call-by-name* reduction)
2. S^h: head-reduction and head-normal terms (*head-reduction*)
3. S^l: head-reduction and abstractions (*lazy-head* reduction)
4. S^v: call-by-value reduction and abstractions (*lazy call-by-value* reduction)
5. S^0: β-reduction and closed terms
6. S^0_I: β-reduction and closed λI-terms

Notations. Given $t,t' \in \Lambda$, $\Lambda^0(t,t')$ will denote the set of *closing contexts* for both t,t', namely the set of those $u \in \Lambda$ such that both $u\langle x := t\rangle$ and $u\langle x := t'\rangle$ are closed.

Definition 52. The *contextual theories generated by* S are the congruences $T^{ctx}(S)$ and $T^{ctx,0}(S)$ defined by

$$T^{ctx}(S) \equiv \{t = t' \mid \forall u \ (u\langle x := t\rangle \Downarrow_S \Leftrightarrow u\langle x := t'\rangle \Downarrow_S)\},$$

$$T^{ctx,0}(S) \equiv \{t = t' \mid \forall u \in \Lambda^0(t,t') \ (u\langle x := t\rangle \Downarrow_S \Leftrightarrow u\langle x := t'\rangle \Downarrow_S)\}.$$

Comments. Let us consider closed terms as programs. Two terms are "equal" or "*operationally equivalent*" (relatively to S) if, whenever placed in the same *closing context* the resulting programs behave in the same way, w.r.t. convergence to some value (this often forces the value to be the same). This computational interpretation fits $T^{ctx,0}(S)$, and obvious variations lead to $T^{ctx}(S)$, which has to be considered in cases where, as for S^0 and S^0_I here, $T^{ctx,0}(S)$ is not relevant.[11]

Remark 6. If S is such that $t \Downarrow_S$ if and only if $\lambda x.t \Downarrow_S$, then $T^{ctx}(S) = T^{ctx,0}(S)$. This is the case for S^h and S^n.

Remark 7. Though contextual, $T^{ctx}(S)$ and $T^{ctx,0}(S)$ are not necessarily λ-theories, since they do not have to contain $=_\beta$; for example $T^{ctx}(S^h)$ and $T^{ctx}(S^n)$ are λ-theories[12] but $T^{ctx}(S^v)$ is not.

Remark 8 (*Laziness*). It is easy to check that if S is lazy then $T^{ctx}(S)$ and $T^{ctx,0}(S)$ are also lazy.

Let us consider now the case of S^h, which is especially relevant for us (the other examples will be used too, but only in Section 6.7). We claim:

Proposition 53. *$T^{ctx}(S^h)$ is the unique maximal contextual semi-sensible theory.*

[10] For a more comprehensive survey see the introduction of [57].

[11] However it is often the case in practice that, given S, there exists a S' very close to S and such that $T^{ctx}(S) = T^{ctx,0}(S')$.

[12] This is indeed equivalent to proving the head- or left-reduction theorem.

Proof. Let T be a contextual and semi-sensible theory and suppose $t = t' \in T$. Since T is contextual the equation $u\langle x := t\rangle = u\langle x := t'\rangle$ is also in T (for all u), and since T is semi-sensible this implies that, for all u, both $u\langle x := t\rangle$ and $u\langle x := t'\rangle$ are simultaneously solvable or unsolvable; which is exactly to say that $t = t' \in T^{ctx}(S^h)$ (using the head-normalization theorem (Theorem 29)).

Corollary 54. *$T^{ctx}(S^h)$ is the unique maximal sensible λ-theory.*

One of our interests for $T^{ctx}(S^h)$ here is that $T^{ctx}(S^h) = Th(\mathscr{D}_\infty)$ (only $\supseteq$ will be proved in the paper).

It is worth noting that there is an alternative characterization of $T^{ctx}(S^h)$ in terms of trees:

* $T^{ctx}(S^h) = \{t = t' \mid NT(t) = NT(t')\}$ where $NT(t)$ is the Nakajima tree of t, which is, roughly speaking, an infinite η-expansion of the Böhm tree of t (cf. [8, ex. 19.4.4]).

Lemma 55. * *$T^{ctx}(S^n) \subsetneq T^{ctx}(S^h)$ and $T^{ctx}(S^n)$ is semi-sensible.*

Proof. Suppose $t = t' \in T^{ctx}(S^n)$ and t is solvable. Then there are $\bar{x}$ and $\bar{v}$ such that $(\lambda\bar{x}.t)\bar{v} =_\beta I$. I is a normal term, hence $(\lambda\bar{x}.t)\bar{v}$ is normalizable (by CR). Thus $(\lambda\bar{x}.t')\bar{v}$ is normalizable, hence head-normalizable, hence solvable, which forces t' to be also solvable. This proof works more generally if we replace t and t' by $u\langle y := t\rangle$ and $u\langle y := t'\rangle$, hence $t = t' \in T^{ctx}(S^h)$. The inclusion is strict since it can be shown that the equation $J = I$, where J is the non-normalizable term $Y(\lambda x.\lambda y.\lambda z.y(x)z)$ is in $T^{ctx}(S^h)$.[13]

4. Models

4.1. Introduction

We are interested in finding models which, unlike the term model, allow conceptual views of the syntax and lead to a better understanding of β-equivalence and β-conversion. We would like to recover some of the intuitions which were behind the calculus, to begin with the functional intuition, and finally we would like to have (partial) mathematical characterizations of the notion of "computable function".

All the models presented here are in fact models of β-equivalence, in the sense that two β-congruent terms always get the same interpretation. They can however be used to get information on β-reduction, or on the term model itself (this will be developed in Section 5.2).

The present section presents the conceptual basis of Scott's continuous semantics, and its further variations. In particular, these basic ideas are behind all the models built in Section 5.

[13] See [36] for a direct and purely syntactical proof. Alternatively it can be observed that I and J have the same Nakajima tree. The most meaningful proof is however semantic.

4.2. Applicative structures and reflexive models of λ-calculus

Since all λ-terms can be used as arguments, and since abstractions are essentially names for functions, it is reasonable to take as model $\mathscr{M}$ of λ-calculus an applicative structure which encodes, in an injective way, a significant part of its set of functions.

Thus, we start from an *applicative structure* $(M,\cdot)$, which just means a set endowed with a binary function "·". This dot will often be omitted.

Definition 56. A function $f:M\to M$ is *representable in* M if there is some $d\in M$ such that $f(x)=dx$. We let $R(M)$ be the set of functions which are representable in M.

We are interested in injective encodings of $R(M)$ in M; this amounts to finding a function G mapping $R(M)$ into M and such that $A\circ G=id_{R(M)}$, where $A(d)$ is $x\mapsto dx$. Then, for obvious cardinality reasons $R(M)$ cannot be the set of all functions from M to M.

This leads us to the following provisional definition:

Definition 57 (*provisional, because too weak*). *A model of lambda-calculus* is a triple $\mathscr{M}\equiv(M,\cdot,G)$ such that G maps $R(M)$ into M, and such that $A\circ G=id_{R(M)}$, where $A(d)$ is $x\mapsto dx$.

The aim is now to interpret any λ-term t, in such a way that each closed t is interpreted by an element of M, and we will see very soon that the definition above has to be restricted, in order that everything works.

The interpretation goes by induction on the structure of t and can only be done relative to an environment ρ, which fixes the interpretation of the free variables of t (an alternative presentation is to work with closed terms with parameters in M).

Definition 58. An *environment* is a function $\rho:V\to M$. We let M^V be the set of all environments w.r.t. M. For $d\in M$ we denote by $\rho[x:=d]$ the environment ρ' which coincides with ρ, except on x, where ρ' takes the value d. This definition generalizes obviously to $\rho[\bar{x}:=\bar{d}]$, where we understand that all variables in $\bar{x}$ are distinct and that $l(\bar{d})=l(\bar{x})\geqslant 0$.

Definition 59 (*Interpretation of λ-terms*). At each step t one defines $|t|_\rho\in M$ for all possible ρ's:

$$|x|_\rho\equiv\rho(x);$$

$$|tt'|_\rho\equiv|t|_\rho|t'|_\rho;$$

$$|\lambda x.t|_\rho\equiv G(d\mapsto|t|_{\rho[x:=d]}).$$

It is very easy to check, then, by induction on t, that, given t, $|t|_\rho$ depends only on the value of ρ on the free variables of t. So we can adopt the *simplified notation*:

$$|t[\vec{x} := \vec{d}]| \text{ instead of } |t|_{\rho[\vec{x}:=\vec{d}]}, \text{ if } FV(t) \subseteq \vec{x}.^{14}$$

Definition 59 *is however correct only if one can ensure at each step that the latter function is in* $R(M)$, for this we need to prove by induction on t (as soon as $|t|_\rho$ has been defined, for all ρ's), that:

Proposition 60. *For any* $t \in \Lambda$, *for any* $x, \vec{y}, \vec{d}$ *such that* $FV(t) \subseteq x, \vec{y}$, *the function* $d \mapsto |t[x := d, \vec{y} := \vec{d}]|$ *is in* $R(M)$.

This will be true for all the models we consider in this paper (and can easily be checked by hand, in each case). More generally, this proposition can be proved exactly when (M, A, G) is a reflexive object of an ambient cartesian closed category (c.c.c.) "with enough points".[15] This will be the case for all our models since all the semantics we will be interested in correspond to such ccc's. If we admit this point about the general frames, then we can work freely with the weak definition of models, without having to check Proposition 60 in each case.[16]

Read globally, this proposition expresses the "*combinatorial completeness*" of models of λ-calculus, namely that any function which is definable, with the possible help of parameters from M, is representable in M.

Then, from $A \circ G = id$, it is easy to prove [8, p. 105]:

Proposition 61. *If* $t =_\beta t'$ *then, for any environment* ρ, $|t|_\rho = |t'|_\rho$.

4.3. Equational theories of models

Definition 62. For any $t, t' \in \Lambda$, $\mathscr{M}$ *satisfies* $t = t'$ if for any environment ρ we have $|t|_\rho = |t'|_\rho$. This is denoted by "$\mathscr{M} \models t = t'$" or "$t =_{\mathscr{M}} t'$".

Definition 63. The *equational theory of* $\mathscr{M}$, $Th(\mathscr{M})$, is by definition the set of equations which are satisfied by $\mathscr{M}$. If we consider the set $Th^0(\mathscr{M})$ of closed equations,[17] then both are (deductively) equivalent since from the very definition of the interpretation of abstraction, $t =_{\mathscr{M}} u$ iff $\lambda\vec{x}.t =_{\mathscr{M}} \lambda\vec{x}.u$. However, $Th(\mathscr{M})$ is a λ-theory while $Th^0(\mathscr{M})$ is not even a theory (in the sense of Definition 34).

Definition 64. A model is *extensional* if $\lambda_{\beta\eta} \subseteq Th(\mathscr{M})$. Two other equivalent characterizations are $\mathscr{M} \models \varepsilon = I$ or $G \circ A = id$.

[14] * There is no harm in taking a notation similar to that of correct substitution; actually, both are cases of the "*correct substitution on terms with parameters in the model*".

[15] For the definitions of a *c.c.c.* and of a *reflexive object* see [8, pp. 107–108].

[16] To be honest, introducing models directly via *c.c.c.*'s would have been cleaner. We chose the presentation above since it allows us to keep close to intuition.

[17] That is, equations between closed terms.

Definition 65. A model $\mathscr{M}$ is *sensible* (resp. *semi-sensible*, *lazy*, *strongly lazy*) if its equational theory is.

Definition 66. A model $\mathscr{M}$ is *fully abstract* for the strategy S if $Th(\mathscr{M}) = T^{ctx}(S)$. It is only *correct* or *adequate* if two terms which are operationally different w.r.t. S have distinct interpretations in the model, that is, if $Th(\mathscr{M}) \subseteq T^{ctx}(S)$. Similar definitions can be given w.r.t. $T^{ctx,0}(S)$.

Remark 9. If $\mathscr{M}$ is semi-sensible then $\mathscr{M}$ is adequate for S^h (Corollary 54).

4.4. Ordered applicative structures and monotonicity

We will in fact only be concerned with *partially ordered applicative structures* $(M, \sqsubseteq, \cdot)$. This means that $\sqsubseteq$ is a partial order and that application is monotone w.r.t. both of its arguments. This forces all representable functions to be monotone. It also forces the interpretation $|t|_\rho$ of any term t to be monotone w.r.t. ρ, where environments are ordered pointwise.

The monotonicity condition is justified by Scott's view of models as sets of sets of observations (or informations) and of computable functions and functionals as monotone functions over such sets [108]; see also [111, 84], cf. Section 4.5. It is also, indirectly, justified by the fact that it happens to be difficult to find unorderable models: the term model itself has been shown only very recently to be unorderable (Section 6.6).

Still for cardinality reasons, not all monotone functions can be representable in M. That it is possible to restrict monotonicity to suitable notions is the subject of the next subsection.

4.5. Refining monotonicity

The models are classified into "*semantics*" according to the nature of their representable functions. The next subsection gives a sketchy presentation of the most interesting semantics, which all are issued from Scott's *continuous semantics* [103]. The *stable semantics* [20, 21, 49] and the recent *strongly stable semantics* [25] are strengthening of the continuous semantics, and we will also consider weakenings of it.

In other words, the underlying categories are classified according to the nature of their morphisms.

In the categories related to the semantics above, all morphisms are true functions.[18] Furthermore, for each model $\mathscr{M}$, $R(M)$ is an object of the *c.c.c.* and it coincides with the set of morphisms on $\mathscr{M}$. Finally the objects are (at least) *complete partial orders* and the morphisms are (at least) *continuous functions*, as defined below.

Definition 67. *Complete partial orders* (or *cpo's* or *domains*) are partially ordered sets having a least or "*bottom*" element (denoted by $\perp$) and in which every directed subset

[18] Though sometimes represented by their graphs or relevant part of them (the "traces" we define later on).

has a least upper bound (or sup). A subset B of $(M, \sqsubseteq)$ is *directed* if it is non-empty and any two elements of B are (upper) bounded in B.

Example 68. Complete lattices are cpo's. Another useful example is that of *flat domains* $E_\perp$, which are obtained by adding a bottom element below a set E viewed as an antichain: two distinct elements of E are incomparable.

Definition 69. A function between cpo's is *continuous* if it is monotone and commutes with sup's of directed sets.

For M, M' two cpo's, we let $[M \to M']$ denote the set of continuous functions from M to M', ordered with the pointwise order. It is easy to check that it is also a cpo.

Remark 10. * This is also a notion of continuity in the usual topological sense, here relative to Scott's topology, which is defined as follows. A subset U of M is (Scott-) *open* iff it is upwards closed and meets any directed subset B of M such that U contains $\sup(B)$.

Scott's computational motivations and intuitions behind the choice of domains and of continuous functions was that "computable functions induce continuous functions on domains" [108, 88]. The model P_ω is particularly adapted to highlight Scott's insight [106], but we will not develop this point here. We rather choose to illustrate the connection between computability and continuity with the (classical and) more immediate example of Myhill–Shepherdson's Theorem.

Let us consider partial functions on N. These can be viewed as total functions on $N_\perp$, where $N_\perp$ is the "flat" *cpo* associated to N. The intended meaning of $\perp$ is that it is a virtual integer on which we have no information.[19] For functions $f : N_\perp^n \to N_\perp$ continuity coincides with monotonicity, since $N_\perp$ is flat. This is no longer true when one considers functionals, namely functions of functions (where functions are ordered pointwise). For example a functional $F : (N_\perp \to N_\perp) \to N_\perp$ is continuous if and only if F is monotone and for all $f : N_\perp \to N_\perp$ there is a finite restriction g of f such that $F(f) = F(g)$.[20] Thus, the value of F on f (when different from $\perp$) only depends on a finite part of the graph of f. Such a result is known to be true for partial recursive functionals on partial recursive functions (this is Myhill–Shepherdson Theorem), and is computationally intuitive since if the computation of $F(f)$ had to look at infinitely many $f(m)$'s, then it would never give a value back.

Scott's continuous semantics deals with the case where $R(M)$ is exactly the set of continuous functions. The simplest relevant example of a c.c.c. in Scott's framework is the category of complete lattices and continuous functions.

Strengthenings of continuity. For further applications to lambda-calculus *stable*, and recently *strongly stable*, functions have been considered. These notions try to ap-

[19] Thus, $N_\perp$ describes the quantity of information we can have on an integer in a very manichean way: either we have no information at all ($\perp$) or we have a complete information ($n \in N$).

[20] Where g is *finite* if $\{x \in N \mid g(x) \neq \perp\}$ is finite. The above comment uses that f is the sup of the directed set $A_f \equiv \{g : N_\perp \to N_\perp \mid g \text{ finite and } g \leqslant f\}$ and that there is no infinite chain in $N_\perp$.

proximate the "sequential" aspects of λ-calculus and are stronger than continuity: one requires furthermore commutation with *infs* of large, and more or less natural, families of subsets of M. For example *stability* requires that $f(\inf(x,y)) = \inf(f(x), f(y))$ whenever x and y have an upper bound.

Weakenings of continuity (κ-*continuity*, where κ is any regular cardinal) also happen to be useful, and are already mentioned in [104]. They are necessary for modelling the "non-deterministic" computational extensions of λ-calculus (with $\kappa \equiv \aleph_1$) [97, 40]. Here "*non-deterministic*" means that we are in a frame where the normalizing procedure can give rise to several possible results. The notion of κ-continuity, for big κ's is also necessary for dealing with foundational questions [45, 19]; see Section 5.7. Scott's continuity corresponds to $\kappa \equiv \aleph_0$.

Since $R(M)$ is a domain, it is partially ordered. In the case of the continuous and weakly continuous semantics this order is taken as the *pointwise order* on functions. In the case of the stable or strongly stable semantics, the relevant order is *Berry's stable order*, which is a refinement of the pointwise order: $f \leqslant_s g$ iff $\forall x, y \in M$ $(x \leqslant y \Rightarrow f(x) = \inf(f(y), g(x)))$.

* It is worth noting that Proposition 60 can be strengthened to:

Proposition 70. *For each $t \in \Lambda$, the function $\rho \mapsto |t|_\rho$ is continuous* (*resp. weakly continuous*, *resp. stable*).

4.6. The basic example: continuity over full powerset domains

In this section we make precise what *continuous* means in the case where $(M, \sqsubseteq)$ is a full powerset, ordered by inclusion, namely

$$(M, \sqsubseteq) \equiv (P(D), \subseteq)$$

where D is a non-empty set, called here the *web* of M

The relevance of this example comes from the fact that all graph models, and in particular Engeler's model, are based on such full power sets.

Notation. From now on $\alpha, \gamma, ..$ always denote elements of D; $a, b, ..$ denote finite subsets of D (elements of $P^{\langle\omega}(D)$); and $d, d', ..$ denote elements of M.

Since $P(D)$ is a complete lattice, every subset of M, whether directed or not, has a sup, namely the *union* of its elements. Also $\perp = \emptyset$.

Thus, a function f on M is continuous iff it is monotone and commutes with all directed unions.[21] Since each $d \in M$ is the directed union of its finite subsets, a continuous function on M is completely determined by its *trace*, which is defined as follows:

Definition 71. The *trace* of a continuous function f is the subset $Tr(f)$ of $P^{<\omega}(D) \times D$ defined by:

$$Tr(f) \equiv \{(a, \alpha) \mid \alpha \in f(a)\}.$$

[21] For most practical purposes one can take D countable, which allows us to define continuity as monotonicity plus commutation with countable increasing unions.

Thus, $Tr(f)$ can be seen as the relevant part of the graph of f.

Indeed, it is easy to check that, for any $d \in M$,

$$f(d) = \{\alpha \mid \exists a \subseteq d \ (a, \alpha) \in Tr(f)\}.$$

Lemma 72. *Tr is a continuous and injective function from* $[M \to M]$ *to* $P(D^{<\omega} \times D)$, *respectively ordered by the pointwise order on functions and inclusion, and we have*: $f \leqslant g$ *iff* $Tr(f) \subseteq Tr(g)$.

Proof. Straightforward.

4.7. A basic tool: fixed point operators on p.o. sets

The following results will prove useful for solving domain equations, and hence for finding webbed models whose underlying *cpo* satisfies some given domain-constraint. They will also allow us to show that "positive partial webs" are "interpretable" in $P(\Lambda)$, allowing then to apply the reducibility technique to the models obtained by completion of such webs.

Incidentally, these results show that the existence of fixed point operators in a model of λ-calculus holds in fact in other wide settings.

Lemma 73. *Suppose* $(M, \sqsubseteq)$ *is a cpo. Then any continuous function* f *on* M *admits a least fixed point, given by* $T(f) \equiv \sup_{n\in\omega}(f^n(\bot))$. *More generally, suppose that* $f(x) \geqslant x$, *then* $T_x(f) \equiv \sup_{n\in\omega}(f^n(x))$ *is the least fixed point of* f *above* x.

Definition 74. Thus T is a uniform fixed point operator, called *Tarski's fixed point operator*. It is easy to see that it is the least possible fixed point operator.

In particular, T exists in all the models which belong to the continuous, stable, or strongly stable semantics. It is not necessarily the interpretation of Curry's fixed point combinator (defined in Example 16), but it can also be the simultaneous interpretation of all syntactic fixed point operators (cf. Theorem 47 and Corollary 48).

The existence of least fixed points is true in a wider setting (but then there is no guarantee that T will work):

Definition 75. A p.o. is *chain complete* if every chain in it (= totally ordered subset) has a *sup* (obviously cpo's are chain complete).

Lemma 76 (Knaster and Tarski). *Suppose* $(E, \sqsubseteq)$ *is a chain complete p.o. and that* f *is monotone on* M. *Suppose furthermore that* $x \sqsubseteq f(x)$. *Then there is a least fixed point* y *of* f *such that* $x \sqsubseteq y$.

Proof. The hypothesis ensures that the sequence $(y_\alpha)_{\alpha\in On}$ defined by $y_0 \equiv x$ and $y_\alpha \equiv \sup\{f(y_\beta) \mid \beta<\alpha\}$ is well defined and non-decreasing. Since E is a set the sequence is stationary, and its final value is clearly the least fixed point of f above x.

For building webs and models we will in fact use the existence of fixed point operators on the simple cpo's below.

Example 77. The set of preorders on a given set D is a cpo, once ordered with inclusion. Its bottom element is "=".

Example 78. Let (E,R,S) be a set equipped with two binary relations. Then the set of all (D,R_D,S_D), where D is a subset of E and R_D,S_D are the restrictions of R,S to D, is a *cpo*, once ordered by the substructure relation (or, equivalently here, by the inclusion relation on $P(E)$).

For interpreting webs we will rather make use of fixed point operators on chain complete p.o.'s (e.g. Proposition 112).

5. Webbed models

5.1. Introduction

In practice, all the models built for applications are "webbed models", which means, roughly speaking, that their domain is a subdomain of some $(P(D),\subseteq)$.

Scott's first model $\mathscr{D}_\infty$ was first built in 1969 as an inverse limit of a projective system [103]. Actually $\mathscr{D}_\infty$ is an *extensional* model, i.e. it equates all $\beta\eta$-equivalent terms. A second model, connected to ordinary recursion theory soon followed: Plotkin and Scott's P_ω (cf. [104, 106] and), built via an elementary construction (see Sections 5.5 and 5.6.3). Then came Engeler and Plotkin's model $\mathscr{E}$ [43, 93], and then other models, often built with practical purposes. Nearly all of these models belong to Scott's continuous semantics.

It was noticed progressively thereafter that, in fact, all practical models of the continuous semantics models admitted elementary constructions, as "reflexive information systems" or as "filter models" (Sections 5.6.8 and 5.6.9). This was already a webbed presentation of the models, and already allowed elimination of the inverse limit construction. Moreover, individual filter models themselves were systematically presented and studied in a proof-theoretic style, as "intersection type assignment systems" (a view which goes back to Coppo–Dezani–Honsell–Longo [32]).

The presentation that we will give here is simpler (we take more simple webs, even for the same models) and in particular simplifies the computation of the interpretations of terms and the equational study of the models. It allows us to view most of these models as sophisticated variations of Engeler's model and to replace the inverse limit construction by a simple completion process of partial webs, which can be treated algebraically and uniformly. This presentation takes its source in Krivine [76], Longo

[86] and Girard [49]. It can also be related, at least at the level of domains, to the event structures of [87],[22] so it is clearly not revolutionary. However, it clearly needs to be promoted since, probably for lack of a systematic treatment like here, it is hardly ever used for studying continuous models of untyped λ-calculus, in spite of its obvious advantages.[23,24]

* The simplified framework is made possible since most of the interesting models are in fact based on "prime-algebraic domains", which can be depicted from the structured subset of their "prime" elements, while the other webbed descriptions focus on the "algebraic" elements of the model (this is explained in Section 5.6.7). One should note here that, even in this simplified framework, the "same" model of λ-calculus can arise from different webs.

* *The logical view of* (*some*) *models.* As mentioned above, one of the traditional presentations of individual models is to describe them as logical systems. We prefer the present algebraic presentation since it is more direct, more general and much more synthetic.[25] But it is fair to say that the reducibility method, that we will use massively and which is one of the few general tools that are known for studying models, comes from proof theory.

If one wishes to put the models here in a logical framework, then the best view is to see them as "strict" intersection type systems, which amounts to the prime description and avoids redundancies, and hence avoids useless intermediate technical lemmas. This logical reading of the models is sketched in Section 5.6.1.

* *The key difference between both views* is that in the algebraic presentation the interpretation of a term is defined globally, by induction on the structure of the term, while the logical view focuses on the proofs showing that a given element α of the web belongs to the interpretation of t and points out that each such proof can be obtained from a finite amount of information. Thus the logical view is more effective, which can explain why it is the one most often used for applications in theoretical computer science. However this effectivity (and hence the logical reading) only makes sense for the models which can be generated by a recursive partial web. Furthermore, the presentation is feasible only if the partial web is very simple, and finally it does not allow a global treatment of systems.

Plan of the section (comments). The general idea is to follow a bottom-up approach, so that the reader gets familiar with the most simple (and already very useful) models before meeting the complex ones. We however insert more general commentaries as soon as we have met enough concrete examples (Sections 5.2 5.4 and 5.6.1). This bottom-up presentation corresponds in fact to the natural way one uses or should use

[22] *Event structures* were introduced for modelling concurrent processes.

[23] See, for example, the many equivalent definitions choosen by Abramsky and Ong for their lazy model in [2], which should be compared with the one presented in Example 146 below.

[24] Leading to opaque computations which hide the real mathematical reasons and hence the obvious possible generalizations. The example of [37] was already mentioned.

[25] The development of logical rules and related terminology is paper and time-consuming, and often repetitive from a paper to another one, since each model gives rise to a full system.

these models (except maybe for answering global questions). The idea is that we leave a class of models for a more complicated one only if the first one proves to be inadequate. In general, it is for being able to get a model which solves a specific *domain equation*. By this we mean a constraint of the shape $\mathscr{M} \simeq \mathscr{F}(\mathscr{M})$, where $\mathscr{F}$ is some given functor of the category of partially ordered sets, that the domain underlying the model has to satisfy. Such equations can, for example, help to interpret new constants, coming for example from programming motivations.[26]

Of course, a more concise presentation was possible, but the intention of this paper was also to attract people ordinarily reluctant to use models, and we thought that a much too abstract setting had to be avoided.

5.2. Intended applications of webbed models

1. *Proving properties of the syntax*. Models are able to *reflect some operational features* of λ-calculus, even though the λ-congruence they induce on Λ is coarser than β-equivalence, and hence flattens β-reduction. This shows up throughout the literature on λ-models and is clearly illustrated in Sections 5.3.3, 5.3.4, and 5.6.2 below, where normalization theorems are proved using specific models, and in Section 6.7.

It is moreover even possible to trace head-reduction in depth and uniformly for all graph models (up to some complexity point).[27]

A different kind of applications to the syntax is the existence of *refutation arguments*, also called *non-definability results*. Models can indeed be used to prove that "there is no term $t \in \Lambda$ such that ...". Examples are given in Sections 5.3.2 and 5.6.3 and others in [8, Chapter 20.3].

2. *Consistency arguments*. Models can be used to prove the consistency of a given theory $\mathscr{T}$ extending β-reduction (even with constants added). This is of most value if no *CR* reduction underlies $\mathscr{T}$, but also *CR*, even if true, could be too tedious, or too difficult to prove (as a matter of fact it is always tedious !). Of course a model can also prove directly the consistency of a given reduction system. Examples were already given, and others will be given in Sections 6, 6.8, and 5.7.

5.3. A paradigm: Engeler's model $\mathscr{E}$

5.3.1. Definition of $\mathscr{E}$

The idea behind Engeler's model is simple: we set $M \equiv (P(D), \subseteq)$ and the set D is chosen such that $P^{<\omega}(D) \times D \subseteq D$, so that the trace of any continuous $f : M \to M$ is an element of M. For building such a D we start from an arbitrary non empty set A of "atoms", not containing pairs, and we build inductively the smallest set D such that

$$D = A \cup (P^{<\omega}(D) \times D)$$

by taking $D = \bigcup D_n$, where $D_0 = \emptyset$ and $D_{n+1} = A \cup (P^{<\omega}(D_n) \times D_n)$.

[26] Many examples are given in Plotkin's postgraduate course [96].

[27] The deepest (and most technical) result of this kind is probably Kerth [68] since it even concerns the head-reduction of unsolvable terms.

The *rank* of $\alpha \in D$ is then the smallest n such that $\alpha \in D_n$, the rank of a is defined as the sup of the ranks of its elements (0 if a is empty).

Definition 79. Engeler's model is defined as $\mathscr{E} \equiv (M, A, G)$,[28] where $M \equiv (P(D), \subseteq)$ and $G \equiv Tr$ while application A is defined by

$$A(d)(d') \equiv dd' \equiv \{\alpha \in D \mid \exists a \in P^{<\omega}(D)\ a \subseteq d'(a,\alpha) \in d\}. \tag{1}$$

Thus, for any continuous $h : M \to M$ we have

$$G(h) \equiv Tr(h) \equiv \{(a,\alpha) \mid \alpha \in h(a)\}.$$

It is easy to check that application is continuous, that $A \circ G = id$, and that $A(d)$ is continuous for all d. It follows from these two last facts that the representable functions are exactly the continuous ones.

Remark 11. $|\lambda x_1 \ldots x_n.t|_\rho = \{(a_1,(\ldots(a_n,\alpha)\ldots)) \mid \alpha \in |t|_{\rho[\vec{x} := \vec{a}]}\}$.

Example 80. In Engeler's model,

$$|I| = \{(a,\alpha) \mid \alpha \in a\};\ \ |K| = \{(a,(b,\alpha)) \mid \alpha \in a\};$$

$$|\varepsilon| \equiv |\lambda x \lambda y.xy| = \{(a,(b,\alpha)) \mid \exists b' \subseteq b\ (b',\alpha) \in a\};$$

$$|\delta| \equiv |\lambda x.xx| = \{(a,\alpha) \mid \alpha \in aa\}$$

$$|\Omega| = \emptyset \quad \text{and} \quad |\lambda x_1 \ldots x_n.\Omega| = \emptyset \quad \text{and} \quad |\Omega t|_\rho = \emptyset \quad \text{for all } t.$$

Proof of the last line. $|\Omega| \equiv |\delta\delta| \equiv \{\alpha \mid \exists a \subseteq |\delta|\ (a,\alpha) \in |\delta|\} = \{\alpha \mid \exists a \subseteq |\delta|\ \alpha \in aa\} = \{\alpha \mid \exists a \subseteq |\delta|\ \exists a' \subseteq a\ (a',\alpha) \in a\}$.

Suppose that there is such an α. Choose $a \subseteq |\delta|$ minimal such that $\alpha \in aa$, and $a' \subseteq a$ such that $(a',\alpha) \in a$. Since $a \subseteq |\delta|$ we have $\alpha \in a'a'$, hence $a' = a$ by minimality of a. Thus $(a,\alpha) \in a$, which is easily shown to be impossible by a simple argument on the rank of a. The other two assertions follow easily from the first.

Remark 12. $|t|_\rho = \bigcup\{|t[\vec{x} := \vec{a}]| \mid \vec{a} \in (P^{<\omega}(D))^{l(\vec{x})}$ and $\vec{a} \subseteq \rho(\vec{x})\}$, if $FV(t) \subseteq \vec{x}$, where inclusion on tuples is pointwise inclusion.

This is immediately checked by induction on t, and is a particular case of Proposition 70.

[28] *Warning*: The author realized much too late (and apologizes for this) that she used the letter "A" for two different uses, namely:

1. the "application" function of a model, and
2. the set of atoms of $\mathscr{E}$, and, more generally later on, the underlying set of partial webs.

Since it is most likely that changing it at that stage would have introduced inconsistencies we chose to leave it to the reader to distinguish between both uses. This will be in any case always clear from the context.

5.3.2. Applications of Engeler's model to λ-calculus

Normalization theorems. Concerning solvable terms it will be proved that:

Theorem 81. *For any $t \in \Lambda$, t is solvable iff $\mathscr{E} \models t \neq \bot$ iff t is head-normalizable.*

And we will also prove the left-normalization theorem, in a similar way.

In the next comments we compare the proof given here to the main previous proofs of the head-normalization theorem.

- *The first proof that solvable terms are indeed head-normalizable was done via a similar statement relative to Scott's $\mathscr{D}_\infty$, [61, 112, 113], and $\mathscr{P}_\omega$ [61] The proof was very involved and relied on the approximation method, which was introduced for this purpose.
- *More recent proofs of this result were done, syntactically, by working directly on "intersection assignment type systems" via the "reducibility method". Such a proof is presented in [77], using Dezani's system D_Ω, and was already simplifying previous presentations.
- *The normalization proofs that we give here use Krivine's variant of the reductibility argument, but directly in Engeler's model. This allows simplification in two further places. First we avoid the redundancies coming from the use of a non-strict type system like D_Ω. Second we do not have to prove that types are preserved by β-reduction, since the stronger semantic counterpart of this fact, which states that β-equivalent terms have the same interpretation, has been shown once and for all for all models. Finally, this proof can be extended to cover, uniformly, many webbed models, including $\mathscr{D}_\infty$ and $\mathscr{P}_\omega$, and can be adapted to treat other normalization theorems.

Refutation arguments. We just give a simple example, from which it is easy to extrapolate other applications.

Proposition 82. *There is no term t such that $tu = T$ if u is head normalizable, and $tu = F$ otherwise.*

Proof. We look at the interpretations of terms in Engeler's model. Using the monotonicity of application, and the fact that $\Omega = \bot$, we deduce first that $|F| \leqslant |T|$, since $|F| = |t\Omega| = |t|\bot \leqslant |t|\,|I| = |tI| = |T|$, and then that $d' \leqslant d$ for all d, d' in the model, since $d' = |F|dd' \leqslant |T|dd' = d$, which is a contradiction.

This result could also have been obtained by straightforward syntactical considerations; it is however clear that for more complicated statements such a semantic argument is a significant gain, if not the only possible option.

Consistencies. Thus, $\mathscr{E}$ can be used to prove that it is consistent to equate all unsolvables. It could also be used to show that it is consistent to equate all fixed point operators (to Tarski's fixed point operator on $\mathscr{E}$ indeed) and more generally it could

also be used to prove that equating all terms wich have the same Böhm tree is consistent, but this is more difficult and calls for the approximation technique.

A more natural, and very simple, example is the consistency of the (*CR*) system obtained by adding to λ-calculus the rewriting rules $\lambda x.\Omega \to \Omega$ and $\Omega t \to \Omega$ for any t, using Example 80.

5.3.3. Proving the head-normalization theorem via Engeler's model

Theorem 83. *For any $t \in \Lambda$ the following are equivalent*:
1. *t is solvable*,
2. $\mathscr{E} \models t \neq \emptyset$,
3. *t is head-normalizable.*

Proof. The easy (3) $\Rightarrow$ (1) has already been proved in Section 3.6.

(1) $\Rightarrow$ (2). First we note that $|I| = \{(a, \alpha) \mid \alpha \in a\} \neq \emptyset$.

Suppose now that t is a *closed* solvable term; then $t\bar{u} =_\beta I$ for some tuple $\bar{u}$, of closed terms. Since it is immediate from the definition of the interpretation of application, that $|v|$ is non-empty as soon as $|vw|$ is we get easily $|t| \neq \emptyset$ in this case. Finally, if $\lambda x_1 \ldots x_n.t$ is a closure of an open solvable term t, then $|\lambda \bar{x}.t| = \{(a_1, (\ldots (a_n, \alpha) \ldots)) \mid \alpha \in |t[\bar{x} := \bar{a}]|\}$ is non-empty, hence $|t|_{\rho[\bar{x} := \bar{a}]}$ is non-empty, and $\mathscr{E} \models t \neq \emptyset$.

The rest of the section is devoted to the proof of (2) $\Rightarrow$ (3). We need the following definitions and notations:

Definition 84. For any two $X, Y \subseteq \Lambda$, let $X \to Y \equiv \{t \in \Lambda \mid \forall u \in X \ tu \in Y\}$.

Obviously, $X' \subseteq X$ and $Y \subseteq Y'$ imply $X \to Y \subseteq X' \to Y'$.

Definition 85. A subset $X \subseteq \Lambda$ is *saturated* if for all $n \geqslant 0$, $x, u, t, t_1, \ldots, t_n \in \Lambda$, $u[x := t]\, t_1 \ldots t_n \in X$ implies $(\lambda x.u) t t_1 \ldots t_n \in X$. Note that Y saturated implies $X \to Y$ saturated, for any X.

Definition 86. $\mathscr{N}_h$ is the set of head-normalizable terms.

$\mathscr{N}_h^0 \equiv \{x t_1 \ldots t_n \mid n \geqslant 0, t_1, \ldots, t_n \in \Lambda\}$; thus $V \subseteq \mathscr{N}_h^0 \subseteq \mathscr{N}_h$.

$S_h \equiv \{X \in \Lambda \mid X$ is saturated and $\mathscr{N}_h^0 \subseteq X \subseteq \mathscr{N}_h\}$;

Lemma 87. (i) *S_h is closed under the arrow operator, under all intersections and unions, and contains $\mathscr{N}_h$.*

(ii) *S_h is closed under the function H defined by $H(X) = (\Lambda \to X)$.*

Proof. (i) The two last points are obvious. To prove the first statement it is indeed sufficient to check that $\mathscr{N}_h^0 \subseteq \mathscr{N}_h \to \mathscr{N}_h^0$ and $V \to \mathscr{N}_h \subseteq \mathscr{N}_h$, and then to use the co/contravariance property of the arrow. The first point follows directly from the definition of $\mathscr{N}_h^0$ and the second is easy: suppose that $tx \in \mathscr{N}_h$; we want to conclude that $t \in \mathscr{N}_h$.

Suppose that the head-sequence starting from t is infinite; then it obviously contains an abstraction, otherwise tx would not be head-normalizable. If $t \to_h \lambda x.u$ then $tx \to_h u$; since tx is head-normalizable it is also the case for u and $\lambda x.u$, hence for t, which is a contradiction.

(ii) Suppose $X \in S_h$. Then $\Lambda \to \mathscr{N}_h^0 \subseteq \Lambda \to X \subseteq \mathscr{N}_h \to X$ by the co/contra-variant properties of the arrow. Now $\mathscr{N}_h \to X \subseteq \mathscr{N}_h$ since S_h is closed under the arrow, and $\mathscr{N}_h^0 \subseteq \Lambda \to \mathscr{N}_h^0$ by the very definition of $\mathscr{N}_h^0$, hence $\Lambda \mapsto X \in S_h$.

Proving $(2) \Rightarrow (3)$ amounts to showing that for each $\alpha \in D$ we have $J(\alpha) \subseteq \mathscr{N}_h$, where $J(\alpha)$ is the set of all terms $t \in \Lambda$ such that $\exists \rho\ \alpha \in |t|_\rho$. This is not directly possible, but we will approximate J by a function $I_h : D \to P(\Lambda)$ for which one can prove $\alpha \in |t|_\rho \Rightarrow t \in I_h(\alpha)$ and $I_h(\alpha) \subseteq \mathscr{N}_h$ for all $\alpha \in D$.

Definition 88. $I_h : D \to P(\Lambda)$ is defined by induction on the rank of $\alpha \in D$. First one fixes $I_h(\alpha) \in S_h$ for all atoms $\alpha \in A$ (for example $I_h(\alpha) = \mathscr{N}_h$) and then we let $I_h((a,\alpha)) \equiv I_h(a) \to I_h(\alpha)$, where, for any $d \subseteq D$, $I_h(d) \equiv \bigcap \{I_h(\beta) \mid \beta \in d\}$ if $d \neq \emptyset$ and $I_h(\emptyset) \equiv \Lambda$.

Lemma 89. *$I_h(\alpha) \in S_h$ for all $\alpha \in D$.*

Proof. By induction on the rank of α. This is obvious if α is an atom. If $\alpha \equiv (b,\beta)$ there are two cases to consider: if $b \neq \emptyset$ then $I_h(\alpha) \in S_h$ follows from the closure property of S_h w.r.t. the arrow, and if $b = \emptyset$ from its closure w.r.t. H (Lemma 87). □

That $\alpha \in |t|_\rho \Rightarrow t \in I_h(\alpha)$ is proved via a stronger property:

Proposition 90 (Adequacy (of I_h for $\mathscr{E}$)). *For all $t \in \Lambda, \rho$, and $\alpha \in D$, for all $\vec{x} \supseteq FV(t)$ and $\vec{v} \in \Lambda^{l(\vec{x})}$: if $\alpha \in |t|_\rho$ and $\forall i \leqslant l(\vec{x})\ v_i \in I_h(\rho(x_i))$ then $t[\vec{x} := \vec{v}] \in I_h(\alpha)$.*

Corollary 91. *For all t, ρ, α, we have $\alpha \in |t|_\rho \Rightarrow t \in I_h(\alpha)$.*

Proof. The corollary follows from the proposition just by taking $\vec{v} = \vec{x}$, which is made possible since V is included in $I_h(\emptyset)$ and in all $I_h(\beta)$.

The proof of Proposition 90 is by induction on the length of t, and we use of course the formulas defining the interpretation of application and abstraction, namely (1) and Remark 11. The case where t is an abstraction is the only non-trivial case and it motivated both the introduction of saturation and the stronger statement of the lemma.

Case 1. $t \equiv x_i \in \vec{x}$, then $\alpha \in \rho(x_i)$. Also $t[\vec{x} := \vec{v}] \equiv v_i \in I_h(\rho(x_i)) \subseteq I_h(\alpha)$.

Case 2. $t \equiv uw$. Since $|t|_\rho = |u|_\rho |w|_\rho$ there exists $a \subseteq |w|_\rho$ such that $(a,\alpha) \in |u|_\rho$. By induction hypothesis we have $u[\vec{x} := \vec{v}] \in I_h(a) \to I_h(\alpha)$ and $w[\vec{x} := \vec{v}] \in I_h(a)$ (for $a = \emptyset$ we use also that $I_h(\emptyset) = \Lambda$). Thus $t[\vec{x} := \vec{v}] \equiv u[\vec{x} := \vec{v}]w[\vec{x} := \vec{v}] \in I_h(\alpha)$.

Case 3. $t \equiv \lambda y.r$ and we assume $y \notin \vec{x}$. If $\alpha \in |\lambda y.r|_\rho$, then $\alpha \equiv (b,\beta)$ with $\beta \in |r|_{\rho[y:=b]}$. Now $t[\vec{x} := \vec{v}] \equiv \lambda y.r[\vec{x} := \vec{v}]$ and we have to show that, for all $u \in I_h(b)$, $t[\vec{x} := \vec{v}]u \in I_h(\beta)$. Now, $t[\vec{x} := \vec{v}]u \equiv (\lambda y.r)[\vec{x} := \vec{v}]u \equiv (\lambda y.r[\vec{x} := \vec{v}])u$. Since $I_h(\beta)$ is saturated, it is enough to prove that $r[\vec{x} := \vec{v}, y := u] \in I_h(\beta)$, which follows from the induction hypothesis applied to r and $\rho[y := b]$ (since $u \in I_h(b)$).

Corollary 92. *For all t, if $\mathscr{E} \models t \neq \emptyset$ then t is head-normalizable.*

Proof. By Lemma 89 and Corollary 91.

5.3.4. Proving the left-normalization theorem via Engeler's model

We now indicate how to modify the preceding proof in order to obtain the left-normalization theorem.

We consider the subset D^- of D which is the smallest set D^- such that

$$D^- = A \cup ((P^{<\omega}(D^-) - \{\emptyset\}) \times D^-)$$

and which can be inductively constructed along the same lines than D. The elements of D^- are therefore those elements of D in which $\emptyset$ does not appear.

We call *strict* the elements of D^- and the (possibly empty) subsets of D^-; we say that the environment $\rho : V \to \mathscr{E}$ is t-strict, for t a term, if $\rho(x)$ is strict for all x free in t. Finally we say that $\mathscr{E}$ *strictly satisfies* $t \neq \emptyset$ ($\mathscr{E} \models_s t \neq \emptyset$) if there is a strict α and a t-strict ρ such that $\alpha \in |t|_\rho$.

Theorem 93. *For any $t \in \Lambda$ the following are equivalent*:
1. *t is normalizable*,
2. *$\mathscr{E} \models_s t \neq \emptyset$*,
3. *t is left-normalizable.*

Proof. (3) $\Rightarrow$ (1) is trivial.

(1) $\Rightarrow$ (2). It is enough to check it for normal terms, since β-equivalent terms have the same interpretation. Then the proof goes by induction on t. For variables it is immediate; otherwise $t = \lambda x_1 \dots x_m . y t_1 \dots t_n$, with all t_i's normal. By the induction hypothesis there are, for each i, a strict α_i and a t_i-strict ρ_i such that $\alpha_i \in |t_i|_{\rho_i}$ and by Remark 12 we can assume all ρ_i's finite. Since $|.|_\rho$ is monotone in ρ, we can even assume that $\alpha_i \in |t_i|_\rho$ for the same $\bar{t}$-strict and finite ρ. Let α be any strict element, let $\beta \equiv (\{\alpha_1\}, (\dots, (\{\alpha_n\}, \alpha) \dots))$, let $\rho' \equiv \rho[y := (\rho(y) \cup \{\beta\})]$, and finally let $a_i \equiv \rho'(x_i) \cup \{\alpha\}$ (which ensures that the a_i's are strict, non-empty, and that $\rho'[\bar{x} := \bar{a}] \geqslant \rho'$). It is easy to check, using the definition of application and abstraction in $\mathscr{E}$, that $(a_1, (\dots, (a_m, \alpha) \dots)) \in |t|_{\rho'}$ (since $|t|_{\rho'} = \{(a_1, (\dots, (a_m, \alpha) \dots)) \mid \exists b_1 \subseteq |t_1|_{\rho'[\bar{x} := \bar{a}]} \dots \exists b_n \subseteq |t_n|_{\rho'[\bar{x} := \bar{a}]} (b_1, (\dots (b_n, \alpha) \dots)) \in |y|_{\rho'[\bar{x} := \bar{a}]}\}$). Note that the proof, as written, works even if y is one of the x_i.

The proof of (2) $\Rightarrow$ (3) is now nearly the same as in the previous section. This time we set:

- $\mathscr{N}_l$ is the set of left-normalizable terms.
- $\mathscr{N}_l^0 \equiv \{x t_1 \dots t_n \mid n \geqslant 0, t_1, \dots, t_n \in \mathscr{N}_l\}$; thus $V \subseteq \mathscr{N}_l^0 \subseteq \mathscr{N}_l$.
- $S_l \equiv \{X \subseteq \Lambda \mid X$ is saturated and $\mathscr{N}_l^0 \subseteq X \subseteq \mathscr{N}_l\}$.

We define I_l like I_h before, except that we require now that $I(\alpha) \in S_l$ for all atoms α.

The proofs of the results below are exactly the same as before, except for Lemma 95 where it is still simpler since we never meet the case $b = \emptyset$.

Lemma 94. *S_I is closed under the arrow operator, under all intersections and unions, and contains $\mathcal{N}_I$.*

Lemma 95. *$I_I(\alpha) \in S_I$ for all strict $\alpha \in D$.*

Proposition 96 (Adequacy (of I_I for $\mathscr{E}$)). *For all t, α, ρ, for all $\vec{x} \supseteq FV(t)$ and $\vec{v} \in \Lambda^{l(\vec{x})}$: if $\alpha \in |t|_\rho$ and $\forall i \leqslant l(\vec{x})\ v_i \in I_I(\rho(x_i))$, then $t[\vec{x} := \vec{v}] \in I_I(\alpha)$.*

Corollary 97. *For all t, for all strict α, and all t-strict ρ, we have $\alpha \in |t|_\rho \Rightarrow t \in I_I(\alpha)$.*

Corollary 98. *For all t, $\mathscr{E} \models_s t \neq \emptyset$ implies t left-normalizable.*

5.4. Methods for constructing and studying webbed models

In practice, there are only two methods for building webbed models. The first one is general and amounts to applying a *canonical completion process* to a "partial web" satisfying some constraints. The second one, called *forcing*, can also be seen as another kind of completion process, but certainly *anti-canonical* and *interactive*, so to say.

There are also two main tools for studying the equational theories of webbed models, which are clearly not sufficient. The first one is the *reducibility technique*, and the second one is the *approximation method*. The approximation method can in fact be understood as a deepening of the reducibility technique. Or, rather, the reducibility method can be seen as a simplification of the "computability technique"[29] which is used in most papers to prove approximation theorems (for individual filter models) and normalization theorems (w.r.t. given strategies).

These four methods apply to all the classes of webbed models that we present in this paper (and hence work for the continuous, weakly continuous, stable and strongly stable semantics).

A further interesting tool deserves to be mentioned, namely Plotkin's method of *logical relations* which allows, for example, new refutation arguments. However, it has to be adapted specifically for each particular problem, has been rarely used for models of (untyped) λ-calculus yet, and has no special concern with webbed models (at this state of our knowledge).

In this section we give a direct and algebraic presentation of the *completion method* and of the *reducibility technique*. Furthermore, we make it as uniform as possible, in order to be able to express and prove some transfer theorems. The *approximation method* would clearly deserve a similar treatment, but it could not be treated here, and will certainly be more delicate. The *forcing method* is more specific, so we will not

[29] Reducibility defines predicates $J(\alpha)$, for each $\alpha \in D$, where computability uses predicates $Comp(\rho, \alpha)$, for each α, ρ, where ρ is an environment.

describe it either, even if we will comment on it. We discuss these four methods a little more deeply now.

Canonical completion (s). This method encompasses the inverse limit construction, at least if we only consider models built from webs. In fact, it replaces the inverse limit construction by an elementary set theoretic construction at the level of webs, starting from a "partial web". This method allows us to build models satisfying prescribed constraints, in general domain equations or/and inequations, and it allows variants which give models of specific λ-calculi (extensional, lazy, call-by-value, λI-, aso.) or of extensions of these calculi (for example with parallel or concurrent features).

Once a semantics has been fixed, there is a uniform way to complete any partial web into a model of λ-calculus, which works for all webs. The other variations work for many, but not all, webs. The *l-completion* will be presented here, in parallel to the more general method, since it is the key for getting lazy models of λ-calculus, and we will also discuss a variant useful for Map Theory (which is lazy, *plus something else*).

Lazy models, extensional models and models of Map Theory are in particular models of ordinary λ-calculus. This is not the case with the models of call-by-value or of λI-calculus, and we will not touch them here, even if similar webbed presentations and completion processes could be extrapolated from the examples treated in [41, 56], aso..

Forcing. This method allows one to build models satisfying specified term-equations (for example $\Omega = t$, for any t). Introduced for graph models the method works for other classes and for different semantics. However the scope of this method is rather limited. For more comments see Section 6.8 and the end of Section 5.5.

Reducibility. The method is originally due to Tait and is issued from proof-theoretic considerations. We use here the simplified version due to Krivine, and moreover apply it directly in models, in a more traditional algebraic or model-theoretic style, which allows further simplifications.

The method can be applied to all models generated by *positive* partial webs, in a sense we make precise in the course of the paper, and then at least proves the sensibility or the strong laziness of these models and their adequation w.r.t. relevant strategies.

The approximation method. Issued from the pioneering work of Hyland and Wadsworth. Used in connection with syntactical results (separability theorems) it allows one to study more deeply the equational theories of (some) webbed models, and in some cases to determine it exactly. This is for example the case for Scott's $\mathscr{D}_\infty$ [61, 112, 113], $\mathscr{P}_\omega$ [61] and $\mathscr{E}$. When these models are presented in an intersection typing system style, the approximation theorem appears as the semantic version of a normalization theorem for the proofs (also called *cut-elimination*) in this typing system (a view which originates in [31]). There is also a weak form which allows a (very) partial study of the equational theory of non-semi-sensible models [59].

5.5. Graph models

The class of models we present now belongs to the continuous semantics and is the simplest generalization of the Engeler–Plotkin construction. Following [102] we call

these models *graph models*. The equational study of this class was initiated by Longo in [86], where these models are called *PSE-algebras* (for Plotkin, Scott and Engeler); they were also considered by Plotkin.[30] Historically, the first graph model was Plotkin and Scott's P_ω, which is also known in the literature as "the graph model". "Graph" referred then to the fact that the continuous functions were encoded in the model via (a sufficient fragment of) their graph. In fact, this is more generally the case of all the models presented in Section 5.

Although *formally* all graph models are close to Engeler's model, their equational properties can be very different.

The class contains *no extensional model*. In fact, we will see that *graph models never satisfy* $|\varepsilon| \leqslant |I|$ though many, like P_ω but unlike $\mathscr{E}$, can satisfy $|I| < |\varepsilon|$. Also, all graph models are complete lattices. So there are many functors $\mathscr{F}$ on partially ordered sets for which the domain equation $\mathscr{D} \simeq \mathscr{F}(\mathscr{D})$ cannot be solved within this class.[31]

The class of graph models is however rich enough to help us to solve various consistency problems (like the *easiness* of Ω[32]). As a matter of fact there are 2^ω distinct equational theories of graph models [66], which should give us a reasonable freedom; however the real representativeness of this numerical richness is not well understood and some irritating natural questions are left open. We do not know, for example, whether there is a (graph) model whose theory is exactly λ_β (see end of this section and Section 6.2). Such problems are presented in Sections 6.2, 6.3 and 6.8.

A graph model is defined from a set D endowed with an injection $i : P^{<\omega}(D) \times D \to D$ which allows the model to encode the traces of the continuous functions. In the case of Engeler's model i was the inclusion.

Definition 99. A *graph model* is a model generated by a pair (D, i), where D is a non-empty set and $i : P^{<\omega}(D) \times D \to D$ is an injection, in the following way:

$$\mathscr{G} \equiv (M, A, G) \quad \text{with } M \equiv (P(D), \subseteq)$$

where application, and hence A, is defined by

$$dd' \equiv \{\alpha \in D \mid \exists a \in P^{<\omega}(D)\ a \subseteq d'\ i(a, \alpha) \in d\}$$

and for any continuous $h : M \to M$,

$$G(h) \equiv \{i(a, \alpha) \mid \alpha \in h(a)\}.$$

The pair (D, i) is called, for short,[33] the *web of the model* and D itself is the *web of the underlying domain*.

[30] Compare Plotkin [98] which is the recent version of a 20-year old manuscript.

[31] Examples are provided in the sequel.

[32] *Easy terms* were defined in Definition 49.

[33] The correct denotation would be "a web generating the model (or the domain)", since different webs can generate the same domain (e.g. P_ω below).

It is easy to check directly that "application" and G are continuous, that $A \circ G = id$, and that the representable functions are exactly the continuous ones.

The *interpretation of terms* in a graph model is similar to that in $\mathscr{E}$: we just change (a, α) for $i(a, \alpha)$.

In particular, it is easy to check that always $i(i(\emptyset, \alpha), i(\{\alpha\}, \alpha)) \in |\varepsilon| - |I|$, and that $|I| \subset |\varepsilon|$ if i is a bijection, as it is the case for P_ω below.

Example 100 (*Plotkin and Scott's P_ω (historical definition)*). We let D be N, the set of integers, and $i(a, \alpha) \equiv \langle e(a), \alpha \rangle$, where $\langle , \rangle$ and e are the usual bijective encodings of pairs and finite subsets in N, namely: $\langle m, n \rangle \equiv (m + n)(m + n + 1)/2 + n$ and $e(a) \equiv \sum_{k \in a} 2^k$ if $a \neq \emptyset$ and $e(\emptyset) \equiv 0$.

This presentation of P_ω is issued from recursion-theoretic motivations [106].

Less known is that there is a much simpler presentation of P_ω due to Longo [86]. This second presentation is particularly convenient for proving that P_ω is sensible, along the same lines as we did for $\mathscr{E}$ (see Lemma 102 below). More generally, this second view is much more practical for dealing with the equational theory of P_ω. Actually, this model is one of the few whose equational theory is completely known and $Th(P_\omega) = Th(\mathscr{E}) = BT$ [86, 61] (as mentioned in Section 3.11) but proving this would need further work, the approximation method and separability theorems.

Example 101 (*Longo's definition of P_ω, and the models $\mathscr{P}$*). D is defined inductively from a set A (not containing pairs), as the smallest set D such that

$$D = A \cup [(P^{<\omega}(D) \times D) - (\{\emptyset\} \times A)]$$

and the injection i is defined by

$$i(a, \alpha) \equiv (a, \alpha) \quad \text{if } a \neq \emptyset \text{ or } \alpha \notin A,$$

$$i(\emptyset, \alpha) \equiv \alpha \quad \text{if } \alpha \in A \ (*).$$

The model P_ω corresponds to the case where A is a singleton, say $\{p\}$.

The proofs that both presentations of P_ω are equivalent is easy but outside the scope of the present paper (see [86], or [68, 69] for a more general setting). The idea is that there is a correspondence between the webs, which induces an isomorphism of the models (in a sense to be defined).

Lemma 102. *P_ω is sensible.*

Proof. The proof goes exactly along the same lines as for $\mathscr{E}$ (Section 5.3.3) except that, to take $(*)$ into account, we have to choose I, at rank zero, such that $I(\emptyset) \to I(p) = I(p)$; in other words $I(p)$ has to be a fixed point of H in S_h (recall H was defined by $H(X) \equiv \Lambda \to X$). There are such fixed points since $(S_h, \subseteq)$ is closed under intersections and under H, and since H is monotone and commutes with

intersections: as a matter of fact $\bigcap\{H^n(\mathcal{N}_h) \mid n \in \omega\}$ works; one could alternatively use the fact that H is monotone and S_h is a complete lattice (Lemma 76).

The completion method for building graph models from partial pairs. This method generalizes the construction of $\mathscr{E}$ and the second construction of P_ω. It was initiated by Longo in [86] and recently developed and used on a wide scale by Kerth in [68, 69].

Definition 103. A *partial pair* is a pair (A, j) where A is a set, and $j : A^{<\omega} \times A \to A$ is a partial injective map. A *total pair* is a partial pair such that j is total, i.e. (A, j) is the web of a graph model. In the sequel, (D, i) will always denote a total pair.

Definition 104 (*Completion of partial pairs*). From any partial pair one can inductively define a total pair, called its *completion*.

First the set D is defined inductively as the smallest solution of the set-theoretic equation $D = A \cup ((P^{<\omega}(D) \times D) - dom(j))$. The completion is then taken as (D, i), where $i(a, \alpha) \equiv j(a, \alpha)$ if $(a, \alpha) \in dom(j)$ and $i(a, \alpha) = (a, \alpha)$ otherwise (but see the next remark).

A notion of rank can then be naturally defined, in such a way that the elements of A are the elements of rank 0.

Remark 13. *The above definition works in fact only if $range(j) \cap [(P^{<\omega}(D) \times D) - dom(j)] = \emptyset$, since otherwise i would not be injective. In the general case one should replace the definition of D above by a less canonical one. We will not bother about this since the condition is obviously satisfied if A is a set not containing pairs, which will be the case for all our examples. The condition is also trivially satisfied if (A, j) is total; in this case $(D, i) = (A, j)$.

The reducibility method. We now extract the core of the reducibility method, in order to see that it works for all completions of "positive partial pairs", as defined below. [34]

Definition 105. An *interpretation* of (A, j) is a function $I : A \to P(\Lambda)$ such that 1. $I(\alpha)$ is saturated for all $\alpha \in A$ [35] and 2. $I(j(a, \alpha)) = I(a) \to I(\alpha)$, where $I(a) \equiv \bigcap\{I(\alpha) \mid \alpha \in a\}$ if a non-empty, and $I(\emptyset) \equiv \Lambda$.

For $S \subseteq P(\Lambda)$ the interpretation I is an *S-interpretation* if $I(\alpha) \in S$ for all α. A pair (A, j) is *S-interpretable* if there exists an S-interpretation of (A, j).

Proposition 106. *If I is an interpretation of (D, i), then I is adequate for the graph model generated by (D, i) (in the sense of Proposition 90).*

[34] A very recent syntactical result of R. David has the consequence that there exist a lot of sensible models which are built from "non-positive" partial pairs (see Section 6.3). Such models are not accessible to the reducibility technique.

[35] *Saturated sets* were defined in Definition 85.

Proof. Same as for $\mathscr{E}$, except that $i(a,\alpha)$ replaces (a,α) everywhere.

Proposition 107. *Any interpretation J of (A,j) admits a unique extension to an interpretation I of the completion (D,i) of (A,j).*

Proof. I is defined inductively on D (induction on the rank) as the unique function which extends J and is such that $I((a,\alpha)) \equiv I(a) \to I(\alpha)$. I is an interpretation since the set of saturated sets is closed under intersection, $\to$, and $X \mapsto (\Lambda \to X)$.

Definition 108. The *operator H generated by a partial pair* (A,j) on $P(\Lambda)^A$ is defined as follows. Let X_α, $\alpha \in A$, denote the αth component of $\vec{X} \in P(\Lambda)^A$. Then $\vec{Y} \equiv H(\vec{X})$ if

(i) $Y_\alpha \equiv X_\alpha$ for all $\alpha \in A$-*range*(j),
(ii) $Y_\alpha \equiv (\bigcap \{X_\beta \mid \beta \in b\}) \to X_\gamma)$ if $\alpha = j(b,\gamma)$ and $b \neq \emptyset$,
(iii) $Y_\alpha \equiv (\Lambda \to X_\gamma)$ if $\alpha = j(\emptyset,\gamma)$.

Definition 109. The partial pair (A,j) is *positive* if there is some $B \subseteq A$ such that H is monotone over $P(\Lambda)$ ordered by the B-inclusion order. This order is defined by: $\vec{X} \subseteq_B \vec{Y}$ if $X_\alpha \subseteq Y_\alpha$ for all $\alpha \in B$, and $Y_\alpha \subseteq X_\alpha$ otherwise.

In all practical cases it is clear from the definition of a partial pair whether it is positive or not (using the co-contra-variance properties of $\to$) as will show the examples below. A formal characterization speaking of the occurrences of the X_α in the system $\vec{X} = H(\vec{X})$ could also be made explicit.

Definition 110. $S \subseteq P(\Lambda)$ is *closed* if S consists only of saturated sets, is closed under $\cap$, $\to$, and is chain-complete for inclusion. It is *strongly closed* if it is furthermore closed under $X \mapsto (\Lambda \to X)$.

Example 111. S^l is closed and S^h is strongly closed (Lemmas 87 and 94).

Proposition 112. *Let (D,i) be the completion of the* positive *web (A,j).*

(i) *If S is closed and S^A is closed under H, then there is an interpretation I of (D,i) such that $I(\alpha) \in S$ for all $\alpha \in A$.*
(ii) *If S is strongly closed (D,i) is S-interpretable.*
(iii) *In both cases these interpretations are adequate for the model of web (D,i).*

Proof. (i) Owing to Proposition 107, it is enough to prove that there is an S-interpretation J of (A,j). Since S is chain complete for inclusion and closed under all intersections, S^A is chain complete for all B-inclusion orders. Hence H has a fixed point $\vec{J}$ in S, and J can be defined by $J(\alpha) \equiv J_\alpha$.

(ii) follows from (i) once remarked that, if S is strongly closed, then S^A is invariant under H and $I(\alpha) \in S$ for all $\alpha \in D$.

(iii) Follows from Proposition 106.

Corollary 113. *If $\mathscr{G}$ is the completion of a positive pair, then $\mathscr{G}$ is sensible and moreover $\mathscr{G} \models t \neq \emptyset$ iff t is solvable.*

Proof. Since S^h is strongly closed it follows from the proposition above that $\mathscr{G}$ is adequate for some S^h-interpretation of its web. Then the proof ends as for Engeler's model.

Example 114 (*Examples of positive pairs*).
1. $\mathscr{E}$ is the completion of $(A,\emptyset)$ and generates $H(\vec{X})\equiv\vec{X}$.
2. P_ω corresponds to $A\equiv\{p\}$, $j(\emptyset,p)\equiv p$, $H(X)\equiv\Lambda\to X$.
3. $A\equiv\{p,q\}$ and $j(\{p\},q)\equiv q$, with $H(X,Y)\equiv\{X,X\to Y\}$.
4. Another example will be given in Example 140 below.

Example 115 (*Example of a non-positive pair*). A typical example is $A\equiv\{\alpha\}$ and j defined by $j(\{\alpha\},\alpha)\equiv\alpha$, since the function $H(X)\equiv X\to X$ is not monotone, neither w.r.t. inclusion or reverse inclusion. This is the case of Park's model in Example 133.

Remark 14. It is rather easy to build 2^ω non-equationally equivalent graph models and Corollary 113 can be used to build 2^ω non-isomorphic sensible graph models [66, 69]. Whether one can build 2^ω *sensible* graph models *with* different equational theories is a more difficult question which is discussed in Section 6.3.

**Forcing*. For proving the easiness of Ω (Proposition 50), Baeten and Boerbom [6] constructed for any t a graph model satisfying $\Omega=t$, by building step by step an injection i (over a given countable set D), forcing at each step the "unavoidable" elements of $|t|$ to be in $|\Omega|$. The core of the method was extracted in [115], where the easiness of other unsolvable terms is proved, and was then used by several authors to prove consistency and incompleteness results (see Sections 6.1, 6.3, 6.8). Thus forcing amounts to completing a pair $(D,\emptyset)$ into a pair (D,i), in an interactive way.

**The l-variation of graph models*. When i is not onto we can vary the definition of G in order to obtain other models of λ-calculus, which will happen to be *strongly lazy* if we start from a positive pair. We keep the same domain and the same application A, but replace G by G' defined by

$$G'(h)\equiv G(h)\cup B,$$

where B is any given subset of $D-range(i)$.

**Reducibility for l-graph models*. For dealing with the l-variation we need to work with *l-interpretations*, which are usual interpretations I satisfying furthermore $\{\lambda x.u\mid u\in\Lambda\}\subseteq I(f)$, for all $f\in B$. Then a straightforward variation of the proofs given for ordinary graph models gives that:
1. Any l-interpretations of an l-model is adequate for this model.
2. If (D,i) is the completion of a positive pair, then the l-graph model over (D,i) is S_h^l-interpretable, where S_h^l is defined similarly as S_h but starting from $\mathscr{N}_h^l\equiv\{t\in\Lambda\mid\exists x\ \exists u\ t\to_h\lambda x.u$ or $\exists x\in V\exists\vec{u}\in\Lambda^{<\omega}\ t\to_h x\vec{u}\}$.
   (note that S_h^l is strongly closed).

3. If an l-graph model is S^l_h-interpretable, then it is strongly lazy. Furthermore,
   (a) $|t|_\rho = \emptyset$ iff t is unsolvable of (functionality) order 0.
   (b) t has functionality order $\geqslant n$ iff $a_n(f) \in |t|_\rho$ for some (all) $f \in B$, where $a_0(f) \equiv f$ and $a_{n+1}(f) \equiv i(\emptyset, a_n(f))$.

Example 116 (**Longo's lazy variant $\mathscr{L}$ of Engeler's model* [86]). $\mathscr{L}$ is the simplest model(s) of this kind, being generated by the positive pair $(A, \emptyset)$, $A \neq \emptyset$, and an arbitrary choice of B. $\mathscr{L}$ is strongly lazy and satisfies (a) and (b) above. In this case it is even easy to compute the exact value of all unsolvables.

5.6. Other models built from webs

5.6.1. Common features

We will continue our bottom-up approach in the next subsection. Here we present the general common features of the classes, in order to decrease the possible impression that we are doing zoology. Graph models correspond to the case were the binary relations introduced below are trivial.

Terminology

Definition 117. A *preorder* $\leqslant$ on D is a reflexive and transitive binary relation on D. We denote by $\downarrow e$ the *initial segment* generated by $e \subseteq D$, namely

$$\downarrow e \equiv \{\alpha \in D \mid \exists \beta \in e \ \ \alpha \leqslant \beta\}.$$

Note that the reflexive closure $\prec^r$ of a transitive relation is a preorder.

Definition 118. A *coherence relation* $\frown$ on D is a reflexive and symmetric binary relation on D. A subset d of D is "*coherent*" if $x \frown y$ for all $x, y \in d$. Finally, d and d' are *compatible* or *coherent* if $d \cup d'$ is coherent.

$P_{\mathrm{coh}}(D)$ and $P^{<\omega}_{\mathrm{coh}}(D)$ denote the set of coherent (resp. finite and coherent) subsets of D.

Definition 119. We say that $\leqslant$ and $\frown$ are *compatible* if $\alpha \frown \beta, \alpha' \leqslant \alpha$, and $\beta' \leqslant \beta$ imply $\alpha' \frown \beta'$. An equivalent formulation is: the relations are *compatible* if the initial segment generated by a coherent subset of D is coherent.

$S_{\mathrm{coh}}(D, \leqslant, \frown)$ denotes the set of coherent initial segments of D.

Preordered coherent spaces. The domains underlying all the models we are interested in are all[36] built from the following triples $(D, \leqslant, \frown)$.

Definition 120. A *preordered set with coherence* (pcs for short) is a triple $(D, \leqslant, \frown)$, where D a non-empty set, $\leqslant$ is a preorder on D and $\frown$ is a coherence relation on

[36] Except for H-models.

D, both being compatible. This pcs will be viewed as the web generating the domain $S_{\mathrm{coh}}(D, \leqslant, \asymp)$, or simply the *web of this domain*.

One or both of these relations *can be trivial*. This means here that $\leqslant$ is equality or/and that $\asymp$ is $D \times D$. Then they will be omitted. We are not interested in the other cases of triviality, since they would lead to trivial models (singletons). When the preorder is trivial then the pcs is simply called a *coherence space.*

A pcs generates a domain of the form.

$$(M, \sqsubseteq) \equiv (Q(D), \subseteq) \quad \text{with } Q(D) \equiv S_{\mathrm{coh}}(D, \leqslant, \asymp)$$

Some closure and non-closure properties of pcs. Whatever the semantics we are working with, adding a preorder or a coherence relation, and both if necessary, gives more power to express and solve domain equations. For example, the class of domains generated by pcs is closed under *direct* (*or coalesced*) *sum* $\oplus$, under *lifting*, and under *lifted* (*or separated*) *sum*, while it is not necessarily the case with subclasses. *Let* $\uplus$ denote *disjoint union* of sets.

Definition 121. The *coalesced sum* $M \oplus N$ of two domains consists in taking the disjoint union of the domains, except for the bottom elements, which are identified. Then $d \sqsubseteq d'$ iff d, d' are in the same component and $d \sqsubseteq d'$ in this component.

Lifting $M \mapsto M_\perp$ is the operation which consists in adding a new bottom element below M.

The *separated sum* $M \oplus_\perp N$ takes the disjoint union of the domains and adds a new bottom element below (and $d \sqsubseteq d'$ iff $d = \perp$ or same as for $\oplus$).

Now, the domains associated to pcs with trivial coherences are complete lattices, and hence are not closed under $\oplus$ and $\oplus_\perp$. Similarly, restricting to pcs with trivial preorders forbids lifting (and $\oplus_\perp$). This is clear from the fact that, at the level of webs, things work as follows:

Assume $(D, \leqslant, \asymp)$ and $(D', \leqslant', \asymp')$ are the webs of M and M'. Then

- the web of $M_\perp$ is: $[D \uplus \{f\}, (\leqslant \bigcup \{(f, \alpha) \mid \alpha \in D\}^r), (\asymp_{\mathrm{coh}} \bigcup \{(f, \alpha) / \alpha \in D\}^r)]$
- the web of $M \oplus N$ is $[D \uplus D', (\leqslant \uplus \leqslant')^r, \asymp \uplus \asymp']$, and $\asymp \uplus \asymp'$ is never trivial.
- the web of $M \oplus_\perp N$ is $[D \uplus D' \uplus \{f\}, (\leqslant \uplus \leqslant' \bigcup \{(f, \alpha) / \alpha \in D \uplus D'\}^r), (\asymp \uplus \asymp' \bigcup \{(f, \alpha) \mid \alpha \in D\})^r]$.

Models. The *models* we are interested in will all have a web of the form $(D, \leqslant, \asymp, i)$, where $i : (P^{<\omega}_{\mathrm{coh}}(D) \times D) \to D$ is an injection, which will allow us to encode the representable functions via their traces. More precisely, the main models will be of the form (M, A, G) with M the domain of web $(D, \leqslant, \asymp)$ and $A \equiv Tr^{-1} \circ \downarrow \circ i^-$ and $G \equiv \downarrow \circ i^+ \circ Tr$.[37] Here $i^+(u) \equiv \{ i(x) / x \in u \}$ and $i^-(u) \equiv \{ x / i(x) \in u \}$ and Tr is a notion of trace which depends on the semantics. Slight variations of A and G will

[37] The "$\downarrow$" in the definition of A corresponds to the following poreorder on $P^{<\omega}_{\mathrm{coh}}(D) \times D$: $(a, \alpha) \leqslant (b, \beta)$ iff $\downarrow b \subseteq \downarrow a$ and $\alpha \leqslant \beta$.

however sometimes be possible and will give further models; in this case the same web can give rise to several models. In fact, compatibility conditions on i w.r.t. $\leqslant$ and $\asymp$ will be stated in order to ensure that G, A are well defined and suitable. These compatibility conditions will hence depend on the semantics we are working in. For the continuous semantics they will be given in full generality in Section 5.6.4; we do not state them now, since we prefer that the reader meets before the simpler versions which occur when one relation is trivial.

Definition 122 (*Informal*). A tuple $(D, \leqslant, \asymp, i)$, where $(D, \leqslant, \asymp)$ is a pcs and i a compatible injection, will be considered as the *web* of the model(s) it generates.

**Logical reading of* (*some of*) *the webs*. There is a logical interpretation of each web $(D, \leqslant, \asymp, i)$ as an "*intersection type system*", which makes sense when this web is the completion of a recursive partial web $(A, \prec, \asymp_A, j)$.

The elements of A are then viewed as *propositional variables* and $\emptyset$ is understood as a propositional constant.

The elements of D are viewed as *propositional formulas* built from propositional variables, using implication and a kind of conjunction. More precisely, a, if non-empty, is understood as the *conjunction* of its elements, and $i(a, \alpha)$ is written $a \to \alpha$, and understood as an *implication*. Thus, conjunction is set-theoretic like, and furthermore it is allowed only on coherent sets of formulas. In this setting $\alpha \leqslant \beta$ has to be understood as "α is less informative than β".

As soon as $dom(j)$ is non-empty, the elements of A are supposed to satisfy a system of recursive equations, which is the logical translation of the system $\vec{X} = H(\vec{X})$ associated to (A, j).

Then a system of logical rules is established for deriving expressions ("sequents") of the form $\Gamma \vdash x : \alpha$, to be read "t is of type α in the context Γ", where $\Gamma \equiv x_1 : a_1, \ldots, x_n : a_n$, in such a way that one can derive the sequent $x_1 : a_1, \ldots, x_n : a_n \vdash t : \alpha$ exactly in case $\alpha \in |t|_{[x_1 := a_1, \ldots, x_n : a_n]}$. As usual the x_i's are supposed to be distinct.

This view is similar to the one introduced Dezani et al. [32] and later on systematized in [32], but it is technically more simple since we work with *strict* intersection type systems, that is: no conjunction, even empty, is allowed on the right-hand side of the implication.

The simplest of all is System E below,[38] which corresponds to Engeler's model. It is the essence of all other systems.

Example 123 (* *System E*). A context $\Gamma \equiv x_1 : a_1, \ldots, x_n : a_n$ is viewed as a partial function: $V \to P^{<\omega}(D)$. The following is understood to hold for all $\Gamma, t, u, v, a, \alpha, \beta$:
(1) (axiome/variable) $\alpha \in \Gamma(x) \Longrightarrow \Gamma \vdash x : \alpha$

[38] System E coincides with the restriction of Dezani's intersection type system which is presented in [7] as "the strict intersection type system".

(2) (→-intro/abstraction) $\Gamma, x:a \vdash t:\alpha \Longrightarrow \Gamma \vdash \lambda x.t : a \to \alpha$

(3) (→-élim/application) $\left.\begin{array}{l}\Gamma \vdash u : a \to \alpha \\ \Gamma \vdash v : a\end{array}\right\} \Longrightarrow \Gamma \vdash (u)v : \alpha$

where $\Gamma \vdash v : a$ is an abbreviation for $\Gamma \vdash v : \beta$ for all $\beta \in a$, and is hence the empty condition if $a = \emptyset$.

Example 124 (**The system corresponding to Longo's definition of* P_ω). Just add, for the atom p and for all Γ, t, the two rules (summarized in one):

(4) and (5) $\Gamma \vdash t : p \Longleftrightarrow \Gamma \vdash t : \emptyset \to p$

corresponding to the equation $p = \emptyset \to p$.

*The systems axiomatizing graph models can easily be interpolated from the above one. In the case where $(\leqslant, \asymp)$ is non-trivial things are a little more delicate since one must also give rules which axiomatize these relations (and since we have to restrict to coherent subsets a). Finally if one leaves the continuous semantics, then rules (1) and (3) have to be modified.

The correspondence between the logical and the algebraic views of the models is a Stone-like duality [1].

**Algebraic reading of* (*all*) *the webs*. The domains which can be described by pcs webs are the "*binary prime algebraic domains*", as defined in Section 5.6.7 later on. Given such a domain $\mathscr{D}$, there is a natural underlying pcs, namely $(\mathscr{D}_p, \leqslant_p, \asymp_p)$ where $\mathscr{D}_p$ is the set of prime elements of $\mathscr{D}$ (cf. Section 5.6.7), $\leqslant_p$ is the restriction of the order of $\mathscr{D}$ to $\mathscr{D}_p$, and $x \asymp_p y$ if x, y are upper-bounded in $\mathscr{D}$. If $\mathscr{D} \equiv S_{\mathrm{coh}}(D, \leqslant, \asymp)$, then $\alpha \mapsto \downarrow\{\alpha\}$ is an isomorphism between $(D, \leqslant / \sim, \asymp / \sim)$ and $(\mathscr{D}_p, \leqslant_p, \asymp_p)$, where $\sim$ is the equivalence relation associated to the preorder $\leqslant$.

The compatibility conditions between i, the coherence and the preorder relations reflect in fact the relations between the structured set of the prime elements of the model and that of its space of representable functions.[39]

Plan of the rest of the section. We return now to specific classes of models. We begin with the two ones which have a trivial relation: K-models for the continuous semantics (trivial coherence) and G-models for the stable semantics (trivial preorders). Then we present the pcs-models of the continuous semantics (pcs-models of the stable semantics exist, but we will only describe them in two lines, since we will not use them here). We then briefly describe the κ-pcs models, which we need for modelling Map Theory, and the H-models, which belong to the strongly stable semantics. And finally, for the continuous semantics, we make comparison with more traditional and more permissive classes of models.

5.6.2. K-models

This class of models, which contains all graph models, was isolated by Krivine [76] within the continuous semantics. The class contains many extensional models;

[39] This correspondence is fully developed in the preliminaries of [18].

in particular it contains Scott's $\mathscr{D}_\infty$. The domains underlying K-models are complete lattices, and even prime algebraic complete lattices (as defined in Section 5.6.7).

Definition of a K-model. We work here with preordered sets $(D, \leqslant)$. The underlying domains will be the power sets $S(D)$ consisting of all *initial segments* of $(D, \leqslant)$, and hence will be complete lattices.

Definition 125. A *K-web* is a triple $(D, \leqslant, i)$, where $(D, \leqslant)$ is a preorder and $i : P^{<\omega}(D) \times D \to D$ is an injection which are subject to the compatibility property

$$\forall a, b, \alpha, \beta \quad i(a,\alpha) \leqslant i(b,\beta) \Longrightarrow (\downarrow b \subseteq \downarrow a \text{ and } \alpha \leqslant \beta). \qquad (C_\beta)$$

Definition 126. A *K-model* is a model generated by a K-web $(D, \leqslant, i)$ as follows:

$$\mathscr{K} \equiv (M, A, G) \quad \text{with } M \equiv (S(D), \subseteq),$$

where application, and hence A, is defined by

$$dd' \equiv \downarrow\{\alpha \in D \mid \exists a \in P^{<\omega}(D)\ a \subseteq d'\ i(a,\alpha) \in d\}$$

and for any continuous $h : M \to M$,

$$G(h) \equiv \downarrow\{i(a,\alpha) \mid \alpha \in h(\downarrow a)\}$$

It is easy to check that "application" is continuous, and that the representable functions are exactly the continuous ones.[40]

Extensional K-models and K-webs. A necessary and sufficient condition to get $G \circ A = id$ and hence an *extensional model* is that:

(1) the implication in (C_β) is an equivalence, and
(2) every $\alpha \in D$ is equivalent to an element in the range of i (for the equivalence relation induced on D by the preorder).[41]

In this case we will also speak of *extensional K-webs.*

Then the definitions of dd' and $G(f)$ can be simplified: in both cases the external $\downarrow$ is redundant and thus can be removed.

The conditions for ensuring extensionality are rather easy to fulfill, as the next proposition shows.

Notations. For R a preorder on D we define R^* on $D^{<\omega}$ by aR^*b iff $\forall \alpha \in a\ \exists \beta \in b$ such that $\alpha R \beta$, and we define $L(R)$ on D by $\alpha L(R) \beta$ iff $\alpha = i(a', \alpha')$ and $\beta = i(b', \beta')$ and $a' R^* b'$ and $\alpha' R \beta'$.

R^* is a preorder and $L(R)$ is transitive; if i is bijective then $L(R)$ is a preorder.

Proposition 127 (Krivine [77]; Extensional completion of a graph model). *Any graph model (D, i) such that i is bijective can be canonically endowed into an extensional K-web $(D, i, \leqslant)$. Moreover, we can even require that $\leqslant$ contains any given*

[40] All this can easily be directly checked, or viewed as a consequence of the fact that complete lattices and continuous maps form a c.c.c.

[41] This is left as an exercise, and is worked out in [76] or [77].

transitive relation $\prec$ *on* D *which is such that* $\prec \subseteq L(\prec^r)$ *and* $i(\emptyset,\alpha) \not\prec i(\{\alpha\},\alpha)$ *for some* $\alpha \in D$.

Proof. Since i is bijective $L(R)$ is a preorder. The set of preorders on D is a cpo and L is continuous on it, so L has a least fixed point on it, namely "$\leqslant$" $\equiv \bigcup \{ L^n (\prec^r) \mid n \in N \}$. Since $\leqslant$ is a fixed point of L and i is bijective we get an extensional model. This model is non-trivial since $i(\{\alpha\}, i(\emptyset,\alpha))$ and $i(\emptyset, i(\{\alpha\},\alpha))$ are incomparable (this is trivial for all α if $\prec \equiv \emptyset$, and follows from the restriction on $\prec$ in the general case).

The reducibility method for K-models

Definition 128. An *interpretation of* $(D,i,\prec)$, for a transitive relation $\prec$ on D, is an interpretation I of (D,i)[42] such that $\alpha \prec \beta$ implies $I(\beta) \subseteq I(\alpha)$, for all $\alpha,\beta \in D$.

The need for such a reverse inclusion property comes naturally from the extension of the proof of the adequacy lemma to the preordered case. Furthermore, it is natural if one thinks of "α" as a property about terms, of "$\alpha \prec \beta$" as "β is more informative, or stronger, than α" and of "$I(\alpha)$" as the set of terms which have property α.

The proof of the following proposition is a straightforward variation of that of Proposition 90, and it is left to the reader.

Proposition 129 (Semantic adequacy). *Let* I *be an interpretation of the K-web* $(D,i,\leqslant)$, *then* I *is adequate for the corresponding K-model, in the sense of Proposition* 90.

Proposition 130 (Reducibility transfer). *Let* I *be an interpretation of* $(D,i,\prec)$. *Suppose that* i *is bijective. Then* I *is also an interpretation of the extensional model* $\mathscr{M} \equiv (D,i,\leqslant)$ *associated to* $(D,i,\prec)$ *by Proposition* 127.

Proof. It is easy to show, by induction on the smallest n such that $(\alpha,\beta) \in L^n(\prec^r)$, that $\alpha \leqslant \beta$ implies $I(\beta) \subseteq I(\alpha)$.

Corollary 131 (Sensibility transfer). *If* $(D,i,\prec)$ *is* S_h*-interpretable and* i *is bijective, then its extensional completion is also* S_h*-interpretable and hence generates a sensible model.*

Example 132 (*Scott's* $\mathscr{D}_\infty$). Scott's $\mathscr{D}_\infty$-family covers all the extensional models canonically associated to the graph models $\mathscr{P}$ of Example 101, using Proposition 127. The simplest one is the one associated to P_ω. These models are sensible (by Corollary 131), and we can even start from a preordered set A of atoms, because of Proposition 127.

[42] Compare Definition 105.

Example 133 (*Park's* $\mathscr{P}_\infty$). Park's $\mathscr{P}_\infty$ is built in the same way as $\mathscr{D}_\infty$ except that we start from $i(\{\alpha\},\alpha)=\alpha$ for all $\alpha\in A$ instead of $i(\emptyset,\alpha)=\alpha$. We already noticed that the reducibility argument did not apply in this case, and in fact the model is not even semi-sensible.

Partial K-webs

Definition 134. A *partial K-web* is a triple $(A,j,\prec)$ where (A,j) is a partial pair and $\prec$ is a transitive relation on A. The *interpretation of a partial K-web* is defined as for K-webs (Definition 128).

To a partial K-web is canonically associated a pair (H,E) where H is the operator associated to (A,j) in Definition 108 and

$$E\equiv\{\,\boldsymbol{X}\in P(\Lambda)^A \mid X_\beta\subseteq X_\alpha \text{ for all } \alpha,\beta\in A \text{ s.t. } \alpha\prec\beta\}.$$

It is easy to check that E is chain complete for all B-inclusion orders.[43]

Definition 135.[44] A partial K-web $(A,j,\prec)$ is *positive* if (A,j) is positive and H acts on E.

Proposition 136. *Suppose* $(A,j,\prec)$ *is positive and* $S\subseteq P(\Lambda)$. *If* S *is closed and* S^A *is invariant under* H, *then* $(A,j,\prec)$ *is* S*-interpretable.*

Proof. $E'\equiv E\cap S^A$ is chain complete for all B-inclusion orders, since both E and S^A are. Owing to the positivity hypothesis H acts on S^A and E, hence on E', and H is monotone for some B-inclusion order. Hence H has a fixed point $\vec{J}$ in E', and J can be defined by $J(\alpha)\equiv J_\alpha$.

Extensional completion of partial K-webs

Definition 137. The partial K-web $(A,j,\prec)$ is extensionalizable if
1. $A=range(j)$.
2. For some $\alpha\in A$, $(\emptyset,\alpha)$ or $(\{\alpha\},\alpha)$ are not in $dom(j)$, or $j(\emptyset,\alpha)\not\prec j(\{\alpha\},\alpha)$.
3. $\prec\subseteq L(\prec\cup=_A)$, where L is defined as in the proof of Proposition 127.

Proposition 138 (Extensional completion of a partial web). *Suppose* $(A,j,\prec)$ *is extensionalizable, then there is a canonical way to extend it to an extensional K-web* $(D,i,\leqslant)$. *This K-web* (*and the K-model it generates*) *is called the extensional completion of* $(A,j,\prec)$, *and it is also the extensional completion of* $(D,i,\prec)$.

[43] Hint: For each pair $\alpha\prec\beta$ there are four cases to consider, according that α and β are in B or not. Only the cases $\alpha\in B$ and $\beta\notin B$ use that we are working with a chain of E.

[44] This presentation generalizes the reducibility argument given for a specific webbed model by Honsell and Lenisa [57].

Proof. Let (D,i) be the completion of (A,j) (Definition 104). The first condition forces $D = range(i)$. Then it suffices to apply Proposition 127. Condition 2. forces $\leqslant$ not to be $D \times D$.

Proposition 139. *Suppose $(A,j,\prec)$ is extensionalisable. Then any interpretation J of $(A,j,\prec)$ extends uniquely to an interpretation of its extensional completion.*

Proof. By Proposition 107, J extends to an interpretation I of (D,i), which is obviously an interpretation of $(D,i,\prec)$. By Proposition 130, I is an interpretation of $(D,i,\leqslant)$.

Example 140 (*Coppo, Dezani and Zachi's model Norm* [33]). Norm is by definition the extensional completion of $(A,j,\prec)$, where: $A = \{p,q\}$, $p \prec q$, and j is defined by $j(\{p\},q) \equiv q$, $j(\{q\},p) \equiv p$.

It is indeed easy to check that this partial web is extensionalizable.

Normalization revisited. The model *Norm* is sensible and allows to prove diverse normalization theorems. A gain is that we do not have to bother about strict formulas. The cost is that we work with a non-trivial preorder here, but a second gain is that, working with an extensional model, we get for free the $\beta\eta$-normalization theorem (which states that any (β-)normalizable term is $\beta\eta$-normalizable).

Notation. $|t|_q$ denotes the interpretation of t in the environment which gives the value $\downarrow\{q\}$ to all variables.

Lemma 141. *The web generating Norm is positive.*

Proof. The pair (H,E) is defined by $H(X,Y) \equiv (Y \to X, X \to Y)$ and $E \equiv \{(X,Y) \mid Y \subseteq X\}$. It is then easy to check that H acts on E and is monotone for the $\{q\}$-inclusion order, whose explicit definition is

$$(X,Y) \sqsubseteq (X',Y') \text{ iff } X' \subseteq X \text{ and } Y \subseteq Y'.$$

Corollary 142. (i) *Norm is S_h-interpretable*

(ii) *There is an interpretation of Norm such that $I(p)$ and $I(q)$ are in S^l.*

Proof. Recall that S_l is closed and S_h is strongly closed. By Proposition 136 ($\{p,q\},j,\prec$) is S_h- and S_l-interpretable. By Proposition 139 these interpretations extend to interpretations of the total web of *Norm*. Furthermore in the h-case the interpretation is an S_h-interpretation, since S_h is strongly closed.

Proposition 143. *The following are true for Norm*:

(a) *t is head-normalizable iff Norm $\models t \neq \emptyset$ iff t is solvable.*

(b) *t is left-normalizable iff $p \in |t|_q$ iff t is β-normalizable iff t is $\beta\eta$-normalizable.*

Proof. (a) The proof is the same as for Engeler's model, starting from (i) of the Corollary 142. Alternatively, one could use Corollaries 113 and 131.

(b) Suppose $p \in |t|_q$; then $p \in |t[\vec{x} := \overline{\downarrow\{q\}}]|$. Now $\forall i \leqslant l(\vec{x})\ x_i \in N_l^0 \subseteq I_l(q)$; hence (semantic adequacy) $t = t[\vec{x} := \vec{x}] \in I_l(p) \subseteq N_l$ and t is left-normalizable.

We show now that normalizable and $\beta\eta$-normalizable terms are such that $p \in |t|_q$. As a matter of fact, since the model is extensional, the interpretation of a term does not depend on its $\beta\eta$-equivalence class, so it is sufficient to prove it for (β-)normal terms. This follows from an easy induction on the structure of the term, and is the only point where one uses the specific definition of the web.

Remark 15. The theory of Norm is in fact completely known and Norm is fully abstract w.r.t. the strategy S^n. Once more this is proved using an approximation theorem.

**l-variations of K-models.* They can be defined as for graph models, replacing G by G', where $G'(h) \equiv G(h) \cup \downarrow B$ for any fixed B such that $\downarrow B \subseteq D - range(i)$. If $D - range(i)$ contains a minimal element f, then one can take $B \equiv \{f\}$. It is easy to check that in this case $G'(h) = G(h)$ if $h \neq d \mapsto \emptyset$ and $G'(d \mapsto \emptyset) = \{f\}$.

We then have a notion of *l-interpretation of a K-web*, similar to the l-interpretations defined at the end of Section 5.5, except that we take the preorder into account. Similar results as the ones stated there are then valid. In particular *positive partial webs generate strongly lazy models.*

**Quasi-extensional models and laziness*

Definition 144. A *quasi-extensional model* is a model whose underlying domain satisfies $M \simeq R(M)_\perp$.

Extensional models, i.e. models whose underlying domain satisfies $M \simeq R(M)$, can be viewed canonically as a model of λ-calculus, and there is only one way to do it.[45] Moreover, the resulting model is not lazy.[46] By contrast, the domains which are solutions of $M \simeq R(M)_\perp$ give rise canonically to two models of λ-calculus.[47] In the K-framework, quasi-extensional models correspond to K-webs $(D, i, \leqslant)$ such that $D = \{f\} \cup range(i)$, for f a minimum element of $(D, \leqslant)$. Such a web will also be called *quasi-extensional.*

Proposition 145 (*Quasi-extensional completions of a graph model). *Any graph model* (D, i) *such that* $card(D - range(i)) = 1$ *can be canonically endowed into a quasi-extensional K-web.*

[45] More accurately, each pair of inverse isomorphisms between M and $R(M)$ gives rise to one model of λ-calculus.

[46] If t has order 0 (e.g. $t \equiv \Omega$), then $\lambda x.tx$ has order 1. Both are however equated in all extensional models.

[47] Such quasi-extensional M obviously have a minimum element d_0 above $\perp$, and we have the freedom of taking $G(x \mapsto \perp) \equiv \perp$ or d_0. Also one should define $A(\perp) = \perp$. But for these two cases, (A, G) coincides with the isomorphism pair of the domain equation.

Proof. Define L' on preorders on D by $L'(R) \equiv L(R) \cup \{ (f,\alpha) \mid \alpha \in D \}$, where f is the element not in $range(i)$, and take $\leqslant$ as the smallest fixed point of L'. Obviously f is the smallest element of D for $\leqslant$. Finally, it is easy to check that the domain generated by $(D,i,\leqslant)$ satisfies $M \simeq R(M)_\perp$.

**Quasi-extensional completion of a partial web.* Proposition 137 can be slightly modified for giving the possibility of getting quasi-extensional models from "quasi-extensionalizable" partial webs. It is indeed easy to check that it is enough to start from a partial web $(A,j,\prec)$ such that 1^q. $A = range(j) \cup \{f\}$, 2. unchanged, and 3^q. $\prec \subseteq L'(\prec^r)$, L' as above. Then the quasi-extensional completion is defined as the K-web $(D,i,\leqslant)$ where (D,i) is the graph completion of (A,j) and $\leqslant \equiv \bigcup L'^n(\prec)$. If the partial web is positive then the corresponding models are respectively sensible and strongly lazy.

Example 146. *The Abramsky-Ong's model for lazy λ-calculus and its parallel K-model* ($\mathscr{A}\mathscr{O}$ and $\mathscr{K}$) are defined as the quasi-extensional models arising from the simplest possible partial K-web, namely $(\{f\},\emptyset,\emptyset)$, which is obviously positive.

Example 147 (*Explicit description of $\mathscr{A}\mathscr{O}$ and $\mathscr{K}$*).

1. *Common features.* D is the smallest fixed point of the set-theoretic equation

   $$D = (P^{<\omega}(D) \times D) \cup \{f\},$$

   i is the inclusion map.
   The preorder on D is defined inductively by

   $$f \leqslant \alpha \quad \text{for all } \alpha$$

   $$(a,\alpha) \leqslant (b,\beta) \text{ iff } (b \geqslant^* a \text{ and } \alpha \leqslant \beta).$$

   The domain, $\mathscr{D} \equiv S(D,\leqslant)$, is the simplest solution of the domain equation $M \simeq R(M)_\perp$ in the category of cpo's and continuous functions.
   The "application" A, is defined by

   $$d d' \equiv \{\alpha \mid \exists a \subseteq d' \ (a,\alpha) \in d\}$$

2. *Differences.* For $\mathscr{A}\mathscr{O}$ the continuous functions are encoded by the lazy version G', which simplifies here into

   $$G'(h) \equiv \{(a,\alpha) \mid \alpha \in h(\downarrow a)\} \cup \{f\} = Tr(h) \cup \{f\}$$

   The reducibility method applies in both cases and proves respectively that $\mathscr{K}$ is sensible and $\mathscr{A}\mathscr{O}$ is strongly lazy. Moreover, the exact value of unsolvables can be easily computed (and all unsolvable of order 0 are equated to $\emptyset$).

Remark 16. (1) $\mathscr{K}$ is in fact the simplest K-model which is not a graph model, and similarly $\mathscr{A}\mathscr{O}$ is the simplest l-K-model which is not an l-graph model.

(2) $\mathscr{A}\mathscr{O}$ and $\mathscr{K}$ are also the quasi-extensional completions of $\mathscr{L}$ and $\mathscr{E}$, respectively.

5.6.3. G-models

This class belongs to the *stable semantics*, and hence allows further refutation arguments. It was isolated by Girard [49].

Compared to the models studied so far, G-models are as simple to manipulate as graph models, and the interpretation of terms is more economical, because the encoding of stable functions is done via a more economical notion of trace. Another advantage of the class is that we have extensional models without having to introduce a preorder. On the other hand, the completion process is slightly more delicate with G-models, because of the presence of the coherence relation.

The domains we are interested in are those generated by coherent spaces $(D, \frown)$, and are hence of the form $P_{\mathrm{coh}}(D)$. In this setting:

Definition 148. A *stable function* on $P_{\mathrm{coh}}(D)$ is a continuous function such that $f(d \cap d') = f(d) \cap f(d')$ whenever d and d' are coherent.

It is easy to see that if f stable and $\alpha \in f(d)$, then there is a unique finite and minimal $b \subseteq d$ such that $\alpha \in f(b)$. In particular, a stable function can be recovered from the following subset of its continuous trace:

$$Tr_s(f) \equiv \{(a, \alpha) \mid \alpha \in f(a) \text{ and } a \text{ is minimal such that } \alpha \in f(a)\}$$

(here of course the metavariables a, b range over $P_{\mathrm{coh}}^{<\omega}(D)$).

This motivates the definition of $G(f)$ below. Stable traces of stable functions are less redundant than the usual ones: for example $Tr_s(id) \equiv \{(\{\alpha\}, \alpha) \mid \alpha \in D\}$. One can note that the p.o. set of stable traces ordered by inclusion is isomorphic to the p.o. set of stable functions ordered by the stable order mentioned in Section 4.5 [49, 77].

Definition 149. A *G-web* is a triple $(D, \frown, i)$ where $(D, \frown)$ is a coherent space, $i : P_{\mathrm{coh}}^{<\omega}(D) \times D \to D$ is an injection and i and $\frown$ satisfy the following compatibility condition, which will allow to embed in $P_{\mathrm{coh}}(D)$ its p.o. set of stable functions

$$i(a, \alpha) \frown i(b, \beta) \text{ iff } \begin{cases} a, b \text{ coherent} \Rightarrow \alpha \frown \beta \\ a, b \text{ coherent and } a \neq b \Rightarrow \alpha \neq \beta \quad (C_\beta) \end{cases}$$

Definition 150. A *G-model* is a model generated by a G-web $(D, \frown, i)$ as follows:

$$\mathscr{M} = (M, A, G) \quad \text{with } M \equiv (P_{\mathrm{coh}}(D), \subseteq)$$

where $G \equiv i^+ \circ Tr_s$, and application is defined by

$$dd' \equiv \{\alpha \in D \mid \exists a \in P_{\mathrm{coh}}^{<\omega}(D)\ a \subseteq d'\ i(a, \alpha) \in d\}$$

Thus, for any stable $h : M \to M$,

$$G(f) \equiv \{i(a, \alpha) \mid \alpha \in h(a) \text{ and } a \text{ is minimal such that } \alpha \in h(a)\}.$$

It is easy to check directly that "application" is stable, that $A \circ G = id$, and that the representable functions are exactly the stable ones; this extends to n-ary functions like $x, y \mapsto mxy$.

It is also easy to check that in any G-model $|K| = |T| = \{i(\{\alpha\}, i(\emptyset, \alpha)) \mid \alpha \in D\}$, while $|F| = \{i(\emptyset, i(\{\alpha\}, \alpha)) \mid \alpha \in D\}$. In particular, $|T|$ and $|F|$ are incompatible and a G-model cannot be a lattice; furthermore $|T| \cap |F| = \emptyset$.

Example 151. Stable versions $\mathscr{E}^s$, $\mathscr{D}^s_\infty$ and $\mathscr{P}^s_\infty$ of $\mathscr{E}$, $\mathscr{D}_\infty$ and $\mathscr{P}_\infty$ can be built along the same lines as their analogues of the continuous semantics. However, one should note that for $\mathscr{E}^s$ and $\mathscr{D}^s_\infty$ one can start from an arbitrary coherence on the set of atoms and keep this coherence during the completion process, while for $\mathscr{P}^s_\infty$ the completion process will make all atoms coherent at the end.[48] Thus, there is only one $\mathscr{P}^s_\infty$ (once the size of the set of atoms has been fixed) and infinitely many variations of $\mathscr{E}^s$ and $\mathscr{D}^s_\infty$.

More generally, the completion process can be applied, but one has first to define a notion of *completable* partial web.[49]

Once more variations of the reducibility argument apply to this kind of models (cf. [57]) and we can in particular deduce that $\mathscr{D}^s_\infty$ is sensible and interprets all unsolvable terms (and only them) by $\perp = \emptyset$. For an alternative proof using "approximation arguments" see [58, 52, 51], and also [58] for another interesting example of stable model. For treating more general cases one should define a notion of *positive completable partial web*.

The forcing method also applies [68, 71].

The standard example of a refutation argument via the stable semantics is the refutation of the existence of the "*parallel or*" below (and its variations). Such a "parallel or" cannot be eliminated using the continuous semantics, since any continuous model do contain "parallel functions".

A refutation argument via the stable semantics. We claim: there is no closed λ-term t such that for all closed $r, s \in \Lambda$,

$trs =_\beta T$ if $r =_\beta T$ or $s =_\beta T$

$trs =_\beta F$ if $r =_\beta F$ and $s =_\beta F$

Proof. As for the example in Section 5.3.2 it is possible to give a direct syntactic proof but the semantic one is more satisfactory: suppose there is such a t and let us interpret the equations above in a sensible G-model, like $\mathscr{D}^s_\infty$. Taking for r, s the key terms T, F, Ω, we get by monotonicity $|t|\emptyset\emptyset \subseteq |tTT| = T$ and $|t|\emptyset\emptyset \subseteq |tFF| = F$, hence $|t|\emptyset\emptyset = \emptyset$. Thus $\emptyset = |t|\emptyset\emptyset = \inf(|t|\emptyset|T|, |t|\,|T|\emptyset) = \inf(|t\Omega T|, |tT\Omega|) = \inf(|T|, |T|) = |T|$, which is a contradiction since T is a normal term and hence cannot be equated to a unsolvable term in the model. The second equality is justified by the fact that all binary functions

[48] Because of the definition of a G-web $\{i(\{\alpha\}, \alpha) \mid \alpha \in D\}$ is a coherent set. Hence α and β have to be coherent if $\alpha = i(\{\alpha\}, \alpha)$ and $\beta = i(\{\beta\}, \beta)$.

[49] A *completable partial web* is a triple $(A, \asymp, j)$ such that, for all $a, b \subseteq A$, and $\alpha \in A$ such that (a, α) and (b, α) are in $dom(j)$, we *cannot* have simultaneously $(j(a, \alpha) \asymp_{\text{coh}} j(b, \alpha)$ and $a \asymp b$ and $a \neq b)$.

representable in the models commute with infs and that $(\emptyset,\emptyset) = \inf((\emptyset,|T|),(|T|,\emptyset))$, and the third by $|\Omega| = \emptyset$.

Remark 17. One should not conclude here that the stable semantics is better than the continuous one, even for refutation arguments: subtle compensations occur which allow the continuous semantics to reach refutation arguments that the stable one cannot treat (cf. [24, 11]).

Finally there are situations which cannot be refuted by any of the continuous or the stable semantics but can be refuted via the strongly stable semantics (which, in turn is not a panacea):

Both the continuous and the stable semantics are useless for proving: There is no closed t such that $tFFF =_\beta F$ and for all closed $r,s,u \in \Lambda$, $trsu =_\beta T$ if $r = T$ and $s = F$, or $s = T$ and $u = F$, or $u = T$ and $r = F$.

5.6.4. pcs-models

The class presented below belongs to the continuous semantics. An analogous class could be defined for the stable semantics, based on Winskel [114].[50]

The simplest example of a domain equation for which we need a pcs is

$$M \simeq R(M) \oplus_\perp U \tag{2}$$

This equation is the starting point for modelling Map Theory (with $U \equiv \{T\}$).

Definition 152. The *web of a* (*continuous*) *pcs-model* is a tuple $(D, \leqslant, \nsim, i)$, where $\leqslant$ and $\nsim$ are compatible and $i : P^{<\omega}_{\mathrm{coh}}(D) \times D \to D$ is an injection such that

$$\forall a,b,\alpha,\beta \begin{cases} i(a,\alpha) \leqslant i(b,\beta) \Longrightarrow (\downarrow b \subseteq \downarrow a \text{ and } \alpha \leqslant \beta) \\ i(a,\alpha) \nsim i(b,\beta) \Longleftrightarrow (a,b \text{ coherent} \Longrightarrow \alpha \nsim \beta) \end{cases} (C_\beta)$$

Definition 153. A (*continuous*) *pcs-model* is a model generated by a pcs web, in the following way:

$$\mathscr{M} = (M, A, G) \quad \text{with } M \equiv ((S_{\mathrm{coh}}(D), \subseteq)$$

where A and G are defined exactly as in the case of K-models, except that the metavariables a,b range now over $P^{<\omega}_{\mathrm{coh}}(D)$.

Partial pairs would be, in the most general case, of the form $(A, \prec, r, j)$ with j a partial injection and r any binary relation. Completions and l-completions of partial webs can be defined, which lead to sensible and strongly lazy models (respectively)

[50] In the stable case it is enough to consider *partial orders* $\leqslant$ instead of general preorders, but they are submitted to a further restriction. Also the second line of condition C_β is modified and looks like that of G-models.

when the web is positive. We will only treat two examples and let the reader infer the general processes from them.

Models of (3) can be defined via a process similar to quasi-extensionalization (l-case). With two differences: one cannot separate the construction of D from that of $\simeq$ (we could still separate $\leqslant$ but we have no special reasons to do so). Second we not only have some freedom for the choice of G, but we even have some freedom for the choice of A, since there is more "room" between $R(M)$ and M. Hence we have new completion processes.

Example 154 (*The simplest lazy solution to* $M \simeq R(M) \oplus_{\perp} \{\top\}$). It is the l-completion of the positive pair $(\{f,t\},\emptyset,\emptyset,\emptyset)$, and it is a strongly lazy model.

Explicit construction of the model.

1. $(D, \simeq, \leqslant)$ is defined as the least fixed point of K[51] defined by
   $K(D,\simeq,\leqslant) \equiv (D',\simeq',\leqslant')$ where $D' \equiv (P^{<\omega}_{\mathrm{coh}}(D) \times D) \cup \{f,t\}$ and
   $(a,\alpha) \simeq' (b,\beta)$ iff $(a \simeq b \Rightarrow \alpha \simeq \beta)$.
   $t \simeq' \alpha$ iff $\alpha = t$, and
   $f \simeq' \alpha$ iff $\alpha \neq t$.
   $(a,\alpha) \leqslant' (b,\beta)$ iff $(\downarrow b \subseteq \downarrow a$ and $\alpha \leqslant \beta)$ (the $\downarrow$ is relative to $\leqslant$)
   $t \leqslant' \alpha$ iff $\alpha = t$
   $f \leqslant' \alpha$ iff $\alpha \neq t$.
2. i is the inclusion map, A is defined as usual and continuous functions are encoded via

$$G'(h) \equiv \{(a,\alpha) \mid \alpha \in h(\downarrow a)\} \cup \{f\}.$$

Example 155 (*A variant useful for Map Theory*). For Map Theory we need a lazy model and furthermore we need that for $T \equiv \{t\}$ we have $Td = T$ for all d. The preceding model does not work since $Td = \emptyset$, but it is indeed enough to change the definition of application on T by taking $A(T) \equiv x \mapsto T$.

5.6.5. κ-pcs

As already mentioned the continuous semantics is not well suited for dealing with some *non-deterministic* extensions of λ-calculus as well as for *foundational purposes*, and one is led to work with the weaker following notion of κ-continuity, for κ a regular cardinal.[52]

Definition 156. A function is κ-*continuous* if it is monotone and commutes with all sups of κ-directed sets, where B is κ-*directed* if any subset of B of cardinality less than κ is bounded in B.

[51] One can consider that K acts on a set (a cpo) and not a class. It is indeed enough to assume that we are working within the set of all hereditarily finite sets built on the atoms (or Ur elements) f,t.

[52] Recall that κ *regular* means that κ is infinite and cannot be the sup of fewer than κ cardinals smaller than κ.

The κ-continuous semantics is a straightforward generalization of Scott's one, which corresponds to the case $\kappa \equiv \omega$. Working with the κ-semantics amounts essentially to replace every occurrence of the word "finite" by "of cardinality less than κ".

Within the κ-semantics the reward of working with webbed models when solving recursive domain equations is doubled since the inverse limit construction, which never leads to easy manipulations, would here still be complicated by the fact that there are limit ordinals below κ, which introduces further theoretical difficulties (cf. [40]).

In particular, κ-versions of the solutions of (3) given in Examples 154 and 155 can easily be written down (cf. [19, Section 8] for the details of the construction, if needed). We then get:

Example 157 (*A basis for a model of Map Theory*). We replace in Example 155 the completion process of Example 154 by its κ-version. The only difference is that $(D, \Rightarrow, \leqslant)$ is now built as the increasing union of the ordinal sequence $((D_\alpha, \Rightarrow_\alpha, \leqslant_\alpha))_{\alpha<\kappa}$ with $D_\alpha \equiv \cup\{D_\beta \mid \beta<\alpha\}$ if α is a limit ordinal. We thus get a lazy model of λ-calculus, which can be enriched (for big enough κ) into a model of Map Theory. This last point will be sketched in Section 5.7.

*5.6.6. *H-models*

We turn now to the *strongly stable semantics* of Bucciarelli and Ehrhard. This semantics gives rise to a class of webbed models of untyped λ-calculus, which are built along similar lines as for G-models (cf. [11] or [14]) and that we call *H-models* here.

The webs of the domains are here hypercoherences. These were introduced by Ehrhard [42].

Definition 158. A *hypercoherence* is a pair (D, Γ), where $\Gamma \subseteq P^{<\omega}(D)$ and Γ contains all singletons. We then define $P_{h\mathrm{coh}}(D)$ as the p.o. set of all *hereditarily coherent* subsets d of D, i.e. those d such that $P^{<\omega}(d) \subseteq \Gamma$. The domain generated by the web is here taken as $(P_{h\mathrm{coh}}(D), \subseteq)$.

We will not give here the definition of strongly stable functions, which is not immediate. It is indeed enough to know that strongly stable functions are stable and are encoded via a function Tr_{ss} which is the same as Tr_s (defined in section *G*-models) except that the metavariable a ranges over $P^{<\omega}_{h\mathrm{coh}}(D)$.

Definition 159. A *H-web* is a triple (D, Γ, i) where $i : P^{<\omega}_{h\mathrm{coh}}(D) \times D \to D$ satisfies the same compatibility conditions than in the stable case.

Definition 160. A *H-model* is a model of the form

$$(M, A, G) \quad \text{where } M \equiv (P_{h\mathrm{coh}}(D), \subseteq)$$

where A and G are defined as in the *G*-case.

All the methods mentioned so far for building or studying webbed models generalize to this setting, but are more delicate to express and prove here. The point is that, because of the definition of strongly stable functions (that we did not give), we are not working here with 2-level domains or models (the webs and the domains or models they generate), but with 3-level objects, since an intermediate level has to be considered for defining strongly stable functions.

All we need to know for Section 6 below is that there are strongly stable analogues $\mathscr{D}_\infty^{ss}$ and $\mathscr{P}_\infty^{ss}$ of $\mathscr{D}_\infty$ and $\mathscr{P}_\infty$ [11, 13].

*5.6.7. *Prime algebraic domains vs. algebraic domains*

We present now the material needed to understand the jump between the classes presented so far and the two below. We also explain why it is preferable to stick to the former presentation whenever possible.

Definition 161. A *ccpo* is any cpo $\mathscr{D}$ such that every bounded subset A of $\mathscr{D}$ also has a *sup*. These domains are very close to complete lattices.[53]

Definition 162. An element $h \in \mathscr{D}$ is *compact* or "*finite*" (resp. *prime*) if for all directed $A \subseteq \mathscr{D}$ (resp. all A), $h \leqslant \sup(A)$ implies $\exists d \in A\ h \leqslant d$. Such an element is *prime* if the same is true for all $A \neq \emptyset$. We let $\downarrow_c d$ (resp. $\downarrow_p d$) denote the set of compact elements (resp. of prime elements) below d.

Definition 163. A ccpo $\mathscr{D}$ is *algebraic* if for all $d \in \mathscr{D}$ the set $\downarrow_c d$ is directed and $d = \sup(\downarrow_c d)$; it is *prime algebraic* if $d = \sup(\downarrow_p d)$ for all d. A *Scott domain* is an algebraic ccpo.

For example, the prime elements of $(P(D), \subseteq)$ are the singletons and the empty set, and the compact elements are the finite subsets. More generally, the prime elements of $(S(D, \leqslant, \asymp), \subseteq)$ are the initial segments generated by one or zero element of D, and its compact elements are those generated by a finite coherent subset of D.

The domains underlying all the webbed models we have considered so far are *prime-algebraic Scott domains*, and so are all historical or classical models of λ-calculus. Except for the H-case they are even *binary*, in the sense that a set of prime elements is upper bounded iff its elements are pairwise upperbounded. By contrast the classes of models which are defined in the next two subsections concern *algebraic* Scott domains.

In our webbed description of a prime algebraic domain $\mathscr{D} \equiv S(D, \leqslant, \asymp)$, the web of a domain was essentially the set $\mathscr{D}_p$ of its prime elements and each $d \in \mathscr{D}$ is viewed as the coherent initial segment $\downarrow_p d$ of $\mathscr{D}_p$. In the general algebraic setting the web should

[53] (1) Complete lattices are ccpo's. (2) Adding a top element to a ccpo gives a complete lattice. (3) Removing the top element of a complete lattice gives a ccpo.

be chosen as the set $\mathscr{D}_c$ of compact elements and d should be viewed as the directed initial segment $\downarrow_c d$ of $\mathscr{D}_c$ (which is, equivalently, a *filter* for the reverse order[54]).

Also, when working with a prime algebraic domain $\mathscr{D}$ we can choose to encode continuous functions as subsets of some $K \times D$, where $K \subseteq P^{<\omega}(D)$ generates the compact elements of $\mathscr{D}$, while D generates the prime elements. Indeed f is determined by the set of pairs (h, p), h compact, p prime, such that $p \leqslant f(h)$. If one only knows, or uses, that $\mathscr{D}$ is algebraic, then one has to use traces which are subsets of $K \times K$, since in this case one can only determine f by the pairs (h,k) of compact elements such that $k \leqslant f(h)$.

The conclusion is that when one works within one of the two frames below, one treats more general models but automatically uses a redundant presentation of all the prime-algebraic models in it.

*5.6.8. *Scott's information systems (SIS)*

Scott's information systems [109], in the presentation of Larsen and Winskel [84], are indeed a webbed presentation of all Scott domains, and the models of λ-calculus arising from information systems are exactly the reflexive Scott's domains (within the continuous semantics).

It is easy to see that pcs correspond exactly to binary prime algebraic SIS.

Also, coherent spaces, and even hypercoherences can be viewed as particular cases of information systems (this remark deals only with webs or domains, not with functions, traces, or models).

*5.6.9. *Filter models*

The class of *filter models* was defined by Coppo et al. [32], but the first examples were given in [31, 10]. This class is another frame where solving recursive equations on domains is replaced by solving more simple set theoretical equations on webs (for such a definition of $\mathscr{D}_\infty$ see [32]). We are here in the general algebraic view and each point d of the domain is viewed as the filter $\downarrow_c d$ (cf. Section 5.6.7).

Not all filter models can be viewed as pcs-models (since there are filter models which are not prime algebraic), and conversely,

Not all pcs-models can be viewed as filter models, since all filter models of [32] are *semi-extensional*, in the sense that they are forced to satisfy the three following equivalent properties: $|\lambda x.\lambda y.xy| \subseteq |\lambda x.x|$ or $G \circ A \leqslant id$ or $|\lambda x.tx|_\rho \subseteq |t|_\rho$ for all t, ρ, and x not free in t. Semi-extensionality makes easier the proof theoretic study of the models when viewed as "intersection type assignment systems" [38], but excludes a lot of pcs-models, to begin with all graph models. Note however that $\mathscr{E}$ and $\mathscr{P}_\omega$ are considered as (generalized) filter models in [99].

Finally, all filter models can be viewed as particular case of *SIS*-models [30]. The converse is false for the classical definition of filter models; but is true for the extended definition of filter models given by Alessi in [3].

[54] Which is the natural order in the logical viewpoint.

*5.6.10. *Related work*

It is worth mentioning that interesting generalizations of the graph models construction have been considered by Di Gianantonio and Honsell [39] and by Plotkin [98].

The redundancy phenomena of the last two classes in the prime algebraic case was also explicitly observed by Alessi [4], where it is solved via the class of "Type preorders". Type preorders are equivalent to the K-webs here, and the precise topological and categorical relations between Type preorders and Intersection type assignment systems is completely worked out there. That continuous models built as inverse limits could be viewed as filter models has been worked out also by Alessi in his thesis [3].

Finally, pcs-webs (of domains) are similar to "*event structures*", at least as defined in [87].[55] Event structures are issued from the work of G. Kahn and G. Plotkin on the sequentiality of λ-calculus, and are used for modelling processes.

5.7. Webbed models for foundations

We sketch here the proof of the consistency of Map Theory relative to that of $ZFC + SI$. Here, as in [82], ZFC includes the Foundation Axiom. Axiom SI asserts the existence of a "strongly inaccessible" cardinal σ.

Definition 164. A cardinal σ is *strongly inaccessible* if σ *regular* and if furthermore $\gamma < \sigma$ implies $2^{\gamma} < \sigma$ for any cardinal γ.

So, the aim here is to build a model $\mathscr{M}$ for Map Theory, within any given model $\mathscr{U}$ of $ZFC + SI$.

Remark 18 (*About the hypothesis SI*). To be a regular cardinal is not difficult: any infinite successor cardinal is regular, and this is the case in particular of all the successive successors ω_n of ω (but ω_ω is not regular). By contrast, to be inaccessible is drastic and SI definitely strengthens ZFC since $ZFC + SI$ implies $ZFC + cons(ZFC)$. The relative consistency[56] of $ZFC + SI$ is open, however SI is generally considered as a reasonable axiom. It is finally *open*, but not unlikely whether the consistency of Map Theory could be proved relatively to $ZFC + cons(ZFC)$.

To give a flavor of the construction of the model we have to go a little more deeply in the presentation of MT than we did in Section 2.6 and we give below a global and sketchy presentation of its axioms and rules. Their exact formulation is of no interest here.[57] Firstly because we will not enter into details, and secondly because the relevant

[55] *Thanks are due to G. Winskel, for pointing this out.*

[56] A set of axioms is *relatively consistent* if its consistency can be proved from that of *ZFC*.

[57] They can be found in [54] or [19, Appendix C].

axioms should be understood as equational instances of two semantic properties of Φ (the *SIP* and the *GCP* below).

The set of axioms and inference rules of *MT* is naturally divided into 4 groups:

(I) The "*λ-calculus axioms*" are the usual axioms and rules of λ-calculus (except that the metavariables $t, u \ldots$ range now over *MT*-terms) together with 5 very simple axioms which fix the applicative behavior of $\top, \bot$ and if. This group hence appears as an extension of the axiomatization of β-equivalence as given Section 3.7).

(II) The second group consists of one single inference rule, the "*Quartum Non Datur*", abbreviated by *QND*.

Before explaining the *QND* we notice that a simple consequence of the λ-calculus axioms [58] is that for a term t it is equivalent to be (provably equal to) an abstraction, or to satisfy the equation $x = \lambda y.xy$, or to satisfy $x = F'x$, where F' is a local denotation for $\lambda x \lambda y.xy$ (since ε has another meaning in *MT*). In *MT* all abstractions are given the truth value "false". More generally in *MT* the terms which are provably equal to $\top, \bot$, or abstractions are respectively given the truth value $\top$ (*true*), $F \equiv \lambda x.\top$ (*false*), and $\bot$ (*undefined*), and no other term has a truth value.

Now, the *QND* rule expresses that only matter (w.r.t. equations) those terms which have truth values. Indeed, *QND* states that we are able to derive an equation $t = t'$ as soon as we can derive the three following instances: $t[x := \top] = t'[x := \top]$, $t[x := \bot] = t'[x := \bot]$, and $t[x := F'x] = t'[x := F'x]$.

(III) Four short "*quantifier axioms*" axiomatize the behavior of the ε operator. They approximate the intuition that ε always chooses elements in Φ, that the quantifiers defined from it range over Φ, and that they have the intended behavior.

(IV) The "*set-theoretic axioms*" axiomatize φ and give its full power to *MT*. They consist in several axioms plus one inference rule called *IND*.

IND is an induction principle which expresses *intuitively* that Φ is *well-founded* w.r.t. the binary relation: $x <_\Phi y$ iff $y \neq T$ and $\exists z \in \Phi x = yz$. All the other axioms express closure properties of Φ, most of them are simple and natural, and all are particular instances of a single property of Φ (the *GCP* below).

It is easy to check that the satisfaction of (I) and (II) is ensured as soon as we have a model of λ-calculus which satisfies the domain equation

$$\mathscr{M} = R(\mathscr{M}) \oplus_\bot \{T\} \tag{3}$$

provided we interpret $\top$ by T and $\bot$ by $\bot_\mathscr{M}$, and we force $A(T) \equiv x \mapsto T$ and $A(\bot_\mathscr{M}) \equiv x \mapsto \bot_\mathscr{M}$. Such a model was built in Section 5.6.5.

From now on we make no typographical distinction between the constants $\top$ and $\bot$ and their interpretations in $\mathscr{M}$.

Eq. (3) is nothing else than (2) except that we write $=$ instead of $\simeq$ for simplifying the exposition below).

[58] Even of usual λ-calculus.

Note that $\mathscr{M}$ satisfies here a strong form of the *QND*, namely the first order axiom which expresses that all elements are equal to $\top, \bot$ or are solutions of $F'x = x$; this property is called the *Strong Non Datur* (*SQND*) in [19].

The interpretation of if is the function defined by

$$if(x,y,z) = \begin{cases} y & \text{if } x = T, \\ z & \text{if } x = F'x, \\ \bot & \text{if } x = \bot. \end{cases}$$

As a matter of fact in any solution of (3) the elements of $R(\mathscr{M})$ coincide with abstractions (with parameters in M), and with the solutions of $F'x = x$. Hence if is totally defined. If furthermore $\mathscr{M}$ is a ccpo, then $R(\mathscr{M})$ has $\lambda x.\bot$ as least element and if is continuous (and stable and strongly stable if $\mathscr{M}$ was built within these semantics).

For satisfying the quantifier axioms (III) we assume now that we are working in the continuous semantics or more generally in the κ-continuous semantics, where κ is any regular cardinal (cf. Section 4.5). So we have available Scott's topology or its κ-analogues. The intention is now to interpret φ by the characteristic function of some open subset Φ of $\mathscr{M}$ ($\varphi x = T$ if $x \in \Phi$ and $\bot$ otherwise). We will not say much about the interpretation of ε here, except that it is rather straightforward to define it from Φ in such a way that (III) is satisfied. For this it is enough that $T \in \Phi$, $\bot \notin \Phi$ and $\Phi = \uparrow \Psi$ for some Ψ such that $card(\Psi) < \kappa$, where:

- $\uparrow H \equiv \{x \in M \mid \exists y \in H \;\; y \sqsubseteq x\}$, for any $H \subseteq M$, where $\sqsubseteq$ is the partial order on M.

For finding κ-continuous (φ, ε) satisfying group (IV) we have to be more drastic and assume furthermore the existence of an inaccessible cardinal σ below κ (then one can as well take $\kappa = \sigma^+$, since the successor of any cardinal is regular). This excludes of course $\kappa \equiv \omega$, and many other regular cardinals.

Group IV of axioms and rules can be seen as an *equational approximation* of the following properties of Φ:

- $T \in \Phi$,
- $\Phi \subseteq \Phi^0$ (Strong Induction Principle *SIP*),
- $\Phi = \bigcup \{G^0 \to \Phi \mid G \subseteq_{o,\sigma} \Phi\}$ (Generic Closure Property *GCP*),

where

– for G κ-open, $G^0 \equiv \{d \in M \mid \forall \bar{x} \in G^\omega \;\; \exists n \geqslant 0 \;\; dx_1 \ldots x_n = T\}$,
– for $X, Y \subseteq M$, $X \to Y \equiv \{d \in M \mid \forall x \in X \;\; dx \in Y\}$,
– $G \subseteq_{o,\sigma} \Phi$ means that G is an *essentially* σ*-small* open subset of Φ, which means: G is open and $\exists H \subseteq G$ $card(H) < \sigma$ and $G = \uparrow H$.

In particular, *SIP* expresses exactly that Φ is well-founded for $<_\Phi$.

SQND, *SIP* and an intuitive form of the *GCP* were the intuitions behind the axiomatization of *MT* (cf. the introductory Part I of [54]). The core of the consistency proof in [19] is to prove that for any solution of (2) in the κ-semantics one can find a Φ satisfying the above conditions. The models built in [19] all satisfy *SQND*, *SIP*, and *GCP*, and hence model a strong version of *MT*. Of course these three properties cannot directly be taken as axioms, since they are non-equational. The key point is

that weaker equational instances are indeed sufficient for ensuring that MT is at least as strong as ZFC.

In the models of MT which satisfy $SQND$, a suitable quotient of Φ is a model of ZFC (equality and membership in it being essentially terms, defined via Curry's fixed point operator). If SIP is satisfied, then the models are ω-models, in the sense that their ω is isomorphic to the one of the big universe $\mathscr{U}$. Finally, we explicitly build "standard models" $\mathscr{M}_{\kappa,\sigma}$ for MT, which are *webbed* models, and in which (the suitable quotient of) Φ is isomorphic to the set V_σ of all well-founded sets of $\mathscr{U}$ which have rank $<\sigma$.

Besides giving a (reasonable) proof of the consistency of MT, these models are intended to support and/or to suggest improved axiomatizations of MT.

6. Recent results and open questions

We are interested in *consistency questions*.[59] More generally, we are concerned with the *representativeness of the classes of models* (and of the semantics) introduced so far with respect to theories. For λ-calculus this is a recent problematic initiated only in [59].

Let us say that a class C of models *represents a theory* T if there is a model in C whose theory is exactly T. It is more than plausible that any conceptual class C of models is *incomplete*, in the sense that it does not represent all theories, this is confirmed below for all the classes of models we have mentioned so far. However we will see that these classes are able to represent 2^ω distinct theories and that there is a theory which is represented everywhere. So the question is: which kind of theories are these classes able (or unable) to represent?

Instances of this problem are:

Given T, C, does C represents T? (ex. $T \equiv \lambda_\beta$?). Given C, C', do C, C' represent the same theories? Given C, is there a "small" $C' \subseteq C$ which represents the same theories? Given a strategy S, is C able to represent $T^{cxt}(S)$? and so on.

In the following, "the class of continuous models" (resp. stable or strongly stable) without any more precision, refers to "the largest class of such models the reader has in mind".

6.1. Incompleteness of the usual semantics

We will see, soon, that the classes of continuous, stable and strongly stable models are incomplete, and that moreover one can find in each of them, a model $\mathscr{M}$ such that $Th(\mathscr{M})$ is not represented in the other two.

On the other hand, there is a (sensible) theory T, namely $Th^{ctx}(S_h)$, which is represented within all three semantics, and in fact within all reasonable functional semantics,

[59] But will rather concentrate on the most recent results. The interested reader is advised to complete his reading with Barendregt's book [8, Chapters 4, 16 and 17], including exercises.

since X. Gouy proved, rather recently, that all possible analogues of Scott's $\mathscr{D}_\infty$ had the same equational theory. T is hence the theory of a K- a G- and a H-model [51, 52]. For the stable case this was conjectured by Honsell and Ronchi [58].

The first incompleteness result was given by Honsell and Ronchi Della Rocca [59]. They proved, via an involved syntactic proof, that $T^{ctx}(S^0)$ (defined in Section 3.12), could not be represented in the continuous semantics. Following a similar method Gouy [51] proved that $T^{ctx}(S_I^0)$ could not be the theory of a stable model; in the stable case the syntactical proof is still much harder, and it was clear that the method would not be feasible at all in the strongly stable case.

Semantic and more conceptual proofs of the incompleteness of the three semantics were given afterwards by Bastonero and Gouy, who proved that $Th(\mathscr{P}_\infty^{\mathrm{s}})$ and $Th(\mathscr{P}_\infty^{\mathrm{ss}})$ could not be the theory of a continuous model [13, 14] and that $Th(\mathscr{P}_\infty^{\mathrm{ss}})$ could not be that of a stable one. Furthermore, the contradictions were coming from simple sets of equations and inequations which were already "in" $T^{ctx}(S^0)$ (resp. $T^{ctx}(S_I^0)$), which reproved that these theories could not be represented within the continuous (resp. stable) class.

Another proof due to Bastonero [11], builds, using forcing, a continuous model $\mathscr{M}$ such that $Th(\mathscr{M})$ cannot be represented in the stable semantics, nor in the strongly stable semantics (more accurately: nor in the class of H-models).

Remark 19. *It is worth comparing the results above with the situation which occurs for Plotkin's *PCF* [95], a typed extension of λ-calculus (with constants and rules added) which has been intensively studied. Each semantics gives rise to a "*standard model*" of *PCF* (a phenomenon with no analogue in the untyped case). It is rather easy to see that the three standard models have different theories (see [24]), using that each of the semantics eliminates functions or functionals which exist in others (like the *parallel or*, or *Berry*'s *function* which is strongly connected to the counter-example given at the end of Section 5.6.3). Though the basic idea for separating the three $\mathscr{P}_\infty$'s comes from the typed case, things are much more subtle to work out in untyped λ-calculus (as already shows Gouy's result concerning $\mathscr{D}_\infty$). Furthermore, the proofs which are given in the untyped case are general enough to cover classes of models and not only isolated models.

The questions which are left open here are:

Question 1. Could there be a categorical characterization of those ccc's which represent $T \equiv T^{cxt}(S_h)$?

Comments. Gouy's result concerns all *regular* c.c.c.'s (as defined in [52]), which cover all reasonable c.c.c.'s whose objects underlie domains and whose morphisms are at least continuous. On the one hand, $T^{cxt}(S_h)$ is closely linked to the very computational nature of λ-calculus and one could expect that it would be represented in all reasonable c.c.c.'s. An ad-hoc counter-example is the full sub-c.c.c. (of, say, the c.c.c.

of complete lattices and continuous functions) generated by any extensional model $\mathscr{M}$ such that $\mathscr{M} \simeq \mathscr{M} \times \mathscr{M}$ and $Th(\mathscr{M}) \neq T$. Indeed in such a c.c.c. all objects, and in particular all reflexive objects, are isomorphic to $\mathscr{M}$ and hence are equationally equivalent, so this c.c.c. cannot represent T ($\mathscr{M} \equiv \mathscr{P}_\infty$ works, provided we start from an infinite set of atoms).

The proof that all regular c.c.c.'s represent T is based on the fact that their $\mathscr{D}_\infty$ satisfy Hyland and Wadsworth's Approximation Theorem, a property which does not even make sense for non orderable models. Unorderable models are however hard to find (see Section 6.6).

This question should also be related to the last question we raise in Section 6.7.

***Question 2.** Is the full class of strongly stable models incomplete?

This is most likely but the techniques developed in [11] only apply to the hypercoherence setting.

***Question 3.** It is *nearly* proved in [11, 13, 14] that the three variants of $\mathscr{P}_\infty$ have *incomparable* theories: one inclusion is left open, and conjectured to be false, namely $Th(\mathscr{P}_\infty^{\mathrm{s}}) \subseteq Th(\mathscr{P}_\infty^{\mathrm{ss}})$.

6.2. Do there exist semantic models of λ_β, $\lambda_{\beta\eta}$, $\mathscr{H}$, $\lambda_{\beta\infty}$?

The following are long-standing questions (as long as the existence of the diverse semantics!) even though they were only first discussed in print in Honsell–Ronchi [59][60] (for the continuous semantics).

Question 1. Is there a continuous (stable, strongly stable) model whose theory is exactly λ_β or $\lambda_{\beta\eta}$? a webbed one? a graph model (for λ_β only)?

Question 1 can be weakened in two ways:

Question 2. Is λ_β the intersection of all theories of continuous models? and similarly for $\lambda_{\beta\eta}$ and extensional models; and similarly for other semantics.

Question 3. Given a (reasonably closed) class of models in a given semantics, is there a minimal theory represented in it?

Known results. A recent paper of Di Gianantonio et al. [40] proves that there is a model whose theory is exactly $\lambda_{\beta\eta}$ in the ω_1-semantics (thus all questions collapse

[60] The referee mentioned that, besides the authors of [59], Plotkin also studied the problem and Dezani was aware of it.

and have positive answer in this case). But Questions 1 and 2 remain open for Scott's continuous semantics, and also for λ_β. The second result of [40] is that Question 3 admits a positive answer for Scott's continuous semantics,[61] at least if we restrict to extensional models. The proof of this result uses essentially that the class of relevant continuous models is closed under a process which builds inverse limits over an infinite product of models.

However, the proofs in [40] use logical relations, and since logical relations do not allow to distinguish terms with the same applicative behavior, the proofs do not carry out to λ_β.

The reader interested in solving this problem should also have a look at Selinger's result presented in Section 6.6.

Question 4. Is there a non-syntactical model of the λ-theory $\mathscr{H}$ generated by $E = \{t = u \mid t, u \text{ unsolvable}\}$?

We end with the following question, in the same spirit than the first one:

Question 5 (*Dezani–Ciancaglini*). Is there a non-syntactical model of the theory $\lambda_{\beta\infty}$? where $\lambda_{\beta\infty}$ is the theory of the "infinite λ-calculus" introduced recently by Berarducci for studying mute terms (cf. Section 6.8).

6.3. Global representativeness of the classes of models w.r.t. λ-theories

Since there are countably many possible equations in λ-calculus, there are at most 2^ω λ-theories. This maximum number is reached, since it is shown in [8, Chapter 16.3] that there are 2^ω (consistent) sensible and extensional λ-theories.

Are the various semantics rich enough to represent the maximum number of consistent (*resp. and sensible*) *theories*? This is a very natural question. We conjectured that the answer was positive. A negative answer would however have shown a very surprising light on continuity and/or its variations. The non-necessarily sensible case is not hard:

Proposition 165. *There exist 2^ω graph models with different equational theories, and similarly with G-models and H-models.*

Comments about the proofs. For graph and G-models this was proved by Kerth in [60, 68, 71, 72]. The proof he gave for the stable setting can be adapted in a straightforward way to the strongly stable setting.[62] The models are however not sensible.

[61] Strictly speaking the proof of [40] is stated and written for a variant of the continuous semantics. But the same proof works for the ω-continuous case.

[62] O. Bastonero, private communication.

The analogous question but restricted to sensible models happened to be much more difficult, and a positive answer has been brought only as I was revising this paper. Thus, now we have:

Proposition 166. *There exist 2^ω sensible graph models with different equational theories.*

Comments about the proof. Kerth proved in his thesis [68] that there existed 2^ω non-equationally equivalent graph models, which he proved to be sensible ... up to a syntactical conjecture on the head-reduction of unsolvable terms that he stated and for which he gave much evidence [70]. This is the conjecture that David has recently proved [35].

Solving the conjecture was needed since Kerth's models were generated by non-positive partial pairs (cf. Remark 104), so that the reducibility technique could not apply.

A natural approach taking the problem at the other end would be to use Remark 112 in Section 5.5 to conceive 2^ω partial injections j which give rise to non-isomorphic sensible graph models. The difficult point is then to ensure that these models could furthermore be forced to be non-equationally equivalent. So we are left with:

Question 1. Are there 2^ω sensible graph models which are non-equationally equivalent and are generated by positive partial pairs?

We end with the following:

Question 2. Are there non-syntactic sensible models whose theory is strictly included in BT? 2^ω?

Comments. Kerth's graph models above would be good candidates, because their sensibility follows from a proof on the syntax of λ-calculus and not from an approximation theorem. By the way, it is clear that the strong form of the approximation theorems does not apply to Kerth's models. Indeed, they all distinguish Y from another classical fixed point operator, Θ, and hence distinguish two terms which have the same Böhm tree.

Can the size of the web affect the equational theory of a model? In other words: given a reasonable class C of webbed models (even in permissive sense) which, say, contains enough small Scott domains:

Question 3. Does C_ω represents the same theories than C? where C_ω the class of models in C which have a countable web.

Comments. This is likely and would be a kind of Löwenheim Skolem Theorem. Note that for any model $\mathscr{M} \in C$ there is a countable $\mathscr{M}'$ inside $\mathscr{M}$ which satisfies the

same equations (by the ordinary Löwenheim–Skolem Theorem applied to $\mathscr{M}$ viewed as a model of CL), but there is no reason why $R(\mathscr{M}')$ should be exactly the set of the continuous (resp. stable) functions on $\mathscr{M}'$.

How does the nature of the web or the domain affect the theory of a continuous model? We are only concerned here with the continuous semantics. It is clear that K-models represent more theories than graph models, because there exist extensional K-models and no extensional graph models, but:

Question 4. Do K-models, pcs-models, and SIS-models represent the same theories?

This question was centered on the webs, and concerned only prime-algebraic domains. A similar question relative to domains is:

Question 5. Do binary prime algebraic, prime algebraic, algebraic or general continuous models represent the same theories?

6.4. Can the shape of the model be influential?

Question. How does the fact that a model $\mathscr{M}$ (of λ-calculus) satisfies a domain equation of the shape $\mathscr{M} \simeq \mathscr{F}(\mathscr{M})$, where $\mathscr{F}$ is a suitable functor, affect the equational theory of $\mathscr{M}$? (in the pure language).

Comments. The only examples I know are "pieces of extensionality".

Extensionality itself ($\varepsilon = I$) is indeed equivalent to $\mathscr{M} \simeq R(\mathscr{M})$, where $R(\mathscr{M})$ is the set of morphisms on $\mathscr{M}$.

Quasi-extensionality: Equations like $x(\varepsilon x) = xx$, $x(uxu_1 \dots u_n) = x(u(\varepsilon x)u_1 \dots u_n)$, a.s.o., are consequences of $\mathscr{M} \simeq R(\mathscr{M})_\perp$, since any such model satisfies $\forall x\ (x = \varepsilon x \vee x = \perp)$ and $\forall y\ (\perp y = \perp)$.

On the other hand, these equations are also satisfied in the model of Example 154, just because it satisfies $\forall x\ (x = \varepsilon x \vee x = \perp \vee x = T)$ and $\forall y\ (\perp y = \perp \wedge Ty = T)$, but they are not consequence of $\mathscr{M} = R(\mathscr{M}) \oplus_\perp \{T\}$, since their satisfaction uses the precise applicative behavior of T (for which there is freedom, while for $\perp$ there is none).

Are there other natural (or less natural) equations *or inequations* which are true independently of the pair of inverse isomorphisms which realize the domain equation, and of the retraction pair (A, G) which are derived from it and realize the fact that $\mathscr{M}$ is a model of λ-calculus? Recall that there can be unicity of (A, G), for example in the extensional case, but that it is not the general case.

6.5. Towards a general approximation theorem for comparing theories of models?

Obviously, basic uniform tools are still lacking in connection with the problem of the comparison of the theories of different models.

The first thing to do would be to develop the approximation method along the lines of this paper, and to widen its scope as much as possible. This would help us to compare directly, the theories of models which have comparable webs. By this we mean a pair of webbed models such that the web of the first is a quotient or a substructure of the second, or which are generated by the same webs but in different semantics.

When looking at this question, we got partial answers, that we hope to complete. The difficulties are of two kinds.

First the proofs of the approximation theorem, when done via the computability method, use the semi-extensionality hypothesis $|\lambda x.tx|_\rho \subseteq |t|_\rho$. This is obviously unsatisfactory since proof theoretic arguments show that the approximation theorem is true for the non-semi-extensional model P_ω [63] and for $\mathscr{E}$, while semi-extensionality is true for no graph model. We hope that a technical trick will be sufficient to eliminate the hypothesis.

A second difficulty is that "computability" is "much too stratified". By this we mean that one works with predicates $Comp(\alpha, \rho)$ which are parametrized not only with $\alpha \in D$, but also with environments ρ. Environments can be eliminated when proving normalization theorems (this gives the reducibility method in Krivine's style), however we failed to eliminate it for proving approximation theorems. This has two drawbacks. The first is that notations are much more heavy in the stratified setting, and the second is that fixed point theorems should be technically more difficult to reach, even if we restrict to positive partial webs. Thus we raise:

Question 1. Is the (strong version of) the approximation theorem true for all webbed models with positive webs? (let us begin with graph or K-models).

Question 2. Can such a theorem be proved by a reducibility-like method.

A uniform approximation theorem for, say, K-models to begin with, would allow us to prove that the interpretation of any closed term t in an extensional K-model $\mathscr{K}$, is the initial segment generated by the interpretation of t in its underlying graph model $\mathscr{G}$, [64] essentially because it is easy to prove it for normal terms and more generally for terms having a finite Böhm tree.

Intended applications (*some concrete examples*). Compute uniformly the theory of the extensional completion of a graph model from the theory of this graph model, at least when it is generated by a positive pair.

Give a general reason why $Th(\mathscr{E}) = Th(P_\omega)$.

Prove that Ω is $\beta\eta$-easy (cf. Section 6.8) directly from the fact that it is easy (there one would have to deal with non-positive pairs), while at present we have to redo the proof in the extensional setting (cf. Section 6.8).

[63] S. Ronchi, private communication.

[64] The usefulness of the approximation theorem for solving this question was pointed out to me by S. van Bakel.

Extend Gouy's results about $\mathscr{D}_\infty$ to as many models as possible:

Question 3. When do models generated by the same partial webs, but in different semantics, have the same theory? Is it true for all the models generated by the same positive partial web?

Comments. In view of Gouy's results about $\mathscr{D}_\infty$ and $\mathscr{P}_\infty$ above, it is reasonable to restrict oneself to positive partial pairs. Also, the analogues of Engeler's model (in the stable or strongly stable semantics) have the same theory.[65]

6.6. Orderability and orders

As already mentioned there are *computational motivations* for taking ordered structures as models of λ-calculus. *However not every model of lambda-calculus is orderable* (in such a way that application is monotone).

This is not trivial, in fact the mere question about the orderability of the term model $\Lambda/=_\beta$ was solved only very recently, and *negatively*, by Selinger [110]. It can be noticed however that the term model admits 2^ω non-trivial preorders since there are 2^ω ordered models with different equational theories [66]. Selinger's result can be reformulated like this: in any partially ordered applicative structure $\mathscr{M}$ which models λ-calculus, if $Th(\mathscr{M}) = \lambda_\beta$ (*or* $\lambda_{\beta\eta}$) then the order is trivial on the interpretations of closed terms.

Just before, an unorderable model had been exhibited by Plotkin [100]. Answering a question of Friedman, which arose with foundational motivations (see [45]), Plotkin showed that there existed ω-separable models of λ-calculus. Here $\mathscr{M}$ *ω-separable* means that any partial function on $\mathscr{M}$ with finite domain extends to a total function which is representable in $\mathscr{M}$. Any partial order on such a model is trivial since, if $d < d'$, a function exchanging d, d' cannot be monotone, and hence cannot be representable.

For interesting related questions, which are however outside the scope of the present paper we refer to [110].

Question. For the continuous, the stable, and the strongly stable semantics it can be shown that the partial order of a reflexive model is first-order definable from application only, but the formulas which have been proposed by Plotkin [98], Kerth [68, 67], and Bastonero [11], depend on the semantics. Is there a general reason? a uniform formula?

6.7. Correctness and completeness w.r.t. strategies

The following general question has been generating a lot of work for the last 20 years, at least in typed λ-calculus (*PCF* and variants). The necessary background on strategies was given in Section 3.12.

[65] O. Bastonero, private communication, who notes furthermore that this is no longer true if one considers the inequational theories of the models ($t \sqsubseteq t'$ instead of $t = t'$).

Question. Given a strategy S (such that $T^{ctx}(S)$ is a λ-theory), does there exists a model whose theory is induced by S? can be approximated using S?

Dual point of view: given a model M what is its computational meaning (if any), in other words: what kind of information about β-reduction is it able to bring? Are there strategies for which it is correct? fully abstract? (Definitions in Section 4.3).

We rephrase within this terminology some results that we have already met, and go a little deeper.

Example 167. The three $\mathscr{D}_\infty$ models are fully abstract w.r.t. S^h. The model Norm is fully abstract w.r.t. S^n, but only adequate for S^h.

Example 168. By contrast the operational meanings of the (non-sensible) different Park's theories (if any!) is only approximated: we know that $\mathscr{P}^c_\infty$ is adequate for S^0 [59], and $\mathscr{P}^s_\infty$ and $\mathscr{P}^{ss}_\infty$ are adequate for S^0_l [13] or [14], but they are not fully abstract (as a matter of fact, this is still open for $\mathscr{P}^{ss}_\infty$, but unlikely).

The adequation-but-non-full-abstraction results are indeed the general case.

Example 169. By contrast also to the $\mathscr{D}_\infty$-case, Abramsky–Ong's quasi-extensional model (Example 146) is not fully-abstract w.r.t. S^l [2]. Similarly, the two models of λ-calculus which are presented in Examples 155 and 157 are adequate but not fully abstract for the lazy notion of reduction underlying map theory.

Question. Which kind of positive extensional webs generate fully abstract models w.r.t. S_h?

It is worth quoting here similar results for the *call-by-value strategy* and for the *λI-calculus*; in both cases the notion of model is slightly different (not any two β-equivalent terms have the same interpretations). Adequate (webbed) models for the call-by-value strategy are provided in [41, 56]. The second paper contains also a fully abstract model for the left reduction on λI-terms, which is generated by a web similar to that of *Norm*.

This kind of problems arose 20 years ago for *PCF* [95], within typed λ-calculi, and concerns also extended languages: it is impossible to find fully abstract (non-syntactical) models for "pure" calculi in the *typed* setting (except if one changes radically the notion of model): the point is that models always code "non-sequential" functions while "pure" terms (or "computable functions") behave "sequentially".

However the addition of one non-sequential operator to the syntax (like a "parallel or") is often sufficient to force full abstraction [95, 2, 23]. By the way, even in the typed case one does not really know what it means for a function (or for a functional) to be sequential (stability and strong stability are only weak approximation of this

informal notion), and in the untyped case things are still much more confused. The only thing which is (intuitively) clear is that λ-calculus (whether typed or untyped) *is* sequential!

6.8. Easiness and forcing revisited

Ω is in fact η-easy in the sense that $\{\Omega = t \text{ and } \varepsilon = I\}$ is consistent for all t, and, also, its easiness can be proved using only models of the stable semantics; this comes from the fact that the forcing technique of [115] (which originates in [6]) can be extended to extensional K-models [64, 65] and can be also adapted to G-models [68, 71].

The present scope of the forcing technique is however quite limited since it only applies to terms u for which, like for Ω, there is a good control on the set of primes which possibly belong to the interpretation of u. Extensions of its fields of application would be useful. The only direction in which forcing considerations have been extended is, up to now, the following.

Easy families. Zylberajch [115] noticed that the forcing method was uniform enough to prove the simultaneous easiness of the members of some infinite family F of easy terms (containing Ω): for each t there is a graph model $\mathscr{G}_{F,t}$ satisfying $\{u = t \mid u \in F\}$. In such a case we will speak of an *easy family*. Zylberajch's easy family was isolated via typing considerations. Another example of an easy family is given by $F = \{\Omega v \mid v \in \Lambda\}$; here the easiness of F is trivially derived from that of Ω, and hence realized via graph models, since to satisfy $\{\Omega v = t \mid v \in \Lambda\}$ it is indeed enough to satisfy $\Omega = Kt$. It is worth noting here that the two easy terms Ω and ΩK do not form an easy family, since they cannot be both consistently equated to I.

*More recently, a very general family of easy terms, called *mute terms*, has been isolated by Berarducci, and proved to be an easy family by means of syntactic tools, including the infinite λ-calculus already mentioned [15]. Moreover Berarducci provided a *syntactic* model which equates mute terms to Ω, and only them.

Question 1. Is there a *webbed* model which behaves like this?

A stronger question is:

Question 2. Can the simultaneous easiness of mute terms be proved by means of forcing?

A technical variant is:

***Question.** Is the typing system recently introduced by Dezani and Berarducci in [16] informative enough to guide the construction of a webbed model equating mute terms to t, for any t?

Acknowledgements

The author is grateful to S. van Bakel, M. Dezani-Ciancaglini, and especially to F. Honsell and S. Ronchi Della Rocca, for very useful and stimulating discussions, and for relevant pointers to the literature on filter models and intersection type assignment systems. She is also grateful to a referee for a very careful revision and for much pertinent advice. This advice has in fact led to a complete rewriting of the part on models. She would finally like to thank M. Schonfield and A. Salibra for pertinent remarks, and M. Schonfield and D. Pelletier for kindly revising the English of substantial parts of the two successive versions.

References

[1] S. Abramsky, Domain theory in logical form, Ann. Pure Appl. Logic 51 (1991) 1–77.
[2] S. Abramsky, C.H. Luke Ong, Full abstraction in the lazy lambda-calculus, Inform. and Comput. 105 (1993) 159–267.
[3] F. Alessi, Sistemi di informazione applicativi, teoria dei domini e modelli del λ_calculo, Tesi di Dottorato di Ricerca in Informatica, Università di Milano e Torino, 1990.
[4] F. Alessi, Type preorders, Trees in Algebra and Programming, CAAP'94 (Edinburg), Lecture Notes in Computer Science, Vol. 787, Springer, Berlin, 1994, pp. 37–51.
[5] J. Backus, Can programming be liberated from the von Neumann style?, A functional Style and its algebra of programs, Comm. ACM (Association of Computing Machinery) 21 (8) (1978) 613–641.
[6] J. Baeten, B. Boerboom, Omega can be anything it should not be, Proc. Koninklijke Nederlandse Akademie van Wetenschappen, Serie A, Indag. Mathematicae, Vol. 41, 1979, pp. 111–120.
[7] S. van Bakel, Complete restrictions of the intersection type discipline, Theoret. Comput. Sci. 102 (1992) 135–163.
[8] H.P. Barendregt, The Lambda-Calculus, its Syntax and Semantics, revised ed. Studies in Logic, Vol. 103, North-Holland, Amsterdam, 1984.
[9] H.P. Barendregt, The impact of the λ-calculus in logic and computer science, Bull. Symbolic Logic 3 (2) (1997) 181–215.
[10] H. Barendregt, M. Coppo, M. Dezani-Ciancaglini, A filter model and the completeness of type assignment, J. Symbolic Logic 48 (4) (1983) 931–940.
[11] O. Bastonero, Modèles fortement stables du λ-calcul et résultats d'incomplétude, Thèse, Université de Paris 7, December 1996.
[12] O. Bastonero, Equational incompleteness and incomparability results for λ-calculus semantics, J. Symbolic Logic, submitted.
[13] O. Bastonero, X. Gouy, Stabilité forte et incomplétude de la classe des modèles stables du λ-calcul, Notes C.-R. Acad. Sci. Sér. I 322 (10) (1996) 905–908.
[14] O. Bastonero, X. Gouy, Strong stability and the incompleteness of stable models of λ-calculus, Ann. Pure Appl. Logic 100 (1999) 247–277.
[15] A. Berarducci, Infinite λ-calculus and non-sensible models, in: A. Ursini, P. Agliano (Eds.), Logic and Algebra, Lecture Notes in Pure and Applied Mathematics, Vol. 180, Marcel Dekker Inc, New York, 1996, p. 339–378.
[16] A. Berarducci, M. Dezani-Ciancaglini, Infinite λ-calculus and Types, Theoret. Comput. Sci. 212 (1999) 29–75.
[17] A. Berarducci, B. Intriglia, Some new results on easy λ-terms, Theoret. Comput. Sci. 121 (1993) 71–88.
[18] S. Berardi, C. Berline, Building continuous webbed models for System *F*, preprint October 1998, to appear in ENTCS (2000).
[19] C. Berline, K. Grue, A kappa-denotational semantics for Map Theory, in ZFC+SI, Theoret. Comput. Sci. 179 (1997) 137–202 (the index, accidently cut at publication, Theoret. Comput. Sci. 211 (1999) 397–398.

[20] G. Berry, Stable models of Typed λ-calculi, Proc. 5th Internat. Coll. on Automata, Languages, and Programming, Lecture Notes in Computer Science, Vol. 62, Springer, Berlin, 1978, pp. 72–89.
[21] G. Berry, Modèles complètement adéquats et stables des λ-calculs typés, Thèse de Doctorat d'état, Université de Paris 7, 1979.
[22] C. Böhm, Alcune proprietà delle forme $\beta\eta$-normali nel λ-K-calcolo, Publicazioni dell'Istituto per le Applicazioni del Calcolo 696, Roma, 1968, 19p.
[23] G. Boudol, Lambda-calculi for (strict) parallel functions, Inform. and Comput. 108 (1994) 51–127.
[24] A. Bucciarelli, Sequential models of PCF: some contributions to the domain theoretic approach to full abstraction, Dottorato di Ricerca in Informatica, Università di Pisa-Genova-Udine, 1993.
[25] A. Bucciarelli, T. Ehrhard, A theory of sequentiality, Theoret. Comput. Sci. 113 (1993) 273–291.
[26] A. Church, A set of postulates for the foundations of logic I, Ann. Math. 33 (1932) 346–366.
[27] A. Church, A set of postulates for the foundations of logic II, Ann. Math. 34 (1933) 839–864.
[28] A. Church, Lecture notes on mathematical logic Oct.35–Janv.36, Princeton University, manuscript, 1936.
[29] A. Church, The calculi of λ-conversion, Princeton University Press, Princeton, NJ, 1941.
[30] M. Coppo, M. Dezani-Ciancaglini, G. Longo, Applicative Info. Systems, Lecture Notes in Computer Science, Vol. 159, Springer, Berlin, 1983, pp. 35–64.
[31] M. Coppo, M. Dezani-Ciangaglini, B. Venneri, Principal type schemes and λ-calculus semantics, in: J.R. Hindley, J.P. Seldin (Eds.), To H.B. Curry, Essays in Combinatory Logic, λ-Calculus and Formalism, Academic Press, New York, 1980, pp. 535–560.
[32] M. Coppo, M. Dezani-Ciangaglini, F. Honsell, G. Longo, Extended type structures and filter λ-models, in: G. Longo, G. Lolli, A. Marcja (Eds.), Logic Colloquium'82, Elsevier Science Publishers, Amsterdam, 1984, pp. 241–262.
[33] M. Coppo, M. Dezani-Ciangaglini, M. Zacchi, Type theories, normal forms and D_∞ lambda models, Inform. and Comput. 72 (2) (1987) 85–116.
[34] H.B. Curry, Functionality in combinatory logic, Proc. Natl. Acad. Sci. USA 20 (1934) 584–590.
[35] R. David, in: Every unsolvable λ-term admits a decoration, preprint Juillet 98, Proc. TLCA'99, Lecture Notes in Computer Science, Vol. 1581, Springer, Berlin, 1999.
[36] R. David, K. Nour, A syntactical proof of the operational equivalence of two λ-terms, Theoret. Comput. Sci. 180 (1997) 371–375.
[37] T. Drakengren, A decidable canonical representation of the compact elements in Scott's reflexive domain in P_ω, Theoret. Comput. Sci. 193 (1998) 181–195.
[38] M. Dezani-Ciancaglini, I. Margaria, A characterization of F-complete type assignments, Theoret. Comput. Sci. 45 (1986) 121–157.
[39] P. Di Gianantonio, F. Honsell, An abstract notion of application, in TLCA'93, Lecture Notes in Computer Science, Vol. 664, Springer, Berlin, 1993, pp. 124–138.
[40] P. Di Gianantonio, F. Honsell, G. Plotkin, Uncountable limits and the λ-calculus, Nordic J. Comput. 2 (1995) 126–145.
[41] L. Egidi, F. Honsell, S. Ronchi della Rocca, Operational, denotational and logical descriptions: a case study, Fund. Inform. XVI (1992) 149–169.
[42] T. Ehrhard, Hypercoherences: a strongly model of linear logic, in: J.Y. Girard, Y. Lafont, L. Regnier (Eds.), Advances in Linear Logic, London Mathematical Society, Lecture Note serie, Vol. 222, Cambridge Univ. Press, Cambridge, 1995, pp. 83–108.
[43] E. Engeler, Algebras and combinators, Algebra Univ. 13 (3) (1981) 289–371.
[44] S. Feferman, Towards useful type-free theories I, J. Symbolic Logic 49 (1984) 75–111.
[45] R.C. Flagg, G. Myhill, A type-free system extending *ZFC*, Ann. Pure Appl. Logic 43 (1989) 79–97.
[46] S. Fortune, D. Leivant, M. O'Donnell, The expressiveness of simple and second order type structures, J. ACM 30 (1983) 151–185.
[47] J.Y. Girard, Une extension de l'interprétation de Gödel à l'analyse, in: J. Fenstadt (Ed.), Proc. 2nd Scandinavian Logic Symp., North-Holland, Amsterdam, 1971, pp. 63–92.
[48] J.Y. Girard, Interprétation fonctionnelle et élimination des coupures de l'Arithmétique d'ordre supérieure, Thèse, Université Paris 7, 1972.
[49] J.Y. Girard, The system F of variable types, fifteen years later, Theoret. Comput. Sci. 45 (1986) 159–192.
[50] J.Y. Girard, Y. Lafont, P. Taylor, Proofs and Types, Cambridge tracts in Computer Science, Vol. 7, Cambridge Univ. Press, Cambridge, 1989.

[51] X. Gouy, Etude des théories équationnelles et des propriétés algébriques des modèles stables du λ-calcul, Thèse, Université de Paris 7, December 1995.
[52] X. Gouy, Une extension du théorème de Hyland et Wadsworth à une classe de modèles du λ-calcul pur, Notes C.-R. Acad. Sci. Sér. I 322 (1996) 419–422.
[53] X. Gouy, Y. Jiang, Universal retractions on DI-domains, Inform. and Comput. 119 (2) (1995) 252–257.
[54] K. Grue, Map Theory, Theoret. Comput. Sci. 102 (1) (1992) 1–133.
[55] J. van Heijenoort, From Frege to Gödel, A Source Book in Mathematical Logic 1879-1931, Harvard Univ. Press, Cambridge, MA, 1967, 3rd ed. 1977.
[56] F. Honsell, M. Lenisa, Some results on the Full Abstraction problem for Restricted Lambda Calculi, in: M. Dezani-Ciancaglini, G. Plotkin (Eds.), Proc. 18th Symp. on MFCS'93 (Poland), Lecture Notes in Computer Science, Vol. 711, Springer, Berlin, 1993, pp. 84–104.
[57] F. Honsell, M. Lenisa, Final semantics for untyped λ-calculus, TLCA'95, Lecture Notes in Computer Science, Vol. 902, Springer, Berlin, 1995, pp. 249–265.
[58] F. Honsell, S. Ronchi della Rocca, Reasoning about interpretations in qualitative λ-models, in: M. Broy, C. Jones (Eds.), Proc. IFIP Conf. on Programming concepts and methods, North-Holland, Amsterdam, 1990, pp. 505–521.
[59] F. Honsell, S. Ronchi della Rocca, An approximation theorem for topological λ-models and the topological incompleteness of λ-calculus, J. Comput. System Sci. 45 (1992) 49–75.
[60] J. Hindley, J. Seldin, Introduction to Combinators and λ-Calculus, Cambridge University Press, Cambridge, 1986.
[61] M. Hyland, A syntactic characterization of the equality in some models for the lambda-calculus, J. London Math. Soc. 12 (2) (1976) 361–370.
[62] C. Jacopini, A condition for identifying two elements in whatever model of combinatory logic, in: C. Bohm (Ed.), Lecture Notes in Computer Science, Vol. 37, Springer, Berlin, 1975.
[63] C. Jacopini, M. Venturini-Zilli, Easy terms in the lambda-calculus, Ann. Soc. Math. Polonae, Ser. IV; Fund. Inform. VIII (2) (1985) 225–233.
[64] Y. Jiang, Consistance et inconsistance de théories de λ-calcul étendus, Thèse, Université Paris 7, Juin 1993.
[65] Y. Jiang, Consistency of a λ-theory with n-tuples and easy terms, Arch. Math. Logic 34 (2) (1995) 79–96.
[66] R. Kerth, $2^{\aleph_0}$ modèles de graphes non équationnellement équivalents, Notes C.-R. Acad. Sci. 318 (1994) 587–590.
[67] R. Kerth, Définissabilité dans les domaines réflexifs, Notes C.-R. Acad. Sci. Sér. I 318 (1994) 685–688.
[68] R. Kerth, Isomorphisme et équivalence équationnelle entre modèles du λ-calcul, Thèse, Université de Paris 7, October 1995.
[69] R. Kerth, Isomorphism and equational equivalence of continuous λ-models, Studia Logica 61 (3) (1998) 403–415.
[70] R. Kerth, The interpretation of unsolvable terms in models of pure λ-calculus, J. Symbolic Logic 63 (1998) 1529–1548.
[71] R. Kerth, Forcing in stable models of untyped λ-calculus, Indagationes Mathematicae 10 (1999) 59–71.
[72] R. Kerth, On the construction of stable models of untyped λ-calculus, Theoret. Comput. Sci., submitted.
[73] S.C. Kleene, Realizability: a retrospective survey, Proc. Cambridge Summer School in Mathematical Logic, Cambridge 1971, Lecture Notes in Mathematics, Vol, 337, Springer, Berlin, 1973, pp. 95–112.
[74] S.C. Kleene, Origins of recursive function theory, Ann. History Comput. 3 (1) (1981) 52–67.
[75] S.C. Kleene, J.B. Rosser, The inconsistency of certain formal logics, Ann. Math. 36 (2) (1935) 630–636.
[76] J.L. Krivine, Lambda-calcul, Types et modèles, Masson, Paris, 1990.
[77] J.L. Krivine, Lambda-Calculus, Types and Models, Ellis-Horwood, Chichester, 1993. (augmented english translation of the above).
[78] J.L. Krivine, Fonctions, Programmes et Démonstrations, Gaz. Math. Soc. Math. Fr. 60 (1994) 63–73.
[79] J.L. Krivine, Classical logic, storage operators and second-order lambda-calculus, Ann. Pure Appl. Logic 68 (1994) 53–78.
[80] J.L. Krivine, About classical logic and imperative programming, Ann. Math. Artificial Intelligence 16 (1996) 405–414.
[81] J.L. Krivine, M. Parigot, Programming with proofs, J. Inform. Process. Cybernet. EIK 26 (3) (1990) 149–167.

[82] K. Kunen, in: Set Theory, An Introduction to Independence proofs, Studies in Logic and the Foundations of Mathematics, Vol. 102, North-Holland, Amsterdam, 1964.
[83] P.J. Landin, The mechanical evaluation of expressions, Comput. J. 6 (1964) 308–320.
[84] K.G. Larsen, G. Winskel, Using Information Systems to Solve Recursive Domain Equations, Lecture Notes in Computer Science, Vol. 173, (Semantics of data types), Springer, Berlin, 1984, pp. 109–130.
[85] D. Leivant, Reasoning about functional programs and complexity classes associated with type discipline, 24th Annual Symp. on Foundations of Computer Science (1983) 460–469.
[86] G. Longo, Set-theoretical models of λ-calculus: theories, expansions and isomorphisms, Ann. Pure Appl. Logic 24 (1983) 153–188.
[87] M. Nielsen, G. Plotkin, G. Winskel, Petri nets, event structures and domains (part I), Theoret. Comput. Sci. 13 (1981) 85–108.
[88] P. Odifreddi, Classical Recursion Theory, Studies in Mathematical Logic, Vol. 125, North-Holland, Amsterdam, 1989.
[89] P. Odifreddi, Introduction, in: P. Odifreddi (Ed.), Logic in Computer Science, APIC Studies in Data Processing, Vol. 31, Academic Press, New York, 1990, pp. 1–17.
[90] M. Parigot, On the representations of data in lambda-calculus, in Proc. CSL'89 (Kaiserlautern), Lecture Notes in Computer Science, Vol. 440, Springer, Berlin, 1989, pp. 309–321.
[91] M. Parigot, P. Roziere, Constant time reduction in λ-calculus, Mathematical Foundations of Computer Science, Lecture Notes in Computer Science, Vol. 711, Springer, Berlin, 1993, pp. 608–617.
[92] D. Park, The Y combinator in Scott's Lambda-calculus models. Theory of Computation Report 13, Department of Computer Science, University of Warwick, 1976.
[93] G. Plotkin, A set-theoretical definition of application, Memorandum MIP-R-95, School of artificial intelligence, University of Edinburgh, 1972.
[94] G. Plotkin, Call-by-name, call-by-value and the λ-calculus, Theoret. Comput. Sci. 1 (1975) 125–159.
[95] G. Plotkin, LCF as a programming language, Theoret. Comput. Sci. 5 (1977) 223–357.
[96] G.D. Plotkin, Post-graduate lecture notes in advanced domain theory (incorporating the "Pisa notes", Department of Computer Science, University of Edinburgh, 1981.
[97] G. Plotkin, A power domain for countable non-determinism. Lecture Notes in Computer Science, Vol. 140, ICALP'82, Springer, Berlin, 1982, pp. 418–428.
[98] G. Plotkin, Set-Theoretical and other elementary models of the λ-calculus (in a collection of contributions in honour to Corrado Böhm on the occasion of his 70th birthday), Theoret. Comput. Sci. 121 (1–2) (1993) 159–192.
[99] G. Plotkin, A semantic for static type inference, Inform. and Comput. 109 (1–2) (1994) pp. 256–299. (Special issue TACS'91).
[100] G. Plotkin, On a question of Friedman, preprint, 1995.
[101] J. Reynolds, Toward a theory of type structures, Colloque sur la programmation, Lecture Notes in Computer Science, Vol. 19, Springer, Berlin, 1974, pp. 408–425.
[102] H. Schellinx, Isomorphisms and non isomorphisms of graph models, J. Symbolic Logic 56 (1) (1991) 227–249.
[103] D. Scott, Continuous lattices, in: F.W. Lawvere (Ed.), Lecture Notes in Mathematics, Vol. 274, Topos, Algebraic geometry and Logic, Proc. Dalhousie Conf., Springer, Berlin, 1972, pp. 97–136.
[104] D. Scott, Some philosophical issues concerning the theory of combinators, in: C. Bohm (Ed.), λ-Calculus and Computer Science Theory, Lecture Notes in Computer Science, Vol. 37, Springer, Berlin, 1975.
[105] D. Scott, Combinators and classes, in: C. Bohm (Ed.), λ-Calculus and Computer Science Theory, Lecture Notes in Computer Science, Vol. 37, 1975, pp. 1–26.
[106] D. Scott, Data types as lattices, SIAM J. Comput. 5 (3) (1976) 522–587.
[107] D. Scott, Lambda-calculus: some models, some philosophy, in: J. Barwise, H.J. Keisler, K. Kunen (Eds.), The Kleene Symp., North-Holland, Amsterdam, 1980, pp. 223–265.
[108] D. Scott, Some ordered sets in computer science, in: I. Rival (Ed.), Ordered Sets, Proc. NATO Advanced Study Institute, (Banff, Canada 1981), Dordrecht, Reidel, 1981, pp. 677–718.
[109] D. Scott, Domains for denotational semantics, Lecture Notes in Computer Science, Vol. 140, (ICALP'82) Springer, Berlin, 1982, pp. 577–613.
[110] P. Selinger, Order-incompleteness and finite lambda-models, in Proc. 11th Annual IEEE Symp. on Logic and Computer Science (LICS'96) (1996) 422–439.

[111] J. Stoy, Denotational semantics: the Scott-Strachey approach to Programming language Theory, MIT Press Series in Computer Science, 3rd printing, 1985.
[112] C.P. Wadsworth, The relation between computational and denotational properties for Scott's D_∞-models of λ-calculus, SIAM J. Comput. 5 (3) (1976) 488–521.
[113] C.P. Wadsworth, Approximate reduction and λ-calculus models, SIAM J. Comput. 7 (3) (1978) 337–356.
[114] G. Winskel, in: An introduction to event structures, Lecture notes for the REX Summer School in Temporal Logic, Lecture Notes in Computer Science, Vol. 354, Springer, Berlin, 1988.
[115] C. Zylberajch, Syntaxe et sémantique de la facilité en λ-calcul, Thèse, Université Paris 7, 1991.