Basic Research in Computer Science

BRICS DS-96-4 T. Braüner: An Axiomatic Approach to Adequacy

An Axiomatic Approach to Adequacy

Torben Braüner

BRICS Dissertation Series **DS-96-4**

ISSN 1396-7002 **November 1996**

See back inner page for a list of recent publications in the BRICS Dissertation Series. Copies may be obtained by contacting:

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK - 8000 Aarhus C
Denmark**

**Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk**

BRICS publications are in general accessible through World Wide Web and anonymous FTP:

`http://www.brics.dk/`
`ftp://ftp.brics.dk/pub/BRICS`

An Axiomatic Approach to Adequacy

Torben Braüner

Ph.D. Dissertation

Department of Computer Science
University of Aarhus
Denmark

An Axiomatic Approach to Adequacy

A Dissertation
Presented to the Faculty of Science
of the University of Aarhus
in Partial Fulfilment of the Requirements for the
Ph.D. Degree

by
Torben Braüner
July 1996

To Anne and Sara

Abstract

This thesis studies adequacy for PCF-like languages in a general categorical framework. An adequacy result relates the denotational semantics of a program to its operational semantics; it typically states that a program terminates whenever the interpretation is non-bottom. The main concern is to generalise to a linear version of PCF, called LPCF, the adequacy results known for standard PCF, keeping in mind that there exists a Girard translation from PCF to LPCF with respect to which the new constructs should be sound. General adequacy results have usually been obtained in order-enriched categories, that is, categories where all hom-sets are typically cpos, maps are assumed to be continuous and fixpoints are interpreted using least upper bounds. One of the main contributions of the thesis is to propose a completely different approach to the problem of axiomatising those categories which yield adequate semantics for PCF and LPCF. The starting point is that the only unavoidable assumption for dealing with adequacy is the existence, in any object, of a particular "undefined" point denoting the non-terminating computation of the corresponding type; hom-sets are pointed sets, and this is the only required structure. In such a category of pointed objects, there is a way of axiomatising fixpoint operators: They are maps satisfying certain equational axioms, and furthermore, satisfying a very natural non-equational axiom called rational openness. It is shown that this axiom is sufficient, and in a precise sense necessary, for adequacy. The idea is developed in the intuitionistic case (standard PCF) as well as in the linear case (LPCF), which is obtained by augmenting a Curry-Howard interpretation of Intuitionistic Linear Logic with numerals and fixpoint constants, appropriate for the linear context. Using instantiations to concrete models of the general adequacy results, various purely syntactic properties of LPCF are proved to hold.

Acknowledgements

I am grateful to my supervisor, Glynn Winskel, for guidance and support. In particular, I thank him for encouragement at a critical time. Also thanks to Glynn for giving me freedom to pursue my own research interests.

Thanks to Doug Gurr for helping get me started. I have benefited greatly from conversations with Samson Abramsky, Gavin Bierman, Gian Luca Cattani, Marcelo Fiore, Claudio Hermida, Martin Hyland, François Lamarche, Søren Bøgh Lassen, Guy McCusker, Valeria de Paiva, Andy Pitts, Gordon Plotkin, John Power, Alex Simpson and Sergei Soloviev. I am grateful to Gavin, Valeria and Andy for hospitality whenever I visited Cambridge. Thanks to Marcelo for his hospitality during my Edinburgh visit.

I would also like to thank my evaluation commitee - Thomas Ehrhard, Mogens Nielsen and Valeria de Paiva - for useful comments and suggestions. These are taken into account in this, the final version of the dissertation.

Thanks to Lars Arge and Allan Cheng for assistance with some typographical matters. The diagrams and proof-rules are produced using Paul Taylor's macros.

I would like to acknowledge the financial support of **BRICS**[1] and Aarhus University Research Foundation in the form of studentships. The CLICS project has contributed towards travel expenses on several occasions.

I am most thankful for the love and support of my wife Anne and for the shared joy of our daughter Sara.

[1] Basic Research in Computer Science, Centre of the Danish National Research Foundation.

Contents

Chapter 1

Introduction

This first chapter contains an introduction to the thesis. In Section 1.1 we give a short chronological account of the work to be presented. Section 1.2 introduces the traditional order-theoretic approach to denotational semantics of the programming language PCF, and in Section 1.3 the problem of giving an adequate denotational semantics is considered from an axiomatic point of view. In Section 1.4 the programming language LPCF, which is a linear version of PCF, is introduced and given an adequate categorical semantics. In Section 1.5 we give an overview of the thesis, and finally, in Section 1.6 prerequisites and notation are discussed.

1.1 Chronology

Historically, this thesis grew out of an attempt to give a denotational semantics to a programming language based on a Curry-Howard interpretation of Intuitionistic Linear Logic augmented with numerals and recursion. After fixing the term language for Intuitionistic Linear Logic, a fixpoint constant appropriate for the linear context was added. Numerals were introduced later on. Considerations in concrete categories of cpos and dI-domains turned out to be very instructive in the development from Intuitionistic Linear Logic, considered as a pure logic, to what was to become the programming language LPCF (*L*inear *P*rogramming language for *C*omputable *F*unctions). This is witnessed by [Bra94a]. Proving adequacy for interpretations in the concrete categories did, however, turn out to be cumbersome, which initiated a search for general categorical adequacy results where only relevant properties of a model are taken into account.

Now, PCF (*P*rogramming language for *C*omputable *F*unctions) is a programming language based on a Curry-Howard interpretation of Intuitionistic Logic, namely the well-known typed λ-calculus, augmented with numerals and recursion. It turns out that there are close connections between (a categorical model for) LPCF and (a categorical model for) PCF via an extension of the Girard Translation as shown in [Bra95a]. Given the connection between LPCF and PCF, it seemed appropriate to sort out proper categorical axioms for the simpler language of PCF before dealing with LPCF. A crucial step in this respect was to leave out the order-structure motivated by the point of view that partiality is the fundamental notion from which order-structure is to be derived. These considerations led to the results of [Bra97b].

After giving proper axioms for an adequate categorical semantics of PCF, we were finally in position to deal with LPCF. As to be expected there are close connections between axioms for adequacy on a categorical model for PCF and axioms for adequacy on a categorical model for LPCF. An adequate categorical semantics of LPCF is given in [Bra97a].

1.2 Denotational Semantics of PCF

This section introduces the traditional order-theoretic approach to denotational semantics of PCF. The reader not acquainted with the formal definition of PCF is advised to browse through Sec-

tion 7.1 before continued reading.

A characterising feature of *denotational* semantics of programming languages is that the piece of text constituting a computer program is considered to *denote* an appropriate mathematical object. This is witnessed by a distinction between the formal language in which the program is written, and the mathematical universe in which the denoted object exists, that is, a distinction between syntax and semantics. The notion of a distinction between syntax and semantics is inherited from logic and its history goes back at least as long as to the work of Frege, [Fre92].

In the area of programming languages, the notion of a distinction between syntax (a program) and semantics (an appropriate mathematical object denoted by the program) can be tracked to the until recently unpublished manuscript [Sco69] where Scott introduced PCF; inspired by the work of Strachey translating programming languages into the untyped λ-calculus (see the introduction to [Sto77]). This is the historical origin of denotational semantics of programming languages. The manuscript remained unpublished, but widely circulated, until it appeared in [Sco93]. The formal system of PCF consists of the typed λ-calculus augmented with booleans and numerals together with recursion where the usual reduction rules are replaced by a lazy operational semantics. It is similar to Gödel's System T, [Göd58, GLT89] which, however, deals with primitive recursive functions, whereas PCF deals with computable functions in general. The original presentation of PCF has exponential types and two ground types, namely types for numerals and booleans; however, we shall instead consider a version with only one ground type, namely a type for numerals. It is straightforward to represent booleans by numerals, so this does not make an essential difference. Later on we shall add product and sum types to the picture, but for now we will consider only exponential types and the type for numerals.

Now, Scott's pioneering idea was to interpret types of PCF as cpos and terms as continuous functions. A cpo[1] is a poset containing a bottom element $\bot$ where every increasing chain $\{x_i\}_{i\in\omega}$ has a least upper bound $\sqcup_{i\in\omega} x_i$, and a continuous function f is a monotone function that preserves the least upper bound of any increasing chain $\{x_i\}_{i\in\omega}$, that is, we have

$$f(\bigsqcup_{i\in\omega} x_i) = \bigsqcup_{i\in\omega} f(x_i).$$

The type for numerals N is then interpreted as ω with a bottom element adjoined. An exponential type $A \Rightarrow B$ is interpreted as the set $[\![A]\!] \Rightarrow [\![B]\!]$ of continuous functions from the interpretation $[\![A]\!]$ of A to the interpretation $[\![B]\!]$ of B equipped with the extensional order. The extensional order is defined as $f \sqsubseteq g$ iff

$$\forall x \in A.\ f(x) \sqsubseteq g(x)$$

for any continuous functions $f, g : A \to B$. This amounts to exponential types being interpreted as categorical exponentials in the model, that is, in the category of cpos and continuous functions. The justification for the interpretation of the numerals type is to consider elements of ω as "values" and the bottom element as "undefined". This corresponds to the possibilities when evaluating a program[2] of numerals type: It either terminates with a numeral as the outcome, or it does not terminate. When interpreting exponential types we simply have $f \sqsubseteq g$ iff the function g is more defined than the function f in the sense that whenever f is defined having a certain value, then g is also defined having the same value. In [Plo77] it was shown that termination of a program of numerals type actually corresponds to the interpretation being non-bottom, that is, we have

$$[\![t]\!] \neq \bot \ \Leftrightarrow\ t \Downarrow$$

for every program t of numerals type. In that setting a semantics with such a property is called adequate. The adequacy result thus expresses that the syntactic notion of termination coincides with the semantic notion of definedness.

[1] Later on we shall consider a notion of cpos and continuous functions where increasing chains are replaced by directed sets, but this difference is not essential for the work presented here.

[2] A closed term is called a *program*. This convention is not followed by all authors.

Now, the paper [Plo77] did not only deal with adequacy, it also introduced the notion of full abstraction. To consider this notion, we need to define the contextual preorder on programs: For every pair u and v of programs of type A we define $u \lesssim_A v$ iff

$$t[u/z] \Downarrow \;\Rightarrow\; t[v/z] \Downarrow$$

for every term t of numerals type with one free variable z of type A. An order-theoretic semantics of PCF is called fully abstract iff

$$[\![u]\!] \sqsubseteq [\![v]\!] \;\Leftrightarrow\; u \lesssim v.$$

for any pair u and v of programs of the same type. It is straightforward to show that adequacy is equivalent to the $\Rightarrow$ direction. The $\Leftarrow$ direction is, however, not satisfied by the cpo semantics given by Scott. The problem is the existence of programs, say u and v, that are indistinguishable by the contextual preorder, but have different interpretation. This has to do with the inherently sequential behaviour of PCF and at the same time the ability to define functions with a "parallel" character in the model: The programs u and v denote functions that can only be distinguished by applying them to the famous "parallel-or" function, which is not definable in PCF. Let B denote the poset obtained by adjoining a bottom element to the set of truth values $\{True, False\}$; the parallel-or function

$$por : B \times B \to B$$

is then defined to be the unique monotone extension of the usual disjunction function

$$\vee : \{True, False\} \times \{True, False\} \to \{True, False\}$$

such that $por(True, \bot) = True$ and $por(\bot, True) = True$.

The obstacle to full abstraction created by the parallel-or function led to the discovery of dI-domains and stable functions, [Ber78]. The idea is to interpret types of PCF as dI-domains and terms as stable continuous functions with the aim of removing the parallel-or function. A dI-domain is a non-empty poset where every finitely compatible subset X has a least upper bound $\sqcup X$, and furthermore, the poset has to be prime algebraic[3], that is, every element is the least upper bound of the prime elements below it, and the poset has to be finitary, that is, the set of elements below a finite element constitutes a finite set. A stable function f is a monotone function that preserves the greatest lower bound of any finitely compatible subset X, that is, we have $f(\sqcap X) = \sqcap f(X)$, and a continuous function f between dI-domains is a monotone function that preserves the least upper bound of any directed subset X, that is, we have $f(\sqcup X) = \sqcup f(X)$. The type for numerals N is then interpreted as ω with a bottom element adjoined; this is the same as with the cpo semantics. But the dI-domain semantics differs from the cpo semantics with respect to an exponential type $A \Rightarrow B$ which is interpreted as the set $[\![A]\!] \Rightarrow [\![B]\!]$ of stable continuous functions from $[\![A]\!]$ to $[\![B]\!]$ equipped with the stable order. The stable order is defined as $f \sqsubseteq g$ iff

$$\forall x, y \in A.\, x \sqsubseteq y \;\Rightarrow\; f(x) = f(y) \sqcap g(x)$$

for any stable continuous functions $f, g : A \to B$. This amounts to exponential types being interpreted as categorical exponentials in the model, that is, in the category of dI-domains and stable continuous functions. Note that the stable order is included in the extensional order. When interpreting exponential types we have $f \sqsubseteq g$ iff the behaviour of the function f is a part of the behaviour of the function g in a sense which will be made precise in Section 2.6 using the notion of trace. The dI-domain semantics does indeed rule out the parallel-or function; this can be seen by considering the compatible elements $(True, \bot)$ and $(\bot, True)$ from the domain of por, but it turns out that it is not fully abstract itself.

Recently, a fully abstract model for PCF has been discovered outside traditional order-theoretic domain theory, namely in the realm of games, [AJM96, HO96, Nic94]. The order of the game model coincides with the stable order of [Ber78] which corroborates the point of view that the dI-domain

[3] The original definition in [Ber78] has distributivity, that is, $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$ whenever y and z are compatible, instead of prime algebraicity, but in [Win87] it is shown that the two definitions are equivalent.

semantics is a major step towards full abstraction. In this thesis we are concerned with adequacy rather than full abstraction, so we will consider the above mentioned categories of cpos and dI-domains as primary examples of concrete models of PCF and thus leave game semantics to further work - see Chapter 11.

1.3 An Axiomatic Approach to Adequacy

This section introduces the axiomatic approach to a denotational semantics of PCF which will be dealt with in details in Chapter 6. We impose appropriate axioms on a categorical model with the aim of proving an adequacy result. We follow Scott's idea in assuming the presence of certain "undefined" maps with the role of being the interpretation of non-terminating terms, but the order-structure is left out motivated by the point of view that partiality is the fundamental notion from which order-structure should be derived (which is in accordance with [Fio94b]). This is different from previous approaches where some kind of order-theoretic structure has been considered to be part of an adequate categorical model for PCF. For example, in [BCL86] **cpo**-enrichment is taken to be part of a categorical model; this means that every hom-set is equipped with a cpo structure such that composition is continuous. In [AJM96] this is replaced by rationality, which means that the categorical model in question is **poset**-enriched such that each poset has a bottom element, and moreover, every increasing chain of the form $\{\bot; f^n\}_{n\in\omega}$ for some endofunction f is assumed to have a least upper bound which is preserved by composition (note that we write composition as $g; h$ rather than $h \circ g$). Our motivation for adhering to the point of view that partiality is the fundamental notion from which order-structure should be derived is that no order-theoretic notions are used in the formal definition of PCF, but an appropriate order-structure is indeed derivable, namely the observational preorder on terms. Moreover, the point of view is corroborated by the observation that our categorical model (to be introduced below) induces a **poset**-enrichment in a canonical way, and, under appropriate circumstances, even rationality; this is with respect to the observational preorders on hom-sets quotiented down to posets.

Historically, the axioms of our categorical model were discovered essentially by extracting the properties of the order-structure of a rational category, and a **cpo**-enriched category as well, which are actually used to obtain an adequacy result. The order-structure is used to define fixpoints such that for any endomap f the chain of finite approximants $\{\bot; f^n\}_{n\in\omega}$ actually approximates the fixpoint $f^\sharp$ in an appropriate sense. So rather than assume the presence of an order-structure, we axiomatise what is actually needed; namely fixpoints which are approximated by their (formal) finite approximants in an appropriate sense stated in non-order-theoretic terms. The starting point of our axiomatisation is a cartesian closed category where for each object B we assume the presence of an "undefined" map $\bot_B\colon 1 \to B$ together with a *fixpoint operator*, that is, an operation which to a map $f : B \to B$ assigns a map $f^\sharp : 1 \to B$ such that $f^\sharp = f^\sharp; f$. We also assume the presence of an object N together with appropriate maps for dealing with numerals. The categorical axioms consist of a couple of equational (that is, first order) ones together with one non-equational, namely the axiom of rational openness on each fixpoint operator. A fixpoint operator is *rationally open* with respect to an object P iff for all maps $f : B \to B$ and $g : B \to P$ it is the case that

$$f^\sharp; g \neq \bot \;\Rightarrow\; \exists n \in \omega.\ \bot; f^n; g \neq \bot\ .$$

Thus, the hidden essential property of an order-theoretic model has been revealed: Definedness of an expression is determined by the (formal) finite approximants to the involved fixpoints.

Having fixed an appropriate categorical model it is proved that the categorical semantics is *adequate*[4] in the sense that we have

$$[\![t]\!] \neq \bot \;\Rightarrow\; t \Downarrow$$

for every program t. If the program t is of numerals type, then we actually have a converse to adequacy. A type where this happens to be the case is called observable, that is, a type B is *observable* iff $t \Downarrow \Rightarrow [\![t]\!] \neq \bot$ for every program t of type B.

[4] This is stronger than the original notion of adequacy which only takes ground types into account.

It is possible to restrict the axiom of rational openness such that it is not only sufficient, but also necessary for the interpretation to be adequate. What is done, is essentially to restrict rational openness to maps definable in PCF. The "necessary" direction of this result relies on the *unwinding theorem*, which is a purely syntactic result saying that if t is a term of observable type with one free variable z of type $(A \Rightarrow A) \Rightarrow A$ then

$$t[\mathtt{Y}/z] \Downarrow \Leftrightarrow \exists n \in \omega \,.\, t[\mathtt{Y}^n/z] \Downarrow$$

where the terms $\mathtt{Y}^n_A$ are the *finite approximants* to the fixpoint constant $\mathtt{Y}_A$ defined by the stipulations

$$\begin{array}{rcl} \mathtt{Y}^0 & = & \Omega \\ \mathtt{Y}^{n+1} & = & \lambda f.f(\mathtt{Y}^n f) \end{array}$$

The unwinding theorem is intuitively clear: A fixpoint constant in a terminating program could possibly only be unwinded a finite number of times. We will prove the theorem using an instance of the adequacy result in the concrete category of cpos and continuous functions.

Very recently, there has been considerable progress in axiomatising sufficient conditions for obtaining full abstraction for PCF, [Abr96]. The intuitions leading to this work stem from the theory of games rather than from traditional order-theoretic domain theory, but rational openness is, however, recognised[5] as the essential notion for an axiomatic approach to adequacy.

Now, the numerals type is observable, but we cannot expect that to be the case for exponential types because the bottom elements here do not correspond to non-termination in general; for example, the terminating term $\lambda x.\Omega$ is interpreted as bottom. We stress that what we do in this thesis is consider the question of adequacy in the situation where exponential types are interpreted as categorical exponentials in the model. But it should be mentioned that it is possible to dodge the impossibility of obtaining a converse to adequacy at exponential types by introducing a monad $(\Leftrightarrow)_\perp$ with appropriate properties and then interpret types in a way that reflects the evaluation strategy following the ideas put forward in [Mog89]. For example, if the evaluation strategy is eager, then the type $A \Rightarrow B$ is interpreted as $[\![A]\!] \Rightarrow [\![B]\!]_\perp$ and terms are interpreted as maps in the Kleisli category induced by the monad. This is the approach taken in [Gun92, Win93] where PCF-like languages are interpreted in the category of cpos and continuous functions using the usual "lift" monad. The use of a monad to give an adequate interpretation with the property that the adequacy result has a converse at all types is implicit in [FP94, Fio94a] where a categorical semantics is given for FPC, which is the λ-calculus augmented with recursive types and equipped with an operational semantics. The starting point is a categorical notion of partial maps put forward in [Mog86, Ros86] together with **cpo**-enrichment. A comparison with our approach shows that they obtain a stronger result, namely an adequacy result which has a converse at all types, at the expense of having to use a more technically involved categorical notion of partial maps; and moreover, they take partiality as well as order as primitive notions, whereas we only take a notion of partiality as primitive. We shall not follow the monad-track here; in Chapter 9 we shall indeed consider the question of adequacy for a linear version of PCF, namely LPCF, where it turns out that the lift functor is not a monad on any of the canonical concrete models of cpos and dI-domains, and moreover, there does not seem to be any other appropriate monads around.

1.4 Introduction to LPCF

This section introduces the programming language LPCF together with an adequate categorical interpretation. This issue will be dealt with in details in Chapter 9.

LPCF is a programming language based on a Curry-Howard interpretation of Intuitionistic Linear Logic, the linear λ-calculus, in the same way as PCF is based on a Curry-Howard interpretation of Intuitionistic Logic, the well-known λ-calculus. The purpose of LPCF is to give a linear account of computable functions.

[5] With due reference to [Bra97b].

Now, the fundamental idea of Linear Logic is to control the use of resources, which is witnessed by the absence of contraction and weakening

$$\frac{\Gamma, \Gamma, A, A, \Delta \Vdash B}{\Gamma, \Gamma, A, \Delta \Vdash B} \qquad \frac{\Gamma, \Gamma, \Delta \Vdash B}{\Gamma, \Gamma, A, \Delta \Vdash B}$$

This prevents us from considering the context Γ, Γ of a sequent $\Gamma, \Gamma \Vdash B$ as a *set* of formulae, but we have to consider it as a *multiset* of formulae instead. For the Curry-Howard interpretation of Intuitionistic Linear Logic, the linear λ-calculus, this has the consequence that every free variable of a typeable term occurs exactly once[6]. A restricted form of contraction and weakening is, however, available by having the proof-rules

$$\frac{\Gamma, \Gamma, !A, !A, \Delta \Vdash B}{\Gamma, \Gamma, !A, \Delta \Vdash B} \qquad \frac{\Gamma, \Gamma, \Delta \Vdash B}{\Gamma, \Gamma, !A, \Delta \Vdash B}$$

explicitly as part of Intuitionistic Linear Logic. A proof of $!A$ amounts to having a proof of A that can be used an arbitrary number of times. It follows that the linear λ-calculus has the operations of copying and discarding explicitly built into the syntax.

The programming language LPCF consists of the linear λ-calculus augmented with numerals and recursion, appropriate for the linear context, where the reduction rules induced by the Curry-Howard isomorphism are replaced by an operational semantics. It turns out that eager as well as lazy evaluation rules for terms of certain types behave properly with respect to an appropriate categorical model; for terms of other types the categorical model motivates a certain choice of evaluation rules. We shall consider an eager as well as a lazy operational semantics. The eager operational semantics is eager in the sense that we take the evaluation strategy used in each particular evaluation rule to be eager whenever there is a choice, etc. In most cases the evaluation rules of the eager operational semantics coincides with the ones given in [Abr90]; however, it is the case that the evaluation rules involving terms of ! types are different, which is dictated by the categorical model. Also the way of introducing recursion in LPCF is motivated by interpretation in an appropriate categorical model: The interpretation of (the evaluation rule for) the linear fixpoint constant of LPCF corresponds to the interpretation of (the evaluation rule for) the fixpoint constant of PCF in the induced Kleisli category.

After having introduced the programming language LPCF we will give it a categorical semantics. The starting point is the categorical model for Intuitionistic Linear Logic given in [BBdPH92b] called a *linear category*, where for each object B we assume the presence of a map $\bot_B \colon I \to B$ together with a *linear fixpoint operator*, that is, an operation which to a map $f : !B \to B$ assigns a map $f^\sharp : I \to B$ such that $f^\sharp = \gamma(f^\sharp); f$. It is assumed that all maps are *strict*, that is, we have $\bot_A; g = \bot_B$ for any map $g : A \to B$. We also assume the presence of an object N together with appropriate maps for dealing with numerals. A linear fixpoint operator corresponds to a fixpoint operator in the Kleisli category induced by the ! comonad of the linear category, which enables us to import the notion of rational openness to the linear setting. Two concrete linear categories satisfying the axioms of the categorical model are given, namely the category of cpos and strict continuous functions, and the category of dI-domains and linear stable functions. In [HP90] it is shown that a cartesian closed category with finite sums and fixpoint operators is *inconsistent*, that is, it is equivalent to the category consisting of one object and one map. But both of the above mentioned concrete linear categories of cpos and dI-domains have finite sums and linear fixpoint operators; so the presence of linear fixpoint operators in a linear category is consistent with the presence of finite sums. Thus, the inconsistency of recursion with this standard construct vanishes when we go to a linear context, which is in accordance with [Plo93].

Having fixed an appropriate categorical model for LPCF it is proved that the categorical semantics is adequate whether we consider the eager or the lazy operational semantics. Adequacy thus follows from the categorical results in the above mentioned concrete linear categories of cpos and dI-domains. The adequacy results make use of what we have called Generalised LPCF, which

[6] Where we disregard the additive fragment.

essentially consists of the typing rules of LPCF extended with an extra context dealt with in an additive fashion.

The introduction of Generalised LPCF enables us to state and prove an unwinding theorem for LPCF. Here we make use of an instance of the adequacy result in the concrete linear category of cpos and strict continuous functions. Using the unwinding theorem it is proved that rational openness, when restricted to definable maps, is not only sufficient but also necessary for adequacy.

It is a notable feature of the categorical interpretation that it is adequate whether we consider the eager or the lazy operational semantics. Using instances of the adequacy results in the concrete linear category of cpos and strict continuous functions, this can be used to prove a purely syntactic result saying that the choice of evaluation strategy does not matter for the observable behaviour of programs of observable types, that is, if t is a program of observable type then

$$t \Downarrow \Leftrightarrow \; t \downarrow$$

where $\Downarrow$ and $\downarrow$ are the eager and the lazy evaluation relation, respectively. This is due to two reasons: The linear character of LPCF and the fact that in both of the operational semantics a program of ! type is always evaluated before it is discarded. So it is simply impossible to get rid of a non-terminating program without trying to evaluate it. For example, in the evaluation rule for application it does not matter whether or not an argument is evaluated before it is plugged into the body u of an abstraction $\lambda x.u$ because linearity entails that the variable x has to occur in u.

1.5 Overview of the Thesis

In this section an overview of the thesis is given. The main topics and the contributions of each chapter are made clear.

Chapter 2: The Semantic Picture. This chapter introduces categorical machinery needed for later together with some concrete categories. First we give an introduction to the notions of monoidal categories, monoidal functors and monoidal natural transformations, as well as to the notion of a comonoid in a monoidal category. Then the categorical model for Intuitionistic Linear Logic of [BBdPH92b] is recalled and various ramifications concerning the category of coalgebras and the Kleisli category are considered. Finally we introduce the category of cpos and strict continuous functions and the category of dI-domains and linear stable functions. This chapter provides the following contributions:

- A couple of observations from Section 2.4 seem to be new; namely Proposition 2.4.2 and Proposition 2.4.3 dealing with certain naturality properties that enable categorical interpretation of the finite sums of the λ-calculus in the Kleisli category.
- In Section 2.6 an account of the category of dI-domains and linear stable functions as a model for Intuitionistic Linear Logic in the sense of [BBdPH92b] is given. This is analogous to the account of the category of dI-domains and affine stable functions given in [Bra94a, Bra94b].

Chapter 3: The λ-Calculus. The primary goal of this chapter is to introduce the λ-calculus. It is shown how the λ-calculus appears as a Curry-Howard interpretation of Intuitionistic Logic. Given an appropriate categorical model, we recall how a categorical interpretation is induced.

Chapter 4: The Linear λ-Calculus. The primary goal of this chapter is to introduce the linear λ-calculus. It is shown how the linear λ-calculus appears as a Curry-Howard interpretation of Intuitionistic Linear Logic. Also we make a detour to Russell's Paradox with the aim of illustrating the fine grained character of Intuitionistic Linear Logic compared to Intuitionistic Logic. Given an appropriate categorical model, we recall how a categorical interpretation of the linear λ-calculus is induced. A generalisation of the linear λ-calculus which will be of use later on is introduced. Contributions of this chapter:

- In Section 4.3 a correct introduction rule for the !-modality of Intuitionistic Linear Logic is given. This was also given in [BBdPH92b] so we shall give an account of the history of the mentioned rule as well - see Section 4.4.
- The generalised linear λ-calculus introduced in Section 4.6 seems to be new. In certain respects it is similar to the variants of Girard's Logic of Unity, [Gir93] considered in [Wad93] and [Plo93]. The discovery the generalised linear λ-calculus is motivated by technical reasons: When in Chapter 9 we consider LPCF it enables us to prove adequacy, and moreover, it enables us to state and prove an unwinding theorem.

Chapter 5: The Girard Translation. This chapter introduces the Girard Translation, [Gir87] which is shown to be sound with respect to the categorical interpretations induced by an appropriate categorical model. Contributions of this chapter:

- In Section 5.2 the Girard Translation is proved to be sound with respect to an appropriate categorical model; this is essentially taken from [Bra95a, Bra95b]. It is a categorical generalisation of a result in [Gir87] showing that the Girard Translation is sound with respect to interpretation in a certain concrete category, namely the category of coherence spaces and linear stable functions.

Chapter 6: PCF - Semantic Issues. This chapter introduces appropriate machinery for giving the categorical model for PCF in Chapter 7. This amounts to a categorical notion of undefinedness, and, moreover, categorical notions of numerals and fixpoints. Also, the observational preorder on maps is considered. Contributions of this chapter:

- In Section 6.3 a couple of apparently new results about categorical fixpoint operators are given, namely Proposition 6.3.8 and Proposition 6.3.9. In this section also the axiom of rational openness on a fixpoint operator is introduced.
- In Section 6.5 it is shown that under appropriate circumstances a rationally open fixpoint operator induces a rational category when the quotients of the observational preorders on hom-sets are considered.

Chapter 7: The Programming Language PCF. This chapter introduces the programming language PCF. An adequacy result for PCF is given using a non-order-theoretic categorical model. Observable types and an unwinding theorem are considered. Also, we make a detour to syntactic notions of rationality and rational openness. This chapter provides the following contributions:

- In Section 7.3 an adequacy result for PCF is given using a non-order-theoretic categorical model. The essential ingredient of the categorical model is a rationally open fixpoint operator. This is taken from [Bra97b]. The result can be seen as a non-order-theoretic categorical generalisation of the original adequacy result of [Plo77].
- In Section 7.5 we prove an unwinding theorem for PCF using a concrete instance of adequacy. This enables us to show that a restricted version of our axiom of rational openness is not only sufficient, but also necessary for the interpretation to be adequate. Essentially, what we do is we restrict rational openness to maps definable in PCF.

Chapter 8: LPCF - Semantic Issues. This chapter introduces appropriate machinery for giving the categorical model for LPCF in Chapter 9. This amounts to a categorical notion of undefinedness and to categorical notions of numerals and fixpoints, appropriate for the linear setting. Also, the observational preorder is considered. Contributions of this chapter:

- In Section 8.3 a categorical notion of fixpoints appropriate for the linear setting is given, and various related results are proved. Also, the axiom of rational openness on a linear fixpoint operator is introduced.

- In Section 8.4 a notion of rationality suitable for the linear setting is given. This stems from [Bra97a]. It is shown how under appropriate circumstances a rationally open linear fixpoint operator induces a rational linear category.

Chapter 9: The Programming Language LPCF. This chapter introduces the programming language LPCF, which is a linear version of PCF. We introduce a generalisation of LPCF needed for technical reasons: It enables us to prove adequacy using a non-order-theoretic categorical model and it enables us to state and prove an unwinding theorem. Also observable types are considered. This chapter provides the following contributions:

- In Section 9.1 the programming language LPCF is introduced, and an eager and a lazy operational semantics is given in Section 9.2 and Section 9.3, respectively. The choice of evaluation rules for terms of certain types is motivated by interpretation in an appropriate categorical model; in the case of terms of ! types this dictates evaluation rules which are different from the rules of [Abr90].
- In Section 9.7 and Section 9.8 adequacy results for eager and lazy LPCF are proved using a non-order-theoretic categorical model. The essential ingredient of the categorical model is a rationally open linear fixpoint operator.
- Using concrete instances of the adequacy results, in Section 9.9 we show that the choice of evaluation strategy for LPCF does not matter for observable behaviour of terms of observable types.
- In Section 9.10 we prove an unwinding theorem for LPCF using a concrete instance of an adequacy result. This depends crucially on Generalised LPCF. Using the unwinding theorem it is shown that a restricted version of our axiom of rational openness is not only sufficient, but also necessary for the interpretation to be adequate. We essentially restrict rational openness to definable maps.

Chapter 10: Extension of the Girard Translation. This chapter extends the Girard Translation of Chapter 5 to a translation from PCF to LPCF, which is shown to be sound with respect to the categorical interpretations induced by an appropriate categorical model. This chapter provides the following contributions:

- In Section 10.1 the Girard Translation is extended to a translation from PCF to LPCF which in Section 10.2 is proved sound with respect to an appropriate categorical model.
- Using a concrete instance of the soundness result together with concrete instances of the adequacy results for PCF and LPCF in Section 10.3 it is shown that the extended Girard Translation preserves and reflects evaluation of programs of ground type N.

Chapter 11: Further Work. This chapter outlines some possibilities for extensions of our work.

1.6 Prerequisites and Notation

The reader of this thesis is assumed to be familiar with the basic notions of Intuitionistic Logic, the λ-calculus, category theory and denotational semantics of programming languages. Some experience with a functional programming language is useful.

The book [GLT89] gives an excellent introduction to Intuitionistic Logic and the λ-calculus. The textbook [BW90] is recommended as an introduction to category theory. Also the standard textbook [Mac71] is useful. A detailed introduction to the categorical notions relevant for Intuitionistic Linear Logic can be found in the dissertation [Bie94]. The textbook [Win93] is recommended to the reader not familiar with denotational semantics of programming languages.

A remark on notation: In order to avoid a proliferation of symbols we denote logical constructs using appropriate categorical notation. For example, the conjunction of Intuitionistic Logic and the additive conjunction of Intuitionistic Linear Logic are both denoted $\times$.

Chapter 2

The Semantic Picture

This chapter introduces categorical machinery needed for later. Also some concrete categories are given. Section 2.1 contains an introduction to the notions of monoidal categories, monoidal functors and monoidal natural transformations, as well as to the notion of a comonoid in a monoidal category. Readers familiar with these notions are advised to skip the section. In Section 2.2 the categorical model for Intuitionistic Linear Logic from [BBdPH92b] is recalled. Various ramifications concerning the category of coalgebras and the Kleisli category can be found in Section 2.3 and Section 2.4, respectively. In Section 2.5 the category of cpos and strict continuous functions is introduced, and in Section 2.6 an account of the category of dI-domains and linear stable functions is given.

2.1 Monoidal Categories

In this section we will recall the notions of monoidal categories, monoidal functors and monoidal natural transformations, originally introduced in [EK66]. We will also recall the notion of a comonoid in a monoidal category, which is dual to the notion of a monoid as given in [Mac71].

Definition 2.1.1 A *monoidal category* is a 6-tuple $(\mathcal{C}, I, \otimes, \alpha, \lambda, \rho)$ consisting of a category $\mathcal{C}$ of which I is an object, a bifunctor $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$, and natural isomorphisms α, λ, ρ with components

$$\alpha_{A,B,C} : A \otimes (B \otimes C) \cong (A \otimes B) \otimes C \qquad \lambda_A : I \otimes A \cong A \qquad \rho_A : A \otimes I \cong A$$

such that the diagrams

$$\begin{array}{ccccc} A \otimes (B \otimes (C \otimes D)) & \xrightarrow{\alpha} & (A \otimes B) \otimes (C \otimes D) & \xrightarrow{\alpha} & ((A \otimes B) \otimes C) \otimes D \\ \downarrow{\scriptstyle A \otimes \alpha} & & & & \uparrow{\scriptstyle \alpha \otimes D} \\ A \otimes ((B \otimes C) \otimes D) & & \xrightarrow{\qquad\qquad\alpha\qquad\qquad} & & (A \otimes (B \otimes C)) \otimes D \end{array}$$

and

$$\begin{array}{ccc} A \otimes (I \otimes B) & \xrightarrow{\alpha} & (A \otimes I) \otimes B \\ & {\scriptstyle A \otimes \lambda} \searrow \quad \swarrow {\scriptstyle \rho \otimes B} & \\ & A \otimes B & \end{array}$$

commute, and moreover, $\lambda_I = \rho_I$. The monoidal category $\mathcal{C}$ is *symmetric* iff it is equipped with a natural isomorphism γ with components

$$\gamma_{A,B} : A \otimes B \cong B \otimes A$$

such that the diagram

$$\begin{array}{ccccc} A \otimes (B \otimes C) & \xrightarrow{\alpha} & (A \otimes B) \otimes C & \xrightarrow{\gamma} & C \otimes (A \otimes B) \\ \downarrow{\scriptstyle A \otimes \gamma} & & & & \downarrow{\scriptstyle \alpha} \\ A \otimes (C \otimes B) & \xrightarrow{\alpha} & (A \otimes C) \otimes B & \xrightarrow{\gamma \otimes B} & (C \otimes A) \otimes B \end{array}$$

commutes, and moreover, $\gamma_{A,B}; \gamma_{B,A} = id$ and $\rho_A = \gamma_{A,I}; \lambda_A$.

Definition 2.1.2 A *monoidal functor* from a monoidal category $(\mathcal{C}, I, \otimes, \alpha, \lambda, \rho)$ to a monoidal category $(\mathcal{C}', I', \otimes', \alpha', \lambda', \rho')$ is a triple (F, m_I, m) consisting of a functor $F : \mathcal{C} \to \mathcal{C}'$ together with a map

$$I' \xrightarrow{m_I} F(I)$$

and a natural transformation m with components

$$FA \otimes' FB \xrightarrow{m_{A,B}} F(A \otimes B)$$

such that the diagrams

$$\begin{array}{ccccc} FA \otimes' (FB \otimes' FC) & \xrightarrow{id \otimes' m} & FA \otimes' F(B \otimes C) & \xrightarrow{m} & F(A \otimes (B \otimes C)) \\ \downarrow{\scriptstyle \alpha'} & & & & \downarrow{\scriptstyle F\alpha} \\ (FA \otimes' FB) \otimes' FC) & \xrightarrow{m \otimes' id} & F(A \otimes B) \otimes' FC & \xrightarrow{m} & F((A \otimes B) \otimes C) \end{array}$$

and

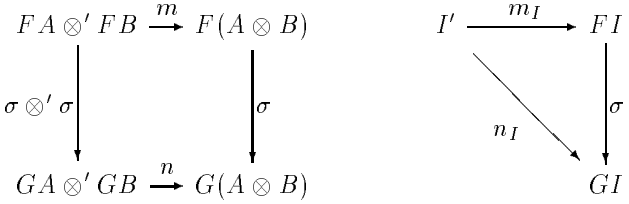
$$\begin{array}{ccc} I' \otimes' FA & \xrightarrow{\lambda'} & FA \\ \downarrow{\scriptstyle m_I \otimes' id} & & \uparrow{\scriptstyle F\lambda} \\ FI \otimes' FA & \xrightarrow{m} & F(I \otimes A) \end{array} \qquad \begin{array}{ccc} FA \otimes' I' & \xrightarrow{\rho'} & FA \\ \downarrow{\scriptstyle id \otimes' m_I} & & \uparrow{\scriptstyle F\rho} \\ FA \otimes' FI & \xrightarrow{m} & F(A \otimes I) \end{array}$$

commute (the map m_I is simply the nullary version of the natural transformation m). The monoidal functor F is *symmetric* iff the monoidal categories $\mathcal{C}$ and $\mathcal{C}'$ are symmetric and the diagram

$$\begin{array}{ccc} FA \otimes' FB & \xrightarrow{\gamma'} & FB \otimes' FA \\ \downarrow{\scriptstyle m} & & \downarrow{\scriptstyle m} \\ F(A \otimes B) & \xrightarrow{F(\gamma)} & F(B \otimes A) \end{array}$$

commutes. The (symmetric) monoidal functor F *preserves* the (symmetric) monoidal structure iff m_I and m are isomorphisms.

Definition 2.1.3 A *monoidal natural transformation* from a monoidal functor (F, m_I, m) to a monoidal functor (G, n_I, n) is a natural transformation $\sigma : F \to G$ such that the diagrams

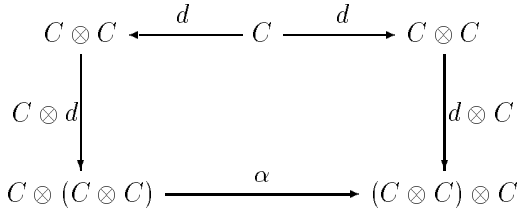


commute.

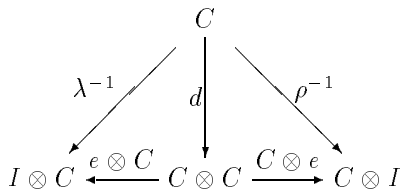
Definition 2.1.4 The (symmetric) monoidal category $\mathcal{C}$ is *closed* when each functor $(\Leftrightarrow) \otimes A$ has a specified right adjoint $A \multimap (\Leftrightarrow)$.

Definition 2.1.5 A *comonoid* in a monoidal category $(\mathcal{C}, I, \otimes, \alpha, \lambda, \rho)$ is a triple (C, e, d) consisting of an object C together with maps

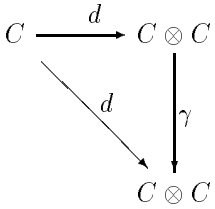
such that the diagrams



and

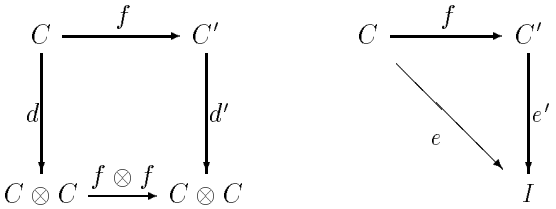


commute. The comonoid C is *commutative* iff the monoidal category $\mathcal{C}$ is symmetric and the diagram



commutes.

Definition 2.1.6 A *comonoid map* from a comonoid (C, e, d) to a comonoid (C', e', d') is a map $f : C \to C'$ such that the diagrams



commute.

Note that one obtains a category of comonoids in a monoidal category.

2.2 The Model for Intuitionistic Linear Logic

We will use the notion of a categorical model for Intuitionistic Linear Logic as it is defined in [BBdPH92b, BBdPH92a]; we introduce the model below and explain its ramifications afterwards. This categorical model was introduced to repair a deficiency of the model for Intuitionistic Linear Logic given in [See89], namely that the induced interpretation of proofs is not necessarily preserved by normalisation. Later on, the ramifications of the work presented in [BBdPH92b] were detailed in [Bie94].

Definition 2.2.1 A *linear category* is a symmetric monoidal closed category $(\mathcal{C}, I, \otimes, \multimap)$ with

- a symmetric monoidal comonad $(!, \varepsilon, \delta, m_I, m)$,
- monoidal natural transformations e and d with components

$$!A \xrightarrow{e_A} I \qquad\qquad !A \xrightarrow{d_A} !A \otimes !A$$

  such that for each object A

  - the maps e_A and d_A are maps of coalgebras,
  - the triple $(!A, e_A, d_A)$ is a commutative comonoid,
  - the map δ_A is a map of commutative comonoids[1].

The assumption that the comonad $(!, \varepsilon, \delta)$ is symmetric monoidal means that $!$ is a symmetric monoidal functor and ε and δ are monoidal natural transformations. When assuming the natural transformations e and d to be monoidal, we are assuming the functors I and $!(\Leftrightarrow)\otimes!(\Leftrightarrow)$ to have the obvious monoidal structure induced by the monoidal structure on $!$. Hence, the assumption that the natural transformation e is monoidal amounts to commutativity of the diagrams

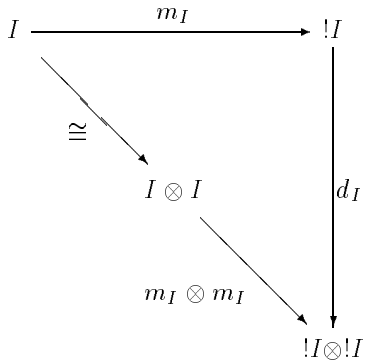
$$\begin{array}{ccc} !A \otimes !B & \xrightarrow{m_{A,B}} & !(A \otimes B) \\ {\scriptstyle e_A \otimes e_B}\downarrow & & \downarrow{\scriptstyle e_{A\otimes B}} \\ I \otimes I & \xrightarrow{\cong} & I \end{array} \qquad\qquad \begin{array}{ccc} I & \xrightarrow{m_I} & !I \\ & {\scriptstyle I}\searrow & \downarrow{\scriptstyle e_I} \\ & & I \end{array}$$

The assumption that the natural transformation d is monoidal amounts to commutativity of the diagrams

$$\begin{array}{ccccc} !A \otimes !B & & \xrightarrow{\qquad m_{A,B} \qquad} & & !(A \otimes B) \\ {\scriptstyle d_A \otimes d_B}\downarrow & & & & \downarrow{\scriptstyle d_{A\otimes B}} \\ !A \otimes !A \otimes !B \otimes !B & \xrightarrow{id \otimes \cong \otimes id} & !A \otimes !B \otimes !A \otimes !B & \xrightarrow{m_{A,B} \otimes m_{A,B}} & !(A \otimes B) \otimes !(A \otimes B) \end{array}$$

[1] This is a simplification of the original clause that maps of free coalgebras are maps of commutative comonoids.

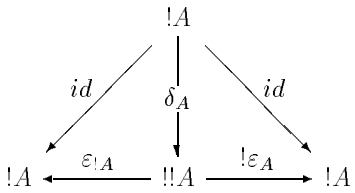
and



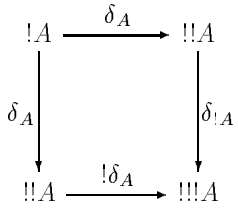
It can be shown that (I, m_I) and $(!A \otimes !A, (\delta_A \otimes \delta_A); m_{!A,!A})$ are coalgebras, so the assumption that e_A is a map of coalgebras amounts to e_A being a map from $(!A, \delta_A)$ to (I, m_I), and the assumption that d_A is a map of coalgebras amounts to d_A being a map from $(!A, \delta_A)$ to $(!A \otimes !A, (\delta_A \otimes \delta_A); m_{!A,!A})$. The notion of a coalgebra is defined below.

Let $\mathcal{C}$ be a category equipped with a comonad $(!, \varepsilon, \delta)$, that is, a functor $! : \mathcal{C} \to \mathcal{C}$ together with natural transformations ε and δ with components

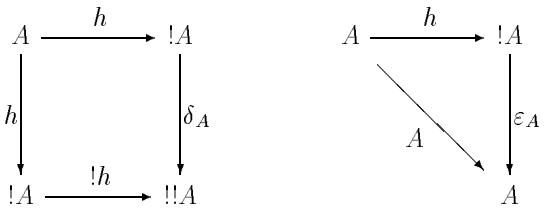
such that the diagrams



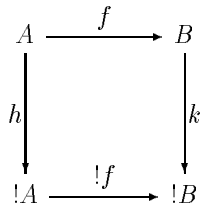
and



commute. Then $\mathcal{C}^!$ denotes the induced category of coalgebras and $\mathcal{C}_!$ denotes the induced Kleisli category. Recall that the category of coalgebras has as objects coalgebras, that is, pairs (A, h) consisting of an object A and a map $h : A \to !A$ such that the diagrams



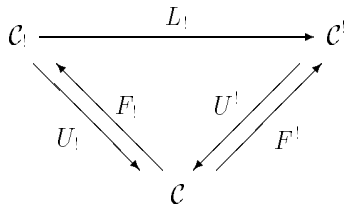
commute, and moreover, a map between the coalgebras (A, h) and (B, k) is a map $f : A \to B$ such that the diagram



commutes. Composition as well as the identity are inherited from $\mathcal{C}$. The Kleisli category has the same objects as $\mathcal{C}$, and a map between the objects A and B, considered as objects of the Kleisli category, is a map $f : !A \to B$ in $\mathcal{C}$. The map $\varepsilon_A : !A \to A$ is the identity on A in the Kleisli category, and given two maps $f : !A \to B$ and $g : !B \to C$ their composition in the Kleisli category is equal to the composition

$$!A \xrightarrow{\gamma(g)} !B \xrightarrow{f} C$$

where $\gamma(g) = \delta_A ; !g$. The operation γ is called the Kleisli operator. We have the following functors



The functor $U^!$ simply forgets the coalgebra structure, while the functor $F^!$ takes an object B to the coalgebra $(!B, \delta)$ and a map $f : B \to C$ to the map of coalgebras $!f : (!B, \delta) \to (!C, \delta)$. We have an adjunction $U^! \dashv F^!$ given by the natural bijection between maps

$$\gamma_{(D,h),B} : \mathcal{C}(U^!(D, h), B) \cong \mathcal{C}^!((D, h), F^!B)$$

where $\gamma(g) = h ; !g : (D, h) \to (!B, \delta)$ and $\gamma^{-1}(f) = f ; \varepsilon_B : D \to B$. Note that if in the definition of the function $\gamma_{(D,h),B}$ the coalgebra (D, h) is equal to $(!A, \delta)$, then γ coincides with the Kleisli operator given above. The functor $U_!$ takes an object B to $!B$ and a map $f : !B \to C$ to the map $\gamma(f) : !B \to !C$, while the functor $F_!$ is the identity on objects and takes a map $f : B \to C$ to the map $\varepsilon_B ; f : !B \to C$. We have an adjunction $U_! \dashv F_!$ induced by the observation that the sets $\mathcal{C}(U_!A, B)$ and $\mathcal{C}_!(A, F_!B)$ are identical. The functor $L_!$, called the comparison functor, takes an object B to the coalgebra $(!B, \delta)$ and a map $f : !B \to C$ to the map of coalgebras $\gamma(f) : (!B, \delta) \to (!C, \delta)$. Note that $F_! ; L_! = F^!$ and $L_! ; U^! = U_!$. The category of free coalgebras is the full subcategory of $\mathcal{C}^!$ whose objects are free coalgebras where a free coalgebra is a coalgebra which is equal to $(!B, \delta)$ for some object B. The category of free coalgebras is equivalent to the Kleisli category; it is straightforward to check that the comparison functor $L_!$ from $\mathcal{C}_!$ to $\mathcal{C}^!$ is an equivalence of categories when considered as a functor from $\mathcal{C}_!$ to the category of free coalgebras.

2.3 The Category of Coalgebras

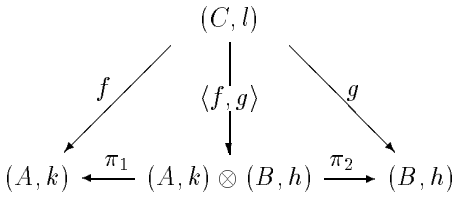
In [Bie94] it is shown that a symmetric monoidal comonad $(!, \varepsilon, \delta, m_I, m)$ on a symmetric monoidal category $(\mathcal{C}, I, \otimes)$ induces a symmetric monoidal structure on $\mathcal{C}^!$: The unit I of the tensor product is given by the coalgebra (I, m_I), and given two coalgebras (A, k) and (B, h), their tensor product $(A, k) \otimes (B, h)$ is given by the coalgebra $(A \otimes B, (k \otimes h) ; m_{A,B})$.

If, moreover, $\mathcal{C}$ is a linear category (it would actually not matter if we left out the closed structure), then the symmetric monoidal structure on $\mathcal{C}^!$ is a finite product structure, that is, I is a terminal object, and $(A, k) \otimes (B, h)$ is a binary product of (A, k) and (B, h) when equipped

with projections π_1 and π_2 given by the maps $(A \otimes (h; e_B)); \cong$ and $((k; e_A) \otimes B); \cong$, respectively. Given a coalgebra (A, k), a unique map of coalgebras

$$(A, k) \xrightarrow{\langle\rangle} I$$

is given by the map $k; e_A$, and given maps of coalgebras $f : (C, l) \to (A, k)$ and $g : (C, l) \to (B, h)$, a unique map of coalgebras $\langle f, g\rangle$ making the diagram



commute is given by the map $l; d_C; (e_C \otimes e_C); (f \otimes g)$. Note that d_A is equal to the diagonal map

$$(!A, \delta) \xrightarrow{\Delta} (!A, \delta) \otimes (!A, \delta)$$

Later we will need a generalisation of the natural transformation d_A: We define $D_{A_1,\ldots,A_n}$ to be the composition

$$!A_1 \otimes \ldots \otimes !A_n \xrightarrow{d_{A_1} \otimes \ldots \otimes d_{A_n}} !A_1 \otimes !A_1 \otimes \ldots \otimes !A_n \otimes !A_n \cong !A_1 \otimes \ldots \otimes !A_n \otimes !A_1 \otimes \ldots \otimes !A_n$$

It can be shown by some equational manipulation that $D_{A_1,\ldots,A_n}$ is equal to the diagonal map

$$(!A_1, \delta) \otimes \ldots \otimes (!A_n, \delta) \xrightarrow{\Delta} (!A_1, \delta) \otimes \ldots \otimes (!A_n, \delta) \otimes (!A_1, \delta) \otimes \ldots \otimes (!A_n, \delta)$$

Note that if in the definition of $\gamma_{(D,h),B}$ in the previous section the coalgebra (D, h) is equal to $(!A_1, \delta) \otimes \ldots \otimes (!A_n, \delta)$, then γ coincides with the generalised Kleisli operator of [BBdPH92b] which takes a map

$$f : !A_1 \otimes \ldots \otimes !A_n \to B$$

to the map $\gamma(f)$ defined as the composition

$$!A_1 \otimes \ldots \otimes !A_n \xrightarrow{\delta_{A_1} \otimes \ldots \otimes \delta_{A_n}} !!A_1 \otimes \ldots \otimes !!A_n \xrightarrow{m_{A_1,\ldots,A_n}} !(!A_1 \otimes \ldots \otimes !A_n) \xrightarrow{!f} !B$$

It is remarkable that the definition of the generalised Kleisli operator does not rely on the presence of finite products; in [See89] another notion of a categorical model for Intuitionistic Linear Logic is given where the induced generalised Kleisli operator does not have this property.

We will now assume that the linear category $\mathcal{C}$ does have finite products (it would not matter if we left out the closed structure). The functor $F^! : \mathcal{C} \to \mathcal{C}^!$ preserves finite products because it has a left adjoint, so a terminal object 1 in $\mathcal{C}$ is sent into a terminal object $(!1, \delta)$ in $\mathcal{C}^!$, and a binary product diagram

$$A \xleftarrow{\pi_1} A \times B \xrightarrow{\pi_2} B$$

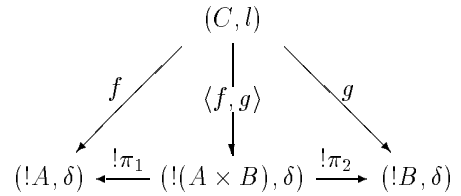
living in $\mathcal{C}$ is sent into a binary product diagram

$$(!A, \delta) \xleftarrow{!\pi_1} (!(A \times B), \delta) \xrightarrow{!\pi_2} (!B, \delta)$$

living in $\mathcal{C}^!$. For later use we will make explicit the constructions involved in this observation. Given a coalgebra (A, k), a unique map of coalgebras

$$(A, k) \xrightarrow{\langle\rangle} (!1, \delta)$$

is given by the map $\gamma(\langle\rangle)$, and given maps of coalgebras $f : (C, l) \to (!A, \delta)$ and $g : (C, l) \to (!B, \delta)$, a unique map of coalgebras $\langle f, g\rangle$ making the diagram



commute is given by the map $\gamma(\langle\gamma^{-1}(f), \gamma^{-1}(g)\rangle)$. We have the finite product structure $(I, \otimes)$ on $\mathcal{C}^!$ which entails that we have an isomorphism

$$n_1 : I \cong (!1, \delta)$$

and an isomorphism

$$n_{A,B} : (!A, \delta) \otimes (!B, \delta) \cong (!(A \times B), \delta)$$

which is natural in $(!A, \delta)$ and $(!B, \delta)$. It can be shown that the isomorphism $n_1 : I \cong !1$ and the natural isomorphism $n_{A,B} : !A \otimes !B \cong !(A \times B)$, where the coalgebras are left out, make $!$ a symmetric monoidal functor from $(\mathcal{C}, 1, \times)$ to $(\mathcal{C}, I, \otimes)$. Given these isomorphisms, we can actually define a model for Intuitionistic Linear Logic as described in [See89]: Calculations show that the way the isomorphisms are defined force $!$ to take the commutative comonoid structure with respect to the finite products to the commutative comonoid structure with respect to the symmetric monoidal structure, that is, the map $!\langle\rangle : !A \to !1$ is equal to the composition

$$!A \xrightarrow{e_A} I \cong !1$$

and the map $!\Delta : !A \to !(A \times A)$ is equal to the composition

$$!A \xrightarrow{d_A} !A \otimes !A \cong !(A \times A)$$

A detailed comparison between the model for Intuitionistic Linear Logic given in [BBdPH92b] and the one from [See89] can be found in [Bie94].

The terminal object in $\mathcal{C}^!$ induced by the terminal object in $\mathcal{C}$ is also a terminal object in the category of free coalgebras, and similarly, the binary product diagram in $\mathcal{C}^!$ induced by the binary product diagram in $\mathcal{C}$, is also a binary product diagram in the category of free coalgebras. This induces a finite product structure on the category of free coalgebras[2].

Now, assume that a symmetric monoidal closed category $(\mathcal{C}, I, \otimes, \multimap)$ is equipped with a symmetric monoidal comonad $(!, \varepsilon, \delta, m_I, m)$. Then $\mathcal{C}^!$ has a symmetric monoidal structure, as mentioned above, and moreover, it is shown in [Bie94] that for every object B the free coalgebra $(!B, \delta)$ is exponentiable with respect to the monoidal structure on $\mathcal{C}^!$. The internal-hom object of a coalgebra (A, h) and the free coalgebra $(!B, \delta)$ is given by the free coalgebra $(!(A \multimap B), \delta)$, and an appropriate bijection between maps is given by the composition

$$\begin{array}{rcll}
\mathcal{C}^!((C,k) \otimes (A,h), (!B,\delta)) & \cong & \mathcal{C}(U^!((C,k) \otimes (A,h)), B) & \text{because } U^! \dashv F^! \\
& = & \mathcal{C}(U^!(C,k) \otimes A, B) & \\
& \cong & \mathcal{C}(U^!(C,k), A \multimap B) & \text{because } (\Leftrightarrow) \otimes A \dashv A \multimap (\Leftrightarrow) \\
& \cong & \mathcal{C}^!((C,k), (!(A \multimap B), \delta)) & \text{because } U^! \dashv F^!
\end{array}$$

which is natural in (C, k). If, moreover, $\mathcal{C}$ is a linear category with finite products, then the category of free coalgebras has finite products as given above, and we have the following composition of bijections

$$\begin{array}{rcll}
\mathcal{C}^!((!(C \times A), \delta), (!B, \delta)) & \cong & \mathcal{C}^!((!C, \delta) \otimes (!A, \delta), (!B, \delta)) & \text{by composition with } n \\
& \cong & \mathcal{C}^!((!C, \delta), (!(!A \multimap B), \delta)) & \text{as } (!B, \delta) \text{ is exponentiable}
\end{array}$$

[2] Note that the binary product structure hinges on the Axiom of Choice; to define the binary product of a pair of free coalgebras (A', k) and (B', h) we need objects A and B such that $(A', k) = (!A, \delta)$ and $(B', h) = (!B, \delta)$.

which is natural in $(!C, \delta)$. This induces a cartesian closed structure on the category of free coalgebras[3].

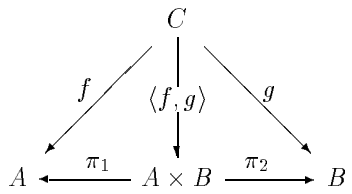
2.4 The Kleisli Category

In the previous section we gave a cartesian closed structure on the category of free coalgebras induced by a linear category with finite products. In this section we will essentially restate this construction in the Kleisli category which is equivalent to the category of free coalgebras. The observation that the Kleisli category is cartesian closed goes back to [See89]. It turns out that the cartesian closed structure on the Kleisli category can be stated in a simpler way[4] enabling a more succinct statement of soundness of the Girard Translation, Theorem 5.2.2.

Now, assume that we are dealing with a linear category $\mathcal{C}$ with finite products. The finite product structure on $\mathcal{C}$ induces a finite product structure on the Kleisli category as follows: The terminal object in the Kleisli category is taken to be 1, and the binary product of the objects A and B in the Kleisli category is taken to be $A \times B$ when equipped with projections π_1 and π_2 given by the maps $\varepsilon; \pi_1 : !(A \times B) \to A$ and $\varepsilon; \pi_2 : !(A \times B) \to B$, respectively. Given an object A, a unique map in the Kleisli category

$$A \xrightarrow{\langle\rangle} 1$$

is given by the map $\langle\rangle : !A \to 1$, and given maps $f : !C \to A$ and $g : !C \to B$, a unique map $\langle f, g\rangle$ making the diagram in the Kleisli category



commute is given by the map $\langle f, g\rangle : !C \to A \times B$. The internal-hom object $A \Rightarrow B$ of the objects A and B in the Kleisli category is taken to be $!A \multimap B$, and an appropriate bijection between maps is given by the composition

$$\begin{array}{rcll} \mathcal{C}_!(C \times A, B) & = & \mathcal{C}(!(C \times A), B) & \\ & \cong & \mathcal{C}(!C \otimes !A, B) & \text{by composition with } n \\ & \cong & \mathcal{C}(!C, !A \multimap B) & \text{because } (\Leftrightarrow) \otimes !A \dashv !A \multimap (\Leftrightarrow) \\ & = & \mathcal{C}_!(C, !A \multimap B) & \\ & = & \mathcal{C}_!(C, A \Rightarrow B) & \end{array}$$

which is natural in C. The definition of the internal-hom object corresponds to Girard's observation which gave rise to Linear Logic; namely the observation that the functor which forgets the linearity of linear stable functions between coherence spaces has a left adjoint.

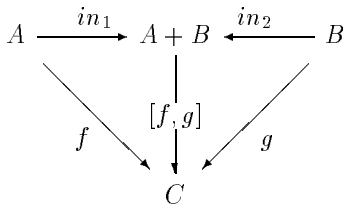
We will now consider finite sums. If $\mathcal{C}$ is a category with a comonad $(!, \varepsilon, \delta)$, then an initial object 0 in $\mathcal{C}$ induces a weak initial object in the Kleisli category and binary sums $+$ in $\mathcal{C}$ induce weak binary sums in the Kleisli category. This is essentially a restatement of an analogous construction for the category of free coalgebras pointed out in [Bie94]. The weak initial object 0 in the Kleisli category is taken to be 0, and the binary weak sum $A + B$ of the objects A and B in the Kleisli category is taken to be $!A + !B$ when equipped with injections in_1 and in_2 given by the maps $in_1 : !A \to !A + !B$ and $in_2 : !B \to !A + !B$, respectively. Given an object A, a map in the Kleisli category

$$0 \xrightarrow{[]} A$$

[3] Note that the closed structure hinges on the Axiom of Choice; to define the internal-hom object of a coalgebra (A, h) and a free coalgebra (B', h) we need an object B such that $(B', h) = (!B, \delta)$.

[4] Avoiding the Axiom of Choice.

is given by the map $\varepsilon; [] : !0 \to A$, and given maps $f : !A \to C$ and $g : !B \to C$, a map $[f, g]$ making the diagram in the Kleisli category



commute is given by the map $\varepsilon; [f, g] : !(!A + !B) \to C$. One can actually show more than that.

Definition 2.4.1 Let $\mathcal{C}$ be a category equipped with a comonad $(!, \varepsilon, \delta)$. A map in the Kleisli category is called *linear* iff it is in the image of $F_!$.

Proposition 2.4.2 Let $\mathcal{C}$ be a category with a comonad. If $\mathcal{C}$ has an initial object 0, then the family of maps in the Kleisli category

$$[] : 0 \to C$$

induced by the weak initial object is natural in C with respect to linear maps. If $\mathcal{C}$ has binary sums $+$, then the family of operations on maps in the Kleisli category

$$[\Leftrightarrow, +] : \mathcal{C}_!(A, C) \times \mathcal{C}_!(B, C) \to \mathcal{C}_!(A + B, C)$$

induced by the weak binary sums is natural in C with respect to linear maps.

Proof: Straightforward equational manipulation. □

Proposition 2.4.3 Let $\mathcal{C}$ be a linear category with finite products. Given a map $f : !A \to B$, the map in the Kleisli category

$$B \Rightarrow C \xrightarrow{f \Rightarrow C} A \Rightarrow C$$

is equal to $F_!(\gamma(f) \multimap C)$.

Proof: Recall that the Kleisli category is cartesian closed. The result follows from straightforward equational manipulation. □

The last two results may seem ad hoc, but it turns out that they are sufficient to give a categorical interpretation of the sums of the λ-calculus in the Kleisli category.

2.5 Categories of Cpos

The category of cpos and continuous functions, **cpo**, and the category of cpos and strict continuous functions, $\mathbf{cpo}_{st}$, are well known from the literature, see for example the textbooks [Win93, Gun92]. The category $\mathbf{cpo}_{st}$ is an example of a linear category according to [Bie94]. It is actually a model for Intuitionistic Relevant Logic in the sense of [Jac94], that is, there is a natural transformation d with components $d_A : A \to A \otimes A$ making appropriate diagrams commute. In this section we recall the constructs for the sake of completeness. We first define the categories **cpo** and $\mathbf{cpo}_{st}$ formally. Then the constructions are given which make $\mathbf{cpo}_{st}$ a linear category with finite products and sums. As a spin-off we obtain a cartesian closed structure on **cpo**.

Definition 2.5.1 A *cpo* is a poset $(D, \sqsubseteq)$ with a least element $\perp$ such that every directed subset X has a least upper bound $\sqcup X$. A monotone function $f : D \to E$ between cpos is called *continuous* iff $f(\sqcup X) = \sqcup f(X)$ for any directed subset X. A monotone function $f : D \to E$ between cpos is called *strict* iff $f(\perp) = \perp$.

The obvious ordering on continuous functions between cpos is the *extensional* ordering defined as

$$f \sqsubseteq g \quad \text{iff} \quad \forall x \in A.\, f(x) \sqsubseteq g(x)$$

for any continuous functions $f, g : A \to B$. First, we will give a symmetric monoidal structure on the category $\mathbf{cpo}_{st}$.

Definition 2.5.2 We define I to be the two-element set $\{\bot, \top\}$ with $\bot$ as bottom element. A bifunctor $\otimes$ is defined as follows: Given cpos D and E we define $D \otimes E$ to be the set

$$\{(d, e) \in D \times E \mid d = \bot \Leftrightarrow e = \bot\}$$

equipped with the coordinate wise ordering. Given strict continuous functions $f : D \to D'$ and $g : E \to E'$, we define the strict continuous function $f \otimes g : D \otimes E \to D' \otimes E'$ as

$$(f \otimes g)(d, e) = \begin{cases} (f(d), g(e)) & \text{if } f(d) \neq \bot \ \wedge g(e) \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

It is now straightforward to check that I together with the bifunctor $\otimes$ constitute a symmetric monoidal structure. It is custom to call I the *Sierpinsky space* and $D \otimes E$ is called the *smash product* of D and E. We will now show that every functor $(\Leftrightarrow) \otimes D$ has a right adjoint $D \multimap (\Leftrightarrow)$.

Definition 2.5.3 Given cpos D and E we define $D \multimap E$ to be the set of strict continuous functions from D to E equipped with the extensional order.

Definition 2.5.4 Given cpos E and D we define a strict continuous function

$$(D \multimap E) \otimes D \xrightarrow{eval_{E,D}} E$$

as

$$eval(f, e) = f(e).$$

It is straightforward to extend the function $D \multimap (\Leftrightarrow)$ to a functor which is right adjoint to $(\Leftrightarrow) \otimes D$ such that $eval$ is counit: Given a strict continuous function $f : C \otimes D \to E$ we define a strict continuous function $\lambda(f) : C \to (D \multimap E)$ as

$$\lambda(f)(c)(d) = \begin{cases} f(c, d) & \text{if } c \neq \bot \ \wedge d \neq \bot \\ \bot & \text{otherwise} \end{cases}$$

It is then the case that $\lambda(f)$ is a unique strict continuous function with the property that the diagram

$$\begin{array}{ccc} C \otimes D & & \\ \downarrow f & \searrow \lambda(f) \otimes D & \\ E & \xleftarrow{eval} & (D \multimap E) \otimes D \end{array}$$

commutes. We will now give a comonad on the category $\mathbf{cpo}_{st}$ by giving a left adjoint to the forgetful functor

$$U : \mathbf{cpo}_{st} \hookrightarrow \mathbf{cpo}.$$

Definition 2.5.5 Given a cpo D we define $D_\bot$ to be the set $(D \times \{*\}) \cup \{\bot\}$, where $\bot$ is different from all the elements of $D \times \{*\}$, equipped with the ordering $\sqsubseteq$ given by stipulating that $(d, *) \sqsubseteq (d', *)$ whenever $d \sqsubseteq d'$ and $\bot \sqsubseteq x$ for any $x \in D_\bot$.

It is custom to call $D_\bot$ the *lift* of D.

Definition 2.5.6 Given a cpo D we define a continuous function

$$D \xrightarrow{\eta_D} D_\perp$$

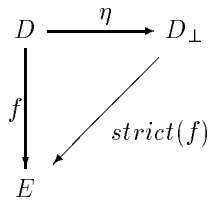
as

$$\eta(d) = (d, *).$$

It is straightforward to extend the function $(\Leftrightarrow)_\perp$ to a functor from **cpo** to $\mathbf{cpo}_{st}$ which is left adjoint to U such that η is unit: Given a continuous function $f : D \to E$, we define a strict continuous function $strict(f) : D_\perp \to E$ by

$$strict(f)(x) = \begin{cases} f(d) & \text{if } x = (d, *) \\ \perp & \text{otherwise} \end{cases}$$

It is then the case that $strict(f)$ is a unique strict continuous function such that the diagram



commutes. Note that the diagram lives in the category **cpo**. Let $f : D \to E$ be a continuous function, the strict continuous function $f_\perp : D_\perp \to E_\perp$ will then be given by

$$f_\perp(x) = \begin{cases} f(d) & \text{if } x = (d, *) \\ \perp & \text{otherwise} \end{cases}$$

The adjunction induces a comonad on $\mathbf{cpo}_{st}$ consisting of the functor $(\Leftrightarrow)_\perp$ restricted to an endofunctor on $\mathbf{cpo}_{st}$ together with a natural transformation ε where a component $\varepsilon_D : D_\perp \to D$ is equal to $strict(id_D)$ and a natural transformation δ where a component $\delta_D : D_\perp \to D_{\perp\perp}$ is equal to $\eta_{D\perp}$. We want the comonad $((\Leftrightarrow)_\perp, \varepsilon, \delta)$ to be symmetric monoidal which motivates the following definition:

Definition 2.5.7 We define a strict continuous function $m_I : I \to I_\perp$ as

$$m_I(x) = \begin{cases} (\top, *) & \text{if } x = \top \\ \perp & \text{otherwise} \end{cases}$$

and we define a natural transformation m with components $m_{D,E} : D_\perp \otimes E_\perp \to (D \otimes E)_\perp$ by defining for each pair of cpos D and E a strict continuous function $m_{D,E}$ as

$$m(x) = \begin{cases} ((d, e), *) & \text{if } x = ((d, *), (e, *)) \text{ such that } d \neq \perp \ \wedge \ e \neq \perp \\ (\perp, *) & \text{if } x = ((d, *), (e, *)) \text{ such that } d = \perp \ \vee \ e = \perp \\ \perp & \text{otherwise} \end{cases}$$

It is now straightforward to check that the function m_I together with the natural transformation m gives symmetric monoidal structure to the functor $(\Leftrightarrow)_\perp$ such that ε and δ are monoidal natural transformations. Furthermore:

Definition 2.5.8 We define a natural transformation e with components $e_D : D_\perp \to I$ by defining for each cpo D a strict continuous function e_D as

$$e(x) = \begin{cases} \top & \text{if } x = (d, *) \\ \perp & \text{otherwise} \end{cases}$$

and we define a natural transformation d with components $d_D : D_\perp \to D_\perp \otimes D_\perp$ by defining for each cpo D a strict continuous function d_D as

$$d(x) = \begin{cases} ((d, *), (d, *)) & \text{if } x = (d, *) \\ \perp & \text{otherwise} \end{cases}$$

It is now straightforward to check that the natural transformations e and d are monoidal such that the three additional conditions on being a linear category, Definition 2.2.1, are satisfied. It is easy to check that $\mathbf{cpo}_{st}$ has finite products and sums; the former is the usual construction for posets and the latter is the "coalesced sum". The Kleisli category corresponding to the comonad $(\Leftrightarrow)_\perp$ is isomorphic to $\mathbf{cpo}$, so the results of Section 2.4 imply that the finite products of $\mathbf{cpo}_{st}$ make $\mathbf{cpo}$ cartesian closed, and the finite sums of $\mathbf{cpo}_{st}$ induce a weak finite sum structure on $\mathbf{cpo}$; this is the "separated sum".

2.6 Categories of dI-Domains

The category of dI-domains and continuous stable functions, $\mathbf{dI}$, is fairly well known from the literature. For example it is described in the article [Ber78] and the textbooks [Win87, Gun92]. The category of dI-domains and linear stable functions, $\mathbf{dI}_{lin}$, does not occur so often. It was originally observed in [Win87] that the functor that forgets the linearity of linear stable functions between dI-domains has a left adjoint. This is analogous to Girard's observation which gave rise to Linear Logic; namely that the functor that forgets the linearity of linear stable functions between coherence spaces has a left adjoint. So the category $\mathbf{dI}_{lin}$ is an obvious candidate for a model for Intuitionistic Linear Logic. The category is described in some details in [Zha93]. The main goal of this section will be to show that $\mathbf{dI}_{lin}$ is a linear category in the sense of [BBdPH92b]. This is analogous to the account of the category of dI-domains and affine stable functions given in [Bra94a, Bra94b]; this category is a model for Intuitionistic Affine Logic. The category $\mathbf{dI}_{lin}$ is not degenerate in the sense of also being a model for Intuitionistic Relevant Logic or Intuitionistic Affine Logic. After defining the categories $\mathbf{dI}$ and $\mathbf{dI}_{lin}$ formally, we introduce the notion of *trace* and prove some results showing how this can be used to handle maps of the categories in an easy way. Then the constructions are given which make $\mathbf{dI}_{lin}$ a linear category with finite products and sums. This induces a cartesian closed structure on $\mathbf{dI}$.

Definition 2.6.1 Let $(D, \sqsubseteq)$ be a non-empty poset, and assume that every finitely compatible subset X has a least upper bound $\sqcup X$. A subset X is finitely compatible iff every finite subset of X has an upper bound. Note that this entails that D has a bottom element $\perp$, and moreover, it entails that every non-empty subset X has a greatest lower bound $\sqcap X$.

A *prime* element of D is an element d such that

$$d \sqsubseteq \sqcup X \;\Rightarrow\; \exists x \in X.\, d \sqsubseteq x$$

for any finitely compatible subset X. We will denote the set of prime elements of D by D_p. The poset D is called *prime algebraic* iff

$$\forall d \in D.\, d = \sqcup\{d' \in D_p \mid d' \sqsubseteq d\}.$$

A *finite* element of D is an element d such that

$$d \sqsubseteq \sqcup X \;\Rightarrow\; \exists x \in X.\, d \sqsubseteq x$$

for any directed subset X. We will denote the set of finite elements of D by D_o. The poset D is called *finitary* iff

$$\forall d \in D_o.\, |\{d' \in D \mid d' \sqsubseteq d\}| < \infty.$$

The poset D is a *dI-domain* iff it is prime algebraic and finitary. A monotone function $f : D \to E$ between dI-domains is called *stable* iff $f(\sqcap X) = \sqcap f(X)$ for any non-empty finitely compatible subset X. A monotone function $f : D \to E$ between dI-domains is called *continuous* iff $f(\sqcup X) = \sqcup f(X)$ for any directed subset X. A monotone function $f : D \to E$ between dI-domains is called *linear* iff $f(\sqcup X) = \sqcup f(X)$ for any finitely compatible subset X.

Proposition 2.6.2 If $f : D \to E$ is a continuous function between two dI-domains such that $e \sqsubseteq f(d)$ where d is an arbitrary element of D and e a finite element of E, then there exists a finite minimal $d' \sqsubseteq d$ with the property that $e \sqsubseteq f(d')$, and moreover, if f is stable, then d' is the least $d'' \sqsubseteq d$ such that $e \sqsubseteq f(d'')$.

Proof: Assume that $f : D \to E$ is a continuous function such that $e \sqsubseteq f(d)$ where d is arbitrary and e is finite. We then have

$$\begin{array}{rcl} f(d) & = & f(\sqcup\{d' \in D_p \mid d' \sqsubseteq d\}) \\ & = & f(\sqcup\{d' \in D_o \mid d' \sqsubseteq d\}) \\ & = & \sqcup\{f(d') \mid d' \in D_o \ \wedge \ d' \sqsubseteq d\} \end{array}$$

The assumption that D is prime algebraic gives us the first equation, the second comes from the fact that $D_p \subseteq D_o$, and the third comes from continuity of f, and the fact that $\sqcup\{d' \in D_o \mid d' \sqsubseteq d\}$ is directed. But e is finite and $\{f(d') \mid d' \in D_o \ \wedge \ d' \sqsubseteq d\}$ is directed, so

$$e \sqsubseteq \sqcup\{f(d') \mid d' \in D_o \ \wedge \ d' \sqsubseteq d\}$$

entails that there is at least one finite $d' \sqsubseteq d$ such that $e \sqsubseteq f(d')$. Now, we can pick a minimal finite $d' \sqsubseteq d$ with that property because D is finitary and because any element below a finite element is finite.

Moreover, assume that f is stable and $d'' \sqsubseteq d$ such that $e \sqsubseteq f(d'')$, then $e \sqsubseteq f(d') \sqcap f(d'')$ and $f(d') \sqcap f(d'') = f(d' \sqcap d'')$. But $d' \sqcap d'' \sqsubseteq d'$ and d' is minimal, so $d' = d' \sqcap d''$. Thus, $d' \sqsubseteq d''$. $\Box$

So if we are in the context of Proposition 2.6.2 where the continuous function f is stable, then $e \sqsubseteq f(d)$ implies that we can find a finite least $d' \sqsubseteq d$ such that $e \sqsubseteq f(d')$. This motivates the following definition:

Definition 2.6.3 If $f : D \to E$ is a continuous stable function then the *trace* of f, denoted $Tr(f)$, is defined as

$$\{(d, e) \in D_o \times E_p \mid e \sqsubseteq f(d) \ \wedge \ \forall d' \sqsubseteq d.\ (e \sqsubseteq f(d') \ \Rightarrow \ d = d')\}$$

In what follows $X\uparrow$ means that X has an upper bound. There is a close connection between functions and traces:

Theorem 2.6.4 Let $\{(d_i, e_i) \mid i \in I\} \subseteq D_o \times E_p$ for some indexing set I such that

- $\forall J \subseteq_{fin} I.\ \{d_j \mid j \in J\}\uparrow \Rightarrow \ \{e_j \mid j \in J\}\uparrow$
- $\forall i, j \in I.\ d_i\uparrow d_j \ \wedge \ e_i = e_j \ \Rightarrow \ d_i = d_j$
- $\forall i \in I.\ \forall e \in E_p.\ e \sqsubseteq e_i \ \Rightarrow \ \exists j \in I.\ (e_j = e \ \wedge \ d_j \sqsubseteq d_i)$

Then the function $f : D \to E$ defined as

$$f(d) = \sqcup\{e \mid \exists d' \sqsubseteq d.\ \exists i \in I.\ (d', e) = (d_i, e_i)\}$$

is a continuous stable function, and conversely, if $f : D \to E$ is a continuous stable function, then $Tr(f)$ has the mentioned three properties. Moreover, the operations are each others' inverses.

Proof: If $\{(d_i, e_i) \mid i \in I\}$ has the three mentioned properties, then it is straightforward to check that f is a continuous stable function.

Conversely, let $\{(d_i, e_i) \mid i \in I\} = Tr(f)$ for some continuous stable function f. The first property is obvious. Assume that $d_i \uparrow d_j$ and $e_i = e_j$. Then $f(d_i \sqcap d_j) = f(d_i) \sqcap f(d_j)$ and so $e_i \sqsubseteq f(d_i \sqcap d_j)$ and $e_j \sqsubseteq f(d_i \sqcap d_j)$, which entails that $d_i = d_i \sqcap d_j = d_j$. This proves the second property. Assume that $e \sqsubseteq e_i$ where $e \in D_p$. Then $e \sqsubseteq f(d_i)$, so we can find a finite minimal $d' \sqsubseteq d_i$ such that $e \sqsubseteq f(d')$, that is, $(d', e) \in Tr(f)$, according to Proposition 2.6.2. This proves the third property. It is straightforward to check that the operations are each others' inverses. $\Box$

From now on, we shall frequently identify a continuous stable function with its trace. Thus, continuous stable functions between dI-domains can be ordered by inclusion; this order can be shown to coincide with the *stable* order defined as

$$f \sqsubseteq g \quad \text{iff} \quad \forall x, y \in A.\ x \sqsubseteq y \ \Rightarrow \ f(x) = f(y) \sqcap g(x)$$

for any continuous stable functions $f, g : A \to B$. There is a nice way of seeing whether a function is linear or not by looking at its trace:

Proposition 2.6.5 A continuous stable function $f : D \to E$ is linear iff $\pi_0(f) \subseteq D_p$.

Proof: Assume that $\pi_0(f) \subseteq D_p$, and let X be a finitely compatible subset. We obviously have $\sqcup f(X) \sqsubseteq f(\sqcup X)$. Conversely, if $e \in E_p$ such that $e \sqsubseteq f(\sqcup X)$, then there exists a $d \in E_o$ such that $(d, e) \in f$ and $d \sqsubseteq \sqcup X$ cf. Theorem 2.6.4. But $d \in E_p$ cf. the assumption that $\pi_0(f) \subseteq D_p$. This entails that there exists an $x \in X$ such that $d \sqsubseteq x$, and thus, $e \sqsubseteq f(x) \sqsubseteq \sqcup f(X)$. We conclude that $f(\sqcup X) \sqsubseteq \sqcup f(X)$.

Conversely, assume that f is linear, and let $(d, e) \in f$. Then $e \sqsubseteq f(d)$. But

$$f(d) = f(\sqcup\{d' \in D_p \mid d' \sqsubseteq d\})$$

which is equal to $\sqcup\{f(d') \mid d' \in D_p \ \wedge \ d' \sqsubseteq d\}$ because f is assumed to be linear. This entails that there exists a $d' \in D_p$ such that $d' \sqsubseteq d$ and $e \sqsubseteq f(d')$. We conclude that $d = d'$, because d is minimal, and thus $d \in D_p$. □

The following result about composition of functions is useful:

Theorem 2.6.6 If $f : C \to D$ and $g : D \to E$ are continuous stable functions then

$$f;g = \{(c, e) \mid \exists (c_1, d_1), ..., (c_n, d_n) \in f.\ \{c_1, ..., c_n\}\uparrow \ \wedge\ c = \sqcup_{1 \le i \le n} c_i \ \wedge \ (\sqcup_{1 \le i \le n} d_i, e) \in g\}.$$

Proof: First, note that Theorem 2.6.4 entails that $\{d_1, ..., d_n\}\uparrow$. Now, assume that $(c, e) \in f;g$. Then $e \sqsubseteq g(f(c))$ entails that there exists a $d \sqsubseteq f(c)$ such that $(d, e) \in g$ cf. Proposition 2.6.2. Now, let $\{d_1, ..., d_n\} = \{d' \in D_p \mid d' \sqsubseteq d\}$. Then we have for each $i \in \{1, ..., n\}$ that $d_i \sqsubseteq f(c)$ entails that there exists a $c_i \sqsubseteq c$ such that $(c_i, d_i) \in f$ cf. Proposition 2.6.2. Thus, $\sqcup_{1 \le i \le n} c_i \sqsubseteq c$ and $\sqcup_{1 \le i \le n} d_i = d$, which entails that $e \sqsubseteq g(f(\sqcup_{1 \le i \le n} c_i))$ cf. Theorem 2.6.4. We conclude that $c = \sqcup_{1 \le i \le n} c_i$, because c is minimal. Conversely, assume that $(c_1, d_1), ..., (c_n, d_n) \in f$ such that $\{c_1, ..., c_n\}\uparrow$ and $(\sqcup_{1 \le i \le n} d_i, e) \in g$. We have $e \sqsubseteq g(f(\sqcup_{1 \le i \le n} c_i))$ cf. Theorem 2.6.4. Now, assume that we have a $c \sqsubseteq \sqcup_{1 \le i \le n} c_i$ such that $e \sqsubseteq g(f(c))$. This entails that there exists a $d \sqsubseteq f(c)$ such that $(d, e) \in g$ cf. Proposition 2.6.2. But $d \sqsubseteq f(c) \sqsubseteq f(\sqcup_{1 \le i \le n} c_i)$ and $\sqcup_{1 \le i \le n} d_i \sqsubseteq f(\sqcup_{1 \le i \le n} c_i)$, that is, $d \uparrow \sqcup_{1 \le i \le n} d_i$, which entails that $d = \sqcup_{1 \le i \le n} d_i$ cf. Theorem 2.6.4. Then we have for each $i \in \{1, ..., n\}$ that $d_i \sqsubseteq f(c)$ entails the existence of a $c'_i \sqsubseteq c$ such that $(c'_i, d_i) \in f$ cf. Proposition 2.6.2. But $c'_i \sqsubseteq c \sqsubseteq \sqcup_{1 \le i \le n} c_i$ and $c_i \sqsubseteq \sqcup_{1 \le i \le n} c_i$, that is, $c'_i \uparrow c_i$, which entails that $c'_i = c_i$ cf. Theorem 2.6.4. So $\sqcup_{1 \le i \le n} c'_i \sqsubseteq c \sqsubseteq \sqcup_{1 \le i \le n} c_i$ entails that $c = \sqcup_{1 \le i \le n} c_i$, and we conclude that $(\sqcup_{1 \le i \le n} c_i, e) \in f;g$. □

Also, the following result about composition of functions is useful:

Corollary 2.6.7 If $f : C \to D$ is a continuous stable function and $g : D \to E$ is a linear stable function then

$$f;g = \{(c, e) \mid \exists d \in D.\ (c, d) \in f \ \wedge \ (d, e) \in g\}$$

that is, the trace of the composition is equal to the traces composed as relations.

Proof: Assume that $(c_1, d_1), ..., (c_n, d_n) \in f$ such that $\{c_1, ..., c_n\}\uparrow$ and $(\sqcup_{1 \le i \le n} d_i, e) \in g$. The function g is linear, so $\sqcup_{1 \le i \le n} d_i \in D_p$ cf. Proposition 2.6.5, which entails that there exists a $q \in \{1, ..., n\}$ such that $\sqcup_{1 \le i \le n} d_i = d_q$. But then $c_i \sqsubseteq c_q$ for every $i \in \{1, ..., n\}$ cf. Theorem 2.6.4, and we conclude that $\sqcup_{1 \le i \le n} c_i = c_q$. □

Now, a small definition: Given $d \in X \subseteq D$ we define $\ulcorner d \urcorner = \{d' \in X \mid d' \sqsubseteq d\}$. We will first give a symmetric monoidal structure on the category $\mathbf{dI}_{lin}$.

Definition 2.6.8 We define I to be the Sierpinsky space. A bifunctor $\otimes$ is defined as follows: Given dI-domains D and E we define $D \otimes E$ to be the set

$$\{t \subseteq D_p \times E_p \mid \pi_1(t)\uparrow \ \wedge\ \pi_2(t)\uparrow \ \wedge\ t \text{ is down-closed}\}$$

ordered by inclusion, and given linear stable functions $f : D \to D'$ and $g : E \to E'$ we define the linear stable function $f \otimes g : D \otimes E \to D' \otimes E'$ as

$$\{(\ulcorner (d, e) \urcorner, \ulcorner (d', e') \urcorner) \mid (d, d') \in f \ \wedge \ (e, e') \in g\}.$$

It is now straightforward to check that I together with the bifunctor $\otimes$ constitutes a symmetric monoidal structure. Note that

$$(D \otimes E)_o = \{t \in D \otimes E \,|\, |t| < \infty\}$$
$$(D \otimes E)_p = \{\ulcorner (d,e) \urcorner \,|\, d \in D_p \ \wedge \ e \in E_p\}$$

The last equality entails that $(D \otimes E)_p$ is isomorphic to $D_p \times E_p$ equipped with the coordinate wise order. This is the key in showing that every functor $(\Leftrightarrow) \otimes D$ has a right adjoint $D \multimap (\Leftrightarrow)$.

Definition 2.6.9 Given dI-domains D and E we define $D \multimap E$ to be the set of linear stable functions from D to E ordered by inclusion of their traces.

It is easy to check that a function f is prime iff $f = f \cap \ulcorner (d,e) \urcorner$ for some $(d,e) \in f$; if the function f is prime then the uniquely determined (d,e) with that property will be denoted $top(f)$.

Definition 2.6.10 Given dI-domains E and D we define a linear stable function

$$(D \multimap E) \otimes D \xrightarrow{eval_{E,D}} E$$

as

$$\{(\ulcorner (f,d) \urcorner, e) \,|\, f \in (D \multimap E)_p \ \wedge \ top(f) = (d,e)\}.$$

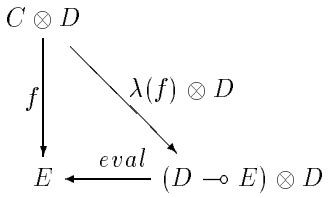
It is straightforward to extend the function $D \multimap (\Leftrightarrow)$ to a functor which is right adjoint to $(\Leftrightarrow) \otimes D$ such that $eval$ is counit: Given a linear stable function $f : C \otimes D \to E$ we define for every $(\ulcorner (c,d) \urcorner, e) \in f$ a linear stable prime function $\| \ (d,e) \ \|^f_c : D \to E$ as

$$\{(d',e') \,|\, d' \sqsubseteq d \ \wedge \ e' \sqsubseteq e \ \wedge \ \exists c' \sqsubseteq c.\ (\ulcorner (c',d') \urcorner, e') \in f\}$$

which enables us to define a linear stable function $\lambda(f) : C \to (D \multimap E)$ as

$$\{(c, \| \ (d,e) \ \|^f_c) \,|\, (\ulcorner (c,d) \urcorner, e) \in f\}.$$

It is then the case that $\lambda(f)$ is a unique linear stable function with the property that the diagram



commutes. We will now give a comonad on the category $\mathbf{dI}_{lin}$ by giving a left adjoint to the forgetful functor

$$U : \mathbf{dI}_{lin} \hookrightarrow \mathbf{dI}.$$

Definition 2.6.11 Given a dI-domain D we define $!D$ as

$$\{t \subseteq D_o \,|\, t\uparrow \ \wedge\, t \text{ is down-closed}\}$$

ordered by inclusion.

Note that

$$(!D)_o = \{t \in !D \,|\, |t| < \infty\}$$
$$(!D)_p = \{\ulcorner d \urcorner \,|\, d \in D_o\}$$

The last equality entails that $(!D)_p$ is isomorphic to D_o. This is the key in showing that U has a left adjoint.

Definition 2.6.12 Given a dI-domain D we define a continuous stable function

$$D \xrightarrow{\eta_D} !D$$

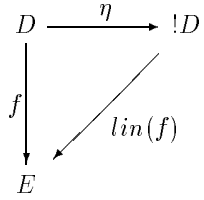
as

$$\{(d, \ulcorner d \urcorner) \mid d \in D_o\}.$$

It is straightforward to extend the function ! to a functor from **dI** to $\mathbf{dI}_{lin}$ which is left adjoint to U such that η is unit: Given a continuous stable function $f : D \to E$ we define a linear stable function $lin(f) : !D \to E$ as

$$\{(\ulcorner d \urcorner, e) \mid (d, e) \in f\}.$$

It is then the case that $lin(f)$ is a unique linear stable function with the property that the diagram



commutes. Note that the diagram lives in the category **dI**. The adjunction induces a comonad on $\mathbf{dI}_{lin}$ consisting of the functor ! restricted to an endofunctor on $\mathbf{dI}_{lin}$ together with a natural transformation ε where a component $\varepsilon_D : !D \to D$ is equal to $lin(id_D)$ and a natural transformation δ where a component $\delta_D : !D \to !!D$ is equal to $!\eta_D$. It will be useful to state some of the constructions explicitly. Given a continuous stable function $f : D \to E$ the linear stable function $!f : !D \to !E$ is equal to

$$\{(\ulcorner d \urcorner, \ulcorner e \urcorner) \mid d \in D_o \ \wedge \ e \in E_o \ \wedge \ e \sqsubseteq f(d) \ \wedge \ \forall d' \sqsubseteq d.\, e \sqsubseteq f(d') \ \Rightarrow \ d = d'\}$$

which is also equal to

$$\{(\ulcorner d_1 \sqcup \ldots \sqcup d_n \urcorner, \ulcorner e_1 \sqcup \ldots \sqcup e_n \urcorner) \mid (d_1, e_1), \ldots, (d_n, e_n) \in f \ \wedge \ \{d_1, \ldots, d_n\}\uparrow\}.$$

The linear stable function $\varepsilon_D : !D \to D$ is equal to

$$\{(\ulcorner d \urcorner, d) \mid d \in D_p\}.$$

The linear stable function $\delta_D : !D \to !!D$ is equal to

$$\{(\ulcorner d_1 \sqcup \ldots \sqcup d_n \urcorner, \ulcorner (\ulcorner d_1 \urcorner \cup \ldots \cup \ulcorner d_n \urcorner) \urcorner) \mid d_1, \ldots, d_n \in D_o \ \wedge \ \{d_1, \ldots, d_n\}\uparrow\}.$$

We want the comonad $(!, \varepsilon, \delta)$ to be symmetric monoidal; this motivates the following definition:

Definition 2.6.13 We define a linear stable function $m_I : I \to !I$ as

$$\{(\top, \{\bot\}), (\top, \{\bot, \top\})\}$$

and we define a natural transformation m with components $m_{D,E} : !D \otimes !E \to !(D \otimes E)$ by defining for each pair of dI-domains D and E a linear stable function $m_{D,E}$ as

$$\{(\ulcorner (\ulcorner \sqcup (\pi_1(t)) \urcorner, \ulcorner \sqcup (\pi_2(t)) \urcorner) \urcorner, \ulcorner t \urcorner) \mid t \in (D \otimes E)_o\}$$

It is now straightforward to check that the function m_I together with the natural transformation m gives symmetric monoidal structure to the functor ! such that ε and δ are monoidal natural transformations. Furthermore:

Definition 2.6.14 We define a natural transformation e with components $e_D : !D \to I$ by defining for each dI-domain D a linear stable function e_D as

$$\{(\{\bot\}, \top)\}$$

and we define a natural transformation d with components $d_D : !D \to !D \otimes !D$ by defining for each dI-domain D a linear stable function d_D as

$$\{(\ulcorner d \sqcup d' \urcorner, \ulcorner (\ulcorner d \urcorner, \ulcorner d' \urcorner) \urcorner) \mid d, d' \in D_o \ \wedge \ d \uparrow d'\}$$

It is now straightforward to check that the natural transformations e and d are monoidal such that the three additional conditions on being a linear category, Definition 2.2.1, are satisfied. It is easy to check that $\mathbf{dI}_{lin}$ has finite products and sums; the former is the usual construction for posets and the latter is the "coalesced sum". The Kleisli category corresponding to the comonad ! is isomorphic to **dI**, so the results of Section 2.4 imply that the finite products of $\mathbf{dI}_{lin}$ make **dI** cartesian closed, and the finite sums of $\mathbf{dI}_{lin}$ induce a weak finite sum structure on **dI**.

Chapter 3

The λ-Calculus

The primary goal of this chapter is to introduce the typed λ-calculus and show its proof-theoretic significance. We start out by introducing Intuitionistic Logic in Section 3.1, the syntax of the λ-calculus is introduced in Section 3.2, and Section 3.3 gives an account of the λ-calculus as a Curry-Howard interpretation of Intuitionistic Logic. Given an appropriate categorical model, in Section 3.4 we recall how this induces a categorical interpretation.

3.1 Intuitionistic Logic

The presentation of Intuitionistic Logic given in this section is based on the book [GLT89]. Formulae of Intuitionistic Logic are given by the grammar

$$s \quad ::= \quad 1 \mid s \times s \mid s \Rightarrow s \mid 0 \mid s + s.$$

The metavariables A, B, C range over formulae. Proof-rules for a Natural Deduction presentation of the logic are given in Figure 3.1; they are used to derive sequents

$$A_1, ..., A_n \vdash B.$$

This amounts to a formula expressing that the conjunction of $A_1, ..., A_n$ implies B. The metavariables $,, \Delta$ range over lists of formulae and π, τ range over proofs. The $,$ part of a sequent $, \vdash B$ is called the *context* of the sequent.

The Natural Deduction style proof-rules were originally introduced by Gentzen in [Gen34] and later on considered by Pravitz in [Pra65]. This style of presentation is characterised by the presence of two different forms of rules for each connective, namely *introduction* and *elimination* rules. Note that in Figure 3.1 the introduction rules have been positioned in the left hand side column, and the elimination rules have been positioned in the right hand side column.

A notable feature of Intuitionistic Logic is the Brouwer-Heyting-Kolmogorov functional interpretation where formulae are interpreted by means of their proofs:

- A proof of a conjunction $A \times B$ consists of a proof of A together with a proof of B,
- a proof of an implication $A \Rightarrow B$ is a function from proofs of A to proofs of B,
- a proof of a disjunction $A + B$ is either a proof of A or a proof of B together with a specification of which of the disjuncts is actually proved.

The proof-rules for Intuitionistic Logic can then be considered as methods for defining functions such that a proof of a sequent $, \vdash B$ gives rise to a function which to a list of proofs proving the respective formulas in the context $,$ assigns a proof of the formula B. Note that *tertium non datur*, $A \vee \neg A$, which distinguishes Classical Logic from Intuitionistic Logic, cannot be interpreted

Figure 3.1: Intuitionistic Logic

$$\frac{}{A_1, \ldots, A_n \vdash A_q}\, Axiom$$

$$\frac{}{, \vdash 1}\, 1_I$$

$$\frac{, \vdash A \quad , \vdash B}{, \vdash A \times B}\, \times_I \qquad \frac{, \vdash A \times B}{, \vdash A}\, \times_{E1} \qquad \frac{, \vdash A \times B}{, \vdash B}\, \times_{E2}$$

$$\frac{, , A \vdash B}{, \vdash A \Rightarrow B}\, \Rightarrow_I \qquad \frac{, \vdash A \Rightarrow B \quad , \vdash A}{, \vdash B}\, \Rightarrow_E$$

$$\frac{, \vdash 0}{, \vdash C}\, 0_E$$

$$\frac{, \vdash A}{, \vdash A + B}\, +_{I1} \qquad \frac{, \vdash B}{, \vdash A + B}\, +_{I2} \qquad \frac{, \vdash A + B \quad , , A \vdash C \quad , , B \vdash C}{, \vdash C}\, +_E$$

in this way. It turns out that the λ-calculus is an appropriate language for expressing the Brouwer-Heyting-Kolmogorov interpretation. We will come back to the λ-calculus in the next section, and in Section 3.3 we will introduce the Curry-Howard isomorphism that makes explicit the relation between the λ-calculus and Intuitionistic Logic.

In Intuitionistic Logic we have the following admissible proof-rules

$$\frac{, , A, A, \Delta \vdash B}{, , A, \Delta \vdash B} \qquad \frac{, , \Delta \vdash B}{, , A, \Delta \vdash B}$$

which are called *contraction* and *weakening*, respectively. The presence of contraction and weakening allows us to consider the context $,$ of a sequent $, \vdash B$ as a *set* of formulae instead of a *multiset* of formulae, which is a feature distinguishing Intuitionistic Logic from Intuitionistic Linear Logic.

Now, a proof may be rewritten into a simpler form using a reduction rule. The reduction rules of the presentation of Intuitionistic Logic given here are as follows:

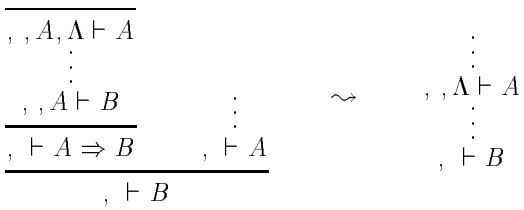
- The $(\times_I, \times_{E1})$ case

$$\frac{\dfrac{\begin{array}{c}\vdots\\ , \vdash A\end{array} \qquad \begin{array}{c}\vdots\\ , \vdash B\end{array}}{, \vdash A \times B}}{, \vdash A} \quad \rightsquigarrow \quad \begin{array}{c}\vdots\\ , \vdash A\end{array}$$

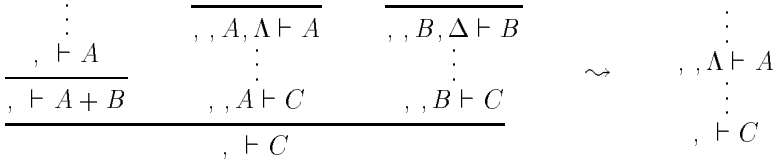
- The $(\times_I, \times_{E2})$ case

$$\frac{\dfrac{\begin{array}{c}\vdots\\ , \vdash A\end{array} \qquad \begin{array}{c}\vdots\\ , \vdash B\end{array}}{, \vdash A \times B}}{, \vdash B} \quad \rightsquigarrow \quad \begin{array}{c}\vdots\\ , \vdash B\end{array}$$

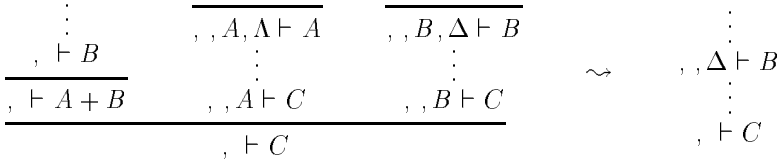
- The $(\Rightarrow_I, \Rightarrow_E)$ case



- The $(+_{I1}, +_E)$ case



- The $(+_{I2}, +_E)$ case



Note how a reduction rule removes a "detour" in the proof created by the introduction of a connective immediately followed by its elimination. Intuitionistic Logic satisfies the *Church-Rosser* property which means that whenever a proof π reduces to π' as well as to π'', there exists a proof π''' to which both of the proofs π' and π'' reduce, and moreover, it satisfies the *strong normalisation* property which means that all reduction sequences originating from a given proof are of finite length. Church-Rosser and strong normalisation imply that any proof π reduces to a unique proof with the property that no reductions can be applied; this is called the *normal form* of π. This corresponds - via the Curry-Howard isomorphism - to analogous results for reduction of terms of the λ-calculus which we will come back to in the next two sections.

3.2 Syntax of the λ-Calculus

The presentation of the λ-calculus given in this section is based on the book [GLT89]. In the next section we shall see how the λ-calculus occurs as a Curry-Howard interpretation of Intuitionistic Logic. Note that we consider products and sums as part of the λ-calculus; this convention is not followed by all authors. Types of the λ-calculus are given by the grammar

$$s \;::=\; 1 \mid s \times s \mid s \Rightarrow s \mid 0 \mid s + s$$

and terms are given by the grammar

$$\begin{array}{rcl} t & ::= & x \mid \\ & & \mathtt{true} \mid (t,t) \mid \mathtt{fst}(t) \mid \mathtt{snd}(t) \mid \\ & & \lambda x^A.t \mid tt \mid \\ & & \mathtt{false}^C(t) \mid \mathtt{inl}^{A+B}(t) \mid \mathtt{inr}^{A+B}(t) \mid \mathtt{case}\ t\ \mathtt{of}\ \mathtt{inl}(x).t \mid \mathtt{inr}(y).t \end{array}$$

Figure 3.2: Type Assignment Rules for the λ-Calculus

$$\frac{}{x_1 : A_1, ..., x_n : A_n \vdash x_q : A_q}$$

$$\frac{}{, \vdash \mathtt{true} : 1} \qquad \frac{, \vdash u : A \quad , \vdash v : B}{, \vdash (u, v) : A \times B} \qquad \frac{, \vdash u : A \times B}{, \vdash \mathtt{fst}(u) : A} \qquad \frac{, \vdash u : A \times B}{, \vdash \mathrm{snd}(u) : B}$$

$$\frac{, , x : A \vdash u : B}{, \vdash \lambda x^A.u : A \Rightarrow B} \qquad \frac{, \vdash f : A \Rightarrow B \quad , \vdash u : A}{, \vdash fu : B}$$

$$\frac{, \vdash w : 0}{, \vdash \mathtt{false}^C(w) : C} \qquad \frac{, \vdash u : A}{, \vdash \mathtt{inl}^{A+B}(u) : A + B} \qquad \frac{, \vdash u : B}{, \vdash \mathtt{inr}^{A+B}(u) : A + B}$$

$$\frac{, \vdash w : A + B \quad , , x : A \vdash u : C \quad , , y : B \vdash v : C}{, \vdash \mathtt{case}\ w\ \mathtt{of}\ \mathtt{inl}(x).u \mid \mathtt{inr}(y).v : C}$$

where x is a variable ranging over terms. The set of free variables, denoted $FV(u)$, of a term u is defined by structural induction on u as follows:

$$\begin{aligned}
FV(x) &= \{x\} \\
FV(\mathtt{true}) &= \emptyset \\
FV((u, v)) &= FV(u) \cup FV(v) \\
FV(\mathtt{fst}(u)) &= FV(u) \\
FV(\lambda x.u) &= FV(u) \Leftrightarrow \{x\} \\
FV(fu) &= FV(f) \cup FV(u) \\
FV(\mathtt{false}(u)) &= FV(u) \\
FV(\mathtt{inl}(u)) &= FV(u) \\
FV(\mathtt{case}\ w\ \mathtt{of}\ \mathtt{inl}(x).u \mid \mathtt{inr}(y).v) &= FV(w) \cup (FV(u) \Leftrightarrow \{x\}) \cup (FV(v) \Leftrightarrow \{y\})
\end{aligned}$$

We say that a term u is *closed* iff $FV(u) = \emptyset$. We also say that the variable x is *bound* in the term $\lambda x.u$. A similar remark applies to the case construction. We need a convention dealing with substitution: If a term v together with n terms $u_1, ..., u_n$ and n pairwise distinct variables $x_1, ..., x_n$ are given, then $v[u_1, ..., u_n/x_1, ..., x_n]$ denotes the term v where simultaneously the terms $u_1, ..., u_n$ have been substituted for free occurrences of the variables $x_1, ..., x_n$ such that bound variables in v have been renamed to avoid capture of free variables of the terms $u_1, ..., u_n$. Occasionally a list $u_1, ..., u_n$ of n terms will be denoted $\overline{u}$ and a list $x_1, ..., x_n$ of n pairwise distinct variables will be denoted $\overline{x}$. Given the definition of free variables above, it should be clear how to formalise substitution. We also need a convention concerning an "inverse" to substitution: If terms v and u are given together with a variable x, then $v[x/u]$ denotes the term v where inductively all occurrences of the term u have been replaced by the variable x. It is clear that if x does not occur in v and none of the free variables of u are bound in $v[x/u]$, then $v[x/u][u/x] = v$.

Rules for assignment of types to terms are given in Figure 3.2. Type assignments have the form of sequents

$$x_1 : A_1, ..., x_n : A_n \vdash u : B$$

where $x_1, ..., x_n$ are pairwise distinct variables. It can be shown by induction on the derivation of the type assignment that

$$FV(u) \subseteq \{x_1, ..., x_n\}.$$

The λ-calculus satisfies the following properties:

Lemma 3.2.1 If the sequent $, \vdash u : A$ is derivable, then for any derivable sequent $, \vdash u : B$ we have $A = B$.

Proof: Induction on the derivation of $, \vdash u : A$. □

The following proposition is the essence of the Curry-Howard isomorphism:

Proposition 3.2.2 If the sequent $, \vdash u : A$ is derivable, then the rule instance above the sequent is uniquely determined.

Proof: Use Lemma 3.2.1 to check each case. □

We need a small lemma dealing with expansion of contexts.

Lemma 3.2.3 If the sequent $\Delta, \Lambda \vdash u : A$ is derivable and the variables in the contexts Δ, Λ and $,$ are pairwise distinct, then the sequent $\Delta, , , \Lambda \vdash u : A$ is also derivable.

Proof: Induction on the derivation of $\Delta, \Lambda \vdash u : A$. □

Now comes a lemma dealing with substitution.

Lemma 3.2.4 (Substitution Property) If the sequents $, \vdash u : A$ and $, , x : A, \Lambda \vdash v : B$ are derivable, then the sequent $, , \Lambda \vdash v[u/x] : B$ is also derivable.

Proof: Induction on the derivation of $, , x : A, \Lambda \vdash v : B$. We need Lemma 3.2.3 for the case where the derivation is an axiom

$$\frac{}{x_1 : A_1, ..., x_n : A_n \vdash x_q : A_q}$$

such that the variable x is equal to x_q. □

We also have a lemma dealing with the "inverse" to substitution; this will be useful when in Chapter 7 we consider PCF. Recall that $v[x/u]$ denotes the term v where inductively all occurrences of the term u have been replaced by the variable x.

Lemma 3.2.5 If the sequents $, \vdash u : A$ and $, , \Lambda \vdash v : B$ are derivable, then the sequent $, , x : A, \Lambda \vdash v[x/u] : B$ is also derivable where x is a new variable. the sequent $, , \Lambda \vdash v : B$.

Proof: Induction on the derivation of $, , \Lambda \vdash v : B$. □

The λ-calculus has the following β-reduction rules each of which is the image under the Curry-Howard isomorphism of a reduction on the proof corresponding to the involved term:

$$\begin{aligned}
\mathtt{fst}((u, v)) &\rightsquigarrow u \\
\mathrm{snd}((u, v)) &\rightsquigarrow v \\
(\lambda x.u)w &\rightsquigarrow u[w/x] \\
\mathtt{case\ inl}(w)\ \mathtt{of\ inl}(x).u \mid \mathtt{inr}(y).v &\rightsquigarrow u[w/x] \\
\mathtt{case\ inr}(w)\ \mathtt{of\ inl}(x).u \mid \mathtt{inr}(y).v &\rightsquigarrow v[w/y]
\end{aligned}$$

We shall not be concerned with η-reductions or commuting conversions. The properties of Church-Rosser and strong normalisation for proofs of Intuitionistic Logic correspond to analogous notions for terms of the λ-calculus via the Curry-Howard isomorphism, and in [GLT89] it is shown that these properties are indeed satisfied by terms of the λ-calculus.

In Chapter 7 we will introduce the programming language PCF which is the λ-calculus augmented with numerals and recursion where the above mentioned reduction rules are replaced by an operational semantics.

3.3 The Curry-Howard Isomorphism

The original Curry-Howard isomorphism, [How80], relates the Natural Deduction formulation of Intuitionistic Logic to the λ-calculus; formulae correspond to types, proofs to terms, and reduction of proofs to reduction of terms. This is dealt with in [GLT89] and in [Abr90]; the first emphasises the logic side of the isomorphism, the second the computational side. In what follows, we will consider the Natural Deduction presentation of Intuitionistic Logic given in Figure 3.1. The relation between formulae of Intuitionistic Logic and types of the λ-calculus is obvious; they are simply identical. The idea of the Curry-Howard isomorphism on the level of proofs is that proof-rules can be "decorated" with terms such that the term induced by a proof encodes the proof. An appropriate term language for this purpose is - in the case of Intuitionistic Logic - the λ-calculus. We get the rules for assigning types to terms of λ-calculus if we decorate the proof-rules of Intuitionistic Logic with terms in the appropriate way, and moreover, we can recover the proof-rules if we take the typing rules of the λ-calculus and remove the variables and terms. We get the Curry-Howard isomorphism on the level of proofs as follows: Given a proof of the sequent $A_1, ..., A_n \vdash B$, that is, a proof of the formula B on assumptions $A_1, ..., A_n$, one can inductively construct a derivation of a sequent $x_1 : A_1, ..., x_n : A_n \vdash u : B$, that is, a term u of type B with free variables $x_1, ..., x_1$ of respective types $A_1, ..., A_n$. Conversely, if one has a derivable sequent $x_1 : A_1, ..., x_n : A_n \vdash u : B$, there is an easy way to get a proof of $A_1, ..., A_n \vdash B$; erase all terms and variables in the derivation of the type assignment. The two processes are each other's inverses modulo renaming of variables. The isomorphism on the level of proofs is essentially given by Proposition 3.2.2.

On the level of reduction the Curry-Howard isomorphism says that a reduction on a proof followed by application of the Curry-Howard isomorphism on the level of proofs, yields the same term as application of the Curry-Howard isomorphism on the level of proofs followed by the term-reduction corresponding to the proof-reduction. This can be verified by applying the Curry-Howard isomorphism to the proofs involved in the reduction rules of Intuitionistic Logic. For example, in the case of a $(\Rightarrow_I, \Rightarrow_E)$ reduction we get

$$\dfrac{\dfrac{\dfrac{}{, , x : A, \Lambda \vdash x : A}}{\begin{array}{c}\vdots\\ , , x : A \vdash u : B\end{array}}}{\dfrac{, \vdash \lambda x.u : A \Rightarrow B \qquad , \vdash v : A}{, \vdash (\lambda x.u)v : B}} \quad \rightsquigarrow \quad \begin{array}{c}\vdots\\ , , \Lambda \vdash v : A\\ \vdots\\ , \vdash u[v/x] : B\end{array}$$

We see that a β-reduction has taken place on the term encoding the proof on which the reduction is performed. In fact all β-reductions appear as Curry-Howard interpretations of reductions on the corresponding proofs.

3.4 Categorical Semantics

The categorical semantics adheres to the following fundamental ideas of the categorical treatment of proof-theory:

- Formulae (that is, types) are interpreted as objects,
- proof-rules (that is, typing rules) are interpreted as natural operations on maps,
- proofs (that is, derivations of type assignments) are interpreted as maps.

We will interpret the product and function types of the λ-calculus using a cartesian closed structure. It is not so obvious how to interpret the sum type. If we assume the presence of finite sums,

then the elimination rule for + can be interpreted using the operation on maps

$$\frac{A \times , \xrightarrow{u} C \qquad B \times , \xrightarrow{u} C}{(A+B) \times , \xrightarrow{\lambda^{-1}([\lambda(u),\lambda(v)])} C}$$

which can be shown to be natural in , . This would be equivalent to using the natural isomorphism

$$(A+B) \times , \cong (A \times ,) + (B \times ,)$$

given by the observation that the functor $(\Leftrightarrow) \times ,$ has a right adjoint and thus preserves sums. An analogous remark applies to the elimination rule for 0. But in [HP90] a cartesian closed category with finite sums together with a fixpoint operator is shown to be *inconsistent*, that is, it is equivalent to the category consisting of one object and one map. In Chapter 7 we shall add a fixpoint operator to the picture, so we have to be content with a less demanding assumption than finite sums.

Definition 3.4.1 A *categorical model for the λ-calculus* is a cartesian closed category with weak finite sums such that the diagram

$$\begin{array}{ccc} A+B & \xrightarrow{[f,g]} & , \Rightarrow C \\ & \searrow^{[(f;(h\Rightarrow C)),(g;(h\Rightarrow C))]} & \downarrow{\scriptstyle h\Rightarrow C} \\ & & \Delta \Rightarrow C \end{array}$$

commutes for any maps $f : A \to , \Rightarrow C$, $g : B \to , \Rightarrow C$ and $h : \Delta \to , .$

Then the above mentioned operation can still be defined, and commutativity of the diagram is equivalent to the operation being natural in , , as can be shown by some equational manipulation.

Examples 3.4.2 It is made clear in Chapter 2 that the concrete categories **cpo** and **dI** are categorical models for the λ-calculus.

Types are interpreted as objects, typing rules as natural operations on maps, and derivations of type assignments as maps. A derivable sequent

$$x_1 : A_1, ..., x_n : A_n \vdash u : B$$

is interpreted as a map

$$[\![A_1]\!] \times ... \times [\![A_n]\!] \xrightarrow{[\![u]\!]} [\![B]\!]$$

by induction on its derivation using the appropriate operations on maps induced by the categorical model cf. below. We consider each case except the symmetric ones. When appropriate we will abuse the notation and omit the $[\![\Leftrightarrow]\!]$ brackets.

- The derivation

  $$\frac{}{x_1 : A_1, ..., x_n : A_n \vdash x_p : A_p}$$

  is interpreted as

  $$\frac{}{A_1 \times ... \times A_n \xrightarrow{\pi_p} A_p}$$

- The derivation

  $$\frac{}{, \vdash \mathtt{true} : 1}$$

  is interpreted as

  $$\frac{}{, \xrightarrow{\langle\rangle} 1}$$

- The derivation

$$\frac{\Gamma \vdash u : A \qquad \Gamma \vdash v : B}{\Gamma \vdash (u, v) : A \times B}$$

  is interpreted as

$$\frac{\Gamma \xrightarrow{u} A \qquad \Gamma \xrightarrow{v} B}{\Gamma \xrightarrow{\langle u,v \rangle} A \times B}$$

- The derivation

$$\frac{\Gamma \vdash u : A \times B}{\Gamma \vdash \mathtt{fst}(u) : A}$$

  is interpreted as

$$\frac{\Gamma \xrightarrow{u} A \times B}{\Gamma \xrightarrow{u} A \times B \xrightarrow{\pi_1} A}$$

- The derivation

$$\frac{\Gamma, x : A \vdash u : B}{\Gamma \vdash \lambda x.u : A \Rightarrow B}$$

  is interpreted as

$$\frac{\Gamma \times A \xrightarrow{u} B}{\Gamma \xrightarrow{\lambda(u)} A \Rightarrow B}$$

- The derivation

$$\frac{\Gamma \vdash f : A \Rightarrow B \qquad \Gamma \vdash u : A}{\Gamma \vdash fu : B}$$

  is interpreted as

$$\frac{\Gamma \xrightarrow{f} A \Rightarrow B \qquad \Gamma \xrightarrow{u} A}{\Gamma \xrightarrow{\langle f,u \rangle} (A \Rightarrow B) \times A \xrightarrow{eval} B}$$

- The derivation

$$\frac{\Gamma \vdash w : 0}{\Gamma \vdash \mathtt{false}(w) : C}$$

  is interpreted as

$$\frac{\Gamma \xrightarrow{w} 0}{\Gamma \xrightarrow{w} 0 \xrightarrow{[]} C}$$

- The derivation

$$\frac{\Gamma \vdash u : A}{\Gamma \vdash \mathtt{inl}(u) : A + B}$$

  is interpreted as

$$\frac{\Gamma \xrightarrow{u} A}{\Gamma \xrightarrow{u} A \xrightarrow{in_1} A + B}$$

- The derivation

$$\frac{\Gamma \vdash w : A + B \qquad \Gamma, x : A \vdash u : C \qquad \Gamma, y : B \vdash v : C}{\Gamma \vdash \mathtt{case}\ w\ \mathtt{of}\ \mathtt{inl}(x).u \mid \mathtt{inr}(y).v : C}$$

  is interpreted as

$$\frac{\Gamma \xrightarrow{w} A + B \qquad \Gamma \times A \xrightarrow{u} C \qquad \Gamma \times B \xrightarrow{v} C}{\Gamma \xrightarrow{\langle w,\Gamma \rangle} (A + B) \times \Gamma \xrightarrow{\lambda^{-1}([\lambda(\cong;u),\lambda(\cong;v)])} C}$$

Note that the derivation of the sequent is uniquely determined according to Proposition 3.2.2 so it makes sense to speak of *the* interpretation of a derivable sequent without mentioning its derivation explicitly. It can be shown that the operations on maps induced by the typing rules are natural in the interpretation of the unchanged components of the sequents. The following lemma corresponds to Lemma 3.2.3 where the categorical interpretation has been taken into account:

Lemma 3.4.3 If the sequent $\Delta, \Lambda \vdash u : A$ is derivable and the variables in the contexts Δ, Λ and $,$ are pairwise distinct, then the sequent $\Delta, , , \Lambda \vdash u : A$ is also derivable and it has the interpretation

$$\Delta \times , \times \Lambda \xrightarrow{\Delta \times \langle\rangle \times \Lambda} \Delta \times 1 \times \Lambda \cong \Delta \times \Lambda \xrightarrow{[\![u]\!]} A$$

Proof: Induction on the derivation of $\Delta, \Lambda \vdash u : A$. We use the observation that the operations on maps induced by the typing rules are natural in the interpretation of the unchanged components of the contexts of the sequents. □

The following lemma corresponds to Lemma 3.2.4 where the categorical interpretation has been taken into account; it says how substitution relates to composition:

Lemma 3.4.4 (Substitution) If the sequents $, \vdash u : A$ and $, , x : A, \Lambda \vdash v : B$ are derivable, then the sequent $, , \Lambda \vdash v[u/x] : B$ is also derivable and it has the interpretation

$$, \times \Lambda \xrightarrow{\Delta \times \Lambda} , \times , \times \Lambda \xrightarrow{\Gamma \times [\![u]\!] \times \Lambda} , \times A \times \Lambda \xrightarrow{[\![v]\!]} B$$

Proof: Induction on the derivation of $, , x : A, \Lambda \vdash v : B$. We use the observation that the operations on maps induced by the typing rules are natural in the interpretation of the unchanged components of the contexts of the sequents. □

Using Lemma 3.4.4, one can show that the categorical interpretation is sound with respect to β-reductions. It should be emphasised that we give a sound interpretation of the λ-calculus by using *weak* finite sums such that the diagram above commutes, and not by using finite sums. This is a categorical generalisation of the sound interpretation given in [GLT89], where the category of coherence spaces and stable continuous functions is used; a category that has weak finite sums, but not finite sums. It is actually the case that if we have a linear category with finite products and finite sums, then the induced Kleisli category is a cartesian closed category with weak finite sums such that the above mentioned diagram commutes according to Proposition 2.4.2 and Proposition 2.4.3.

Chapter 4

The Linear λ-Calculus

The primary goal of this chapter is to introduce the linear λ-calculus. We start out by introducing Intuitionistic Linear Logic in Section 4.1. In Section 4.2 we make a digression to Russell's Paradox with the aim of illustrating the fine grained character of Intuitionistic Linear Logic compared to Intuitionistic Logic. The syntax of the linear λ-calculus is introduced in Section 4.3 and Section 4.4 gives an account of the linear λ-calculus as a Curry-Howard interpretation of Intuitionistic Linear Logic. Given an appropriate categorical model, in Section 4.5 we recall how this induces a categorical interpretation. In Section 4.6 we introduce a generalisation of the linear λ-calculus that will be of use later on. In Section 4.7 the generalised version the linear λ-calculus is given a categorical semantics.

4.1 Intuitionistic Linear Logic

Linear Logic was discovered by J.-Y. Girard in 1987 and published in the now famous paper [Gir87]. In the abstract of this paper, it is stated that "a completely new approach to the whole area between constructive logics and computer science is initiated". In [Gir89] the conceptual background of Linear Logic is worked out. This section deals with the intuitionistic fragment of Linear Logic. The presentation of Intuitionistic Linear Logic we give here is the same as the one given in [BBdPH92b], and later on fleshed out in detail in [Bie94]. Formulae of Intuitionistic Linear Logic are given by the grammar

$$s \ ::= \ I \mid s \otimes s \mid s \multimap s \mid !s \mid 1 \mid s \times s \mid 0 \mid s + s.$$

Proof-rules for a Natural Deduction presentation of the logic are given in Figure 4.1; they are used to derive sequents

$$A_1, ..., A_n \mathbin{|\!\!\Leftrightarrow} B.$$

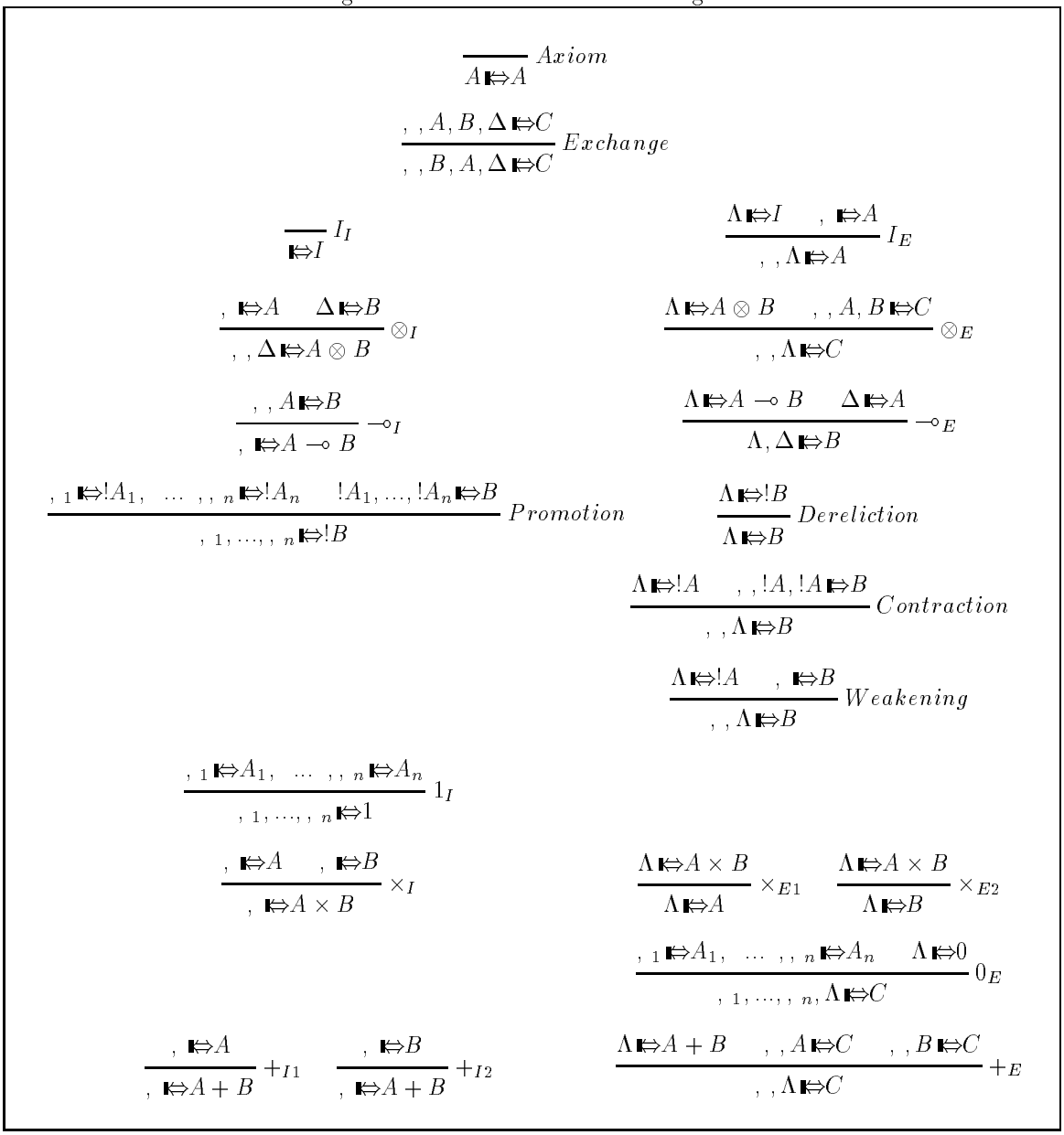
Note that in Figure 4.1 the introduction rules have been positioned in the left hand side column, and the elimination rules have been positioned in the right hand side column. A Girardian turnstile $|\!\!\Leftrightarrow$ is used to distinguish sequents of Intuitionistic Linear Logic from sequents of Intuitionistic Logic, where the usual turnstile $\vdash$ is used. Intuitionistic Linear Logic does not include the "par" construct of Classical Linear Logic; but it is possible to deal with it intuitionistically by allowing sequents to have more than one conclusion - see [BdP96, HdP93].

The fundamental idea of Linear Logic is to control the use of resources which is witnessed by the observation that the contraction and weakening proof-rules

$$\frac{, , A, A, \Delta \mathbin{|\!\!\Leftrightarrow} B}{, , A, \Delta \mathbin{|\!\!\Leftrightarrow} B} \qquad \frac{, , \Delta \mathbin{|\!\!\Leftrightarrow} B}{, , A, \Delta \mathbin{|\!\!\Leftrightarrow} B}$$

are not admissible. The absence of contraction and weakening prevents us from considering the context $,$ of a sequent $, \mathbin{|\!\!\Leftrightarrow} B$ as a *set* of formulae, but we have to consider it to be a *multiset* of

Figure 4.1: Intuitionistic Linear Logic



formulae instead. This should be compared to Intuitionistic Logic where contraction and weakening are admissible and contexts therefore can be considered as sets of formulae. This means that every formula occuring in the context of a sequent has to be used exactly once. Therefore the two conjunctions $\times$ and $\otimes$ of Linear Logic are very different constructs: A proof of $A \times B$ consists of a proof of A together with a proof of B where exactly one of the proofs has to be used. A proof of $A \otimes B$ also consists of a proof of A together with a proof of B but here both of the proofs have to be used. A restricted form of contraction and weakening is, however, available by having the proof-rules

$$\frac{, , !A, !A, \Delta \Vdash B}{, , !A, \Delta \Vdash B} \qquad \frac{, , \Delta \Vdash B}{, , !A, \Delta \Vdash B}$$

explicitly as part of Intuitionistic Linear Logic; they are special cases of the *Contraction* and *Weakening* rules, respectively. So a proof of $!A$ amounts to having a proof of A that can be used an arbitrary number of times. It turns out that the ! modality enables Intuitionistic Logic to be interpreted faithfully in Intuitionistic Linear Logic via the Girard Translation - see Section 5.

As in Intuitionistic Logic, a proof might be rewritten into a simpler form using a reduction rule. The reduction rules of the presentation of Intuitionistic Linear Logic given here are as follows:

- The (I_I, I_E) case

$$\frac{\dfrac{}{\Vdash I} \qquad \begin{array}{c}\vdots\\ , \Vdash A\end{array}}{, \Vdash A} \quad \rightsquigarrow \quad \begin{array}{c}\vdots\\ , \Vdash A\end{array}$$

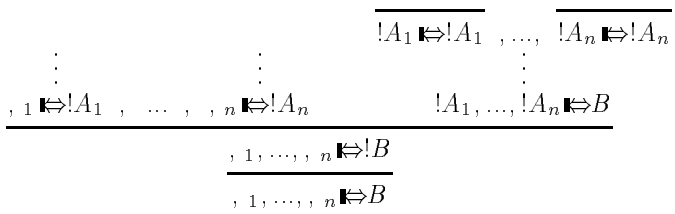
- The $(\otimes_I, \otimes_E)$ case

$$\frac{\dfrac{\begin{array}{c}\vdots\\ , \Vdash A\end{array} \qquad \begin{array}{c}\vdots\\ \Delta \Vdash B\end{array}}{, , \Delta \Vdash A \otimes B} \qquad \dfrac{\dfrac{}{A \Vdash A} \quad \dfrac{}{B \Vdash B}}{\begin{array}{c}\vdots\\ \Lambda, A, B \Vdash C\end{array}}}{\Lambda, , , \Delta \Vdash C} \quad \rightsquigarrow \quad \begin{array}{c}\begin{array}{cc}\vdots & \vdots\\ , \Vdash A & \Delta \Vdash B\end{array}\\ \vdots\\ \Lambda, , , \Delta \Vdash C\end{array}$$

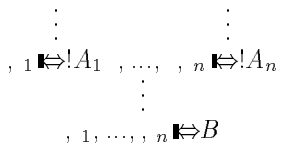
- The $(\multimap_I, \multimap_E)$ case

$$\frac{\dfrac{\dfrac{}{A \Vdash A}}{\begin{array}{c}\vdots\\ , , A \Vdash B\end{array}}}{, \Vdash A \multimap B} \qquad \begin{array}{c}\vdots\\ \Lambda \Vdash A\end{array}}{, , \Lambda \Vdash B} \quad \rightsquigarrow \quad \begin{array}{c}\vdots\\ \Lambda \Vdash A\\ \vdots\\ , , \Lambda \Vdash B\end{array}$$

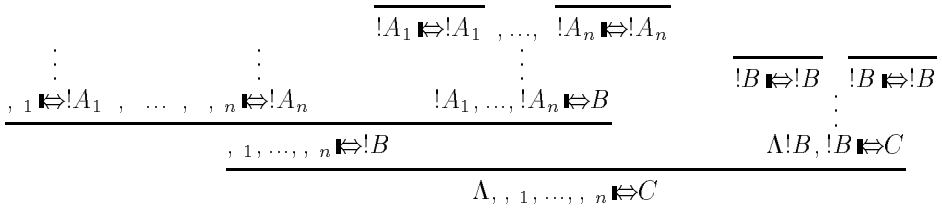
- The $(Promotion, Dereliction)$ case



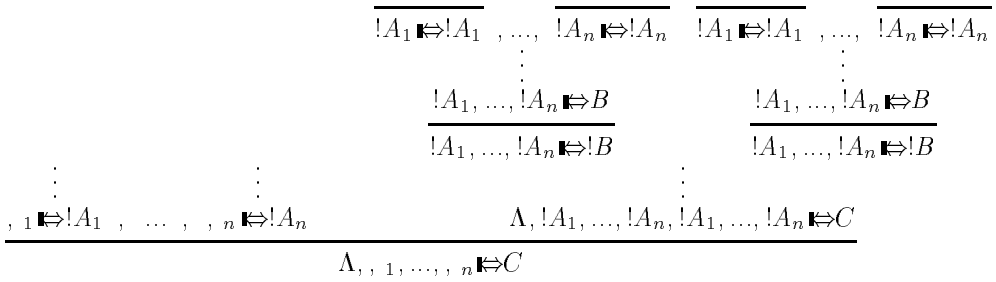
$\rightsquigarrow$



- The $(Promotion, Contraction)$ case



$\rightsquigarrow$



where the last used rule is derivable by using the *Contraction* and *Exchange* rules. Note that a special case of the *Contraction* rule is used.

- The $(Promotion, Weakening)$ case

$$\dfrac{\dfrac{\vdots}{,_1 \Rightarrow !A_1 \;,\; \ldots \;,\; ,_n \Rightarrow !A_n} \qquad \dfrac{\overline{!A_1 \Rightarrow !A_1} \;,\ldots,\; \overline{!A_n \Rightarrow !A_n}}{\vdots}\;{\scriptstyle !A_1,\ldots,!A_n \Rightarrow B}}{,_1,\ldots,,_n \Rightarrow !B} \qquad \Lambda \Rightarrow C}{\Lambda,,_1,\ldots,,_n \Rightarrow C}$$

$$\rightsquigarrow$$

$$\dfrac{\dfrac{\vdots}{,_1 \Rightarrow !A_1 \;,\; \ldots \;,\; ,_n \Rightarrow !A_n} \qquad \Lambda \Rightarrow C}{\Lambda,,_1,\ldots,,_n \Rightarrow C}$$

where the last used rule is derivable by using the $Weakening$ rule.

- The $(\times_I, \times_{E1})$ case

$$\dfrac{\dfrac{\dfrac{\vdots}{, \Rightarrow A} \qquad \dfrac{\vdots}{, \Rightarrow B}}{, \Rightarrow A \times B}}{, \Rightarrow A} \quad \rightsquigarrow \quad \dfrac{\vdots}{, \Rightarrow A}$$

- The $(\times_I, \times_{E2})$ case

$$\dfrac{\dfrac{\dfrac{\vdots}{, \Rightarrow A} \qquad \dfrac{\vdots}{, \Rightarrow B}}{, \Rightarrow A \times B}}{, \Rightarrow B} \quad \rightsquigarrow \quad \dfrac{\vdots}{, \Rightarrow B}$$

- The $(+_{I1}, +_E)$ case

$$\dfrac{\dfrac{\dfrac{\vdots}{\Lambda \Rightarrow A}}{\Lambda \Rightarrow A + B} \qquad \dfrac{\overline{A \Rightarrow A}}{\vdots}\;{\scriptstyle ,,A \Rightarrow C} \qquad \dfrac{\overline{B \Rightarrow B}}{\vdots}\;{\scriptstyle ,,B \Rightarrow C}}{,,\Lambda \Rightarrow C} \quad \rightsquigarrow \quad \dfrac{\vdots}{\Lambda \Rightarrow A}\;{\scriptstyle ,,\Lambda \Rightarrow C}$$

- The $(+_{I2}, +_E)$ case

$$\dfrac{\dfrac{\dfrac{\vdots}{\Lambda \Rightarrow B}}{\Lambda \Rightarrow A + B} \qquad \dfrac{\overline{A \Rightarrow A}}{\vdots}\;{\scriptstyle ,,A \Rightarrow C} \qquad \dfrac{\overline{B \Rightarrow B}}{\vdots}\;{\scriptstyle ,,B \Rightarrow C}}{,,\Lambda \Rightarrow C} \quad \rightsquigarrow \quad \dfrac{\vdots}{\Lambda \Rightarrow B}\;{\scriptstyle ,,\Lambda \Rightarrow C}$$

If we think of the *Promotion* rule as putting a "box" around the right hand side proof, then $(Promotion, Dereliction)$ reduction removes the box, whereas the $(Promotion, Contraction)$ and $(Promotion, Weakening)$ reductions respectively copy and discard the box. Notions of Church-Rosser and strong normalisation for Intuitionistic Linear Logic are defined in analogy with the notions of Church-Rosser and strong normalisation for Intuitionistic Logic. Intuitionistic Linear Logic does indeed satisfy these properties; this corresponds - via a Curry-Howard isomorphism - to analogous results for reduction of terms of the linear λ-calculus which we will return to in Section 4.3 and Section 4.4.

4.2 A Digression - Russell's Paradox and Linear Logic

In this section we will make a digression with the aim of illustrating the fine grained character of Intuitionistic Linear Logic compared to Intuitionistic Logic. We will take set-theoretic comprehension into account: In both of the logics unrestricted comprehension enables a contradiction to be proved via Russell's Paradox, but it turns out that Intuitionistic Linear Logic allows the presence of a restricted form of comprehension, which is not possible in the case of Intuitionistic Logic.

Unrestricted comprehension says that for any predicate $A(x)$ there is a set $\{x \mid A(x)\}$ with the property that

$$t \in \{x \mid A(x)\} \quad \Leftrightarrow \quad A(t).$$

This is a very strong axiom; it has all the axioms of Zermelo-Fraenkel set theory except the Axiom of Choice as special cases. An informal proof of Russell's Paradox now goes as follows: Using comprehension we define a set u as

$$u = \{x \mid \neg(x \in x)\}.$$

Thus, u is the famous set of all those sets that are not elements of themselves. We then define a formula R to be

$$u \in u.$$

We then have $R \Leftrightarrow \neg R$ which enables a contradiction to be proved as follows: *Assume* R, this entails $\neg R$, which *together with* R entails a contradiction. We have thus proved $\neg R$. The proof of $\neg R$ also gives a proof of R. But a proof of $\neg R$ together with a proof of R entails a contradiction.

Note how the contradiction is proved in two stages: First a formula R such that $R \Leftrightarrow \neg R$ is defined, then a contradiction is derived in a proof where the assumption R is used twice. The two applications of the assumption R are emphasised in the informal proof above. The are two ways of remedying this inconsistency:

- Unrestricted comprehension is replaced by weaker axioms such that it is impossible to define the set u and hence the formula R with the property that $R \Leftrightarrow \neg R$ cannot be defined either. This was the option taken historically and which gave rise to Zermelo-Fraenkel set theory.
- Unrestricted comprehension is kept but the surrounding logic is weakened such that the existence of a formula R with the property that $R \Leftrightarrow \neg R$ does not imply a contradiction. The !-free fragment of Linear Logic is one such option as assumptions here can be used only once (recall that in deriving a contradiction from $R \Leftrightarrow \neg R$ the assumption R is used twice).

We will now flesh out some details of the second option. First we give a formal proof of Russell's Paradox in Intuitionistic Logic extended with unrestricted comprehension as prescribed in [Pra65]. Recall that in Intuitionistic Logic negation $\neg A$ is defined as $A \Rightarrow 0$. The grammar for formulae of Intuitionistic Logic is extended with an additional clause

$$s \quad ::= \ldots \mid \; t \in t$$

and a grammar for terms is added

$$t \quad ::= \quad x \quad \mid \; \{x \mid s\}$$

where x is a variable ranging over terms. Terms are to be thought of as sets (they should not be confused with terms of the λ-calculus). Furthermore, proof-rules for introduction and elimination of the connective $\in$ are added

$$\frac{, \vdash A[t/x]}{, \vdash t \in \{x \mid A\}} \in_I \qquad \frac{, \vdash t \in \{x \mid A\}}{, \vdash A[t/x]} \in_E$$

The first stage of Russell's Paradox is formalised as follows: We define the term u and the formula R as above and get

$$\frac{, \vdash R}{, \vdash \neg R}$$

and vice versa, which is equivalent to provability of $\vdash R \Rightarrow \neg R$ and vice versa. The second stage of the paradox is formalised as follows: Two copies of the proof

$$\dfrac{\dfrac{\dfrac{\dfrac{}{R \vdash R}}{R \vdash \neg R} \qquad \dfrac{}{R \vdash R}}{R, R \vdash 0}}{R \vdash 0}}{\vdash \neg R}$$

are applied in the proof

$$\dfrac{\dfrac{\dfrac{\vdots}{\vdash \neg R}}{\vdash R} \qquad \dfrac{\vdots}{\vdash \neg R}}{\vdash 0}$$

of a contradiction. Note that we have used the admissible contraction proof-rule together with a multiplicative version of the $\Rightarrow_E$ rule which is also admissible in Intuitionistic Logic. The presence of contraction is crucial for the proof of inconsistency to go through. In [Pra65] the following observation is made: It can be shown that the sequent $\vdash 0$ is not provable by a normal proof; this means that no reduction sequences originating from the proof of $\vdash 0$ above end in a normal proof. Indeed, the proof reduces in two stages to itself by carrying out the only performable reductions.

Now, we have shown above that unrestricted comprehension in the context of Intuitionistic Logic is inconsistent. But it turns out that we do not get inconsistency if we extend the !-free fragment of Intuitionistic Linear Logic with unrestricted comprehension as above. Negation $\neg A$ is here defined as $A \multimap 0$[1]. We still have a formula R such that $R \multimap \neg R$ and vice versa, but now we cannot prove $\not\vdash 0$ as before because contraction is forbidden. The system was proved consistent in [Gri82] using the following two observations: A proof essentially shrinks under normalisation and there is no normal proof of $\not\vdash 0$. The !-free fragment of Intuitionistic Linear Logic with unrestricted comprehension is, however, very unexpressive. A partial solution to this lack of expressiveness is to extend Intuitionistic Linear Logic with comprehension as above but with the restriction that the ! modality is not allowed to occur in the involved formula $A(t)$. We still have the formula R such that $R \multimap \neg R$ and vice versa, but it turns out that this system is consistent, which was proved in [Shi94].

Hence, the fine-grainedness of Intuitionistic Linear Logic allows the presence of a restricted form of comprehension, which is not possible in the context of Intuitionistic Logic. It should be mentioned that considerations on Russell's Paradox in the context of Linear Logic have been crucial for Girard's discovery of Light Linear Logic - see [Gir94].

[1] This is not the same as the negation $A^\perp$ of Classical Linear Logic which is equivalent to $A \multimap \perp$. This difference is, however, not of importance here.

Figure 4.2: Type Assignment Rules for the Linear λ-Calculus

$$\frac{}{x : A \Mapsto x : A}$$

$$\frac{, , x : A, y : B, \Delta \Mapsto u : C}{, , y : B, x : A, \Delta \Mapsto u : C}$$

$$\frac{}{\Mapsto * : I} \qquad \frac{\Lambda \Mapsto w : I \quad , \Mapsto u : A}{, , \Lambda \Mapsto \texttt{let } w \texttt{ be } * \texttt{ in } u : A}$$

$$\frac{, \Mapsto u : A \quad \Delta \Mapsto v : B}{, , \Delta \Mapsto u \otimes v : A \otimes B} \qquad \frac{\Lambda \Mapsto w : A \otimes B \quad , , x : A, y : B \Mapsto u : C}{, , \Lambda \Mapsto \texttt{let } w \texttt{ be } x \otimes y \texttt{ in } u : C}$$

$$\frac{, , x : A \Mapsto u : B}{, \Mapsto \lambda x^A.u : A \multimap B} \qquad \frac{\Lambda \Mapsto f : A \multimap B \quad \Delta \Mapsto u : A}{\Lambda, \Delta \Mapsto f u : B}$$

$$\frac{, _1 \Mapsto v_1 : !A_1, \ \ldots \ , , _n \Mapsto v_n : !A_n \quad x_1 : !A_1, \ldots, x_n : !A_n \Mapsto u : B}{, _1, \ldots, , _n \Mapsto \texttt{promote } v_1, \ldots, v_n \texttt{ for } x_1, \ldots, x_n \texttt{ in } u : !B} \qquad \frac{\Lambda \Mapsto u : !B}{\Lambda \Mapsto \texttt{derelict}(u) : B}$$

$$\frac{\Lambda \Mapsto w : !A \quad , , x : !A, y : !A \Mapsto u : B}{, , \Lambda \Mapsto \texttt{copy } w \texttt{ as } x, y \texttt{ in } u : B} \qquad \frac{\Lambda \Mapsto w : !A \quad , \Mapsto u : B}{, , \Lambda \Mapsto \texttt{discard } w \texttt{ in } u : B}$$

$$\frac{, \Mapsto u : A \quad , \Mapsto v : B}{, \Mapsto (u, v) : A \times B} \qquad \frac{\Lambda \Mapsto u : A \times B}{\Lambda \Mapsto \texttt{fst}(u) : A} \qquad \frac{\Lambda \Mapsto u : A \times B}{\Lambda \Mapsto \texttt{snd}(u) : B}$$

$$\frac{, _1 \Mapsto w_1 : A_1, \ \ldots \ , , _n \Mapsto w_n : A_n}{, _1, \ldots, , _n \Mapsto \texttt{true}(w_1, \ldots, w_n) : 1} \qquad \frac{, _1 \Mapsto w_1 : A_1, \ \ldots \ , , _n \Mapsto w_n : A_n \quad \Lambda \Mapsto u : 0}{, _1, \ldots, , _n, \Lambda \Mapsto \texttt{false}^C(w_1, \ldots, w_n; u) : C}$$

$$\frac{, \Mapsto u : A}{, \Mapsto \texttt{inl}^{A+B}(u) : A + B} \qquad \frac{, \Mapsto u : B}{, \Mapsto \texttt{inr}^{A+B}(u) : A + B}$$

$$\frac{\Lambda \Mapsto w : A + B \quad , , x : A \Mapsto u : C \quad , , y : B \Mapsto v : C}{, , \Lambda \Mapsto \texttt{case } w \texttt{ of inl}(x).u \mid \texttt{inr}(y).v : C}$$

4.3 Syntax of the Linear λ-Calculus

The presentation of the linear λ-calculus which we give in this section is the same as the one given in [BBdPH92b, BBdPH93], and later on fleshed out in detail in [Bie94]. The next section shows how the linear λ-calculus occurs as a Curry-Howard interpretation of Intuitionistic Linear Logic in the same way as the λ-calculus occurs as a Curry-Howard interpretation of Intuitionistic Logic. Types of the linear λ-calculus are given by the grammar

$$s \ ::= \ I \mid s \otimes s \mid s \multimap s \mid !s \mid 1 \mid s \times s \mid 0 \mid s + s$$

and terms are given by the grammar

$$
\begin{array}{rcl}
t & ::= & x \mid \\
& & * \mid \texttt{let}\ t\ \texttt{be}\ *\ \texttt{in}\ t \mid t \otimes t \mid \texttt{let}\ t\ \texttt{be}\ x \otimes y\ \texttt{in}\ t \mid \\
& & \lambda x^A.t \mid tt \mid \\
& & \texttt{promote}\ t, ..., t\ \texttt{for}\ x_1, ..., x_n\ \texttt{in}\ t \mid \texttt{derelict}(t) \mid \\
& & \texttt{discard}\ t\ \texttt{in}\ t \mid \texttt{copy}\ t\ \texttt{as}\ x, y\ \texttt{in}\ t \mid \\
& & \texttt{true}(t, ..., t) \mid (t, t) \mid \texttt{fst}(t) \mid \texttt{snd}(t) \mid \\
& & \texttt{false}^C(t, ..., t; t) \mid \texttt{inl}^{A+B}(t) \mid \texttt{inr}^{A+B}(t) \mid \texttt{case}\ t\ \texttt{of}\ \texttt{inl}(x).t \mid \texttt{inr}(y).t
\end{array}
$$

where x is a variable ranging over terms and $t, ..., t$ denotes a list of n occurrences of the symbol t. The set of free variables, denoted $FV(u)$, of a term u is defined by induction on u as follows:

$$
\begin{array}{rcl}
FV(x) & = & \{x\} \\
FV(*) & = & \emptyset \\
FV(\texttt{let}\ w\ \texttt{be}\ *\ \texttt{in}\ u) & = & FV(w) \cup FV(u) \\
FV(u \otimes v) & = & FV(u) \cup FV(v) \\
FV(\texttt{let}\ w\ \texttt{be}\ x \otimes y\ \texttt{in}\ u) & = & FV(w) \cup (FV(u) \Leftrightarrow \{x, y\}) \\
FV(\lambda x.u) & = & FV(u) \Leftrightarrow \{x\} \\
FV(fu) & = & FV(f) \cup FV(u) \\
FV(\texttt{promote}\ v_1, ..., v_n\ \texttt{for}\ x_1, ..., x_n\ \texttt{in}\ u) & = & FV(v_1) \cup ... \cup FV(v_n) \cup (FV(u) \Leftrightarrow \{x_1, ..., x_n\}) \\
FV(\texttt{derelict}(u)) & = & FV(u) \\
FV(\texttt{discard}\ w\ \texttt{in}\ u) & = & FV(w) \cup FV(u) \\
FV(\texttt{copy}\ w\ \texttt{as}\ x, y\ \texttt{in}\ u) & = & FV(w) \cup (FV(u) \Leftrightarrow \{x, y\}) \\
FV(\texttt{true}(w_1, ..., w_n)) & = & FV(w_1) \cup ... \cup FV(w_n) \\
FV((u, v)) & = & FV(u) \cup FV(v) \\
FV(\texttt{fst}(u)) & = & FV(u) \\
FV(\texttt{false}(w_1, ..., w_n; u)) & = & FV(w_1) \cup ... \cup FV(w_n) \cup FV(u) \\
FV(\texttt{inl}(u)) & = & FV(u) \\
FV(\texttt{case}\ w\ \texttt{of}\ \texttt{inl}(x).u \mid \texttt{inr}(y).v) & = & FV(w) \cup (FV(u) \Leftrightarrow \{x\}) \cup (FV(v) \Leftrightarrow \{y\})
\end{array}
$$

We use the same convention concerning substitution as for terms of the λ-calculus: If a term v together with n terms $u_1, ..., u_n$ and n pairwise distinct variables $x_1, ..., x_n$ are given, then $v[u_1, ..., u_n/x_1, ..., x_n]$ denotes the term v where simultaneously the terms $u_1, ..., u_n$ have been substituted for free occurrences of the variables $x_1, ..., x_n$ such that bound variables in v have been renamed to avoid capture of free variables of the terms $u_1, ..., u_n$. We also need a convention concerning an "inverse" to substitution: If terms v and u are given together with a variable x, then $v[x/u]$ denotes the term v where inductively all occurrences of the term u have been replaced by the variable x.

Rules for assignment of types to terms are given in Figure 4.2. Type assignments have the form of sequents

$$x_1 : A_1, ..., x_n : A_n \Leftrightarrow u : A$$

where $x_1, ..., x_n$ are pairwise distinct variables. Note that the definition of sequents implicitly restricts use of the rules. For example, it is not possible to use the rule for introduction of $\otimes$ if the contexts , and Δ have common variables. It can be shown by induction on the derivation of the type assignment that

$$FV(u) = \{x_1, ..., x_n\}.$$

Note that this is different from the λ-calculus where we did not have equality, but only an inclusion. If we restrict to the fragment of the linear λ-calculus without additives, it can be shown by induction on the derivation of the type assignment that every free variable of the term u occurs

exactly once. Another characteristic feature of the linear λ-calculus is that we have two different "pairing" constructs; in a pair $u \otimes v$ both of the components have to be used, whereas in a pair (u, v) exactly one of the components has to be used. Via the Curry-Howard isomorphism this corresponds to the two different conjunctions of Intuitionistic Linear Logic. The linear λ-calculus satisfies the following properties:

Lemma 4.3.1 If the sequent $, \mathbb{I}\!\Rightarrow u : A$ is derivable, then for any derivable sequent $,' \mathbb{I}\!\Rightarrow u : B$, where the context $,'$ is a permutation of the context $,$, we have $A = B$.

Proof: Induction on the derivation of $, \mathbb{I}\!\Rightarrow u : A$. □

The following proposition is the essence of the Curry-Howard isomorphism:

Proposition 4.3.2 If the sequent $, \mathbb{I}\!\Rightarrow u : A$ is derivable, then the first rule instance above the sequent which is different from an instance of the *Exchange* rule is uniquely determined up to permutation of the context $,$.

Proof: Use Lemma 4.3.1 to check each case. □

Now comes a lemma dealing with substitution.

Lemma 4.3.3 (Substitution Property) If the sequents $, \mathbb{I}\!\Rightarrow u : A$ and $\Delta, x : A, \Lambda \mathbb{I}\!\Rightarrow v : B$ are derivable and the variables in the contexts $,$ and Δ, Λ are pairwise distinct, then the sequent $\Delta, , , \Lambda \mathbb{I}\!\Rightarrow v[u/x] : B$ is also derivable.

Proof: Induction on the derivation of $\Delta, x : A, \Lambda \mathbb{I}\!\Rightarrow v : B$. □

We now need a couple of conventions: The term

$$\texttt{copy}\ w_1\ \texttt{as}\ x_1, y_1\ \texttt{in}\ (...\texttt{copy}\ w_n\ \texttt{as}\ x_n, y_n\ \texttt{in}\ u...)$$

is denoted

$$\texttt{copy}\ \overline{w}\ \texttt{as}\ \overline{x}, \overline{y}\ \texttt{in}\ u$$

and the term

$$\texttt{discard}\ w_1\ \texttt{in}\ (...\texttt{discard}\ w_n\ \texttt{in}\ u...)$$

is denoted

$$\texttt{discard}\ \overline{w}\ \texttt{in}\ u.$$

The linear λ-calculus has the following β-reduction rules each of which is the image under the

Curry-Howard isomorphism of a reduction on the proof corresponding to the involved term:

$$\begin{array}{rcl}
\texttt{let}\ *\ \texttt{be}\ *\ \texttt{in}\ u & \rightsquigarrow & u \\
u \otimes v\ \texttt{be}\ x \otimes y\ \texttt{in}\ w & \rightsquigarrow & w[u, v/x, y] \\
(\lambda x.u)w & \rightsquigarrow & u[w/x] \\
\texttt{fst}((u, v)) & \rightsquigarrow & u \\
\mathrm{snd}((u, v)) & \rightsquigarrow & v \\
\texttt{case inl}(w)\ \texttt{of inl}(x).u \mid \texttt{inr}(y).v & \rightsquigarrow & u[w/x] \\
\texttt{case inr}(w)\ \texttt{of inl}(x).u \mid \texttt{inr}(y).v & \rightsquigarrow & v[w/y] \\
\texttt{derelict}(\texttt{promote}\ \overline{w}\ \texttt{for}\ \overline{x}\ \texttt{in}\ u) & \rightsquigarrow & v[\overline{w}/\overline{x}] \\
\texttt{discard}\ (\texttt{promote}\ \overline{w}\ \texttt{for}\ \overline{x}\ \texttt{in}\ u)\ \texttt{in}\ v & \rightsquigarrow & \texttt{discard}\ \overline{w}\ \texttt{in}\ v \\
\multicolumn{3}{c}{\texttt{copy}\ (\texttt{promote}\ \overline{w}\ \texttt{for}\ \overline{x}\ \texttt{in}\ u)\ \texttt{as}\ y, z\ \texttt{in}\ v} \\
\multicolumn{3}{c}{\rightsquigarrow} \\
\multicolumn{3}{c}{\texttt{copy}\ \overline{w}\ \texttt{as}\ \overline{x'}, \overline{x''}\ \texttt{in}\ (v[\texttt{promote}\ \overline{x'}\ \texttt{for}\ \overline{x}\ \texttt{in}\ u, \texttt{promote}\ \overline{x''}\ \texttt{for}\ \overline{x}\ \texttt{in}\ u/y, z])}
\end{array}$$

We shall only be concerned with β-reductions. Note how the different character of the two "pairing" constructs $u \otimes v$ and (u, v) is reflected in the corresponding reduction rules. The properties of Church-Rosser and strong normalisation for proofs of Intuitionistic Linear Logic corresponds to analogous notions for terms of the linear λ-calculus via the Curry-Howard isomorphism, and in [Bie94] it is shown that these properties are indeed satisfied by terms of the linear λ-calculus.

We will in Chapter 9 introduce the programming language LPCF which is the linear λ-calculus augmented with numerals and recursion, appropriate for the linear context, where the above mentioned reduction rules are replaced by an operational semantics.

4.4 The Curry-Howard Isomorphism

In what follows, we will consider the Natural Deduction presentation of Intuitionistic Linear Logic given in Figure 4.1. Intuitionistic Linear Logic corresponds to the linear λ-calculus via a Curry-Howard isomorphism in the same way as Intuitionistic Logic corresponds to the λ-calculus. The formulae of Intuitionistic Linear Logic are the same as the types of the linear λ-calculus. We get the rules for assigning types to terms of the linear λ-calculus if we decorate the proof-rules of Intuitionistic Linear Logic with terms in the appropriate way, and moreover, we can recover the proof-rules if we take the typing rules of the linear λ-calculus and remove the terms. The isomorphism on the level of proofs is essentially given by Proposition 4.3.2.

From a historical point of view, the choice of term corresponding to the rule for introduction of ! has been problematic. In [Abr90] the first Curry-Howard interpretation of Intuitionistic Linear Logic was published. Here the rules are given in Gentzen style, named after the discoverer of a similar system of proof-rules for Classical Logic, [Gen34, GLT89]. No Natural Deduction formulation was presented in [Abr90]. In Gentzen style we have only introduction rules; a connective can be introduced on both sides of the sequent, in opposition to Natural Deduction style where we can either introduce or eliminate a connective on the right hand side. The *Promotion* rule of

the Natural Deduction formulation corresponds to the rule

$$\frac{!A_1, ..., !A_n \mathbb{I}\!\Rightarrow A}{!A_1, ..., !A_n \mathbb{I}\!\Rightarrow !A}\ Promotion$$

of the Gentzen style formulation. Now, the Gentzen style system enjoys the substitution property simply because it is a rule of the system, namely the *Cut* rule. In [Abr90] the Gentzen style *Promotion* rule is decorated with terms as follows:

$$\frac{x_1 :!A_1, ..., x_n :!A_n \mathbb{I}\!\Rightarrow u : A}{x_1 :!A_1, ..., x_n :!A_n \mathbb{I}\!\Rightarrow !u :!A}$$

A serious problem with this term-decoration was pointed out in [Wad91]. The problem is as follows: The *Cut* rule together with the Gentzen style *Promotion* rule, decorated with terms as above, forces the categorical model corresponding to the system to collapse; the ! modality is interpreted as a functor, and the two rules together force ! to be isomorphic to !!. The problem is that a given sequent can have several derivations, and they all ought to give rise to the same categorical interpretation. The presence of the *Cut* rule gives us two different interpretations of the same sequent unless $! \cong !!$ in a canonical way.

In 1992 this was remedied by the authors of [BBdPH92b], and by the author of this thesis, by changing the decoration of the Gentzen style *Promotion* rule with terms in an appropriate way, and by discovering a Natural Deduction formulation equivalent to the Gentzen style formulation of Intuitionistic Linear Logic (the Natural Deduction formulation known at that time, [Mac91], did not possess that property). This work settled the question of how to interpret Intuitionistic Linear Logic via a Curry-Howard isomorphism. The new decoration of the *Promotion* rule goes as follows:

$$\frac{x_1 :!A_1, ..., x_n :!A_n \mathbb{I}\!\Rightarrow u : A}{z_1 :!A_1, ..., z_n :!A_n \mathbb{I}\!\Rightarrow \texttt{promote}\ z_1, ..., z_n\ \texttt{for}\ x_1, ..., x_n\ \texttt{in}\ u :!A}\ Promotion$$

The new rule can coexist with the *Cut* rule without collapsing the model, and the derivations which with the old term decoration ended with identical sequents, now end with different sequents because the induced terms are different. The new Natural Deduction formulation of Intuitionistic Linear Logic is the one given in Figure 4.1. We obtain the typing rules for the linear λ-calculus as given in Figure 4.2 by decorating the rules of the Natural Deduction formulation of Intuitionistic Linear Logic appropriately with terms. If we take the Gentzen style formulation of Intuitionistic Linear Logic and decorate it with terms as originally done in [Abr90], except that we pick the new decoration of the *Promotion* rule according to the discussion above, then we get a system equivalent to the linear λ-calculus.

As with the λ-calculus, it is the case that all the β-reductions of the linear λ-calculus appear as Curry-Howard interpretations of reduction rules of Intuitionistic Linear Logic. For example, in the case of a $(\otimes_I, \otimes_E)$ reduction we get

$$\frac{\dfrac{\vdots}{\Gamma \mathbb{I}\!\Rightarrow u : A} \quad \dfrac{\vdots}{\Delta \mathbb{I}\!\Rightarrow v : B}}{\Gamma, \Delta \mathbb{I}\!\Rightarrow u \otimes v : A \otimes B} \quad \frac{\dfrac{}{x : A \mathbb{I}\!\Rightarrow x : A} \quad \dfrac{}{y : B \mathbb{I}\!\Rightarrow y : B}}{\dfrac{\vdots}{\Lambda, x : A, y : B \mathbb{I}\!\Rightarrow w C}}}{\Lambda, \Gamma, \Delta \mathbb{I}\!\Rightarrow \texttt{let}\ u \otimes v\ \texttt{be}\ x \otimes y\ \texttt{in}\ w : C} \quad \rightsquigarrow \quad \frac{\dfrac{\vdots}{\Gamma \mathbb{I}\!\Rightarrow u : A} \quad \dfrac{\vdots}{\Delta \mathbb{I}\!\Rightarrow v : B}}{\dfrac{\vdots}{\Lambda, \Gamma, \Delta \mathbb{I}\!\Rightarrow w[u, v/x, y] : C}}$$

We see that a β-reduction has taken place on the term encoding the proof on which the reduction is performed. All β-reductions do actually appear as Curry-Howard interpretations of reductions on the corresponding proofs.

4.5 Categorical Semantics

We proceed as prescribed in [BBdPH92b] when giving a categorical semantics. In this article the notion of a linear category is introduced explicitly with the aim of giving a sound categorical interpretation of the linear λ-calculus.

Definition 4.5.1 A *categorical model for the linear λ-calculus* is a linear category with finite products and sums.

Note that we assume the presence of finite sums, not just *weak* finite sums. This does not force the categorical model to be inconsistent, that is, equivalent to the category consisting of one object and one map, when in Chapter 9 we add a linear fixpoint operator to the picture. This is contrary to the cartesian closed case where the presence of finite sums together with a fixpoint operator forces the category in question to be inconsistent.

Examples 4.5.2 In Chapter 2 it is made clear that the concrete linear categories $\mathbf{cpo}_{st}$ and $\mathbf{dI}_{lin}$ are categorical models for the linear λ-calculus.

Types are interpreted as objects, typing rules as natural operations on maps, and derivations of type assignments as maps. A derivable sequent

$$x_1 : A_1, ..., x_n : A_n \mathbb{I}\!\!\Rightarrow u : B$$

is interpreted as a map

$$[\![A_1]\!] \otimes ... \otimes [\![A_n]\!] \xrightarrow{[\![u]\!]} [\![B]\!]$$

by induction on its derivation using the appropriate operations on maps induced by the categorical model cf. below. We consider each case except the symmetric ones.

- The derivation

$$\frac{}{x : A \mathbb{I}\!\!\Rightarrow x : A}$$

  is interpreted as

$$\frac{}{A \xrightarrow{A} A}$$

- The derivation

$$\frac{, , x : A, y : B, \Delta \mathbb{I}\!\!\Rightarrow u : C}{, , y : B, x : A, \Delta \mathbb{I}\!\!\Rightarrow u : C}$$

  is interpreted as

$$\frac{, \otimes A \otimes B \otimes \Delta \xrightarrow{u} C}{, \otimes B \otimes A \otimes \Delta \cong , \otimes A \otimes B \otimes \Delta \xrightarrow{u} C}$$

- The derivation

$$\frac{}{\mathbb{I}\!\!\Rightarrow * : I}$$

  is interpreted as

$$\frac{}{I \xrightarrow{I} I}$$

- The derivation

$$\frac{\Lambda \mathbb{I}\!\!\Rightarrow w : I \quad , \mathbb{I}\!\!\Rightarrow u : A}{, , \Lambda \mathbb{I}\!\!\Rightarrow \texttt{let } w \texttt{ be } * \texttt{ in } u : A}$$

  is interpreted as

$$\frac{\Lambda \xrightarrow{w} I \quad , \xrightarrow{u} A}{, \otimes \Lambda \xrightarrow{\Gamma \otimes w} , \otimes I \cong , \xrightarrow{u} A}$$

- The derivation

$$\frac{, \Vdash u : A \qquad \Delta \Vdash v : B}{, , \Delta \Vdash u \otimes v : A \otimes B}$$

  is interpreted as

$$\frac{, \xrightarrow{u} A \qquad \Delta \xrightarrow{v} B}{, \otimes \Delta \xrightarrow{u \otimes v} A \otimes B}$$

- The derivation

$$\frac{\Lambda \Vdash w : A \otimes B \qquad , , x : A, y : B \Vdash u : C}{, , \Lambda \Vdash \texttt{let}\ w\ \texttt{be}\ x \otimes y\ \texttt{in}\ u : C}$$

  is interpreted as

$$\frac{\Lambda \xrightarrow{w} A \otimes B \qquad , \otimes A \otimes B \xrightarrow{u} C}{, \otimes \Lambda \xrightarrow{\Gamma \otimes w} , \otimes A \otimes B \xrightarrow{u} C}$$

- The derivation

$$\frac{, , x : A \Vdash u : B}{, \Vdash \lambda x.u : A \multimap B}$$

  is interpreted as

$$\frac{, \otimes A \xrightarrow{u} B}{, \xrightarrow{\lambda(u)} A \multimap B}$$

- The derivation

$$\frac{\Lambda \Vdash f : A \multimap B \qquad \Delta \Vdash u : A}{\Lambda, \Delta \Vdash fu : B}$$

  is interpreted as

$$\frac{\Lambda \xrightarrow{f} A \multimap B \qquad \Delta \xrightarrow{u} A}{\Lambda \otimes \Delta \xrightarrow{f \otimes u} (A \multimap B) \otimes A \xrightarrow{eval} B}$$

- The derivation

$$\frac{, _1 \Vdash v_1 : !A_1, \quad \ldots \quad , , _n \Vdash v_n : !A_n \qquad x_1 : !A_1, \ldots, x_n : !A_n \Vdash u : B}{, _1, \ldots, , _n \Vdash \texttt{promote}\ v_1, \ldots, v_n\ \texttt{for}\ x_1, \ldots, x_n\ \texttt{in}\ u : !B}$$

  is interpreted as

$$\frac{, _1 \xrightarrow{v_1} !A_1, \quad \ldots \quad , , _n \xrightarrow{v_n} !A_n \qquad !A_1 \otimes \ldots \otimes !A_n \xrightarrow{u} B}{, _1 \otimes \ldots \otimes , _n \xrightarrow{v_1 \otimes \ldots \otimes v_n} !A_1 \otimes \ldots \otimes !A_n \xrightarrow{\gamma(u)} !B}$$

- The derivation

$$\frac{\Lambda \Vdash u : !A}{\Lambda \Vdash \texttt{derelict}(u) : A}$$

  is interpreted as

$$\frac{\Lambda \xrightarrow{u} !A}{\Lambda \xrightarrow{u} !A \xrightarrow{\varepsilon} A}$$

- The derivation

$$\frac{\Lambda \Vdash w : !A \qquad , , x : !A, y : !A \Vdash u : B}{, , \Lambda \Vdash \texttt{copy}\ w\ \texttt{as}\ x, y\ \texttt{in}\ u : B}$$

  is interpreted as

$$\frac{\Lambda \xrightarrow{w} !A \qquad , \otimes !A \otimes !A \xrightarrow{u} B}{, \otimes \Lambda \xrightarrow{\Gamma \otimes w} , \otimes !A \xrightarrow{\Gamma \otimes d} , \otimes !A \otimes !A \xrightarrow{u} B}$$

- The derivation

$$\frac{\Lambda \mathrel{\mathbb{I}\!\!\Leftrightarrow} w : !A \quad , \mathrel{\mathbb{I}\!\!\Leftrightarrow} u : B}{, , \Lambda \mathrel{\mathbb{I}\!\!\Leftrightarrow} \texttt{discard}\ w\ \texttt{in}\ u : B}$$

  is interpreted as

$$\frac{\Lambda \xrightarrow{w} !A \quad , \xrightarrow{u} B}{, \otimes \Lambda \xrightarrow{\Gamma \otimes w} , \otimes !A \xrightarrow{\Gamma \otimes e} , \otimes I \cong , \xrightarrow{u} B}$$

- The derivation

$$\frac{, _1 \mathrel{\mathbb{I}\!\!\Leftrightarrow} w_1 : A_1, \quad \ldots \quad , , _n \mathrel{\mathbb{I}\!\!\Leftrightarrow} w_n : A_n}{, _1, \ldots, , _n \mathrel{\mathbb{I}\!\!\Leftrightarrow} \texttt{true}(w_1, \ldots, w_n) : 1}$$

  is interpreted as

$$\frac{, _1 \xrightarrow{w_1} A_1, \quad \ldots \quad , , _n \xrightarrow{w_n} A_n}{, _1, \ldots, , _n \xrightarrow{\langle\rangle} 1}$$

- The derivation

$$\frac{, \mathrel{\mathbb{I}\!\!\Leftrightarrow} u : A \quad , \mathrel{\mathbb{I}\!\!\Leftrightarrow} v : B}{, \mathrel{\mathbb{I}\!\!\Leftrightarrow} (u, v) : A \times B}$$

  is interpreted as

$$\frac{, \xrightarrow{u} A \quad , \xrightarrow{v} B}{, \xrightarrow{\langle u, v \rangle} A \times B}$$

- The derivation

$$\frac{\Lambda \mathrel{\mathbb{I}\!\!\Leftrightarrow} u : A \times B}{\Lambda \mathrel{\mathbb{I}\!\!\Leftrightarrow} \texttt{fst}(u) : A}$$

  is interpreted as

$$\frac{\Lambda \xrightarrow{u} A \times B}{\Lambda \xrightarrow{u} A \times B \xrightarrow{\pi_1} A}$$

- The derivation

$$\frac{, _1 \mathrel{\mathbb{I}\!\!\Leftrightarrow} w_1 : A_1, \quad \ldots \quad , , _n \mathrel{\mathbb{I}\!\!\Leftrightarrow} w_n : A_n \qquad \Lambda \mathrel{\mathbb{I}\!\!\Leftrightarrow} u : 0}{, _1, \ldots, , _n, \Lambda \mathrel{\mathbb{I}\!\!\Leftrightarrow} \texttt{false}^C(w_1, \ldots, w_n; u) : C}$$

  is interpreted as

$$\frac{, _1 \xrightarrow{w_1} A_1, \quad \ldots \quad , , _n \xrightarrow{w_n} A_n \qquad \Lambda \xrightarrow{u} 0}{, _1 \otimes \ldots \otimes , _n \otimes \Lambda \xrightarrow{w_1 \otimes \ldots \otimes w_n \otimes u} A_1 \otimes \ldots \otimes A_n \otimes 0 \cong 0 \xrightarrow{[]} C}$$

  where the isomorphism $A_1 \otimes \ldots \otimes A_n \otimes 0 \cong 0$ is given by the observation that the functor $, \otimes (\Leftrightarrow)$ has a right adjoint and thus preserves the initial object.

- The derivation

$$\frac{, \mathrel{\mathbb{I}\!\!\Leftrightarrow} u : A}{, \mathrel{\mathbb{I}\!\!\Leftrightarrow} \texttt{inl}(u) : A + B}$$

  is interpreted as

$$\frac{, \xrightarrow{u} A}{, \xrightarrow{u} A \xrightarrow{in_1} A + B}$$

- The derivation

$$\frac{\Lambda \mathbin{|\!\Leftrightarrow} w : A + B \qquad , , x : A \mathbin{|\!\Leftrightarrow} u : C \qquad , , y : B \mathbin{|\!\Leftrightarrow} v : C}{, , \Lambda \mathbin{|\!\Leftrightarrow} \texttt{case } w \texttt{ of inl}(x).u \mid \texttt{inr}(y).v : C}$$

  is interpreted as

$$\frac{\Lambda \xrightarrow{w} A + B \qquad , \otimes A \xrightarrow{u} C \qquad , \otimes B \xrightarrow{v} C}{, \otimes \Lambda \xrightarrow{\Gamma \otimes w} , \otimes (A + B) \cong (, \otimes A) + (, \otimes B) \xrightarrow{[u,v]} C}$$

  where the natural isomorphism $, \otimes (A + B) \cong (, \otimes A) + (, \otimes B)$ is given by the observation that the functor $, \otimes (\Leftrightarrow)$ has a right adjoint and thus preserves the binary sums.

Note that the derivation of the sequent is uniquely determined up to permutation of assumptions according to Proposition 4.3.2 so it makes sense to speak of *the* interpretation of a derivable sequent without mentioning its derivation explicitly. It can be shown that the operations on maps induced by the typing rules are natural in the interpretation of the unchanged components of the sequents. The following lemma corresponds to Lemma 4.3.3 where the categorical interpretation is taken into account; it essentially says that substitution corresponds to composition:

Lemma 4.5.3 (Substitution) If the sequents $, \mathbin{|\!\Leftrightarrow} u : A$ and $\Delta, x : A, \Lambda \mathbin{|\!\Leftrightarrow} v : B$ are derivable and the variables in the contexts $,$ and Δ, Λ are pairwise distinct, then the sequent $\Delta, , , \Lambda \mathbin{|\!\Leftrightarrow} v[u/x] : B$ is also derivable and it has the interpretation

$$\Delta \otimes , \otimes \Lambda \xrightarrow{\Delta \otimes [\![u]\!] \otimes \Lambda} \Delta \otimes A \otimes \Lambda \xrightarrow{[\![v]\!]} B$$

Proof: Induction on the derivation of $\Delta, x : A, \Lambda \mathbin{|\!\Leftrightarrow} v : B$. We use the observation that the operations on maps induced by the typing rules are natural in the interpretation of the unchanged components of the contexts of the sequents. □

By using Lemma 4.5.3 it can be shown that the interpretation is sound with respect to β-reductions - see [BBdPH92b].

4.6 The Generalised Linear λ-Calculus - Syntax

In this section we will introduce what we have called the generalised linear λ-calculus. This system was discovered for technical reasons; it enables us to prove adequacy for LPCF, and moreover, to state and prove an unwinding theorem - see Chapter 9. In certain respects the generalised linear λ-calculus is similar to the variants of Girard's Logic of Unity, [Gir93] considered in [Wad93] and [Plo93]. The extent of this similarity is to be determined by further work - see Chapter 11.

The types and terms of the generalised linear λ-calculus are the same as for the linear λ-calculus, but the rules for type assignment are more general; they have two contexts instead of one. Rules for assignment of types to terms are given in Figure 4.3; they consist of the rules of the linear λ-calculus extended with an extra context dealt with in an additive fashion, and furthermore, there is an extra axiom. Type assignments thus have the form of sequents

$$x_1 : A_1, ..., x_n : A_n ; y_1 : B_1, ..., y_m : B_m \mathbin{|\!\Leftrightarrow} u : C$$

where $x_1, ..., x_n, y_1, ..., y_m$ are pairwise distinct variables. Note how the two contexts of a sequent are separated by a semicolon. It can be shown by induction on the derivation of the type assignment that

$$FV(u) \Leftrightarrow \{x_1, ..., x_n\} = \{y_1, ..., y_m\}$$

and

$$FV(u) \Leftrightarrow \{y_1, ..., y_m\} \subseteq \{x_1, ..., x_n\}.$$

The variables occuring on the left hand side of the semicolon are called *intuitionistic* variables and the variables occuring on the right hand side of the semicolon are called *linear* variables.

Figure 4.3: Type Assignment Rules for the Generalised Linear λ-Calculus

$$\frac{}{\Sigma; x : A \Vdash x : A} \qquad \frac{}{x_1 : A_1, ..., x_n : A_n; \Vdash x_q : A_q}$$

$$\frac{\Sigma; , , x : A, y : B, \Delta \Vdash u : C}{\Sigma; , , y : B, x : A, \Delta \Vdash u : C}$$

$$\frac{}{\Sigma; \Vdash * : I} \qquad \frac{\Sigma; \Lambda \Vdash w : I \quad \Sigma; , \Vdash u : A}{\Sigma; , , \Lambda \Vdash \texttt{let } w \texttt{ be } * \texttt{ in } u : A}$$

$$\frac{\Sigma; , \Vdash u : A \quad \Sigma; \Delta \Vdash v : B}{\Sigma; , , \Delta \Vdash u \otimes v : A \otimes B} \qquad \frac{\Sigma; \Lambda \Vdash w : A \otimes B \quad \Sigma; , , x : A, y : B \Vdash u : C}{\Sigma; , , \Lambda \Vdash \texttt{let } w \texttt{ be } x \otimes y \texttt{ in } u : C}$$

$$\frac{\Sigma; , , x : A \Vdash u : B}{\Sigma; , \Vdash \lambda x^A.u : A \multimap B} \qquad \frac{\Sigma; \Lambda \Vdash f : A \multimap B \quad \Sigma; \Delta \Vdash u : A}{\Sigma; \Lambda, \Delta \Vdash fu : B}$$

$$\frac{\Sigma; , _1 \Vdash v_1 : !A_1, \ \ldots \ , \Sigma; , _n \Vdash v_n : !A_n \quad \Sigma; x_1 : !A_1, ..., x_n : !A_n \Vdash u : B}{\Sigma; , _1, ..., , _n \Vdash \texttt{promote } v_1, ..., v_n \texttt{ for } x_1, ..., x_n \texttt{ in } u : !B} \qquad \frac{\Sigma; \Lambda \Vdash u : !B}{\Sigma; \Lambda \Vdash \texttt{derelict}(u) : B}$$

$$\frac{\Sigma; \Lambda \Vdash w : !A \quad \Sigma; , , x : !A, y : !A \Vdash u : B}{\Sigma; , , \Lambda \Vdash \texttt{copy } w \texttt{ as } x, y \texttt{ in } u : B} \qquad \frac{\Sigma; \Lambda \Vdash w : !A \quad \Sigma; , \Vdash u : B}{\Sigma; , , \Lambda \Vdash \texttt{discard } w \texttt{ in } u : B}$$

$$\frac{\Sigma; , \Vdash u : A \quad \Sigma; , \Vdash v : B}{\Sigma; , \Vdash (u, v) : A \times B} \qquad \frac{\Sigma; \Lambda \Vdash u : A \times B}{\Sigma; \Lambda \Vdash \texttt{fst}(u) : A} \qquad \frac{\Sigma; \Lambda \Vdash u : A \times B}{\Sigma; \Lambda \Vdash \texttt{snd}(u) : B}$$

$$\frac{\Sigma; , _1 \Vdash v_1 : A_1, \ \ldots \ , \Sigma; , _n \Vdash v_n : A_n}{\Sigma; , _1, ..., , _n \Vdash \texttt{true}(v_1, ..., v_n) : 1} \qquad \frac{\Sigma; , _1 \Vdash v_1 : A_1, \ \ldots \ , \Sigma; , _n \Vdash v_n : A_n \quad \Sigma; \Lambda \Vdash u : 0}{\Sigma; , _1, ..., , _n, \Lambda \Vdash \texttt{false}^C(v_1, ..., v_n; u) : C}$$

$$\frac{\Sigma; , \Vdash u : A}{\Sigma; , \Vdash \texttt{inl}^{A+B}(u) : A + B} \qquad \frac{\Sigma; , \Vdash u : B}{\Sigma; , \Vdash \texttt{inr}^{A+B}(u) : A + B}$$

$$\frac{\Sigma; \Lambda \Vdash w : A + B \quad \Sigma; , , x : A \Vdash u : C \quad \Sigma; , , y : B \Vdash v : C}{\Sigma; , , \Lambda \Vdash \texttt{case } w \texttt{ of inl}(x).u \mid \texttt{inr}(y).v : C}$$

Correspondingly, the context on the left hand side of the semicolon is called the *intuitionistic* context and the context on the right hand side of the semicolon is called the *linear* context. Note that an intuitionistic variable cannot be bound. It is straightforward to check that the generalised linear λ-calculus is a conservative extension of the linear λ-calculus in the sense that a sequent $, \mathrel{▮\!\!\Rightarrow} u : A$ is derivable in the linear λ-calculus iff the sequent $;, \mathrel{▮\!\!\Rightarrow} u : A$ is derivable in the generalised linear λ-calculus.

Now, it turns out that we do not have a result dealing with the "inverse" to substitution in the linear λ-calculus analogous to Lemma 3.2.5 of the λ-calculus. The role of the intuitionistic context of the generalised linear λ-calculus is to make such a result possible. This is best explained by looking at an example: The term

$$\texttt{promote for in} *$$

(where the lists of variables and terms are empty) is typeable in the linear λ-calculus as follows:

$$\dfrac{\dfrac{}{\mathrel{▮\!\!\Rightarrow} * : I}}{\mathrel{▮\!\!\Rightarrow}\texttt{promote for in} * : !I}$$

It is also typeable in the generalised linear λ-calculus as follows:

$$\dfrac{\dfrac{}{; \mathrel{▮\!\!\Rightarrow} * : I}}{; \mathrel{▮\!\!\Rightarrow}\texttt{promote for in} * : !I}$$

But the term

$$(\texttt{promote for in} *)[x/*] = \texttt{promote for in}\, x$$

is not typeable in the linear λ-calculus. Recall that $v[x/u]$ denotes the term v where inductively all occurrences of the term u have been replaced by the variable x. The mentioned term is, however, typeable in the generalised linear λ-calculus as follows:

$$\dfrac{\dfrac{}{x : I; \mathrel{▮\!\!\Rightarrow} x : I}}{x : I; \mathrel{▮\!\!\Rightarrow}\texttt{promote for in}\, x : !I}$$

Note that the variable x is an intuitionistic variable. Such situations are taken care of by Lemma 4.6.6 below which deals with the "inverse" to substitution in the generalised linear λ-calculus. So, such a result is available for the generalised linear λ-calculus, but not for the linear λ-calculus. The generalised linear λ-calculus satisfies the following properties:

Lemma 4.6.1 If the sequent $\Sigma;, \mathrel{▮\!\!\Rightarrow} u : A$ is derivable, then for any derivable sequent $\Sigma;, ' \mathrel{▮\!\!\Rightarrow} u : B$, where the context $, '$ is a permutation of the context $,$, we have $A = B$.

Proof: Induction on the derivation of $\Sigma;, \mathrel{▮\!\!\Rightarrow} u : A$. □

The following result says that the term of a derivable sequent essentially encodes the derivation:

Proposition 4.6.2 If the sequent $\Sigma;, \mathrel{▮\!\!\Rightarrow} u : A$ is derivable, then the first rule instance above the sequent which is different from an instance of the *Exchange* rule is uniquely determined up to permutation of the context $,$.

Proof: Use Lemma 4.6.1 to check each case. □

We need a small lemma dealing with expansion of intuitionistic contexts.

Lemma 4.6.3 If the sequent $\Sigma, \Theta; \Lambda \mathrel{▮\!\!\Rightarrow} u : A$ is derivable and the variables in the contexts $\Sigma, \Theta; \Lambda$ and Φ are pairwise distinct, then the sequent $\Sigma, \Phi, \Theta; \Lambda \mathrel{▮\!\!\Rightarrow} u : A$ is also derivable.

Proof: Induction on the derivation of $\Sigma, \Theta; \Lambda \mathrel{▮\!\!\Rightarrow} u : A$. □

The Substitution Property now splits up into two cases; one for each kind of variables. The first case deals with linear variables:

Lemma 4.6.4 (Linear Substitution Property) If the sequent $\Sigma;, \not\Leftrightarrow u : A$ as well as the sequent $\Sigma;\Pi, x : A, \Lambda \not\Leftrightarrow v : B$ both are derivable and the variables in the contexts , and Π, Λ are pairwise distinct, then the sequent $\Sigma;\Pi,,,\Lambda \not\Leftrightarrow v[u/x] : B$ is also derivable.

Proof: Induction on the derivation of $\Sigma;\Pi, x : A, \Lambda \not\Leftrightarrow v : B$. We use Lemma 4.6.3. □

The second case deals with intuitionistic variables:

Lemma 4.6.5 (Intuitionistic Substitution Property) If the sequent $\Sigma; \not\Leftrightarrow u : A$ as well as the sequent $\Sigma, x : A, \Theta; \Lambda \not\Leftrightarrow v : B$ both are derivable, then the sequent $\Sigma, \Theta; \Lambda \not\Leftrightarrow v[u/x] : B$ is also derivable.

Proof: Induction on the derivation of $\Sigma, x : A, \Theta; \Lambda \not\Leftrightarrow v : B$. We need Lemma 4.6.3 for the case where the derivation is an axiom

$$\frac{}{x_1 : A_1, ..., x_n : A_n; \not\Leftrightarrow x_q : A_q}$$

such that the variable x is equal to x_q. □

Note that the linear context in the sequent $\Sigma; \not\Leftrightarrow u : A$ is empty. The following lemma deals with the "inverse" to substitution analogous to Lemma 3.2.5 of the λ-calculus. This will be necessary when in Chapter 9 we consider LPCF. Recall that $v[x/u]$ denotes the term v where inductively all occurrences of the term u have been replaced by the variable x.

Lemma 4.6.6 If the sequents $\Sigma; \not\Leftrightarrow u : A$ and $\Sigma, \Theta; \Lambda \not\Leftrightarrow v : B$ are derivable, then the sequent $\Sigma, x : A, \Theta; \Lambda \not\Leftrightarrow v[x/u] : B$ is also derivable where x is a new variable.

Proof: Induction on the derivation of $\Sigma, \Theta; \Lambda \not\Leftrightarrow v : B$. □

4.7 The Generalised Linear λ-Calculus - Semantics

Given a categorical model for the linear λ-calculus we are able to interpret the generalised linear λ-calculus: Types are interpreted as objects, typing rules as natural operations on maps, and derivations of type assignments as maps. A derivable sequent

$$x_1 : A_1, ..., x_n : A_n; y_1 : B_1, ..., y_m : B_m \not\Leftrightarrow u : C$$

is interpreted as a map

$$!\llbracket A_1 \rrbracket \otimes ... \otimes !\llbracket A_n \rrbracket \otimes \llbracket B_1 \rrbracket \otimes ... \otimes \llbracket B_m \rrbracket \xrightarrow{\llbracket u \rrbracket} \llbracket C \rrbracket$$

by induction on its derivation using appropriate operations on maps induced by the categorical model. Note that $!A_1 \otimes ... \otimes !A_n$ is the underlying object of the coalgebra $(!A_1, \delta) \otimes ... \otimes (!A_n, \delta)$; this gives us projection maps

$$(!A_1, \delta) \otimes ... \otimes (!A_n, \delta) \xrightarrow{\pi_q} (!A_q, \delta)$$

together with a diagonal map

$$(!A_1, \delta) \otimes ... \otimes (!A_n, \delta) \xrightarrow{\Delta} (!A_1, \delta) \otimes ... \otimes (!A_n, \delta) \otimes (!A_1, \delta) \otimes ... \otimes (!A_n, \delta)$$

and a map

$$(!A_1, \delta) \otimes ... \otimes (!A_n, \delta) \xrightarrow{\langle\rangle} I$$

to the terminal object all living in the category of coalgebras. We use these maps to give operations on maps corresponding to the typing rules of the generalised linear λ-calculus. Only two cases are considered, the operations on maps corresponding to the other typing rules are straightforward extensions of the operations on maps induced by the typing rules for the linear λ-calculus where we make use of the above mentioned diagonal map together with the map to the terminal object.

- The derivation

$$\frac{}{x_1 : A_1, ..., x_n : A_n ; \mathbb{I}\!\!\Rightarrow x_q : A_q}$$

is interpreted as

$$\frac{}{!A_1 \otimes ... \otimes !A_n \xrightarrow{\pi_q} !A_q \xrightarrow{\varepsilon} A_q}$$

where $\pi_q : !A_1 \otimes ... \otimes !A_n \rightarrow !A_q$ is the projection map in the category of coalgebras.

- The derivation

$$\frac{\Sigma; \, , \, _1 \mathbb{I}\!\!\Rightarrow v_1 : !A_1, \quad ... \quad , \Sigma; \, , \, _n \mathbb{I}\!\!\Rightarrow v_n : !A_n \qquad \Sigma; x_1 : !A_1, ..., x_n : !A_n \mathbb{I}\!\!\Rightarrow u : B}{\Sigma; \, , \, _1, ..., \, , \, _n \mathbb{I}\!\!\Rightarrow \mathtt{promote}\ v_1, ..., v_n\ \mathtt{for}\ x_1, ..., x_n\ \mathtt{in}\ u : !B}$$

is interpreted as

$$\frac{\Sigma \otimes \, , \, _1 \xrightarrow{v_1} !A_1, \quad ... \quad , \Sigma \otimes \, , \, _n \xrightarrow{v_n} !A_n \qquad \Sigma \otimes !A_1 \otimes ... \otimes !A_n \xrightarrow{u} B}{\Sigma \otimes \, , \, _1 \otimes ... \otimes \, , \, _n \xrightarrow{l} \Sigma \otimes \Sigma \otimes \, , \, _1 \otimes ... \otimes \Sigma \otimes \, , \, _n \xrightarrow{\Sigma \otimes v_1 \otimes ... \otimes v_n} \Sigma \otimes !A_1 \otimes ... \otimes !A_n \xrightarrow{\gamma(u)} !B}$$

using the map l defined as the composition

$$\Sigma \otimes \, , \, _1 \otimes ... \otimes \, , \, _n \xrightarrow{\Delta \otimes \Gamma_1 \otimes ... \otimes \Gamma_n} \Sigma \otimes \Sigma \otimes ... \otimes \Sigma \otimes \, , \, _1 \otimes ... \otimes \, , \, _n \cong \Sigma \otimes \Sigma \otimes \, , \, _1 \otimes ... \otimes \Sigma \otimes \, , \, _n$$

where $\Delta : \Sigma \rightarrow \Sigma \otimes \Sigma \otimes ... \otimes \Sigma$ is the diagonal map in the category of coalgebras. Note that the use of the generalised Kleisli operator, γ, is possible because a formula C in the intuitionistic context Σ is interpreted as $![\![C]\!]$.

It can be shown that the operations on maps induced by the typing rules are natural in the interpretation of the unchanged components of the linear contexts of the sequents, and moreover, they are natural in the interpretation of the unchanged components of the intuitionistic contexts of the sequents with respect to maps of coalgebras. Note that the categorical interpretation of the generalised linear λ-calculus is a conservative extension of the categorical interpretation of the linear λ-calculus in the sense that the interpretation of a linear λ-calculus sequent $\, , \, \mathbb{I}\!\!\Rightarrow u : A$ coincides with the interpretation of the sequent $; \, , \, \mathbb{I}\!\!\Rightarrow u : A$ of the generalised linear λ-calculus. The following lemma corresponds to Lemma 4.6.3 where the categorical interpretation has been taken into account:

Lemma 4.7.1 If the sequent $\Sigma, \Theta; \Lambda \mathbb{I}\!\!\Rightarrow u : A$ is derivable and the variables in the contexts $\Sigma, \Theta; \Lambda$ and Φ are pairwise distinct, then the sequent $\Sigma, \Phi, \Theta; \Lambda \mathbb{I}\!\!\Rightarrow u : A$ is also derivable and it has the interpretation

$$\Sigma \otimes \Phi \otimes \Theta \otimes \Lambda \xrightarrow{\Sigma \otimes \langle\rangle \otimes \Theta \otimes \Lambda} \Sigma \otimes I \otimes \Theta \otimes \Lambda \cong \Sigma \otimes \Theta \otimes \Lambda \xrightarrow{[\![u]\!]} A$$

Proof: Induction on the derivation of $\Sigma, \Theta; \Lambda \mathbb{I}\!\!\Rightarrow u : A$. We use the observation that the operations on maps induced by the typing rules of the generalised linear λ-calculus are natural in the interpretation of the unchanged components of the intuitionistic contexts of the sequents with respect to maps of coalgebras. □

The following lemma corresponds to Lemma 4.6.4 where the categorical interpretation has been taken into account; it essentially says that substitution with respect to linear variables corresponds to composition:

Lemma 4.7.2 (Linear Substitution) If the sequents $\Sigma; \, , \, \mathbb{I}\!\!\Rightarrow u : A$ and $\Sigma; \Pi, x : A, \Lambda \mathbb{I}\!\!\Rightarrow v : B$ are derivable and the variables in the contexts $\, , \, $ and Π, Λ are pairwise distinct, then the sequent $\Sigma; \Pi, \, , \, , \Lambda \mathbb{I}\!\!\Rightarrow v[u/x] : B$ is also derivable and it has the interpretation

$$\Sigma \otimes \Pi \otimes \, , \, \otimes \Lambda \xrightarrow{\Delta \otimes \Pi \otimes \Gamma \otimes \Lambda} \Sigma \otimes \Sigma \otimes \Pi \otimes \, , \, \otimes \Lambda \cong \Sigma \otimes \Pi \otimes \Sigma \otimes \, , \, \otimes \Lambda \xrightarrow{\Sigma \otimes \Pi \otimes [\![u]\!] \otimes \Lambda} \Sigma \otimes \Pi \otimes A \otimes \Lambda \xrightarrow{[\![v]\!]} B$$

Proof: Induction on the derivation of $\Sigma; \Pi, x : A, \Lambda \mathbin{|\!\Leftrightarrow} v : B$. We use naturality of the appropriate operations on maps induced by the typing rules of the generalised linear λ-calculus. □

The following lemma corresponds to Lemma 4.6.5 where the categorical interpretation has been taken into account; it essentially says that substitution with respect to intuitionistic variables corresponds to composition in the category of coalgebras:

Lemma 4.7.3 (Intuitionistic Substitution) If the sequents $\Sigma; \mathbin{|\!\Leftrightarrow} u : A$ and $\Sigma, x : A, \Theta; \Lambda \mathbin{|\!\Leftrightarrow} v : B$ are derivable, then the sequent $\Sigma, \Theta; \Lambda \mathbin{|\!\Leftrightarrow} v[u/x] : B$ is also derivable and it has the interpretation

$$\Sigma \otimes \Theta \otimes \Lambda \xrightarrow{\Delta \otimes \Theta \otimes \Lambda} \Sigma \otimes \Sigma \otimes \Theta \otimes \Lambda \xrightarrow{\Sigma \otimes \gamma(\llbracket u \rrbracket) \otimes \Theta \otimes \Lambda} \Sigma \otimes !A \otimes \Theta \otimes \Lambda \xrightarrow{\llbracket v \rrbracket} B$$

Proof: Induction on the derivation of $\Sigma, x : A, \Theta; \Lambda \mathbin{|\!\Leftrightarrow} v : B$. We use the observation that the operations on maps induced by the typing rules of the generalised linear λ-calculus are natural in the interpretation of the unchanged components of the intuitionistic contexts of the sequents with respect to maps of coalgebras. □

Chapter 5

The Girard Translation

This chapter introduces the Girard Translation which embeds Intuitionistic Logic into Intuitionistic Linear Logic. The syntactic matters are dealt with in Section 5.1, and in Section 5.2 the Girard Translation is shown to be sound with respect to the categorical interpretations induced by an appropriate categorical model.

5.1 Syntax

The [Gir87] paper introduced the Girard Translation which embeds Intuitionistic Logic into Intuitionistic Linear Logic. We will state the Girard Translation in terms of the Natural Deduction presentations of Intuitionistic Logic and Intuitionistic Linear Logic given in Figure 3.1 and Figure 4.1, respectively. The translation works at the level of formulae as well as at the level of proofs. At the level of formulae the Girard Translation is defined inductively as follows:

$$\begin{array}{rcl} 1^o & = & 1 \\ (A \times B)^o & = & A^o \times B^o \\ (A \Rightarrow B)^o & = & !A^o \multimap B^o \\ 0^o & = & 0 \\ (A + B)^o & = & !A^o + !B^o \end{array}$$

At the level of proofs the Girard Translation inductively translates a proof of a sequent

$$A_1, ..., A_n \vdash B$$

into a proof of the sequent

$$!A_1^o, ..., !A_n^o \mathrel{|\!\!\Rightarrow} B^o .$$

In what follows we consider each case except the symmetric ones. Special cases of rules are used when appropriate, and a double bar means a number of applications of rules.

- A derivation

$$\frac{}{A_1, ..., A_n \vdash A_p}$$

  is translated into

$$\cfrac{\cfrac{\cfrac{}{!A_q^o \mathrel{|\!\!\Rightarrow} !A_p^o}}{!A_q^o \mathrel{|\!\!\Rightarrow} A_p^o}}{!A_1^o, ..., !A_n^o \mathrel{|\!\!\Rightarrow} A_p^o}$$

- A derivation

$$\frac{}{, \vdash 1}$$

is translated into

$$\frac{}{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft 1}$$

- A derivation

$$\frac{,\ \vdash A \qquad ,\ \vdash B}{,\ \vdash A \times B}$$

is translated into

$$\frac{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft A^o \qquad !,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft B^o}{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft A^o \times B^o}$$

- A derivation

$$\frac{,\ \vdash A \times B}{,\ \vdash A}$$

is translated into

$$\frac{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft A^o \times B^o}{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft A^o}$$

- A derivation

$$\frac{,,A \vdash B}{,\ \vdash A \Rightarrow B}$$

is translated into

$$\frac{!,\ ^o, !A^o\!\!\blacktriangleright\!\!\blacktriangleleft B^o}{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft !A^o \multimap B^o}$$

- A derivation

$$\frac{,\ \vdash A \Rightarrow B \qquad ,\ \vdash A}{,\ \vdash B}$$

is translated into

$$\frac{\dfrac{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft !A^o \multimap B^o \qquad \dfrac{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft A^o}{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft !A^o}}{!,\ ^o, !,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft B^o}}{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft B^o}$$

- A derivation

$$\frac{,\ \vdash 0}{,\ \vdash C}$$

is translated into

$$\frac{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft 0}{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft C^o}$$

- A derivation

$$\frac{,\ \vdash A}{,\ \vdash A + B}$$

is translated into

$$\frac{\dfrac{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft A^o}{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft !A^o}}{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft !A^o + !B^o}$$

- A derivation

$$\frac{,\ \vdash A + B \qquad ,,A \vdash C \qquad ,,B \vdash C}{,\ \vdash C}$$

is translated into

$$\frac{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft !A^o + !B^o \qquad !,\ ^o, !A^o\!\!\blacktriangleright\!\!\blacktriangleleft C^o \qquad !,\ ^o, !B^o\!\!\blacktriangleright\!\!\blacktriangleleft C^o}{\dfrac{!,\ ^o, !,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft C^o}{!,\ ^o\!\!\blacktriangleright\!\!\blacktriangleleft C^o}}$$

The translation is sound with respect to provability in the sense that $A_1, ..., A_n \vdash B$ is provable (in Intuitionistic Logic) iff $!A_1^o, ..., !A_n^o \multimap B^o$ is provable (in Intuitionistic Linear Logic). Moreover, it is shown in [Bie94] that the translation preserves β-reductions. Now, the Girard Translation induces a translation from types and derivable sequents in the λ-calculus to types and derivable sequents in the linear λ-calculus via the appropriate Curry-Howard isomorphisms. A type A is translated into a type A^o and a sequent

$$x_1 : A_1, ..., x_n : A_n \vdash u : B$$

is translated into a sequent

$$x_1 :!A_1^o, ..., x_n :!A_n^o \multimap u^o : B^o$$

where the term u^o encodes the translation of the proof encoded by the term u. We obtain an inductive definition of the translation from derivable sequents in the λ-calculus to derivable sequents in the linear λ-calculus by decorating the rules of the Girard Translation at the level of proofs appropriately with terms. In what follows we consider each case except the symmetric ones. If , denotes a context $A_1, ..., A_n$ and $\overline{x}$ denotes a list of n pairwise distinct variables, then $\overline{x} :$, denotes the context $x_1 : A_1, ..., x_n : A_n$.

- A derivation

$$\frac{}{x_1 : A_1, ..., x_n : A_n \vdash x_p : A_p}$$

  is translated into

$$\frac{\dfrac{\dfrac{}{x_p :!A_q^o \multimap x_p :!A_p^o}}{x_p :!A_q^o \multimap \mathtt{derelict}(x_p) : A_p^o}}{x_1 :!A_1^o, ..., x_n :!A_n^o \multimap \mathtt{discard}\, x_1, ..., x_{p-1}, x_{p+1}, ..., x_n\ \mathtt{in}\ (\mathtt{derelict}(x_p)) : A_p^o}$$

- A derivation

$$\frac{}{\overline{x} : , \vdash \mathtt{true} : 1}$$

  is translated into

$$\frac{}{\overline{x} :!, {}^o \multimap \mathtt{true}(\overline{x}) : 1}$$

- A derivation

$$\frac{\overline{x} : , \vdash u : A \qquad \overline{x} : , \vdash v : B}{\overline{x} : , \vdash (u, v) : A \times B}$$

  is translated into

$$\frac{\overline{x} :!, {}^o \multimap u^o : A^o \qquad \overline{x} :!, {}^o \multimap v^o : B^o}{\overline{x} :!, {}^o \multimap (u^o, v^o) : A^o \times B^o}$$

- A derivation

$$\frac{\overline{x} : , \vdash u : A \times B}{\overline{x} : , \vdash \mathtt{fst}(u) : A}$$

  is translated into

$$\frac{\overline{x} :!, {}^o \multimap u^o : A^o \times B^o}{\overline{x} :!, {}^o \multimap \mathtt{fst}(u^o) : A^o}$$

- A derivation

$$\frac{\overline{x} : , , x : A \vdash u : B}{\overline{x} : , \vdash \lambda x.u : A \Rightarrow B}$$

  is translated into

$$\frac{\overline{x} :!, {}^o, x :!A^o \multimap u^o : B^o}{\overline{x} :!, {}^o \multimap \lambda x.u^o :!A^o \multimap B^o}$$

- A derivation

$$\frac{\overline{x} : , \vdash f : A \Rightarrow B \qquad \overline{x} : , \vdash u : A}{\overline{x} : , \vdash fu : B}$$

is translated into

$$\frac{\dfrac{\overline{x'} :!, {}^{o} \mathrel{\blacktriangleright\!\!\blacktriangleleft} f^{o} :!A^{o} \multimap B^{o} \qquad \dfrac{\overline{x''} :!, {}^{o} \mathrel{\blacktriangleright\!\!\blacktriangleleft} u^{o} : A^{o}}{\overline{x''} :!, {}^{o} \mathrel{\blacktriangleright\!\!\blacktriangleleft} \mathtt{promote}\ \overline{x''}\ \mathtt{for}\ \overline{x''}\ \mathtt{in}\ u^{o} :!A^{o}}}{\overline{x'} :!, {}^{o}, \overline{x''} :!, {}^{o} \mathrel{\blacktriangleright\!\!\blacktriangleleft} f^{o}\, \mathtt{promote}\ \overline{x''}\ \mathtt{for}\ \overline{x''}\ \mathtt{in}\ u^{o} : B^{o}}}{\overline{x} :!, {}^{o} \mathrel{\blacktriangleright\!\!\blacktriangleleft} \mathtt{copy}\ \overline{x}\ \mathtt{as}\ \overline{x'}, \overline{x''}\ \mathtt{in}\ (f^{o}\, \mathtt{promote}\ \overline{x''}\ \mathtt{for}\ \overline{x''}\ \mathtt{in}\ u^{o}) : B^{o}}$$

- A derivation

$$\frac{\overline{x} : , \vdash w : 0}{\overline{x} : , \vdash \mathtt{false}(w) : C}$$

is translated into

$$\frac{\overline{x} :!, {}^{o} \mathrel{\blacktriangleright\!\!\blacktriangleleft} w^{o} : 0}{\overline{x} :!, {}^{o} \mathrel{\blacktriangleright\!\!\blacktriangleleft} \mathtt{false}(; w^{o}) : C^{o}}$$

- A derivation

$$\frac{\overline{x} : , \vdash u : A}{\overline{x} : , \vdash \mathtt{inl}(u) : A + B}$$

is translated into

$$\frac{\dfrac{\overline{x} :!, {}^{o} \mathrel{\blacktriangleright\!\!\blacktriangleleft} u^{o} : A^{o}}{\overline{x} :!, {}^{o} \mathrel{\blacktriangleright\!\!\blacktriangleleft} \mathtt{promote}\ \overline{x}\ \mathtt{for}\ \overline{x}\ \mathtt{in}\ u^{o} :!A^{o}}}{\overline{x} :!, {}^{o} \mathrel{\blacktriangleright\!\!\blacktriangleleft} \mathtt{inl}(\mathtt{promote}\ \overline{x}\ \mathtt{for}\ \overline{x}\ \mathtt{in}\ u^{o}) :!A^{o} + !B^{o}}$$

- A derivation

$$\frac{\overline{x} : , \vdash w : A + B \qquad \overline{x} : , , y : A \vdash u : C \qquad \overline{x} : , , z : B \vdash v : C}{\overline{x} : , \vdash \mathtt{case}\ w\ \mathtt{of}\ \mathtt{inl}(y).u \mid \mathtt{inr}(z).v : C}$$

is translated into

$$\frac{\overline{x'} :!, {}^{o} \mathrel{\blacktriangleright\!\!\blacktriangleleft} w^{o} :!A^{o} + !B^{o} \qquad \overline{x''} :!, {}^{o}, y :!A^{o} \mathrel{\blacktriangleright\!\!\blacktriangleleft} u^{o} : C^{o} \qquad \overline{x''} :!, {}^{o}, z :!B^{o} \mathrel{\blacktriangleright\!\!\blacktriangleleft} v^{o} : C^{o}}{\dfrac{\overline{x'} :!, {}^{o}, \overline{x''} :!, {}^{o} \mathrel{\blacktriangleright\!\!\blacktriangleleft} \mathtt{case}\ w^{o}\ \mathtt{of}\ \mathtt{inl}(y).u^{o} \mid \mathtt{inr}(z).v^{o} : C^{o}}{\overline{x} :!, {}^{o} \mathrel{\blacktriangleright\!\!\blacktriangleleft} \mathtt{copy}\ \overline{x}\ \mathtt{as}\ \overline{x'}, \overline{x''}\ \mathtt{in}\ (\mathtt{case}\ w^{o}\ \mathtt{of}\ \mathtt{inl}(y).u^{o} \mid \mathtt{inr}(z).v^{o}) : C^{o}}}$$

The translation from types and derivable sequents in the λ-calculus to types and derivable sequents in the linear λ-calculus induced by the Girard Translation will be extended to a translation from PCF to LPCF in Chapter 10.

5.2 Soundness

Let $\mathcal{C}$ be a categorical model for the linear λ-calculus, that is, a linear category with finite products and finite sums; we can then interpret types and derivable sequents of the linear λ-calculus as objects and maps in $\mathcal{C}$. The Kleisli category induced by the ! comonad is cartesian closed with weak finite sums such that the diagram of Definition 3.4.1 commutes according to Proposition 2.4.2 and Proposition 2.4.3. It is therefore a categorical model for the λ-calculus, so furthermore we can interpret types and derivable sequents of the λ-calculus as objects and maps in the Kleisli category. It turns out that the interpretation of a type can be written in a simple way using the Girard Translation at the level of types:

Proposition 5.2.1 Let $\mathcal{C}$ be a categorical model for the linear λ-calculus. If a type A of the λ-calculus is interpreted in the Kleisli category, and the type A^o of the linear λ-calculus is interpreted in $\mathcal{C}$, then $[\![A]\!] = [\![A^o]\!]$.

Proof: Induction on the structure of A. □

Before showing that the Girard Translation is sound with respect to the above mentioned categorical interpretations we need a convention; we define *lin* to be the composition of bijections between maps

$$\begin{array}{rcll} \mathcal{C}_!([\![A_1]\!] \times ... \times [\![A_n]\!], [\![B]\!]) & = & \mathcal{C}_!([\![A_1^o]\!] \times ... \times [\![A_n^o]\!], [\![B^o]\!]) & \text{by Proposition 5.2.1} \\ & = & \mathcal{C}(!([\![A_1^o]\!] \times ... \times [\![A_n^o]\!]), [\![B^o]\!]) & \text{because } U_! \dashv F_! \\ & \cong & \mathcal{C}(!([\![A_1^o]\!] \otimes ... \otimes ![\![A_n^o]\!]), [\![B^o]\!]) & \text{by composing with } n \end{array}$$

where $A_1, ..., A_n$ and B are types of the λ-calculus. The soundness result essentially says that the Girard Translation corresponds to the adjunction between the Kleisli category and $\mathcal{C}$, or to be precise, to the function *lin*. It is a categorical generalisation of a result in [Gir87] which shows that the Girard Translation is sound with respect to interpretation in a certain concrete category, namely the category of coherence spaces and linear stable functions. Recently categorical soundness results in the sense of ours, but for somewhat different calculi, have been given in [BW96]. Recall that a derivable sequent

$$x_1 : A_1, ..., x_n : A_n \vdash t : B$$

of the λ-calculus is translated into the derivable sequent

$$x_1 : !A_1^o, ..., x_n : !A_n^o \vdash t^o : B^o$$

of the linear λ-calculus, and observe that the maps $[\![t]\!]$ and $[\![t^o]\!]$ live in the domain and the codomain of of the function *lin*, respectively.

Theorem 5.2.2 (Soundness) Let $\mathcal{C}$ be a categorical model for the linear λ-calculus. If the sequent

$$x_1 : A_1, ..., x_n : A_n \vdash t : B$$

is derivable in the λ-calculus, then $lin([\![t]\!]) = [\![t^o]\!]$.

Proof: Induction on the derivation of the sequent $x_1 : A_1, ..., x_n : A_n \vdash t : B$. We proceed case by case. Symmetric cases are omitted.

- In the case

$$\frac{}{x_1 : A_1, ..., x_n : A_n \vdash x_p : A_p}$$

the following calculation suffices:

$$\begin{array}{rcll} lin([\![x_p]\!]) & = & n ; \pi_p & \\ & = & n ; !\pi_p ; \varepsilon & \text{by def. of } \pi_p \text{ in } \mathcal{C}_! \\ & = & \pi_p ; \varepsilon & \text{Note 1.} \\ & = & (e \otimes ... \otimes e \otimes id \otimes e \otimes ... \otimes e) ; \cong ; \varepsilon & \text{Note 2.} \\ & = & [\![\texttt{discard}\ x_1, ..., x_{p-1}, x_{p+1}, ..., x_n\ \texttt{in}\ (\texttt{derelict}(x_p))]\!] & \\ & = & [\![x_p^o]\!] & \end{array}$$

Note 1. We obtain the projection map π_p in $\mathcal{C}^!$ by composing the map n with $!\pi_p$ according to Section 2.3.
Note 2. By definition of π_p in $\mathcal{C}^!$ according to the discussion in Section 2.3.

- In the case

$$\frac{}{\overline{x} : , \vdash \texttt{true} : 1}$$

the following calculation suffices:

$$\begin{array}{rcll} lin([\![\texttt{true}]\!]) & = & [\![\texttt{true}(\overline{x})]\!] & \text{as 1 is terminal} \\ & = & [\![\texttt{true}^o]\!] & \end{array}$$

- In the case

$$\frac{\overline{x} : , \vdash u : A \qquad \overline{x} : , \vdash v : B}{\overline{x} : , \vdash (u, v) : A \times B}$$

the following calculation suffices:

$$\begin{array}{rcll} lin([\![(u,v)]\!]) & = & n;\langle [\![u]\!], [\![v]\!]\rangle & \\ & = & \langle (n;[\![u]\!]), (n;[\![v]\!])\rangle & \text{by def. of } \langle \Leftrightarrow, + \rangle \texttt{ in } \mathcal{C} \\ & = & \langle lin([\![u]\!]), lin([\![v]\!])\rangle & \\ & = & \langle [\![u^o]\!], [\![v^o]\!]\rangle & \text{by ind. hyp.} \\ & = & [\![(u^o, v^o)]\!] & \\ & = & [\![(u,v)^o]\!] & \end{array}$$

- In the case

$$\frac{\overline{x} : , \vdash u : A \times B}{\overline{x} : , \vdash \texttt{fst}(u) : A}$$

the following calculation suffices:

$$\begin{array}{rcll} lin([\![\texttt{fst}(u)]\!]) & = & n;\gamma([\![u]\!]);\pi_1 & \\ & = & n;[\![u]\!];\pi_1 & \text{by def. of } \pi_1 \texttt{ in } \mathcal{C} \\ & = & lin([\![u]\!]);\pi_1 & \\ & = & [\![u^o]\!];\pi_1 & \text{by ind. hyp.} \\ & = & [\![\texttt{fst}(u^o)]\!] & \\ & = & [\![\texttt{fst}(u)^o]\!] & \end{array}$$

- In the case

$$\frac{\overline{x} : , , x : A \vdash u : B}{\overline{x} : , \vdash \lambda x.u : A \Rightarrow B}$$

the following calculation suffices:

$$\begin{array}{rcll} lin([\![\lambda x.u]\!]) & = & n;\lambda([\![u]\!]) & \\ & = & n;\lambda(n;[\![u]\!]) & \text{by def. of } \lambda \texttt{ in } \mathcal{C} \\ & = & \lambda((n \otimes id);n;[\![u]\!]) & \\ & = & \lambda(n;[\![u]\!]) & \text{Note 1.} \\ & = & \lambda(lin([\![u]\!])) & \\ & = & \lambda([\![u^o]\!]) & \text{by ind. hyp.} \\ & = & [\![\lambda x.u^o]\!] & \\ & = & [\![(\lambda x.u)^o]\!] & \end{array}$$

Note 1. Because n makes ! a monoidal functor from $(\mathcal{C}, 1, \times)$ to $(\mathcal{C}, I, \otimes)$.

- In the case

$$\frac{\overline{x} : , \vdash f : A \Rightarrow B \qquad \overline{x} : , \vdash u : A}{\overline{x} : , \vdash fu : B}$$

the following calculation suffices:

$$\begin{array}{rcll} lin([\![fu]\!]) & = & n;\gamma(\langle [\![f]\!], [\![u]\!]\rangle);eval & \\ & = & \gamma(\langle (n;[\![f]\!]), (n;[\![u]\!])\rangle);n^{-1};(\varepsilon \otimes id);eval & \text{def. of op. in } \mathcal{C} \\ & = & \gamma(\langle lin([\![f]\!]), lin([\![u]\!])\rangle);n^{-1};(\varepsilon \otimes id);eval & \\ & = & \gamma(\langle [\![f^o]\!], [\![u^o]\!]\rangle);n^{-1};(\varepsilon \otimes id);eval & \text{by ind. hyp.} \\ & = & \langle \gamma([\![f^o]\!]), \gamma([\![u^o]\!])\rangle;(\varepsilon \otimes id);eval & \text{Note 1.} \\ & = & D;(\gamma([\![f^o]\!]) \otimes \gamma([\![u^o]\!]));(\varepsilon \otimes id);eval & \text{Note 2.} \\ & = & D;([\![f^o]\!] \otimes \gamma([\![u^o]\!]));eval & \\ & = & [\![\texttt{copy } \overline{x} \texttt{ as } \overline{x'}, \overline{x''} \texttt{ in } (f^o \texttt{ promote } \overline{x''} \texttt{ for } \overline{x''} \texttt{ in } u^o)]\!] & \\ & = & [\![(fu)^o]\!] & \end{array}$$

Note 1. We obtain a map $\langle f, g\rangle$ in $\mathcal{C}^!$ by composing the map $\gamma(\langle\gamma^{-1}(f), \gamma^{-1}(g)\rangle)$ with n^{-1} according to Section 2.3.
Note 2. By definition of $\langle \Leftrightarrow, + \rangle$ in $\mathcal{C}^!$ according to the discussion in Section 2.3; recall that D is the diagonal map.

- In the case

$$\frac{\overline{x} : , \ \vdash w : 0}{\overline{x} : , \ \vdash \mathtt{false}(w) : C}$$

the following calculation suffices:

$$\begin{array}{rcll} lin([\![\mathtt{false}(w)]\!]) & = & n; \gamma([\![w]\!]); [] & \\ & = & n; [\![w]\!]; [] & \text{by def. of } [] \text{ in } \mathcal{C}^! \\ & = & lin([\![w]\!]); [] & \\ & = & [\![w^o]\!]; [] & \text{by ind. hyp.} \\ & = & [\![\mathtt{false}(; w^o)]\!] & \\ & = & [\![\mathtt{false}(w)^o]\!] & \end{array}$$

- In the case

$$\frac{\overline{x} : , \ \vdash u : A}{\overline{x} : , \ \vdash \mathtt{inl}(u) : A + B}$$

the following calculation suffices:

$$\begin{array}{rcll} lin([\![\mathtt{inl}(u)]\!]) & = & n; \gamma([\![u]\!]); in_1 & \\ & = & \gamma(n; [\![u]\!]); in_1 & \text{by def. of } in_1 \text{ in } \mathcal{C}^! \\ & = & \gamma(lin([\![u]\!])); in_1 & \\ & = & \gamma([\![u^o]\!]); in_1 & \text{by ind. hyp.} \\ & = & [\![\mathtt{inl}(\mathtt{promote}\ \overline{x}\ \mathtt{for}\ \overline{x}\ \mathtt{in}\ u^o)]\!] & \\ & = & [\![\mathtt{inl}(u)^o]\!] & \end{array}$$

- In the case

$$\frac{\overline{x} : , \ \vdash w : A + B \quad \overline{x} : , , y : A \vdash u : C \quad \overline{x} : , , z : B \vdash v : C}{\overline{x} : , \ \vdash \mathtt{case}\ w\ \mathtt{of}\ \mathtt{inl}(y).u \mid \mathtt{inr}(z).v : C}$$

the following calculation suffices:

$$\begin{array}{rcll} & lin([\![\mathtt{case}\ w\ \mathtt{of}\ \mathtt{inl}(y).u \mid \mathtt{inr}(z).v]\!]) & & \\ = & n; \gamma(\langle [\![w]\!], id\rangle); \lambda^{-1}([\lambda(\gamma(\cong); [\![u]\!]), \lambda(\gamma(\cong); [\![v]\!])]) & & \\ = & \gamma(\langle (n; [\![w]\!]), (n; \varepsilon)\rangle); n^{-1}; \lambda^{-1}(\varepsilon; [\lambda(n; ! \cong; [\![u]\!]), \lambda(n; ! \cong; [\![v]\!])]) & & \text{def. of op. in } \mathcal{C}^! \\ = & \langle \gamma(n; [\![w]\!]), n\rangle; \lambda^{-1}(\varepsilon; [\lambda(n; ! \cong; [\![u]\!]), \lambda(n; ! \cong; [\![v]\!])]) & & \text{Note 1.} \\ = & D; ((n; [\![w]\!]) \otimes n); \lambda^{-1}([\lambda(n; ! \cong; [\![u]\!]), \lambda(n; ! \cong; [\![v]\!])]) & & \text{Note 2.} \\ = & D; ((n; [\![w]\!]) \otimes id); \lambda^{-1}([\lambda((id \otimes n); n; ! \cong; [\![u]\!]), \lambda((id \otimes n); n; ! \cong; [\![v]\!])]) & & \\ = & D; ((n; [\![w]\!]) \otimes id); \lambda^{-1}([\lambda(\cong; n; [\![u]\!]), \lambda(\cong; n; [\![v]\!])]) & & \text{Note 3.} \\ = & D; (lin([\![w]\!]) \otimes id); \lambda^{-1}([\lambda(\cong; lin([\![u]\!])), \lambda(\cong; lin([\![v]\!]))]) & & \\ = & D; ([\![w^o]\!] \otimes id); \lambda^{-1}([\lambda(\cong; [\![u^o]\!]), \lambda(\cong; [\![v^o]\!])]) & & \text{by ind. hyp.} \\ = & D; (id \otimes [\![w^o]\!]); \cong; [[\![u^o]\!], [\![v^o]\!]] & & \\ = & [\![\mathtt{copy}\ \overline{x}\ \mathtt{as}\ \overline{x'}, \overline{x''}\ \mathtt{in}\ (\mathtt{case}\ w^o\ \mathtt{of}\ \mathtt{inl}(y).u^o \mid \mathtt{inr}(z).v^o)]\!] & & \\ = & [\![(\mathtt{case}\ w\ \mathtt{of}\ \mathtt{inl}(y).u \mid \mathtt{inr}(z).v)^o]\!] & & \end{array}$$

Note 1. We obtain a map $\langle f, g\rangle$ in $\mathcal{C}^!$ by composing the map $\gamma(\langle\gamma^{-1}(f), \gamma^{-1}(g)\rangle)$ with n^{-1} according to Section 2.3.
Note 2. By definition of $\langle \Leftrightarrow, + \rangle$ in $\mathcal{C}^!$ according to the discussion in Section 2.3; recall that D is the diagonal map.
Note 3. Because n makes ! a symmetric monoidal functor from $(\mathcal{C}, 1, \times)$ to $(\mathcal{C}, I, \otimes)$.

□

Chapter 6

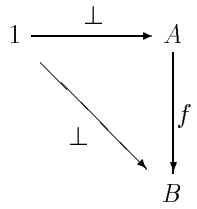
PCF - Semantic Issues

In this chapter appropriate machinery for giving the categorical model for PCF is introduced. The introductions of categorical constructs are intertwined with examples of such in concrete categories. Section 6.1 introduces a categorical notion of undefinedness. Categorical notions of numerals and fixpoints are given in Section 6.2 and Section 6.3, respectively. Also the axiom of rational openness on a fixpoint operator is introduced. In Section 6.4 the observational preorder is introduced and in Section 6.5 it is shown how under certain circumstances a rationally open fixpoint operator induces a rational category when the quotients of the observational preorders on hom-sets are considered.

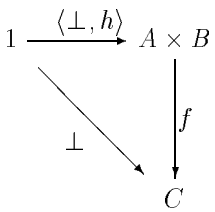
6.1 Undefinedness

We will introduce a notion of undefinedness by assuming for each object A the existence of an "undefined" map $\bot_A\colon 1 \to A$ which is supposed to be the interpretation of arbitrary non-terminating programs of type A. So we assume that our interpretation identifies all non-terminating computations.

Definition 6.1.1 A category with a terminal object is *pointed* iff a map $\bot_A\colon 1 \to A$ is given for each object A. A map $f : A \to B$ in a pointed category is *strict* iff the diagram



commutes. A map $f : A \times B \to C$ in a pointed cartesian category is *left-strict* iff the diagram



commutes for any map $h : 1 \to B$. Right-strictness is defined analogously.

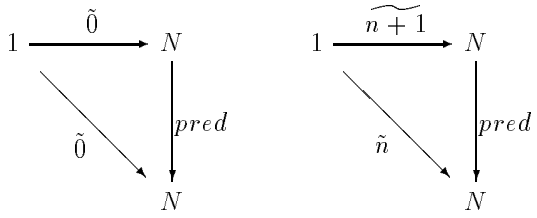
Examples 6.1.2 In the categories **cpo** and **dI** the obvious choice of $\bot$-maps is the appropriate bottom elements.

Observation: Strictness of all maps does not go well together with cartesian closure; it can simply be shown that a pointed cartesian closed category where all maps are strict is inconsistent, that is, it is equivalent to the category consisting of one object and one map. This problem vanishes when we replace cartesian closure by monoidal closure.

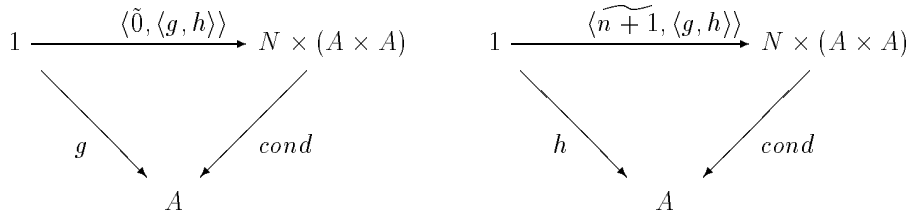
6.2 Numerals

The following definition is essentially as given in [HO96]. Note that booleans are represented by numerals.

Definition 6.2.1 An *object of numerals* in a cartesian category is an object N equipped with maps $zero : 1 \to N$ and $succ, pred : N \to N$ such that the diagrams



commute for any number n where the maps $\tilde{n} : 1 \to N$ are defined in the obvious way using the *zero* and *succ* maps. Furthermore, a map $cond_A : N \times (A \times A) \to A$ is given for each object A such that the diagrams



commute for any maps $g, h : 1 \to A$ and any number n. An object of numerals in a pointed cartesian category is *standard* iff for any $h : 1 \to N$ we have $h = \perp$ or $h = \tilde{n}$ for some number n.

Note that in a cartesian category $\mathcal{C}$ with an object of numerals such that $\tilde{0} = \tilde{1}$ every hom-set $\mathcal{C}(1, A)$ has at most one element. A notable feature of an object of numerals in a pointed cartesian category is that for each number n it is possible to define a map $\phi_n : N \to N$ with the property that $\tilde{n}; \phi_n = \tilde{0}$ and $\tilde{p}; \phi_n = \perp$ whenever $p \neq n$. The ϕ_n maps are defined by the stipulations

$$\begin{array}{rcl} \phi_0 & = & \langle id, (\langle\rangle; \langle \tilde{0}, \perp \rangle) \rangle; cond \\ \phi_{n+1} & = & \langle id, \langle (\langle\rangle; \perp), (pred; \phi_n) \rangle \rangle; cond \end{array}$$

If the map *cond* is left-strict then each map ϕ_n has the property of being strict.

Examples 6.2.2 In what follows we will give a couple of objects of numerals in the concrete categories **cpo** and **dI**; note that the finite products of **dI** coincide with the ones of **cpo**. The obvious choice of a standard object of numerals in these categories is to take the object N to be equal to ω with a bottom element adjoined, and equipped with the maps

$$zero(\perp) = 0$$

$$succ(x) = \begin{cases} \perp & \text{if } x = \perp \\ x + 1 & \text{otherwise} \end{cases} \qquad pred(x) = \begin{cases} \perp & \text{if } x = \perp \\ 0 & \text{if } x = 0 \\ x \Leftrightarrow 1 & \text{otherwise} \end{cases}$$

$$cond_A(x, (y, z)) = \begin{cases} \perp & \text{if } x = \perp \\ y & \text{if } x = 0 \\ z & \text{otherwise} \end{cases}$$

This is a generalisation of an option considered in [Plo77] where only one $cond_A$ map is considered, namely the one where the object A is equal to N.

In [Plo77] another option is suggested; the object N is taken to be equal to ω with a bottom element adjoined together with an element ∞ unrelated to any other element but bottom, and it is equipped with the maps from the standard object of numerals extended as follows

$$succ(\infty) = \infty \qquad pred(\infty) = \infty \qquad cond_A(\infty, (y, z)) = z$$

where we have generalised the original version of *cond* as appropriate. This gives a non-standard object of numerals in **cpo** as well as in **dI**.

Yet another option is considered in [Plo77]; the object N is taken to be equal to ω with a bottom element adjoined together with a top element $\top$, and it is equipped with the maps from the standard object of numerals extended as follows

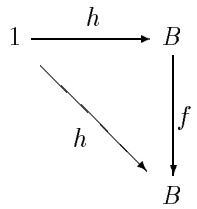
$$succ(\top) = \top \qquad pred(\top) = \top \qquad cond(\top, (y, z)) = \top$$

where we consider only the original version of *cond*. It is actually the case that the *cond* map as defined here cannot be generalised to fit into our notion of object of numerals because $cond(\top, y, z)$ whenever x and y are incompatible cannot be defined such that the function *cond* is monotone. So it works in **cpo** if one is content with the original version of *cond*, but if one goes to the category **dI** it does not work anyway because the function is not stable; this can be seen by considering the compatible elements $(0, (1, 1))$ and $(1, (1, 1))$ from the domain of *cond*.

6.3 Fixpoints

The main concern of this section is fixpoints as introduced in [Law69].

Definition 6.3.1 Let $\mathcal{C}$ be a category with a terminal object. A map $h : 1 \to B$ is a *fixpoint* of a map $f : B \to B$ iff the diagram



commutes. A *fixpoint operator* for an object B is an operation on maps

$$(\Leftrightarrow)^\sharp_B : \mathcal{C}(B, B) \Leftrightarrow\!\!\!\rightarrow \mathcal{C}(1, B)$$

such that $f^\sharp$ is a fixpoint of f for any map $f : B \to B$.

We now need a convention: Given a map $g : B \to B$ we define a map $g^n : B \to B$ for every number n by stipulating $g^0 = id$ and $g^{n+1} = g^n; g$. So the map g^n is simply n iterations of g.

Examples 6.3.2 In the categories **cpo** and **dI** the obvious choice of fixpoint operators is the usual one that defines the fixpoint $f^\sharp$ of a map $f : B \to B$ as

$$f^\sharp = \bigsqcup_{n \in \omega} f^n(\bot).$$

Definition 6.3.3 A fixpoint operator for an object B in a pointed category is *rationally open* with respect to an object P iff for all maps $f : B \to B$ and $g : B \to P$ it is the case that

$$f^\sharp; g \neq \bot \;\Rightarrow\; \exists n \in \omega.\ \bot; f^n; g \neq \bot\ .$$

Note that if a fixpoint operator for an object P is rationally open with respect to P, then $id_P^\sharp = \bot_P$ as can be seen by taking f and g in the definition to be equal to id_P.

Examples 6.3.4 Consider a map $g : B \to P$ living **cpo** or in **dI**. In the category **cpo** the set $\{x \in B \mid g(x) \neq \bot\}$ is a Scott-open subset of B, and in the category **dI**, it is a union of sets of the form $\{x \in B \mid x \geq b\}$ where b is a finite element of B, so it is straightforward to see that the fixpoint operators in the categories **cpo** and **dI** are rationally open with respect to any object.

Under appropriate circumstances rational openness with respect to an object of numerals N can be restated to a different form.

Proposition 6.3.5 Let $\mathcal{C}$ be a pointed cartesian category equipped with a standard object of numerals N such that the map *cond* is left strict. Assume that a fixpoint operator is given for an object B. Then the fixpoint operator is rationally open with respect to N iff for all maps $f : B \to B$ and $g : B \to N$ and numbers p it is the case that

$$f^\sharp ; g = \tilde{p} \;\; \Rightarrow \;\; \exists n \in \omega .\; \bot ; f^n ; g = \tilde{p}.$$

Proof: A straightforward application of the ϕ_p maps defined in the remark following Definition 6.2.1. □

The role of rational openness in the adequacy result for PCF in Section 7.1 is analogous to the role of the *absoluteness* condition in [Fio94b]. Note, however, that our notion of rational openness does not assume the presence of any order-structure; this is not the case with the notion of absoluteness.

Definition 6.3.6 Let $\mathcal{C}$ be a cartesian category. A map $h : X \to B$ is a *parametrised fixpoint* of a map $f : X \times B \to B$ iff the diagram

$$\begin{array}{ccc} X & \xrightarrow{\Delta} & X \times X \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle id \times h} \\ B & \xleftarrow{f} & X \times B \end{array}$$

commutes. A *parametrised fixpoint operator* for an object B is an operation on maps

$$(\Leftrightarrow)^\sharp_{X,B} : \mathcal{C}(X \times B, B) \Leftrightarrow \mathcal{C}(X, B)$$

for each object X such that $f^\sharp$ is a parametrised fixpoint of f for any map $f : X \times B \to B$. A parametrised fixpoint operator is *natural* if the operations are natural in X.

Note how the diagonal map $\Delta_X : X \to X \times X$ is used to copy parameters. We can internalise the notion of a fixpoint operator when we deal with an exponentiable object in a cartesian category.

Definition 6.3.7 An *internal fixpoint operator* for an exponentiable object B in a cartesian category is a map $Y_B : B \Rightarrow B \to B$ such that the diagram

$$\begin{array}{ccc} B \Rightarrow B & \xrightarrow{\Delta} & (B \Rightarrow B) \times (B \Rightarrow B) \\ {\scriptstyle Y}\downarrow & & \downarrow{\scriptstyle id \times Y} \\ B & \xleftarrow{eval} & (B \Rightarrow B) \times B \end{array}$$

commutes.

In [Poi92] it is shown that a cartesian closed category has a parametrised fixpoint operator for each object iff it has an internal fixpoint operator for each object but here we will show a more informative result:

Proposition 6.3.8 Let B be an exponentiable object in a cartesian category. There is a bijective correspondence between natural parametrised fixpoint operators and internal fixpoint operators for the object B.

Proof: Assume that the category has a natural parametrised fixpoint operator $(\Leftrightarrow)^\sharp$ for an object B, and define the map Y by the equation

$$Y = eval^\sharp.$$

This is clearly an internal fixpoint operator for B cf. the diagram in Definition 6.3.6. Conversely, assume that Y is an internal fixpoint operator for the object B, and define an operation on maps

$$(\Leftrightarrow)^\sharp = \lambda(\Leftrightarrow); Y$$

for each object X. This is a parametrised fixpoint operator because

$$\begin{array}{rcl} \lambda(f); Y & = & \lambda(f); \Delta; (id \times Y); eval \\ & = & \Delta; (id \times (\lambda(f); Y)); f \end{array}$$

for any map $f : X \times B \to B$. Naturality of the operation follows from naturality of λ.

The construction of internal fixpoint operators from natural parametrised fixpoint operators is injective because we have

$$\begin{array}{rcl} \lambda(f); eval^\sharp & = & ((\lambda(f) \times id); eval)^\sharp \\ & = & f^\sharp \end{array}$$

for any map $f : X \times B \to B$, which entails that the natural parametrised fixpoint operator induced by the internal fixpoint operator $eval^\sharp$ is equal to $(\Leftrightarrow)^\sharp$. The construction of natural parametrised fixpoint operators from internal fixpoint operators is injective because the following calculation

$$\begin{array}{rcl} \lambda(eval); Y & = & id; Y \\ & = & Y \end{array}$$

shows that the internal fixpoint operator induced by the natural parametrised fixpoint operator $\lambda(\Leftrightarrow); Y$ is equal to Y. □

Now a remark on notation: Assume that B is an exponentiable object in a monoidal category. Given a map $f : A \to B$, the map $\lambda(\cong; f) : I \to A \multimap B$ will be denoted $\ulcorner f \urcorner$. It is straightforward to check that we actually have a bijection between maps

$$\ulcorner \Leftrightarrow \urcorner : \mathcal{C}(A, B) \cong \mathcal{C}(I, A \multimap B).$$

Proposition 6.3.9 Let B be an exponentiable object in a cartesian category. A fixpoint operator for the object $(B \Rightarrow B) \Rightarrow B$ induces an internal fixpoint operator for B.

Proof: Firstly, define a map

$$(B \Rightarrow B) \Rightarrow B \xrightarrow{H_B} (B \Rightarrow B) \Rightarrow B$$

as the exponential transpose of the map given in Figure 6.1. For every map $h : B \Rightarrow B \to B$ the diagram

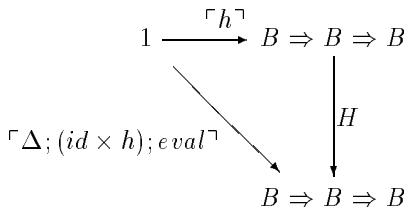
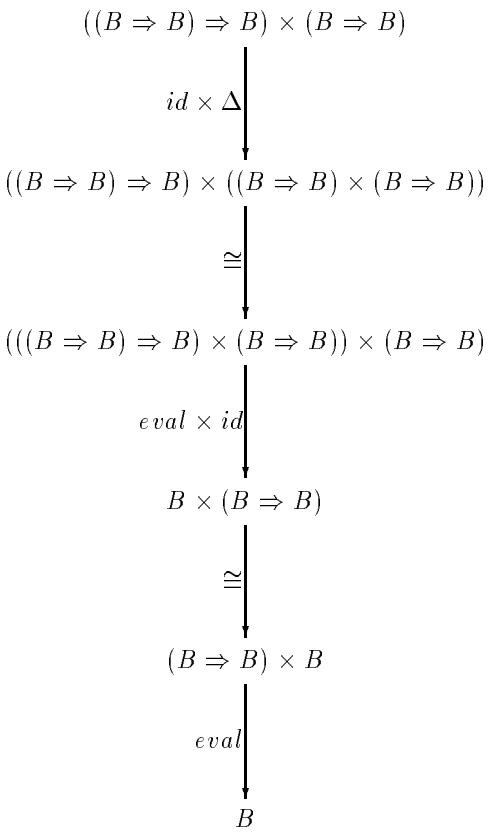


Figure 6.1: The map used in Proposition 6.3.9



commutes, which can be shown by some equational manipulation; this entails that $\ulcorner h \urcorner$ is a fixpoint for H iff $h = \Delta; (id \times h); eval$. We now define the internal fixpoint operator $Y : B \Rightarrow B \to B$ by the equation $\ulcorner Y \urcorner = H^\sharp$; hence $Y = \Delta; (id \times Y); eval$. □

This entails that if every object in a cartesian closed category has a fixpoint operator, then every object has an internal fixpoint operator too. Note that in the context of Proposition 6.3.9 the map $\ulcorner f \urcorner; Y$ is a fixpoint for f for every map $f : B \to B$, and, moreover, if the category in question is pointed such that the map $eval$ is left-strict, then it can be shown by a small induction proof that

$$\langle (\bot; H^n), \ulcorner f \urcorner \rangle; eval = \bot; f^n.$$

So indeed the map H does work as expected.

6.4 The Observational Preorder

The goal of this section is to introduce the observational preorder. First, we need a couple of notions from enriched category theory: A **preorder**-enriched (**poset**-enriched, **cpo**-enriched) category $\mathcal{C}$ is a category where each hom-set $\mathcal{C}(A, B)$ is equipped with a preorder (poset, cpo) $\lesssim_{A,B}$ such that each composition function

$$(\Leftrightarrow); (+) : \mathcal{C}(A, B) \times \mathcal{C}(B, C) \Leftrightarrow\!\!\!\to \mathcal{C}(A, C)$$

is monotone (monotone, continuous). A **preorder**-enriched (**poset**-enriched, **cpo**-enriched) functor between **preorder**-enriched (**poset**-enriched, **cpo**-enriched) categories $\mathcal{C}$ and $\mathcal{D}$ is a functor

between the underlying categories such that each function

$$F : \mathcal{C}(A, B) \Leftrightarrow\!\!\!\rightarrow \mathcal{D}(FA, FB)$$

is monotone (monotone, continuous). A **preorder**-enriched category $\mathcal{C}$ induces a **poset**-enriched category $\widehat{\mathcal{C}}$ by taking the objects of $\widehat{\mathcal{C}}$ to be the objects of $\mathcal{C}$ and by taking for each pair of objects A and B the poset $\widehat{\mathcal{C}}(A, B)$ to be the quotient of the preorder $\mathcal{C}(A, B)$. The **poset**-enriched category $\widehat{\mathcal{C}}$ is called the *quotient* of $\mathcal{C}$. This is sufficient enriched category theory for our purpose; an introduction to the topic can be found in [Poi92]. The definitions and results of the remaining part of this section are essentially as given in [HO96]. To define the observational preorder we need the notion of observables.

Definition 6.4.1 Let $(\mathcal{C}, I, \otimes, \multimap)$ be a symmetric monoidal closed category. A *notion of observables* $\mathcal{O}$ associates to each object A a set $\mathcal{O}_A$ of subsets of $\mathcal{C}(I, A)$ with the property that if $S \in \mathcal{O}_A$ then $g^*S \in \mathcal{O}_B$ for any map $g : B \to A$ where $g^*S = \{h \in \mathcal{C}(I, B) \mid h; g \in S\}$.

Definition 6.4.2 Let $\mathcal{C}$ be a symmetric monoidal closed category with a notion of observables $\mathcal{O}$. The *observational preorder* $\lesssim_{A,B}$ on each hom-set $\mathcal{C}(A, B)$ is defined as

$$f \lesssim g \quad \text{iff} \quad \forall R \in \mathcal{O}_{A \multimap B}.\ \ulcorner f \urcorner \in R \ \Rightarrow\ \ulcorner g \urcorner \in R$$

for any maps $f, g : A \to B$.

The following proposition states that the observational preorder behaves properly with respect to points, that is, maps with domain I.

Proposition 6.4.3 If $\mathcal{C}$ is a symmetric monoidal closed category with a notion of observables $\mathcal{O}$, then

$$f \lesssim g \quad \text{iff} \quad \forall R \in \mathcal{O}_A.\ f \in R \ \Rightarrow\ g \in R$$

for any maps $f, g : I \to A$.

Proof: The two maps

$$A \xrightarrow{\lambda(\cong)} I \multimap A$$

and

$$I \multimap A \cong (I \multimap A) \otimes I \xrightarrow{eval} A$$

make the objects A and $I \multimap A$ isomorphic. The result now follows from the observation that $f = \ulcorner f \urcorner; \cong; eval$ for any map $f : I \to A$. □

The following proposition states that all the structure involved enriches with respect to the observational preorder.

Proposition 6.4.4 Let $\mathcal{C}$ be a symmetric monoidal closed category with a notion of observables and consider the observational preorder. The symmetric monoidal closed category is **preorder**-enriched. If the category is actually cartesian closed then it is **preorder**-enriched as such.

Proof: Let a map $h : B \to C$ be given. The map

$$A \multimap B \xrightarrow{A \multimap h} A \multimap C$$

has the property that $\ulcorner f \urcorner; (A \multimap h) = \ulcorner f; h \urcorner$ for any map $f : A \to B$, and analogously, the map

$$C \multimap A \xrightarrow{h \multimap A} B \multimap A$$

has the property that $\ulcorner g \urcorner; (h \multimap A) = \ulcorner h; g \urcorner$ for any map $g : C \to A$, which shows that composition on either side with h preserves the observational preorder. Consider the map

$$(A \multimap B) \otimes (A \otimes D) \cong ((A \multimap B) \otimes A) \otimes D \xrightarrow{eval \otimes D} B \otimes D$$

the transpose of which

$$A \multimap B \xrightarrow{\lambda(\cong;(eval\otimes D))} (A \otimes D) \multimap (B \otimes D)$$

has the property that $\ulcorner f \urcorner ; \lambda(\cong; (eval \otimes D)) = \ulcorner f \otimes D \urcorner$ for any map $f : A \to B$, which shows that the $\otimes$ functor preserves the preorder. We will now show that the function between maps

$$\mathcal{C}(A \otimes B, C) \xrightarrow{\lambda} \mathcal{C}(A, B \multimap C)$$

preserves the preorder. Consider the map

$$(((A \otimes B) \multimap C) \otimes A) \otimes B \cong ((A \otimes B) \multimap C) \otimes (A \otimes B) \xrightarrow{eval} C$$

the double transpose of which

$$(A \otimes B) \multimap C \xrightarrow{\lambda(\lambda(\cong;eval))} A \multimap (B \multimap C)$$

has the property that $\ulcorner f \urcorner ; \lambda(\lambda(\cong; eval)) = \ulcorner \lambda(f) \urcorner$ for any map $f : A \otimes B \to C$, which shows that λ preserves the preorder. It is now straightforward to check that the function λ^{-1} as well as the $\multimap$ functor preserve the preorder because these constructs can be defined in terms of operations already shown to be preorder preserving. Thus we have shown that the symmetric monoidal closed structure is **preorder**-enriched.

If the category is actually cartesian closed then we have to show that it is **preorder**-enriched as such, that is, we furthermore have to show that the function between maps

$$\mathcal{C}(A, C) \times \mathcal{C}(A, D) \xrightarrow{\langle -, + \rangle} \mathcal{C}(A, C \times D)$$

preserves the preorder. But this follows from the observation that $\langle f, g \rangle = \Delta ; (f \times g)$ for any maps $f : A \to C$ and $g : A \to D$ because we have already shown that composition and the $\times$ functor are preorder preserving. □

6.5 Rationality

In this section we will show that a rationally open fixpoint operator under appropriate circumstances induces a rational category when the quotients of the observational preorders on hom-sets are considered. The definition of a rational category given here is weaker than the original one introduced in [AJM96]; the essential difference is that we do not assume the *eval* map to be left-strict because this condition is irrelevant to the order-theoretic considerations of this section. In that article a rational category is taken to be an appropriate (order-theoretic) notion of an adequate categorical model for a certain fragment of PCF.

Definition 6.5.1 A *rational category* is a pointed cartesian closed category which is **poset**-enriched as a cartesian closed category such that for every object A it is the case that

- the map $\bot_A$ is least,
- for every map $f : A \to A$ the increasing chain $\{\bot; f^n\}_{n\in\omega}$ has a least upper bound $f^\sharp$ with the property that for any map $g : A \to D$ the map $f^\sharp ; g$ is a least upper bound for the increasing chain $\{\bot; f^n; g\}_{n\in\omega}$.

Examples 6.5.2 The categories **cpo** and **dI** are rational in the obvious way.

We will now consider a notion of observables for a pointed cartesian closed category that assumes the presence of a distinguished object P. The object P should be thought of as the interpretation of a distinguished type. In [HO96] the following notion of observables is called the termination notion of observables when a map $f : 1 \to P$ is taken to be a *value* iff $f \neq \bot$. Recall that a cartesian closed category is a special kind of symmetric monoidal closed category.

Definition 6.5.3 Let $\mathcal{C}$ be a pointed cartesian closed category with a distinguished object P. The *termination notion of observables* is defined by assigning the set

$$\mathcal{O}_A = \{\mathcal{O}_h \mid h \in \mathcal{C}(A, P)\}$$

to any object A where

$$\mathcal{O}_h = \{f \in \mathcal{C}(1, A) \mid f; h \neq \bot\}.$$

This is a notion of observables because $g^* \mathcal{O}_h = \mathcal{O}_{g;h}$ for any maps $g : B \to A$ and $h : A \to P$. The induced observational preorder can be stated explicitly as

$$f \lesssim g \quad \text{iff} \quad \forall h \in \mathcal{C}(A \Rightarrow B, P).\ \ulcorner f \urcorner; h \neq \bot \ \Rightarrow\ \ulcorner g \urcorner; h \neq \bot$$

for any maps $f, g : A \to B$. We will refer to this preorder as the *termination preorder*.

Note that the map $\bot_P$ is minimal with respect to the termination preorder; this is so because if $f \lesssim \bot$ for some map $f : 1 \to P$, then $\bot; id = \bot$ entails that $f; id = \bot$, that is, $f = \bot$. It follows from Proposition 6.4.4 that $\mathcal{C}$ is **preorder**-enriched (as a cartesian closed category) with respect to the termination preorder. If $\leq$ is another **preorder**-enrichment on $\mathcal{C}$ (as a cartesian closed category) with respect to which $\bot_P$ is minimal, then $\leq$ can be shown to be included in $\lesssim$ in the sense that for any objects A and B the preorder $\leq_{A,B}$ on $\mathcal{C}(A, B)$ is included in the $\lesssim_{A,B}$ preorder. If we are dealing with a pointed cartesian closed category equipped with an object of numerals, then we take the object P to be N unless otherwise is stated.

Examples 6.5.4 In the concrete categories **cpo** and **dI** the termination preorder coincides with the given order.

Now comes the results saying that under appropriate circumstances a rationally open fixpoint operator induces a rational category.

Theorem 6.5.5 Let $\mathcal{C}$ be a pointed cartesian closed category equipped with a standard object of numerals such that the map *cond* is left-strict. Assume that a fixpoint operator is given for each object. Then for any object A it is the case that

- the map $\bot_A$ is least and for any maps $f : A \to A$ and $g : A \to D$ the map $f^\sharp; g$ is an upper bound for the increasing chain $\{\bot; f^n; g\}_{n \in \omega}$,
- the fixpoint operator for A is rationally open with respect to N iff for any maps $f : A \to A$ and $g : A \to D$ the map $f^\sharp; g$ is a least upper bound for the increasing chain $\{\bot; f^n; g\}_{n \in \omega}$,

where we consider the termination preorder.

Proof: Without loss of generality we will assume that $\tilde{0} \neq \bot$ because $\tilde{0} = \bot$ entails that every hom-set $\mathcal{C}(1, A)$ has exactly one element.

We first show that the map $\bot_A$ is least[1]. We will use the $\phi_n : N \to N$ maps defined in the remark following Definition 6.2.1. Recall that each map ϕ_n has the property of being strict, and moreover, $\tilde{n}; \phi_n = \tilde{0}$ and $\tilde{p}; \phi_n = \bot$ whenever $p \neq n$. Take any map $h : 1 \to A$. We will then show that $\bot \lesssim h$. Assume $\bot; f \neq \bot$ for some map $f : A \to N$, that is, $\bot; f = \tilde{n}$ for some number n; we then have to show that $h; f \neq \bot$. Now, define a map $k : N \to A$ as

$$k = \langle id, (\langle\rangle; \langle h, \bot\rangle)\rangle; cond$$

and consider the fixpoint $(k; f; \phi_n)^\sharp : 1 \to N$ of the composition

$$N \xrightarrow{k} A \xrightarrow{f} N \xrightarrow{\phi_n} N$$

[1] Here we essentially make use of an argument suggested by Alex Simpson while the author visited Edinburgh in March '96.

which has the property that

$$(k; f; \phi_n)^\sharp = (k; f; \phi_n)^\sharp; k; f; \phi_n.$$

We cannot have $(k; f; \phi_n)^\sharp = \bot$ because this contradicts $\bot; f; \phi_n = \tilde{0}$, so we have $(k; f; \phi_n)^\sharp = \tilde{0}$ which entails that $h; f; \phi_n = \tilde{0}$. We conclude that $h; f = \tilde{n}$, and thus $h; f \neq \bot$.

For any maps $f : A \to A$ and $g : A \to D$ it is straightforward to check by induction that the chain $\{\bot; f^n\}_{n\in\omega}$ is increasing and $f^\sharp$ is an upper bound, so the chain $\{\bot; f^n; g\}_{n\in\omega}$ is increasing and $f^\sharp; g$ is an upper bound.

Assume that the fixpoint operator for the object A is rationally open with respect to N, and consider any maps $f : A \to A$ and $g : A \to D$. Let $k : 1 \to D$ be an arbitrary upper bound for the increasing chain $\{\bot; f^n; g\}_{n\in\omega}$. If $f^\sharp; g; h \neq \bot$ for some map $h : D \to N$ then there exists a number p such that $\bot; f^p; g; h \neq \bot$ because the fixpoint operator is assumed to be rationally open with respect to N, which entails that $k; h \neq \bot$. We conclude that $f^\sharp; g \lesssim k$.

Consider any maps $f : A \to A$ and $g : A \to N$, and assume that the map $f^\sharp; g$ is a least upper bound for the increasing chain $\{\bot; f^n; g\}_{n\in\omega}$. Then $\bot; f^n; g = \bot$ for every number n entails that $\bot$ is an upper bound for the increasing chain $\{\bot; f^n; g\}_{n\in\omega}$, but $f^\sharp; g$ is a least upper bound, so $f^\sharp; g \lesssim \bot$ and thus $f^\sharp; g = \bot$. We conclude that the fixpoint operator is rationally open with respect to N. □

Note that in the context of Theorem 6.5.5 none of the maps in the hom-set $\mathcal{C}(1, N)$ are equivalent with respect to the equivalence relation induced by the termination preorder.

Corollary 6.5.6 Let $\mathcal{C}$ be a pointed cartesian closed category equipped with a standard object of numerals such that the map *cond* is left-strict. For each object assume that a fixpoint operator which is rationally open with respect to N is given. Then the quotient $\widehat{\mathcal{C}}$ is a rational category when $\mathcal{C}$ is considered as **preorder**-enriched with respect to the termination preorder.

Proof: Follows from Theorem 6.5.5 and the observation that the fixpoint operator is a congruence with respect to the equivalence relation induced by the termination preorder. □

The condition in Theorem 6.5.5 of the object of numerals being standard cannot be left out, as the following example shows:

Examples 6.5.7 Consider the category where the objects are pairs (D, d) such that D is a cpo of which d is an element, and where a map from (D, d) to (D', d') is a continuous function from D to D'. The object $(\{\bot\}, \bot)$ is terminal and the category is pointed when each object (D, d) is equipped with the constantly d function from $\{\bot\}$ to D. Note that d might be different from the bottom element $\bot$ of D. This category inherits the cartesian closed structure from **cpo**; given objects (D, d) and (D', d') the product is defined to be $(D \times D', (d, d'))$ and the exponential object is defined to be $D \Rightarrow D'$ together with the constantly d' function from D to D'. We define a non-standard object of numerals (N, ∞) by taking N to be equal to ω with a bottom element adjoined together with an element ∞ unrelated to any other element but bottom, and it is equipped with the maps from the standard object of numerals from Section 6.2 extended as follows

$$succ(\infty) = \infty \qquad pred(\infty) = \infty \qquad cond_{(D,d)}(\infty, y, z) = d$$

This gives a non-standard object of numerals such that the map *cond* is left-strict, and moreover, a fixpoint operator for each object is inherited from **cpo**. Now, the termination preorder coincides with the dual to the poset on maps inherited from **cpo**, in which the constantly ∞ function from $\{\bot\}$ to N is not least. There is no least element, in fact. Note that the fixpoint operator is not rationally open with respect to this object of numerals as $id_N^\sharp = \bot_N$ but $\bot_N \neq \infty$ cf. the remark following Definition 6.3.3.

Above we have considered a notion of observables that gave rise to the termination preorder. We will now have a look at another notion of observables for a cartesian closed category that assumes the presence of an object of numerals but does not assume the category in question to be pointed; in [HO96] it is called the termination to value notion of observables when a map $f : 1 \to N$ is taken to be a value iff $f = \tilde{n}$ for some number n.

Definition 6.5.8 Let $\mathcal{C}$ be a cartesian closed category equipped with an object of numerals. The *termination to value notion of observables* is defined by assigning the set

$$\mathcal{O}_A = \{\mathcal{O}_{h,n} \mid h \in \mathcal{C}(A, N), n \in \omega\}$$

to any object A where

$$\mathcal{O}_{h,n} = \{f \in \mathcal{C}(1, A) \mid f; h = \tilde{n}\}.$$

This is a notion of observables because $g^*\mathcal{O}_{h,n} = \mathcal{O}_{(g;h),n}$ for any maps $g : B \to A$ and $h : A \to N$. The induced observational preorder can be stated explicitly as

$$f \lesssim g \quad \text{iff} \quad \forall h \in \mathcal{C}(A \Rightarrow B, N).\forall n \in \omega.\ \ulcorner f \urcorner; h = \tilde{n} \ \Rightarrow\ \ulcorner g \urcorner; h = \tilde{n}$$

for any maps $f, g : A \to B$. We will refer to this preorder as the *termination to value preorder*.

Note that for any number p the map $\tilde{p}$ is maximal with respect to the termination to value preorder; this is so because if $\tilde{p} \lesssim f$ for some map $f : 1 \to N$, then $\tilde{p}; id = \tilde{p}$ entails that $f; id = \tilde{p}$, that is, $f = \tilde{p}$. It follows from Proposition 6.4.4 that $\mathcal{C}$ is **preorder**-enriched (as a cartesian closed category) with respect to the termination to value preorder. If $\leq$ is another **preorder**-enrichment on $\mathcal{C}$ (as a cartesian closed category) with respect to which $\tilde{p}$ is maximal for every number p, then $\leq$ can be shown to be included in $\lesssim$.

Examples 6.5.9 In the concrete categories **cpo** and **dI** the termination to value preorder coincides with the given order.

If the category giving rise to the termination to value preorder is pointed such that the object of numerals is standard and the map *cond* is left-strict, then for every number p it can be shown using the map $\phi_p : N \to N$ defined in the remark following Definition 6.2.1 that the map $\tilde{p}$ is maximal with respect to the termination preorder, and furthermore, it can be shown that the map $\perp_N$ is minimal with respect to the termination to value preorder, so we conclude that under such circumstances the two preorders coincide. Thus this is also the case in the context of Theorem 6.5.5, so this result could equally well have been stated in terms of the termination to value preorder instead of the termination preorder.

Chapter 7

The Programming Language PCF

This chapter introduces the programming language PCF. The syntax is introduced in Section 7.1 and a sound categorical interpretation is given in Section 7.2. In Section 7.3 additional non-order-theoretic axioms are imposed on the categorical model with the aim of proving an adequacy result. The essential ingredient of the categorical model is a rationally open fixpoint operator. Observable types are discussed in Section 7.4; an observable type is a type where there is a converse to adequacy. In Section 7.5 we prove an unwinding theorem for PCF using adequacy. This enables us to show that a restricted version of our axiom of rational openness is not only sufficient, but also necessary for the interpretation to be adequate. Essentially, what we do is we restrict rational openness to maps definable in PCF. In Section 7.6 we make a detour to syntactic notions of rationality and rational openness.

7.1 Syntax and Operational Semantics

The programming language PCF as originally presented in [Sco69] has exponential types and two ground types, namely types for numerals and booleans. We shall, however, consider a different version with product and sum types in addition to the exponential types, but only one ground type, namely a type for numerals. Booleans are then represented by numerals in the traditional way. It is an extension of the λ-calculus with numerals and recursion where the usual reduction rules are replaced by a Martin-Löf style operational semantics, [ML84]. The units for product and sum types are of limited computational interest, but we have included them for the sake of completeness. Types of PCF are given by the grammar

$$s \quad ::= \quad N \mid 1 \mid s \times s \mid s \Rightarrow s \mid 0 \mid s + s$$

and terms are given by the grammar

$$\begin{array}{rcl} t & ::= & x \mid \\ & & \mathtt{zero} \mid \mathtt{succ}(t) \mid \mathtt{pred}(t) \mid \mathtt{if}\ t\ \mathtt{then}\ t\ \mathtt{else}\ t \mid \\ & & \mathtt{true} \mid (t,t) \mid \mathtt{fst}(t) \mid \mathrm{snd}(t) \mid \\ & & \lambda x^A.t \mid tt \mid \\ & & \mathtt{false}^C(t) \mid \mathtt{inl}^{A+B}(t) \mid \mathtt{inr}^{A+B}(t) \mid \mathtt{case}\ t\ \mathtt{of}\ \mathtt{inl}(x).t \mid \mathtt{inr}(y).t \mid \\ & & \Omega_A \mid \mathtt{Y}_A \end{array}$$

Figure 7.1: Type Assignment Rules for PCF

$$\frac{}{x_1 : A_1, \ldots, x_n : A_n \vdash x_p : A_p}$$

$$\frac{}{, \vdash \mathtt{zero} : N} \qquad \frac{, \vdash u : N}{, \vdash \mathtt{succ}(u) : N} \qquad \frac{, \vdash u : N}{, \vdash \mathtt{pred}(u) : N}$$

$$\frac{, \vdash u : N \quad , \vdash v : A \quad , \vdash w : A}{, \vdash \mathtt{if}\ u\ \mathtt{then}\ v\ \mathtt{else}\ w : A}$$

$$\frac{}{, \vdash \mathtt{true} : 1} \qquad \frac{, \vdash u : A \quad , \vdash v : B}{, \vdash (u, v) : A \times B} \qquad \frac{, \vdash u : A \times B}{, \vdash \mathtt{fst}(u) : A} \qquad \frac{, \vdash u : A \times B}{, \vdash \mathrm{snd}(u) : B}$$

$$\frac{, , x : A \vdash u : B}{, \vdash \lambda x^A.u : A \Rightarrow B} \qquad \frac{, \vdash f : A \Rightarrow B \quad , \vdash u : A}{, \vdash fu : B}$$

$$\frac{, \vdash w : 0}{, \vdash \mathtt{false}^C(w) : C} \qquad \frac{, \vdash u : A}{, \vdash \mathtt{inl}^{A+B}(u) : A + B} \qquad \frac{, \vdash u : B}{, \vdash \mathtt{inr}^{A+B}(u) : A + B}$$

$$\frac{, \vdash w : A + B \quad , , x : A \vdash u : C \quad , , y : B \vdash v : C}{, \vdash \mathtt{case}\ w\ \mathtt{of}\ \mathtt{inl}(x).u \mid \mathtt{inr}(y).v : C}$$

$$\frac{}{, \vdash \Omega_A : A} \qquad \frac{}{, \vdash \mathtt{Y}_A : (A \Rightarrow A) \Rightarrow A}$$

where x is a variable ranging over terms. The set of free variables, $FV(u)$, of a term u is defined by induction on u. We consider each case except those of the λ-calculus which are given in Section 3.2.

$$\begin{array}{rcl} FV(\mathtt{zero}) & = & \emptyset \\ FV(\mathtt{succ}(u)) & = & FV(u) \\ FV(\mathtt{pred}(u)) & = & FV(u) \\ FV(\mathtt{if}\ w\ \mathtt{then}\ u\ \mathtt{else}\ v) & = & FV(w) \cup FV(u) \cup FV(v) \\ FV(\Omega) & = & \emptyset \\ FV(\mathtt{Y}) & = & \emptyset \end{array}$$

Rules for assignment of types to terms are given in Figure 7.1. Type assignments have the form of sequents

$$x_1 : A_1, ..., x_n : A_n \vdash u : B$$

where $x_1, ..., x_n$ are pairwise distinct variables. It can be shown by induction on the derivation of the type assignment that

$$FV(u) \subseteq \{x_1, ..., x_n\}$$

We will now show some properties of PCF that will be of use later on. The following results are simple extensions of analogous results for the λ-calculus - see Section 3.2.

Lemma 7.1.1 If the sequent $, \vdash u : A$ is derivable, then for any derivable sequent $, \vdash u : B$ we have $A = B$.

Proof: See Lemma 3.2.1. □

The following result says that the term of a derivable sequent encodes the derivation:

Proposition 7.1.2 If the sequent $, \vdash u : A$ is derivable, then the rule instance above the sequent is uniquely determined.

Proof: See Proposition 3.2.2. □

We need a small lemma dealing with expansion of contexts.

Lemma 7.1.3 If the sequent $\Delta, \Lambda \vdash u : A$ is derivable and the variables in the contexts Δ, Λ and $,$ are pairwise distinct, then the sequent $\Delta, , , \Lambda \vdash u : A$ is also derivable.

Proof: See Lemma 3.2.3. □

Now comes a lemma dealing with substitution:

Lemma 7.1.4 (Substitution Property) If the sequents $, \vdash u : A$ and $, , x : A, \Lambda \vdash v : B$ are derivable, then the sequent $, , \Lambda \vdash v[u/x] : B$ is also derivable.

Proof: See Lemma 3.2.4. □

We also have a lemma dealing with the "inverse" to substitution; this will be useful when proving adequacy in Section 7.3. Recall that $v[x/u]$ denotes the term v where inductively all occurrences of the term u have been replaced by the variable x.

Lemma 7.1.5 If the sequents $, \vdash u : A$ and $, , \Lambda \vdash v : B$ are derivable, then the sequent $, , x : A, \Lambda \vdash v[x/u] : B$ is also derivable where x is a new variable.

Proof: See Lemma 3.2.5. □

We will now give a lazy operational semantics for PCF. A *program* is an arbitrary closed term and a *value* is a closed term of one of the forms

$$\mathtt{succ}^n(\mathtt{zero}) \qquad \mathtt{true} \qquad (v, w) \qquad \lambda x.u \qquad \mathtt{inl}(u) \qquad \mathtt{inr}(u)$$

Figure 7.2: Operational Semantics for PCF

$$\frac{}{\mathtt{zero} \Downarrow \mathtt{zero}} \qquad \frac{u \Downarrow c}{\mathtt{succ}(u) \Downarrow \mathtt{succ}(c)} \qquad \frac{u \Downarrow \mathtt{zero}}{\mathtt{pred}(u) \Downarrow \mathtt{zero}} \qquad \frac{u \Downarrow \mathtt{succ}(c)}{\mathtt{pred}(u) \Downarrow c}$$

$$\frac{u \Downarrow \mathtt{zero} \quad v \Downarrow d}{\mathtt{if}\ u\ \mathtt{then}\ v\ \mathtt{else}\ w \Downarrow d} \qquad \frac{u \Downarrow \mathtt{succ}(c) \quad w \Downarrow e}{\mathtt{if}\ u\ \mathtt{then}\ v\ \mathtt{else}\ w \Downarrow e}$$

$$\frac{}{\mathtt{true} \Downarrow \mathtt{true}} \qquad \frac{}{(u, v) \Downarrow (u, v)} \qquad \frac{u \Downarrow (v, w) \quad v \Downarrow c}{\mathtt{fst}(u) \Downarrow c} \qquad \frac{u \Downarrow (v, w) \quad w \Downarrow d}{\mathrm{snd}(u) \Downarrow d}$$

$$\frac{}{\lambda x.u \Downarrow \lambda x.u} \qquad \frac{f \Downarrow \lambda x.v \quad v[u/x] \Downarrow c}{fu \Downarrow c}$$

$$\frac{}{\mathtt{inl}(u) \Downarrow \mathtt{inl}(u)} \qquad \frac{}{\mathtt{inr}(u) \Downarrow \mathtt{inr}(u)}$$

$$\frac{w \Downarrow \mathtt{inl}(t) \quad u[t/x] \Downarrow c}{\mathtt{case}\ w\ \mathtt{of}\ \mathtt{inl}(x).u \mid \mathtt{inr}(y).v \Downarrow c} \qquad \frac{w \Downarrow \mathtt{inr}(t) \quad v[t/x] \Downarrow c}{\mathtt{case}\ w\ \mathtt{of}\ \mathtt{inl}(x).u \mid \mathtt{inr}(y).v \Downarrow c}$$

$$\frac{}{\mathtt{Y} \Downarrow \lambda f.f(\mathtt{Y}f)}$$

where n is a number and a term $\mathtt{succ}^n(\mathtt{zero})$ is defined in the obvious way. Let T be the set of programs and C the set of values. The evaluation rules given in Figure 7.2 induce a relation $\Downarrow \subseteq T \times C$ called the *evaluation relation*. Note how the choice of lazy evaluation strategy is reflected in the evaluation rule for application; we do not evaluate an argument before plugging it into the body of an abstraction. Also note that there is no evaluation rule for the term Ω_A which entails that the term does not terminate. There is no evaluation rule for the term $\mathtt{false}(w)$ either. Given a program u, we will write $u \Downarrow$ iff there exists a term c such that $u \Downarrow c$. It can be shown that the evaluation rules corresponding to the λ-calculus fragment of PCF can be matched by β-reductions in the sense that if $v \Downarrow c$ then the term v reduces to the a value c. So the operational semantics has a clear logical content via the Curry-Howard interpretation.

The choice of evaluation strategy actually matters for the observational behaviour of PCF terms. For example, we have $(\lambda x.\mathtt{zero})\Omega \Downarrow \mathtt{zero}$ with our lazy evaluation rule for application

$$\frac{f \Downarrow \lambda x.v \quad v[u/x] \Downarrow c}{fu \Downarrow c}$$

but if we replace it with the eager evaluation rule

$$\frac{f \Downarrow \lambda x.v \quad u \Downarrow d \quad v[d/x] \Downarrow c}{fu \Downarrow c}$$

then the term $(\lambda x.\mathtt{zero})\Omega$ does not terminate. In Chapter 9 we shall consider a linear version of PCF, namely LPCF, where it turns out that the choice of evaluation strategy does not matter for the observable behaviour. The operational semantics enjoys the following properties:

Proposition 7.1.6 (Subject Reduction) If u is a program of type A and $u \Downarrow c$ then c is also of type A.

Proof: Induction on the derivation of $u \Downarrow c$ where we use Lemma 7.1.4. □

So if T_A is the set of programs of type A and C_A the set of values of type A, then this shows that the evaluation relation $\Downarrow \subseteq T \times C$ can be split up into a family of relations such that a relation $\Downarrow_A \subseteq T_A \times C_A$ is given for each type A.

Proposition 7.1.7 (Determinacy) If $u \Downarrow c$ and $u \Downarrow d$ then $c = d$.

Proof: Induction on the derivation of $u \Downarrow c$. □

7.2 Categorical Semantics

In this section we will give a sound categorical interpretation of PCF. In Section 7.3 we shall impose additional assumptions on our category with the aim of proving an adequacy result.

Definition 7.2.1 A *categorical premodel for PCF* is a pointed cartesian closed category equipped with

- an object of numerals,
- a fixpoint operator for each object,
- weak finite sums such that the diagram

$$\begin{array}{ccc} A + B & \xrightarrow{[f,g]} & , \Rightarrow C \\ & \searrow & \downarrow h \Rightarrow C \\ [(f;(h \Rightarrow C)),(g;(h \Rightarrow C))] & & \Delta \Rightarrow C \end{array}$$

commutes for any maps $f : A \to , \Rightarrow C$, $g : B \to , \Rightarrow C$ and $h : \Delta \to , .$

Examples 7.2.2 In Chapter 2 and Chapter 6 it is made clear that the concrete categories **cpo** and **dI** are categorical premodels for PCF. This is also the case with the game model of [Abr96].

In Section 3.4 the motivation for the choice of weak finite sums such that the above mentioned diagram commutes, instead of the stronger assumption of finite sums, is discussed. Given a premodel for PCF, we can interpret types as objects, typing rules as natural operations on maps, and derivations of type assignments as maps. A derivable sequent

$$x_1 : A_1, ..., x_n : A_n \vdash u : B$$

is interpreted as a map

$$[\![A_1]\!] \times ... \times [\![A_n]\!] \xrightarrow{[\![u]\!]} [\![B]\!]$$

by induction on its derivation using the appropriate operations on maps induced by the categorical assumptions cf. below. We consider each case except those of the λ-calculus which can be found in Section 3.4.

- The derivation

$$\frac{}{, \vdash \mathtt{zero} : N}$$

is interpreted as

$$\frac{}{, \xrightarrow{\langle\rangle} 1 \xrightarrow{zero} N}$$

- The derivation

$$\frac{\Gamma, \vdash u : N}{\Gamma, \vdash \mathtt{succ}(u) : N}$$

  is interpreted as

$$\frac{\Gamma, \xrightarrow{u} N}{\Gamma, \xrightarrow{u} N \xrightarrow{succ} N}$$

- The derivation

$$\frac{\Gamma, \vdash u : N}{\Gamma, \vdash \mathtt{pred}(u) : N}$$

  is interpreted as

$$\frac{\Gamma, \xrightarrow{u} N}{\Gamma, \xrightarrow{u} N \xrightarrow{pred} N}$$

- The derivation

$$\frac{\Gamma, \vdash u : N \quad \Gamma, \vdash v : A \quad \Gamma, \vdash w : A}{\Gamma, \vdash \mathtt{if}\ u\ \mathtt{then}\ v\ \mathtt{else}\ w : A}$$

  is interpreted as

$$\frac{\Gamma, \xrightarrow{u} N \quad \Gamma, \xrightarrow{v} A \quad \Gamma, \xrightarrow{w} A}{\Gamma, \xrightarrow{\langle u,\langle v,w\rangle\rangle} N \times (A \times A) \xrightarrow{cond} A}$$

- The derivation

$$\frac{}{\Gamma, \vdash \Omega_A : A}$$

  is interpreted as

$$\frac{}{\Gamma, \xrightarrow{\langle\rangle} 1 \xrightarrow{\bot} A}$$

- The derivation

$$\frac{}{\Gamma, \vdash \mathtt{Y}_A : (A \Rightarrow A) \Rightarrow A}$$

  is interpreted as

$$\frac{}{\Gamma, \xrightarrow{\langle\rangle} 1 \xrightarrow{H^\sharp} (A \Rightarrow A) \Rightarrow A}$$

  using the internal fixpoint operator induced by the appropriate fixpoint operator according to Proposition 6.3.9.

The following lemma corresponds to Lemma 7.1.3 where the categorical semantics is taken into account:

Lemma 7.2.3 If the sequent $\Delta, \Lambda \vdash u : A$ is derivable and the variables in the contexts Δ, Λ and Γ, Γ are pairwise distinct, then the sequent $\Delta, \Gamma, \Gamma, \Lambda \vdash u : A$ is also derivable and it has the interpretation

$$\Delta \times \Gamma \times \Gamma \times \Lambda \xrightarrow{\Delta\times\langle\rangle\times\Lambda} \Delta \times 1 \times \Lambda \cong \Delta \times \Lambda \xrightarrow{[\![u]\!]} A$$

Proof: Extend the proof of Lemma 3.4.3 for the λ-calculus as appropriate. □

The following lemma corresponds to Lemma 7.1.4 where the categorical semantics is taken into account; it says essentially that substitution corresponds to composition:

Lemma 7.2.4 (Substitution) If the sequents $\Gamma, \vdash u : A$ and $\Gamma, \Gamma, x : A, \Lambda \vdash v : B$ are derivable, then the sequent $\Gamma, \Gamma, \Lambda \vdash v[u/x] : B$ is also derivable and it has the interpretation

$$\Gamma \times \Gamma \times \Lambda \xrightarrow{\Delta\times\Lambda} \Gamma \times \Gamma \times \Gamma \times \Lambda \xrightarrow{\Gamma\times[\![u]\!]\times\Lambda} \Gamma \times \Gamma \times A \times \Lambda \xrightarrow{[\![v]\!]} B$$

Proof: Extend the proof of Lemma 3.4.4 for the λ-calculus as appropriate. □

The interpretation is preserved by evaluation:

Theorem 7.2.5 (Soundness) Given a program u such that $u \Downarrow c$, then $[\![u]\!] = [\![c]\!]$.

Proof: Induction on the derivation of $u \Downarrow c$ where we use Lemma 7.2.4. Note that the evaluation rule for the fixpoint constant preserves the interpretation because it is essentially a syntactic restatement of the defining diagram for an internal fixpoint operator, Definition 6.3.7. □

7.3 Adequacy

In our adequacy proof for PCF we will use the now standard technique of logical relations originally introduced in [Plo73].

The original adequacy proof in [Plo77] made use of a unary logical relation $Comp_A \subseteq T_A$ called the *computability predicate*. Recall that T_A is the set of programs of type A. The logical relation $Comp_A$ is defined by induction on the type A such that at ground type N it coincides with adequacy, that is, $Comp_N(t)$ is defined to hold iff $[\![t]\!] \neq \bot$ entails $t \Downarrow$. It is then proved that every program satisfies the computability predicate.

Another method for proving adequacy is given in [Gun92, Win93]; here a binary logical relation $\preceq_A \subseteq [\![A]\!] \times T_A$ is used, where $d \preceq_A t$ expresses that the element $d \in A$ "approximates" the term t. The logical relation $\preceq_A$ is defined by induction on the type A such that at ground type $d \preceq_N t$ is defined to hold iff $d \neq \bot$ entails $t \Downarrow \mathtt{succ}^q(\mathtt{zero})$ for some number q such that $d = \tilde{q}$. One then proves that $[\![t]\!] \preceq_A t$ for every program t of type A. A crucial part of this proof amounts to showing that every predicate $(\Leftrightarrow) \preceq_A t$ is *inclusive* in the sense of being closed under least upper bounds of increasing chains, that is, if $\{d_n\}_{n\in\omega}$ is an increasing chain in A such that $d_n \preceq_A t$ holds for every number n, then $\sqcup_{n\in\omega} d_n \preceq_A t$ also holds. If we disregard sums we could use the same idea here adapted to our order-free categorical setting; an appropriate inclusiveness result would then be as follows: Let a program u of type A and maps $f : D \to D$ and $g : D \to A$ be given; it is then the case that if $\bot; f^n; g \preceq_A u$ holds for every number n, then $f^\sharp; g \preceq_A u$ holds too. Here we consider the binary logical relation $\preceq_A$ given below. Such a result can be proved by induction on the type A. Inclusiveness together with Lemma 7.3.7 then gives $[\![\mathtt{Y}_A]\!] \preceq_{(A\Rightarrow A)\Rightarrow A} \mathtt{Y}_A$ which allows us to remove the no-fixpoint-constant restriction of Lemma 7.3.6, and thus obtain $[\![t]\!] \preceq_A t$ for every program t of type A. But it turns out that this approach does not carry over to the linear version of PCF which we will consider in Section 9.7. We prefer to use a method that generalises to the linear setting, so we will prove adequacy using the binary logical relation $\preceq_A$ in a way that dodges the inclusiveness result.

Now, we need some extra assumptions on our premodel for PCF:

Definition 7.3.1 A *categorical model for PCF* is a categorical premodel in the sense of Definition 7.2.1 such that

- the maps π_1 and π_2 are strict and the map *eval* is left-strict,
- the maps *succ* and *pred* are strict and the map *cond* is left-strict,
- the map $[] : 0 \to C$ is strict for any object C and the map $[f, g] : A + B \to C$ is strict for any maps $f : A \to C$ and $g : B \to C$.

Examples 7.3.2 In Chapter 2 and Chapter 6 it is made clear that the concrete categories **cpo** and **dI** are categorical models for PCF. This is also the case with the game model of [Abr96].

Note that the adequacy result below is stated solely in terms of the categorical premodel, Definition 7.2.1, but the assumptions added in Definition 7.3.1 are indeed used in the proof.

Definition 7.3.3 For each type A the binary logical relation

$$\preceq_A \subseteq \mathcal{C}(1, A) \times T_A$$

is defined by induction on the structure of A as

$$\begin{array}{rll}
f \preceq_N t & \text{iff} & f \neq \bot \Rightarrow \exists n \in \omega.\, t \Downarrow \mathtt{succ}^n(\mathtt{zero}) \ \wedge \ f = \tilde{n} \\
f \preceq_1 t & \text{iff} & true \\
f \preceq_{B \times C} t & \text{iff} & (f \neq \bot \Rightarrow t \Downarrow) \wedge \\
 & & (f;\pi_1 \preceq_B \mathtt{fst}(t) \ \wedge \ f;\pi_2 \preceq_C \mathrm{snd}(t)) \\
f \preceq_{B \Rightarrow C} t & \text{iff} & (f \neq \bot \Rightarrow t \Downarrow) \wedge \\
 & & (\forall g \in \mathcal{C}(1,B).\forall v \in T_B.\, g \preceq_B v \ \Rightarrow \ \langle f, g\rangle; eval \preceq_C tv) \\
f \preceq_0 t & \text{iff} & f = \bot \\
f \preceq_{B+C} t & \text{iff} & f \neq \bot \Rightarrow \\
 & & (\exists h \in \mathcal{C}(1,B).\exists v \in T_B.\, f = h; in_1 \ \wedge \ t \Downarrow \mathtt{inl}(v) \ \wedge \ h \preceq_B v) \vee \\
 & & (\exists h \in \mathcal{C}(1,C).\exists v \in T_C.\, f = h; in_2 \ \wedge \ t \Downarrow \mathtt{inr}(v) \ \wedge \ h \preceq_C v)
\end{array}$$

Lemma 7.3.4 For each program t of type A we have $\bot \preceq_A t$.

Proof: Induction on the structure of A; note that the induction hypothesis is not needed for the sum case. □

In the following lemma an essential connection between the logical relation $\preceq$ and the evaluation relation $\Downarrow$ is shown:

Lemma 7.3.5 If $f \preceq_A t$ and $u \Downarrow c$ whenever $t \Downarrow c$, then $f \preceq_A u$.

Proof: Induction on the structure of A; we proceed case by case.

- The $A = N$ case. If $f \neq \bot$ then there exists a number n such that $f = \tilde{n}$ and $t \Downarrow \mathtt{succ}^n(\mathtt{zero})$. But then also $u \Downarrow \mathtt{succ}^n(\mathtt{zero})$.
- The $A = 1$ case. Obvious.
- The $A = B \times C$ case. First observe that if $f \neq \bot$ then $t \Downarrow$ which entails that $u \Downarrow$. We have to show that $f;\pi_1 \preceq_B \mathtt{fst}(u)$ and $f;\pi_2 \preceq_C \mathrm{snd}(u)$. The first stipulation follows from the induction hypothesis because $f;\pi_1 \preceq_B \mathtt{fst}(t)$, and moreover, $\mathtt{fst}(t) \Downarrow c$ entails $\mathtt{fst}(u) \Downarrow c$. The argument is similar for the second stipulation.
- The $A = B \Rightarrow C$ case. First observe that if $f \neq \bot$ then $t \Downarrow$ which entails that $u \Downarrow$. Assume that we have a map g and a term v such that $g \preceq_B v$, we then have to show that $\langle f, g\rangle; eval \preceq_B uv$. This follows from the induction hypothesis because $\langle f, g\rangle; eval \preceq_B tv$, and moreover, $tv \Downarrow c$ entails $uv \Downarrow c$.
- The $A = 0$ case. Obvious.
- The $A = B + C$ case. If $f \neq \bot$ then without loss of generality there exists a map h and a term v such that $f = h; in_1$ and $t \Downarrow \mathtt{inl}(v)$, and moreover, $h \preceq_A v$. But then also $u \Downarrow \mathtt{inl}(v)$.

□

Note that in the following lemma the term u is assumed not to contain any occurrences of the fixpoint constant. Also note that such a restriction is not imposed on the t_i terms.

Lemma 7.3.6 Consider a derivable sequent

$$x_1 : A_1, ..., x_n : A_n \vdash u : C$$

such that the term u does not contain any occurrences of the fixpoint constant. Assume that for each $i \in \{1, ..., n\}$ we have a map $f_i : 1 \to A_i$ and a program t_i of type A_i such that $f_i \preceq_{A_i} t_i$. We then have

$$\langle f_1, ..., f_n \rangle ; [\![u]\!] \preceq_C u[t_1, ..., t_n / x_1, ..., x_n].$$

Proof: We proceed by induction on the derivation of $x_1 : A_1, ..., x_n : A_n \vdash u : C$. Without loss of generality we will assume that none of the variables $x_1, ..., x_n$ are bound in u. In what follows we will denote the context $x_1 : A_1, ..., x_n : A_n$ by $, \ ,$ and moreover, ρ denotes the map

$$1 \xrightarrow{\langle f_1, ..., f_n \rangle} A_1 \times ... \times A_n$$

and σ denotes the substitution $[t_1, ..., t_n / x_1, ..., x_n]$. We consider each case except symmetric ones. Note that since the term u is assumed to be without occurrences of the fixpoint constant, there is no case for that situation.

- In the case

  $$\frac{}{x_1 : A_1, ..., x_n : A_n \vdash x_p : A_p}$$

  we have to show that

  $$\langle f_1, ..., f_n \rangle ; \pi_p \preceq_{A_p} x_p[t_1 / x_1, ..., t_n / x_n]$$

  which amounts to $f_p \preceq_{A_p} t_p$.

- In the case

  $$\frac{}{, \ \vdash \mathtt{zero} : N}$$

  we have to show that $\rho ; \langle\rangle ; zero \preceq_N \mathtt{zero}\sigma$ which amounts to $zero \preceq_N \mathtt{zero}$.

- In the case

  $$\frac{, \ \vdash u : N}{, \ \vdash \mathtt{succ}(u) : N}$$

  we have to show that $\rho ; [\![u]\!] ; succ \preceq_N \mathtt{succ}(u)\sigma$. Now, $\rho ; [\![u]\!] ; succ \neq \bot$ entails that $\rho ; [\![u]\!] \neq \bot$, so $u\sigma \Downarrow \mathtt{succ}^n(\mathtt{zero})$ for some number n such that $\rho ; [\![u]\!] = \tilde{n}$ cf. the induction hypothesis. We therefore get

  $$\frac{u\sigma \Downarrow \mathtt{succ}^n(\mathtt{zero})}{\mathtt{succ}(u)\sigma \Downarrow \mathtt{succ}^{n+1}(\mathtt{zero})}$$

  and $\rho ; [\![u]\!] ; succ = \widetilde{n+1}$.

- In the case

  $$\frac{, \ \vdash u : N}{, \ \vdash \mathtt{pred}(u) : N}$$

  we have to show that $\rho ; [\![u]\!] ; pred \preceq_N \mathtt{pred}(u)\sigma$. Now, $\rho ; [\![u]\!] ; pred \neq \bot$ entails that $\rho ; [\![u]\!] \neq \bot$, so $u\sigma \Downarrow \mathtt{succ}^n(\mathtt{zero})$ for some number n such that $\rho ; [\![u]\!] = \tilde{n}$ cf. the induction hypothesis. If $n = 0$ then we get

  $$\frac{u\sigma \Downarrow \mathtt{zero}}{\mathtt{pred}(u)\sigma \Downarrow \mathtt{zero}}$$

  and $\rho ; [\![u]\!] ; pred = zero$. If $n \geq 1$ then we get

  $$\frac{u\sigma \Downarrow \mathtt{succ}^n(\mathtt{zero})}{\mathtt{pred}(u)\sigma \Downarrow \mathtt{succ}^{n-1}(\mathtt{zero})}$$

  and $\rho ; [\![u]\!] ; pred = \widetilde{n \Leftrightarrow 1}$.

- In the case
$$\frac{, \vdash u : N \quad , \vdash v : A \quad , \vdash w : A}{, \vdash \mathtt{if}\ u\ \mathtt{then}\ v\ \mathtt{else}\ w : A}$$
we have to show that
$$l \preceq_A (\mathtt{if}\ u\ \mathtt{then}\ v\ \mathtt{else}\ w)\sigma$$
where the map l is defined as
$$l = \rho; \langle [\![u]\!], \langle [\![v]\!], [\![w]\!] \rangle \rangle; cond.$$
Now
$$l = \langle (\rho; [\![u]\!]), \langle (\rho; [\![v]\!]), (\rho; [\![w]\!]) \rangle \rangle; cond$$
so if $\rho; [\![u]\!] = \perp$ then $l = \perp$ and we are done according to Lemma 7.3.4; otherwise $\rho; [\![u]\!] \neq \perp$ and hence $u\sigma \Downarrow \mathtt{succ}^n(\mathtt{zero})$ for some number n such that $\rho; [\![u]\!] = \tilde{n}$ cf. the induction hypothesis. If $n = 0$ then $\rho; [\![u]\!] = \tilde{0}$ and thus $l = \rho; [\![v]\!]$. But $\rho; [\![v]\!] \preceq_A v\sigma$ cf. the induction hypothesis, and $v\sigma \Downarrow c$ for some value c entails that we get
$$\frac{u\sigma \Downarrow \mathtt{zero} \quad v\sigma \Downarrow c}{(\mathtt{if}\ u\ \mathtt{then}\ v\ \mathtt{else}\ w)\sigma \Downarrow c}$$
so we are done according to Lemma 7.3.5. If $n \geq 1$ then the situation is analogous.

- In the case
$$\frac{}{, \vdash \mathtt{true} : 1}$$
we have to show that $\rho; \langle\rangle \preceq_1 \mathtt{true}\sigma$ which holds trivially.

- In the case
$$\frac{, \vdash u : A \quad , \vdash v : B}{, \vdash (u, v) : A \times B}$$
we have to show that $\rho; \langle [\![u]\!], [\![v]\!] \rangle \preceq_{A \times B} (u, v)\sigma$. First, observe that $(u, v)\sigma \Downarrow$. Note that
$$\rho; \langle [\![u]\!], [\![v]\!] \rangle = \langle (\rho; [\![u]\!]), (\rho; [\![v]\!]) \rangle.$$
Now, $\rho; [\![u]\!] \preceq_A u\sigma$ cf. the induction hypothesis, and $u\sigma \Downarrow c$ for some value c entails that we get
$$\frac{\dfrac{}{(u, v)\sigma \Downarrow (u, v)\sigma} \quad u\sigma \Downarrow c}{\mathtt{fst}(u, v)\sigma \Downarrow c}$$
so we conclude that $\rho; [\![u]\!] \preceq_A \mathtt{fst}(u, v)\sigma$ according to Lemma 7.3.5. An analogous argument shows that $\rho; [\![v]\!] \preceq_B \mathrm{snd}(u, v)\sigma$.

- In the case
$$\frac{, \vdash u : A \times B}{, \vdash \mathtt{fst}(u) : A}$$
we have to show that $\rho; [\![u]\!]; \pi_1 \preceq_A \mathtt{fst}(u)\sigma$. But $\rho; [\![u]\!] \preceq_{A \times B} u\sigma$ cf. the induction hypothesis.

- In the case
$$\frac{, , y : A \vdash u : B}{, \vdash \lambda y.u : A \Rightarrow B}$$
we have to show that $\rho; \lambda([\![u]\!]) \preceq_{A \Rightarrow B} (\lambda y.u)\sigma$. First, observe that $(\lambda y.u)\sigma \Downarrow$. For any map h and term v such that $h \preceq_A v$ we have
$$\langle (\rho; \lambda([\![u]\!])), h \rangle; eval \preceq_B u\sigma[v/y]$$

cf. the induction hypothesis, and $u\sigma[v/y] \Downarrow c$ for some value c entails that we get

$$\frac{\overline{(\lambda y.u)\sigma \Downarrow (\lambda y.u)\sigma} \qquad u\sigma[v/y] \Downarrow c}{(\lambda y.u)\sigma v \Downarrow c}$$

so we conclude that

$$\langle(\rho; \lambda([\![u]\!])), h\rangle; eval \preceq_B (\lambda y.u)\sigma v$$

according to Lemma 7.3.5.

- In the case

$$\frac{\Gamma \vdash g : A \Rightarrow B \qquad \Gamma \vdash u : A}{\Gamma \vdash gu : B}$$

we have to show that $\rho; \langle [\![g]\!], [\![u]\!]\rangle; eval \preceq_B (gu)\sigma$. Note that

$$\rho; \langle [\![g]\!], [\![u]\!]\rangle = \langle(\rho; [\![g]\!]), (\rho; [\![u]\!])\rangle.$$

But $\rho; [\![g]\!] \preceq_{A\Rightarrow B} g\sigma$ and $\rho; [\![u]\!] \preceq_A u\sigma$ cf. the induction hypothesis.

- In the case

$$\frac{\Gamma \vdash w : 0}{\Gamma \vdash \mathtt{false}(w) : C}$$

we have to show that $\rho; [\![w]\!]; [] \preceq_C \mathtt{false}(w)\sigma$. We have $\rho; [\![w]\!] \preceq_0 w\sigma$ cf. the induction hypothesis, so $\rho; [\![w]\!] = \bot$ which entails that $\rho; [\![w]\!]; [] = \bot$ and we are done according to Lemma 7.3.4.

- In the case

$$\frac{\Gamma \vdash u : A}{\Gamma \vdash \mathtt{inl}(u) : A + B}$$

we have to show that $\rho; [\![u]\!]; in_1 \preceq_{A+B} \mathtt{inl}(u)\sigma$. But we have $\rho; [\![u]\!] \preceq_A u\sigma$ cf. the induction hypothesis and $\mathtt{inl}(u)\sigma \Downarrow \mathtt{inl}(u)\sigma$.

- In the case

$$\frac{\Gamma \vdash w : A + B \qquad \Gamma, y : A \vdash u : C \qquad \Gamma, z : B \vdash v : C}{\Gamma \vdash \mathtt{case}\ w\ \mathtt{of}\ \mathtt{inl}(y).u \mid \mathtt{inr}(z).v : C}$$

we have to show that

$$l \preceq_C (\mathtt{case}\ w\ \mathtt{of}\ \mathtt{inl}(y).u \mid \mathtt{inr}(z).v)\sigma$$

where the map l is defined as

$$l = \rho; \langle [\![w]\!], id\rangle; \lambda^{-1}([\lambda(\cong; [\![u]\!]), \lambda(\cong; [\![v]\!])]).$$

Now

$$l = \langle(\rho; [\![w]\!]; [\lambda(\cong; [\![u]\!]), \lambda(\cong; [\![v]\!])]), f\rangle; eval$$

so if $\rho; [\![w]\!] = \bot$ then $l = \bot$ and we are done according to Lemma 7.3.4; otherwise $\rho; [\![w]\!] \neq \bot$ and without loss of generality we have a map h and a term v such that $\rho; [\![w]\!] = h; in_1$ and $w\sigma \Downarrow \mathtt{inl}(v)$, and moreover, $h \preceq_A v$ cf. the induction hypothesis. This entails that $l = \langle f, h\rangle; [\![u]\!]$, but $\langle f, h\rangle; [\![u]\!] \preceq_C u\sigma[v/y]$ cf. the induction hypothesis, and $u\sigma[v/y] \Downarrow c$ for some value c entails that we get

$$\frac{w\sigma \Downarrow \mathtt{inl}(v) \qquad u\sigma[v/y] \Downarrow c}{(\mathtt{case}\ w\ \mathtt{of}\ \mathtt{inl}(y).u \mid \mathtt{inr}(z).v)\sigma \Downarrow c}$$

so we are done according to Lemma 7.3.5.

- In the case

$$\frac{}{\Gamma, \vdash \Omega : A}$$

we have to show that $\rho; \langle\rangle; \bot \preceq_A \Omega\sigma$ which amounts to $\bot \preceq_A \Omega$, but this assertion is true according to Lemma 7.3.4.

□

The following lemma says that the (formal) finite approximants to an internal fixpoint operator are $\preceq$-related to the corresponding fixpoint constant:

Lemma 7.3.7 For every type A and number n we have $\bot; H_A^n \preceq_{(A \Rightarrow A) \Rightarrow A} \mathtt{Y}_A$.

Proof: Recall that the map H_A is defined in the proof of Proposition 6.3.9. We proceed by induction on n. The assertion is true in case $n = 0$ according to Lemma 7.3.4. Now assume that the assertion is true for an arbitrary number n; we then have to show that $\bot; H^{n+1} \preceq_{(A \Rightarrow A) \Rightarrow A} \mathtt{Y}$. First, observe that $\mathtt{Y} \Downarrow$. Assume that we have a map h and a term v such that $h \preceq_{A \Rightarrow A} v$; we then have to show that

$$\langle (\bot; H^{n+1}), h \rangle; eval \preceq_A \mathtt{Y}v. \tag{7.1}$$

We have

$$\bot; H^{n+1} = \ulcorner \Delta; (id \times \ulcorner \bot; H^{n \urcorner - 1}); eval \urcorner$$

cf. the proof of Proposition 6.3.9, so we get

$$\begin{array}{rcl} \langle (\bot; H^{n+1}), h \rangle; eval & = & h; \Delta; (id \times \ulcorner \bot; H^{n \urcorner - 1}); eval \\ & = & \langle h, (\langle (\bot; H^n), h \rangle; eval) \rangle; eval \end{array}$$

Now

$$\langle (\bot; H^n), h \rangle; eval \preceq_A \mathtt{Y}v$$

cf. the induction hypothesis, which entails that

$$\langle h, (\langle (\bot; H^n), h \rangle; eval) \rangle; eval \preceq_A v(\mathtt{Y}v)$$

and $v(\mathtt{Y}v) \Downarrow c$ for some value c entails that we get

$$\frac{\dfrac{}{\mathtt{Y} \Downarrow \lambda f.f(\mathtt{Y}f)} \qquad v(\mathtt{Y}v) \Downarrow c}{\mathtt{Y}v \Downarrow c}$$

so we conclude (7.1) according to Lemma 7.3.5. □

We are finally able to state a result expressing that the categorical interpretation is adequate with respect to the operational semantics. Note how Lemma 7.1.5 is used to "extract" the fixpoint constants of a term.

Theorem 7.3.8 (Adequacy) Let u be a program of type B. If the fixpoint operator is rationally open with respect to B, then $[\![u]\!] \neq \bot$ entails that $u \Downarrow$.

Proof: We apply Lemma 7.1.5 to the derivable sequent $\vdash u : B$ and obtain a derivable sequent

$$z_1 : (A_1 \Rightarrow A_1) \Rightarrow A_1, \ \ldots, z_n : (A_n \Rightarrow A_n) \Rightarrow A_n \vdash u' : B$$

such that

$$u = u'[\mathtt{Y}_{A_1}, \ldots, \mathtt{Y}_{A_n} / z_1, \ldots, z_n]$$

and such that the term u' does not contain any occurrences of the fixpoint constant. Note that only a special case of Lemma 7.1.5 is used where a constant, namely the fixpoint constant, is replaced by a variable. But

$$[\![u'[\mathtt{Y}_{A_1}, \ldots, \mathtt{Y}_{A_n} / z_1, \ldots, z_n]]\!] = \langle [\![\mathtt{Y}_{A_1}]\!], \ldots, [\![\mathtt{Y}_{A_n}]\!] \rangle; [\![u']\!]$$

and for each $i \in \{1, ..., n\}$ we have $[\![\mathtt{Y}_{A_i}]\!] = H^{\sharp}_{A_i}$ so there exist numbers $p_1, ..., p_n$ such that

$$\langle (\bot; H^{p_1}_{A_1}),\ \ldots\ , (\bot; H^{p_n}_{A_n})\rangle; [\![u']\!] \neq \bot$$

because the fixpoint operator is rationally open with respect to B and because $[\![u]\!] \neq \bot$. But for each $i \in \{1, ..., n\}$ it is the case that

$$\bot; H^{p_i}_{A_i} \preceq_{(A_i \Rightarrow A_i) \Rightarrow A_i} \mathtt{Y}_{A_i}$$

according to Lemma 7.3.7, which entails that

$$\langle (\bot; H^{p_1}_{A_1}), ..., (\bot; H_{A_n})^{p_n}\rangle; [\![u']\!] \preceq_B u'[\mathtt{Y}_{A_1}, ..., \mathtt{Y}_{A_n}/z_1, ..., z_n]$$

according to Lemma 7.3.6. We conclude that $u \Downarrow$ cf. the definition of $\preceq_B$. □

Note that the preceding theorem reveals information about how the semantics is related to termination/non-termination behaviour at any type in the sense that $[\![u]\!] \neq \bot$ implies $u \Downarrow$ whichever type the program u has.

Examples 7.3.9 The adequacy result, Theorem 7.3.8, holds when PCF is interpreted in the concrete categories **cpo** and **dI**. This is also the case with the game model of [Abr96].

7.4 Observable Types

Types where we have a converse to adequacy, that is, to Theorem 7.3.8, will be called *observable*. To be explicit, a type B is observable iff $u \Downarrow$ entails that $[\![u]\!] \neq \bot$ for any program u of type B when we assume that $\tilde{0} \neq \tilde{1}$.

Now, the ground type N is observable according to soundness and the observation that if $\tilde{n} = \bot$ for some number n then $\tilde{0} = \tilde{1}$. It is actually possible to obtain a more informative result, which is a categorical generalisation of the traditional notion of adequacy for PCF:

Corollary 7.4.1 Let u be a program of type N. If $\tilde{0} \neq \tilde{1}$ and the fixpoint operator is rationally open with respect to N, then $[\![u]\!] = \tilde{q}$ iff $u \Downarrow \mathtt{succ}^q(\mathtt{zero})$.

Proof: The result follows from Theorem 7.2.5 and Theorem 7.3.8 together with the observation that if $\tilde{n} = \bot$ for some number n, or if $\tilde{p} = \tilde{q}$ for different numbers p and q, then $\tilde{0} = \tilde{1}$. □

Sum types are observable too; at binary sum types $A + B$ it follows from soundness and the observation that if in_1 or in_2 factors through $\bot\colon 1 \to A + B$, then $\tilde{0} = \tilde{1}$, and at unary sum type 0 it follows from the observation that we cannot have $t \Downarrow$, where t is a program of type 0, as there are no a values of type 0. It actually turns out that a result analogous to Corollary 7.4.1 can be obtained at binary sum types:

Corollary 7.4.2 Let u be a program of type $A+B$. If $\tilde{0} \neq \tilde{1}$ and the fixpoint operator is rationally open with respect to $A + B$, then the map $[\![u]\!]$ has the property that it factors through in_1 iff $u \Downarrow \mathtt{inl}(t)$ for some term t, and analogously, it factors through in_2 iff $u \Downarrow \mathtt{inr}(v)$ for some term v.

Proof: The result follows from Theorem 7.2.5 and Theorem 7.3.8 together with the observation that if in_1 or in_2 factors through $\bot\colon 1 \to A+B$, or if in_1 and in_2 both factor through an arbitrary map $f : 1 \to A + B$, then $\tilde{0} = \tilde{1}$. □

So the $(N, 0, +)$ types are observable. Product and exponential types are not observable as the following examples show: The programs $\mathtt{true}$ and (Ω, Ω) of product type are values, but are interpreted as $\bot$, and the program $\lambda x.\Omega$ of exponential type is canonical, but is interpreted as $\bot$.

7.5 Unwinding

The adequacy result can be used to prove non-trivial syntactic properties of PCF in the presence of an appropriate model. Here we shall give the *unwinding theorem* which establishes a relation between the fixpoint constant $\mathtt{Y}$ and its finite approximants. A purely syntactic proof of a similar result can be found in [Gun92]. We first need a convention: For any type A we define a program $\mathtt{Y}^n_A$ of type $(A \Rightarrow A) \Rightarrow A$ for every number n by the stipulations

$$\begin{array}{rcl} \mathtt{Y}^0 & = & \Omega \\ \mathtt{Y}^{n+1} & = & \lambda f.f(\mathtt{Y}^n f) \end{array}$$

The terms $\mathtt{Y}^n$ are called *finite approximants* to the fixpoint constant $\mathtt{Y}$. By a small induction proof it can be shown that $[\![\mathtt{Y}^n]\!] = \bot ; H^n$ where the map H is defined in the proof of Proposition 6.3.9.

Theorem 7.5.1 (Unwinding) If t is a term of observable type with one free variable z of type $(A \Rightarrow A) \Rightarrow A$, then

$$t[\mathtt{Y}/z] \Downarrow \;\Leftrightarrow\; \exists n \in \omega \,.\, t[\mathtt{Y}^n/z] \Downarrow .$$

Proof: In what follows we will interpret PCF in the concrete category **cpo**; this is a model for PCF in the sense of Definition 7.3.1 such that the fixpoint operator is rationally open with respect to the interpretation of any observable type as is made clear in Section 7.3 and Section 6.3. We then have $t[\mathtt{Y}/z] \Downarrow$ iff $[\![t[\mathtt{Y}/z]]\!] \neq \bot$ cf. adequacy and its converse. But $[\![t[\mathtt{Y}/z]]\!]$ is a least upper bound for the increasing chain $\{[\![t[\mathtt{Y}^n/z]]\!]\}_{n\in\omega}$ because $[\![t[\mathtt{Y}/z]]\!] = H^\sharp ; [\![t]\!]$ and $[\![t[\mathtt{Y}^n/z]]\!] = \bot ; H^n ; [\![t]\!]$ according to the remark above, so $[\![t[\mathtt{Y}/z]]\!] \neq \bot$ iff there exists a number n such that $[\![t[\mathtt{Y}^n/z]]\!] \neq \bot$, that is, iff there exists a number n such that $t[\mathtt{Y}^n/z] \Downarrow$ cf. adequacy and its converse. □

At ground type N it is possible to obtain a more informative result: If t is a term of ground type N with one free variable z of type $(A \Rightarrow A) \Rightarrow A$, then $t[\mathtt{Y}/z] \Downarrow c$ iff there exists a number n such that $t[\mathtt{Y}^n/z] \Downarrow c$. The proof of the result is similar to the proof of Theorem 7.5.1. An analogous result can be obtained at binary sum types.

In the previous section we showed how the assumption of rational openness on a fixpoint operator entails that the categorical interpretation is adequate. It is possible to weaken this assumption such that it is not only sufficient, but also necessary for the interpretation to be adequate[1]. Essentially, what we do is we restrict rational openness to maps definable in PCF. Note that the "upwards" direction of the following theorem relies on the unwinding theorem.

Theorem 7.5.2 Assume that we have a category that is a model for PCF in the sense of Definition 7.3.1 such that $\tilde{0} \neq \tilde{1}$. Let an observable type B be given, then the assertions

- for all types A and programs t of type B with one free variable z of type $(A \Rightarrow A) \Rightarrow A$ it is the case that
$$H^\sharp ; [\![t]\!] \neq \bot \;\Rightarrow\; \exists n \in \omega .\; \bot ; H^n ; [\![t]\!] \neq \bot,$$
- for every program u of type B we have $[\![u]\!] \neq \bot$ entails that $u \Downarrow$,

are equivalent.

Proof: The "downwards" direction comes from the observation that the assumption made here is sufficient to prove Theorem 7.3.8 in the relevant case. The proof of the "upwards" direction goes as follows: Assume that the assumption made here holds, and consider a term t of type B with one free variable z of type $(A \Rightarrow A) \Rightarrow A$. If $H^\sharp ; [\![t]\!] \neq \bot$ then $t[\mathtt{Y}/z] \Downarrow$ according to the assumption because $H^\sharp ; [\![t]\!] = [\![t[\mathtt{Y}/z]]\!]$. But then there exists a number n such that $t[\mathtt{Y}^n/z] \Downarrow$ cf. the unwinding theorem, which entails the existence of a number n such that $\bot ; H^n ; [\![t]\!] \neq \bot$ because the type B is observable and $[\![t[\mathtt{Y}^n/z]]\!] = \bot ; H^n ; [\![t]\!]$. □

[1] This result was conjectured by Gordon Plotkin while the author visited Edinburgh in March '96.

7.6 A Digression - Syntactic Rational Openness

Using a preorder on terms, the notions of rationality and rational openness can be cast in a syntactic fashion. We shall first have a look at the contextual preorder on PCF-programs. Recall that the contextual preorder is defined as follows: For every pair u and v of programs of type A we define $u \lesssim_A v$ iff

$$t[u/z] \Downarrow \;\Rightarrow\; t[v/z] \Downarrow$$

for every term t of numerals type with one free variable z of type A. The symmetric closure $\approx$ of $\lesssim$ is then given by $\approx = \lesssim \cap \lesssim^{op}$. Using a concrete instance of adequacy it is straightforward to prove that Ω is least with respect to the contextual preorder and that the contextual preorder satisfies a syntactic version of rationality saying that $t[\mathtt{Y}/z]$ is a least upper bound for the increasing chain $\{t[\mathtt{Y}^n/z]\}_{n\in\omega}$ for any term t with one free variable z of type $(A \Rightarrow A) \Rightarrow A$. This can also be proved using purely syntactic means - see for example [Pit95] and [Las97]. It follows that a syntactic version of rational openness is satisfied, namely

$$t[\mathtt{Y}/z] \approx \Omega \;\Leftrightarrow\; \forall n \in \omega.\, t[\mathtt{Y}^n/z] \approx \Omega$$

for any term t with one free variable z of type $(A \Rightarrow A) \Rightarrow A$.

We will now have a look at an interesting example[2] of a preorder on terms with respect to which (the syntactic version of) rational openness is satisfied whereas (the syntactic version of) rationality does not hold. The starting point is the untyped eager λ-calculus extended with a non-deterministic choice operator. A description of the untyped eager λ-calculus can be found in [Win93]. Terms are given by the grammar

$$t \;::=\; x \mid \lambda x.t \mid tt \mid t \oplus t$$

where x is a variable ranging over terms. An operational semantics is given as follows: A program is an arbitrary closed term and a value is a program of the form $\lambda x.t$. Let T be the set of programs and let C be the set of values. An evaluation relation $\Downarrow \subseteq T \times C$ is given by the rules

$$\frac{}{\lambda x.t \Downarrow \lambda x.t} \qquad \frac{f \Downarrow \lambda x.t \quad u \Downarrow c \quad t[c/x] \Downarrow d}{fu \Downarrow d}$$

$$\frac{u \Downarrow c}{u \oplus v \Downarrow c} \qquad \frac{v \Downarrow d}{u \oplus v \Downarrow d}$$

We then define an applicative simulation preorder $\lessapprox \subseteq T \times T$ as the greatest fixpoint of a monotone operator

$$\langle \Leftrightarrow \rangle : \mathcal{P}(T \times T) \to \mathcal{P}(T \times T)$$

which is defined as follows: For every $R \subseteq T \times T$ we define $u \langle R \rangle v$ iff

$$\forall \lambda x.u' \in C.\, u \Downarrow \lambda x.u' \;\Rightarrow\; \exists \lambda x.v' \in C.\, v \Downarrow \lambda x.v' \;\wedge\; \forall c \in C.\, u'[c/x] R v'[c/x]$$

for any pair u and v of programs. Let $\approx$ be the symmetric closure of $\lessapprox$. Before considering rational openness and rationality we need a notion of a fixpoint operator: We shall use the standard eager fixpoint operator $\mathtt{Y}$ which is defined as

$$\mathtt{Y} = \lambda f.(\lambda x.\lambda y.f(xx)y)(\lambda x.\lambda y.f(xx)y)$$

The finite approximants to the fixpoint operator are defined by the stipulations

$$\begin{aligned} \mathtt{Y}^0 &= (\lambda x.xx)(\lambda x.xx) \\ \mathtt{Y}^{n+1} &= \lambda f.f(\lambda y.\mathtt{Y}^n f y) \end{aligned}$$

[2] This example was provided by Søren Bøgh Lassen who has described an analogous situation in [Las96].

One can then show that rational openness in the above mentioned sense is satisfied when considering $\approx$. But rationality does not hold with respect to $\lessapprox$.

An example contradicting rationality goes as follows: We can represent natural numbers using Church numerals; this enables us to define a value `zero` representing zero and a program `succ` representing the successor function, and moreover, we can define a program `min` representing the function which returns the least of two numbers. Now, define a program `?` as

$$\texttt{?} = (\texttt{Y}\lambda f.\lambda x.x \oplus (\texttt{succ}(fx)))\texttt{zero}$$

It is clear that this program has the property that whenever it terminates, the result is a value representing a random number. This has to be compared to the n'th finite approximant to `?` (obtained by replacing `Y` by $\texttt{Y}^n$) which has the property that whenever it terminates, the result is a value representing is a random number less than n. Then two programs g and h are defined as

$$g = \lambda x.\texttt{?} \qquad\qquad h = (\lambda y.\lambda x.\texttt{min}y\texttt{?})\texttt{?}$$

It is straightforward to check that h is an upper bound for all the finite approximants to g. But we do not have $g \lessapprox h$ because `?` may evaluate to a value representing a number greater than it is possible to simulate by a given result of evaluating h. Hence, rationality is not satisfied when considering the applicative simulation preorder. It should, however, be mentioned that the same calculus satisfies rationality, and thus also rational openness, if the applicative simulation preorder is replaced by a contextual preorder.

Chapter 8

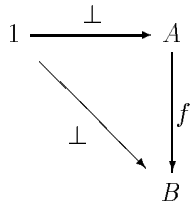
LPCF - Semantic Issues

In this chapter appropriate machinery for giving the categorical model for LPCF is introduced. Section 8.1 introduces a categorical notion of undefinedness appropriate for the linear setting. Categorical notions of numerals and fixpoints appropriate for the linear setting are given in Section 8.2 and Section 8.3, respectively. Also, the axiom of rational openness on a linear fixpoint operator is introduced. In Section 8.4 a notion of rationality appropriate for the linear setting is given, and it is shown how under certain circumstances a rationally open linear fixpoint operator induces a rational linear category.

8.1 Undefinedness

In this section we will introduce a linear version of the notion of undefinedness of Section 6.1.

Definition 8.1.1 A linear category is *pointed* iff a map $\bot_A \colon I \to A$ is given for each object A. A map $f : A \to B$ in a pointed linear category is *strict* iff the diagram



commutes.

Examples 8.1.2 In the categories $\mathbf{cpo}_{st}$ and $\mathbf{dI}_{lin}$ the obvious choice of $\bot$-maps are the appropriate bottom elements; it is then the case that every map is strict.

We will now have a look at a simple result concerning the Kleisli category induced by a linear category.

Proposition 8.1.3 Let $\mathcal{C}$ be a linear category with a terminal object. The bijection between maps $\mathcal{C}(I, A) \cong \mathcal{C}_!(1, A)$ given by composition with the isomorphism $!1 \cong I$ induces a bijective correspondence between families of maps making the linear category $\mathcal{C}$ pointed in the sense of Definition 8.1.1 and families of maps making the Kleisli category pointed in the sense of Definition 6.1.1.

Proof: Obvious. Recall that 1 is a terminal object in the Kleisli category. □

If we have a pointed linear category with finite products, then the Kleisli category can be considered as pointed according to Proposition 8.1.3 and it turns out that if every map in the linear category in question is strict, then the maps π_1 and π_2 in the Kleisli category are strict and the map *eval* is left-strict.

Examples 8.1.4 If this construction is applied to the pointed linear category $\mathbf{cpo}_{st}$, then the resulting choice of $\perp$-maps in the category $\mathbf{cpo}$ is the obvious one, namely the appropriate bottom elements. An analogous remark applies to the category $\mathbf{dI}_{lin}$.

Definition 8.1.5 Let $\mathcal{C}$ be a pointed linear category. For any pair of objects A and B we define a map $\perp_{A,B}: A \to B$ as

$$\perp_{A,B} = \ulcorner \perp_{A \multimap B} \urcorner^{-1}.$$

It can be shown that if every map is strict, then for every object B we have $\perp_{I,B} = \perp_B$. The $\perp$-maps are thus generalised to have arbitrary domains and codomains. We will now prove some results that will be of use later on; the first one says that composition is left-strict:

Proposition 8.1.6 Let $\mathcal{C}$ be a pointed linear category where every map is strict. We have

$$\perp_{A,B}; f = \perp_{A,C}$$

for any map $f : B \to C$.

Proof: Let a map $f : B \to C$ be given. We have

$$\begin{array}{rcl} \perp_{A,B}; f & = & \cong; \lambda^{-1}(\perp_{A \multimap B}); f \\ & = & \cong; \lambda^{-1}(\perp_{A \multimap B}; (A \multimap f)) \\ & = & \cong; \lambda^{-1}(\perp_{A \multimap C}) \\ & = & \perp_{A,C} \end{array}$$

□

The following proposition shows that composition is also right-strict:

Proposition 8.1.7 Let $\mathcal{C}$ be a pointed linear category where every map is strict. We have

$$f; \perp_{B,C} = \perp_{A,C}$$

for any map $f : A \to B$.

Proof: Let a map $f : A \to B$ be given. We have

$$\begin{array}{rcl} f; \perp_{B,C} & = & f; \cong; \lambda^{-1}(\perp_{B \multimap C}) \\ & = & \cong; \lambda^{-1}(\perp_{B \multimap C}; (f \multimap C)) \\ & = & \cong; \lambda^{-1}(\perp_{A \multimap C}) \\ & = & \perp_{A,C} \end{array}$$

□

The following proposition shows that the $\otimes$ functor is strict in both arguments:

Proposition 8.1.8 Let $\mathcal{C}$ be a pointed linear category where every map is strict. We have

$$\perp_{A,B} \otimes f = \perp_{A \otimes C, B \otimes D}$$

and analogously

$$f \otimes \perp_{A,B} = \perp_{C \otimes A, D \otimes B}$$

for any map $f : C \to D$.

Proof: Let a map $f : C \to D$ be given. Consider the map

$$(A \multimap B) \otimes (A \otimes C) \cong ((A \multimap B) \otimes A) \otimes C \xrightarrow{eval \otimes f} B \otimes D.$$

Some equational manipulation shows that $\ulcorner h \otimes f \urcorner = \ulcorner h \urcorner; \lambda(\cong; (eval \otimes f))$ for any map $h : A \to B$, and thus

$$\begin{array}{rcl} \ulcorner \perp_{A,B} \otimes f \urcorner & = & \perp_{A \multimap B}; \lambda(\cong; (eval \otimes f)) \\ & = & \perp_{(A \otimes C) \multimap (B \otimes D)} \end{array}$$

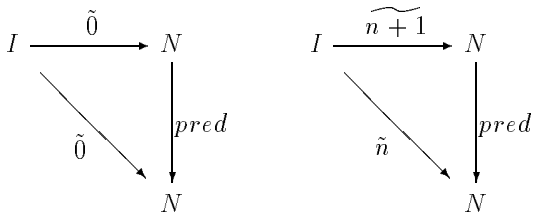
which entails that $\perp_{A,B} \otimes f = \perp_{A\otimes C, B\otimes D}$. □

It should be remarked that the presence of a zero object (an object which is both initial and terminal) in a linear category (an assumption put forward in [Lam96]) also induces a notion of undefinedness in the sense of a family of maps $\perp_{A,B}\colon A \to B$ satisfying the conclusions of Proposition 8.1.6, Proposition 8.1.7 and Proposition 8.1.8. It is straightforward to see that the two approaches coincide whenever a zero object is present in a pointed linear category where every map is strict.

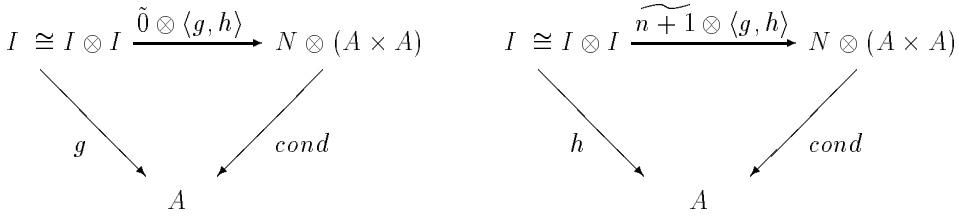
8.2 Linear Numerals

We will now introduce a linear version of the object of numerals of Section 6.2.

Definition 8.2.1 A *linear object of numerals* in a linear category with finite products is an object N equipped with maps $zero : I \to N$ and $succ, pred : N \to N$ such that the diagrams



commute for any number n where the maps $\tilde{n} : I \to N$ are defined in the obvious way using the *zero* and *succ* maps. Furthermore, a map $cond_A : N \otimes (A \times A) \to A$ is given for each object A such that the diagrams



commute for any maps $g, h : I \to A$ and any number n. An object of numerals in a pointed linear category with finite products is *standard* iff for any map $h : I \to N$ we have $h = \perp$ or $h = \tilde{n}$ for some number n.

A comparison between the conditional map

$$cond_A : N \times (A \times A) \to A$$

of an object of numerals in the sense of Definition 6.2.1 and the conditional map

$$cond_A : N \otimes (A \times A) \to A$$

of a linear object of numerals in the sense of Definition 8.2.1 shows how the intuitionistic construct $\times$ splits up into the two linear constructs $\times$ and $\otimes$: Both components of the tensor product $N \otimes (A \times A)$ have to be used, and exactly one component of $A \times A$ has to be used, namely the one determined by the test. Note that if $\mathcal{C}$ is a linear category with finite products and an object of numerals such that $\tilde{0} = \tilde{1}$, then every hom-set $\mathcal{C}(I, A)$ has at most one element.

Examples 8.2.2 In what follows we will give a couple of linear objects of numerals in the concrete categories $\mathbf{cpo}_{st}$ and $\mathbf{dI}_{lin}$. The obvious choice of a standard linear object of numerals in these

categories is to take the object N to be equal to ω with a bottom element adjoined, and equipped with the maps

$$zero(x) = \begin{cases} \perp & \text{if } x = \perp \\ 0 & \text{otherwise} \end{cases}$$

$$succ(x) = \begin{cases} \perp & \text{if } x = \perp \\ x+1 & \text{otherwise} \end{cases} \qquad pred(x) = \begin{cases} \perp & \text{if } x = \perp \\ 0 & \text{if } x = 0 \\ x \Leftrightarrow 1 & \text{otherwise} \end{cases}$$

$$cond_A(x,(y,z)) = \begin{cases} y & \text{if } x = 0 \\ z & \text{otherwise} \end{cases}$$

Note that the finite products of $\mathbf{dI}_{lin}$ coincide with the ones of $\mathbf{cpo}_{st}$, and, moreover, given the object N here, the part of the monoidal structure on $\mathbf{dI}_{lin}$ involved in Definition 8.2.1 coincides (up to isomorphism) with the corresponding structure on $\mathbf{cpo}_{st}$.

Another option is to take the object N to be equal to ω with a bottom element adjoined together with an element ∞ incompatible with any other element but bottom, and equipped with the maps from the standard object of numerals extended as follows

$$succ(\infty) = \infty \qquad pred(\infty) = \infty \qquad cond_A(\infty,(y,z)) = z$$

Also in this case the relevant structure on $\mathbf{dI}_{lin}$ coincides (up to isomorphism) with the corresponding structure on $\mathbf{cpo}_{st}$. This gives a non-standard object of numerals in $\mathbf{cpo}_{st}$ as well as in $\mathbf{dI}_{lin}$.

Given a linear object of numerals in a linear category with finite products, the object N together with the maps

$$!1 \cong I \xrightarrow{zero} N$$

$$!N \xrightarrow{\varepsilon} N \xrightarrow{succ} N \qquad !N \xrightarrow{\varepsilon} N \xrightarrow{pred} N$$

$$!(N \times (A \times A)) \cong !N \otimes !(A \times A) \xrightarrow{\varepsilon \otimes \varepsilon} N \otimes (A \times A) \xrightarrow{cond} A$$

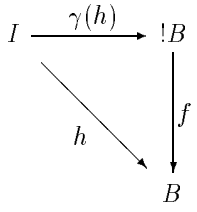
constitutes an object of numerals in the Kleisli category. This can be verified by straightforward equational manipulation. If the linear category in question is pointed, then the Kleisli category can be considered as pointed according to Proposition 8.1.3, and it turns out that if every map in the linear category is strict, then the maps *succ* and *pred* in the Kleisli category are strict and the map *cond* is left-strict.

Examples 8.2.3 If this construction is applied to any of the two linear objects of numerals in the linear category $\mathbf{cpo}_{st}$, then the resulting choice of object of numerals in the category $\mathbf{cpo}$ is the corresponding one mentioned in Section 6.2. An analogous remark applies to the category $\mathbf{dI}_{lin}$.

8.3 Linear Fixpoints

We will now consider fixpoints in a linear context.

Definition 8.3.1 Let $\mathcal{C}$ be a linear category. A map $h : I \to B$ is a *linear fixpoint* of a map $f : !B \to B$ iff the diagram



commutes. A *linear fixpoint operator* for an object B is an operation on maps

$$(\Leftrightarrow)^\sharp_B : \mathcal{C}(!B, B) \Leftrightarrow\!\!\!\rightarrow \mathcal{C}(I, B)$$

such that $f^\sharp$ is a linear fixpoint of f for any map $f : !B \to B$.

The definition of a linear fixpoint operator in a linear category $\mathcal{C}$ is essentially the definition of a fixpoint operator in the category of coalgebras stated in terms of maps in $\mathcal{C}$. To be precise:

Proposition 8.3.2 Let $\mathcal{C}$ be a linear category. A map $h : I \to B$ is a linear fixpoint of a map $f : !B \to B$ in $\mathcal{C}$ iff $\gamma(h) : (I, m_I) \to (!B, \delta)$ is a fixpoint of $\gamma(f) : (!B, \delta) \to (!B, \delta)$ in the category of coalgebras. There is a bijective correspondence between linear fixpoint operators for the object B in $\mathcal{C}$ and fixpoint operators for the free coalgebra $(!B, \delta)$ in the category of coalgebras.

Proof: Recall that the coalgebra (I, m_I) is a terminal object in the category of coalgebras. The first assertion follows from the observation that $\gamma(h) = \gamma(h); \gamma(f)$ is equivalent to $h = \gamma(h); f$. The last assertion follows from the first assertion. □

We will now have a look at a result concerning the Kleisli category induced by a linear category; the definition of a linear fixpoint operator in a linear category $\mathcal{C}$ with a terminal object is essentially the definition of a fixpoint operator in the Kleisli category stated in terms of maps in $\mathcal{C}$. To be precise:

Proposition 8.3.3 Let $\mathcal{C}$ be a linear category with a terminal object. The bijection between maps $\mathcal{C}(I, B) \cong \mathcal{C}_!(1, B)$ given by composition with the isomorphism $!1 \cong I$ induces a bijective correspondence between linear fixpoint operators for an object B in $\mathcal{C}$ and fixpoint operators for the object B in the Kleisli category.

Proof: Straightforward. Recall that 1 is a terminal object in the Kleisli category. □

Examples 8.3.4 In the categories **cpo** and **dI** there is an obvious choice of fixpoint operator for every object; this induces a linear fixpoint operator for every object in the linear categories $\mathbf{cpo}_{st}$ and $\mathbf{dI}_{lin}$.

We now need a convention: Given a map $g : !B \to B$ we define a map $g^n : !B \to B$ for every number n by stipulating $g^0 = \varepsilon$ and $g^{n+1} = \gamma(g^n); g$. The map g^n is simply n iterations of g considered as a map in the Kleisli category.

Definition 8.3.5 A linear fixpoint operator for an object B in a pointed linear category is *rationally open* with respect to an object P iff for all maps $f : !B \to B$ and $g : !B \to P$ it is the case that

$$\gamma(f^\sharp); g \neq \perp \;\Rightarrow\; \exists n \in \omega.\, \gamma(\gamma(\perp); f^n); g \neq \perp .$$

The definition of rational openness of a linear fixpoint operator in a pointed linear category $\mathcal{C}$ with a terminal object is essentially the definition of rational openness of the corresponding fixpoint operator in the Kleisli category stated in terms of maps in $\mathcal{C}$. To be precise:

Proposition 8.3.6 Assume that we have a linear fixpoint operator for an object B in a pointed linear category with a terminal object. Then the linear fixpoint operator for the object B in $\mathcal{C}$ is rationally open with respect to an object P in the sense of Definition 8.3.5 iff the corresponding fixpoint operator for the object B in the Kleisli category is rationally open with respect to the object P in the sense of Definition 6.3.3.

Proof: Recall that 1 is a terminal object in the Kleisli category and it can be considered as pointed according to Proposition 8.1.3, and, moreover, the linear fixpoint operator for the object B in $\mathcal{C}$ corresponds to a fixpoint operator for the object B in the Kleisli category according to Proposition 8.3.3. The result follows from straightforward calculation. □

Examples 8.3.7 It follows from Proposition 8.3.6 that the obvious linear fixpoint operators in the linear categories $\mathbf{cpo}_{st}$ and $\mathbf{dI}_{lin}$ are rationally open with respect to any objects.

Definition 8.3.8 Let $\mathcal{C}$ be a linear category. A map $h : !X \to B$ is a *parametrised linear fixpoint* of a map $f : !X \otimes !B \to B$ iff the diagram

$$\begin{array}{ccc} !X & \xrightarrow{d} & !X \otimes !X \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle id \otimes \gamma(h)} \\ B & \xleftarrow{f} & !X \otimes !B \end{array}$$

commutes. A *parametrised linear fixpoint operator* for an object B is an operation on maps

$$(\Leftrightarrow)^{\sharp}_{X,B} : \mathcal{C}(!X \otimes !B, B) \Leftrightarrow \mathcal{C}(!X, B)$$

for each object X such that $f^{\sharp}$ is a parametrised linear fixpoint of f for any map $f : !X \otimes !B \to B$. A parametrised linear fixpoint operator is *natural* if the operations are natural in $!X$ with respect to maps of free coalgebras.

Recall that maps of free coalgebras are the same as maps in the image of γ. Note how the "duplicate" map $d_X : !X \to !X \otimes !X$ is used to copy parameters. We can internalise the notion of a linear fixpoint operator using the closed structure of a linear category.

Definition 8.3.9 An *internal linear fixpoint operator* for an object B in a linear category is a map $Y_B : !(!B \multimap B) \to B$ such that the diagram

$$\begin{array}{ccc} !(!B \multimap B) & \xrightarrow{d} & !(!B \multimap B) \otimes !(!B \multimap B) \\ {\scriptstyle Y}\downarrow & & \downarrow{\scriptstyle \varepsilon \otimes \gamma(Y)} \\ B & \xleftarrow{eval} & (!B \multimap B) \otimes !B \end{array}$$

commutes.

The definition of an internal linear fixpoint operator in a linear category $\mathcal{C}$ is essentially the definition of an internal fixpoint operator in the category of coalgebras stated in terms of maps in $\mathcal{C}$. To be precise:

Proposition 8.3.10 Let $\mathcal{C}$ be a linear category. A map $Y : !(!B \multimap B) \to B$ is an internal linear fixpoint operator for the object B in $\mathcal{C}$ iff $\gamma(Y) : (!(!B \multimap B), \delta) \to (!B, \delta)$ is an internal fixpoint operator for the object $(!B, \delta)$ in the category of coalgebras.

Proof: Recall that the category of coalgebras has a cartesian structure such that for every object B the free coalgebra $(!B, \delta)$ is exponentiable; the internal-hom object of a coalgebra (A, h) and the free coalgebra $(!B, \delta)$ is given by the free coalgebra $(!(A \multimap B), \delta)$. Now

$$\gamma(Y) = \Delta; (id \times \gamma(Y)); eval$$

in $\mathcal{C}^!$ is equivalent to

$$Y = \Delta; (id \times \gamma(Y)); eval; \varepsilon$$

in $\mathcal{C}$, and the calculation

$$\begin{array}{rcll} \Delta; (id \times \gamma(Y)); eval; \varepsilon & = & d; (id \otimes \gamma(Y)); \gamma(\lambda^{-1}(\gamma^{-1}(id))); \varepsilon & \text{by def. of op. in } \mathcal{C}^! \\ & = & d; (id \otimes \gamma(Y)); (\varepsilon \otimes id); eval & \\ & = & d; (\varepsilon \otimes \gamma(Y)); eval & \end{array}$$

shows that this is equivalent to $Y = d;(\varepsilon \otimes \gamma(Y));eval$. □

One can show that a linear category has a parametrised linear fixpoint operator for each object iff it has an internal linear fixpoint operator for each object, but we will show here a more informative result:

Proposition 8.3.11 Let B be an object in a linear category. There is a bijective correspondence between natural parametrised linear fixpoint operators and internal linear fixpoint operators for the object B.

Proof: Assume that the category has a natural parametrised linear fixpoint operator $(\Leftrightarrow)^\sharp$ for an object B, and define the map Y by the equation

$$Y = ((\varepsilon \otimes id);eval)^\sharp .$$

It is clearly an internal linear fixpoint operator for B cf. the diagram in Definition 8.3.8. Conversely, assume that Y is an internal linear fixpoint operator for the object B, and define an operation on maps

$$(\Leftrightarrow)^\sharp = \gamma(\lambda(\Leftrightarrow));Y$$

for each object X. This is a parametrised linear fixpoint operator because

$$\begin{array}{rcl} \gamma(\lambda(f));Y & = & \gamma(\lambda(f));d;(\varepsilon \otimes \gamma(Y));eval \\ & = & d;(id \otimes \gamma(\gamma(\lambda(f));Y));f \end{array}$$

for any map $f : !X \otimes !B \to B$. The operations are natural with respect to maps in the image of γ because we have

$$\begin{array}{rcl} \gamma(h);\gamma(\lambda(f));Y & = & \gamma(\gamma(h);\lambda(f));Y \\ & = & \gamma(\lambda((\gamma(h) \otimes id);f));Y \end{array}$$

for any maps $f : !X \otimes !B \to B$ and $h : !X' \to X$.

The construction of internal linear fixpoint operators from natural parametrised linear fixpoint operators is injective because we have

$$\begin{array}{rcl} \gamma(\lambda(f));((\varepsilon \otimes id);eval)^\sharp & = & ((\gamma(\lambda(f)) \otimes id);(\varepsilon \otimes id);eval)^\sharp \\ & = & f^\sharp \end{array}$$

for any map $f : !X \otimes !B \to B$, which entails that the natural parametrised linear fixpoint operator induced by the internal linear fixpoint operator $((\varepsilon \otimes id);eval)^\sharp$ is equal to $(\Leftrightarrow)^\sharp$. The construction of natural parametrised linear fixpoint operators from internal linear fixpoint operators is injective because the following calculation

$$\begin{array}{rcl} \gamma(\lambda((\varepsilon \otimes id);eval));Y & = & \gamma(\lambda(\lambda^{-1}(\varepsilon)));Y \\ & = & \gamma(\varepsilon);Y \\ & = & Y \end{array}$$

shows that the internal linear fixpoint operator induced by the natural parametrised linear fixpoint operator $\gamma(\lambda(\Leftrightarrow));Y$ is equal to Y. □

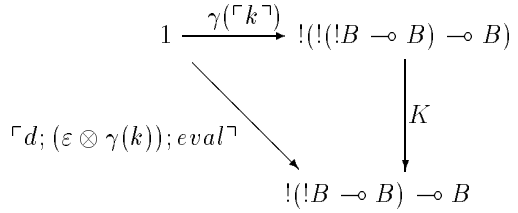
The following result can be obtained from Proposition 8.3.2, Proposition 6.3.9 and Proposition 8.3.10, but we will also give a direct proof:

Proposition 8.3.12 A linear fixpoint operator for an object $!(!B \multimap B) \multimap B$ in a linear category induces an internal linear fixpoint operator for B.

Proof: First define a map

$$!(!(!B \multimap B) \multimap B) \xrightarrow{K_B} !(!B \multimap B) \multimap B$$

as the exponential transpose of the map given in Figure 8.1. For every map $k : !(!B \multimap B) \to B$ the diagram



commutes, which can be shown by some equational manipulation; this entails that $\ulcorner k \urcorner$ is a linear fixpoint for K iff $k = d; (\varepsilon \otimes \gamma(k)); eval$. We now define the internal linear fixpoint operator $Y : !(!B \multimap B) \to B$ by the equation $\ulcorner Y \urcorner = K^\sharp$; hence $Y = d; (\varepsilon \otimes \gamma(Y)); eval$. □

This entails that if any object in a linear category has a linear fixpoint operator, then any object has an internal linear fixpoint operator too. Note that in the context of Proposition 8.3.12 the map $\gamma(\ulcorner f \urcorner); Y$ is a linear fixpoint for f for every map $f : !B \to B$, and, moreover, if the linear category in question is pointed such that every map is strict, then it can be shown by a small induction proof that

$$\cong; ((\gamma(\bot); K^n) \otimes \gamma(\ulcorner f \urcorner)); eval = \gamma(\bot); f^n.$$

So indeed the map K does work as expected.

8.4 Linear Rationality

In this section we will show how under appropriate circumstances a rationally open linear fixpoint operator induces a rational linear category; this is analogous to what is done in Section 6.5 in the context of a cartesian closed category. In [Bra97a] a rational linear category is taken to be an appropriate order-theoretic notion of an adequate categorical model for a fragment of LPCF.

Definition 8.4.1 A *rational linear category* is a pointed linear category which is **poset**-enriched as a linear category such that for every object A it is the case that

- the map $\bot_A$ is least,
- for every map $f : !A \to A$ the increasing chain $\{\gamma(\bot); f^n\}_{n \in \omega}$ has a least upper bound $f^\sharp$ with the property that for any map $g : !A \to D$ the map $\gamma(f^\sharp); g$ is a least upper bound for the increasing chain $\{\gamma(\gamma(\bot); f^n); g\}_{n \in \omega}$.

Examples 8.4.2 The linear categories $\mathbf{cpo}_{st}$ and $\mathbf{dI}_{lin}$ are rational in the obvious way.

We will now consider a notion of observables for a pointed linear category that assumes the presence of a distinguished object P. It will be called the intuitionistic termination notion of observables because it corresponds to Definition 6.5.3 in a sense made precise by Proposition 8.4.5 below.

Definition 8.4.3 Let $\mathcal{C}$ be a pointed linear category with a distinguished object P. The *intuitionistic termination notion of observables* is defined by assigning the set

$$\mathcal{O}_A = \{\mathcal{O}_h \mid h \in \mathcal{C}(!A, P)\}$$

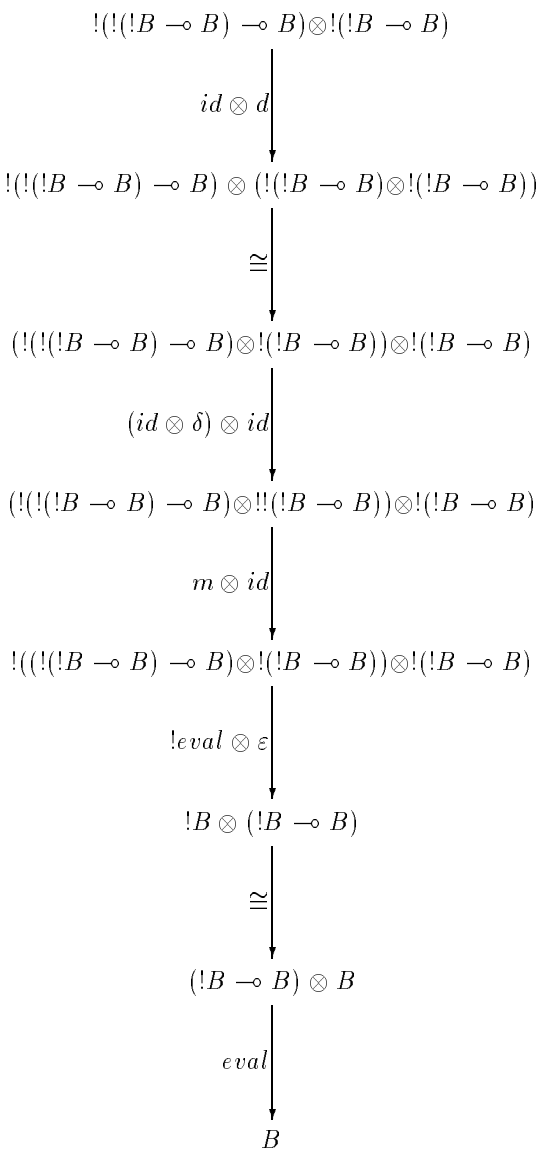
to any object A where

$$\mathcal{O}_h = \{f \in \mathcal{C}(I, A) \mid \gamma(f); h \neq \bot\}.$$

This is a notion of observables because $g^* \mathcal{O}_h = \mathcal{O}_{!g;h}$ for any maps $g : B \to A$ and $h : !A \to P$. The induced observational preorder can be stated explicitly as

$$f \lesssim g \quad \text{iff} \quad \forall h \in \mathcal{C}(!(A \multimap B), P).\ \gamma(\ulcorner f \urcorner); h \neq \bot \ \Rightarrow\ \gamma(\ulcorner g \urcorner); h \neq \bot$$

Figure 8.1: The map used in Proposition 8.3.12



for any maps $f, g : A \to B$. We will refer to this preorder as the *intuitionistic termination preorder*.

Note that the map $\bot_P$ is minimal with respect to the intuitionistic termination preorder. It follows from Proposition 6.4.4 that the symmetric monoidal closed structure on $\mathcal{C}$ is **preorder**-enriched with respect to the intuitionistic termination preorder. Moreover, we have the following result:

Proposition 8.4.4 Let $\mathcal{C}$ be a pointed linear category with a distinguished object P. The functor $!$ is **preorder**-enriched with respect to the intuitionistic termination preorder.

Proof: Consider the map

$$!(A \multimap B) \otimes !A \xrightarrow{m} !((A \multimap B) \otimes A) \xrightarrow{!eval} !B$$

the transpose of which

$$!(A \multimap B) \xrightarrow{\lambda(m;!eval)} !A \multimap !B$$

has the property that $\gamma(\ulcorner k \urcorner); \lambda(m, !eval) = \ulcorner !k \urcorner$ for any map $k : A \to B$, which shows the wanted result. □

This entails that $\mathcal{C}$ is **preorder**-enriched (as a linear category) with respect to the intuitionistic termination preorder. If $\leq$ is another **preorder**-enrichment on $\mathcal{C}$ (as a linear category) with respect to which the map $\bot_P$ is minimal, then $\leq$ can be shown to be included in $\lesssim$ in the sense that for any objects A and B the preorder $\leq_{A,B}$ on $\mathcal{C}(A, B)$ is included in the $\lesssim_{A,B}$ preorder. The definition of the intuitionistic termination preorder on a hom-set of a pointed linear category $\mathcal{C}$ with finite products and a distinguished object P is essentially the definition of the termination preorder on the corresponding hom-set in the Kleisli category stated in terms of maps in $\mathcal{C}$. To be precise:

Proposition 8.4.5 Let $\mathcal{C}$ be a pointed linear category with finite products and a distinguished object P. The bijection between maps

$$\mathcal{C}(I, A) \cong \mathcal{C}_!(1, A)$$

is an isomorphism of preorders when we consider the intuitionistic termination preorder on $\mathcal{C}(I, A)$ in the sense of Definition 8.4.3 and the termination preorder on $\mathcal{C}_!(1, A)$ in the sense of Definition 6.5.3.

Proof: Recall that the Kleisli category is cartesian closed, and it can be considered as pointed according to Proposition 8.1.3. The result follows from straightforward calculation. □

If we are dealing with a pointed linear category with finite products and a linear object of numerals, then we take the object P to be N unless otherwise stated.

Examples 8.4.6 In the linear categories $\mathbf{cpo}_{st}$ and $\mathbf{dI}_{lin}$ the intuitionistic termination preorder coincides with the given order.

Theorem 8.4.7 Let $\mathcal{C}$ be a pointed linear category with finite products such that every map is strict. Assume that the category is equipped with a standard linear object of numerals, and moreover, assume that a linear fixpoint operator is given for each object. Then for any object A it is the case that

- the map $\bot_A$ is least and for any maps $f : !A \to A$ and $g : !A \to D$ the map $\gamma(f^\sharp); g$ is an upper bound for the increasing chain $\{\gamma(\gamma(\bot); f^n); g\}_{n \in \omega}$,
- the linear fixpoint operator for the object A is rationally open with respect to N iff for any maps $f : !A \to A$ and $g : !A \to D$ the map $\gamma(f^\sharp); g$ is a least upper bound for the increasing chain $\{\gamma(\gamma(\bot); f^n); g\}_{n \in \omega}$,

where we consider the intuitionistic termination preorder.

Proof: Recall that the Kleisli category is cartesian closed, and it can be considered as pointed according to Proposition 8.1.3 and equipped with a standard object of numerals such that the map *cond* is left-strict according to Section 8.2, and, moreover, a fixpoint operator is given for each object in the Kleisli category according to Proposition 8.3.3. So we apply Theorem 6.5.5 and restate the two assertions of the result appropriately using Proposition 8.4.5. □

Note that in the context of Theorem 8.4.7 none of the maps in the hom-set $\mathcal{C}(I, N)$ are equivalent with respect to the equivalence relation induced by the intuitionistic termination preorder.

Corollary 8.4.8 Let $\mathcal{C}$ be a pointed linear category with finite products such that every map is strict. Assume that the category is equipped with a standard linear object of numerals, and, moreover, assume that a linear fixpoint operator is given for each object. Then the quotient $\widehat{\mathcal{C}}$ is a rational linear category when $\mathcal{C}$ is considered as **preorder**-enriched with respect to the intuitionistic termination preorder.

Proof: Follows from Theorem 8.4.7 and the observation that the linear fixpoint operator is a congruence with respect to the equivalence relation induced by the intuitionistic termination preorder. □

Above we have considered a notion of observables that gave rise to the intuitionistic termination preorder. We will now have a look at another notion of observables that assumes the presence of finite products and a linear object of numerals but does not assume the linear category in question to be pointed. It will be called the intuitionistic termination to value notion of observables because it corresponds to Definition 6.5.8 in a sense made precise by Proposition 8.4.11 below.

Definition 8.4.9 Let $\mathcal{C}$ be a linear category with finite products and a linear object of numerals. The *intuitionistic termination to value notion of observables* is defined by assigning the set

$$\mathcal{O}_A = \{\mathcal{O}_{h,n} \mid h \in \mathcal{C}(!A, N), n \in \omega\}$$

to any object A where

$$\mathcal{O}_{h,n} = \{f \in \mathcal{C}(1, A) \mid \gamma(f); h = \tilde{n}\}.$$

This is a notion of observables because $g^*\mathcal{O}_{h,n} = \mathcal{O}_{(!g;h),n}$ for any maps $g : B \to A$ and $h : !A \to N$. The induced observational preorder can be stated explicitly as

$$f \lesssim g \quad \text{iff} \quad \forall h \in \mathcal{C}(!(A \multimap B), N).\forall n \in \omega.\ \gamma(\ulcorner f \urcorner); h = \tilde{n} \ \Rightarrow\ \gamma(\ulcorner g \urcorner); h = \tilde{n}$$

for any maps $f, g : A \to B$. We will refer to this preorder as the *intuitionistic termination to value preorder*.

Note that for any number p the map $\tilde{p}$ is maximal with respect to the intuitionistic termination to value preorder. It follows from Proposition 6.4.4 that the symmetric monoidal closed structure on $\mathcal{C}$ is **preorder**-enriched with respect to the intuitionistic termination to value preorder. Moreover we have the following result:

Proposition 8.4.10 Let $\mathcal{C}$ be a linear category with finite products and a linear object of numerals. The functor ! is **preorder**-enriched with respect to the intuitionistic termination to value preorder.

Proof: Analogous to the proof of Theorem 8.4.4. □

This entails that $\mathcal{C}$ is **preorder**-enriched (as a linear category) with respect to the intuitionistic termination to value preorder. If $\leq$ is another **preorder**-enrichment on $\mathcal{C}$ (as a linear category) with respect to which $\tilde{p}$ is maximal for every number p, then $\leq$ can be shown to be included in $\lesssim$. The definition of the intuitionistic termination to value preorder on a hom-set of a linear category $\mathcal{C}$ with finite products and a linear object of numerals is essentially the definition of the termination preorder to value on the corresponding hom-set in the Kleisli category stated in terms of maps in $\mathcal{C}$. To be precise:

Proposition 8.4.11 Let $\mathcal{C}$ be a linear category with finite products and a linear object of numerals. The bijection between maps

$$\mathcal{C}(I, A) \cong \mathcal{C}_!(1, A)$$

is an isomorphism of preorders when we consider the intuitionistic termination to value preorder on $\mathcal{C}(I, A)$ in the sense of Definition 8.4.9 and the termination to value preorder on $\mathcal{C}_!(1, A)$ in the sense of Definition 6.5.8.

Proof: Recall that the Kleisli category is cartesian closed, and it can be considered as equipped with an object of numerals according to Section 8.2. The result follows from straightforward calculation. □

Examples 8.4.12 In the categories $\mathbf{cpo}_{st}$ and $\mathbf{dI}_{lin}$ the intuitionistic termination to value preorder coincides with the given order.

If the linear category giving rise to the intuitionistic termination to value preorder is pointed such that every map is strict and the linear object of numerals is standard, then it can be shown that the map $\tilde{p}$ is maximal with respect to the intuitionistic termination preorder for every number p, and, moreover, the map $\bot_N$ is minimal with respect to the intuitionistic termination to value preorder, so we conclude that under such circumstances the two preorders coincide. This is thus also the case in the context of Theorem 8.4.7, so this result could equally well have been stated in terms of the intuitionistic termination to value preorder instead of the intuitionistic termination preorder.

For completeness we will have a brief look at yet another couple of notions of observables. It will, however, turn out that the ! functor does not necessarily enrich with respect to the induced observational preorders. The first additional notion of observables will be called the linear termination notion of observables. Compare with Definition 8.4.3.

Definition 8.4.13 Let $\mathcal{C}$ be a pointed linear category with a distinguished object P. The *linear termination notion of observables* is defined by assigning the set

$$\mathcal{O}_A = \{\mathcal{O}_h \mid h \in \mathcal{C}(A, P)\}$$

to any object A where

$$\mathcal{O}_h = \{f \in \mathcal{C}(I, A) \mid f; h \neq \bot\}.$$

This is a notion of observables because $g^* \mathcal{O}_h = \mathcal{O}_{g;h}$ for any maps $g : B \to A$ and $h : A \to P$. The induced observational preorder can be stated explicitly as

$$f \lesssim g \quad \text{iff} \quad \forall h \in \mathcal{C}(A \multimap B, P).\ \ulcorner f \urcorner; h \neq \bot \ \Rightarrow\ \ulcorner g \urcorner; h \neq \bot$$

for any maps $f, g : A \to B$. We will refer to this preorder as the *linear termination preorder*. We have the following result:

Proposition 8.4.14 Let $\mathcal{C}$ be a pointed linear category with a distinguished object P. The functor ! is **preorder**-enriched with respect to the linear termination preorder iff the linear termination preorder coincides with the intuitionistic termination preorder.

Proof: The "if" direction follows from Proposition 8.4.4. To prove the "only if" direction, assume that the functor ! is **preorder**-enriched with respect to the linear termination preorder. It is clear that the intuitionistic termination preorder is included in the linear termination preorder. The converse to this inclusion follows from the observation that for any maps $f, g : I \to B$ we have $f \lesssim g$ with respect to the intuitionistic termination preorder iff we have $\gamma(f) \lesssim \gamma(g)$ with respect to the linear termination preorder. □

Examples 8.4.15 In the linear categories $\mathbf{cpo}_{st}$ and $\mathbf{dI}_{lin}$ the linear termination preorder coincides with the given order.

The second additional notion of observables will be called the linear termination to value notion of observables. Compare with Definition 8.4.9.

Definition 8.4.16 Let $\mathcal{C}$ be a linear category with finite products and a linear object of numerals. The *linear termination to value notion of observables* is defined by assigning the set

$$\mathcal{O}_A = \{\mathcal{O}_{h,n} \mid h \in \mathcal{C}(A, N), n \in \omega\}$$

to any object A where

$$\mathcal{O}_{h,n} = \{f \in \mathcal{C}(I, A) \mid f; h = \tilde{n}\}.$$

This is a notion of observables because $g^*\mathcal{O}_{h,n} = \mathcal{O}_{(g;h),n}$ for any maps $g : B \to A$ and $h : !A \to N$. The induced observational preorder can be stated explicitly as

$$f \lesssim g \quad \text{iff} \quad \forall h \in \mathcal{C}(A \multimap B, N).\forall n \in \omega.\ \ulcorner f \urcorner; h = \tilde{n} \ \Rightarrow\ \ulcorner g \urcorner; h = \tilde{n}$$

for any maps $f, g : A \to B$. We will refer to this preorder as the *linear termination to value preorder*. We have the following result:

Proposition 8.4.17 Let $\mathcal{C}$ be a linear category with finite products and a linear object of numerals. The functor ! is **preorder**-enriched with respect to the linear termination to value preorder iff the linear termination to value preorder coincides with the intuitionistic termination to value preorder.

Proof: Analogous to the proof of Theorem 8.4.14. □

Examples 8.4.18 In the categories $\mathbf{cpo}_{st}$ and $\mathbf{dI}_{lin}$ the linear termination to value preorder coincides with the given order.

Chapter 9

The Programming Language LPCF

This chapter introduces the programming language LPCF, which is a linear version of PCF. The purpose of LPCF is to give a linear account of computable functions. The syntax is introduced in Section 9.1 and an eager and a lazy operational semantics is given in Section 9.2 and Section 9.3, respectively. The choice of evaluation rules for terms of certain types is motivated by interpretation in an appropriate categorical model; in the case of terms of ! types this dictates evaluation rules which are different from the ones given in [Abr90, Abr93]. A sound categorical interpretation for LPCF is given in Section 9.4. In Section 9.5 we introduce a generalisation of LPCF needed for technical reasons: It enables us to prove adequacy and it enables us to state and prove an unwinding theorem. In Section 9.6 the generalised version of LPCF is given a categorical semantics. In Section 9.7 additional non-order-theoretic axioms are imposed on the categorical model with the aim of proving an adequacy result with respect to the eager operational semantics. The essential ingredient of the categorical model is a rationally open linear fixpoint operator. In Section 9.8 adequacy is proved with respect to the lazy operational semantics. In Section 9.9 observable types are discussed. It is shown that the choice of evaluation strategy does not matter for observable behaviour of programs of observable types. In Section 9.10 we prove an unwinding theorem for LPCF using adequacy; this enables us to show that a restricted version of our axiom of rational openness is not only sufficient, but also necessary for the interpretation to be adequate.

9.1 Syntax

The programming language LPCF is an extension of the linear λ-calculus with numerals and recursion, appropriate for the linear context, where the usual reduction rules are replaced by a Martin-Löf style operational semantics. The first operational semantics for the linear λ-calculus was introduced in [Abr90], and numerals and recursion were added in [Mac91]. We shall consider an eager and a lazy operational semantics in Section 9.7 and Section 9.8, respectively. Types of LPCF are given by the grammar

$$s \ ::= \ N \mid I \mid s \otimes s \mid s \multimap s \mid !s \mid 1 \mid s \times s \mid 0 \mid s + s$$

Figure 9.1: Type Assignment Rules for LPCF

$$\frac{}{x : A \Vdash x : A}$$

$$\frac{, , x : A, y : B, \Delta \Vdash u : C}{, , y : B, x : A, \Delta \Vdash u : C}$$

$$\frac{}{\Vdash \mathtt{zero} : N} \qquad \frac{, \Vdash u : N}{, \Vdash \mathtt{succ}(u) : N} \qquad \frac{, \Vdash u : N}{, \Vdash \mathtt{pred}(u) : N}$$

$$\frac{\Lambda \Vdash u : N \qquad , \Vdash v : A \qquad , \Vdash w : A}{\Lambda, , \Vdash \mathtt{if}\ u\ \mathtt{then}\ v\ \mathtt{else}\ w : A}$$

$$\frac{}{\Vdash * : I} \qquad \frac{\Lambda \Vdash w : I \qquad , \Vdash u : A}{, , \Lambda \Vdash \mathtt{let}\ w\ \mathtt{be}\ *\ \mathtt{in}\ u : A}$$

$$\frac{, \Vdash u : A \qquad \Delta \Vdash v : B}{, , \Delta \Vdash u \otimes v : A \otimes B} \qquad \frac{\Lambda \Vdash w : A \otimes B \qquad , , x : A, y : B \Vdash u : C}{, , \Lambda \Vdash \mathtt{let}\ w\ \mathtt{be}\ x \otimes y\ \mathtt{in}\ u : C}$$

$$\frac{, , x : A \Vdash u : B}{, \Vdash \lambda x^A.u : A \multimap B} \qquad \frac{\Lambda \Vdash f : A \multimap B \qquad \Delta \Vdash u : A}{\Lambda, \Delta \Vdash fu : B}$$

$$\frac{, _1 \Vdash w_1 : !A_1, \quad \ldots \quad , , _n \Vdash w_n : !A_n \qquad x_1 : !A_1, \ldots, x_n : !A_n \Vdash u : B}{, _1, \ldots, , _n \Vdash \mathtt{promote}\ w_1, \ldots, w_n\ \mathtt{for}\ x_1, \ldots, x_n\ \mathtt{in}\ u : !B} \qquad \frac{\Lambda \Vdash u : !B}{\Lambda \Vdash \mathtt{derelict}(u) : B}$$

$$\frac{\Lambda \Vdash w : !A \qquad , , x : !A, y : !A \Vdash u : B}{, , \Lambda \Vdash \mathtt{copy}\ w\ \mathtt{as}\ x, y\ \mathtt{in}\ u : B} \qquad \frac{\Lambda \Vdash w : !A \qquad , \Vdash u : B}{, , \Lambda \Vdash \mathtt{discard}\ w\ \mathtt{in}\ u : B}$$

$$\frac{, \Vdash u : A \qquad , \Vdash v : B}{, \Vdash (u, v) : A \times B} \qquad \frac{\Lambda \Vdash u : A \times B}{\Lambda \Vdash \mathtt{fst}(u) : A} \qquad \frac{\Lambda \Vdash u : A \times B}{\Lambda \Vdash \mathtt{snd}(u) : B}$$

$$\frac{, _1 \Vdash w_1 : A_1, \quad \ldots \quad , , _n \Vdash w_n : A_n}{, _1, \ldots, , _n \Vdash \mathtt{true}(w_1, \ldots, w_n) : 1} \qquad \frac{, _1 \Vdash w_1 : A_1, \quad \ldots \quad , , _n \Vdash w_n : A_n \qquad \Lambda \Vdash u : 0}{, _1, \ldots, , _n, \Lambda \Vdash \mathtt{false}^C(w_1, \ldots, w_n; u) : C}$$

$$\frac{, \Vdash u : A}{, \Vdash \mathtt{inl}^{A+B}(u) : A + B} \qquad \frac{, \Vdash u : B}{, \Vdash \mathtt{inr}^{A+B}(u) : A + B}$$

$$\frac{\Lambda \Vdash w : A + B \qquad , , x : A \Vdash u : C \qquad , , y : B \Vdash v : C}{, , \Lambda \Vdash \mathtt{case}\ w\ \mathtt{of}\ \mathtt{inl}(x).u \mid \mathtt{inr}(y).v : C}$$

$$\frac{}{\Vdash \Omega_A : A} \qquad \frac{}{\Vdash \mathtt{Y}_A : !(!A \multimap A) \multimap A}$$

and terms are given by the grammar

$$
\begin{array}{rcl}
t & ::= & x \mid \\
& & \mathtt{zero} \mid \mathtt{succ}(t) \mid \mathtt{pred}(t) \mid \mathtt{if}\ t\ \mathtt{then}\ t\ \mathtt{else}\ t \mid \\
& & * \mid \mathtt{let}\ t\ \mathtt{be}\ *\ \mathtt{in}\ t \mid t \otimes t \mid \mathtt{let}\ t\ \mathtt{be}\ x \otimes y\ \mathtt{in}\ t \mid \\
& & \lambda x^A.t \mid tt \mid \\
& & \mathtt{promote}\ t, ..., t\ \mathtt{for}\ x_1, ..., x_n\ \mathtt{in}\ t \mid \mathtt{derelict}(t) \mid \\
& & \mathtt{discard}\ t\ \mathtt{in}\ t \mid \mathtt{copy}\ t\ \mathtt{as}\ x, y\ \mathtt{in}\ t \mid \\
& & \mathtt{true}(t, ..., t) \mid (t, t) \mid \mathtt{fst}(t) \mid \mathtt{snd}(t) \mid \\
& & \mathtt{false}^C(t, ..., t; t) \mid \mathtt{inl}^{A+B}(t) \mid \mathtt{inr}^{A+B}(t) \mid \mathtt{case}\ t\ \mathtt{of}\ \mathtt{inl}(x).t \mid \mathtt{inr}(y).t \\
& & \Omega_A \mid \mathtt{Y}_A
\end{array}
$$

where x is a variable ranging over terms and $t, ..., t$ denotes a list of n occurrences of the symbol t. The set of free variables, $FV(u)$, of a term u is defined by induction on u. We consider each case except those of the linear λ-calculus which can be found in Section 4.3.

$$
\begin{array}{rcl}
FV(\mathtt{zero}) & = & \emptyset \\
FV(\mathtt{succ}(u)) & = & FV(u) \\
FV(\mathtt{pred}(u)) & = & FV(u) \\
FV(\mathtt{if}\ w\ \mathtt{then}\ u\ \mathtt{else}\ v) & = & FV(w) \cup FV(u) \cup FV(v) \\
FV(\Omega) & = & \emptyset \\
FV(\mathtt{Y}) & = & \emptyset
\end{array}
$$

Rules for assignment of types to terms are given in Figure 9.1. Type assignments have the form of sequents

$$x_1 : A_1, ..., x_n : A_n \mathrel{|\!\!\Rightarrow} u : A$$

where $x_1, ..., x_n$ are pairwise distinct variables. It can be shown by induction on the derivation of the type assignment that

$$FV(u) = \{x_1, ..., x_n\}$$

Note that this is different from PCF where we did not have equality, but only an inclusion. We will now show some properties of LPCF that will be of use later on. The proofs of the following results are simple extensions of analogous results for the linear λ-calculus - see Section 4.3.

Lemma 9.1.1 If the sequent $\Gamma, \mathrel{|\!\!\Rightarrow} u : A$ is derivable, then for any derivable sequent $\Gamma', \mathrel{|\!\!\Rightarrow} u : B$, where the context Γ' is a permutation of the context Γ, we have $A = B$.

Proof: See Lemma 4.3.1. □

The following result says that the term of a derivable sequent essentially encodes the derivation:

Proposition 9.1.2 If the sequent $\Gamma, \mathrel{|\!\!\Rightarrow} u : A$ is derivable, then the first rule instance above the sequent which is different from an instance of the *Exchange* rule is uniquely determined up to permutation of the context Γ.

Proof: See Proposition 4.3.2. □

Now comes a lemma dealing with substitution:

Lemma 9.1.3 (Substitution Property) If the sequents $\Gamma, \mathrel{|\!\!\Rightarrow} u : A$ and $\Delta, x : A, \Lambda \mathrel{|\!\!\Rightarrow} v : B$ are derivable and the variables in the contexts Γ and Δ, Λ are pairwise distinct, then the sequent $\Delta, \Gamma, \Lambda \mathrel{|\!\!\Rightarrow} v[u/x] : B$ is also derivable.

Proof: See Lemma 4.3.3. □

9.2 Eager Operational Semantics

It turns out that eager as well as lazy evaluation rules for terms of certain types behave properly with respect to an appropriate categorical model. For terms of other types an appropriate categorical model motivates a certain choice of evaluation rules. In this section we will give an eager operational semantics for LPCF. It is eager in the sense that we take the evaluation strategy used in each particular evaluation rule to be eager whenever there is a choice. In the next section we consider an operational semantics where a lazy evaluation strategy is taken when possible.

A *program* is an arbitrary closed term. A *value* for the eager operational semantics is a program of one of the forms

$$\mathtt{succ}^p(\mathtt{zero}) \qquad * \qquad d \otimes e \qquad \lambda x.u \qquad \mathtt{promote}\ c_1, ..., c_n\ \mathtt{for}\ x_1, ..., x_n\ \mathtt{in}\ u$$

$$\mathtt{true}(c_1, ..., c_n) \qquad (v, w) \qquad \mathtt{inl}(c) \qquad \mathtt{inr}(d)$$

where $d, e, c_1, ..., c_n$ are values and where a term $\mathtt{succ}^p(\mathtt{zero})$ is defined in the obvious way. Let T be the set of programs and C the set of values. The evaluation rules given in Figure 9.2 induce an evaluation relation $\Downarrow \subseteq T \times C$. It can be shown that the eager evaluation rules corresponding to the linear λ-calculus fragment of LPCF can be matched by β-reductions in the sense that if $v \Downarrow c$ then the program v reduces to the value c. So the operational semantics has a clear logical content via the Curry-Howard interpretation.

It turns out that eager as well as lazy evaluation rules for terms of the types $(\otimes, \multimap, 1, +)$ give rise to sound and adequate interpretations. The choice of evaluation rules for terms of LPCF outside the $(\otimes, \multimap, 1, +)$ fragment is motivated by interpretation in an appropriate categorical model. This is made explicit for some cases below.

- The operational semantics involving terms of binary product types is motivated by the following considerations: The interpretation of a program $\mathtt{fst}((u, v))$ is not strict in the interpretation of the program v, which motivates a lazy evaluation strategy where we have the rule

$$\frac{w \Downarrow (u, v) \qquad u \Downarrow c}{\mathtt{fst}(w) \Downarrow c}$$

  and any pair (u, v) of programs is a value. It actually turns out that the interpretation is not adequate if we take an eager evaluation strategy where we have the rule

$$\frac{w \Downarrow (c, d)}{\mathtt{fst}(w) \Downarrow c}$$

  and any pair (c, d) of values is a value. This can be seen by considering the non-terminating program $\mathtt{fst}((*, \Omega))$ whose interpretation is equal to id_I, and thus different from $\perp$.

- The operational semantics involving terms of ! types is motivated by the following considerations: The evaluation rules for the terms taking care of copying and discarding, that is, the terms

$$\mathtt{copy}\ w\ \mathtt{as}\ x, y\ \mathtt{in}\ u \qquad\qquad \mathtt{discard}\ w\ \mathtt{in}\ u$$

  have to give rise to a sound interpretation. Semantically, only maps of coalgebras can be copied or discarded in general, so to obtain soundness we have to restrict copying and discarding to programs of ! types whose interpretations indeed are maps of coalgebras. But the interpretation of a value of ! type will always be a map of coalgebras when it is defined to be a closed term of the form

$$\mathtt{promote}\ c_1, ..., c_n\ \mathtt{for}\ x_1, ..., x_n\ \mathtt{in}\ u$$

  where $c_1, ..., c_n$ are values. We thus obtain a sound interpretation by restricting copying and discarding to values, which amounts to the evaluation rules

$$\frac{w \Downarrow d \qquad u[d, d/x, y] \Downarrow c}{\mathtt{copy}\ w\ \mathtt{as}\ x, y\ \mathtt{in}\ u \Downarrow c} \qquad\qquad \frac{w \Downarrow d \qquad u \Downarrow c}{\mathtt{discard}\ w\ \mathtt{in}\ u \Downarrow c}$$

Figure 9.2: Eager Operational Semantics for LPCF

$$\frac{}{\texttt{zero} \Downarrow \texttt{zero}} \qquad \frac{u \Downarrow c}{\texttt{succ}(u) \Downarrow \texttt{succ}(c)} \qquad \frac{u \Downarrow \texttt{zero}}{\texttt{pred}(u) \Downarrow \texttt{zero}} \qquad \frac{u \Downarrow \texttt{succ}(c)}{\texttt{pred}(u) \Downarrow c}$$

$$\frac{u \Downarrow \texttt{zero} \quad v \Downarrow d}{\texttt{if } u \texttt{ then } v \texttt{ else } w \Downarrow d} \qquad \frac{u \Downarrow \texttt{succ}(c) \quad w \Downarrow e}{\texttt{if } u \texttt{ then } v \texttt{ else } w \Downarrow e}$$

$$\frac{}{* \Downarrow *} \qquad \frac{w \Downarrow * \quad u \Downarrow c}{\texttt{let } w \texttt{ be } * \texttt{ in } u \Downarrow c}$$

$$\frac{u \Downarrow c \quad v \Downarrow d}{u \otimes v \Downarrow c \otimes d} \qquad \frac{w \Downarrow c \otimes d \quad u[c, d/x, y] \Downarrow e}{\texttt{let } w \texttt{ be } x \otimes y \texttt{ in } u \Downarrow e}$$

$$\frac{}{\lambda x.u \Downarrow \lambda x.u} \qquad \frac{f \Downarrow \lambda x.v \quad u \Downarrow d \quad v[d/x] \Downarrow c}{fu \Downarrow c}$$

$$\frac{w_1 \Downarrow c_1, \quad \ldots \quad, w_n \Downarrow c_n}{\texttt{promote } w_1, \ldots, w_n \texttt{ for } x_1, \ldots, x_n \texttt{ in } u \Downarrow \texttt{promote } c_1, \ldots, c_n \texttt{ for } x_1, \ldots, x_n \texttt{ in } u}$$

$$\frac{u \Downarrow \texttt{promote } c_1, \ldots, c_n \texttt{ for } x_1, \ldots, x_n \texttt{ in } v \quad v[c_1, \ldots, c_n/x_1, \ldots, x_n] \Downarrow c}{\texttt{derelict}(u) \Downarrow c}$$

$$\frac{w \Downarrow d \quad u[d, d/x, y] \Downarrow c}{\texttt{copy } w \texttt{ as } x, y \texttt{ in } u \Downarrow c} \qquad \frac{w \Downarrow d \quad u \Downarrow c}{\texttt{discard } w \texttt{ in } u \Downarrow c}$$

$$\frac{}{(u, v) \Downarrow (u, v)} \qquad \frac{u \Downarrow (v, w) \quad v \Downarrow c}{\texttt{fst}(u) \Downarrow c} \qquad \frac{u \Downarrow (v, w) \quad w \Downarrow d}{\mathrm{snd}(u) \Downarrow d}$$

$$\frac{w_1 \Downarrow c_1, \quad \ldots \quad, w_n \Downarrow c_n}{\texttt{true}(w_1, \ldots, w_n) \Downarrow \texttt{true}(c_1, \ldots, c_n)}$$

$$\frac{u \Downarrow c}{\texttt{inl}(u) \Downarrow \texttt{inl}(c)} \qquad \frac{u \Downarrow d}{\texttt{inr}(u) \Downarrow \texttt{inr}(d)}$$

$$\frac{w \Downarrow \texttt{inl}(c) \quad u[c/x] \Downarrow c}{\texttt{case } w \texttt{ of } \texttt{inl}(x).u \mid \texttt{inr}(y).v \Downarrow c} \qquad \frac{w \Downarrow \texttt{inr}(d) \quad v[d/x] \Downarrow c}{\texttt{case } w \texttt{ of } \texttt{inl}(x).u \mid \texttt{inr}(y).v \Downarrow c}$$

$$\frac{}{\texttt{Y} \Downarrow \lambda f.\texttt{copy } f \texttt{ as } g, h \texttt{ in } (\texttt{derelict}(g)\texttt{promote } h \texttt{ for } k \texttt{ in } (\texttt{Y}k))}$$

In [Abr90] another choice of evaluation rules for the terms taking care of copying and discarding is made, namely

$$\frac{u[w, w/x, y] \Downarrow c}{\texttt{copy}\ w\ \texttt{as}\ x, y\ \texttt{in}\ u \Downarrow c} \qquad \frac{u \Downarrow c}{\texttt{discard}\ w\ \texttt{in}\ u \Downarrow c}$$

But in the presence of non-terminating programs the second of these evaluation rules is not sound. For example, we have $\texttt{discard}\,\Omega\ \texttt{in}\ * \Downarrow *$, but the interpretation of the program $\texttt{discard}\,\Omega\ \texttt{in}\ *$ is strict in the interpretation of Ω and therefore equal to $\perp$, whereas the interpretation of $*$ is equal to id_I.

- Also, the way of introducing recursion in LPCF is motivated by interpretation in an appropriate categorical model: The interpretation of (the evaluation rule for) the linear fixpoint constant of LPCF corresponds to the interpretation of (the evaluation rule for) the fixpoint constant of PCF in the induced Kleisli category. This is consistent with syntactic notions in the sense that the linear fixpoint constant of LPCF is the image under the Girard translation of the fixpoint constant of PCF, and, moreover, the evaluation rule for the linear fixpoint constant is essentially the image under the Girard translation of the evaluation rule for the fixpoint constant of PCF.

The operational semantics enjoys the following properties:

Proposition 9.2.1 (Eager Subject Reduction) If u is a program of type A and $u \Downarrow c$ then c is also of type A.

Proof: Induction on the derivation of $u \Downarrow c$ where we use Lemma 9.1.3. □

This shows that the evaluation relation $\Downarrow \subseteq T \times C$ can be split up into a family of relations such that a relation $\Downarrow_A \subseteq T_A \times C_A$ is given for each type A.

Proposition 9.2.2 (Eager Determinacy) If $u \Downarrow c$ and $u \Downarrow d$ then $c = d$.

Proof: Induction on the derivation of $u \Downarrow c$. □

9.3 Lazy Operational Semantics

We will now give a lazy operational semantics for LPCF. It is lazy in the sense that the evaluation strategy used in each particular evaluation rule is taken to be lazy whenever there is a choice.

Recall that a program is an arbitrary closed term. A *value* for the lazy operational semantics is a program of one of the forms

$$\texttt{succ}^p(\texttt{zero}) \qquad * \qquad u \otimes v \qquad \lambda x.u \qquad \texttt{promote}\ c_1, ..., c_n\ \texttt{for}\ x_1, ..., x_n\ \texttt{in}\ u$$

$$\texttt{true}(w_1, ..., w_n) \qquad (v, w) \qquad \texttt{inl}(u) \qquad \texttt{inr}(v)$$

where $c_1, ..., c_n$ are values. The evaluation rules given in Figure 9.3 induce an evaluation relation $\downarrow \subseteq T \times C$. The lazy evaluation rules for terms within the $(\otimes, \multimap, 1, +)$ fragment are different from the corresponding eager rules. It can be shown that the evaluation rules corresponding to the linear λ-calculus fragment of LPCF can be matched by β-reductions in the sense that if $v \downarrow c$ then the program v reduces to the value c. So the operational semantics has a clear logical content via the Curry-Howard interpretation. The operational semantics enjoys the following properties:

Proposition 9.3.1 (Lazy Subject Reduction) If u is a program of type A and $u \downarrow c$ then c is also of type A.

Figure 9.3: Lazy Operational Semantics for LPCF

$$\frac{}{\texttt{zero} \downarrow \texttt{zero}} \qquad \frac{u \downarrow c}{\texttt{succ}(u) \downarrow \texttt{succ}(c)} \qquad \frac{u \downarrow \texttt{zero}}{\texttt{pred}(u) \downarrow \texttt{zero}} \qquad \frac{u \downarrow \texttt{succ}(c)}{\texttt{pred}(u) \downarrow c}$$

$$\frac{u \downarrow \texttt{zero} \quad v \downarrow d}{\texttt{if } u \texttt{ then } v \texttt{ else } w \downarrow d} \qquad \frac{u \downarrow \texttt{succ}(c) \quad w \downarrow e}{\texttt{if } u \texttt{ then } v \texttt{ else } w \downarrow e}$$

$$\frac{}{* \downarrow *} \qquad \frac{w \downarrow * \quad u \downarrow c}{\texttt{let } w \texttt{ be } * \texttt{ in } u \downarrow c}$$

$$\frac{}{u \otimes v \downarrow u \otimes v} \qquad \frac{w \downarrow u \otimes v \quad t[u, v/x, y] \downarrow e}{\texttt{let } w \texttt{ be } x \otimes y \texttt{ in } t \downarrow e}$$

$$\frac{}{\lambda x.u \downarrow \lambda x.u} \qquad \frac{f \downarrow \lambda x.v \quad v[u/x] \downarrow c}{fu \downarrow c}$$

$$\frac{w_1 \downarrow c_1, \quad \ldots \quad , w_n \downarrow c_n}{\texttt{promote } w_1, \ldots, w_n \texttt{ for } x_1, \ldots, x_n \texttt{ in } u \downarrow \texttt{promote } c_1, \ldots, c_n \texttt{ for } x_1, \ldots, x_n \texttt{ in } u}$$

$$\frac{u \downarrow \texttt{promote } c_1, \ldots, c_n \texttt{ for } x_1, \ldots, x_n \texttt{ in } v \quad v[c_1, \ldots, c_n/x_1, \ldots, x_n] \downarrow c}{\texttt{derelict}(u) \downarrow c}$$

$$\frac{w \downarrow d \quad u[d, d/x, y] \downarrow c}{\texttt{copy } w \texttt{ as } x, y \texttt{ in } u \downarrow c} \qquad \frac{w \downarrow d \quad u \downarrow c}{\texttt{discard } w \texttt{ in } u \downarrow c}$$

$$\frac{}{(u, v) \downarrow (u, v)} \qquad \frac{u \downarrow (v, w) \quad v \downarrow c}{\texttt{fst}(u) \downarrow c} \qquad \frac{u \downarrow (v, w) \quad w \downarrow d}{\mathrm{snd}(u) \downarrow d}$$

$$\frac{}{\texttt{true}(w_1, \ldots, w_n) \downarrow \texttt{true}(w_1, \ldots, w_n)}$$

$$\frac{}{\texttt{inl}(u) \downarrow \texttt{inl}(u)} \qquad \frac{}{\texttt{inr}(u) \downarrow \texttt{inr}(u)}$$

$$\frac{w \downarrow \texttt{inl}(u) \quad t[u/x] \downarrow c}{\texttt{case } w \texttt{ of } \texttt{inl}(x).t \mid \texttt{inr}(y).v \downarrow c} \qquad \frac{w \downarrow \texttt{inr}(u) \quad v[u/x] \downarrow c}{\texttt{case } w \texttt{ of } \texttt{inl}(x).t \mid \texttt{inr}(y).v \downarrow c}$$

$$\frac{}{\texttt{Y} \downarrow \lambda f.\texttt{copy } f \texttt{ as } g, h \texttt{ in } (\texttt{derelict}(g)\texttt{promote } h \texttt{ for } k \texttt{ in } (\texttt{Y}k))}$$

Proof: Analogous to the proof of Proposition 9.2.1. □

So the evaluation relation $\downarrow \subseteq T \times C$ can be split up into a family of relations such that a relation $\downarrow_A \subseteq T_A \times C_A$ is given for each type A.

Proposition 9.3.2 (Lazy Determinacy) If $u \downarrow c$ and $u \downarrow d$ then $c = d$.

Proof: Analogous to the proof of Proposition 9.2.2. □

9.4 Categorical Semantics

In this section we will give a sound categorical interpretation of LPCF. In Section 9.7 and Section 9.8 we shall impose additional assumptions on our category with the aim of proving adequacy.

Definition 9.4.1 A *categorical premodel for LPCF* is a pointed linear category equipped with

- finite products and sums
- a linear object of numerals,
- a linear fixpoint operator for each object.

Examples 9.4.2 In Chapter 2 and Chapter 8 it is made clear that the concrete linear categories $\mathbf{cpo}_{st}$ and $\mathbf{dI}_{lin}$ are categorical premodels for LPCF.

Given a premodel for LPCF, we can interpret types as objects, typing rules as natural operations on maps, and derivations of type assignments as maps. A derivable sequent

$$x_1 : A_1, ..., x_n : A_n \mathbb{I}\!\!\Rightarrow u : B$$

is interpreted as a map

$$[\![A_1]\!] \otimes ... \otimes [\![A_n]\!] \xrightarrow{[\![u]\!]} [\![B]\!]$$

by induction on its derivation using the appropriate operations on maps induced by the categorical assumptions cf. below. We consider each case except those of the linear λ-calculus which can be found in Section 4.5.

- The derivation

$$\frac{}{\mathbb{I}\!\!\Rightarrow \mathtt{zero} : N}$$

  is interpreted as

$$\frac{}{I \xrightarrow{zero} N}$$

- The derivation

$$\frac{, \mathbb{I}\!\!\Rightarrow u : N}{, \mathbb{I}\!\!\Rightarrow \mathtt{succ}(u) : N}$$

  is interpreted as

$$\frac{, \xrightarrow{u} N}{, \xrightarrow{u} N \xrightarrow{succ} N}$$

- The derivation

$$\frac{, \mathbb{I}\!\!\Rightarrow u : N}{, \mathbb{I}\!\!\Rightarrow \mathtt{pred}(u) : N}$$

  is interpreted as

$$\frac{, \xrightarrow{u} N}{, \xrightarrow{u} N \xrightarrow{pred} N}$$

- The derivation

$$\frac{\Lambda \Vdash u : N \quad , \Vdash v : A \quad , \Vdash w : A}{\Lambda, , \Vdash \texttt{if } u \texttt{ then } v \texttt{ else } w : A}$$

  is interpreted as

$$\frac{\Lambda \xrightarrow{u} N \quad , \xrightarrow{v} A \quad , \xrightarrow{w} A}{\Lambda \otimes , \xrightarrow{u \otimes (v,w)} N \otimes (A \times A) \xrightarrow{cond} A}$$

- The derivation

$$\frac{}{\Vdash \Omega_A : A}$$

  is interpreted as

$$\frac{}{I \xrightarrow{\perp} A}$$

- The derivation

$$\frac{}{\Vdash \mathtt{Y}_A : !(!A \multimap A) \multimap A}$$

  is interpreted as

$$\frac{}{I \xrightarrow{K^\sharp} !(!A \multimap A) \multimap A}$$

  using the internal linear fixpoint operator induced by the appropriate linear fixpoint operator according to Proposition 8.3.12.

The following lemma corresponds to Lemma 9.1.3 where the categorical semantics is taken into account; it says essentially that substitution corresponds to composition:

Lemma 9.4.3 (Substitution) If the sequents , $\Vdash u : A$ and $\Delta, x : A, \Lambda \Vdash v : B$ are derivable and the variables in the contexts , and Δ, Λ are pairwise distinct, then the sequent $\Delta, , , \Lambda \Vdash v[u/x] : B$ is also derivable and it has the interpretation

$$\Delta \otimes , \otimes \Lambda \xrightarrow{\Delta \otimes [\![u]\!] \otimes \Lambda} \Delta \otimes A \otimes \Lambda \xrightarrow{[\![v]\!]} B$$

Proof: Extend the proof of Lemma 4.5.3 for the linear λ-calculus as appropriate. □

The interpretation is preserved by eager evaluation:

Theorem 9.4.4 (Eager Soundness) Given a program u such that $u \Downarrow c$, then $[\![u]\!] = [\![c]\!]$.

Proof: Induction on the derivation of $u \Downarrow c$ where we use Lemma 9.4.3. Note that the evaluation rule for the fixpoint constant preserves the interpretation because it is essentially a syntactic restatement of the defining diagram for an internal linear fixpoint operator, Definition 8.3.9. □

And analogously, the interpretation is preserved by lazy evaluation:

Theorem 9.4.5 (Lazy Soundness) Given a program u such that $u \downarrow c$, then $[\![u]\!] = [\![c]\!]$.

Proof: Analogous to the proof of Theorem 9.4.4. □

Figure 9.4: Type Assignment Rules for Generalised LPCF

$$\frac{}{\Sigma; x : A \Mapsto x : A} \qquad \frac{}{x_1 : A_1, ..., x_n : A_n; \Mapsto x_q : A_q}$$

$$\frac{\Sigma; , , x : A, y : B, \Delta \Mapsto u : C}{\Sigma; , , y : B, x : A, \Delta \Mapsto u : C}$$

$$\frac{}{\Sigma; \Mapsto \mathtt{zero} : N} \qquad \frac{\Sigma; , \Mapsto u : N}{\Sigma; , \Mapsto \mathtt{succ}(u) : N} \qquad \frac{\Sigma; , \Mapsto u : N}{\Sigma; , \Mapsto \mathtt{pred}(u) : N}$$

$$\frac{\Sigma; \Lambda \Mapsto u : N \qquad \Sigma; , \Mapsto v : A \qquad \Sigma; , \Mapsto w : A}{\Sigma; \Lambda, , \Mapsto \mathtt{if}\ u\ \mathtt{then}\ v\ \mathtt{else}\ w : A}$$

$$\frac{}{\Sigma; \Mapsto * : I} \qquad \frac{\Sigma; \Lambda \Mapsto w : I \qquad \Sigma; , \Mapsto u : A}{\Sigma; , , \Lambda \Mapsto \mathtt{let}\ w\ \mathtt{be}\ *\ \mathtt{in}\ u : A}$$

$$\frac{\Sigma; , \Mapsto u : A \qquad \Sigma; \Delta \Mapsto v : B}{\Sigma; , , \Delta \Mapsto u \otimes v : A \otimes B} \qquad \frac{\Sigma; \Lambda \Mapsto w : A \otimes B \qquad \Sigma; , , x : A, y : B \Mapsto u : C}{\Sigma; , , \Lambda \Mapsto \mathtt{let}\ w\ \mathtt{be}\ x \otimes y\ \mathtt{in}\ u : C}$$

$$\frac{\Sigma; , , x : A \Mapsto u : B}{\Sigma; , \Mapsto \lambda x^A.u : A \multimap B} \qquad \frac{\Sigma; \Lambda \Mapsto f : A \multimap B \qquad \Sigma; \Delta \Mapsto u : A}{\Sigma; \Lambda, \Delta \Mapsto fu : B}$$

$$\frac{\Sigma; , _1 \Mapsto v_1 : !A_1, \quad ... \quad , \Sigma; , _n \Mapsto v_n : !A_n \qquad \Sigma; x_1 : !A_1, ..., x_n : !A_n \Mapsto u : B}{\Sigma; , _1, ..., , _n \Mapsto \mathtt{promote}\ v_1, ..., v_n\ \mathtt{for}\ x_1, ..., x_n\ \mathtt{in}\ u : !B} \qquad \frac{\Sigma; \Lambda \Mapsto u : !B}{\Sigma; \Lambda \Mapsto \mathtt{derelict}(u) : B}$$

$$\frac{\Sigma; \Lambda \Mapsto w : !A \qquad \Sigma; , , x : !A, y : !A \Mapsto u : B}{\Sigma; , , \Lambda \Mapsto \mathtt{copy}\ w\ \mathtt{as}\ x, y\ \mathtt{in}\ u : B} \qquad \frac{\Sigma; \Lambda \Mapsto w : !A \qquad \Sigma; , \Mapsto u : B}{\Sigma; , , \Lambda \Mapsto \mathtt{discard}\ w\ \mathtt{in}\ u : B}$$

$$\frac{\Sigma; , \Mapsto u : A \qquad \Sigma; , \Mapsto v : B}{\Sigma; , \Mapsto (u, v) : A \times B} \qquad \frac{\Sigma; \Lambda \Mapsto u : A \times B}{\Sigma; \Lambda \Mapsto \mathtt{fst}(u) : A} \qquad \frac{\Sigma; \Lambda \Mapsto u : A \times B}{\Sigma; \Lambda \Mapsto \mathtt{snd}(u) : B}$$

$$\frac{\Sigma; , _1 \Mapsto v_1 : A_1, \quad ... \quad , \Sigma; , _n \Mapsto v_n : A_n}{\Sigma; , _1, ..., , _n \Mapsto \mathtt{true}(v_1, ..., v_n) : 1} \qquad \frac{\Sigma; , _1 \Mapsto v_1 : A_1, \quad ... \quad , \Sigma; , _n \Mapsto v_n : A_n \qquad \Sigma; \Lambda \Mapsto u : 0}{\Sigma; , _1, ..., , _n, \Lambda \Mapsto \mathtt{false}^C(v_1, ..., v_n; u) : C}$$

$$\frac{\Sigma; , \Mapsto u : A}{\Sigma; , \Mapsto \mathtt{inl}^{A+B}(u) : A + B} \qquad \frac{\Sigma; , \Mapsto u : B}{\Sigma; , \Mapsto \mathtt{inr}^{A+B}(u) : A + B}$$

$$\frac{\Sigma; \Lambda \Mapsto w : A + B \qquad \Sigma; , , x : A \Mapsto u : C \qquad \Sigma; , , y : B \Mapsto v : C}{\Sigma; , , \Lambda \Mapsto \mathtt{case}\ w\ \mathtt{of}\ \mathtt{inl}(x).u \mid \mathtt{inr}(y).v : C}$$

$$\frac{}{\Sigma; \Mapsto \Omega_A : A} \qquad \frac{}{\Sigma; \Mapsto \mathtt{Y}_A : !(!A \multimap A) \multimap A}$$

9.5 Generalised LPCF - Syntax

In this section we will introduce what we call Generalised LPCF. The goal is twofold: It enables us to prove the adequacy results of Section 9.7 and Section 9.8 and it enables us to state and prove the unwinding theorem of Section 9.9. Generalised LPCF is an extension of the generalised linear λ-calculus with numerals and recursion, appropriate for the linear context.

Types and terms of Generalised LPCF are the same as for LPCF, but the rules for type assignment are more general; they have two contexts instead of one. Rules for assignment of types to terms are given in Figure 9.4. They consist of the rules of LPCF extended with an extra context dealt with in an additive fashion, and furthermore, there is an extra axiom. Type assignments thus have the form of sequents

$$x_1 : A_1, ..., x_n : A_n; y_1 : B_1, ..., y_m : B_m \Leftmapsto u : C$$

where $x_1, ..., x_n, y_1, ..., y_m$ are pairwise distinct variables. It can be shown by induction on the derivation of the type assignment that

$$FV(u) \Leftrightarrow \{x_1, ..., x_n\} = \{y_1, ..., y_m\}$$

and

$$FV(u) \Leftrightarrow \{y_1, ..., y_m\} \subseteq \{x_1, ..., x_n\}$$

We use the same convention concerning the two different kinds of variables as for the generalised linear λ-calculus: The variables occuring on the left hand side of the semicolon are called *intuitionistic* variables and the variables occuring on the right hand side of the semicolon are called *linear* variables. Correspondingly, the context on the left hand side of the semicolon is called the *intuitionistic* context and the context on the right hand side of the semicolon is called the *linear* context. Note that an intuitionistic variable cannot be bound. It is straightforward to check that Generalised LPCF is a conservative extension of LPCF in the sense that a sequent $, \Leftmapsto u : A$ is derivable iff the sequent $; , \Leftmapsto u : A$ is derivable.

The role of the intuitionistic context is best explained by looking at an example: The term

$$\texttt{promote for in}\ \mathtt{Y}_A$$

(where the lists of variables and terms are empty) is typeable in LPCF as follows:

$$\frac{\overline{\Leftmapsto \mathtt{Y}_A : !(!A \multimap A) \multimap A}}{\Leftmapsto \texttt{promote for in}\ \mathtt{Y}_A : !(!(!A \multimap A) \multimap A)}$$

It is also typeable in Generalised LPCF as follows:

$$\frac{\overline{; \Leftmapsto \mathtt{Y}_A : !(!A \multimap A) \multimap A}}{; \Leftmapsto \texttt{promote for in}\ \mathtt{Y}_A : !(!(!A \multimap A) \multimap A)}$$

We want to express this term as the result of substituting the linear fixpoint constant $\mathtt{Y}$ for a free variable in a suitable term where the linear fixpoint constant does not occur. The only choice of term

$$\texttt{promote for in}\ x$$

is, however, not typeable in LPCF, but it is typeable in Generalised LPCF as follows:

$$\frac{\overline{x : !(!A \multimap A) \multimap A; \Leftmapsto x : !(!A \multimap A) \multimap A}}{x : !(!A \multimap A) \multimap A; \Leftmapsto \texttt{promote for in}\ x : !(!(!A \multimap A) \multimap A)}$$

Note that the variable x is an intuitionistic variable. Such situations are taken care of by Lemma 9.5.8. We will now show some properties of Generalised LPCF that will be of use later on. The following results are simple extensions of analogous results for the generalised linear λ-calculus - see Section 4.6.

Lemma 9.5.1 If the sequent $\Sigma;, \mathrel{\shortmid\!\!\Rightarrow} u : A$ is derivable, then for any sequent $\Sigma;, '\mathrel{\shortmid\!\!\Rightarrow} u : B$ which also is derivable where the context $, '$ is a permutation of the context $, ,$ we have $A = B$.

Proof: See Lemma 4.6.1. □

The following result says that the term of a derivable sequent essentially encodes the derivation:

Proposition 9.5.2 If the sequent $\Sigma;, \mathrel{\shortmid\!\!\Rightarrow} u : A$ is derivable, then the first rule instance above the sequent which is different from an instance of the *Exchange* rule is uniquely determined up to permutation of the context $, ,$.

Proof: See Proposition 4.6.2. □

We need a small lemma dealing with expansion of intuitionistic contexts.

Lemma 9.5.3 If the sequent $\Sigma, \Theta; \Lambda \mathrel{\shortmid\!\!\Rightarrow} u : A$ is derivable and the variables in the contexts $\Sigma, \Theta; \Lambda$ and Φ are pairwise distinct, then the sequent $\Sigma, \Phi, \Theta; \Lambda \mathrel{\shortmid\!\!\Rightarrow} u : A$ is also derivable.

Proof: See Lemma 4.6.3. □

The Substitution Property now splits up into two cases; one for each kind of variables. The first case deals with linear variables:

Lemma 9.5.4 (Linear Substitution Property) If the sequent $\Sigma;, \mathrel{\shortmid\!\!\Rightarrow} u : A$ as well as the sequent $\Sigma; \Pi, x : A, \Lambda \mathrel{\shortmid\!\!\Rightarrow} v : B$ both are derivable and the variables in the contexts $,$ and Π, Λ are pairwise distinct, then the sequent $\Sigma; \Pi, , , \Lambda \mathrel{\shortmid\!\!\Rightarrow} v[u/x] : B$ is also derivable.

Proof: See Lemma 4.6.4. □

The preceding result might be more comprehensible if we restrict to substitution of programs which is what we will need later on.

Corollary 9.5.5 If the sequents $; \mathrel{\shortmid\!\!\Rightarrow} u : A$ and $; \Pi, x : A, \Lambda \mathrel{\shortmid\!\!\Rightarrow} v : B$ are derivable, then the sequent $; \Pi, \Lambda \mathrel{\shortmid\!\!\Rightarrow} v[u/x] : B$ is also derivable.

Proof: Restrict Lemma 9.5.4. □

The second case deals with intuitionistic variables:

Lemma 9.5.6 (Intuitionistic Substitution Property) If the sequent $\Sigma; \mathrel{\shortmid\!\!\Rightarrow} u : A$ as well as the sequent $\Sigma, x : A, \Theta; \Lambda \mathrel{\shortmid\!\!\Rightarrow} v : B$ both are derivable, then the sequent $\Sigma, \Theta; \Lambda \mathrel{\shortmid\!\!\Rightarrow} v[u/x] : B$ is also derivable.

Proof: See Lemma 4.6.5. □

This result might be more comprehensible if we restrict to substitution of programs which is what we will need later on.

Corollary 9.5.7 If the sequents $; \mathrel{\shortmid\!\!\Rightarrow} u : A$ and $x : A, \Theta; \Lambda \mathrel{\shortmid\!\!\Rightarrow} v : B$ are derivable, then the sequent $\Theta; \Lambda \mathrel{\shortmid\!\!\Rightarrow} v[u/x] : B$ is also derivable.

Proof: Restrict Lemma 9.5.6. □

The following lemma is dealing with the "inverse" to substitution in LPCF analogous to Lemma 7.1.5 of PCF. Recall that $v[x/u]$ denotes the term v where inductively all occurrences of the term u have been replaced by the variable x.

Lemma 9.5.8 If the sequents $\Sigma; \mathrel{\shortmid\!\!\Rightarrow} u : A$ and $\Sigma, \Theta; \Lambda \mathrel{\shortmid\!\!\Rightarrow} v : B$ are derivable, then the sequent $\Sigma, x : A, \Theta; \Lambda \mathrel{\shortmid\!\!\Rightarrow} v[x/u] : B$ is also derivable where x is a new variable.

Proof: See Lemma 4.6.6. □

9.6 Generalised LPCF - Semantics

Given a categorical premodel for LPCF we are able to interpret Generalised LPCF; types are interpreted as objects, typing rules as natural operations on maps, and derivations of type assignments as maps. A derivable sequent

$$x_1 : A_1, ..., x_n : A_n; y_1 : B_1, ..., y_m : B_m \mathbb{I}\!\!\Rightarrow u : C$$

is interpreted as a map

$$!\llbracket A_1 \rrbracket \otimes ... \otimes !\llbracket A_n \rrbracket \otimes \llbracket B_1 \rrbracket \otimes ... \otimes \llbracket B_m \rrbracket \xrightarrow{\llbracket u \rrbracket} \llbracket C \rrbracket$$

by induction on its derivation using appropriate operations on maps induced by the categorical model. The cases corresponding to the generalised linear λ-calculus are considered in Section 4.7; the operations on maps corresponding to the remaining typing rules are straightforward extensions of the operations on maps induced by the typing rules for LPCF. The categorical interpretation of Generalised LPCF is a conservative extension of the categorical interpretation of LPCF in the sense that the interpretation of a LPCF sequent $, \mathbb{I}\!\!\Rightarrow u : A$ coincides with the interpretation of the Generalised LPCF sequent $;, \mathbb{I}\!\!\Rightarrow u : A$.

The following lemma corresponds to Lemma 9.5.3 where the categorical semantics is taken into account:

Lemma 9.6.1 If the sequent $\Sigma, \Theta; \Lambda \mathbb{I}\!\!\Rightarrow u : A$ is derivable and the variables in the contexts $\Sigma, \Theta; \Lambda$ and Φ are pairwise distinct, then the sequent $\Sigma, \Phi, \Theta; \Lambda \mathbb{I}\!\!\Rightarrow u : A$ is also derivable and it has the interpretation

$$\Sigma \otimes \Phi \otimes \Theta \otimes \Lambda \xrightarrow{\Sigma \otimes \langle\rangle \otimes \Theta \otimes \Lambda} \Sigma \otimes I \otimes \Theta \otimes \Lambda \cong \Sigma \otimes \Theta \otimes \Lambda \xrightarrow{\llbracket u \rrbracket} A$$

Proof: Extend the proof of Lemma 4.7.1 for the generalised linear λ-calculus as appropriate. □

The following lemma corresponds to Lemma 9.5.4 where the categorical semantics is taken into account; it essentially says that substitution with respect to linear variables corresponds to composition:

Lemma 9.6.2 (Linear Substitution) If the sequents $\Sigma; , \mathbb{I}\!\!\Rightarrow u : A$ and $\Sigma; \Pi, x : A, \Lambda \mathbb{I}\!\!\Rightarrow v : B$ are derivable and the variables in the contexts $,$ and Π, Λ are pairwise distinct, then the sequent $\Sigma; \Pi, , , \Lambda \mathbb{I}\!\!\Rightarrow v[u/x] : B$ is also derivable and it has the interpretation

$$\Sigma \otimes \Pi \otimes , \otimes \Lambda \xrightarrow{\Delta \otimes \Pi \otimes \Gamma \otimes \Lambda} \Sigma \otimes \Sigma \otimes \Pi \otimes , \otimes \Lambda \cong \Sigma \otimes \Pi \otimes \Sigma \otimes , \otimes \Lambda \xrightarrow{\Sigma \otimes \Pi \otimes \llbracket u \rrbracket \otimes \Lambda} \Sigma \otimes \Pi \otimes A \otimes \Lambda \xrightarrow{\llbracket v \rrbracket} B$$

Proof: Extend the proof of Lemma 4.7.2 for the generalised linear λ-calculus as appropriate. □

The preceding result might be more comprehensible if we restrict to the special cases dealing with substitution of programs. In fact this is what we will need later on.

Corollary 9.6.3 If the sequents $; \mathbb{I}\!\!\Rightarrow u : A$ and $; \Pi, x : A, \Lambda \mathbb{I}\!\!\Rightarrow v : B$ are derivable, then the sequent $; \Pi, \Lambda \mathbb{I}\!\!\Rightarrow v[u/x] : B$ is also derivable and it has the interpretation

$$\Pi \otimes \Lambda \cong \Pi \otimes I \otimes \Lambda \xrightarrow{\Pi \otimes \llbracket u \rrbracket \otimes \Lambda} \Pi \otimes A \otimes \Lambda \xrightarrow{\llbracket v \rrbracket} B$$

Proof: Restrict Lemma 9.6.2. □

The following lemma corresponds to Lemma 9.5.6 where the categorical semantics is taken into account; it essentially says that substitution with respect to intuitionistic variables corresponds to composition in the category of coalgebras:

Lemma 9.6.4 (Intuitionistic Substitution) If the sequents $\Sigma; \mathbb{I}\!\!\Leftrightarrow u : A$ and $\Sigma, x : A, \Theta; \Lambda \mathbb{I}\!\!\Leftrightarrow v : B$ are derivable, then the sequent $\Sigma, \Theta; \Lambda \mathbb{I}\!\!\Leftrightarrow v[u/x] : B$ is also derivable and it has the interpretation

$$\Sigma \otimes \Theta \otimes \Lambda \xrightarrow{\Delta \otimes \Theta \otimes \Lambda} \Sigma \otimes \Sigma \otimes \Theta \otimes \Lambda \xrightarrow{\Sigma \otimes \gamma([\![u]\!]) \otimes \Theta \otimes \Lambda} \Sigma \otimes !A \otimes \Theta \otimes \Lambda \xrightarrow{[\![v]\!]} B$$

Proof: Extend the proof of Lemma 4.7.3 for the generalised linear λ-calculus as appropriate. □

This result might be more comprehensible if we restrict to the special cases dealing with substitution of programs. In fact this is what we will need later on.

Corollary 9.6.5 If the sequents $; \mathbb{I}\!\!\Leftrightarrow u : A$ and $x : A, \Theta; \Lambda \mathbb{I}\!\!\Leftrightarrow v : B$ are derivable, then the sequent $\Theta; \Lambda \mathbb{I}\!\!\Leftrightarrow v[u/x] : B$ is also derivable and it has the interpretation

$$\Theta \otimes \Lambda \cong I \otimes \Theta \otimes \Lambda \xrightarrow{\gamma([\![u]\!]) \otimes \Theta \otimes \Lambda} !A \otimes \Theta \otimes \Lambda \xrightarrow{[\![v]\!]} B$$

Proof: Restrict Lemma 9.6.4. □

9.7 Eager Adequacy

In this section we will show that the categorical interpretation is adequate when LPCF is equipped with the eager operational semantics. Our adequacy proof proceeds using logical relations in a way analogous to the use of logical relations in the adequacy proof for PCF. We use the binary logical relations $\preceq^{\circ}_A$ and $\preceq_A$ given below.

While it is possible to prove adequacy for PCF without sums using an inclusiveness result adapted to our order-free categorical setting, it turns out that this approach does not carry over to LPCF. In the setting here an appropriate inclusiveness result would be as follows: Let a program u of type A and maps $f : !D \to D$ and $g : !D \to A$ be given; it is then the case that if $\gamma(\gamma(\bot); f^n); g \preceq_A u$ holds for every number n, then $\gamma(f^\sharp); g \preceq_A u$ holds too. We will illustrate the problems involved in carrying out an induction proof of this assertion by considering a couple of special cases, namely the two "pairing" constructs (u, v) and $u \otimes v$. The map $\gamma(f^\sharp); g$ will be denoted l and for every number n the map $\gamma(\gamma(\bot); f^n); g$ will be denoted l_n. The (u, v) case works well: If $l_n \preceq_{B \times C} (u, v)$ for every number n, then $l_n; \pi_1 \preceq_B u$ and $l_n; \pi_2 \preceq_C v$ for every number n, which gives by induction $l; \pi_1 \preceq_B u$ and $l; \pi_2 \preceq_C v$, and thus $l \preceq_{B \times C} (u, v)$. But this cannot be done in the $u \otimes v$ case because there are no projections around for the tensor product $B \otimes C$, and moreover, there does not seem to be any other reasonable way to prove that case. It is actually a characteristic feature of the pair $u \otimes v$ that both components have to be used, whereas exactly one of the components of the pair (u, v) has to be used, corresponding to the presence of projections. So we prove adequacy in a way dodging inclusiveness since the obstacle in obtaining such a result corresponds to a feature characteristic to LPCF. It should be mentioned that an adequacy result using a computability predicate, Section 7.3, involves an analogous obstacle.

We need an extra assumption on our premodel for LPCF.

Definition 9.7.1 A *categorical model for LPCF* is a categorical premodel in the sense of Definition 9.4.1 such that every map is strict.

Examples 9.7.2 In Chapter 2 and Chapter 8 it is made clear that the concrete linear categories $\mathbf{cpo}_{st}$ and $\mathbf{dI}_{lin}$ are categorical models for LPCF.

Note that the adequacy result below is stated solely in terms of the categorical premodel, Definition 9.4.1, but the assumption added in Definition 9.7.1 is indeed used in the proof. Recall that T_A is the set of programs of type A and C_A the set of values of type A.

Definition 9.7.3 For each type A the binary logical relations

$$\preceq^{\circ}_A \subseteq (\mathcal{C}(I,A) \Leftrightarrow \{\bot\}) \times C_A \qquad\qquad \preceq_A \subseteq \mathcal{C}(I,A) \times T_A$$

are defined by induction on the structure of A. The relation $\preceq^{\circ}_A$ is defined as

$$\begin{array}{rll}
f \preceq^{\circ}_N \mathtt{succ}^n(\mathtt{zero}) & \text{iff} & f = \tilde{n} \\
f \preceq^{\circ}_I * & \text{iff} & f = id \\
f \preceq^{\circ}_{B\otimes C} d \otimes e & \text{iff} & \exists g \in \mathcal{C}(I,B).\exists h \in \mathcal{C}(I,C). \\
 & & f = \cong; (g \otimes h) \ \wedge \ g \preceq^{\circ}_B d \ \wedge \ h \preceq^{\circ}_C e \\
f \preceq^{\circ}_{B\multimap C} \lambda x.u & \text{iff} & \forall g \in \mathcal{C}(I,B).\forall c \in C_B. \\
 & & g \preceq^{\circ}_B c \ \Rightarrow \cong; (f \otimes g); eval \preceq_C u[c/x] \\
f \preceq^{\circ}_{!B} \mathtt{promote}\ \overline{c}\ \mathtt{for}\ \overline{x}\ \mathtt{in}\ u & \text{iff} & f;\delta = m_I; !f \ \wedge \ f;\varepsilon \preceq_B u[\overline{c}/\overline{x}] \\
f \preceq^{\circ}_{B\times C} (v,w) & \text{iff} & f;\pi_1 \preceq_B v \ \wedge \ f;\pi_2 \preceq_C w \\
f \preceq^{\circ}_{B+C} \mathtt{inl}(c) & \text{iff} & \exists h \in \mathcal{C}(1,B).\ f = h; in_1 \ \wedge \ h \preceq^{\circ}_B c \\
f \preceq^{\circ}_{B+C} \mathtt{inr}(d) & \text{iff} & \exists h \in \mathcal{C}(1,C).\ f = h; in_2 \ \wedge \ h \preceq^{\circ}_C d
\end{array}$$

and the relation $\preceq_A$ is defined as

$$f \preceq_A u \quad \text{iff} \quad f \neq \bot \Rightarrow \ \exists c \in C_A.\ u \Downarrow c \ \wedge \ f \preceq^{\circ}_A c.$$

Note that $f \preceq^{\circ}_A c$ entails $f \preceq_A c$ because $c \Downarrow c$ for any value c. The equation $f;\delta = m_I; !f$ in the ! case simply says that f is a map of coalgebras. In the definition there is no case for the $\preceq^{\circ}_1$ relation because every map $f : I \to 1$ is equal to $\bot$, and similarly, there is no case for the $\preceq^{\circ}_0$ relation because there are no values of type 0.

Note that in the following lemma the term u is assumed not to contain any occurrences of the linear fixpoint constant. Also note that such a restriction is not imposed on the r_i and s_j terms.

Lemma 9.7.4 Assume that Σ denotes a context $x_1 : A_1, ..., x_n : A_n$ and , denotes a context $y_1 : B_1, ..., y_m : B_m$ and consider a derivable sequent

$$\Sigma; , \ \mathbb{I}\!\Leftrightarrow u : C$$

of Generalised LPCF such that the term u does not contain any occurrences of the linear fixpoint constant. Assume that for each $i \in \{1, ..., n\}$ we have a map $f_i : I \to A_i$ and a program r_i of type A_i such that $f_i \preceq_{A_i} r_i$, and similarly, assume that for each $j \in \{1, ..., m\}$ we have a map $l_j : I \to B_j$ and a program s_j of type B_j such that $l_j \preceq_{B_j} s_j$. We then have

$$\rho_{\Sigma;\Gamma}; [\![u]\!] \preceq_C u[\overline{r}, \overline{s}/\overline{x}, \overline{y}]$$

where the map $\rho_{\Sigma;\Gamma}$ is defined as

$$I \cong I \otimes I \xrightarrow{\rho_\Sigma \otimes \rho_\Gamma} !A_1 \otimes ... \otimes !A_n \otimes B_1 \otimes ... \otimes B_m$$

using the maps ρ_Σ and ρ_Γ defined as

$$I \cong I \otimes ... \otimes I \xrightarrow{\gamma(f_1)\otimes ... \otimes \gamma(f_n)} !A_1 \otimes ... \otimes !A_n$$

and

$$I \cong I \otimes ... \otimes I \xrightarrow{l_1 \otimes ... \otimes l_m} B_1 \otimes ... \otimes B_m$$

respectively.

Proof: We proceed by induction on the derivation of the sequent. The substitution $[\overline{r}/\overline{x}]$ is denoted σ_Σ, the substitution $[\overline{s}/\overline{y}]$ is denoted σ_Γ and the substitution $[\overline{r},\overline{s}/\overline{x},\overline{y}]$ is denoted $\sigma_{\Sigma;\Gamma}$. We consider each case except symmetric ones. Note that since the term u is assumed to be without occurrences of the linear fixpoint constant, there is no case for that situation.

- In the case
$$\frac{}{\Sigma; y : B \Mapsto y : B}$$
we have to show
$$\cong; (\rho_\Sigma \otimes l); [\![y]\!] \preceq_B y\sigma_\Sigma[s/y].$$
But $\cong; (\rho_\Sigma \otimes l); [\![y]\!] = l$ and $y\sigma_\Sigma[s/y] = s$, and moreover, $l \preceq_B s$.

- In the case
$$\frac{}{x_1 : A_1, ..., x_n : A_n; \Mapsto x_q : A_q}$$
we have to show
$$\rho_\Sigma; [\![x_q]\!] \preceq_{A_q} x_q\sigma_\Sigma.$$
But $\rho_\Sigma; [\![x_q]\!] = f_q$ and $x_q\sigma_\Sigma = r_q$, and moreover, $f_q \preceq_{A_q} r_q$.

- In the case
$$\frac{}{\Sigma; \Mapsto \mathtt{zero} : N}$$
we have to show
$$\rho_\Sigma; [\![\mathtt{zero}]\!] \preceq_N \mathtt{zero}\sigma_\Sigma.$$
But this amounts to $zero \preceq_N \mathtt{zero}$.

- In the case
$$\frac{\Sigma; , \Mapsto u : N}{\Sigma; , \Mapsto \mathtt{succ}(u) : N}$$
we have to show
$$\rho_{\Sigma;\Gamma}; [\![\mathtt{succ}(u)]\!] \preceq_N \mathtt{succ}(u)\sigma_{\Sigma;\Gamma}.$$
Note that
$$\rho_{\Sigma;\Gamma}; [\![\mathtt{succ}(u)]\!] = \rho_{\Sigma;\Gamma}; [\![u]\!]; succ.$$
Assume $\rho_{\Sigma;\Gamma}; [\![\mathtt{succ}(u)]\!] \neq \perp$. We then have $\rho_{\Sigma;\Gamma}; [\![u]\!] \neq \perp$ which entails that $u\sigma_{\Sigma;\Gamma} \Downarrow \mathtt{succ}^n(\mathtt{zero})$ for some number n such that $\rho_{\Sigma;\Gamma}; [\![u]\!] = \tilde{n}$ cf. the induction hypothesis. We then get
$$\frac{u\sigma_{\Sigma;\Gamma} \Downarrow \mathtt{succ}^n(\mathtt{zero})}{\mathtt{succ}(u)\sigma_{\Sigma;\Gamma} \Downarrow \mathtt{succ}^{n+1}(\mathtt{zero})}$$
such that $\rho_{\Sigma;\Gamma}; [\![\mathtt{succ}(u)]\!] = \widetilde{n+1}$.

- In the case
$$\frac{\Sigma; , \Mapsto u : N}{\Sigma; , \Mapsto \mathtt{pred}(u) : N}$$
we have to show
$$\rho_{\Sigma;\Gamma}; [\![\mathtt{pred}(u)]\!] \preceq_N \mathtt{pred}(u)\sigma_{\Sigma;\Gamma}.$$
Note that
$$\rho_{\Sigma;\Gamma}; [\![\mathtt{pred}(u)]\!] = \rho_{\Sigma;\Gamma}; [\![u]\!]; pred.$$
Assume $\rho_{\Sigma;\Gamma}; [\![\mathtt{pred}(u)]\!] \neq \perp$. We then have $\rho_{\Sigma;\Gamma}; [\![u]\!] \neq \perp$ which entails that $u\sigma_{\Sigma;\Gamma} \Downarrow \mathtt{succ}^n(\mathtt{zero})$ for some number n such that $\rho_{\Sigma;\Gamma}; [\![u]\!] = \tilde{n}$ cf. the induction hypothesis. If $n = 0$ then we get
$$\frac{u\sigma_{\Sigma;\Gamma} \Downarrow \mathtt{zero}}{\mathtt{pred}(u)\sigma_{\Sigma;\Gamma} \Downarrow \mathtt{zero}}$$

such that $\rho_{\Sigma;\Gamma};[\![\mathtt{pred}(u)]\!] = zero$. If $n \geq 1$ then we get

$$\frac{u\sigma_{\Sigma;\Gamma} \Downarrow \mathtt{succ}^{n}(\mathtt{zero})}{\mathtt{pred}(u)\sigma_{\Sigma;\Gamma} \Downarrow \mathtt{succ}^{n-1}(\mathtt{zero})}$$

such that $\rho_{\Sigma;\Gamma};[\![\mathtt{pred}(u)]\!] = \widetilde{n \Leftrightarrow 1}$.

- In the case

$$\frac{\Sigma;\Lambda \Vdash u : N \qquad \Sigma;, \Vdash v : A \qquad \Sigma;, \Vdash w : A}{\Sigma;\Lambda,, \Vdash \mathtt{if}\ u\ \mathtt{then}\ v\ \mathtt{else}\ w : A}$$

we have to show

$$l \preceq_A (\mathtt{if}\ u\ \mathtt{then}\ v\ \mathtt{else}\ w)\sigma_{\Sigma;\Lambda,\Gamma}$$

where the map l is defined as

$$l = \rho_{\Sigma;\Lambda,\Gamma};[\![\mathtt{if}\ u\ \mathtt{then}\ v\ \mathtt{else}\ w]\!].$$

Note that

$$l = \cong;((\rho_{\Sigma;\Lambda};[\![u]\!]) \otimes \langle(\rho_{\Sigma;\Gamma};[\![v]\!]),(\rho_{\Sigma;\Gamma};[\![w]\!])\rangle);cond.$$

Assume $l \neq \perp$. We then have $\rho_{\Sigma;\Lambda};[\![u]\!] \neq \perp$ which entails that $u\sigma_{\Sigma;\Lambda} \Downarrow \mathtt{succ}^{n}(\mathtt{zero})$ for some number n such that $\rho_{\Sigma;\Lambda};[\![u]\!] = \tilde{n}$ cf. the induction hypothesis. If $n = 0$ then $l = \rho_{\Sigma;\Gamma};[\![v]\!]$ which entails that $v\sigma_{\Sigma;\Gamma} \Downarrow c$ for some value c such that $\rho_{\Sigma;\Gamma};[\![v]\!] \preceq^{\circ}_{A} c$ cf. the induction hypothesis. We then get

$$\frac{u\sigma_{\Sigma;\Lambda} \Downarrow \mathtt{zero} \qquad v\sigma_{\Sigma;\Gamma} \Downarrow c}{(\mathtt{if}\ u\ \mathtt{then}\ v\ \mathtt{else}\ w)\sigma_{\Sigma;\Lambda,\Gamma} \Downarrow c}$$

such that $l \preceq^{\circ}_{A} c$. If $n \geq 1$ then the situation is analogous.

- In the case

$$\frac{}{\Sigma; \Vdash * : I}$$

we have to show

$$\rho_{\Sigma};[\![\,*\,]\!] \preceq_B *\sigma_{\Sigma}.$$

But $\rho_{\Sigma};[\![\,*\,]\!] = id$ and $*\sigma_{\Sigma} = *$.

- In the case

$$\frac{\Sigma;\Lambda \Vdash w : I \qquad \Sigma;, \Vdash u : C}{\Sigma;, ,\Lambda \Vdash \mathtt{let}\ w\ \mathtt{be}\ *\ \mathtt{in}\ u : C}$$

we have to show

$$l \preceq_C (\mathtt{let}\ w\ \mathtt{be}\ *\ \mathtt{in}\ u)\sigma_{\Sigma;\Gamma,\Lambda}$$

where the map l is defined as

$$l = \rho_{\Sigma;\Gamma,\Lambda};[\![\mathtt{let}\ w\ \mathtt{be}\ *\ \mathtt{in}\ u]\!].$$

It can be shown that

$$l = \rho_{\Sigma;\Lambda};[\![w]\!];\rho_{\Sigma;\Gamma};[\![u]\!].$$

Assume $l \neq \perp$. We then have $\rho_{\Sigma;\Lambda};[\![w]\!] \neq \perp$ which according to the induction hypothesis entails that $w\sigma_{\Sigma;\Lambda} \Downarrow *$ and $\rho_{\Sigma;\Lambda};[\![w]\!] \preceq^{\circ}_{I} *$. This entails that $\rho_{\Sigma;\Lambda};[\![w]\!] = id$. We thus have $l = \rho_{\Sigma;\Gamma};[\![u]\!]$ which according to the induction hypothesis entails that $u\sigma_{\Sigma;\Gamma} \Downarrow c$ for some value c such that $\rho_{\Sigma;\Gamma};[\![u]\!] \preceq^{\circ}_{C} c$. We then get

$$\frac{w\sigma_{\Sigma;\Lambda} \Downarrow * \qquad u\sigma_{\Sigma;\Gamma} \Downarrow c}{(\mathtt{let}\ w\ \mathtt{be}\ *\ \mathtt{in}\ u)\sigma_{\Sigma;\Gamma,\Lambda} \Downarrow c}$$

such that $l \preceq^{\circ}_{C} c$.

- In the case
$$\frac{\Sigma; ,\ \Vdash u : A \qquad \Sigma; \Delta \Vdash v : B}{\Sigma; ,\ , \Delta \Vdash u \otimes v : A \otimes B}$$
we have to show
$$\rho_{\Sigma;\Gamma,\Delta}; [\![u \otimes v]\!] \preceq_{A\otimes B} (u \otimes v)\sigma_{\Sigma;\Gamma,\Delta}.$$
It can be shown that
$$\rho_{\Sigma;\Gamma,\Delta}; [\![u \otimes v]\!] = \cong; ((\rho_{\Sigma;\Gamma}; [\![u]\!]) \otimes (\rho_{\Sigma;\Delta}; [\![v]\!])).$$
Assume $\rho_{\Sigma;\Gamma,\Delta}; [\![u \otimes v]\!] \neq \bot$. We then have $\rho_{\Sigma;\Gamma}; [\![u]\!] \neq \bot$ and $\rho_{\Sigma;\Delta}; [\![v]\!] \neq \bot$ which according to the induction hypothesis entails that $u\sigma_{\Sigma;\Gamma} \Downarrow d$ and $v\sigma_{\Sigma;\Delta} \Downarrow e$ for some values d and e such that $\rho_{\Sigma;\Gamma}; [\![u]\!] \preceq^\circ_A d$ and $\rho_{\Sigma;\Delta}; [\![v]\!] \preceq^\circ_B e$. We then get
$$\frac{u\sigma_{\Sigma;\Gamma} \Downarrow d \qquad v\sigma_{\Sigma;\Delta} \Downarrow e}{(u \otimes v)\sigma_{\Sigma;\Gamma,\Delta} \Downarrow d \otimes e}$$
such that $\rho_{\Sigma;\Gamma,\Delta}; [\![u \otimes v]\!] \preceq^\circ_{A\otimes B} d \otimes e$.

- In the case
$$\frac{\Sigma; \Lambda \Vdash w : A \otimes B \qquad \Sigma; ,\ , x : A, y : B \Vdash u : C}{\Sigma; ,\ , \Lambda \Vdash \texttt{let } w \texttt{ be } x \otimes y \texttt{ in } u : C}$$
we have to show
$$l \preceq_C (\texttt{let } w \texttt{ be } x \otimes y \texttt{ in } u)\sigma_{\Sigma;\Gamma,\Lambda}$$
where the map l is defined as
$$l = \rho_{\Sigma;\Gamma,\Lambda}; [\![\texttt{let } w \texttt{ be } x \otimes y \texttt{ in } u]\!].$$
It can be shown that
$$l = \cong; (\rho_{\Sigma;\Gamma} \otimes (\rho_{\Sigma;\Lambda}; [\![w]\!])); [\![u]\!].$$
Assume $l \neq \bot$. We then have $\rho_{\Sigma;\Lambda}; [\![w]\!] \neq \bot$ which according to the induction hypothesis entails that $w\sigma_{\Sigma;\Lambda} \Downarrow d \otimes e$ for some value $d \otimes e$ such that $\rho_{\Sigma;\Lambda}; [\![w]\!] \preceq^\circ_{A\otimes B} d \otimes e$. This entails that $\rho_{\Sigma;\Lambda}; [\![w]\!] = \cong; (g \otimes h)$ for some maps g and h such that $g \preceq^\circ_A d$ and $h \preceq^\circ_B e$. We thus have $l = \cong; (\rho_{\Sigma;\Gamma} \otimes g \otimes h); [\![u]\!]$ which according to the induction hypothesis entails that $u\sigma_{\Sigma;\Gamma}[d, e/x, y] \Downarrow c$ for some value c such that $\cong; (\rho_{\Sigma;\Gamma} \otimes g \otimes h); [\![u]\!] \preceq^\circ_C c$. We then get
$$\frac{w\sigma_{\Sigma;\Lambda} \Downarrow d \otimes e \qquad u\sigma_{\Sigma;\Gamma}[d, e/x, y] \Downarrow c}{(\texttt{let } w \texttt{ be } x \otimes y \texttt{ in } u)\sigma_{\Sigma;\Gamma,\Lambda} \Downarrow c}$$
such that $l \preceq^\circ_C c$.

- In the case
$$\frac{\Sigma; ,\ , x : A \Vdash u : B}{\Sigma; ,\ \Vdash \lambda x.u : A \multimap B}$$
we have to show
$$\rho_{\Sigma;\Gamma}; [\![\lambda x.u]\!] \preceq_{A\multimap B} (\lambda x.u)\sigma_{\Sigma;\Gamma}.$$
First observe that $(\lambda x.u)\sigma_{\Sigma;\Gamma} \Downarrow (\lambda x.u)\sigma_{\Sigma;\Gamma}$. Assume $\rho_{\Sigma;\Gamma}; [\![\lambda x.u]\!] \neq \bot$. We then have to show that
$$\rho_{\Sigma;\Gamma}; [\![\lambda x.u]\!] \preceq^\circ_{A\multimap B} (\lambda x.u)\sigma_{\Sigma;\Gamma}$$
So assume that a map g and a value c is given such that $g \preceq^\circ_A c$; we then have to show that
$$\cong; (\rho_{\Sigma;\Gamma}; [\![\lambda x.u]\!] \otimes g); eval \preceq_B u\sigma_{\Sigma;\Gamma}[c/x]$$
which follows from
$$\cong; (\rho_{\Sigma;\Gamma}; [\![\lambda x.u]\!] \otimes g); eval = \cong; (\rho_{\Sigma;\Gamma} \otimes g); [\![u]\!].$$
cf. the induction hypothesis.

- In the case

$$\frac{\Sigma;\Lambda \Vdash f : A \multimap B \qquad \Sigma;\Delta \Vdash u : A}{\Sigma;\Lambda,\Delta \Vdash fu : B}$$

we have to show

$$\rho_{\Sigma;\Lambda,\Delta};[\![fu]\!] \preceq_B (fu)\sigma_{\Sigma;\Lambda,\Delta}.$$

It can be shown that

$$\rho_{\Sigma;\Lambda,\Delta};[\![fu]\!] = \cong;((\rho_{\Sigma;\Lambda};[\![f]\!]) \otimes (\rho_{\Sigma;\Delta};[\![u]\!]));eval.$$

Assume $\rho_{\Sigma;\Lambda,\Delta};[\![fu]\!] \neq \bot$. We then have $\rho_{\Sigma;\Lambda};[\![f]\!] \neq \bot$ and $\rho_{\Sigma;\Delta};[\![u]\!] \neq \bot$ which according to the induction hypothesis entails that $f\sigma_{\Sigma;\Lambda} \Downarrow \lambda x.t$ for some value $\lambda x.t$ such that $\rho_{\Sigma;\Lambda};[\![f]\!] \preceq^\circ_{A\multimap B} \lambda x.t$ and $u\sigma_{\Sigma;\Delta} \Downarrow d$ for some value d such that $\rho_{\Sigma;\Delta};[\![u]\!] \preceq^\circ_A d$. But we then have

$$\cong;((\rho_{\Sigma;\Lambda};[\![f]\!]) \otimes (\rho_{\Sigma;\Delta};[\![u]\!]));eval \preceq_B t[d/x]$$

which entails that $t[d/x] \Downarrow c$ for some value c such that

$$\cong;((\rho_{\Sigma;\Lambda};[\![f]\!]) \otimes (\rho_{\Sigma;\Delta};[\![u]\!]));eval \preceq^\circ_B c.$$

We then get

$$\frac{f\sigma_{\Sigma;\Lambda} \Downarrow \lambda x.t \qquad u\sigma_{\Sigma;\Delta} \Downarrow d \qquad t[d/x] \Downarrow c}{(fu)\sigma_{\Sigma;\Lambda,\Delta} \Downarrow c}$$

such that $\rho_{\Sigma;\Lambda,\Delta};[\![fu]\!] \preceq^\circ_B c$.

- In the case

$$\frac{\Sigma;,_1 \Vdash v_1 :!A_1, \ \ldots\ ,\Sigma;,_n \Vdash v_n :!A_n \qquad \Sigma;x_1 :!A_1,\ldots,x_n :!A_n \Vdash u : B}{\Sigma;,_1,\ldots,,_n \Vdash \mathtt{promote}\ v_1,\ldots,v_n\ \mathtt{for}\ x_1,\ldots,x_n\ \mathtt{in}\ u :!B}$$

we have to show

$$l \preceq_{!B} (\mathtt{promote}\ \overline{v}\ \mathtt{for}\ \overline{x}\ \mathtt{in}\ u)\sigma_{\Sigma;\Gamma_1,\ldots,\Gamma_n}$$

where the map l is defined as

$$l = \rho_{\Sigma;\Gamma_1,\ldots,\Gamma_n};[\![\mathtt{promote}\ \overline{v}\ \mathtt{for}\ \overline{x}\ \mathtt{in}\ u]\!].$$

It can be shown that

$$l = \cong;(\rho_\Sigma \otimes (\rho_{\Sigma;\Gamma_1};[\![v_1]\!]) \otimes \ldots \otimes (\rho_{\Sigma;\Gamma_n};[\![v_n]\!]));\gamma([\![u]\!]).$$

Assume $l \neq \bot$. For each $i \in \{1,\ldots,n\}$ we then have $\rho_{\Sigma;\Gamma_i};[\![v_i]\!] \neq \bot$ which according to the induction hypothesis entails that $v_i\sigma_{\Sigma;\Gamma_i} \Downarrow c_i$ for some value c_i such that $\rho_{\Sigma;\Gamma_i};[\![v_i]\!] \preceq^\circ_{!A_i} c_i$. We therefore have

$$\frac{v_1\sigma_{\Sigma;\Gamma_1} \Downarrow c_1, \ \ldots\ , v_n\sigma_{\Sigma;\Gamma_n} \Downarrow c_n}{(\mathtt{promote}\ \overline{v}\ \mathtt{for}\ \overline{x}\ \mathtt{in}\ u)\sigma_{\Sigma;\Gamma_1,\ldots,\Gamma_n} \Downarrow \mathtt{promote}\ \overline{c}\ \mathtt{for}\ \overline{x}\ \mathtt{in}\ (u\sigma_\Sigma)}$$

so we now have to show that

$$l \preceq^\circ_{!B} (\mathtt{promote}\ \overline{c}\ \mathtt{for}\ \overline{x}\ \mathtt{in}\ u)\sigma.$$

But l is a map of coalgebras because the map $\rho_{\Sigma;\Gamma_i};[\![v_i]\!]$ is a map of coalgebras for each $i \in \{1,\ldots,n\}$, and moreover

$$\cong;(\rho_\Sigma \otimes (\rho_{\Sigma;\Gamma_1};[\![v_1]\!]) \otimes \ldots \otimes (\rho_{\Sigma;\Gamma_n};[\![v_n]\!]));[\![u]\!] \preceq_B u\sigma_\Sigma[\overline{c}/\overline{x}]$$

cf. the induction hypothesis.

- In the case

$$\frac{\Sigma;\Lambda \Mapsto u :!B}{\Sigma;\Lambda \Mapsto \mathtt{derelict}(u) : B}$$

we have to show

$$\rho_{\Sigma;\Lambda};[\![\mathtt{derelict}(u)]\!] \preceq_B \mathtt{derelict}(u)\sigma_{\Sigma;\Lambda}.$$

Note that

$$\rho_{\Sigma;\Lambda};[\![\mathtt{derelict}(u)]\!] = \rho_{\Sigma;\Lambda};[\![u]\!];\varepsilon.$$

Assume $\rho_{\Sigma;\Lambda};[\![\mathtt{derelict}(u)]\!] \neq \bot$. We then have $\rho_{\Sigma;\Lambda};[\![u]\!] \neq \bot$ which according to the induction hypothesis entails that $u\sigma_{\Sigma;\Lambda} \Downarrow d$ for some value

$$d = \mathtt{promote}\ \overline{c}\ \mathtt{for}\ \overline{x}\ \mathtt{in}\ v$$

such that $\rho_{\Sigma;\Lambda};[\![u]\!] \preceq^{\circ}_{!B} d$. But we then have $\rho_{\Sigma;\Lambda};[\![u]\!];\varepsilon \preceq_B v[\overline{c}/\overline{x}]$ and thus $v[\overline{c}/\overline{x}] \Downarrow e$ for some value e such that $\rho_{\Sigma;\Lambda};[\![u]\!];\varepsilon \preceq^{\circ}_B e$. We then get

$$\frac{u\sigma_{\Sigma;\Lambda} \Downarrow \mathtt{promote}\ \overline{c}\ \mathtt{for}\ \overline{x}\ \mathtt{in}\ v \qquad v[\overline{c}/\overline{x}] \Downarrow e}{\mathtt{derelict}(u)\sigma_{\Sigma;\Lambda} \Downarrow e}$$

such that $\rho_{\Sigma;\Lambda};[\![\mathtt{derelict}(u)]\!] \preceq^{\circ}_B e$.

- In the case

$$\frac{\Sigma;\Lambda \Mapsto w :!A \qquad \Sigma;\ ,\ ,x :!A, y :!A \Mapsto u : B}{\Sigma;\ ,\ ,\Lambda \Mapsto \mathtt{copy}\ w\ \mathtt{as}\ x, y\ \mathtt{in}\ u : B}$$

we have to show

$$l \preceq_B (\mathtt{copy}\ w\ \mathtt{as}\ x, y\ \mathtt{in}\ u)\sigma_{\Sigma;\Gamma,\Lambda}$$

where the map l is defined as

$$l = \rho_{\Sigma;\Gamma,\Lambda};[\![\mathtt{copy}\ w\ \mathtt{as}\ x, y\ \mathtt{in}\ u]\!].$$

It can be shown that

$$l = \cong;(\rho_{\Sigma;\Gamma} \otimes (\rho_{\Sigma;\Lambda};[\![w]\!];d));[\![u]\!].$$

Assume $l \neq \bot$. We then have $\rho_{\Sigma;\Lambda};[\![w]\!] \neq \bot$ which according to the induction hypothesis entails that $w\sigma_{\Sigma;\Lambda} \Downarrow d$ for some value d such that $\rho_{\Sigma;\Lambda};[\![w]\!] \preceq^{\circ}_{!A} d$. This entails that $\rho_{\Sigma;\Lambda};[\![w]\!]$ is a map of coalgebras and thus

$$l = \cong;(\rho_{\Sigma;\Gamma} \otimes (\rho_{\Sigma;\Lambda};[\![w]\!]) \otimes (\rho_{\Sigma;\Lambda};[\![w]\!]));[\![u]\!]$$

which according to the induction hypothesis entails that $u\sigma_{\Sigma;\Gamma}[d, d/x, y] \Downarrow c$ for some value c such that

$$\cong;(\rho_{\Sigma;\Gamma} \otimes (\rho_{\Sigma;\Lambda};[\![w]\!]) \otimes (\rho_{\Sigma;\Lambda};[\![w]\!]));[\![u]\!] \preceq^{\circ}_B c.$$

We then get

$$\frac{w\sigma_{\Sigma;\Lambda} \Downarrow d \qquad u\sigma_{\Sigma;\Gamma}[d, d/x, y] \Downarrow c}{(\mathtt{copy}\ w\ \mathtt{as}\ x, y\ \mathtt{in}\ u)\sigma_{\Sigma;\Gamma,\Lambda} \Downarrow c}$$

such that $l \preceq^{\circ}_B c$.

- In the case

$$\frac{\Sigma;\Lambda \Mapsto w :!A \qquad \Sigma;\ ,\ \Mapsto u : B}{\Sigma;\ ,\ ,\Lambda \Mapsto \mathtt{discard}\ w\ \mathtt{in}\ u : B}$$

we have to show

$$l \preceq_B (\mathtt{discard}\ w\ \mathtt{in}\ u)\sigma_{\Sigma;\Gamma,\Lambda}$$

where the map l is defined as

$$l = \rho_{\Sigma;\Gamma,\Lambda};[\![\texttt{discard}\ w\ \texttt{in}\ u]\!].$$

It can be shown that

$$l = \rho_{\Sigma;\Lambda};[\![w]\!];e;\rho_{\Sigma;\Gamma};[\![u]\!].$$

Assume $l \neq \perp$. We then have $\rho_{\Sigma;\Lambda};[\![w]\!] \neq \perp$ which according to the induction hypothesis entails that $w\sigma_{\Sigma;\Lambda} \Downarrow d$ for some value d such that $\rho_{\Sigma;\Lambda};[\![w]\!] \preceq^{\circ}_{!A} d$. This entails that $\rho_{\Sigma;\Lambda};[\![w]\!]$ is a map of coalgebras and thus $l = \rho_{\Sigma;\Gamma};[\![u]\!]$ which according to the induction hypothesis entails that $u\sigma_{\Sigma;\Gamma} \Downarrow c$ for some value c such that $\rho_{\Sigma;\Gamma};[\![u]\!] \preceq^{\circ}_{B} c$. We then get

$$\frac{w\sigma_{\Sigma;\Lambda} \Downarrow d \qquad u\sigma_{\Sigma;\Gamma} \Downarrow c}{(\texttt{discard}\ w\ \texttt{in}\ u)\sigma_{\Sigma;\Gamma,\Lambda} \Downarrow c}$$

such that $l \preceq^{\circ}_{B} c$.

- In the case

$$\frac{\Sigma;,_1 \Vdash w_1 : A_1, \quad \ldots \quad, \Sigma;,_n \Vdash w_n : A_n}{\Sigma;,_1,\ldots,,_n \Vdash \texttt{true}(w_1,\ldots,w_n) : 1}$$

we have to show

$$l \preceq_1 \texttt{true}(w_1,\ldots,w_n)\sigma_{\Sigma;\Gamma_1,\ldots,\Gamma_n}$$

where the map l is defined as

$$l = \rho_{\Sigma;\Gamma_1,\ldots,\Gamma_n};[\![\texttt{true}(w_1,\ldots,w_n)]\!].$$

But this holds trivially as $l = \perp$.

- In the case

$$\frac{\Sigma;, \Vdash u : A \qquad \Sigma;, \Vdash v : B}{\Sigma;, \Vdash (u,v) : A \times B}$$

we have to show

$$\rho_{\Sigma;\Gamma};[\![(u,v)]\!] \preceq_{A\times B} (u,v)\sigma_{\Sigma;\Gamma}.$$

First observe that $(u,v)\sigma_{\Sigma;\Gamma} \Downarrow (u,v)\sigma_{\Sigma;\Gamma}$. Assume $\rho_{\Sigma;\Gamma};[\![(u,v)]\!] \neq \perp$. We then have to show that

$$\rho_{\Sigma;\Gamma};[\![(u,v)]\!] \preceq^{\circ}_{A\times B} (u,v)\sigma_{\Sigma;\Gamma}$$

which follows from

$$\rho_{\Sigma;\Gamma};[\![(u,v)]\!] = \langle(\rho_{\Sigma;\Gamma};[\![u]\!]),(\rho_{\Sigma;\Gamma};[\![v]\!])\rangle$$

cf. the induction hypothesis.

- In the case

$$\frac{\Sigma;\Lambda \Vdash u : A \times B}{\Sigma;\Lambda \Vdash \texttt{fst}(u) : A}$$

we have to show

$$\rho_{\Sigma;\Lambda};[\![\texttt{fst}(u)]\!] \preceq_A \texttt{fst}(u)\sigma_{\Sigma;\Lambda}.$$

Note that

$$\rho_{\Sigma;\Lambda};[\![\texttt{fst}(u)]\!] = \rho_{\Sigma;\Lambda};[\![u]\!];\pi_1.$$

Assume $\rho_{\Sigma;\Lambda};[\![\texttt{fst}(u)]\!] \neq \perp$. We then have $\rho_{\Sigma;\Lambda};[\![u]\!] \neq \perp$ which according to the induction hypothesis entails that $u\sigma_{\Sigma;\Lambda} \Downarrow (v,w)$ for some value (v,w) such that $\rho_{\Sigma;\Lambda};[\![u]\!] \preceq^{\circ}_{A\times B} (v,w)$. But we then have $\rho_{\Sigma;\Lambda};[\![u]\!];\pi_1 \preceq_A v$ and thus $v \Downarrow e$ for some value e such that $\rho_{\Sigma;\Lambda};[\![u]\!];\pi_1 \preceq^{\circ}_{A} e$. We then get

$$\frac{u\sigma_{\Sigma;\Lambda}(v,w) \qquad v \Downarrow e}{\texttt{fst}(u)\sigma_{\Sigma;\Lambda} \Downarrow e}$$

such that $\rho_{\Sigma;\Lambda};[\![\texttt{fst}(u)]\!] \preceq^{\circ}_{A} e$.

- In the case
$$\frac{\Sigma;\,_1 \Vdash w_1 : A_1, \quad \ldots \quad ,\Sigma;\,_n \Vdash w_n : A_n \qquad \Sigma;\Lambda \Vdash u : 0}{\Sigma;\,_1,\ldots,\,_n,\Lambda \Vdash \mathtt{false}(w_1,\ldots,w_n;u) : C}$$
we have to show
$$l \preceq_C \mathtt{false}(w_1,\ldots,w_n;u)\sigma_{\Sigma;\Gamma_1,\ldots,\Gamma_n;\Lambda}$$
where the map l is defined as
$$l = \rho_{\Sigma;\Gamma_1,\ldots,\Gamma_n;\Lambda};[\![\mathtt{false}(w_1,\ldots,w_n;u)]\!].$$
Note that
$$l = \cong;((\rho_{\Sigma;\Gamma_1};[\![w_1]\!]) \otimes \ldots \otimes (\rho_{\Sigma;\Gamma_n};[\![w_n]\!]) \otimes (\rho_{\Sigma;\Lambda};[\![u]\!]));\cong;[\,].$$
But we have $\rho_{\Sigma;\Lambda};[\![u]\!] \preceq u\sigma_{\Sigma;\Lambda}$ cf. the induction hypothesis, so $\rho_{\Sigma;\Lambda};[\![u]\!] = \perp$ and thus $l = \perp$.

- In the case
$$\frac{\Sigma;\,_,\, \Vdash u : A}{\Sigma;\,_,\, \Vdash \mathtt{inl}(u) : A + B}$$
we have to show
$$\rho_{\Sigma;\Gamma};[\![\mathtt{inl}(u)]\!] \preceq_{A+B} \mathtt{inl}(u)\sigma_{\Sigma;\Gamma}.$$
Note that
$$\rho_{\Sigma;\Gamma};[\![\mathtt{inl}(u)]\!] = \rho_{\Sigma;\Gamma};[\![u]\!];in_1.$$
Assume $\rho_{\Sigma;\Gamma};[\![\mathtt{inl}(u)]\!] \neq \perp$. We then have $\rho_{\Sigma;\Gamma};[\![u]\!] \neq \perp$ which according to the induction hypothesis entails that $u\sigma_{\Sigma;\Gamma} \Downarrow d$ for some value d such that $\rho_{\Sigma;\Gamma};[\![u]\!] \preceq^\circ_A d$. We then get
$$\frac{u\sigma_{\Sigma;\Gamma} \Downarrow d}{\mathtt{inl}(u)\sigma_{\Sigma;\Gamma} \Downarrow \mathtt{inl}(d)}$$
such that $\rho_{\Sigma;\Gamma};[\![\mathtt{inl}(u)]\!] \preceq^\circ_{A+B} \mathtt{inl}(d)$.

- In the case
$$\frac{\Sigma;\Lambda \Vdash v : A + B \qquad \Sigma;\,_,\,_,x : A \Vdash t : C \qquad \Sigma;\,_,\,_,y : B \Vdash u : C}{\Sigma;\,_,\,_,\Lambda \Vdash \mathtt{case}\ v\ \mathtt{of}\ \mathtt{inl}(x).t \mid \mathtt{inr}(y).u : C}$$
we have to show
$$l \preceq_C (\mathtt{case}\ v\ \mathtt{of}\ \mathtt{inl}(x).t \mid \mathtt{inr}(y).u)\sigma_{\Sigma;\Gamma,\Lambda}.$$
where the map l is defined as
$$l = \rho_{\Sigma;\Gamma,\Lambda};[\![\mathtt{case}\ v\ \mathtt{of}\ \mathtt{inl}(x).t \mid \mathtt{inr}(y).u]\!].$$
Note that
$$l = \rho_{\Sigma;\Lambda};[\![v]\!];[(\cong;(\rho_{\Sigma;\Gamma} \otimes A);[\![t]\!]),(\cong;(\rho_{\Sigma;\Gamma} \otimes B);[\![u]\!])].$$
Assume $l \neq \perp$. We then have $\rho_{\Sigma;\Lambda};[\![v]\!] \neq \perp$ which without loss of generality entails that $v\sigma_{\Sigma;\Lambda} \Downarrow \mathtt{inl}(d)$ for some value $\mathtt{inl}(d)$ such that $\rho_{\Sigma;\Lambda};[\![v]\!] \preceq^\circ_{A+B} \mathtt{inl}(d)$ cf. the induction hypothesis. This entails that $\rho_{\Sigma;\Lambda};[\![v]\!] = h;in_1$ for some map h such that $h \preceq^\circ_A d$. We thus have $l = \cong;(\rho_{\Sigma;\Gamma} \otimes h);[\![t]\!]$ which according to the induction hypothesis entails that $t\sigma_{\Sigma;\Gamma}[d/x] \Downarrow c$ for some value c such that $\cong;(\rho_{\Sigma;\Gamma} \otimes h);[\![t]\!] \preceq^\circ_C c$. We then get
$$\frac{v\sigma_{\Sigma;\Lambda} \Downarrow \mathtt{inl}(d) \qquad t\sigma_{\Sigma;\Gamma}[d/x] \Downarrow c}{(\mathtt{case}\ v\ \mathtt{of}\ \mathtt{inl}(x).t \mid \mathtt{inr}(y).u)\sigma_{\Sigma;\Gamma,\Lambda} \Downarrow c}$$
such that $l \preceq^\circ_C c$.

- In the case

$$\frac{}{\Sigma;\, \mathbb{I}\!\mapsto\!\Omega : A}$$

we have to show

$$\rho_\Sigma;\, [\![\Omega]\!] \preceq_A \Omega\sigma_\Sigma.$$

But this holds trivially as $\rho_\Sigma;\, [\![\Omega]\!] = \bot$.

□

The following lemma says that the (formal) finite approximants to an internal linear fixpoint operator are $\preceq$-related to the corresponding linear fixpoint constant:

Lemma 9.7.5 For every type A and number n we have $\gamma(\bot); K_A^n \preceq_{!(!A\multimap A)\multimap A} \mathtt{Y}_A$.

Proof: Recall that the map K_A is defined in the proof of Proposition 8.3.12. We proceed by induction on n. The assertion is clearly true in case $n = 0$. Assume that the assertion is true for an arbitrary number n, and assume that $\gamma(\bot); K^{n+1} \neq \bot$. We then have to show that

$$\begin{array}{l}\gamma(\bot); K^{n+1} \preceq^\circ_{!(!A\multimap A)\multimap A} \\ \lambda f.\mathtt{copy}\ f\ \mathtt{as}\ f', f''\ \mathtt{in}\ (\mathtt{derelict}(f')\mathtt{promote}\ f''\ \mathtt{for}\ f'''\ \mathtt{in}\ (\mathtt{Y}f'''))\end{array}$$

So assume we are given a map $g : I \to !(!A \multimap A)$ and a value

$$d = \mathtt{promote}\ \overline{c}\ \mathtt{for}\ \overline{x}\ \mathtt{in}\ u$$

such that $g \preceq^\circ_{!(!A\multimap A)} d$. Note that this entails that g is a map of coalgebras and $g;\varepsilon \preceq_{!A\multimap A} u[\overline{c}/\overline{x}]$. We then have to show that

$$\cong;((\gamma(\bot); K^{n+1}) \otimes g); eval \preceq_A d' \tag{9.1}$$

where

$$d' = \mathtt{copy}\ d\ \mathtt{as}\ f', f''\ \mathtt{in}\ (\mathtt{derelict}(f')\mathtt{promote}\ f''\ \mathtt{for}\ f'''\ \mathtt{in}\ (\mathtt{Y}f''')).$$

So assume $\cong;((\gamma(\bot); K^{n+1}) \otimes g); eval \neq \bot$. We have

$$\gamma(\bot); K^{n+1} = \ulcorner d; (\varepsilon \otimes \gamma(\ulcorner \gamma(\bot); K^{n\urcorner -1})); eval^\urcorner$$

cf. the proof of Proposition 8.3.12, so we get

$$\begin{array}{rcl}\cong;((\gamma(\bot); K^{n+1}) \otimes g); eval & = & \cong;(\ulcorner d; (\varepsilon \otimes \gamma(\ulcorner \gamma(\bot); K^{n\urcorner -1})); eval^\urcorner \otimes g); eval \\ & = & g; d; (\varepsilon \otimes \gamma(\ulcorner \gamma(\bot); K^{n\urcorner -1})); eval \\ & = & \cong;((g;\varepsilon) \otimes \gamma(g; \ulcorner \gamma(\bot); K^{n\urcorner -1})); eval \\ & = & \cong;((g;\varepsilon) \otimes \gamma(\cong;((\gamma(\bot); K^n) \otimes g); eval)); eval\end{array}$$

which entails that $g;\varepsilon \neq \bot$ and $\gamma(\cong;((\gamma(\bot); K^n) \otimes g); eval) \neq \bot$. Note that $g;\varepsilon \neq \bot$ entails $u[\overline{c}/\overline{x}] \Downarrow \lambda y.t$ for some value $\lambda y.t$ such that $g;\varepsilon \preceq^\circ_{!A\multimap A} \lambda y.t$.

We now want to show that

$$\gamma(\cong;((\gamma(\bot); K^n) \otimes g); eval) \preceq^\circ_{!A} d'' \tag{9.2}$$

where

$$d'' = \mathtt{promote}\ d\ \mathtt{for}\ f'''\ \mathtt{in}\ (\mathtt{Y}f''')$$

which amounts to

$$\cong;((\gamma(\bot); K^n) \otimes g); eval \preceq_A \mathtt{Y}d. \tag{9.3}$$

But $\cong;((\gamma(\bot); K^n) \otimes g); eval \neq \bot$ entails that $\gamma(\bot); K^n \neq \bot$ and thus cf. to the induction hypothesis

$$\begin{array}{l}\gamma(\bot); K^n \preceq^\circ_{!(!A\multimap A)\multimap A} \\ \lambda f.\mathtt{copy}\ f\ \mathtt{as}\ f', f''\ \mathtt{in}\ (\mathtt{derelict}(f')\mathtt{promote}\ f''\ \mathtt{for}\ f'''\ \mathtt{in}\ (\mathtt{Y}f'''))\end{array}$$

which entails that

$$\cong; ((\gamma(\bot); K^n) \otimes g); eval \preceq_A d'$$

because $g \preceq^{\circ}_{!(!A \multimap A)} d$ so we get $d' \Downarrow c$ for some value c and thus

$$\frac{\mathtt{Y} \Downarrow \lambda f.\mathtt{copy}\ f\ \mathtt{as}\ f', f''\ \mathtt{in}\ (\mathtt{derelict}(f')\mathtt{promote}\ f''\ \mathtt{for}\ f'''\ \mathtt{in}\ (\mathtt{Y}f''')) \qquad d \Downarrow d \qquad d' \Downarrow c}{\mathtt{Y}d \Downarrow c}$$

such that $\cong; ((\gamma(\bot); K^n) \otimes g); eval \preceq^{\circ}_A c$ and therefore (9.3). We conclude (9.2).

Now, (9.2) entails

$$\cong; ((g; \varepsilon) \otimes \gamma(\cong; ((\gamma(\bot); K^n) \otimes g); eval)); eval \preceq_A t[d''/y]$$

because $g; \varepsilon \preceq^{\circ}_{!A \multimap A} \lambda y.t$. But we have shown that

$$\cong; ((g; \varepsilon) \otimes \gamma(\cong; ((\gamma(\bot); K^n) \otimes g); eval)); eval = \cong; ((\gamma(\bot); K^{n+1}) \otimes g); eval$$

and we assume $\cong; ((\gamma(\bot); K^{n+1}) \otimes g); eval \neq \bot$ so $t[d''/y] \Downarrow e$ for some value e such that

$$\cong; ((\gamma(\bot); K^{n+1}) \otimes g); eval \preceq^{\circ}_A e. \tag{9.4}$$

By putting together the derivations obtained under the assumption that

$$\cong; ((\gamma(\bot); K^{n+1}) \otimes g); eval \neq \bot$$

we get the derivation

$$\dfrac{d \Downarrow d \qquad \dfrac{\dfrac{d \Downarrow d \qquad u[\overline{c}/\overline{x}] \Downarrow \lambda y.t}{\mathtt{derelict}(d) \Downarrow \lambda y.t} \qquad d'' \Downarrow d'' \qquad t[d''/y] \Downarrow e}{\mathtt{derelict}(d)d'' \Downarrow e}}{\mathtt{copy}\ d\ \mathtt{as}\ f', f''\ \mathtt{in}\ (\mathtt{derelict}(f')\mathtt{promote}\ f''\ \mathtt{for}\ f'''\ \mathtt{in}\ (\mathtt{Y}f''')) \Downarrow e}$$

which together with (9.4) entails (9.1). □

We are finally able to state a result expressing that the categorical interpretation is adequate with respect to the eager operational semantics. Note how Lemma 9.5.8 is used to "extract" the linear fixpoint constants of a term.

Theorem 9.7.6 (Eager Adequacy) Let u be a program of type B. If the linear fixpoint operator is rationally open with respect to B, then $[\![u]\!] \neq \bot$ entails that $u \Downarrow$.

Proof: We apply Lemma 9.5.8 to the derivable sequent $; \mathbb{I} \Rightarrow u : B$ of Generalised LPCF and obtain a derivable sequent

$$z_1 :!(!A_1 \multimap A_1) \multimap A_1,\ \ldots, z_n :!(!A_n \multimap A_n) \multimap A_n; \mathbb{I} \Rightarrow u' : C$$

of Generalised LPCF such that

$$u = u'[\mathtt{Y}_{A_1}, \ldots, \mathtt{Y}_{A_n}/z_1, \ldots, z_n]$$

and such that the term u' does not contain any occurrences of the linear fixpoint constant. Note that only a special case of Lemma 9.5.8 is used where a constant, namely the linear fixpoint constant, is replaced by a variable. But

$$[\![u'[\mathtt{Y}_{A_1}, \ldots, \mathtt{Y}_{A_n}/z_1, \ldots, z_n]]\!] = \cong; (\gamma([\![\mathtt{Y}_{A_1}]\!]) \otimes \ldots \otimes \gamma([\![\mathtt{Y}_{A_n}]\!])); [\![u']\!]$$

and for each $i \in \{1, \ldots, n\}$ we have $[\![\mathtt{Y}_{A_i}]\!] = K^{\sharp}_{A_i}$ so there exist numbers $p_1, \ldots, p_n$ such that

$$\cong; (\gamma(\gamma(\bot); K^{p_1}_{A_1}) \otimes \ldots \otimes \gamma(\gamma(\bot); K^{p_n}_{A_n})); [\![u']\!] \neq \bot$$

because the linear fixpoint operator is rationally open with respect to B and because $[\![u]\!] \neq \bot$. But for each $i \in \{1, ..., n\}$ it is the case that

$$\gamma(\bot); K^{p_i}_{A_i} \preceq_{(A_i \Rightarrow A_i) \Rightarrow A_i} \mathtt{Y}_{A_i}$$

according to Lemma 9.7.5 which entails that

$$\cong; (\gamma(\gamma(\bot); K^{p_1}_{A_1}) \otimes ... \otimes \gamma(\gamma(\bot); K^{p_n}_{A_n})); [\![u']\!] \preceq_B u'[\mathtt{Y}_{A_1}, ..., \mathtt{Y}_{A_n}/z_1, ..., z_n]$$

according to Lemma 9.7.4. We conclude that $u \Downarrow$ cf. the definition of $\preceq_B$. □

Examples 9.7.7 The eager adequacy result, Theorem 9.7.6, holds when LPCF is interpreted in the concrete linear categories $\mathbf{cpo}_{st}$ and $\mathbf{dI}_{lin}$.

We have thus proved an adequacy result where the eager operational semantics is considered. In the next section we shall prove adequacy where we consider the lazy operational semantics. The reason why both of the operational semantics give rise to adequacy is that the differences between them only occur at types where the evaluation strategy is not modelled by the categorical interpretation.

9.8 Lazy Adequacy

In this section we will show that the categorical interpretation is adequate when LPCF is equipped with the lazy operational semantics; it is analogous to the result of the previous section dealing with the eager operational semantics. We use the binary logical relations $\trianglelefteq^\circ_A$ and $\trianglelefteq_A$ given below. The following definition is similar to Definition 9.7.3. Recall that T_A is the set of programs of type A and C_A the set of values of type A.

Definition 9.8.1 For each type A the binary relations

$$\trianglelefteq^\circ_A \subseteq (\mathcal{C}(I, A) \Leftrightarrow \{\bot\}) \times C_A \qquad\qquad \trianglelefteq_A \subseteq \mathcal{C}(I, A) \times T_A$$

are defined by induction on the structure of A. The relation $\trianglelefteq^\circ_A$ is defined as

$$\begin{aligned}
f \trianglelefteq^\circ_N \mathtt{succ}^n(\mathtt{zero}) \quad &\text{iff} \quad f = \tilde{n} \\
f \trianglelefteq^\circ_I * \quad &\text{iff} \quad f = id \\
f \trianglelefteq^\circ_{B \otimes C} u \otimes v \quad &\text{iff} \quad \exists g \in \mathcal{C}(I, B). \exists h \in \mathcal{C}(I, C). \\
&\qquad f = \cong; (g \otimes h) \ \wedge \ g \trianglelefteq_B u \ \wedge \ h \trianglelefteq_C v \\
f \trianglelefteq^\circ_{B \multimap C} \lambda x.u \quad &\text{iff} \quad \forall g \in \mathcal{C}(I, B). \forall v \in T_B. \\
&\qquad g \trianglelefteq_B v \ \Rightarrow \cong; (f \otimes g); eval \trianglelefteq_C u[v/x] \\
f \trianglelefteq^\circ_{!B} \mathtt{promote}\ \overline{c}\ \mathtt{for}\ \overline{x}\ \mathtt{in}\ u \quad &\text{iff} \quad f; \delta = m_I; !f \ \wedge \ f; \varepsilon \trianglelefteq_B u[\overline{c}/\overline{x}] \\
f \trianglelefteq^\circ_{B \times C} (v, w) \quad &\text{iff} \quad f; \pi_1 \trianglelefteq_B v \ \wedge \ f; \pi_2 \trianglelefteq_C w \\
f \trianglelefteq^\circ_{B+C} \mathtt{inl}(u) \quad &\text{iff} \quad \exists h \in \mathcal{C}(1, B).\ f = h; in_1 \ \wedge \ h \trianglelefteq_B u \\
f \trianglelefteq^\circ_{B+C} \mathtt{inr}(v) \quad &\text{iff} \quad \exists h \in \mathcal{C}(1, C).\ f = h; in_2 \ \wedge \ h \trianglelefteq_C v
\end{aligned}$$

and the relation $\trianglelefteq_A$ is defined as

$$f \trianglelefteq_A u \quad \text{iff} \quad f \neq \bot \Rightarrow \ \exists c \in C_A.\ u \downarrow c \ \wedge \ f \trianglelefteq^\circ_A c.$$

Note that $f \trianglelefteq^{\circ}_{A} c$ entails $f \trianglelefteq_{A} c$ because $c \Downarrow c$ for any value c. In the definition there is no case for the $\trianglelefteq^{\circ}_{1}$ relation because every map $f : I \to 1$ is equal to $\bot$, and similarly, there is no case for the $\trianglelefteq^{\circ}_{0}$ relation because there are no values of type 0.

The following lemma is similar to Lemma 9.7.4.

Lemma 9.8.2 Assume that Σ denotes a context $x_1 : A_1, ..., x_n : A_n$ and , denotes a context $y_1 : B_1, ..., y_m : B_m$ and consider a derivable sequent

$$\Sigma; , \mathrel{|\!\!\mapsto} u : C$$

of Generalised LPCF such that the term u does not contain any occurrences of the linear fixpoint constant. Assume that for each $i \in \{1, ..., n\}$ we have a map $f_i : I \to A_i$ and a program r_i of type A_i such that $f_i \trianglelefteq_{A_i} r_i$, and similarly, assume that for each $j \in \{1, ..., m\}$ we have a map $l_j : I \to B_j$ and a program s_j of type B_j such that $l_j \trianglelefteq_{B_j} s_j$. We then have

$$\rho_{\Sigma;\Gamma}; [\![u]\!] \trianglelefteq_{C} u[\overline{r}, \overline{s}/\overline{x}, \overline{y}]$$

where the map $\rho_{\Sigma;\Gamma}$ is defined as

$$I \cong I \otimes I \xrightarrow{\rho_{\Sigma} \otimes \rho_{\Gamma}} !A_1 \otimes ... \otimes !A_n \otimes B_1 \otimes ... \otimes B_m$$

using the maps ρ_{Σ} and ρ_{Γ} defined as

$$I \cong I \otimes ... \otimes I \xrightarrow{\gamma(f_1) \otimes ... \otimes \gamma(f_n)} !A_1 \otimes ... \otimes !A_n$$

and

$$I \cong I \otimes ... \otimes I \xrightarrow{l_1 \otimes ... \otimes l_m} B_1 \otimes ... \otimes B_m$$

respectively.

Proof: We proceed by induction on the derivation of the sequent. We make use of some conventions from Lemma 9.7.4: The substitution $[\overline{r}/\overline{x}]$ is denoted σ_{Σ}, the substitution $[\overline{s}/\overline{y}]$ is denoted σ_{Γ} and the substitution $[\overline{r}, \overline{s}/\overline{x}, \overline{y}]$ is denoted $\sigma_{\Sigma;\Gamma}$. We consider each case except those covered by analogous cases of Lemma 9.7.4. Also symmetric cases are omitted.

- In the case

$$\frac{\Sigma; , \mathrel{|\!\!\mapsto} u : A \qquad \Sigma; \Delta \mathrel{|\!\!\mapsto} v : B}{\Sigma; , , \Delta \mathrel{|\!\!\mapsto} u \otimes v : A \otimes B}$$

  we have to show

$$\rho_{\Sigma;\Gamma,\Delta}; [\![u \otimes v]\!] \trianglelefteq_{A \otimes B} (u \otimes v)\sigma_{\Sigma;\Gamma,\Delta}.$$

  First observe that $(u \otimes v)\sigma_{\Sigma;\Gamma,\Delta} \downarrow (u \otimes v)\sigma_{\Sigma;\Gamma,\Delta}$. Assume $\rho_{\Sigma;\Gamma,\Delta}; [\![u \otimes v]\!] \neq \bot$. We then have to show that

$$\rho_{\Sigma;\Gamma,\Delta}; [\![u \otimes v]\!] \trianglelefteq^{\circ}_{A \otimes B} (u \otimes v)\sigma_{\Sigma;\Gamma,\Delta}$$

  which follows from

$$\rho_{\Sigma;\Gamma,\Delta}; [\![u \otimes v]\!] = \cong; ((\rho_{\Sigma;\Gamma}; [\![u]\!]) \otimes (\rho_{\Sigma;\Delta}; [\![v]\!]))$$

  cf. the induction hypothesis.

- In the case

$$\frac{\Sigma; \Lambda \mathrel{|\!\!\mapsto} w : A \otimes B \qquad \Sigma; , , x : A, y : B \mathrel{|\!\!\mapsto} t : C}{\Sigma; , , \Lambda \mathrel{|\!\!\mapsto} \texttt{let}\ w\ \texttt{be}\ x \otimes y\ \texttt{in}\ t : C}$$

  we have to show

$$l \trianglelefteq_{C} (\texttt{let}\ w\ \texttt{be}\ x \otimes y\ \texttt{in}\ t)\sigma_{\Sigma;\Gamma,\Lambda}$$

  where the map l is defined as

$$l = \rho_{\Sigma;\Gamma,\Lambda}; [\![\texttt{let}\ w\ \texttt{be}\ x \otimes y\ \texttt{in}\ t]\!].$$

It can be shown that

$$l = \cong; (\rho_{\Sigma;\Gamma} \otimes (\rho_{\Sigma;\Lambda}; [\![w]\!])); [\![t]\!].$$

Assume $l \neq \bot$. We then have $\rho_{\Sigma;\Lambda}; [\![w]\!] \neq \bot$ which according to the induction hypothesis entails that $w\sigma_{\Sigma;\Lambda} \downarrow u \otimes v$ for some value $u \otimes v$ such that $\rho_{\Sigma;\Lambda}; [\![w]\!] \trianglelefteq^{\circ}_{A \otimes B} u \otimes v$. This entails that $\rho_{\Sigma;\Lambda}; [\![w]\!] = \cong; (g \otimes h)$ for some maps g and h such that $g \trianglelefteq_A u$ and $h \trianglelefteq_B v$. We thus have $l = \cong; (\rho_{\Sigma;\Gamma} \otimes g \otimes h); [\![t]\!]$ which according to the induction hypothesis entails that $t\sigma_{\Sigma;\Gamma}[u, v/x, y] \downarrow c$ for some value c such that $\cong; (\rho_{\Sigma;\Gamma} \otimes g \otimes h); [\![t]\!] \trianglelefteq^{\circ}_C c$. We then get

$$\frac{w\sigma_{\Sigma;\Lambda} \downarrow u \otimes v \qquad t\sigma_{\Sigma;\Gamma}[u, v/x, y] \downarrow c}{(\texttt{let}\ w\ \texttt{be}\ x \otimes y\ \texttt{in}\ t)\sigma_{\Sigma;\Gamma,\Lambda} \downarrow c}$$

such that $l \trianglelefteq^{\circ}_C c$.

- In the case

$$\frac{\Sigma; , , x : A \Vdash u : B}{\Sigma; , \Vdash \lambda x.u : A \multimap B}$$

we have to show

$$\rho_{\Sigma;\Gamma}; [\![\lambda x.u]\!] \trianglelefteq_{A \multimap B} (\lambda x.u)\sigma_{\Sigma;\Gamma}.$$

First observe that $(\lambda x.u)\sigma_{\Sigma;\Gamma} \downarrow (\lambda x.u)\sigma_{\Sigma;\Gamma}$. Assume $\rho_{\Sigma;\Gamma}; [\![\lambda x.u]\!] \neq \bot$. We then have to show that

$$\rho_{\Sigma;\Gamma}; [\![\lambda x.u]\!] \trianglelefteq^{\circ}_{A \multimap B} (\lambda x.u)\sigma_{\Sigma;\Gamma}.$$

So assume that a map g and a program v is given such that $g \trianglelefteq_A v$; we then have to show that

$$\cong; (\rho_{\Sigma;\Gamma}; [\![\lambda x.u]\!] \otimes g); eval \trianglelefteq_B u\sigma_{\Sigma;\Gamma}[v/x]$$

which follows from

$$\cong; (\rho_{\Sigma;\Gamma}; [\![\lambda x.u]\!] \otimes g); eval = \cong; (\rho_{\Sigma;\Gamma} \otimes g); [\![u]\!].$$

cf. the induction hypothesis.

- In the case

$$\frac{\Sigma; \Lambda \Vdash f : A \multimap B \qquad \Sigma; \Delta \Vdash u : A}{\Sigma; \Lambda, \Delta \Vdash fu : B}$$

we have to show

$$\rho_{\Sigma;\Lambda,\Delta}; [\![fu]\!] \trianglelefteq_B (fu)\sigma_{\Sigma;\Lambda,\Delta}.$$

It can be shown that

$$\rho_{\Sigma;\Lambda,\Delta}; [\![fu]\!] = \cong; ((\rho_{\Sigma;\Lambda}; [\![f]\!]) \otimes (\rho_{\Sigma;\Delta}; [\![u]\!])); eval.$$

Assume $\rho_{\Sigma;\Lambda,\Delta}; [\![fu]\!] \neq \bot$. We then have $\rho_{\Sigma;\Lambda}; [\![f]\!] \neq \bot$ which according to the induction hypothesis entails that $f\sigma_{\Sigma;\Lambda} \downarrow \lambda x.t$ for some value $\lambda x.t$ such that $\rho_{\Sigma;\Lambda}; [\![f]\!] \trianglelefteq^{\circ}_{A \multimap B} \lambda x.t$. Moreover, $\rho_{\Sigma;\Delta}; [\![u]\!] \trianglelefteq_A u\sigma_{\Sigma;\Delta}$ cf. the induction hypothesis. But we then have

$$\cong; ((\rho_{\Sigma;\Lambda}; [\![f]\!]) \otimes (\rho_{\Sigma;\Delta}; [\![u]\!])); eval \trianglelefteq_B t[u\sigma_{\Sigma;\Delta}/x]$$

which entails that $t[u\sigma_{\Sigma;\Delta}/x] \downarrow c$ for some value c such that

$$\cong; ((\rho_{\Sigma;\Lambda}; [\![f]\!]) \otimes (\rho_{\Sigma;\Delta}; [\![u]\!])); eval \trianglelefteq^{\circ}_B c.$$

We then get

$$\frac{f\sigma_{\Sigma;\Lambda} \downarrow \lambda x.t \qquad t[u\sigma_{\Sigma;\Delta}/x] \downarrow c}{(fu)\sigma_{\Sigma;\Lambda,\Delta} \downarrow c}$$

such that $\rho_{\Sigma;\Lambda,\Delta}; [\![fu]\!] \trianglelefteq^{\circ}_B c$.

- In the case
$$\frac{\Sigma;\,_1 \Vdash w_1 : A_1,\ \ldots\ ,\Sigma;\,_n \Vdash w_n : A_n}{\Sigma;\,_1,\ldots,\,_n \Vdash \mathtt{true}(w_1,\ldots,w_n) : 1}$$
we have to show
$$l \trianglelefteq_1 \mathtt{true}(w_1,\ldots,w_n)\sigma_{\Sigma;\Gamma_1,\ldots,\Gamma_n}$$
where the map l is defined as
$$l = \rho_{\Sigma;\Gamma_1,\ldots,\Gamma_n};[\![\mathtt{true}(w_1,\ldots,w_n)]\!].$$
But this holds trivially as $l = \perp$.

- In the case
$$\frac{\Sigma;\, \Vdash u : A}{\Sigma;\, \Vdash \mathtt{inl}(u) : A + B}$$
we have to show
$$\rho_{\Sigma;\Gamma};[\![\mathtt{inl}(u)]\!] \trianglelefteq_{A+B} \mathtt{inl}(u)\sigma_{\Sigma;\Gamma}.$$
First observe that $\mathtt{inl}(u)\sigma_{\Sigma;\Gamma} \downarrow \mathtt{inl}(u)\sigma_{\Sigma;\Gamma}$. Assume $\rho_{\Sigma;\Gamma};[\![\mathtt{inl}(u)]\!] \neq \perp$. We then have to show that
$$\rho_{\Sigma;\Gamma};[\![\mathtt{inl}(u)]\!] \trianglelefteq^{\circ}_{A+B} \mathtt{inl}(u)\sigma_{\Sigma;\Gamma}$$
which follows from
$$\rho_{\Sigma;\Gamma};[\![\mathtt{inl}(u)]\!] = \rho_{\Sigma;\Gamma};[\![u]\!];in_1$$
cf. the induction hypothesis.

- In the case
$$\frac{\Sigma;\Lambda \Vdash v : A + B \qquad \Sigma;\,,\,,x : A \Vdash t : C \qquad \Sigma;\,,\,,y : B \Vdash u : C}{\Sigma;\,,\,,\Lambda \Vdash \mathtt{case}\ v\ \mathtt{of}\ \mathtt{inl}(x).t \mid \mathtt{inr}(y).u : C}$$
we have to show
$$l \trianglelefteq_C (\mathtt{case}\ v\ \mathtt{of}\ \mathtt{inl}(x).t \mid \mathtt{inr}(y).u)\sigma_{\Sigma;\Gamma,\Lambda}$$
where the map l is defined as
$$l = \rho_{\Sigma;\Gamma,\Lambda};[\![\mathtt{case}\ v\ \mathtt{of}\ \mathtt{inl}(x).t \mid \mathtt{inr}(y).u]\!].$$
Note that
$$l = \rho_{\Sigma;\Lambda};[\![v]\!];[(\cong;(\rho_{\Sigma;\Gamma} \otimes A);[\![t]\!]),(\cong;(\rho_{\Sigma;\Gamma} \otimes B);[\![u]\!])].$$
Assume $l \neq \perp$. We then have $\rho_{\Sigma;\Lambda};[\![v]\!] \neq \perp$ which without loss of generality entails that $v\sigma_{\Sigma;\Lambda} \downarrow \mathtt{inl}(u)$ for some value $\mathtt{inl}(u)$ such that $\rho_{\Sigma;\Lambda};[\![v]\!] \trianglelefteq^{\circ}_{A+B} \mathtt{inl}(u)$ cf. the induction hypothesis. This entails that $\rho_{\Sigma;\Lambda};[\![v]\!] = h;in_1$ for some map h such that $h \trianglelefteq_A u$. We thus have $l = \cong;(\rho_{\Sigma;\Gamma} \otimes h);[\![t]\!]$ which according to the induction hypothesis entails that $t\sigma_{\Sigma;\Gamma}[u/x] \downarrow c$ for some value c such that $\cong;(\rho_{\Sigma;\Gamma} \otimes h);[\![t]\!] \trianglelefteq^{\circ}_C c$. We then get
$$\frac{v\sigma_{\Sigma;\Lambda} \downarrow \mathtt{inl}(u) \qquad t\sigma_{\Sigma;\Gamma}[u/x] \downarrow c}{(\mathtt{case}\ v\ \mathtt{of}\ \mathtt{inl}(x).t \mid \mathtt{inr}(y).u)\sigma_{\Sigma;\Gamma,\Lambda} \downarrow c}$$
such that $l \trianglelefteq^{\circ}_C c$.

□

The following lemma is similar to Lemma 9.7.5; it says that the (formal) finite approximants to an internal linear fixpoint operator are $\trianglelefteq$-related to the corresponding linear fixpoint constant:

Lemma 9.8.3 For every type A and number n we have $\gamma(\perp);K^n_A \trianglelefteq_{!(!A\multimap A)\multimap A} \mathtt{Y}_A$.

Proof: Recall that the map K_A is defined in the proof of Proposition 8.3.12. We proceed by induction on n. The assertion is clearly true in case $n = 0$. Assume that the assertion is true for an arbitrary number n, and assume that $\gamma(\bot); K^{n+1} \neq \bot$. We then have to show that

$$\begin{array}{l}\gamma(\bot); K^{n+1} \trianglelefteq^{\circ}_{!(!A \multimap A) \multimap A} \\ \lambda f.\mathtt{copy}\ f\ \mathtt{as}\ f', f''\ \mathtt{in}\ (\mathtt{derelict}(f')\mathtt{promote}\ f''\ \mathtt{for}\ f'''\ \mathtt{in}\ (\mathtt{Y}f'''))\end{array}$$

So assume we are given a map $g : I \to !(!A \multimap A)$ and a program v such that $g \trianglelefteq_{!(!A \multimap A)} v$. We then have to show that

$$\begin{array}{l}\cong; ((\gamma(\bot); K^{n+1}) \otimes g); eval \trianglelefteq_A \\ \mathtt{copy}\ v\ \mathtt{as}\ f', f''\ \mathtt{in}\ (\mathtt{derelict}(f')\mathtt{promote}\ f''\ \mathtt{for}\ f'''\ \mathtt{in}\ (\mathtt{Y}f'''))\end{array} \tag{9.5}$$

So assume $\cong; ((\gamma(\bot); K^{n+1}) \otimes g); eval \neq \bot$. Then $g \neq \bot$ which entails that $v \downarrow d$ for some value

$$d = \mathtt{promote}\ \overline{c}\ \mathtt{for}\ \overline{x}\ \mathtt{in}\ u$$

such that $g \trianglelefteq^{\circ}_{!(!A \multimap A)} d$. Note that this entails that g is a map of coalgebras and $g; \varepsilon \trianglelefteq_{!A \multimap A} u[\overline{c}/\overline{x}]$. We have

$$\gamma(\bot); K^{n+1} = \ulcorner d; (\varepsilon \otimes \gamma(\ulcorner \gamma(\bot); K^{n \urcorner - 1})); eval \urcorner$$

cf. the proof of Proposition 8.3.12, so we get

$$\begin{array}{rcl}\cong; ((\gamma(\bot); K^{n+1}) \otimes g); eval & = & \cong; (\ulcorner d; (\varepsilon \otimes \gamma(\ulcorner \gamma(\bot); K^{n \urcorner - 1})); eval \urcorner \otimes g); eval \\ & = & g; d; (\varepsilon \otimes \gamma(\ulcorner \gamma(\bot); K^{n \urcorner - 1})); eval \\ & = & \cong; ((g; \varepsilon) \otimes \gamma(g; \ulcorner \gamma(\bot); K^{n \urcorner - 1})); eval \\ & = & \cong; ((g; \varepsilon) \otimes \gamma(\cong; ((\gamma(\bot); K^n) \otimes g); eval)); eval\end{array}$$

which entails that $g; \varepsilon \neq \bot$ and $\gamma(\cong; ((\gamma(\bot); K^n) \otimes g); eval) \neq \bot$. Note that $g; \varepsilon \neq \bot$ entails $u[\overline{c}/\overline{x}] \downarrow \lambda y.t$ for some value $\lambda y.t$ such that $g; \varepsilon \trianglelefteq^{\circ}_{!A \multimap A} \lambda y.t$.

We now want to show that

$$\gamma(\cong; ((\gamma(\bot); K^n) \otimes g); eval) \trianglelefteq^{\circ}_{!A} d'' \tag{9.6}$$

where

$$d'' = \mathtt{promote}\ d\ \mathtt{for}\ f'''\ \mathtt{in}\ (\mathtt{Y}f''')$$

which amounts to

$$\cong; ((\gamma(\bot); K^n) \otimes g); eval \trianglelefteq_A \mathtt{Y}d. \tag{9.7}$$

But $\cong; ((\gamma(\bot); K^n) \otimes g); eval \neq \bot$ entails that $\gamma(\bot); K^n \neq \bot$ and thus cf. the induction hypothesis

$$\begin{array}{l}\gamma(\bot); K^n \trianglelefteq^{\circ}_{!(!A \multimap A) \multimap A} \\ \lambda f.\mathtt{copy}\ f\ \mathtt{as}\ f', f''\ \mathtt{in}\ (\mathtt{derelict}(f')\mathtt{promote}\ f''\ \mathtt{for}\ f'''\ \mathtt{in}\ (\mathtt{Y}f'''))\end{array}$$

which entails that

$$\cong; ((\gamma(\bot); K^n) \otimes g); eval \trianglelefteq_A d'$$

where

$$d' = \mathtt{copy}\ d\ \mathtt{as}\ f', f''\ \mathtt{in}\ (\mathtt{derelict}(f')\mathtt{promote}\ f''\ \mathtt{for}\ f'''\ \mathtt{in}\ (\mathtt{Y}f'''))$$

because $g \trianglelefteq^{\circ}_{!(!A \multimap A)} d$ so we get $d' \downarrow c$ for some value c and thus

$$\frac{\mathtt{Y} \downarrow \lambda f.\mathtt{copy}\ f\ \mathtt{as}\ f', f''\ \mathtt{in}\ (\mathtt{derelict}(f')\mathtt{promote}\ f''\ \mathtt{for}\ f'''\ \mathtt{in}\ (\mathtt{Y}f''')) \qquad d \downarrow d \qquad d' \downarrow c}{\mathtt{Y}d \downarrow c}$$

such that $\cong; ((\gamma(\bot); K^n) \otimes g); eval \trianglelefteq^{\circ}_A c$ and therefore (9.7). We conclude (9.6).

Now, (9.6) entails

$$\cong;((g;\varepsilon)\otimes\gamma(\cong;((\gamma(\bot);K^n)\otimes g);eval));eval \trianglelefteq_A t[d''/y]$$

because $g;\varepsilon \trianglelefteq^{\circ}_{!A\multimap A} \lambda y.t$. But we have shown that

$$\cong;((g;\varepsilon)\otimes\gamma(\cong;((\gamma(\bot);K^n)\otimes g);eval));eval = \cong;((\gamma(\bot);K^{n+1})\otimes g);eval$$

and we assume $\cong;((\gamma(\bot);K^{n+1})\otimes g);eval \neq \bot$ so $t[d''/y]\downarrow e$ for some value e such that

$$\cong;((\gamma(\bot);K^{n+1})\otimes g);eval \trianglelefteq^{\circ}_A e. \tag{9.8}$$

By putting together the derivations obtained under the assumption that

$$\cong;((\gamma(\bot);K^{n+1})\otimes g);eval \neq \bot$$

we get the derivation

$$\dfrac{v\downarrow d \qquad \dfrac{\dfrac{d\downarrow d \quad u[\overline{c}/\overline{x}]\downarrow\lambda y.t}{\texttt{derelict}(d)\downarrow\lambda y.t} \qquad d''\downarrow d'' \quad t[d''/y]\downarrow e}{\texttt{derelict}(d)d''\downarrow e}}{\texttt{copy } v \texttt{ as } f', f'' \texttt{ in } (\texttt{derelict}(f')\texttt{promote } f'' \texttt{ for } f''' \texttt{ in } (\texttt{Y}f'''))\downarrow e}$$

which together with (9.8) entails (9.5). □

We are finally able to state a result expressing that the categorical interpretation is adequate with respect to the lazy operational semantics.

Theorem 9.8.4 (Lazy Adequacy) Let u be a program of type B. If the linear fixpoint operator is rationally open with respect to B, then $[\![u]\!]\neq\bot$ entails that $u\downarrow$.

Proof: Analogous to the proof of Theorem 9.7.6. □

Examples 9.8.5 The lazy adequacy result, Theorem 9.8.4, holds when LPCF is interpreted in the concrete linear categories $\mathbf{cpo}_{st}$ and $\mathbf{dI}_{lin}$.

Together with the adequacy result of the previous section we have thus shown that the categorical semantics is adequate with respect to the eager as well as the lazy operational semantics. We discuss the significance of this result further in the next section

9.9 Observable Types

Types where we have a converse to eager adequacy, that is, to Theorem 9.7.6, will be called *observable*. To be explicit, a type B is observable iff $u\Downarrow$ entails that $[\![u]\!]\neq\bot$ for any program u of type B when we assume that $\tilde{0}\neq\tilde{1}$. This is analogous to the notion of observable types for PCF. Formally, we have defined the notion of observable types using the eager operational semantics, but this choice does not matter because the considerations below would have been the same had we taken the lazy operational semantics instead.

Now, the ground type N is observable according to soundness and the observation that if $\tilde{n}=\bot$ for some number n then $\tilde{0}=\tilde{1}$. It is actually possible to obtain a more informative result:

Corollary 9.9.1 Let u be a program of type N. If $\tilde{0}\neq\tilde{1}$ and the fixpoint operator is rationally open with respect to N, then $[\![u]\!]=\tilde{q}$ iff $u\Downarrow \texttt{succ}^q(\texttt{zero})$.

Proof: The result follows from Theorem 9.4.4 and Theorem 9.7.6 together with the observation that if $\tilde{n} = \perp$ for some number n, or if $\tilde{p} = \tilde{q}$ for different numbers p and q, then $\tilde{0} = \tilde{1}$. □

Also ! types are observable, which follows from soundness and the observation that if $\perp\colon I \to !A$ is a map of coalgebras, then $\tilde{0} = \tilde{1}$. The type I is observable according to soundness and the observation that if $\perp\colon I \to I$ is equal to the identity, then $\tilde{0} = \tilde{1}$. Furthermore, the unary sum type 0 is observable, which follows from the observation that we cannot have $t \Downarrow$, where t is a program of type 0, as there are no values of type 0.

So the types $(N, I, !, 0)$ are observable. Product and exponential types are not observable as the following examples show: The programs $\mathtt{true}()$ and (Ω, Ω) of product types are values, but are interpreted as $\perp$, and similarly, the program $\lambda x.\mathtt{discard}\ x\ \mathtt{in}\ \Omega$ of exponential type is a value, but is interpreted as $\perp$. Neither are $\otimes$ types observable; the program $\mathtt{true}() \otimes \mathtt{true}()$ is a value, but has interpretation $\perp$. Binary sum types are not observable; the program $\mathtt{inl}(\mathtt{true}())$ is a value, but is interpreted as $\perp$.

The choice of evaluation strategy does not matter for the notion of observable types. We can actually use this observation together with adequacy to show a purely syntactic result saying that the choice of evaluation strategy does not matter for the observable behaviour of programs of observable types.

Theorem 9.9.2 If t is a program of observable type, then $t \Downarrow$ iff $t \downarrow$.

Proof: In what follows we will interpret LPCF in the concrete linear category $\mathbf{cpo}_{st}$; this is a model for LPCF in the sense of Definition 9.7.1 such that the linear fixpoint operator is rationally open with respect to the interpretation of any observable type, as is made clear in Section 9.7 and Section 8.3. We then have $t \Downarrow$ iff $[\![t]\!] \neq \perp$ cf. eager adequacy and its converse. But $[\![t]\!] \neq \perp$ iff $t \downarrow$ cf. lazy adequacy and its converse. □

At ground type N it is possible to obtain a more informative result: If t is a program of ground type N then $t \Downarrow c$ iff $t \downarrow c$. The proof of this result is similar to the proof of Theorem 9.9.2.

There are two reasons why the observable behaviour of a program of observable type is the same whichever of the evaluation strategies is chosen: The linear character of LPCF and the fact that in both of the operational semantics a program of ! type is always evaluated before it is discarded. So it is simply impossible to get rid of a non-terminating program without trying to evaluate it. For example, in the evaluation rule for application it does not matter whether or not an argument is evaluated before it is plugged into the body u of an abstraction $\lambda x.u$ because linearity entails that the variable x has to occur in u.

So we have shown that the choice of operational semantics does not matter for the observable behaviour of programs of observable types. One might think that this contradicts the point of view put forward in [Abr90] that the linear context motivates a certain choice of evaluation rules, but this is not quite true as the considerations of the mentioned article include the question of efficiency, with respect to which the evaluation strategies of the article are sensible.

9.10 Unwinding

We will now use the adequacy result of the previous section to give an unwinding theorem for LPCF in a way analogous to the unwinding theorem for PCF. In the considerations below we have used the eager operational semantics, but this choice is not essential; the lazy operational semantics would do as well. Note how the statement and proof of the unwinding theorem hinges crucially on the intuitionistic variables of Generalised LPCF. We first need a convention: For any type A we define a program $\mathtt{Y}^n_A$ of type $!(!A \multimap A) \multimap A$ for every number n by the stipulations

$$\begin{array}{rcl} \mathtt{Y}^0 & = & \Omega \\ \mathtt{Y}^{n+1} & = & \lambda f.\mathtt{copy}\ f\ \mathtt{as}\ g, h\ \mathtt{in}\ (\mathtt{derelict}(g)\mathtt{promote}\ h\ \mathtt{for}\ k\ \mathtt{in}\ (\mathtt{Y}^n k)) \end{array}$$

It can be shown by a small induction proof that $[\![\mathtt{Y}^n]\!] = \gamma(\perp); K^n$ where the map K is defined in the proof of Proposition 8.3.12.

Theorem 9.10.1 (Unwinding) If t is a term of observable type with one free intuitionistic variable z of type $!(!A \multimap A) \multimap A$ and no free linear variables, then

$$t[\mathtt{Y}/z] \Downarrow \;\Leftrightarrow\; \exists n \in \omega\,.\, t[\mathtt{Y}^n/z] \Downarrow .$$

Proof: In what follows we will interpret LPCF in the concrete linear category $\mathbf{cpo}_{st}$; this is a model for LPCF in the sense of Definition 9.7.1 such that the linear fixpoint operator is rationally open with respect to the interpretation of any observable type, as is made clear in Section 9.7 and Section 8.3. We then have $t[\mathtt{Y}/z] \Downarrow$ iff $[\![t[\mathtt{Y}/z]]\!] \neq \bot$ cf. adequacy and its converse. But $[\![t[\mathtt{Y}/z]]\!]$ is a least upper bound for the increasing chain $\{[\![t[\mathtt{Y}^n/z]]\!]\}_{n\in\omega}$ because $[\![t[\mathtt{Y}/z]]\!] = \gamma(K^\sharp); [\![t]\!]$ and $[\![t[\mathtt{Y}^n/z]]\!] = \gamma(\gamma(\bot); K^n); [\![t]\!]$ according to the remark above, so $[\![t[\mathtt{Y}/z]]\!] \neq \bot$ iff there exists a number n such that $[\![t[\mathtt{Y}^n/z]]\!] \neq \bot$, that is, iff there exists a number n such that $t[\mathtt{Y}^n/z] \Downarrow$ cf. adequacy and its converse. □

At ground type N it is possible to obtain a more informative result: If t is a term of ground type N with one free intuitionistic variable z of type $!(!A \multimap A) \multimap A$ and no free linear variables, then $t[\mathtt{Y}/z] \Downarrow c$ iff there exists a number n such that $t[\mathtt{Y}^n/z] \Downarrow c$. The proof of this result is similar to the proof of Theorem 9.10.1.

It is possible to weaken the assumption of rational openness such that it is not only sufficient, but also necessary for the interpretation to be adequate; this is analogous to Theorem 7.5.2 for PCF. Note that the "upwards" direction of the following theorem relies on the unwinding theorem.

Theorem 9.10.2 Assume that we have a category that is a model for LPCF in the sense of Definition 9.7.1 such that $\tilde{0} \neq \tilde{1}$. Let an observable type B be given, then the assertions

- for all types A and terms t of type B with one free intuitionistic variable z having the type $!(!A \multimap A) \multimap A$ and no free linear variables it is the case that

$$\gamma(K^\sharp); [\![t]\!] \neq \bot \;\Rightarrow\; \exists n \in \omega\,.\, \gamma(\gamma(\bot); K^n); [\![t]\!] \neq \bot,$$

- for any program u of type B we have $[\![u]\!] \neq \bot$ entails that $u \Downarrow$,

are equivalent.

Proof: The "downwards" direction comes from the observation that the assumption made here is sufficient to prove Theorem 9.7.6 in the relevant case. The proof of the "upwards" direction goes as follows: Assume that the assumption made here holds, and consider a term t of type B with one free intuitionistic variable z of type $!(!A \multimap A) \multimap A$ and no free linear variables. If $\gamma(K^\sharp); [\![t]\!] \neq \bot$ then $t[\mathtt{Y}/z] \Downarrow$ according to the assumption because $\gamma(K^\sharp); [\![t]\!] = [\![t[\mathtt{Y}/z]]\!]$. But then there exists a number n such that $t[\mathtt{Y}^n/z] \Downarrow$ cf. the unwinding theorem, which entails the existence of a number n such that $\gamma(\gamma(\bot); K^n); [\![t]\!] \neq \bot$ as the type B is observable and $[\![t[\mathtt{Y}^n/z]]\!] = \gamma(\gamma(\bot); K^n); [\![t]\!]$. □

Chapter 10

Extension of the Girard Translation

In this chapter the Girard Translation of Chapter 5 is extended to a translation from PCF to LPCF. The syntactic matters are dealt with in Section 10.1 and in Section 10.2 the extended Girard Translation is shown to be sound with respect to the categorical interpretations induced by an appropriate categorical model. Using the soundness result together with adequacy for PCF and LPCF it is shown in Section 10.3 that the extended Girard Translation preserves and reflects evaluation of programs of numerals type.

10.1 Syntax

In this section we will extend the Girard Translation to a translation from PCF to LPCF. To be precise: In Section 5.1 it is shown how the Girard Translation corresponds to a translation from types and derivable sequents in the λ-calculus to types and derivable sequents in the linear λ-calculus via the Curry-Howard isomorphisms. A type A is translated into a type A^o and a sequent

$$x_1 : A_1, ..., x_n : A_n \vdash u : B$$

is translated into a sequent

$$x_1 :!A_1^o, ..., x_n :!A_n^o \mathbin{!\!\Rightarrow} u^o : B^o.$$

We have seen that PCF is an extension of the λ-calculus, and similarly, we have seen that LPCF is an extension of the linear λ-calculus. It will now be shown how to extend the Girard Translation to a translation from types and derivable sequents of PCF into types and derivable sequents of LPCF. At the level of types the translation is extended as follows:

$$N^o \quad = \quad N.$$

At the level of derivable sequents the translation is extended below. Each case is considered.

- A derivation

$$\frac{}{\overline{x} : , \vdash \mathtt{zero} : N}$$

  is translated into

$$\frac{\dfrac{}{\mathbin{!\!\Rightarrow}\mathtt{zero} : N}}{\overline{x} :!, {}^o \mathbin{!\!\Rightarrow}\mathtt{discard}\ \overline{x}\ \mathtt{in}\ (\mathtt{zero}) : N}$$

- A derivation

$$\frac{\overline{x} : , \vdash u : N}{\overline{x} : , \vdash \mathtt{succ}(u) : N}$$

is translated into

$$\frac{\overline{x} :!, {}^{o}\!\blacktriangleright\!\!\blacktriangleleft u^{o} : N}{\overline{x} :!, {}^{o}\!\blacktriangleright\!\!\blacktriangleleft \mathtt{succ}(u^{o}) : N}$$

- A derivation

$$\frac{\overline{x} : , \vdash u : N}{\overline{x} : , \vdash \mathtt{pred}(u) : N}$$

is translated into

$$\frac{\overline{x} :!, {}^{o}\!\blacktriangleright\!\!\blacktriangleleft u^{o} : N}{\overline{x} :!, {}^{o}\!\blacktriangleright\!\!\blacktriangleleft \mathtt{pred}(u^{o}) : N}$$

- A derivation

$$\frac{\overline{x} : , \vdash u : N \quad \overline{x} : , \vdash v : A \quad \overline{x} : , \vdash w : A}{\overline{x} : , \vdash \mathtt{if}\ u\ \mathtt{then}\ v\ \mathtt{else}\ w : A}$$

is translated into

$$\frac{\dfrac{\overline{x'} :!, {}^{o}\!\blacktriangleright\!\!\blacktriangleleft u^{o} : N \quad \overline{x''} :!, {}^{o}\!\blacktriangleright\!\!\blacktriangleleft v^{o} : A^{o} \quad \overline{x''} :!, {}^{o}\!\blacktriangleright\!\!\blacktriangleleft w^{o} : A^{o}}{\overline{x'} :!, {}^{o}, \overline{x''} :!, {}^{o}\!\blacktriangleright\!\!\blacktriangleleft \mathtt{if}\ u^{o}\ \mathtt{then}\ v^{o}\ \mathtt{else}\ w^{o} : A^{o}}}{\overline{x} :!, {}^{o}\!\blacktriangleright\!\!\blacktriangleleft \mathtt{copy}\ \overline{x}\ \mathtt{as}\ \overline{x'}, \overline{x''}\ \mathtt{in}\ (\mathtt{if}\ u^{o}\ \mathtt{then}\ v^{o}\ \mathtt{else}\ w^{o}) : A^{o}}$$

- A derivation

$$\frac{}{\overline{x} : , \vdash \Omega : A}$$

is translated into

$$\frac{\dfrac{}{\blacktriangleright\!\!\blacktriangleleft \Omega : A^{o}}}{\overline{x} :!, {}^{o}\!\blacktriangleright\!\!\blacktriangleleft \mathtt{discard}\ \overline{x}\ \mathtt{in}\ \Omega : A^{o}}$$

- A derivation

$$\frac{}{\overline{x} : , \vdash \mathtt{Y} : (A \Rightarrow A) \Rightarrow A}$$

is translated into

$$\frac{\dfrac{}{\blacktriangleright\!\!\blacktriangleleft \mathtt{Y} :!(!A^{o} \multimap A^{o}) \multimap A^{o}}}{\overline{x} :!, {}^{o}\!\blacktriangleright\!\!\blacktriangleleft \mathtt{discard}\ \overline{x}\ \mathtt{in}\ \mathtt{Y} :!(!A^{o} \multimap A^{o}) \multimap A^{o}}$$

10.2 Soundness

In this section we will extend the soundness result of Section 5.2 to dealing with the translation from types and derivable sequents of PCF into to types and derivable sequents of LPCF obtained by extending the Girard Translation as appropriate. Assume that we have a categorical premodel $\mathcal{C}$ for LPCF in the sense of Definition 9.4.1; we can then interpret types and derivable sequents of LPCF as objects and maps in $\mathcal{C}$. The Kleisli category induced by the ! comonad is a categorical premodel for PCF in the sense of Definition 7.2.1 according to Section 8.1, Section 8.2 and Section 8.3, so furthermore we can interpret types and derivable sequents of PCF as objects and maps in the Kleisli category. We then have the following result:

Proposition 10.2.1 Let $\mathcal{C}$ be a categorical premodel for LPCF. If a type A of PCF is interpreted in the Kleisli category, and the type A^{o} of LPCF is interpreted in $\mathcal{C}$, then $[\![A]\!] = [\![A^{o}]\!]$.

Proof: The proof of Proposition 5.2.1 is extended in the obvious way. □

Analogous to in Section 5.2 we define lin to be the composition of bijections between maps

$$\begin{array}{rcll} \mathcal{C}_!([\![A_1]\!] \times \ldots \times [\![A_n]\!], [\![B]\!]) & = & \mathcal{C}_!([\![A_1^o]\!] \times \ldots \times [\![A_n^o]\!], [\![B^o]\!]) & \text{by Proposition 10.2.1} \\ & = & \mathcal{C}(!([\![A_1^o]\!] \times \ldots \times [\![A_n^o]\!]), [\![B^o]\!]) & \text{because } U_! \dashv F_! \\ & \cong & \mathcal{C}(![\![A_1^o]\!] \otimes \ldots \otimes ![\![A_n^o]\!]), [\![B^o]\!]) & \text{by composition with } n \end{array}$$

where $A_1, ..., A_n$ and B are types PCF. Recall that a derivable sequent

$$x_1 : A_1, ..., x_n : A_n \vdash t : B$$

of PCF is translated into the derivable sequent

$$x_1 : !A_1^o, ..., x_n : !A_n^o \mapsfrom t^o : B^o$$

of LPCF, and observe that the maps $[\![t]\!]$ and $[\![t^o]\!]$ live in the domain and the codomain of the function lin, respectively.

Theorem 10.2.2 (Soundness) Let $\mathcal{C}$ be a categorical premodel for LPCF. If the sequent

$$x_1 : A_1, ..., x_n : A_n \vdash t : B$$

is derivable in PCF, then $lin([\![t]\!]) = [\![t^o]\!]$.

Proof: The proof of Theorem 5.2.2 is extended as appropriate. Induction on the derivation of the sequent $x_1 : A_1, ..., x_n : A_n \vdash t : B$. We proceed case by case.

- In the case

$$\frac{}{\overline{x} : , \ \vdash \mathtt{zero} : N}$$

the following calculation suffices:

$$\begin{array}{rcll} lin([\![\mathtt{zero}]\!]) & = & n; \gamma(\langle\rangle); zero & \\ & = & n; \gamma(\langle\rangle); \cong; zero & \text{by def. of } zero \text{ in } \mathcal{C}_! \\ & = & (e \otimes ... \otimes e); \cong; zero & \text{Note 1.} \\ & = & [\![\mathtt{discard}\ \overline{x}\ \mathtt{in}\ (\mathtt{zero})]\!] & \\ & = & [\![\mathtt{zero}^o]\!] & \end{array}$$

Note 1. Because I is a terminal object in $\mathcal{C}_!$ cf. Section 2.3.

- In the case

$$\frac{\overline{x} : , \ \vdash u : N}{\overline{x} : , \ \vdash \mathtt{succ}(u) : N}$$

the following calculation suffices:

$$\begin{array}{rcll} lin([\![\mathtt{succ}(u)]\!]) & = & n; \gamma([\![u]\!]); succ & \\ & = & n; [\![u]\!]; succ & \text{by def. of } succ \text{ in } \mathcal{C}_! \\ & = & lin([\![u]\!]); succ & \\ & = & [\![u^o]\!]; succ & \text{by ind. hyp.} \\ & = & [\![\mathtt{succ}(u^o)]\!] & \\ & = & [\![\mathtt{succ}(u)^o]\!] & \end{array}$$

- In the case

$$\frac{\overline{x} : , \ \vdash u : N}{\overline{x} : , \ \vdash \mathtt{pred}(u) : N}$$

the following calculation suffices:

$$\begin{array}{rcll} lin([\![\mathtt{pred}(u)]\!]) & = & n; \gamma([\![u]\!]); pred & \\ & = & n; [\![u]\!]; pred & \text{by def. of } pred \text{ in } \mathcal{C}_! \\ & = & lin([\![u]\!]); pred & \\ & = & [\![u^o]\!]; pred & \text{by ind. hyp.} \\ & = & [\![\mathtt{pred}(u^o)]\!] & \\ & = & [\![\mathtt{pred}(u)^o]\!] & \end{array}$$

- In the case

$$\frac{\overline{x} : , \vdash u : N \quad \overline{x} : , \vdash v : A \quad \overline{x} : , \vdash w : A}{\overline{x} : , \vdash \texttt{if}\ u\ \texttt{then}\ v\ \texttt{else}\ w : A}$$

the following calculation suffices:

$$\begin{array}{lll}
 & lin([\![\texttt{if}\ u\ \texttt{then}\ v\ \texttt{else}\ w]\!]) & \\
= & n;\gamma(\langle [\![u]\!], \langle [\![v]\!], [\![w]\!]\rangle\rangle);cond & \\
= & \gamma(\langle (n;[\![u]\!]), \langle (n;[\![v]\!]), (n;[\![w]\!])\rangle\rangle);n^{-1};(\varepsilon\otimes\varepsilon);cond & \text{by def. of op. in } \mathcal{C}_! \\
= & \gamma(\langle lin([\![u]\!]), \langle lin([\![v]\!]), lin([\![w]\!])\rangle\rangle);n^{-1};(\varepsilon\otimes\varepsilon);cond & \\
= & \gamma(\langle [\![u^o]\!], \langle [\![v^o]\!], [\![w^o]\!]\rangle\rangle);n^{-1};(\varepsilon\otimes\varepsilon);cond & \text{by ind. hyp.} \\
= & \langle \gamma([\![u^o]\!]), \gamma(\langle [\![v^o]\!], [\![w^o]\!]\rangle)\rangle;(\varepsilon\otimes\varepsilon);cond & \text{Note 1.} \\
= & D;([\![u^o]\!]\otimes\langle [\![v^o]\!], [\![w^o]\!]\rangle);cond & \text{Note 2.} \\
= & [\![\texttt{copy}\ \overline{x}\ \texttt{as}\ \overline{x'}, \overline{x''}\ \texttt{in}\ (\texttt{if}\ u^o\ \texttt{then}\ v^o\ \texttt{else}\ w^o)]\!] & \\
= & [\![(\texttt{if}\ u\ \texttt{then}\ v\ \texttt{else}\ w)^o]\!] &
\end{array}$$

Note 1. We obtain a map $\langle f, g\rangle$ in $\mathcal{C}_!$ by composing the map $\gamma(\langle \gamma^{-1}(f), \gamma^{-1}(g)\rangle)$ with n^{-1} according to Section 2.3.
Note 2. By definition of $\langle \Leftrightarrow, + \rangle$ in $\mathcal{C}_!$ according to the discussion in Section 2.3; recall that D is the diagonal map.

- In the case

$$\frac{}{\overline{x} : , \vdash \Omega : A}$$

the following calculation suffices:

$$\begin{array}{lll}
lin([\![\Omega]\!]) & = & n;\gamma(\langle\rangle);\bot \\
 & = & n;\gamma(\langle\rangle);\cong;\bot \qquad \text{by def. of } \bot \ \texttt{in}\ \mathcal{C}_! \\
 & = & (e\otimes\ldots\otimes e);\cong;\bot \qquad \text{Note 1.} \\
 & = & [\![\texttt{discard}\ \overline{x}\ \texttt{in}\ \Omega]\!] \\
 & = & [\![\Omega^o]\!]
\end{array}$$

Note 1. Because I is a terminal object in $\mathcal{C}_!$ cf. Section 2.3.

- In the case

$$\frac{}{\overline{x} : , \vdash \texttt{Y} : (A \Rightarrow A) \Rightarrow A}$$

the following calculation suffices:

$$\begin{array}{lll}
lin([\![\texttt{Y}]\!]) & = & n;\gamma(\langle\rangle);H^\sharp \\
 & = & n;\gamma(\langle\rangle);n^{-1};H^\sharp \qquad \text{by def. of } (\Leftrightarrow)^\sharp \ \texttt{in}\ \mathcal{C}_! \\
 & = & (e\otimes\ldots\otimes e);\cong;K^\sharp \qquad \text{Note 1.} \\
 & = & [\![\texttt{discard}\ \overline{x}\ \texttt{in}\ \texttt{Y}]\!] \\
 & = & [\![\texttt{Y}^o]\!]
\end{array}$$

Recall that the maps H_A and K_{A° are defined in the proofs of Proposition 6.3.9 and Proposition 8.3.12, respectively.
Note 1. Because I is a terminal object in $\mathcal{C}_!$ cf. Section 2.3, and moreover, $H_A = K_{A^\circ}$ as can be shown by some equational manipulation.

□

10.3 Relating Operational Semantics

In this section we will prove a purely syntactic result saying that evaluation of programs of numerals type is preserved and reflected by the extended Girard Translation. The proof exploits concrete instances of adequacy results for PCF and LPCF together with a concrete instance of the soundness result of the preceding section. In the considerations below we have used the eager operational semantics, but this choice is not essential.

Theorem 10.3.1 If t is a PCF-program of ground type N, then $t \Downarrow c$ iff $t^o \Downarrow c$.

Proof: In what follows we will interpret LPCF in the concrete linear category $\mathbf{cpo}_{st}$; this is a model for LPCF in the sense of Definition 9.7.1 such that the linear fixpoint operator is rationally open with respect to N, as is made clear in Section 9.7 and Section 8.3. Moreover, we will interpret PCF in the induced Kleisli category; this is a model for PCF in the sense of Definition 7.3.1 such that the fixpoint operator is rationally open with respect to N according to Section 8.1, Section 8.2 and Section 8.3. Note that the Kleisli category is isomorphic to the category **cpo**. We then have $t \Downarrow \mathtt{succ}^n(\mathtt{zero})$ iff $[\![t]\!] = \tilde{n}$ cf. Corollary 7.4.1. But $[\![t]\!] = \tilde{n}$ is equivalent to $[\![t^o]\!] = \tilde{n}$ according to Theorem 10.2.2 and we have $[\![t^o]\!] = \tilde{n}$ iff $t^o \Downarrow \mathtt{succ}^n(\mathtt{zero})$ cf. Corollary 9.9.1. □

Note that in Theorem 10.3.1 we use the observation that a PCF-value of ground type N can be considered as a LPCF-value of ground type N, and vice versa. The point in Theorem 10.3.1 is that the type N of PCF is observable which is also the case with the type $N^o = N$ of LPCF. Strictly speaking we have an analogous result for PCF-programs of type 0 because both of the types 0 and $0^o = 0$ are observable. But this holds trivially as there are no values of any of the types.

The image of PCF under the extended Girard Translation can be considered as a version of PCF living inside LPCF. Semantically, this is analogous to the construction of a categorical model for PCF from a categorical model for LPCF (the Kleisli category). Theorem 10.3.1 shows that evaluation in the PCF-fragment of LPCF actually corresponds to evaluation in PCF. The evaluation strategy for PCF is lazy but the theorem works whatever evaluation strategy for LPCF is chosen; this suggests that the evaluation strategy for the PCF-fragment of LPCF has an inherently lazy character. Semantically, this is corroborated by the observation of Section 6.1 saying that cartesian closure does not go well together with strictness.

The paper [MOTW95] is concerned with results similar to Theorem 10.3.1. The setting of this paper is, however, different from ours: The presentation of the linear λ-calculus used is different, and moreover, the paper is concerned with reduction rules in general rather than operational semantics. Given an essentially lazy reduction strategy for the linear λ-calculus it is shown how various translations from the λ-calculus into the linear λ-calculus correspond to various reduction strategies for the λ-calculus: Lazy reduction is preserved and reflected by the Girard Translation (which maps $A \Rightarrow B$ to $!A \multimap B$) and eager reduction is preserved and reflected by a translation based on mapping $A \Rightarrow B$ to $!(A \multimap B)$. The approach taken by the authors of the mentioned paper seems to be orthogonal to ours as in this thesis we are interested in various operational semantics for the linear λ-calculus (in fact for LPCF) whereas we exclusively consider a lazy operational semantics for the λ-calculus (in fact for PCF).

Chapter 11

Further Work

In this final chapter we outline some possibilities for extension of our work.

In certain respects the generalised linear λ-calculus is similar to the variants of Girard's Logic of Unity, [Gir93] considered in [Wad93] and [Plo93]. This connection should be explored further. An operational theory for LPCF should be developed along the lines of [Pit95, Pit96]. We conjecture that the appropriate contextual preorder on terms is based on the intuitionistic variables of the generalised linear λ-calculus. This is justified by the results of Section 8.4 saying that ! enriches with respect to the intuitionistic observational preorders whereas this is not necessarily the case with respect to the linear observational preorders.

A clear-cut problem is to carry out what we have done in this thesis but with recursive types (from which fixpoint constants will follow). Another possible extension is to consider Full Intuitionistic Linear Logic, [BdP96, HdP93], which is an intuitionistic system where the "par" construct known from Classical Linear Logic is included by allowing sequents to have more than one conclusion.

In this thesis we have considered the categories $\mathbf{cpo}_{st}$ and $\mathbf{dI}_{lin}$ as our primary examples of concrete models for LPCF. The induced Kleisli categories, that is, the categories $\mathbf{cpo}$ and $\mathbf{dI}$, respectively, have been considered as primary examples of concrete models for PCF. Other concrete models ought to be taken into account with the aim of corroborating the minimality of our categorical axioms as well as the point of view that partiality is the fundamental notion from which order-structure is to be derived. In particular, the game models of [AJM96, HO96, Nic94] seem interesting because the order-structure here is derived rather that given as part of the model. In [Abr96] rational openness is actually recognised as the essential property of the game model given in [AJM96] when it comes to giving an axiomatic account of adequacy. In [Cur93b] a game model is shown to be equivalent to the category of sequential data structures and symmetric algorithms, suggesting that the concrete data structures of [Cur93a] should be amenable to the same treatment as game semantics. In another vein, the bistructure model of [PW94] and the category of hypercoherences and strongly stable functions of [Ehr93] ought to be considered, along with the traditional coherence space model for linear logic of [Gir87]. In all these categories rational openness is indeed satisfied because maps are some kind of continuous functions. In the concrete categories given in this thesis the observational preorders coincide with the natural orders on hom-sets. This is, however, not the case with certain game models. An interesting line of research would be the study of circumstances under which this correspondence holds.

The logic implicitly used in the adequacy proofs is classical rather than intuitionistic. For example, we frequently split up into cases depending upon whether or not certain maps are equal. It could be interesting to try casting the adequacy result in an intuitionistic fashion[1]. This is relevant to the area of *Synthetic Domain Theory* where the goal is to obtain constructs similar to the usual ones of domain theory, but living in an intuitionistic universe.

Originally, the notion of fixpoints in a cartesian closed category was introduced in [Law69] to

[1] This problem was mentioned by Gordon Plotkin while the author visited Edinburgh in March '96.

turn the famous arguments of Cantor, Russell, Gödel and Tarski into special cases of one single theorem. It could be interesting to see whether the considerations on fixpoints of this thesis, where a notion of undefinedness is taken into account, has something to say from the point of view of logic. Certainly, undefinedness in an appropriate sense does arise in a logical context; for example, the proof of Russell's Paradox using unrestricted comprehension does not have a normal form, which entails that the normalisation process does not terminate.

Bibliography

[Abr90] S. Abramsky. Computational interpretations of linear logic. Technical Report 90/20, Department of Computing, Imperial College, 1990.

[Abr93] S. Abramsky. Computational interpretations of linear logic. *Theoretical Computer Science*, 111, 1993.

[Abr96] S. Abramsky. Axioms for full abstraction and full completeness. Manuscript, 1996.

[AJM96] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. Submitted for publication, 1996.

[BBdPH92a] N. Benton, G. Bierman, V. de Paiva, and M. Hyland. Linear λ-calculus and categorical models revisited. In *Proceedings of CSL '92, LNCS*, volume 702. Springer-Verlag, 1992.

[BBdPH92b] N. Benton, G. Bierman, V. de Paiva, and M. Hyland. Term assignment for intuitionistic linear logic. Technical Report 262, Computer Laboratory, University of Cambridge, 1992.

[BBdPH93] N. Benton, G. Bierman, V. de Paiva, and M. Hyland. A term calculus for intuitionistic linear logic. In *Proceedings of TLCA '93, LNCS*, volume 664. Springer-Verlag, 1993.

[BCL86] G. Berry, P.-L. Curien, and J.-J. Levy. Full abstraction for sequential languages: the state of the art. In *Algebraic Semantics*. Cambridge University Press, 1986.

[BdP96] T. Braüner and V. de Paiva. Cut-elimination for full intuitionistic linear logic. Technical Report 395, Computer Laboratory, University of Cambridge, 1996. 27 pages. Also available as Technical Report RS-96-10, BRICS, Department of Computer Science, University of Aarhus.

[Ber78] G. Berry. Stable models of typed lambda-calculi. In *Proceedings of ICALP '78, LNCS*, volume 62. Springer-Verlag, 1978.

[Bie94] G. Bierman. *On Intuitionistic Linear Logic*. PhD thesis, Computer Laboratory, University of Cambridge, 1994.

[Bra94a] T. Braüner. A model of intuitionistic affine logic from stable domain theory. In *Proceedings of ICALP '94, LNCS*, volume 820. Springer-Verlag, 1994. 12 pages.

[Bra94b] T. Braüner. A model of intuitionistic affine logic from stable domain theory (revised and expanded version). Technical Report RS-94-27, BRICS, Department of Computer Science, University of Aarhus, 1994. Revised and expanded version of Technical Report DAIMI IR-118. Full version of paper in Proceedings of ICALP '94, LNCS 820, 1994.

[Bra95a] T. Braüner. The Girard translation extended with recursion. In *Proceedings of CSL '94, LNCS*, volume 933. Springer-Verlag, 1995. 15 pages.

[Bra95b] T. Braüner. The Girard translation extended with recursion. Technical Report RS-95-13, BRICS, Department of Computer Science, University of Aarhus, 1995. Full version of paper in Proceedings of CSL '94, LNCS 933, 1995.

[Bra97a] T. Braüner. A general adequacy result for a linear functional language. *Theoretical Computer Science*, 1997. 32 pages. To appear.

[Bra97b] T. Braüner. A simple adequate categorical model for PCF. In *Proceedings of TLCA '97, LNCS*. Springer-Verlag, 1997. To Appear.

[BW90] M. Barr and C. Wells. *Category Theory for Computing Science*. Prentice Hall, 1990.

[BW96] N. Benton and P. Wadler. Linear logic, monads, and the lambda calculus. In *11th LICS Conference*. IEEE, 1996.

[Cur93a] P.-L. Curien. *Categorical Combinaters, Sequential Algorithms and Functional Programming*. Birkhäuser, 1993.

[Cur93b] P.-L. Curien. On the symmetry of sequentiality. In *Proceedings of MFPS '93, LNCS*, volume 802. Springer-Verlag, 1993.

[Ehr93] T. Ehrhard. Hypercoherences: A strongly stable model of linear logic. *Mathematical Structures in Computer Science*, 3, 1993.

[EK66] S. Eilenberg and G. M. Kelly. Closed categories. In *Proceedings of the Conference on Categorical Algebra, La Jolla, 1965*. Springer-Verlag, 1966.

[Fio94a] M. P. Fiore. *Axiomatic Domain Theory in Categories of Partial Maps*. PhD thesis, University of Edinburgh, 1994.

[Fio94b] M. P. Fiore. First steps on the representation of domains. Manuscript, 1994.

[FP94] M. P. Fiore and G. D. Plotkin. An axiomatisation of computationally adequate domain theoretic models of FPC. In *9th LICS Conference*. IEEE, 1994.

[Fre92] G. Frege. Über Sinn und Bedeutung. *Zeitschrift für Philosophie und Philosophische Kritik*, 100, 1892.

[Gen34] G. Gentzen. Untersuchungen über das logische Schliessen. *Mathematische Zeitschrift*, 39, 1934.

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50, 1987.

[Gir89] J.-Y. Girard. Towards a geometry of interaction. In *Contemporary Mathematics, Categories in Computer Science and Logic*, volume 92. American Mathematical Society, 1989.

[Gir93] J.-Y. Girard. On the unity of logic. *Annals of Pure and Applied Logic*, 59, 1993.

[Gir94] J.-Y. Girard. Light linear logic. Manuscript, 1994.

[GLT89] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge University Press, 1989.

[Göd58] K. Gödel. Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. *Dialectica*, 12, 1958.

[Gri82] V. N. Grishin. Predicate and set-theoretic calculi based on logics without contractions. *Math. USSR Izvestiya*, 18, 1982.

[Gun92] C. A. Gunter. *Semantics of Programming Languages: Structures and Techniques*. The MIT Press, 1992.

[HdP93] M. Hyland and V. de Paiva. Full intuitionistic linear logic (extended abstract). *Annals of Pure and Applied Logic*, 64, 1993.

[HO96] J. M. E. Hyland and C.-H. L. Ong. On full abstraction for PCF. Submitted for publication, 1996.

[How80] W. A. Howard. The formulae-as-type notion of construction. In *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, 1980.

[HP90] H. Huwig and A. Poigne. A note on inconsistencies caused by fixpoints in a cartesian closed category. *Theoretical Computer Science*, 73, 1990.

[Jac94] B. Jacobs. Semantics of weakening and contraction. *Annals of Pure and Applied Logic*, 69, 1994.

[Lam96] F. Lamarche. Dialectics: a model of linear logic and PCF. *Mathematical Structures in Computer Science*, 1996. To Appear.

[Las96] S. B. Lassen. Action semantics reasoning about functional programs. *Mathematical Structures in Computer Science*, 1996. To Appear.

[Las97] S. B. Lassen. Relational reasoning about contexts. In Andrew D. Gordon and Andrew M. Pitts, editors, *Higher Order Operational Techniques in Semantics*. Cambridge University Press, 1997. (To appear).

[Law69] F. W. Lawvere. Diagonal arguments and cartesian closed categories. In *Category Theory, Homology Theory and their Applications II, LNM*, volume 92. Springer-Verlag, 1969.

[Mac71] S. Mac Lane. *Categories for the Working Matematician*. Springer-Verlag, 1971.

[Mac91] I. Mackie. *Lilac : A Functional Programming Language Based on Linear Logic*. M.Sc. thesis, Imperial College, 1991.

[ML84] P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.

[Mog86] E. Moggi. Categories of partial morphisms and the partial lambda-calculus. In *Proceedings Workshop on Category Theory and Computer Programming, Guildford 1985, LNCS*, volume 240. Springer-Verlag, 1986.

[Mog89] E. Moggi. Computational lambda-calculus and monads. In *4th LICS Conference*. IEEE, 1989.

[MOTW95] J. Maraist, M. Odersky, D. N. Turner, and P. Wadler. Call-by-name, call-by-value, call-by-need and the linear lambda calculus. In *Proceedings of MFPS '95, Electronic Notes in Theoretical Computer Science*, volume 1, 1995.

[Nic94] H. Nickau. Hereditarily sequential functionals. In *Proceedings of LFCS '94, LNCS*, volume 813. Springer-Verlag, 1994.

[Pit95] A. M. Pitts. Operationally-based theories of program equivalence. Notes to accompany lectures given at the Summer School *Semantics and Logics of Computation*, Isaac Newton Institute for Mathematical Sciences, University of Cambridge, 1995.

[Pit96] A. M. Pitts. A note on logical relations between syntax and semantics. *Journal of the Interest Group in Pure and Applied Logics*, 1996. Submitted.

[Plo73] G. D. Plotkin. Lambda-definability and logical relations. Memorandum SAI-RM-4, University of Edinburgh, 1973.

[Plo77] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5, 1977.

[Plo93] G. D. Plotkin. Type theory and recursion (extended abstract). In *8th LICS Conference*. IEEE, 1993.

[Poi92] A. Poigne. Basic category theory. In *Handbook of Logic in Computer Science*. Oxford University Press, 1992.

[Pra65] D. Pravitz. *Natural Deduction. A Proof-Theoretical Study.* Almqvist and Wiksell, 1965.

[PW94] G. Plotkin and G. Winskel. Bistructures, bidomains and linear logic. In *Proceedings of ICALP '94, LNCS*, volume 820. Springer-Verlag, 1994.

[Ros86] G. Rosolini. *Continuity and Effectiveness in Topoi.* PhD thesis, University of Oxford, 1986.

[Sco69] D. S. Scott. A type theoretical alternative to CUCH, ISWIM, OWHY. Manuscript, 1969.

[Sco93] D. S. Scott. A type theoretical alternative to CUCH, ISWIM, OWHY. In *Böhm Festscrift, Theoretical Computer Science*, volume 121. Elsevier, 1993.

[See89] R. A. G. Seely. Linear logic, $*$-autonomous categories, and cofree coalgebras. In *Contemporary Mathematics, Categories in Computer Science and Logic*, volume 92. American Mathematical Society, 1989.

[Shi94] M. Shirahata. *Linear Set Theory.* PhD thesis, Stanford University, 1994.

[Sto77] J. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory.* The MIT Press, 1977.

[Wad91] P. Wadler. There's no substitute for linear logic. Manuscript, 1991.

[Wad93] P. Wadler. A taste of linear logic. In *Proceedings of MFCS '93, LNCS*, volume 711. Springer-Verlag, 1993.

[Win87] G. Winskel. Event structures. In *LNCS*, volume 255. Springer-Verlag, 1987.

[Win93] G. Winskel. *The Formal Semantics of Programming Languages.* The MIT Press, 1993.

[Zha93] G.-Q. Zhang. Some monoidal closed categories of stable domains and event structures. *Mathematical Structures in Computer Science*, 3, 1993.

Index

Recent Publications in the BRICS Dissertation Series

DS-96-4 **Torben Braüner.** ***An Axiomatic Approach to Adequacy*****. November 1996. Ph.D. thesis. 168 pp.**

DS-96-3 **Lars Arge.** ***Efficient External-Memory Data Structures and Applications*****. August 1996. Ph.D. thesis. xii+169 pp.**

DS-96-2 **Allan Cheng.** ***Reasoning About Concurrent Computational Systems*****. August 1996. Ph.D. thesis. xiv+229 pp.**

DS-96-1 **Urban Engberg.** ***Reasoning in the Temporal Logic of Actions — The design and implementation of an interactive computer system*****. August 1996. Ph.D. thesis. xvi+222 pp.**