Hereditarily Sequential Functionals: A Game-Theoretic Approach to Sequentiality

DISSERTATION

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

vorgelegt von

Dipl.–Math. Hanno Nickau

aus Hamburg

angenommen vom Fachbereich 6

der Universität–Gesamthochschule–Siegen

Siegen 1996

Dekan: Prof. Dr. Franz-Jürgen Delvos

1. Gutachter: Prof. Dr. Dieter Spreen
2. Gutachter: Prof. Dr. Achim Jung

Tag der mündlichen Prüfung: 21. Juni 1996

Abstract

The aim of this thesis is to give a new understanding of sequential computations in higher types. We present a new computation model for higher types based on a game describing the interaction between a functional and its arguments. The functionals which may be described in this way are called hereditarily sequential. We show that this computation model captures exactly the notion of computability in higher types introduced by Kleene in his pioneering work starting 1959. We study the order structure of the hereditarily sequential functionals and discuss the occurring difficulties. These functionals form a fully abstract model for PCF and we discuss which problems remain still open for a satisfactory solution to the full abstraction problem of PCF.

Zusammenfassung

Ziel dieser Arbeit ist es, eine neue Beschreibung sequentieller Berechnungen in höheren Typen zu geben. Wir stellen dazu ein neues Berechnungsmodell für höhere Typen vor, in dem die Interaktion zwischen einem Funktional und seinen Argumenten durch ein Spiel beschrieben wird. Die so beschreibbaren Funktionale werden vererbt sequentiell genannt. Wir zeigen, daß mit diesem Berechnungsmodell genau der Begriff von Berechenbarkeit in höheren Typen charakterisiert wird, der von Kleene in seinen Arbeiten ab 1959 untersucht wurde. Wir untersuchen die Ordnungsstruktur der vererbt sequentiellen Funktionale und diskutieren die auftretenden Probleme. Diese Funktionale bilden ein vollabstraktes Modell für PCF, und wir diskutieren die Probleme, die für eine zufriedenstellende Lösung des Vollabstraktheitsproblems für PCF noch bestehen bleiben.

Acknowledgements

I would like to thank Dieter Spreen for working with me and supervising this thesis. During the years we had many interesting and illuminating discussions. Without his support this work would not have been possible.

Special thanks go to Achim Jung for having kindly accepted to referee this thesis. His comments and suggestions helped me in improving the first version of the thesis.

I am also grateful to Samson Abramsky, Gerard Berry, Antonio Bucciarelli, Pierre-Louis Curien, Claudio Dupre, Thomas Ehrhard, Shai Geva, Robin Gandy, Martin Hyland, Rhada Jagadeesan, Neil Jones, Roger Hindley, Bruce Kapron, Pasquale Malacaria, Wolfgang Merzenich, Peter O'Hearn, Luke Ong, Giovanni Pani, Jon Riecke, Vladimir Sazonov, Helmut Schwichtenberg, Dana Scott, Kurt Sieber, Allen Stoughton, and Glynn Winskel for taking an interest in this work, for encouraging discussions and precious support on various occasions through the years.

Contents

Chapter 1

Introduction

In ordinary, first-order, computability theory all reasonable computation models are equivalent leading to Church's Thesis that this is the right formalization of computability, i.e. it coincides with the intuitive notion of computability. In contrast, for higher types, there has been less consensus about which computation model captures computable functionals, in particular, if we are interested in higher type complexity.

The various approaches to higher type computability and complexity developed so far differ in the decision which properties of ordinary computability theory should be kept for higher types. The main question is how to deal with higher type arguments.

The study of computation in higher types was initiated by Kleene in (1959, 1963). In this first part he worked over the hereditarily total functionals leading to some anomalies compared with ordinary computability theory. Later, in the second half of his series (1978, 1980, 1982, 1985), Kleene considered partial functionals too. In this part his aim was:

> *To generate a class of function[al]s which shall coincide with all the partial function[al]s which are "computable" or "effectively decidable", so that Church's 1936 thesis will apply with the higher types included* (Kleene 1978).

He used two approaches to describe this class of functionals. The first is definitorial by schemata (S1) – (S8), (S11) and the other is operational with the help of a certain class of "licensed" oracles. There are two main problems in this approach of oracles. First the presentation is mathematically unclear and it is difficult to understand how it extends to arbitrary types. Kleene actually described the oracles only for pure types up to level 4. The second problem concerns the criterion for oracles to be "licensed". It is used to force the oracles to describe (1) monotone functionals and (2) to behave extensional in the sense that they behave the same when applied to different oracles describing the same functional.

In this thesis we present a new computation model for higher types to overcome this difficulties in Kleene's approach of oracles. It is guided by the following requirements:

(1) Continuity. Terminating computations should be finite objects. In particular the amount of information of the inputs (oracles) used in a terminating computation has to be finite. So the considered functionals should be partial continuous.

(2) Extensionality. The result of a computation should only depend on the extent of its input not on a particular intensional description of it.

(3) Sequentiality. Algorithms should be sequential in the sense that computations should proceed stepwise with one focus at a time.

The first point is not exactly what Kleene did since he asked only for monotonicity. It is not difficult to extend our approach to infinite terminating computations, but in view of real computations we think it is the right restriction.

The model is based on a game called *Game of Higher Types*. It describes the interaction between the functional (program) and the arguments (oracle, input) as a play between two players. Each move is an enabled question (enabled by a certain opposing question), or an answer to the last open question. To prevent the use of intensional aspects the choice of the next move must not depend on all moves already played, but only on an "extensional view" of them. Then a sequential strategy is a partial function from the set of these views to the set of possible moves. The sequential strategies form a computation model that is not extensional w.r.t. application of sequential strategies but is extensional in the sense of (2).

The main advantage of this approach over Kleene's is that it gives a description of the interaction for all types, and there is no need for a "license" criterion. Both monotonicity and extensionality have been built in the rules of the game with the help of the "views".

To each sequential strategy p we relate a functional F_p describing the extent of p. We call the functionals F_p hereditarily sequential and the set of these functionals HSF. The correspondence between sequential strategies and hereditarily sequential functionals is the same as that of oracles and unimonotone functionals in Kleene's approach in (Kleene 1978).

We give two definitions for elements of HSF to be computable. We have the recursive elements of HSF being extents of recursive sequential strategies. Secondly we consider the class of Kleene-recursive functionals, i.e. definable by Kleene's schemata (S1) – (S8), (S11) (Kleene 1959) over HSF. We show that both notions coincide. So we have found a computation model for higher types fulfilling the requirements (1) – (3) and with a robust notion of computable functional.

Another approach to computability in higher types is due to Scott and Ershov (Scott 1993, Ershov 1972a, 1973, 1975, 1977b). The idea behind this approach is to deal with finite approximations instead of the arguments itself. Then an element is computable if the set of its finite approximations is recursively enumerable. This approach turns out to lead to parallel aspects as explained in the following.

Scott introduced the formal system LCF a logical framework for computable functionals. Later on Plotkin presented LCF explicitly as a programming language called PCF and studied the relation between its operational and denotational semantics (Plotkin 1977). Plotkin showed that the standard model of PCF, i.e. the category of Scott-domains and continuous functions, is not fully abstract. Plotkin and independently Sazonov (1976a) have shown that the continuous model is fully abstract for PCF extended by a parallel conditional. Moreover the Scott-computable functionals are those that are definable by PCF extended by the parallel conditional and a continuous approximation to the existential quantifier. This existential quantifier is necessary to get the strength of unbounded dove-tailing in ordinary recursion theory.

Since then there has been made much effort to find a denotational semantics which coincides with the operational one. Milner (1977) has shown that there is a unique order extensional, continuous, fully abstract model for PCF. In view of this fact a possible formulation of the full abstraction problem for PCF could be stated as:

> *Find a description of the order extensional, continuous, fully abstract model for* PCF *via some abstract denotational model.*

The main step towards a solution for the full abstraction problem for PCF is to understand sequentiality since the evaluation of PCF is sequential in a definite sense (Berry 1976).

We show, that the class of hereditarily sequential functionals form a fully abstract model for PCF. We do not know if our model is isomorphic to Milner's model. The open question is whether the hereditarily sequential functionals are complete partial orders at any type. Anyway we argue that the notion of sequentiality is captured in an intuitive sense.

Plan of the work

In Chapter 2 we start with a review of Kleene's schemata, a description of Kleene's oracles for unimonotone functionals, and an introduction to the full abstraction problem of PCF.

In Chapter 3 we introduce the game and sequential strategies. In Chapter 4 we study the extensional behaviour of the strategies leading to the hereditarily sequential functionals (HSF). We study the order structure of HSF and show that it gives a fully abstract model for PCF.

Chapter 5 is devoted to computability. We introduce two notions of recursiveness and show that both notions describe the same class of computable functionals.

In Chapter 6 we end with a discussion of some open problems connected with the full abstraction problem of PCF.

Related work

Independently of the author Hyland and Ong (1994) have introduced the same kind of game where the strategies are called innocent, since the strategies use neither all the history (history sensitive strategies) nor only the last move (history free strategies) but depend on the view actually in force. The presentation there is more categorical and is used for giving directly an (intensional) fully abstract model of PCF. For that reason this approach is called game semantics.

Another approach in the same direction is due to Abramsky, Jagadeesan, and Malacaria (1994) who use a certain class of history free strategies to interpret a fragment of linear logic. They arrive at the same result for PCF by the usual factorization of intuitionistic implication by linear implication and exponential !, i.e. via $A \to B = !A \multimap B$.

A first version of the work presented in this thesis appeared as (Nickau 1994).

Chapter 2

Background

2.1 Types and type structures

We introduce the type symbols for simple types over a set of groundtype symbols and a notation system for the occurrences of the groundtype symbols in a type symbol.

2.1.1 Definition. Let Γ be a set of groundtype symbols. The set of *type symbols* TS and their *level* is inductively defined by

- $\Gamma \subseteq \mathsf{TS}$. These are the only type symbols of level 0.
- If $\tau_1, \ldots, \tau_k \in \mathsf{TS}$ and $\gamma \in \Gamma$ then $(\tau_1, \ldots, \tau_k \to \gamma) \in \mathsf{TS}$. The level of $(\tau_1, \ldots, \tau_k \to \gamma)$ is one greater then the maximum of the levels of $\tau_1, \ldots, \tau_k$.

$\Diamond$

2.1.2 Remark. The common definition of simple types TS' is given by

$$\Gamma \subseteq \mathsf{TS}' \qquad \text{and} \qquad \tau, \sigma \in \mathsf{TS}' \implies (\tau \Rightarrow \sigma) \in \mathsf{TS}'.$$

It is well known that each type $\tau \in \mathsf{TS}'$ has a unique decomposition of the form

$$\tau = (\tau_1 \Rightarrow (\tau_2 \Rightarrow (\ldots(\tau_k \Rightarrow \gamma)\ldots)))$$

where each τ_i has the same form. Our definition of simple types takes this view as basic and we write $\tau = (\tau_1, \ldots, \tau_k \to \gamma)$. $\Diamond$

We use γ, γ', ... as metavariables for groundtype symbols. The groundtype symbols $\gamma \in \Gamma$ serve as names for fixed sets of values V_γ. We use v, v', ... as metavariables for values. We denote the set of all values by $V = \bigcup\{V_\gamma \mid \gamma \in \Gamma\}$. In the examples we use $\mathtt{N}$,

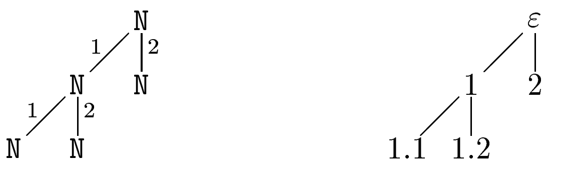
$\mathtt{B}$ and $\mathtt{O}$ as fixed groundtype symbols with associated sets of values $V_{\mathtt{N}} = \mathbb{N} = \{0, 1, 2, \ldots\}$, the set of natural numbers, $V_{\mathtt{B}} = \mathbb{B} = \{\mathtt{t}, \mathtt{f}\}$ the set of boolean values, and $V_{\mathtt{O}} = \mathbb{O} = \{\mathtt{T}\}$ the singleton set.

2.1.3 Definition. Each *occurrence* of $\gamma \in \Gamma$ in a type symbol τ can be identified by a finite sequence of integers in the following way: The only occurrence of γ in $\tau = \gamma$ and the occurrence of γ on the right hand side of the arrow in $\tau = (\tau_1, \ldots, \tau_k \to \gamma)$ is identified by the empty sequence ε. If γ is an occurrence in τ_i with identification a, then the corresponding occurrence in $\tau = \tau_1, \ldots, \tau_k \to \gamma$ is identified by $i.a$.

By $|a|$ we denote the length of the sequence a. Let a be an occurrence in τ, then we denote by τ_a the type corresponding to the subtree with root a of the tree related to τ.

$\Diamond$

2.1.4 Example. The type $(\mathtt{N}, \mathtt{N} \to \mathtt{N}), \mathtt{N} \to \mathtt{N}$ considered as a tree and as a tree of occurrences:



$\Diamond$

2.1.5 Definition. A type τ is called *unary* or *pure* iff all occurrences in τ are identified by sequences of ones. $\Diamond$

2.1.6 Remark. If there is only one groundtype symbol γ in Γ then the pure types are uniquely determined by their level. In this case we use the abbreviation $\underline{n}$ for the pure type of level n, i.e. $\underline{0} = \gamma$ and $\underline{n{+}1} = (\underline{n} \to \gamma)$. $\Diamond$

2.1.7 Definition. A *type structure* over TS is a collection of sets $\{D^\tau \mid \tau \in \mathsf{TS}\}$ such that

- $D^\gamma \supseteq V_\gamma$ for $\gamma \in \Gamma$, and
- if $\tau = (\tau_1, \ldots, \tau_k \to \gamma)$ then D^τ is a set of maps from $D^{\tau_1} \times \cdots \times D^{\tau_k}$ into D^γ.

$\Diamond$

2.1.8 Example. Examples of type structures are

- the hereditarily total functionals $(\mathsf{HT}^\tau)_{\tau \in \mathsf{TS}}$ given by $\mathsf{HT}^\gamma = V_\gamma$and $\mathsf{HT}^{(\tau_1, \ldots, \tau_k \to \gamma)}$ being the set of all (total) maps from $\mathsf{HT}^{\tau_1} \times \cdots \times \mathsf{HT}^{\tau_k}$ to V_γ, and
- the Scott-Ershov-hierarchy $(C^\tau)_{\tau \in \mathsf{TS}}$ of the partial continuous functionals over the flat domains $C^\gamma = (V_\gamma)_\perp$. $\Diamond$

In the sequel it will be useful to have a lambda notation for defining functions of type τ in some type structure $(D^\sigma)_\sigma$. For that purpose we assume a set of variables $\{X_i^a \mid a \text{ occurrence in } \tau,\ i \in \mathbb{N}\}$, where X_i^a ranges over D^{τ_a}. By $\overline{X}_i^a$ we denote the sequence $(X_i^{a.1}, \dots, X_i^{a.l})$ if a has l successors and the empty sequence ε if a is a leaf.

If t is a parameterized definition of an element in D^γ, then $\lambda\overline{X}^a.t = \lambda(X^{a.1}, \dots, X^{a.l}).t$ stands for the corresponding map from $D^{\tau_{a.1}} \times \cdots \times D^{\tau_{a.l}}$ to D^γ possibly with some parameters left. Depending on the type structure it will be an element of D^{τ_a} or not. If a is a leaf we set $\lambda\overline{X}^a.t = t$.

2.1.9 Example. Let $\tau = ((\mathtt{N}, \mathtt{N} \to \mathtt{N}), \mathtt{N} \to \mathtt{N})$ be the type of example 2.1.4, then we have $\tau_1 = (\mathtt{N}, \mathtt{N} \to \mathtt{N})$ and $\tau_2 = \mathtt{N}$ well corresponding to the notation $\tau = (\tau_1, \tau_2 \to \mathtt{N})$ and $\tau_{1.1} = \tau_{1.2} = \mathtt{N}$. Moreover we have $\overline{X} = (X^1, X^2)$, $\overline{X}^1 = (X^{1.1}, X^{1.2})$, and $\overline{X}^{1.1} = \overline{X}^{1.2} = \overline{X}^2 = \varepsilon$. $\Diamond$

2.2 Kleene's schemata

Kleene began his study of higher type computation in his papers (1959) over the hereditarily total functionals. Actually, Kleene only considered computation over pure types.

Kleene's approach is based on a collection of schemata for defining functionals. The schemata (S1)–(S8) and (S11) from (Kleene 1978) are given below. The schemata define functionals F under the hypothesis that F_1, F_2, …, F_l have been defined before, the variables X_a range over D^{τ_a} where $(D^\tau)_{\tau \in \mathsf{TS}}$ is the type structure under consideration. The $\lambda\overline{X}^a$-notation is explained in section 2.1.

(S1) successor $\quad F(\overline{X}) = X_1 + 1 \quad (\tau_1 = \gamma)$

(S2) constants $\quad F(\overline{X}) = c \quad (c \in \omega)$

(S3) identity $\quad F(\overline{X}) = X_1 \quad (\tau_1 = \gamma)$

(S4) composition $\quad F(\overline{X}) = F_2(F_1(\overline{X}), \overline{X})$

(S5) primitive recursion

$$F(\overline{X}) = \begin{cases} F_1(X_2, \dots, X_k) & \text{if } X_1 = 0 \\ F_2(F(X_1 - 1, X_2, \dots, X_k), \overline{X}) & \text{if } X_1 > 0 \end{cases}$$

(S6) permutation $\quad F(\overline{X}) = F_1(\pi(\overline{X})) \quad (\pi(\overline{X}) \text{ permutation of } \overline{X})$

(S7) function-application $\quad F(\overline{X}) = X_1(X_2, \dots X_{l+1}) \quad (\tau_1 = \gamma, \dots, \gamma \to \gamma)$ $(\tau_i = \gamma,\ i = 2, \dots, l{+}1)$

(S8) functional-application

$$F(\overline{X}) = X_1(\lambda\overline{X}^{1.1}F_1(\overline{X},\overline{X}^{1.1}),...,\lambda\overline{X}^{1.l}F_l(\overline{X},\overline{X}^{1.l}))$$
$$(\tau_1 = \tau_{1.1},\ldots,\tau_{1.l} \to \gamma)$$

(S11) recursion $\quad F(\overline{X}) = F_1(F,\overline{X})$

2.3 Kleene's unimonotone functionals

To explain the problems that arise in higher type computability we look first at Kleene's approach of unimonotone functionals. The primary problem of higher type computability is to explain how should a higher type functional gather information about its arguments. In this section, we consider only pure types, as did Kleene. The argument to a function is always given via an oracle even for the simplest ones, the type-0-objects. This is necessary to distinguish the strict and non-strict constant functions that arise in the context of partial continuous functionals.

Type-0-oracles

For an α^0-oracle there are two possible ways to act:

Case 0.1**:** The α^0-oracle does not answer our question. So $F_{\alpha^0} = \bot$. The corresponding decision tree is empty.

Case 0.2**:** The α^0-oracle gives the answer n. So $F_{\alpha^0} = n$. The corresponding decision tree is $\underline{?0}\,!n$.

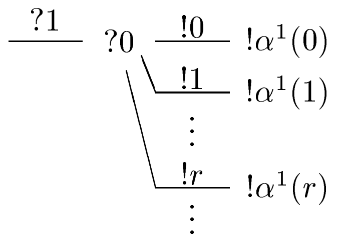
Type-1-oracles

For an α^1-oracle presented with an α^0-oracle there are three possible ways to act:

Case 1.1**:** Without interrogating the α^0-oracle the α^1-oracle does not answer our question. So $\alpha^1 = \bot_1 = \lambda x^0.\bot$. The corresponding decision tree is empty or equivalently $\underline{?1}\,\bot$.

Case 1.2**:** Without interrogating the α^0-oracle the α^1-oracle gives the answer n. So $\alpha^1 = \lambda x^0.n$. The corresponding decision tree is $\underline{?1}\,!n$.

Case 1.3**:** The α^1-oracle asks the α^0-oracle for a value. If the α^0-oracle does not answer (case 1^0) the α^1-oracle stands mute too. If the α^0-oracle gives a value, let's say r, then, depending (in general) on the r, the α^1-oracle may fail to answer, so $\alpha^1(r) = \bot$, or she may answer n, indicating that $\alpha^1(r) = n$. The corresponding decision tree is



It is obvious that each function represented by an type-1-oracle is monotone, and vice versa each monotone function from $\mathbb{N}_\perp$ to $\mathbb{N}_\perp$ can be represented by a type-1-oracle.

Type-2-oracles

Next we want to describe how a α^2-oracle operates given an α^1-oracle as input:

Case 2.1**:** Without interrogating the α^1-oracle the α^2-oracle stands mute. So $F_{\alpha^2} = \perp_2 = \lambda x^1.\perp$.

Case 2.2**:** Without interrogating the α^1-oracle the α^2-oracle gives the answer n. So $F_{\alpha^2} = \lambda x^1.n$.

Case 2.3**:** The α^2-oracle asks the α^1-oracle for a value. Depending on the behaviour of the α^1-oracle there are three subcases:

Subcase 2.3.1**:** If the α^1-oracle does not answer (case 1^1) the α^2-oracle stands mute too, so $F_{\alpha^2}(\perp_1) = \perp$.

Subcase 2.3.2**:** If the α^1-oracle, without asking for the value of its argument (a type-0-oracle), gives a value, let's say r (case 1.2), then, depending (in general) on the r, the α^2-oracle may fail to answer, so $F_{\alpha^2}(\lambda x^0.r) = \perp$, or she may answer n, indicating that $F_{\alpha^2}(\lambda x^0.r) = n$.

Subcase 2.3.3**:** If the α^1-oracle asks for its argument the α^2-oracle may stand mute or present a first value r_0 as the argument. Now α^1 will either stand mute, in which case α^2 stands mute too, or give an answer n_0. Now α^2 may stand mute, thus deciding that the information collected about α^1 (namely that $F_{\alpha^1}(r_0) = n_0$) is sufficient to rule out that $F_{\alpha^2}(F_{\alpha^1})$ be defined, or give a result m declaring that $F_{\alpha^2}(F_{\alpha^1}) = m$ or ask α^1 with another integer r_1. In general in this subcase a series of questions will be asked to α^1 by α^2 with arguments $r_0, \ldots, r_k, \ldots$ and will be answered by α^1 with numbers $n_0, \ldots, n_k, \ldots$ where $n_i = F_{\alpha^1}(r_i)$ and r_i is determined by α^2 from only the information that α^1 needs the argument and $F_{\alpha^1}(r_j) = n_j$ for $j < i$. This continues until either, for some i, α^1 does not answer to r_i, which makes $F_{\alpha^2}(F_{\alpha^1})$ undefined, or α^2 decides that the information so far collected about α^1 makes $F_{\alpha^2}(F_{\alpha^1})$ undefined, or finally that it is sufficient to give m as result, so $F_{\alpha^2}(F_{\alpha^1}) = m$. Actually Kleene allowed transfinite series as he did not require continuity.

Discussion of Kleene's oracles

The first observation is that a question will only be answered if all counter-questions have been answered. A condition sometimes called "no-dangling questions".

The second point is continuity and hence monotonicity. For example a α^2-oracle if it gives an answer m under subcase 2.3.3 on the basis that $\alpha^1(r_i) = n$ for $1 \leq i \leq j$ and no other information about α^1 then it should give the same answer for $\alpha^1 = \lambda x^0.n$ under subcase 2.3.2. In Kleene's approach this will be enforced by an additional condition for an oracle to be "licensed". But this could easily be achieved by the following variation. The α^2-oracle must not recognize if the α^1-oracle needs its argument or not.

The third point is well definedness in the following sense. The answer of an $\alpha^{(j+1)}$-oracle should depend only on the functional F_{α^j} not on the chosen oracle α^j. For types 0 and 1 the oracles representing unimonotone functionals are unique except for the undefined functional $\perp_1$ that may represented by an oracle under case 1.1 or under case 1.3. Because of the first observation there is no problem for types 0, 1 and 2. The following example shows that for higher types this could be a more difficult point. Consider the functional α^2 defined by

$$\alpha^2(\alpha^1) = \begin{cases} 0, & \text{if } \alpha^1(0) = 0 \text{ and } \alpha^1(1) = 1, \\ \perp, & \text{otherwise.} \end{cases}$$

Then an α^2 may work in two ways $r_1 = 0$ and $r_2 = 1$ or $r_1 = 1$ and $r_2 = 0$. So in the definition of type-3-oracles we have to be careful that they behave the same for these two oracles.

This last point is the most subtle one. Kleene overcame this difficulty just by an additional requirement for the oracles to be licensed. In contrast in the approach presented here this functionality or extensionality will follow from the general rules for the oracles. But it is still an involved proof for higher types.

2.4 The full abstraction problem for PCF

In (1977) Plotkin presented PCF as a programming language and studied the relationship between its operational and denotational semantics. PCF is a simply typed λ-calculus with arithmetic constants and fixed points, originally defined by Dana Scott in 1969 as the term language of the logic LCF (Logic of Computable Functions) (Scott 1993). We give here a general definition of simply typed λ-calculus with recursion built on a set Γ of groundtypes and a set K a of typed constants ($\lambda Y\Gamma K$).

2.4.1 Definition. The types of $\lambda Y\Gamma K$ are the simple types over Γ as defined in Definition 2.1.1. Terms are inductively defined by typed application and λ-abstraction starting from typed variables and constants:

- Any variable of type τ is a term of type τ.
- Any constant c of type τ is a term of type τ.
- If M is a term of type $\tau \Rightarrow \sigma$ and N is a term of type τ then $(M\,N)$ is a term of type σ.
- If M is a term of type τ and x is a variable of type σ then $\lambda x.M$ is a term of type $\tau \Rightarrow \sigma$.
- If M is term of type $\tau \Rightarrow \tau$ then $\mathsf{Y}(M)$ is term of type τ.

We write $M : \tau$ if M is a term of type τ. $\Diamond$

The language PCF has as ground types $\mathtt{N}$ and $\mathtt{B}$ the types of natural numbers and booleans, respectively. Its constants are numerals $\mathsf{k}_n : \mathtt{N}$ for each natural number n, boolean values $\mathsf{t}, \mathsf{f} : \mathtt{B}$, the successor $\mathsf{succ} : (\mathtt{N} \to \mathtt{N})$, the predecessor $\mathsf{pred} : (\mathtt{N} \to \mathtt{N})$, the test for zero $\mathsf{iszero} : (\mathtt{N} \to \mathtt{B})$ and the conditionals $\mathsf{cond}^\gamma : (\mathtt{B}, \gamma, \gamma \to \gamma)$.

The notions of free and bound variables are the usual ones. Terms without free variables are closed. Closed terms of groundtype are called programs. In order to define the operational semantics we use a formal system defining the two place predicate $M \Downarrow C$ over closed terms with the intended meaning: "M evaluates to the canonical form C", where canonical forms are either constants or abstractions:

$$\frac{}{C \Downarrow C} \qquad \frac{F \Downarrow \lambda x.M \qquad M[N/x] \Downarrow C}{F\,N \Downarrow C} \qquad \frac{M \Downarrow C \qquad C\,N \Downarrow C'}{M\,N \Downarrow C'}$$

$$\frac{M\,\mathsf{Y}(M) \Downarrow C}{\mathsf{Y}(M) \Downarrow C}$$

The operational semantics of the PCF constants are given by extending the system by the following rules:

$$\frac{M \Downarrow \mathsf{k}_n}{\mathsf{succ}\,M \Downarrow \mathsf{k}_{n+1}} \qquad \frac{M \Downarrow \mathsf{k}_{n+1}}{\mathsf{pred}\,M \Downarrow \mathsf{k}_n} \qquad \frac{M \Downarrow \mathsf{k}_0}{\mathsf{iszero}\,M \Downarrow \mathsf{t}} \qquad \frac{M \Downarrow \mathsf{k}_{n+1}}{\mathsf{iszero}\,M \Downarrow \mathsf{f}}$$

$$\frac{M \Downarrow \mathsf{t} \qquad N \Downarrow C}{\mathsf{cond}^\gamma\,M\,N\,P \Downarrow C} \qquad \frac{M \Downarrow \mathsf{f} \qquad P \Downarrow C}{\mathsf{cond}^\gamma\,M\,N\,P \Downarrow C}$$

The following properties of $\Downarrow$ are easily established:

2.4.2 Lemma. *Let M be a closed term and C, C' be closed canonical forms.*

1. *If $M \Downarrow C$ and $M : \tau$ then $C : \tau$.*
2. *If $M \Downarrow C$ and $M \Downarrow C'$ then $C = C'$.*
3. *If $M \Downarrow C$ and $M : \mathtt{B}$ then $C = \mathsf{f}$ or $C = \mathsf{t}$.*
4. *If $M \Downarrow C$ and $M : \mathtt{N}$ then $C = \mathsf{k}_n$ for some $n \in \mathbb{N}$.* $\square$

A context $C[\ ]$ is a term with a typed hole. As usual $C[M]$ denotes the term that is obtained from the context $C[\ ]$ by filling the hole with M. Note that variable capture is possible, in contrast with the usual substitution $M[N/x]$. We say that M *operationally approximates* N ($M \sqsubseteq_{op} N$) if for every context $C[\ ]$ such that both $C[M]$ and $C[N]$ are programs, and for any value v, i.e. a canonical form of groundtype, if $C[M] \Downarrow v$ then $C[N] \Downarrow v$.

2.4.3 Definition. A (*functional*) *model* of $\lambda Y\Gamma K$ is a type structure $(D^\tau)_{\tau\in\mathsf{TS}(\Gamma)}$ where each D^τ is a pointed partial order with least fixpoints together with a mapping $[\![\ ]\!]$ from terms and environments to $\bigcup_{\tau\in\mathsf{TS}} D^\tau$ that satisfies

- If $M : \tau$ then $[\![M]\!]\rho \in D^\tau$.
- For any variable $x : \tau$ $[\![x]\!]\rho = \rho(x)$.
- If $\rho(x) = \rho'(x)$ for all free variables in M then $[\![M]\!]\rho = [\![M]\!]\rho'$.
- If $M : \tau \Rightarrow \tau$ then $[\![\mathsf{Y}(M)]\!]\rho = \mathit{fix}([\![M]\!]\rho)$.
- If $M : \tau \Rightarrow \sigma$ and $N : \tau$ then $[\![M\,N]\!]\rho = [\![M]\!]\rho([\![N]\!]\rho)$.
- If $M : \sigma$, $x : \tau$ and $d \in D^\sigma$ then $[\![\lambda x.M]\!]\rho(d) = [\![M]\!]\rho[d/x]$.

The model is *order extensional* if for all $\tau \Rightarrow \sigma \in \mathsf{TS}(\Gamma)$ and for all $f, g \in D^{\tau\Rightarrow\sigma}$

$$f \sqsubseteq g \qquad \Longleftrightarrow \qquad \text{for all } d \in D^\tau\ f(d) \sqsubseteq g(d).$$

The model is *continuous* if each D^τ is a cpo and the applications are continuous. $\diamondsuit$

Any model $\mathcal{D} = ((D^\tau, [\![\ ]\!])$ of $\lambda Y\Gamma K$ induces a preorder over the terms by

$$M \sqsubseteq_{\mathcal{D}} N \qquad \Longleftrightarrow \qquad [\![M]\!]\rho \sqsubseteq [\![N]\!]\rho \text{ for all } \rho\ .$$

2.4.4 Definition. Let $\mathcal{D}$ be any model of $\lambda Y\Gamma K$.

1. $\mathcal{D}$ is *computationally adequate* if for all terms M and N

$$M \sqsubseteq_{\mathcal{D}} N \qquad \Longrightarrow \qquad M \sqsubseteq_{op} N.$$

2. $\mathcal{D}$ is *fully abstract* if for all terms M and N

$$M \sqsubseteq_{op} N \qquad \Longrightarrow \qquad M \sqsubseteq_{\mathcal{D}} N. \qquad \Diamond$$

The standard continuous model of PCF is given by the type structure of the partial continuous functionals $(C^\tau)_{\tau \in \mathsf{TS}}$ over the flat domains $C^{\mathtt{N}} = \mathbb{N}_\perp$ and $C^{\mathtt{B}} = \mathbb{B}_\perp$ with

$$\begin{array}{rcl}
[\![\mathsf{k}_n]\!] & = & n \\
[\![\mathsf{t}]\!] & = & \mathtt{t} \\
[\![\mathsf{f}]\!] & = & \mathtt{f} \\
[\![\mathsf{succ}]\!](d) & = & \begin{cases} n+1 & \text{if } d = n \\ \perp & \text{otherwise} \end{cases} \\
[\![\mathsf{pred}]\!](d) & = & \begin{cases} n & \text{if } d = n+1 \\ \perp & \text{otherwise} \end{cases} \\
[\![\mathsf{iszero}]\!](d) & = & \begin{cases} \mathtt{t} & \text{if } d = 0 \\ \mathtt{f} & \text{if } d = n+1 \\ \perp & \text{otherwise} \end{cases} \\
[\![\mathsf{cond}^\gamma]\!](d_1, d_2, d_3) & = & \begin{cases} d_2 & \text{if } d_1 = \mathtt{t} \\ d_3 & \text{if } d_1 = \mathtt{f} \\ \perp & \text{if } d_1 = \perp \end{cases}
\end{array}$$

2.4.5 Theorem. *The standard continuous model of* PCF *is computationally adequate but not fully abstract.* □

2.4.6 Theorem. *In the standard continuous model of* PCF *it holds for any program M and any constant c*

$$M \Downarrow c \qquad \Longleftrightarrow \qquad [\![M]\!] = [\![c]\!].$$

□

2.4.7 Theorem (Milner 1977). *There is a unique (up to isomorphism) order extensional, continuous, fully abstract model for* PCF*.* □

A possible formulation of the full abstraction problem for PCF is:

> *Find a description of the order extensional, continuous, fully abstract model for* PCF *via some abstract denotational model.*

Berry, Curien, and Levy's article (1985) gives a summary of approaches to the full abstraction problem before 1985. The actual state of the art can be found in (Jung, Fiore, Moggi, et al. 1996).

To end this section we consider some of the approaches to the full abstraction problem. The reason for the standard model of PCF not being fully abstract is that it contains functions not being sequential. There is a natural notion of sequentiality for first-order functions due to Vuillemin (1974). The intuition is that the arguments have to be visited in a given order: if the function is strict then the computation begins by the evaluation of v_i, the i-th argument. If v_i is undefined then the whole computation is undefined, else the computation may proceed by the evaluation of the j-th argument, j depending in general on v_i, and so on. It is easy to see that the parallel-or function is not sequential in this sense. This notion of sequentiality fully characterizes PCF-definability for first-order functions. But it is not clear how it can be extended to higher-order functionals.

Kahn and Plotkin (1978) introduced concrete data structures as representation of concrete domains and sequential functions as such an extension. However, as shown by Curien (1993) the category of concrete data structures and sequential functions is not cartesian closed.

Two main directions have been followed to overcome this difficulty. The first one consists in weakening sequentiality in a notion which can be extended to higher order to get cartesian closedness. This has been done by Berry (1978) with the definition of the stable model. The parallel-or function is not stable and all sequential first-order functions are stable, but there are stable functions which are not sequential. Bucciarelli and Ehrhard (Bucciarelli and Ehrhard 1991, Bucciarelli 1993) refined this model with stronger stability conditions to arrive at a model that captures sequentiality at first order but is still not fully abstract. The second one consists in keeping concrete data structures as objects but using sequential algorithms rather than functions as morphisms (Berry and Curien 1982, 1985, Curien 1993).

Chapter 3

Sequential Strategies

In this chapter we develop the theory of sequential (or deterministic) strategies in the game of higher types. As described in the introduction, the aim is to describe the interaction between a functional and its arguments as a play. In Section 3.1 we introduce the rules of the game for simple types. In Section 3.2 we introduce the sequential strategies and their representation as decision trees. In Section 3.3 we describe some basic properties of sequential strategies and present examples. In section 3.5 we introduce a PCF-like syntax for finite decision trees. In section 3.4 we extend the set of types with products to get a cartesian closed category. In Section 3.6 we show that all types can be faithfully embedded in pure types on the level of strategies. We end in Section 3.7 with some variants of sequential strategies.

3.1 The game

As mentioned before we describe the interaction between a functional of type τ with its arguments as a play between two players in the Game of Higher Types. We start with an informal description of the game in form of rules how to play the game. Then we formalize these rules and finally we prove some basic properties of plays.

3.1.1 Rules of the game. A play is a sequence of alternating moves of two players P (player, program) and O (opponent, oracle).

Moves are either *questions* denoted by $?a$, where a is an occurrence in τ, or *answers* denoted by $!v$, where $v \in V = \bigcup\{V_\gamma \mid \gamma \in \Gamma\}$ is a value.

The choice of moves is restricted by the following rules:

- Each play starts with the (not enabled and not repeatable) question $?\varepsilon$ of O (*initial question*).

- The question $?a.i$ can only be played if question $?a$ is visible at that moment. If there is more than one visible occurrence of $?a$ it has to be specified by the player which one is the *enabling for* $?a.i$, or in other words to which question $?a$ the new question $?a.i$ refers. All moves between the enabling $?a$ and $?a.i$ are hidden for the opposing player while $?a.i$ is visible.

- An answer $!v$ answers the last open question $?a$. In this case we say $!v$ refers to $?a$ or $?a$ enables $!v$. All moves between this question $?a$ and $!v$ are hidden for the opposing player. If a is an occurrence of γ in τ then v has to be in V_γ.

- The last move is *visible*, as are all moves that are not hidden by the above rules.

- A question is *open* if it is played but not answered.

A play is *finished* if the initial question $?\varepsilon$ is answered. If this last answer is $!v$ then we call v the *result* of this play. If s is a non-finished play, then we call the sequence of moves of s that are visible for the player in turn *view* of the play. If P is in turn we call it P*-view* otherwise O*-view*.

Note that Player P can only play questions $?a$ with $|a|$ odd and player O only those with $|a|$ even. $\Diamond$

Now we give some justifications for the choice of the rules. The rules for hiding some of the played moves are to avoid the use of "intensional aspects".

The question $?a.i$ asks for the i-th argument of $?a$. The answer to $?a.i$ must not depend on the knowledge which arguments of $?a$ have already been evaluated, especially, it must not depend on the order in which the arguments are evaluated. The hiding of the moves between $?a$ and $?a.i$ serves exactly for this request.

Hiding the moves between $?a$ and the respective answer $!v$ causes that only the answer counts, not the way to get it. For example, it is not visible which arguments $?a.i$ had to be evaluated to get the result. So, in contrast to the sequential algorithm approach, a strictness tester is not definable.

Formally we describe a play by two sequences of the same length, the first sequence consisting of the moves and the second sequence consisting of the references to the respective enablings. The best thing to understand the references is to see them as pointers.

3.1.2 Definition. The set of *moves* over a type τ is given by

$$\mathsf{Moves}^\tau = \{?a \mid a \text{ occurrence in } \tau\} \cup \{!v \mid v \in V\}.$$

We set $\mathsf{Moves} = \bigcup\{\mathsf{Moves}^\tau \mid \tau \in \mathsf{TS}\}$. Moves of the form $?a$ are called *questions* and those of the form $!v$ are called *answers*. $\Diamond$

The definition of a play will be given in two steps. First we introduce pre-plays respecting the rules without visibility conditions. For pre-plays we can define the notion of view. Finally we use the views to determine those pre-plays that are plays.

3.1.3 Definition. A *pre-play* (s, r) consists of two finite sequences $s \in \mathsf{Moves}^*$ of moves and $r \in \mathbb{N}^*$ of references satisfying for all $i, j \le |s|$:

- $|s| = |r|$,
- $s_1 = ?\varepsilon$, $r_1 = 0$, and if $s_i = ?\varepsilon$ then $i = 1$, i.e. the first move is always $?\varepsilon$ and this is the only occurrence of $?\varepsilon$,
- for $i > 1$ hold $r_i < i$, $i - r_i$ odd, and $s_{r_i} = ?a$ for some occurrence a, i.e. each reference points to a question of the opposing player already played,
- if $s_i = !v$, $s_j = !v'$ with $i \neq j$ then $r_i \neq r_j$, i.e. each question will be answered at most once,
- if $s_i = !v$, $r_i < j < i$, $s_j = ?a$ then there is a $j' < i$ with $s'_j = !v'$ and $r'_j = j$, i.e. each answer refers to the last open question,
- if $s_i = ?a.l$ then $s_{r_i} = ?a$.

If all moves in s are in Moves^τ we say that (s, r) is a pre-play over τ. ◊

3.1.4 Example. First we list some pre-plays:

1. $(?\varepsilon \cdot !\mathtt{t}, 0 \cdot 1)$
2. $(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot !\mathtt{t} \cdot !\mathtt{t} \cdot !\mathtt{f}, 0 \cdot 1 \cdot 2 \cdot 3 \cdot 2 \cdot 1)$
3. $(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.2, 0 \cdot 1 \cdot 2 \cdot 1 \cdot 4)$
4. $(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.1 \cdot ?1.1.1, 0 \cdot 1 \cdot 2 \cdot 1 \cdot 4 \cdot 3)$
5. $(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.2 \cdot ?1.1.1, 0 \cdot 1 \cdot 2 \cdot 1 \cdot 4 \cdot 3)$

and here are some sequences which are not pre-plays:

6. $(?\varepsilon \cdot ?1 \cdot ?2, 0 \cdot 1 \cdot 1)$, since $3 - r_3 = 2$ is not odd.
7. $(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot !\mathtt{t}, 0 \cdot 1 \cdot 2 \cdot 1)$, since $?1$ and $?1.1$ are not answered, but $?\varepsilon$ is. ◊

Now we can formalize the notion of view:

3.1.5 Definition. The *view* of a pre-play (s, r) is the shortest substring s' of s closed under the following clauses:

- $s_{|s|} \in s'$,
- If $s_k \in s'$ with $k > 1$ and $|s| - k$ even, then $s_{r_k} \in s'$,
- If $s_k \in s'$ with $k > 1$ and $|s| - k$ odd, then $s_{k-1} \in s'$.

We write $s' = \mathsf{view}(s, r)$. If $|s|$ is odd, we call the view *P-view*, if it is even *O-view*. $\Diamond$

3.1.6 Example. We show the views of the pre-plays from Example 3.1.4:

1. $\mathsf{view}(?\varepsilon \cdot !\mathtt{t}, 0 \cdot 1) = ?\varepsilon \cdot !\mathtt{t}$
2. $\mathsf{view}(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot !\mathtt{t} \cdot !\mathtt{t} \cdot !\mathtt{f}, 0 \cdot 1 \cdot 2 \cdot 3 \cdot 2 \cdot 1) = ?\varepsilon \cdot !\mathtt{f}$
3. $\mathsf{view}(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.2, 0 \cdot 1 \cdot 2 \cdot 1 \cdot 4) = ?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.2$
4. $\mathsf{view}(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.1 \cdot ?1.1.1, 0 \cdot 1 \cdot 2 \cdot 1 \cdot 4 \cdot 3) = ?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1.1.1$
5. $\mathsf{view}(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.2 \cdot ?1.1.1, 0 \cdot 1 \cdot 2 \cdot 1 \cdot 4 \cdot 3) = ?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1.1.1$ $\Diamond$

Finally we get

3.1.7 Definition. A pre-play (s, r) is a *play* iff for all $1 \leq i < |s|$ we have $s_{r_{i+1}} \in \mathsf{view}(s_{\leq i}, r_{\leq i})$. These are the *finite plays.* An *infinite play* (s, r) consists of two infinite sequences such that each initial segment is a finite play, i.e. for all $i \in \mathbb{N}$ $(s_{\leq i}, r_{\leq i})$ is a finite play. $\Diamond$

3.1.8 Example. All the pre-plays from Example 3.1.4 are finite plays. The following pre-plays are not plays:

- Let $(s, r) = (?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.2, 0 \cdot 1 \cdot 2 \cdot 1 \cdot 2)$, then $\mathsf{view}(s_{\leq 4}, r_{\leq 4}) = s_1 \cdot s_4 = ?\varepsilon \cdot ?1$ but $s_5 = ?1.2$ refers to s_2 which is not in the view.
- Let $(s, r) = (?\varepsilon \cdot ?1 \cdot ?1.1 \cdot !\mathtt{t} \cdot ?1.2 \cdot ?1.1.1, 0 \cdot 1 \cdot 2 \cdot 1 \cdot 2 \cdot 3)$, then $\mathsf{view}(s_{\leq 5}, r_{\leq 5}) = s_1 \cdot s_2 \cdot s_5 = ?\varepsilon \cdot ?1 \cdot ?1.2$ but $s_6 = ?1.1.1$ refers to s_3 which is not in the view. $\Diamond$

3.1.9 Lemma.
(a) Let (s, r) be a play of length n then

- *either $r_n = 1$ and $\mathsf{view}(s, r) = ?\varepsilon \cdot s_n$,*
- *or $\mathsf{view}(s, r) = \mathsf{view}(s_{<r_n}, r_{<r_n}) \cdot s_{r_n} \cdot s_n$*

(b) Let (s, r) be a play of length n and $1 \leq i \leq n$ with $n - i$ even. If $s_i \in \mathsf{view}(s, r)$ then $\mathsf{view}(s_{\leq i}, r_{\leq i})$ is an initial segment of $\mathsf{view}(s, r)$.

Proof. (a) Follows directly from the definitions and (b) follows from (a) by induction over $n - i$. □

The next lemma states that a player does never see his own answers.

3.1.10 Lemma. *Let (s, r) be a play. For all $i \leq |s|$ if s_i is an answer with $s_i \in$ $\mathsf{view}(s, r)$ then $|s| - i$ is even.*

Proof. We prove this by induction over the length of the view $n = |\mathsf{view}(s, r)|$. For $n = 1$, then $\mathsf{view}(s, r) = ?\varepsilon$, and for $n = 2$, then $\mathsf{view}(s, r) = ?\varepsilon \cdot !v$ or $\mathsf{view}(s, r) = \varepsilon \cdot ?j$ the lemma is true. If $n > 2$ then

$$\mathsf{view}(s, r) = \mathsf{view}(s_{<r_n}, r_{r_n}) \cdot s_{r_n} \cdot s_n$$

Now s_n refers to s_{r_n}, hence s_{r_n} is a question and $n - r_n$ is odd. Let s_i be an answer in $\mathsf{view}(s_{<r_n}, r_{<r_n})$. By the induction hypothesis $(r_n - 1) - i$ is even, hence $n - i = (n - r_n) + 1 + ((r_n - 1) - i)$ is even too. □

As an immediate consequence we get:

3.1.11 Lemma. *Let $\tau = (\tau_1, \ldots, \tau_k \to \gamma)$. For all O-views $\mathsf{view}(s, r)$ (i.e. $|s|$ even) over τ there exists an $i \leq k$ such that all questions but the first in the view are of the form $?i.a$.*

Proof. We prove the lemma by induction over $n = |s|$. The base case is trivial. So we consider $n > 2$. If $s_{r_n} = ?\varepsilon$ then $\mathsf{view}(s, r) = ?\varepsilon \cdot ?i$ for some $i \leq k$. If $s_{r_n} = ?a.j$ then $\mathsf{view}(s) = \mathsf{view}(s_{<r_n}) \cdot s_{r_n} \cdot s_n$ and $s_n = !v$ or $s_n = ?a.j.j'$. Since s is a play we have $s_{r_{r_n}} = ?a \in \mathsf{view}(s_{<r_n})$. By induction all questions in $\mathsf{view}(s_{<r_n})$ are of the form $?i.b$ for the same $i \leq k$. With $?a \in \mathsf{view}(s_{<r_n})$ this is also true for $?a.j$ and $?a.j.j'$. □

For plays we can introduce yet another representation using references relative to the view. These relative references are similar to de Bruijn indices for lambda terms.

3.1.12 Definition. Let (s, r) be a play, then we say (s, r') describes the same play *with relative references* (*w.r.r.*) where r' is defined as follows:

- $r'_1 = 0$.
- If $s_{i+1} = !v$, then $r'_{i+1} = 0$.
- If $s_{i+1} = ?a.l$ and $r_{i+1} = j$, then $r'_{i+1} = k$ where k is the number of $?a$ in $\mathsf{view}(s_{\leq i+1}, r_{\leq i+1})$ between s_j and s_{i+1}, not including s_j.

We write $\mathsf{relref}(s, r) = r'$. ◊

3.1.13 Remark. The relation between relative references and absolute references is one to one, so we can define a function absref that translates a play given with relative references into one with absolute references such that

$$\mathsf{absref}(s, \mathsf{relref}(s, r)) = r \qquad \text{and} \qquad \mathsf{relref}(s, \mathsf{absref}(s, r')) = r'. \qquad \diamondsuit$$

3.1.14 Example.

- $\mathsf{relref}(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.1 \cdot ?1.1.1, 0 \cdot 1 \cdot 2 \cdot 1 \cdot 4 \cdot 3) = 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \cdot 1$
- $\mathsf{relref}(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.1 \cdot ?1.1.1, 0 \cdot 1 \cdot 2 \cdot 1 \cdot 4 \cdot 5) = 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0$
- $\mathsf{relref}(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.2 \cdot ?1.1.1, 0 \cdot 1 \cdot 2 \cdot 1 \cdot 4 \cdot 3) = 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0 \cdot 0$ $\diamondsuit$

3.1.15 Definition. We call a play (s, r) with relative references *deterministic*, if for all $i, i' \leq |s|$:

$$\mathsf{view}(s_{\leq i}, r_{\leq i}) = \mathsf{view}(s_{\leq i'}, r_{\leq i'}) \qquad \text{implies} \qquad s_{i+1} = s_{i'+1} \text{ and } r_{i+1} = r_{i'+1}. \qquad \diamondsuit$$

3.1.16 Example. From the plays in Example 3.1.4, 1, 2, and 4 are deterministic whereas the plays 3. and 5. are not. We consider play 3., i.e.

$$(s, r) = (?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.2, 0 \cdot 1 \cdot 2 \cdot 1 \cdot 4)$$

where all relative references are 0. Then

$$\mathsf{view}(s_{\leq 2}, r_{\leq 2}) = \mathsf{view}(s_{\leq 4}, r_{\leq 4}) = ?\varepsilon \cdot ?1$$

but $s_3 = ?1.1 \neq ?1.2 = s_5$.

Now we consider another play (s, r):

$$\begin{array}{rl}
s = & ?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.1 \cdot ?1.1.1 \cdot !\mathtt{t} \cdot !\mathtt{t} \cdot ?1.1 \cdot ?1.1.1 \\
r = & 0 \cdot \; 1 \cdot \; 2 \cdot \; 1 \cdot \; 4 \cdot \; 5 \cdot 6 \cdot 5 \cdot \; 4 \cdot \; 3 \\
 & \\
r' = \mathsf{relref}(s, r) = & 0 \cdot \; 0 \cdot \; 0 \cdot \; 0 \cdot \; 0 \cdot \; 0 \cdot 0 \cdot 0 \cdot \; 0 \cdot \; 1 \\
 & \\
\mathsf{view}(s_{\leq 5}, r_{\leq 5}) = & ?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.1 \\
\mathsf{view}(s_{\leq 9}, r_{\leq 9}) = & ?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot \qquad ?1.1
\end{array}$$

Here is $s_6 = s_{10} = ?1.1.1$ but $r'_6 = 0 \neq 1 = r'_{10}$. Hence this play is not deterministic either. $\diamondsuit$

In the following it will be useful to have some decompositions of plays:

3.1.17 Definition. Let s be a play over $\tau = (\tau_1, \ldots, \tau_k \to \gamma)$ and $s_i = ?j$ with $1 \leq j \leq k$ then we call the subsequence of s consisting of s_i and all moves in s that are hereditarily enabled by s_i the *component of s starting in s_i*. We denote it by $s|_{s_i}$. $\diamondsuit$

3.1.18 Lemma. *Let s be a play and $s_i = ?j$ then all questions in $s|_{s_i}$ are of the form $?j.a$.*

Proof. This follows from the fact that all moves in s_{s_i} are hereditarily enabled by $s_i = ?j$. $\square$

The following lemma shows that the components of a play yield indeed a decomposition of the play:

3.1.19 Lemma. *Each move in a play but the first (and for a finished play the last) belongs exactly to one component.* $\square$

In general a component of a play does not consist of consecutive moves of the play. The maximal parts of a component that consist of consecutive moves are called *sections*:

3.1.20 Definition. Let s be a play and

$$s = ?\varepsilon \cdot x_1 \cdot \ldots \cdot y_1 \cdot x_2 \cdot \ldots \cdot y_2 \cdot \ldots \cdot x_i \cdot \ldots \cdot y_i \cdot \ldots$$

such that for all i the moves $x_i, \ldots, y_i$ belong to the same component and y_i and x_{i+1} belong to different components then we call $s^i = x_i, \ldots, y_i$ the *i-th section* of s. $\diamondsuit$

3.1.21 Lemma. *Let s be a play of length n.*

(a) If n is even then all moves in the O-view $\mathsf{view}(s)$ besides $?\varepsilon$ belong to the same component.

(b) If $n \geq 3$ is odd then the last two moves s_n and s_{n-1} belong to the same component.

Proof. We prove (a) and (b) simultaneously by induction on the length of s. The respective base cases are trivial. So let $|s| = n > 3$:

(a) If $s_n = ?j$ for some j then $\mathsf{view}(s) = ?\varepsilon \cdot ?j$ and we are done. If $s_n \neq ?j$ for any j then by Lemma 3.1.9 $\mathsf{view}(s) = \mathsf{view}(s_{<r_n}) \cdot s_{r_n} \cdot s_n$. Note that s_{r_n} and s_n are in the same component because s_n is enabled by s_{r_n}. By induction hypothesis (a) all moves in $\mathsf{view}(s_{<r_n})$ besides $?\varepsilon$ belong to the same component. By induction hypothesis (b) is s_{r_n-1}, the last move of $\mathsf{view}(s_{<r_n})$, in the same component as s_{r_n}. So (a) is true for s.

(b) The last move s_n refers to some move in $\mathsf{view}(s_{<n})$. By induction hypothesis (a) all moves in $\mathsf{view}(s_{<n})$ besides $?\varepsilon$ belong to the same component. As s_n is an O-move it can not refer to $?\varepsilon$. Hence s_n and s_{n-1} which is the last move in $\mathsf{view}(s_{<n})$ are in the same component. $\square$

As a consequence we get that only the player, not the opponent, may change the component:

3.1.22 Lemma. *Let $x_i, \ldots, y_i$ be a section of a play s, then x_i is a P-move.* $\square$

3.2 Sequential strategies and decision trees

As a play describes the interaction between a functional and its arguments, functionals and their arguments are represented by strategies for P and for O respectively.

3.2.1 Definition. A *sequential strategy p for P* (over τ) is a partial function from P-views (over τ) to allowed moves with relative references, that is prefixed closed in the following sense:

$$s \cdot m \cdot m' \in \mathsf{dom}(p) \quad \text{implies} \quad s \in \mathsf{dom}(p) \text{ and } p(s) = (m, r).$$

A *sequential strategy for O* (over τ) is a partial function from O-views (over τ) to allowed moves with relative references that is prefixed closed. $\Diamond$

In the examples we will show the relative references only if they are not 0.

3.2.2 Example. (a) We give an example of a strategy p for P over τ (τ as in example 2.1.4) representing the functional $\lambda gz.g(z, 5)$:

$$\begin{array}{lcll}
p(?\varepsilon) & = & ?1 & \\
p(?\varepsilon \cdot ?1 \cdot !n) & = & !n & (n \in \mathbb{N}) \\
p(?\varepsilon \cdot ?1 \cdot ?1.1) & = & ?2 & \\
p(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?2 \cdot !m) & = & !m & (m \in \mathbb{N}) \\
p(?\varepsilon \cdot ?1 \cdot ?1.2) & = & !5 &
\end{array}$$

(b) The following strategy q is a strategy for O over τ representing in the first component the function $\lambda xy.x^2 + y$ and in the second the constant 3:

$$\begin{array}{lcll}
q(?\varepsilon \cdot ?1) & = & ?1.1 & \\
q(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot !n) & = & ?1.2 & (n \in \mathbb{N}) \\
q(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot !n \cdot ?1.2 \cdot !m) & = & !k & (n, m \in \mathbb{N},\ k = n^2 + m) \\
q(?\varepsilon \cdot ?2) & = & !3 &
\end{array}$$

$\Diamond$

Now we show how O-strategies can be defined from P-strategies of lower types.

3.2.3 Definition. a) Let s be an O-view over $\tau = (\tau_1, \ldots, \tau_k \to \gamma)$ and let i be such that all questions but the first are of the form $?i.a$ which exists by Lemma 3.1.11. In this case let $s^{(-i)}$ result from s by omitting the first question $?\varepsilon$ and by substituting each question $?i.a$ with $?a$. If the questions in s are of the form $?j.a$ with $j \neq i$ then $s^{(-i)}$ is undefined.

b) Vice versa, if s' is a P-view over τ_i than we define $s'^{(i)}$ as the result of substituting each question $?a$ in s' with $?i.a$ and of adding the question $?\varepsilon$ as the new first element.

$\Diamond$

Immediately from the definition we get the following observation:

3.2.4 Lemma. *Let $\tau = (\tau_1, \dots, \tau_k \to \gamma)$ and $1 \leq i \leq k$. Let s be an O-view over τ and s' a P-view over τ_i. Then*

a) if $s^{(-i)}$ is defined then it is a P-view over τ_i and $(s^{(-i)})^{(i)} = s$,

b) $s'^{(i)}$ is an O-view over τ, and $(s'^{(i)})^{(-i)} = s'$. □

3.2.5 Example. Consider the following O-view s.

$$\begin{array}{rrcl} & s & = & ?\varepsilon \cdot ?1 \cdot ?1.2 \cdot !3 \cdot ?1.1 \cdot !5 \\ \text{then} & s^{(-1)} & = & ?\varepsilon \cdot \quad ?2 \cdot !3 \cdot \quad ?1 \cdot !5 \end{array}$$ ◊

3.2.6 Definition. Let q be a sequential strategy for O over $\tau = (\tau_1, \dots, \tau_k \to \gamma)$ then we get a sequential strategy $q^{(-i)}$ for P over τ_i by

$$q^{(-i)}(s') = \begin{cases} (?a, r), & \text{if } q(s'^{(i)}) = (?i.a, r), \\ q(s'^{(i)}), & \text{otherwise.} \end{cases}$$

Vice versa, if p is a sequential strategy for P over τ_i then we get a sequential strategy $p^{(i)}$ for O over τ by

$$p^{(i)}(s) = \begin{cases} (?i.a, r), & \text{if } p(s^{(-i)}) = (?a, r), \\ p(s^{(-i)}), & \text{otherwise.} \end{cases}$$

We have $q = \bigcup_{i=1}^{k} (q^{(-i)})^{(i)}$, hence all sequential strategies for O over τ are representable by a tuple of sequential strategies p_i for P over τ_i $(i = 1, \dots, k)$. In this case we write $q = (p_1, \dots, p_k)^O$. ◊

3.2.7 Example. The strategy q from example 3.2.2 (b) can be written as $q = (p_1, p_2)^O$ with:

$$\begin{array}{lcll} p_1(?\varepsilon) & = & ?1 & \\ p_1(?\varepsilon \cdot ?1 \cdot !n) & = & ?2 & (n \in \omega) \\ p_1(?\varepsilon \cdot ?1 \cdot !n \cdot ?2 \cdot !m) & = & !k & (n, m \in \mathbb{N},\ k = n^2 + m) \\ p_2(?\varepsilon) & = & !3 & \end{array}$$ ◊

So we need only sequential strategies for P (over τ) called simply *sequential strategies* (over τ). By SeqStr^τ we denote the set of sequential strategies over τ and the set of all sequential strategies is $\mathsf{SeqStr} = \bigcup_{\tau \in \mathsf{TS}} \mathsf{SeqStr}^\tau$.

3.2.8 Definition. Let p be a sequential strategy over $\tau = (\tau_1, \dots, \tau_k \to \gamma)$ and p_i $(i = 1, \dots, k)$ sequential strategies over τ_i. If there exists a finished play between P with strategy p and O with strategy $(p_1, \dots, p_k)^O$ with value v then we write $p[p_1, \dots, p_k] = v$, otherwise $p[p_1, \dots, p_k]$ is undefined $(= \perp)$. ◊

3.2.9 Remark. The finished play is unique, if it exists. That means $p[\,\cdot\,]$ is a partial function from $\mathsf{SeqStr}^{\tau_1} \times \cdots \times \mathsf{SeqStr}^{\tau_k}$ to V_γ or equivalently a function from $\mathsf{SeqStr}^{\tau_1} \times \cdots \times \mathsf{SeqStr}^{\tau_k}$ to $(V_\gamma)_\perp$ ◊

3.2.10 Example. Let p, p_1 and p_2 be the strategies from examples 3.2.2 and 3.2.7 respectively. The play for $p[p_1, p_2]$ proceeds as follows:

player	view	next move
O		$?\varepsilon$
P	$?\varepsilon$	$?1$
O	$?\varepsilon \cdot ?1$	$?1.1$
P	$?\varepsilon \cdot ?1 \cdot ?1.1$	$?2$
O	$?\varepsilon \cdot \quad ?2$	$!3$
P	$?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?2 \cdot !3$	$!3$
O	$?\varepsilon \cdot ?1 \cdot ?1.1 \cdot \quad !3$	$?1.2$
P	$?\varepsilon \cdot ?1 \cdot \quad ?1.2$	$!5$
O	$?\varepsilon \cdot ?1 \cdot ?1.1 \cdot \quad !3 \cdot ?1.2 \cdot !5$	$!14$
P	$?\varepsilon \cdot ?1 \cdot \quad !14$	$!14$

Hence the value of this play is $p[p_1, p_2] = 14$. ◊

Now we introduce another representation of the sequential strategies. The form of this representation is similar to that used by Cartwright and Felleisen (1992) for sequential algorithms and by Bucciarelli (1993) for Kleene's oracles.

3.2.11 Definition. A sequential strategy p can be visualized as a *decision tree* being a labeled tree where the labels at the edges stand for O-moves and the labels at the vertices stand for P-moves (with relative references if not 0). So if $p(?\varepsilon \cdot \ldots \cdot s_{n-1}) = s_n$ then there is a path labeled with $?\varepsilon, \ldots, s_{n-1}$ from the root to a vertex labeled with s_n. ◊

3.2.12 Example. Let p, p_1 and p_2 be the strategies from examples 3.2.2 and 3.2.7 respectively. The corresponding decision trees are the following:

$$\begin{tikzcd}
p: \arrow[r, dash, "?\varepsilon"] & ?1 \arrow[r, dash, "!n"] \arrow[dr, dash, "?1.1"] \arrow[ddr, dash, "?1.2"] & !n \\
& & ?2 \arrow[r, dash, "!m"] & !m \\
& & !5
\end{tikzcd}$$

$$\begin{tikzcd}
p_1: \arrow[r, dash, "?\varepsilon"] & ?1 \arrow[r, dash, "!n"] & ?2 \arrow[r, dash, "!m"] & !(n^2{+}m)
\end{tikzcd}$$

$$\begin{tikzcd}
p_2: \arrow[r, dash, "?\varepsilon"] & !3
\end{tikzcd}$$

In fact the trees for p and p_1 are infinite. For instance the notation $\overset{!n}{\rule{1.5em}{0.4pt}}\,!n$ means that there is a subtree of this form for each $n \in \mathbb{N}$. ◊

3.2.13 Definition. The *rank* of a strategy p is the supremum of the returned relative references, i.e.

$$\mathsf{rank}(p) = \sup\{r \mid (?a, r) \in \mathsf{im}(p)\}$$

It might be a natural number or ∞. $\diamondsuit$

3.3 Properties and examples

A strategy p of type $\tau = (\tau_1, \dots, \tau_k \to \gamma)$ may be seen as an operator from SeqStr^{τ_1} to SeqStr^{σ} with $\sigma = (\tau_2, \dots, \tau_k \to \gamma)$ in the sense that for each $p_1 \in \mathsf{SeqStr}^{\tau_1}$ there exists a unique $q \in \mathsf{SeqStr}^{\sigma}$ such that for all $p_i \in \mathsf{SeqStr}^{\tau_i}$ $2 \le i \le k$ the plays for $p[p_1, \dots, p_k]$ and $q[p_2, \dots, p_k]$ correspond in a natural way to each other.

3.3.1 Lemma. *Let $\tau = (\tau_1, \dots, \tau_k \to \gamma)$ and $\sigma = (\tau_2, \dots, \tau_k \to \gamma)$. Moreover let $p \in \mathsf{SeqStr}^{\tau}$ and $p_1 \in \mathsf{SeqStr}^{\tau_1}$. Then there exists a unique strategy $q \in \mathsf{SeqStr}^{\sigma}$ such that for all $p_i \in \mathsf{SeqStr}^{\tau_i}$ with $2 \le i \le k$*

$$q[p_2, \dots, p_k] = p[p_1, \dots, p_k],$$

and such that the play for $q[p_2, \dots, p_k]$ is the same as the σ-projection of the play of $p[p_1, \dots, p_k]$. We write $q = p[p_1]$ for this q.

Proof. Let s be a P-view over σ. By $s^{\to}$ we denote the corresponding sequence w.r.t. τ, i.e. $s^{\to}$ is the same as s with each occurrence $?i.a$ substituted by $?(i{+}1).a$. Now we define a simulation sequence (t, r) with relative references according to p, p_1 and s. Simultaneously we define a function $h(i)$ for $i \le |t|$ with the meaning that in defining $(t_{\le i}, r_{\le i})$ we have already used $s_{\le h(i)}$. So at the beginning we have used nothing, hence $h(0) = 0$. The sequence t starts with $?\varepsilon$ as does s. Hence we define $(t_1, r_1) = (?\varepsilon, 0)$ and $h(1) = 1$. Now let us assume that $(t_{\le i}, r_{\le i})$ and $h(i)$ have been defined so far. Then (t_{i+1}, r_{i+1}) and $h(i+1)$ will be defined by cases:

- If i is even, $\mathsf{view}(t_{\le i}) = ?\varepsilon \cdot ?1 \cdot s'$ and $p_1^{(1)}(\mathsf{view}(t_{\le i}))$ is defined, then $(t_{i+1}, r_{i+1}) = p_1^{(1)}(\mathsf{view}(t_{\le i}))$ and $h(i+1) = h(i)$.

- If i is even and $\mathsf{view}(t_{\le i}) = ?\varepsilon \cdot ?j \cdot s'$ with $2 \le j \le k$ then $(t_{i+1}, r_{i+1}) = (s^{\to}_{h(i)+1}, 0)$ and $h(i+1) = h(i) + 1$.

- If i is odd, $p(\mathsf{view}(t_{\le i})) = (!v, 0)$ and the last open question in $t_{\le i}$ is of the form $?1.a$ then $(t_{i+1}, r_{i+1}) = (!v, 0)$ and $h(i+1) = h(i)$.

- If i is odd and $p(\mathsf{view}(t_{\le i})) = (?1.a, m)$ then $(t_{i+1}, r_{i+1}) = (?1.a, m)$ and $h(i{+}1) = h(i)$.

- If i is odd, $p(\mathsf{view}(t_{\leq i})) = (?j.a, m)$ with $j \geq 2$ and $s^{\rightarrow}_{h(i)+1} = ?j.a$ then $(t_{i+1}, r_{i+1}) = (?j.a, m)$ and $h(i+1) = h(i) + 1$.
- In all other cases (t_{i+1}, r_{i+1}) and $h(i+1)$ are undefined.

The simulation sequence t is a possible infinite play but not necessarily deterministic. Now we are in a position to define $q(s)$. Let t be the simulation sequence of p, p_1 and s. If $i = |t|$ is odd, $h(i) = |s|$ and $p(\mathsf{view}(t_{\leq i})) = (!v, 0)$ then $q(s) = (!v, 0)$. If $i = |t|$ is odd, $h(i) = |s|$ and $p(\mathsf{view}(t_{\leq i})) = (?j.a, m)$ with $j \geq 2$ then $q(s) = (?(j-1).a, m')$ where m' is the corresponding relative reference w.r.t. s as m is w.r.t. t. □

3.3.2 Remark. For $k = 1$ there is a conflict of notation. $p[p_1]$ denotes on one side the value v of the final answer $!v$ if it exists or $\perp$ if not and now a strategy of type γ. But in the former case the strategy $p[p_1]$ as defined in the preceding lemma is the strategy $\xrightarrow{?\varepsilon} !v$ and in the later it is the undefined strategy. ◊

3.3.3 Remark. Note that the construction of $q = p[p_1]$ is monotone in p and p_1. ◊

3.3.4 Example. Let p and p_1 be the strategies from example 3.2.12. We want to determine the strategy $q = p[p_1]$ as given by the preceding lemma. The type of q will be $(\mathbf{N} \to \mathbf{N})$. We construct the decision tree for q from the root.

- Let $s = ?\varepsilon$, then $s^{\rightarrow} = ?\varepsilon$. The simulation sequence t for p, p_1 and s is $t = ?\varepsilon \cdot ?1 \cdot ?1.1$ with all references 0 and $h(i) = 1$ for $1 \leq i \leq 3$. $\mathsf{view}(t) = t$ and $p(t) = ?2$. Hence $q(?\varepsilon) = ?1$.
- Let $s = ?\varepsilon \cdot ?1 \cdot !n$, then $s^{\rightarrow} = ?\varepsilon \cdot ?2 \cdot !n$. The simulation sequence t for p, p_1 and s is $t = ?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?2 \cdot !n \cdot !n \cdot ?1.2 \cdot !5 \cdot !(n^2{+}5)$ with all references 0 and $h(i) = 1$ for $1 \leq i \leq 3$, $h(4) = 2$ and $h(i) = 3$ for $5 \leq i \leq 9$. $\mathsf{view}(t) = ?\varepsilon \cdot ?1 \cdot !(n^2{+}5)$ and $p(?\varepsilon \cdot ?1 \cdot !(n^2{+}5)) = !(n^2{+}5)$. Hence $q(?\varepsilon \cdot ?1 \cdot !n) = !(n^2{+}5)$.

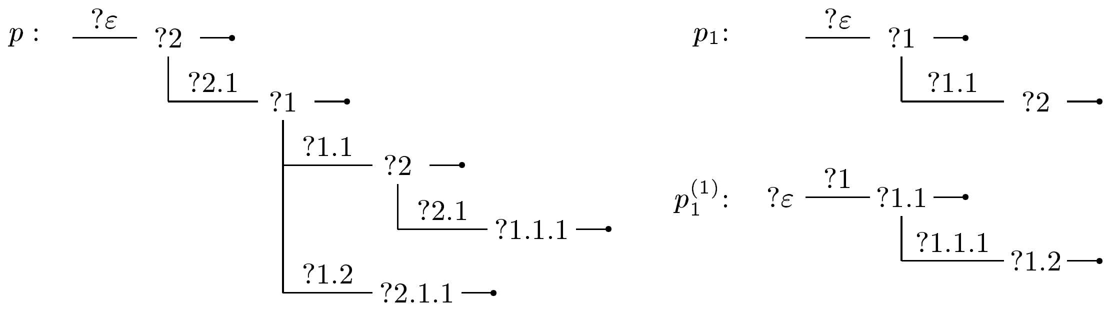
So the decision tree for $q = p[p_1]$ is:

$$q : \quad \xrightarrow{?\varepsilon} ?1 \xrightarrow{!n} !(n^2{+}5)$$

◊

The next example is a little bit more involved but it is the simplest one with $\mathsf{rank}(p) = \mathsf{rank}(p_1) = 0$ and $\mathsf{rank}(p[p_1]) = 1$.

3.3.5 Example. Let $p \in \mathsf{SeqStr}^{\tau}$ and $p_1 \in \mathsf{SeqStr}^{\tau_1}$ with $\tau = (((\gamma \to \gamma), \gamma \to \gamma), ((\gamma \to \gamma) \to \gamma) \to \gamma)$ be the strategies given by the following decision trees, where $\longrightarrow\!\bullet$ stands for $\xrightarrow{!v} !v$ $(v \in V_\gamma)$.



We construct the decision tree for $q = p[p_1] \in \mathsf{SeqStr}^{(((\gamma\to\gamma)\to\gamma)\to\gamma)}$ from the root.

- Let $s = ?\varepsilon$, then $s^{\to} = ?\varepsilon$. The simulation sequence t for p, p_1 and s is $t = ?\varepsilon$, $\mathsf{view}(t) = t$ and $p(t) = ?2$. Hence $q(?\varepsilon) = ?1$.

- Let $s = ?\varepsilon \cdot ?1 \cdot !v$, then $s^{\to} = ?\varepsilon \cdot ?2 \cdot !v$. The simulation sequence t for p, p_1 and s is $t = ?\varepsilon \cdot ?2 \cdot !v$, $\mathsf{view}(t) = t$ and $p(t) = !v$. Hence $q(?\varepsilon \cdot ?1 \cdot !v) = !v$.

- Let $s = ?\varepsilon \cdot ?1 \cdot ?1.1$, then $s^{\to} = ?\varepsilon \cdot ?2 \cdot ?2.1$. The simulation sequence t for p, p_1 and s is $t = ?\varepsilon \cdot ?2 \cdot ?2.1 \cdot ?1 \cdot ?1.1$, $\mathsf{view}(t) = t$ and $p(t) = ?2$. Hence $q(?\varepsilon \cdot ?1 \cdot ?1.1) = ?1$.

- Let $s = ?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot !v$, then $s^{\to} = ?\varepsilon \cdot ?2 \cdot ?2.1 \cdot ?2 \cdot !v$. The simulation sequence t for p, p_1 and s is $t = ?\varepsilon \cdot ?2 \cdot ?2.1 \cdot ?1 \cdot ?1.1 \cdot ?2 \cdot !v$, $\mathsf{view}(t) = t$ and $p(t) = !v$. Hence $q(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot !v) = !v$.

- Let $s = ?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.1$, then $s^{\to} = ?\varepsilon \cdot ?2 \cdot ?2.1 \cdot ?2 \cdot ?2.1$. The simulation sequence t for p, p_1 and s is

$$t = ?\varepsilon \cdot ?2 \cdot \underline{?2.1} \cdot ?1 \cdot ?1.1 \cdot ?2 \cdot ?2.1 \cdot ?1.1.1 \cdot ?1.2$$

  with

$$\mathsf{view}(t) = ?\varepsilon \cdot ?2 \cdot \underline{?2.1} \cdot ?1 \qquad\qquad\qquad\qquad \cdot ?1.2$$

  and $p(\mathsf{view}(t)) = ?2.1.1$. The relative reference of this move is 0 w.r.t. t but with respect to

$$s = ?\varepsilon \cdot ?2 \cdot \underline{?2.1} \qquad\qquad \cdot ?2 \cdot ?2.1$$

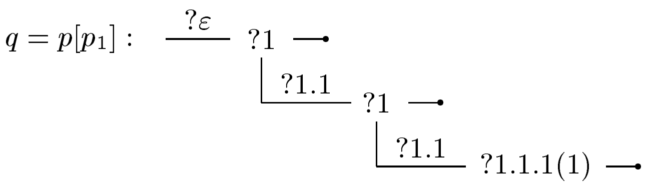
  it is 1. Hence $q(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.1) = (?1.1.1, 1)$.

- Let $s = ?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.1 \cdot ?1.1.1 \cdot !v$, then $s^{\to} = ?\varepsilon \cdot ?2 \cdot ?2.1 \cdot ?2 \cdot ?2.1 \cdot ?2.1.1 \cdot !v$. The simulation sequence t for p, p_1 and s is

$$t = ?\varepsilon \cdot ?2 \cdot ?2.1 \cdot ?1 \cdot ?1.1 \cdot ?2 \cdot ?2.1 \cdot ?1.1.1 \cdot ?1.2 \cdot ?2.1.1 \cdot !v,$$

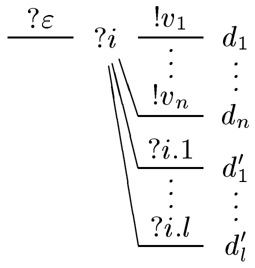
  $\mathsf{view}(t) = ?\varepsilon \cdot ?2 \cdot ?2.1 \cdot ?1 \cdot ?1.2 \cdot ?2.1.1 \cdot !v$ and $p(\mathsf{view}(t)) = !v$. Hence $q(?\varepsilon \cdot ?1 \cdot ?1.1 \cdot ?1 \cdot ?1.1 \cdot ?1.1.1 \cdot !v) = !v$.

So the decision tree for q is:



$\diamondsuit$

3.3.6 Theorem. *Let p be a finite sequential strategy of type $\tau = (\tau_1, \dots, \tau_k \to \gamma)$ of the form*



and let $q_i \in \mathsf{SeqStr}^{\tau_i}$ for $1 \le i \le k$. Then $p[q_1, \dots, q_k] = n$ if and only if there exists a $1 \le j \le n$ such that

$$q_i[\widehat{p_{i.1}}[q_1, \dots, q_k], \dots, \widehat{p_{i.l}}[q_1, \dots, q_k]] = v_j \quad \text{and} \quad p_{v_j}[q_q, \dots, q_k] = v$$

where $\widehat{p_{i.j}}$ and p_{v_j} are defined as follows: The decision tree for p_{v_j} is $\xrightarrow{?\varepsilon} d_j$.

The decision tree for $\widehat{p_{i.j}}$ is formed from $\xrightarrow{?i.j} d'_j$ by replacing this $?i.j$ by $?\varepsilon$ and all questions $?i.j.l.a$ hereditarily referring to this $?i.j$ by $?(l{+}k).a$, where all relative references have to be changed accordingly. $\square$

In the rest of this section we show that Kleene's schemata (S1) – (S8) and (S11) can be realized by sequential strategies. For that purpose let $\tilde{n}$ denote the strategy with the decision tree $\xrightarrow{?\varepsilon} !n$ for each $n \in \mathbb{N}$. Note that

$$\mathsf{SeqStr}^{\mathbf{N}} = \{\tilde{n} \mid n \in \mathbb{N}\} \cup \{\bot\}$$

where $\bot$ denotes the undefined strategy.

3.3.7 Example (S1) Successor.
Let $\tau = (\mathbf{N}, \tau_2, \dots, \tau_k \to \mathbf{N})$ and let the strategy $p_{succ} \in \mathsf{SeqStr}^{\tau}$ be given by

$$\begin{aligned} p_{succ}(?\varepsilon) &= ?1 \\ p_{succ}(?\varepsilon \cdot ?1 \cdot !n) &= !(n+1) \quad \text{for } n \in \mathbb{N}. \end{aligned}$$

Then for all $q_1 \in \mathsf{SeqStr}^{\mathbf{N}}$, $q_i \in \mathsf{SeqStr}^{\tau_i}$ $(2 \leq i \leq k)$ holds

$$p_{succ}[q_1, \ldots, q_k] = \begin{cases} \widetilde{(n+1)} & \text{if } q_1 = \tilde{n} \\ \perp & \text{if } q_1 = \perp. \end{cases} \qquad \Diamond$$

3.3.8 Example (S2) Constants.
Let $\tau = (\tau_1, \ldots, \tau_k \to \mathbf{N})$, $c \in \mathbb{N}$, and let the strategy $p^c_{const} \in \mathsf{SeqStr}^{\tau}$ be given by $p^c_{const}(?\varepsilon) = !c$. Then for all $q_i \in \mathsf{HSF}^{\tau_i}$ $(1 \leq i \leq k)$ we have

$$p^c_{const}[q_1, \ldots, q_k] = \tilde{c}. \qquad \Diamond$$

3.3.9 Example (S3) Identity.
Let $\tau = (\mathbf{N}, \tau_2, \ldots, \tau_k \to \mathbf{N})$ and let the strategy $p_{id} \in \mathsf{SeqStr}^{\tau}$ be given by

$$\begin{array}{lcll} p_{id}(?\varepsilon) & = & ?1 & \\ p_{id}(?\varepsilon \cdot ?1 \cdot !n) & = & !n & \text{for } n \in \mathbb{N}. \end{array}$$

Then for all $q_1 \in \mathsf{SeqStr}^{\mathbf{N}}$, $q_i \in \mathsf{SeqStr}^{\tau_i}$ $(2 \leq i \leq k)$ holds

$$p_{id}[q_1, \ldots, q_k] = q. \qquad \Diamond$$

3.3.10 Example (S4) Composition.
Let $\tau = (\tau_1, \ldots, \tau_k \to \mathbf{N})$, $\sigma = (\mathbf{N}, \tau_1, \ldots, \tau_k \to \mathbf{N})$, and $\rho = (\sigma, \tau, \tau_1, \ldots, \tau_k \to \mathbf{N}))$. Let the strategy $p_{compos} \in \mathsf{SeqStr}^{\rho}$ be given by

$$\begin{array}{lcll} p_{compos}(?\varepsilon) & = & ?1 & \\ p_{compos}(s \cdot ?1.1) & = & ?2 & \\ p_{compos}(s \cdot ?1.(j{+}1).a) & = & ?(j{+}2).a & \text{for } 1 \leq j \leq k \\ p_{compos}(s \cdot ?2.j.a) & = & ?(j{+}2).a & \text{for } 1 \leq j \leq k \\ p_{compos}(s \cdot ?1.(j{+}1).b \cdot ?(j{+}2).b \cdot ?(j{+}2).a) & = & ?1.(j{+}1).a & \text{for } 1 \leq j \leq k \\ p_{compos}(s \cdot ?2.j.b \cdot ?(j{+}2).b \cdot ?(j{+}2).a) & = & ?2.j.a & \text{for } 1 \leq j \leq k \\ p_{compos}(s \cdot !v) & = & !v & \end{array}$$

Let $p_1 \in \mathsf{SeqStr}^{\sigma}$ and $p_2 \in \mathsf{SeqStr}^{\tau}$ then $p = p_{compos}[p_1, p_2] \in \mathsf{SeqStr}^{\tau}$ and for all $q_i \in \mathsf{SeqStr}^{\tau_i}$ $(1 \leq i \leq k)$ holds

$$p[q_1, \ldots, q_k] = p_1[p_2[q_1, \ldots, q_k], q_1, \ldots, q_k]. \qquad \Diamond$$

3.3.11 Example (S5) Primitive Recursion.
Let $\tau = (\tau_1, \ldots, \tau_k \to \mathbf{N})$, $\sigma = (\mathbf{N}, \mathbf{N}, \tau_1, \ldots, \tau_k \to \mathbf{N})$, and $\rho = (\tau, \sigma, \mathbf{N}, \tau_1, \ldots, \tau_k \to \mathbf{N}))$. Let the strategy $p_{pr} \in \mathsf{SeqStr}^{\rho}$ be given by

$$\begin{array}{lll}
p_{pr}(?\varepsilon) & = ?3 & \\
p_{pr}(s \cdot ?a \cdot !v) & = !v & \text{for } a \neq 3 \\
p_{pr}(s \cdot ?3 \cdot !0) & = ?1 & \\
p_{pr}(s \cdot ?1.j.a) & = ?(j{+}3).a & \text{for } 1 \leq j \leq k \\
p_{pr}(s \cdot ?2.(j{+}2).a) & = ?(j{+}3).a & \text{for } 1 \leq j \leq k \\
p_{pr}(s \cdot ?1.j.b \cdot ?(j{+}3).b \cdot ?(j{+}3).a) & = ?1.j.a & \text{for } 1 \leq j \leq k \\
p_{pr}(s \cdot ?2.(j{+}2).b \cdot ?(j{+}3).b \cdot ?(j{+}3).a) & = ?2.(j{+}2).a & \text{for } 1 \leq j \leq k \\
p_{pr}(s \cdot ?3 \cdot !n \cdot (?2 \cdot ?2.1)^m) & = ?2 & \text{for } n \in \mathbb{N},\ 0 \leq m < n \\
p_{pr}(s \cdot ?3 \cdot !n \cdot (?2 \cdot ?2.1)^n) & = ?1 & \text{for } n \in \mathbb{N} \\
p_{pr}(s \cdot ?3 \cdot !n \cdot (?2 \cdot ?2.1)^m \cdot ?2 \cdot ?2.2) & = !(n{-}m) & \text{for } n \in \mathbb{N},\ 0 \leq m < n
\end{array}$$

Let $p_1 \in \mathsf{HSF}^\tau$ and $p_2 \in \mathsf{HSF}^\sigma$ then $p = p_{pr}[p_1, p_2] \in \mathsf{SeqStr}^{(\mathbf{N},\tau_1,\ldots,\tau_k \to \mathbf{N})}$ and for all $x \in \mathsf{SeqStr}^{\mathbf{N}}$ and $q_i \in \mathsf{HSF}^{\tau_i}$ $(1 \leq i \leq k)$ we have

$$p[x, \overline{q}] = \begin{cases} p_1[\overline{q}] & \text{if } x = \tilde{0} \\ p_2(p[\widetilde{(x{-}1)}, \overline{q}], x, \overline{q}] & \text{if } x = \tilde{n} \text{ and } n > 0 \\ \bot & \text{if } x = \bot \end{cases}$$

where $\overline{q} = q_1, \ldots, q_k$. ◊

3.3.12 Example (S6) Permutation.
Let $\pi : \{1, \ldots, k\} \to \{1, \ldots, k\}$ be a permutation, let $\tau = (\tau_1, \ldots, \tau_k \to \mathbf{N})$, $\sigma = (\tau_{\pi(1)}, \ldots, \tau_{\pi(k)} \to \mathbf{N})$, and $\rho = (\tau, \tau_{\pi(1)}, \ldots, \tau_{\pi(k)} \to \mathbf{N})$. Let the strategy $p_\pi \in \mathsf{SeqStr}^\rho$ be given by

$$\begin{array}{lll}
p_\pi(?\varepsilon) & = ?1 & \\
p_\pi(s \cdot !v) & = !v & \\
p_\pi(s \cdot ?1.j.a) & = ?(\pi(j){+}1).a & \text{for } 1 \leq j \leq k \\
p_\pi(s \cdot ?(j{+}1).a) & = ?\pi^{-1}(j).a & \text{for } 1 \leq j \leq k
\end{array}$$

Let $q \in \mathsf{HSF}^\tau$ then $p = p_\pi[q] \in \mathsf{SeqStr}^\sigma$ and for all $q_i \in \mathsf{HSF}^{\tau_i}$ $(1 \leq i \leq k)$ holds

$$p[q_1, \ldots, q_k] = q[q_{\pi(1)}, \ldots, q_{\pi(k)}]$$ ◊

3.3.13 Example (S7) Function-application.
Let $\tau_1 = (\mathbf{N}^l \to \mathbf{N})$ and $\tau = (\tau_1, \mathbf{N}^l, \tau_{l+2}, \ldots, \tau_k \to \mathbf{N})$, where $\mathbf{N}^l$ stands for $\mathbf{N}, \ldots, \mathbf{N}$ with l occurrences of $\mathbf{N}$. Let the strategy $p_{fnappl} \in \mathsf{SeqStr}^\tau$ be given by

$$\begin{array}{lll}
p_{fnappl}(?\varepsilon) & = ?1 & \\
p_{fnappl}(s \cdot !v) & = !v & \\
p_{fnappl}(s \cdot ?1.j) & = ?(j{+}1) & \text{for } 1 \leq j \leq k
\end{array}$$

Then for all $q_i \in \mathsf{HSF}^{\tau_i}$ $(1 \leq i \leq k)$ holds

$$p_{fnappl}[q_1, \ldots, q_k] = q_1[q_2, \ldots, q_{l+1}].$$ ◊

3.3.14 Example (S8) Functional-application.
Let $\tau = (\tau_1, \ldots, \tau_k \to \mathbf{N})$ and $\sigma = (\sigma_1, \ldots, \sigma_l, \tau_1, \ldots, \tau_k \to \mathbf{N})$ with $\tau_1 = (\tau_{1.1}, \ldots, \tau_{1.l} \to \mathbf{N})$, $\tau_{1.i} = (\tau_{1.i.1}, \ldots, \tau_{1.i.l_i} \to \mathbf{N})$ and $\sigma_i = (\tau_1, \ldots, \tau_k, \tau_{1.i.1}, \ldots, \tau_{1.i.l_i} \to \mathbf{N})$ for $1 \leq i \leq l$. Let the strategy $p_{\mathit{flappl}} \in \mathsf{SeqStr}^\sigma$ be given by

$$\begin{array}{lll}
p_{\mathit{flappl}}(?\varepsilon) & = ?(l{+}1) & \\
p_{\mathit{flappl}}(s \cdot !v) & = !v & \\
p_{\mathit{flappl}}(?\varepsilon \cdot ?(l{+}1) \cdot ?(l{+}1).i) & = ?i & \\
p_{\mathit{flappl}}(s \cdot ?i.j.a) & = ?(l{+}j).a & \text{for } 1 \leq j \leq k \\
p_{\mathit{flappl}}(s \cdot ?i.j.b \cdot ?(l{+}j).b \cdot ?(l{+}j).a) & = ?i.j.a & \text{for } 1 \leq j \leq k \\
p_{\mathit{flappl}}(s \cdot ?i.(k{+}j).a) & = ?(l{+}1).i.j.a & \text{for } 1 \leq j \leq l_i \\
p_{\mathit{flappl}}(s \cdot ?i.(k{+}j).b \cdot ?(l{+}1).i.j.b \cdot ?(l{+}1).i.j.a) & = ?i.(k{+}j).a & \text{for } 1 \leq j \leq l_i
\end{array}$$

Let $p_i \in \mathsf{SeqStr}^{\sigma_i}$ $(1 \leq i \leq l)$ then $p = p_{\mathit{flappl}}[p_1, \ldots, p_l] \in \mathsf{SeqStr}^{\tau_1}$ and for all $q_i \in \mathsf{HSF}^{\tau_i}$ $(1 \leq i \leq k)$ holds

$$p[q_1, \ldots, q_k] = q_1[p_1[q_1, \ldots, q_k], \ldots, p_l[q_1, \ldots, q_k]]. \qquad \diamondsuit$$

3.3.15 Example (S11) Recursion.
Let $\tau = (\tau_1, \ldots, \tau_k \to \mathbf{N})$, $\sigma = (\tau, \tau_1, \ldots, \tau_k \to \mathbf{N})$ and $\rho = (\sigma, \tau_1, \ldots, \tau_k \to \mathbf{N})$. Let the strategy $p_{rec} \in \mathsf{SeqStr}^\rho$ be given by

$$\begin{array}{lll}
p_{rec}(?\varepsilon) & = ?1 & \\
p_{rec}(s \cdot ?1.1.j.a) & = ?1.(j{+}1).a & \text{for } 1 \leq j \leq k \\
p_{rec}(s \cdot ?1.1.j.b \cdot ?1.(j{+}1).b \cdot ?1.(j{+}1).a) & = ?1.1.j.a & \text{for } 1 \leq j \leq k \\
p_{rec}(s \cdot ?(j{+}1).j'.a) & = ?1.(j'{+}1).a & \text{for } 1 \leq j, j' \leq k \\
p_{rec}(s \cdot ?(j{+}1).j'.b \cdot ?1.(j'{+}1).b \cdot ?1.(j'{+}1).a) & = ?(j{+}1).j'.a & \text{for } 1 \leq j \leq k \\
p_{rec}(s \cdot !v) & = !v &
\end{array}$$

Let $q \in \mathsf{HSF}^\sigma$ then $p = p_{rec}[q] \in \mathsf{SeqStr}^\tau$ and for all $q_i \in \mathsf{HSF}^{\tau_i}$ $(1 \leq i \leq k)$ holds

$$p[q_1, \ldots, q_k] = q[p, q_1, \ldots, q_k]. \qquad \diamondsuit$$

3.4 The category of sequential strategies

If we consider types as finite trees, then we arrive at a cartesian closed category if we expand this idea to finite forests.

For that purpose we extend our notion of type symbols to include also products:

3.4.1 Definition. Let Γ be a set of groundtype symbols. The set of *type symbols with products* $\mathsf{TS}^\times$ and their *level* is defined inductively by

- $\Gamma \subseteq \mathsf{TS}^{\times}$. These are type symbols of level 0.
- If $\tau_1, \tau_2 \in \mathsf{TS}^{\times}$ then $\tau_1 \times \tau_2 \in \mathsf{TS}^{\times}$. The level of $\tau_1 \times \tau_2$ is the maximum of the levels of τ_1 and τ_2.
- If $\tau_1, \tau_2 \in \mathsf{TS}^{\times}$ then $\tau_1 \Rightarrow \tau_2 \in \mathsf{TS}^{\times}$. The level of $\tau_1 \Rightarrow \tau_2$ is the maximum of the level of τ_1 plus one and the level of τ_2.

$\diamondsuit$

3.4.2 Remark. To get really an inclusion of TS in $\mathsf{TS}^{\times}$ we identify $(\tau_1, \ldots, \tau_k \to \gamma)$ with $\tau_1 \times \cdots \times \tau_k \Rightarrow \gamma$. $\diamondsuit$

The extension of the indexing convention of occurrences is as follows. As we are now dealing with forests instead of trees we have to indicate the tree too. So an occurrence a in a single tree will now be named $(a)_1$. If we have a forest of k trees, i.e. the type is a product of k factors $\tau = \tau_1 \times \cdots \times \tau_k$, then an occurrence a in τ_i will be named $(a)_i$.

3.4.3 Definition. The only occurrence of γ in $\tau = \gamma$ is $(\varepsilon)_1$. If γ is an occurrence in τ_1 with identification $(a)_i$, then the corresponding occurrence in $\tau = \tau_1 \times \tau_2$ is $(a)_i$ too. If γ is an occurrence in τ_2 with identification $(a)_i$, and τ_1 consists of k trees (factors) then the corresponding occurrence in $\tau = \tau_1 \times \tau_2$ is $(a)_{(i+k)}$. If γ is an occurrence in τ_2 with identification $(\varepsilon)_i$, and then the corresponding occurrence in $\tau = \tau_1 \Rightarrow \tau_2$ is $(\varepsilon)_i$ too. If γ is an occurrence in τ_2 with identification $(j.a)_i$, and τ_1 consists of k trees (factors) then the corresponding occurrence in $\tau = \tau_1 \Rightarrow \tau_2$ is $((j{+}k).a)_i$. If γ is an occurrence in τ_1 with identification $(a)_i$, and τ_2 consists of k trees (factors) then there are k-many corresponding occurrences in $\tau = \tau_1 \Rightarrow \tau_2$ with identifications $(i.a)_j$ with $1 \leq j \leq k$ respectively. $\diamondsuit$

3.4.4 Remark. The preceding definition is such that the corresponding occurrences in $\tau_1 \Rightarrow (\tau_2 \Rightarrow \tau_3)$ and in $(\tau_1 \times \tau_2) \Rightarrow \tau_3$ have exactly the same names. In other words, the two trees (or forests in general) are exactly the same. For this reason currying and uncurrying will just be the identity. $\diamondsuit$

Composition.

For composition we generalize the strategy p_{compos} from Example 3.3.10 to a strategy $p_{;}$ of type $((\tau \Rightarrow \rho) \times (\sigma \Rightarrow \tau)) \Rightarrow (\sigma \Rightarrow \rho)$. Let k_1, k_2 and k_3 be the number of factors (trees) of ρ, τ and σ respectively and $k = k_1 + k_2 + k_3$. The strategy

$p_; \in \mathsf{SeqStr}^{((\tau\Rightarrow\rho)\times(\sigma\Rightarrow\tau))\Rightarrow(\sigma\Rightarrow\rho)}$ is given by

$$\begin{array}{lcl}
p_;(?(\varepsilon)_i) & = & ?(i)_i \\
p_;(s\cdot ?(i.(k_2{+}i').a)_i) & = & ?((k{+}i').a)_i \\
p_;(s\cdot ?((k{+}i').a)_i) & = & ?(i.(k_2{+}i').a)_i \\
p_;(s\cdot ?(i.j)_i) & = & ?((k_1{+}j))_i \\
p_;(s\cdot ?(i.j.j'.a)_i) & = & ?((k_1{+}j).(k_3{+}j').a)_i \\
p_;(s\cdot ?((k_1{+}j).(k_3{+}j').a)_i) & = & ?(i.j.j'.a)_i \\
p_;(s\cdot ?((k_1{+}j).l.a)_i & = & ?((k_1{+}k_2{+}l).a)_i \\
p_;(s\cdot ?((k_1{+}k_2{+}l).a)_i & = & ?((k_1{+}j).l.a)_i \\
p_;(s\cdot !v) & = & !v
\end{array}$$

3.4.5 Lemma. *Let $p_;$ defined as above and let $p_1 \in \mathsf{SeqStr}^{\sigma\Rightarrow\tau}$ and $p_2 \in \mathsf{SeqStr}^{\tau\Rightarrow\rho}$ then $p = p_;[p_2, p_1] \in \mathsf{SeqStr}^{\sigma\Rightarrow\rho}$ and for all $q \in \mathsf{SeqStr}^{\sigma}$ holds*

$$p[q] = p_1[p_2[q]].$$

We denote p defined in this way by $p_1; p_2$. □

3.4.6 Lemma. *Let $p \in \mathsf{SeqStr}^{\tau\Rightarrow\tau'}$, $p' \in \mathsf{SeqStr}^{\tau'\Rightarrow\tau''}$ and $p'' \in \mathsf{SeqStr}^{\tau''\Rightarrow\tau'''}$ then*

$$p; (p'; p'') = (p; p'); p''.$$ □

Identity.

The identity map of a type τ is the "copy-cat strategy" in $\tau \Rightarrow \tau$. If τ consists of k trees (factors) then the identity strategy is defined by

$$\begin{array}{lcl}
id(?(\varepsilon)_i) & = & ?(i)_i \\
id(s\cdot ?(i.j.a)_i) & = & ?((j{+}k)).a)_i \\
id(s\cdot ?((j{+}k).a)_i) & = & ?(i.j.a)_i \\
id(s\cdot !v) & = & !v
\end{array}$$

with $1 \le i \le k$ and $j \ge 1$.

3.4.7 Lemma. *The strategy id is indeed an identity, i.e. for all strategies $p \in \mathsf{SeqStr}^{\sigma\Rightarrow\tau}$ and all strategies $q \in \mathsf{SeqStr}^{\tau\Rightarrow\rho}$ we have*

$$p; id = p \qquad \text{and} \qquad id; q = q.$$ □

Definition of the category.

Now we are in the position to define the category. The category of sequential strategies has as objects SeqStr^{τ} with $\tau \in \mathsf{TS}^{\times}$ And the morphisms from SeqStr^{τ} to SeqStr^{σ} are the sequential strategies in $\mathsf{SeqStr}^{\tau\Rightarrow\sigma}$.

Terminal object.

The terminal object 1 is the empty product, i.e. no tree at all. There are no answers and no questions. So the only strategy is the empty strategy (considered as a graph).

3.4.8 Lemma. *The object* 1 *is a terminal object in the category of sequential strategies.*

Proof. For each $\tau \in \mathsf{TS}^{\times}$ holds $\tau \times 1 = 1 \times \tau = \tau$. Moreover $\tau \Rightarrow 1 = 1$ hence there is exactly one morphism from τ to 1. On the other hand $1 \Rightarrow \tau = \tau$ hence this category has enough points. □

Products.

The projections $\pi_1 : \sigma \times \tau \Rightarrow \sigma$ and $\pi_2 : \sigma \times \tau \Rightarrow \tau$ are again "copy-cat strategies". Let k_1 and k_2 be the number of factors (trees) of σ and τ respectively and be $k = k_1 + k_2$.

$$\begin{array}{lcll}
\pi_1(?(\varepsilon)_i) & = & ?(i)_i & (1 \le i \le k_1), \\
\pi_1(s \cdot ?(i.j.a)_i) & = & ?((k{+}j).a)_i & (1 \le i \le k_1, j \ge 1, |a| \text{ even}), \\
\pi_1(s \cdot ?((k{+}j).a)_i) & = & ?(i.j.a)_i & (1 \le i \le k_1, j \ge 1, |a| \text{ odd}), \\
\pi_1(s \cdot !v) & = & !v. &
\end{array}$$

and

$$\begin{array}{lcll}
\pi_2(?(\varepsilon)_i) & = & ?(k_1{+}i)_i & (1 \le i \le k_1), \\
\pi_2(s \cdot ?((k_1{+}i).j.a)_i) & = & ?((k{+}j).a)_i & (1 \le i \le k_1, j \ge 1, |a| \text{ even}), \\
\pi_2(s \cdot ?((k{+}j).a)_i) & = & ?((k_1{+}i).j.a)_i & (1 \le i \le k_1, j \ge 1, |a| \text{ odd}), \\
\pi_2(s \cdot !v) & = & !v. &
\end{array}$$

3.4.9 Lemma. *The object $\sigma \times \tau$ is the categorical product of σ and τ in the category of sequential strategies.*

Proof. For the universal property we note that $\rho \Rightarrow (\sigma \times \tau)$ and $(\rho \Rightarrow \sigma) \times (\rho \Rightarrow \tau)$ are the same. So given strategies $p : \rho \Rightarrow \sigma$ and $q : \rho \Rightarrow \tau$ the pairing $[p, q] : \rho \Rightarrow (\sigma \times \tau)$ is defined by the "disjoint" union of p and q in the following way. Let k be the number of factors in σ then all questions $?(a)_i$ occurring in the decision tree of q are substituted by $?(a)_{i+k}$. The strategy p is taken unchanged. □

Function space.

The function space from SeqStr^{τ} to SeqStr^{σ} is $\mathsf{SeqStr}^{\tau \Rightarrow \sigma}$. Since the trees for $(\rho \times \tau) \Rightarrow \sigma$ and $\rho \Rightarrow (\sigma \Rightarrow \tau)$ are exactly the same, currying and uncurrying is just the identity. Hence the strategy $eval_{\sigma,\tau} \in \mathsf{SeqStr}^{(\sigma \Rightarrow \tau) \times \sigma \Rightarrow \tau}$ that makes $\mathsf{SeqStr}^{\sigma \Rightarrow \tau}$ to an categorical exponent is the identity on $\mathsf{SeqStr}^{\sigma \Rightarrow \tau}$.

Order enrichment

3.4.10 Lemma. *For all* $\tau \in \mathsf{TS}$ *the poset* $(\mathsf{SeqStr}^\tau, \subseteq)$ *is a* dI*-domain.* □

All together we have shown:

3.4.11 Theorem. *The category* SeqStr *is cartesian closed and* dI*-enriched.* □

3.5 A PCF-like syntax for decision trees

We first define an extension of PCF by an case-construct called PCFC:

- For each groundtype γ and each $k \geq 0$, if $M_0, \ldots, M_k$ are terms of type γ and N is a term of type $\mathtt{N}$ then $\mathsf{case}_k^\gamma\, N\, M_0 \ldots M_k$ is a term of type γ.

The operational semantics of PCFC is obtained from that of PCF by adding

$$\frac{N \Downarrow \mathsf{k}_j \qquad M_j \Downarrow C}{\mathsf{case}_k^\gamma\, N\, M_0 \ldots M_k \Downarrow C}$$

We identify some of the PCFC-terms as special forms that will be shown in a 1-1-correspondence to parts of finite decision trees:

3.5.1 Definition. A *special form* T is a PCFC-term of groundtype of one of the following shapes:

- $T = \Omega$.
- $T = \mathsf{k}_v$.
- $T = \mathsf{case}_k N\, T_1 \ldots T_l$ where T_i are special forms and either $N = X^\gamma$ or $N = X^\tau\, \lambda\overline{X}^1.T_1' \ldots \lambda\overline{X}^k.T_k'$ where T_i' are special forms. ◊

Now we define a translation from finite decision trees to closures of special forms.

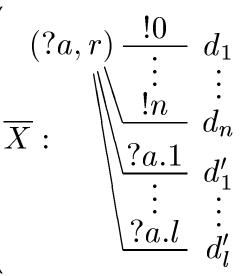
3.5.2 Definition. Let $p \in \mathsf{SeqStr}^\tau$ be finite and let $d = \frac{?\varepsilon}{}d'$ be the corresponding decision tree. Then:

$$\mathsf{pcfc}(p) = \lambda\overline{X}.\mathsf{sf}(\overline{X} : d')$$

where the translation of a part of a finite decision tree in the environment $\overline{X}$ is inductively given by:

$$\begin{aligned}\mathsf{sf}(\overline{X} : \emptyset) &= \Omega,\\ \mathsf{sf}(\overline{X} : !v) &= \mathsf{k}_v \qquad (v \in V),\end{aligned}$$

$\mathsf{sf}\left(\overline{X} :\right.$



$\left.\right) = \mathsf{case}_n\, N\, M_1 \ldots M_n$

where

$$N = (\overline{X}^{[a:r]}\, \lambda\overline{X}^{a.1}\mathsf{sf}(\overline{X}^{a.1}, \overline{X} : d'_1) \ldots \lambda\overline{X}^{a.l}\mathsf{sf}(\overline{X}^{a.l}, \overline{X} : d'_l)$$

and

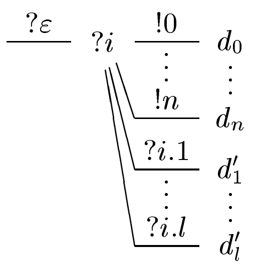
$$M_j = \mathsf{sf}(\overline{X} : d_j). \qquad \diamondsuit$$

This correspondence reflects application too:

3.5.3 Lemma. *Let $\tau = (\tau_1, \ldots, \tau_k \to \gamma)$ and let p, q_1, ..., q_k be finite sequential strategies of type τ, τ_1, ..., τ_k respectively. Then*

$$p[q_1, \ldots, q_k] = v \qquad \Longleftrightarrow \qquad \mathsf{pcfc}(p)\,\mathsf{pcfc}(q_1^{(1)}) \ldots \mathsf{pcfc}(q_k^{(k)}) \Downarrow \mathsf{k}_v.$$

Proof. Let p be a finite sequential strategy of type $\tau = (\tau_1, \ldots, \tau_k \to \gamma)$ of the form



and let $q_i \in \mathsf{SeqStr}^{\tau_i}$ for $1 \leq i \leq k$. Then $\mathsf{pcfc}(p) = \lambda\overline{X}.\mathsf{case}_n\, N\, M_0 \ldots M_n$ with

$$\begin{aligned}N &= (X_i\, N_1 \ldots N_l)\\ N_j &= \lambda\overline{X}^{i.j}\mathsf{sf}(\overline{X}^{i.j}, \overline{X} : d'_j)\\ M_j &= \mathsf{sf}(\overline{X} : d_0)\end{aligned}$$

By Lemma 3.3.6 $p[q_1, \ldots, q_k] = v$ if and only if there exists a $0 \leq j \leq n$ such that

$$q_i[\widehat{p_{i.1}}[q_1, \ldots, q_k], \ldots, \widehat{p_{i.l}}[q_1, \ldots, q_k]] = j \quad \text{and} \quad p_j[q_1, \ldots, q_k] = v$$

Note that

$$\begin{array}{lllll} \mathsf{pcfc}(\widehat{p_{i.j}}) & = & \lambda\overline{X}.N_j & = & \lambda\overline{X},\lambda\overline{X}^{i.j}.\mathsf{sf}(\overline{X}^{i.j}\overline{X} : d'_j) \\ \mathsf{pcfc}(p_j) & = & \lambda\overline{X}.M_j & = & \lambda\overline{X}.\mathsf{sf}(\overline{X} : d_0). \end{array}$$

Hence by induction hypothesis we get $p[q_1,\ldots,q_k] = v$ if and only if there exists a $0 \le j \le n$ such that

$$\mathsf{pcfc}(q_i^{(i)})\,(\mathsf{pcfc}(\widehat{p_{i.1}})\mathsf{pcfc}(q_1^{(1)})\ldots\mathsf{pcfc}(q_k^{(k)}))\ldots(\mathsf{pcfc}(\widehat{p_{i.l}})\mathsf{pcfc}(q_1^{(1)})\ldots\mathsf{pcfc}(q_k^{(k)}))\Downarrow \mathsf{k}_j$$

and

$$\lambda\overline{X}.M_j\,\mathsf{pcfc}(q_1^{(1)})\ldots\mathsf{pcfc}(q_k^{(k)})\Downarrow \mathsf{k}_v.$$

This is by the definition of the case-construct equivalent to

$$\mathsf{pcfc}(p)\,\mathsf{pcfc}(q_1^{(1)})\ldots\mathsf{pcfc}(q_k^{(k)})\Downarrow \mathsf{k}_v.$$ □

3.6 Reduction to pure types

In this section we show that all types over **N** can be reduced to the pure types over **N** at the level of strategies. That means for each $\tau \in \mathsf{TS}^{\times}$ SeqStr^{τ} is a retract of $\mathsf{SeqStr}^{\underline{n}}$ for some $n \in \mathbb{N}$.

3.6.1 Definition. SeqStr^{σ} is a retract of SeqStr^{τ} if there exist strategies $p \in \mathsf{SeqStr}^{\sigma\Rightarrow\tau}$ and $q \in \mathsf{SeqStr}^{\tau\Rightarrow\sigma}$ such that

$$p;q = id^{\sigma} \qquad \text{and} \qquad q;p \subseteq id^{\tau}.$$

In this case we write $\tau \rhd \sigma$ or $\tau \rhd^p_q \sigma$. ◊

First we show that each pure type $\underline{n}$ is a retract of $\underline{m}$ for $n \le m$.

3.6.2 Lemma. *$\underline{n} \rhd \underline{m}$ for $n \le m \in \mathbb{N}$.*

Proof. Let the decision tree of $p \in \mathsf{SeqStr}^{\underline{n}\Rightarrow\underline{m}}$ and that of $q \in \mathsf{SeqStr}^{\underline{m}\Rightarrow\underline{n}}$ both be the decision tree of $id^{\underline{n}}$. Then

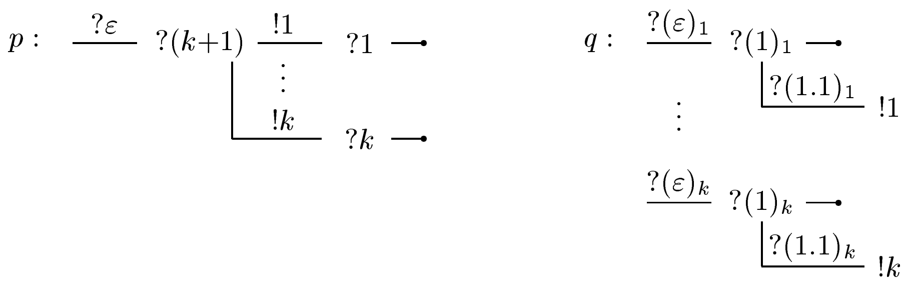
$$p;q = id^{\underline{n}} \qquad \text{and} \qquad q;p = id^{\underline{n}} \subseteq id^{\underline{m}}.$$

Hence $\underline{n} \rhd^p_q \underline{m}$. □

Next we show how to embed a product of pure types into the next higher pure type.

3.6.3 Lemma. *$\underline{n} \times \cdots \times \underline{n} \rhd \underline{n{+}1}$ for all $n \in \mathbb{N}$.*

Proof. Suppose $n = 0$ and let k be the number of factors in $\underline{0}\times\cdots\times\underline{0}$ then the following strategies p and q may be used:



Now suppose $\underline{n} \times \cdots \times \underline{n} \rhd \underline{n{+}1}$ then for $\underline{n{+}1} \times \cdots \times \underline{n{+}1}$ holds

$$\begin{aligned}
\underline{n{+}1} \times \cdots \times \underline{n{+}1} &= (\underline{n} \Rightarrow \underline{0}) \times \cdots \times (\underline{n} \Rightarrow \underline{0}) \\
&\simeq \underline{n} \Rightarrow (\underline{0} \times \cdots \times \underline{0}) \\
&\rhd \underline{n} \Rightarrow \underline{1} \\
&= \underline{n} \Rightarrow (\underline{0} \Rightarrow \underline{0}) \\
&\simeq (\underline{n} \times \underline{0}) \Rightarrow \underline{0} \\
&\rhd (\underline{n} \times \underline{n}) \Rightarrow \underline{0} \\
&\rhd \underline{n{+}1} \Rightarrow \underline{0} \\
&= \underline{n{+}2}
\end{aligned}$$

□

3.6.4 Lemma. *Let τ be a type over $\mathbf{N}$. Then there exists $n \in \mathbb{N}$ with $\tau \rhd \underline{n}$.*

Proof. For $\tau = \mathbf{N} = \underline{0}$ there is nothing to prove. Now let $\tau = (\tau_1, \ldots, \tau_k \to \mathbf{N})$. By induction hypothesis there are $n_1, \ldots, n_k \in \mathbb{N}$ with $\tau_i \rhd \underline{n_i}$ for $i \leq k$. Let $m = \max(n_1, \ldots, n_k)$, then

$$\begin{aligned}
(\tau_1, \ldots, \tau_k \to \mathbf{N}) &= \tau_1 \times \cdots \times \tau_k \Rightarrow \underline{0} \\
&\rhd \underline{n_1} \times \cdots \times \underline{n_k} \Rightarrow \underline{0} \\
&\rhd \underline{m} \times \cdots \times \underline{m} \Rightarrow \underline{0} \\
&\rhd \underline{m+1} \Rightarrow \underline{0} \\
&= \underline{m+2}.
\end{aligned}$$

□

3.7 Variants of SeqStr

In this section we describe which restrictions on the sequential strategies correspond to various subsystems of PCFC. For what follows we assume only one ground type γ with

$V^\gamma = \{0, \dots, k-1\}$ or $V^\gamma = \mathbb{N}$. To simplify notation we write κ instead of Γ in $\lambda Y \Gamma K$ with $\kappa = k$ or $\kappa = \omega$ respectively. Next we consider variants with Y, with Ω or without these two. Note that Y is only needed in case of $\kappa = \omega$. Moreover Ω is definable from Y. For the systems without Y and Ω the standard model for $\kappa < \omega$ is the full type hierarchy over the finite set V^γ. In the other cases it is the full monotone hierarchy over $V^\gamma_\perp$ for $\kappa < \omega$ and the partial continuous hierarchy over $V^\gamma_\perp$ for $\kappa = \omega$.

- $\mathsf{PCF} = \mathsf{PCF}_\omega = \lambda Y \omega \mathsf{KC}$: Constants k_i, Y-terms, case_n-terms for all n. In the game model this corresponds to all sequential strategies.

- $\mathsf{PCF}_k = \lambda \Omega k \mathsf{KC}$: Constants k_i for $i < k$, Ω-terms, case_k-terms for all $n < k$. In the game model this corresponds to no restriction on the sequential strategies. For $k = 2$ this is called finitary PCF (FPCF) and for $k = 1$ it is called very finitary PCF (VFPCF).

- $\lambda \Omega \kappa \mathsf{K}$: Constants k_i for $i < \kappa$, Ω-terms. In the game model this corresponds to the following restriction on the sequential strategies. If $p(s)$ is defined and the view s ends in an answer $!v$ then $p(s) = !v$.

- $\lambda \kappa \mathsf{K}$: Constants k_i for $i < \kappa$. In the game model this corresponds to the following restriction on the sequential strategies. The strategy p is maximal with respect to $\subseteq$ and if $p(s)$ is defined and the view s ends in an answer $!v$ then $p(s) = !v$.

- $\lambda \Omega \kappa$: No constants, but Ω-terms. In the game model this corresponds to the following restriction on the sequential strategies. If $p(s)$ is defined and the view s ends in an answer $!v$ then $p(s) = !v$, if $p(s)$ is defined and the view s ends in a question then $p(s)$ is a question too.

- $\lambda \kappa$: No constants, no Ω-terms. In the game model this corresponds to the following restriction on the sequential strategies. The strategy p is maximal with respect to $\subseteq$. If $p(s)$ is defined and the view s ends in an answer $!v$ then $p(s) = !v$, and if $p(s)$ is defined and the view s ends in a question then $p(s)$ is a question too.

Chapter 4

Hereditarily Sequential Functionals

In this chapter we study the input/output behaviour or extent of the sequential strategies. In Section 4.1 we introduce the hereditarily sequential functionals. The rules of the game are designed such that the value of a play for $p[q_1, \ldots, q_k]$ and $p[q', \ldots, q'_k]$ is the same if q_i and q'_i have the same extent (see Theorem 4.1.3). So this kind of extensionality is directly achieved by the rules and is not an additional side condition as in Kleene's work of oracles for unimonotone functionals where 'only those oracles get a license that respect extensionality'.

In Section 4.2 we give an interpretation of the strategies in the type structure of the partial continuous functionals, called sequential functionals. In Section 4.3 we analize the connection between these two interpretations.

In Section 4.4 we study the order structure of the hereditarily sequential functionals ordered by the extensional ordering. Up to now we don't know if it is a *cpo* or not.

4.1 Definition of HSF

For each $p \in \mathsf{SeqStr}^\tau$ we define a functional F_p determined by its behaviour in playing against strategies of the argument types. In this case we call p a *sequential strategy for F_p*. The functionals F that possess sequential strategies are called *hereditarily sequential*. We write HSF^τ for the set of hereditarily sequential functionals of type τ and $\mathsf{HSF} = \bigcup_{\tau \in \mathsf{TS}} \mathsf{HSF}^\tau$.

4.1.1 Definition. For $\tau = \gamma$ there is exactly one P-view $?\varepsilon$. There is the undefined strategy $\bot$, and for each $v \in V_\gamma$ the strategy $\tilde{v}$ giving the final answer $!v$. We set

$\mathsf{HSF}^\gamma = (V_\gamma)_\perp$ with $F_\perp = \perp$ and $F_{\tilde{v}} = v$.

For $\tau = (\tau_1, \dots, \tau_k \to \gamma)$ we define $F_p : \mathsf{HSF}^{\tau_1} \times \cdots \times \mathsf{HSF}^{\tau_k} \to \omega_\perp$ by

$$F_p(G_1, \dots, G_k) = p[p_1, \dots, p_k]$$

with $F_{p_i} = G_i$ for all $i = 1, \dots, k$. We set

$$\mathsf{HSF}^\tau = \{F_p : \mathsf{HSF}^{\tau_1} \times \cdots \times \mathsf{HSF}^{\tau_k} \to \omega_\perp \mid p \in \mathit{SeqStr}^\tau\}. \qquad \diamondsuit$$

For well-definedness we have to show that it is independent of the choice of sequential strategies p_i for the G_i. That means we have to prove the following theorem simultaneously with the definition of the hereditarily sequential functionals.

4.1.2 Definition. Two strategies p and p' of type $\tau = (\tau_1, \dots, \tau_k \to \gamma)$ are (*observationally*) *equivalent* $p \sim p'$ if for all $q_i \in \mathsf{SeqStr}^{\tau_i}$ $(1 \le i \le k)$ holds

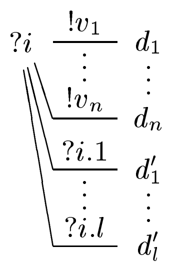
$$p[q_1, \dots, q_k] = p'[q_1, \dots, q_k].$$

For $p, p' \in \mathsf{SeqStr}^\gamma$ we set $p \sim p'$ iff $p = p'$. $\diamondsuit$

4.1.3 Theorem. *Let p be a sequential strategy of type $\tau = (\tau_1, \dots, \tau_k \to \gamma)$ and let $q_i, q_i' \in \mathsf{SeqStr}^{\tau_i}$ for $1 \le i \le k$. If $q_i \sim q_i'$ for all $1 \le i \le k$ then*

$$p[q_1, \dots, q_k] = p[q_1', \dots, q_k'].$$

Proof. By continuity it is enough to consider only the case of a finite strategy p. We prove this theorem by induction on the level of τ and for each level by induction on the size of (the decision tree of) p. For level 1 there is nothing to show. So lets assume that the theorem is true for all levels smaller than that of τ. If $p = \perp$ or $p = \xrightarrow{?\varepsilon} !v$ the theorem is true. Now let p be of the form



Suppose $p[q_1, \dots, q_k] = v$. We use the characterization of Lemma 3.3.6. There exists a $1 \le j \le n$ such that

$$q_i[\widehat{p_{i.1}}[q_1, \dots, q_k], \dots, \widehat{p_{i.l}}[q_1, \dots, q_k]] = v_j \quad \text{and} \quad p_{v_j}[q_1, \dots, q_k] = v$$

where $\widehat{p_{i.j}}$ and p_{v_j} are defined as in the lemma. Note that $\widehat{p_{i.j}}$ is smaller than p. By induction hypothesis we have for all $r_{j'} \in \mathsf{SeqStr}^{\tau_{i.j.j'}}$ with $1 \leq j' \leq l_j$, since $r_{j'} \sim r_{j'}$,

$$\widehat{p_{i.j}}[q_1, \ldots, q_k, r_1, \ldots, r_{l_j}] = \widehat{p_{i.j}}[q'_1, \ldots, q'_k, r_1, \ldots, r_{l_j}].$$

Hence

$$\widehat{p_{i.j}}[q_1, \ldots, q_k] \sim \widehat{p_{i.j}}[q'_1, \ldots, q'_k]$$

The level of τ_i is less than that of τ so we can apply the induction hypothesis for q_i to get:

$$q_i[\widehat{p_{i.1}}[q_1, \ldots, q_k], \ldots, \widehat{p_{i.l}}[q_1, \ldots, q_k]] = q_i[\widehat{p_{i.1}}[q'_1, \ldots, q'_k], \ldots, \widehat{p_{i.l}}[q'_1, \ldots, q'_k]].$$

On the other hand we have by assumption $q_i \sim q'_i$ and hence

$$q_i[\widehat{p_{i.1}}[q'_1, \ldots, q'_k], \ldots, \widehat{p_{i.l}}[q'_1, \ldots, q'_k]] = q'_i[\widehat{p_{i.1}}[q'_1, \ldots, q'_k], \ldots, \widehat{p_{i.l}}[q'_1, \ldots, q'_k]].$$

Together we get

$$v_j = q_i[\widehat{p_{i.1}}[q_1, \ldots, q_k], \ldots, \widehat{p_{i.l}}[q_1, \ldots, q_k]] = q'_i[\widehat{p_{i.1}}[q'_1, \ldots, q'_k], \ldots, \widehat{p_{i.l}}[q'_1, \ldots, q'_k]].$$

Moreover, p_{v_j} is smaller than p and we can apply again the induction hypothesis to get

$$v = p_{v_j}[q_1, \ldots, q_k] = p_{v_j}[q'_1, \ldots, q_k].$$

Hence $p[q'_1, \ldots, q'_k] = v$. By symmetry $p[q'_1, \ldots, q'_k] = v$ implies $p[q_1, \ldots, q_k] = v$ and so

$$p[q_1, \ldots, q_k] = p[q'_1, \ldots, q'_k]. \qquad \square$$

4.1.4 Lemma. *Let* $\tau = (\tau_1, \ldots, \tau_k \to \mathbf{N})$*,* $\sigma = (\tau_2, \ldots, \tau_k \to \mathbf{N})$*,* $F \in \mathsf{HSF}^\tau$ *and* $F_1 \in \mathsf{HSF}^{\tau_1}$*. Then the function* $G : \mathsf{HSF}^{\tau_2} \times \cdots \times \mathsf{HSF}^{\tau_k} \to \mathbb{N}_\perp$ *defined by*

$$G(G_2, \ldots, G_k) = F(F_1, G_2, \ldots, G_k)$$

is hereditarily sequential, i.e. $G \in \mathsf{HSF}^\sigma$*.*

Proof. This follows from Lemma 3.3.1. Let $F = F_p$ and $F_1 = F_{p_1}$, then $q = p[p_1] \in \mathsf{SeqStr}^\sigma$ and for all $q_i \in \mathsf{SeqStr}^{\tau_i}$ $(2 \leq i \leq k)$

$$q[q_2, \ldots, q_k] = p[p_1, q_2, \ldots, q_k],$$

hence $G = F_q \in \mathsf{HSF}^\sigma$. □

4.2 Definition of SF

In this section we consider another possible interpretation of the sequential strategies yielding partial continuous functionals that we call *sequential.* More formally, we define an interpretation map $[\![\cdot]\!] : \mathsf{SeqStr}^\tau \to C^\tau$, where $(C^\tau)_{\tau \in \mathsf{TS}}$ is the Scott-Ershov-hierarchy of the partial continuous functionals over the flat domains $(V_\gamma)_\perp$. The image under $[\![\cdot]\!]$ we call SF^τ. We define this map using the special forms for finite strategies.

4.2.1 Definition. For finite $p \in \mathsf{SeqStr}^\tau$ let $[\![p]\!]$ be the standard interpretation of $\mathsf{pcfc}(p)$.

For arbitrary $p \in \mathsf{SeqStr}^\tau$ we set

$$[\![p]\!] = \bigsqcup\{[\![p']\!] \mid p' \subseteq p,\ p' \text{ finite}\}.$$

The supremum always exists as for all finite p and p' with $p \subseteq p'$ we have $[\![p]\!] \sqsubseteq [\![p']\!]$. In other words the semantics map $[\![\cdot]\!] : \mathsf{SeqStr}^\tau \to C^\tau$ is continuous. $\lozenge$

4.3 Comparing HSF and SF

In this section we study the relation between HSF and SF. First we define a natural inclusion of HSF^τ in SF^τ.

4.3.1 Definition. For each $F \in \mathsf{HSF}^\tau$ we define its *expansion* $\widehat{F} \in \mathsf{SF}^\tau$ inductively over τ.

For $\tau = \gamma$ we have $\mathsf{HSF}^\gamma = \mathsf{SF}^\gamma$ and we set $\widehat{F} = F$.

For $\tau = (\tau_1, \ldots, \tau_k \to \gamma)$ and $F \in \mathsf{HSF}^\tau$ we define $\widehat{F} \in \mathsf{SF}^\tau$ by

$$\widehat{F}(G_1, \ldots, G_k) = v \quad \Longleftrightarrow \quad F(H_1, \ldots, H_k) = v \text{ for some } H_i \in \mathsf{HSF}^{\tau_i}, \widehat{H_i} \sqsubseteq G_i.$$

$\lozenge$

4.3.2 Definition. A strategy $p \in \mathsf{SeqStr}^\tau$ is *strong*, if for each P-view in the domain of p there exist a play $p[q_1, \ldots, q_k]$ in which this view occurs. $\lozenge$

4.3.3 Lemma. *For each $p \in \mathsf{SeqStr}^\tau$ holds $\widehat{F_p} \sqsubseteq [\![p]\!]$. If p is a strong strategy then $\widehat{F_p} = [\![p]\!]$.*

Proof. The first statement follows from the second. For any $p \in \mathsf{SeqStr}^\tau$ there exits a maximal strong substrategy $p' \subseteq p$. Then $F_p = F_{p'}$ and $[\![p']\!] \sqsubseteq [\![p]\!]$. Hence by the second statement we conclude $\widehat{F_p} \sqsubseteq [\![p]\!]$. $\square$

4.3.4 Lemma. *Let p and p_1, ..., p_k be sequential strategies over $\tau = (\tau_1, \ldots, \tau_k \to \gamma)$ and τ_1, ..., τ_k respectively and let $v \in V_\gamma$. Then the following are equivalent:*

1. $F_p(F_{p_1}, \ldots, F_{p_k}) = v$.
2. $p[p_1, \ldots, p_k] = v$.
3. $\widehat{F_p}(\widehat{F_{p_1}}, \ldots, \widehat{F_{p_k}}) = v$.
4. $\widehat{F_p}([\![p_1]\!], \ldots, [\![p_k]\!]) = v$.

5. $[\![p]\!]([\![p_1]\!], \ldots, [\![p_k]\!]) = v$.

6. $\mathsf{pcfc}(p)\,\mathsf{pcfc}(q_1^{(1)}) \ldots \mathsf{pcfc}(q_k^{(k)}) \Downarrow \mathsf{k}_v$.

Proof. "1 ⇔ 2" holds by definition of HSF. "2 ⇔ 6" holds by Lemma 3.5.3. "1 ⇒ 3" holds by definition of $\widehat{F}$. "3 ⇒ 4" and "4 ⇒ 5" follow from the preceding lemma. "5 ⇔ 6" holds by an adaptation of Theorem 2.4.6 to PCFC. □

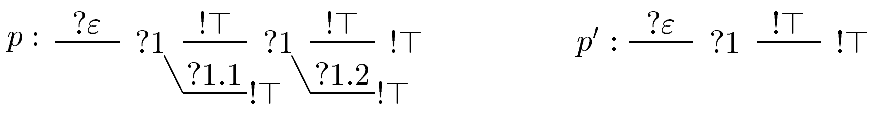
As a consequence we yield the following.

4.3.5 Lemma. *Let p and p' be sequential strategies over $\tau = (\tau_1, \ldots, \tau_k \to \gamma)$. Then $[\![p]\!] = [\![p']\!]$ implies $F_p = F_{p'}$, and $F_p = F_{p'}$ if and only if $[\![p]\!]|_{\mathsf{SF}} = [\![p']\!]|_{\mathsf{SF}}$.* □

Proof. The first statement follows from the second, since $[\![p]\!] = [\![p']\!]$ implies $[\![p]\!]|_{\mathsf{SF}} = [\![p']\!]|_{\mathsf{SF}}$. By definition of SF we have $[\![p]\!]|_{\mathsf{SF}} = [\![p']\!]|_{\mathsf{SF}}$ iff $[\![p]\!]([\![p_1]\!], \ldots, [\![p_k]\!]) = [\![p']\!]([\![p_1]\!], \ldots, [\![p_k]\!])$ for all $p_1 \in \mathsf{SeqStr}^{\tau_1}, \ldots, p_k \in \mathsf{SeqStr}^{\tau_k}$. By the preceding lemma this is the same as $F_p(F_{p_1}, \ldots, F_{p_k}) = F_{p'}(F_{p_1}, \ldots, F_{p_k})$ for all $p_1, \ldots, p_k$. By definition of HSF this is equivalent to $F_p = F_{p'}$. □

So each hereditarily sequential functional can be seen as the restriction of a sequential functional to (hereditarily) sequential arguments, in other words the relation $F_p \mapsto [\![p]\!]|_{\mathsf{SF}}$ defines a bijection between HSF^τ and $\mathsf{SF}^\tau|_{\mathsf{SF}}$. The next example shows that the reverse of the first implication in Lemma 4.3.5 does not hold in general, hence the classes SF and HSF are in fact different.

4.3.6 Example. Let $p, p' \in \mathsf{SeqStr}^{(0,0\to0)\to0}$ be given by the following decision trees:



Then $F_p = F_{p'}$ with $F_p(G) = \top$ iff $G(\bot, \bot) = \top$. But for H with $H(\bot, \top) = H(\top, \bot) = \top$ and $H(\bot, \bot) = \bot$ we have $[\![p]\!](H) = \top$ and $[\![p']\!](H) = \bot$, hence $[\![p]\!] \neq [\![p']\!]$. Note that H is not sequential. ♢

4.4 Structure of HSF

In this section we study the order structure of HSF. Each set HSF^τ is naturally ordered by the pointwise or extensional ordering $\sqsubseteq_e$, defined by:

$$F \sqsubseteq_e F' \iff \forall (F_1, \ldots, F_k) \in \mathsf{dom}(F) : F(F_1, \ldots, F_k) = F'(F_1, \ldots, F_k).$$

4.4.1 Theorem. $(\mathsf{HSF}^\tau, \sqsubseteq_e)$ *is a pointed partial order. The hereditarily sequential functionals are monotone w.r.t. $\sqsubseteq_e$.*

Proof. $(\mathsf{HSF}^\tau, \sqsubseteq_e)$ is trivially a partial order. The minimal point is the totally undefined functional. The monotonicity follows from the proof of Theorem 4.1.3.

$\square$

4.4.2 Lemma. *In $(\mathsf{HSF}^\tau, \sqsubseteq_e)$ the infimum of two elements always exists.*

Proof. Let p_1 and p_2 be sequential strategies over τ. We define a new strategy q over τ. For $i = 1, 2$ let S_{p_i} be the set of all views s such that $p_i(s) = !v$ and $?\varepsilon$ is the only open question in s, i.e. the set of all views for which p_i gives the final answer.

$$\begin{array}{lcll}
q(s) & = & p_1(s) & \text{if } s \notin S_{p_1}, \\
q(s \cdot s') & = & p_2(?\varepsilon \cdot s') & \text{if } s \in S_{p_1} \text{ and } ?\varepsilon \cdot s' \notin S_{p_2}, \\
q(s \cdot s') & = & !v & \text{if } s \in S_{p_1},\ ?\varepsilon \cdot s' \in S_{p_2}, \text{ and } p_1(s) = p_2(?\varepsilon \cdot s') = !v, \\
q(s \cdot s') & = & \bot & \text{if } s \in S_{p_1},\ ?\varepsilon \cdot s' \in S_{p_2}, \text{ and } p_1(s) \neq p_2(?\varepsilon \cdot s').
\end{array}$$

Then $F_q = F_{p_1} \sqcap_e F_{p_2}$. We write $q = p_1 \nabla p_2$. $\square$

4.4.3 Remark. Up to now it is not known if $(\mathsf{HSF}^\tau, \sqsubseteq_e)$ forms a cpo for all $\tau \in \mathsf{TS}$.

$\Diamond$

But for a model of computability it is enough to have all least fixpoints:

4.4.4 Lemma. *Let $\tau = (\tau_1, \dots, \tau_k \to \gamma)$ and $\sigma = (\tau, \tau_1, \dots, \tau_k \to \gamma)$. For each $F \in \mathsf{HSF}^\sigma$ there exists the least fixpoint of the equation*

$$f(x_1, \dots, x_k) = F(f, x_1, \dots, x_k).$$

Proof. Let $F \in \mathsf{HSF}^\sigma$. We define $f^{(n)} \in \mathsf{HSF}^\tau$ in the usual manner by $f^{(0)} = \bot_\tau$ the minimal element in HSF^τ, and $f^{(n+1)}(x_1, \dots, x_k) = F(f^{(n)}, x_1, \dots, x_k)$. We show that the supremum $f = \bigsqcup\{f^{(n)} \mid n \in \mathbb{N}\}$ exists. For that purpose we define strategies $q^{(n)} \in \mathsf{SeqStr}^\tau$ with $F_{q^{(n)}} = f^{(n)}$. Let $p \in \mathsf{SeqStr}^\sigma$ such that $F_p = F$. $q^{(0)} = \bot$ the undefined strategy, and for $n \geq 0$ let $q^{(n+1)} = p[q^{(n)}]$. By Remark 3.3.3 we have

$$q^{(0)} \subseteq q^{(1)} \subseteq q^{(2)} \subseteq \dots$$

Hence the sequence $\{f^{(n)} \mid n \in \mathbb{N}\}$ has as supremum $f = F_q$ with $q = \bigcup\{q^{(n)} \mid n \in \mathbb{N}\}$.

$\square$

The following example shows that the least upper bound of two elements if it exists is not necessarily the pointwise one.

4.4.5 Example. (a) Consider the type $\tau = (0, 0 \to 0)$ and the following two stepfunctions $f_1 = [\top, \bot \to \top]$ and $f_2 = [\bot, \top \to \top]$. Then the supremum of f_1 and f_2 in $(\mathsf{HSF}^\tau, \sqsubseteq_e)$ is $[\bot, \bot \to \top]$.

(b) Consider the type $\tau = ((0, 0 \to 0), 0, 0 \to 0)$ and the following two stepfunctions $f_1 = [[\top, \bot \to \top], \top, \bot \to \top]$ and $f_2 = [[\bot, \top \to \top], \bot, \top \to \top]$. Then the supremum of f_1 and f_2 in $(\mathsf{HSF}^\tau, \sqsubseteq_e)$ is

$$[[\top, \bot \to \top], \top, \bot \to \top] \cup [[\bot, \top \to \top], \bot, \top \to \top] \cup [[\bot, \bot \to \top], \bot, \bot \to \top]$$

◊

The following example shows that the hereditarily functionals in general are not stable, i.e. preserving bounded meets, w.r.t. the extensional ordering:

4.4.6 Example. Let $p \in \mathsf{SeqStr}^{(\gamma, \gamma \to \gamma) \to \gamma}$ and $q, q', q'' \in \mathsf{SeqStr}^{\gamma, \gamma \to \gamma}$ be given by the following decision trees:

$$\begin{tikzcd}
p: \arrow[r, dash, "?\varepsilon"] & ?1 \arrow[r, dash, "!0"] \arrow[dr, dash, "?1.1"] \arrow[dddr, dash, "?1.2"] & !0 & \\
 & & ?1 \arrow[r, dash, "!0"] \arrow[dr, dash, "?1.1"] & !0 \\
 & & & !0 \\
 & & ?1 \arrow[r, dash, "!0"] \arrow[dr, dash, "?1.2"] & !0 \\
 & & & !0
\end{tikzcd}$$

$$\begin{tikzcd}
q: \arrow[r, dash, "?\varepsilon"] & ?1 \arrow[r, dash, "!0"] & !0
\end{tikzcd}$$

$$\begin{tikzcd}
q': \arrow[r, dash, "?\varepsilon"] & ?2 \arrow[r, dash, "!0"] & !0
\end{tikzcd}$$

$$\begin{tikzcd}
q'': \arrow[r, dash, "?\varepsilon"] & ?1 \arrow[r, dash, "!0"] & ?2 \arrow[r, dash, "!0"] & !0
\end{tikzcd}$$

Then F_p, F_q, $F_{q'}$ and $F_{q''}$ are the least hereditarily sequential functionals with $F_p(f) = 0$ if $f(0, \bot) = 0$ or $f(\bot, 0) = 0$, $F_q(0, \bot) = 0$, $F_{q'}(\bot, 0) = 0$ and $F_{q''}(0, 0) = 0$ respectively. Note that $F_q \sqcap_e F_{q'} = F_{q''}$, hence we obtain

$$F_p(F_q \sqcap_e F_{q'}) = F_p(F_{q''}) = \bot \neq 0 = 0 \sqcap_e 0 = F_p(F_q) \sqcap_e F_p(F_{q'})$$

showing that F_p is not stable w.r.t. $\sqsubseteq_e$. ◊

The second ordering is more closely related to the definition of HSF, being the ordering inherited from the sequential strategies. It is called sequential (or observational) ordering $\sqsubseteq_s$.

4.4.7 Definition. The relation $\preceq_s$ on HSF^τ is given by

$$F \preceq_s F' \quad \iff \quad \exists p, p' \in \mathsf{SeqStr}^\tau \text{ with } p \subseteq p', F = F_p, \text{ and } F' = F_{p'}.$$

By $\sqsubseteq_s$ we denote the transitive closure of $\preceq_s$. ◊

4.4.8 Lemma.

1. *The relation $\sqsubseteq_s$ is included in $\sqsubseteq_e$ i.e. for all $F, G \in \mathsf{HSF}^\tau$ if $F \sqsubseteq_s G$ then $F \sqsubseteq_e G$.*
2. *The relation $\preceq_s$ is reflexive and antisymmetric.*
3. *The relation $\sqsubseteq_s$ is a partial order.*

Proof. 1. Let $p, q \in \mathsf{SeqStr}^\tau$ with $p \subseteq q$ ($\tau = (\tau_1, \ldots, \tau_k \to \gamma)$) i.e. $F_p \preceq_s F_q$. And let p_1, …, p_k be strategies over τ_1,…,τ_k respectively. If $F_p(F_{p_1}, \ldots, F_{p_k}) = v$ then $p[p_1, \ldots, p_k] = v$ let's say with play $s = ?\varepsilon \cdots !v$. As $p \subseteq q$ we get the same play s for $q[p_1, \ldots, p_k]$. Hence $F_q(F_{p_1}, \ldots, F_{p_k}) = v$ too and $F_p \sqsubseteq_e F_q$. So $\preceq_s$ is included in $\sqsubseteq_s$. Since $\sqsubseteq_s$ is a partial oreder the transitive closure $\sqsubseteq_s$ of $\preceq_s$ is included in $\sqsubseteq_e$ too.

2. For all $p \in \mathsf{SeqStr}$ is $p \subseteq p$ and hence $F_p \preceq_s F_p$ and $\preceq_s$ is reflexive. Suppose $F \preceq_s G$ and $G \preceq_s F$, then by 1. $F \sqsubseteq_e G$ and $G \sqsubseteq_e F$. Since $\sqsubseteq_e$ is a partial order we conclude $F = G$ and $\preceq_s$ is antisymmetric.

3. Reflexivity of $\sqsubseteq_s$ follows from 2., antisymmetry of $\sqsubseteq_s$ is proven by the same argument as the antisymmetry of $\preceq_s$ in 2., and $\sqsubseteq_s$ is transitive by definition. □

4.4.9 Remark. We do not know if $\preceq_s$ is itself transitive, i.e. if $\preceq_s = \sqsubseteq_s$ ◊

4.4.10 Lemma. *Let $\tau = (\tau_1, \ldots, \tau_k \to \gamma)$, $p \in \mathsf{SeqStr}^\tau$, and for $i \leq k$ let $p_i \subseteq q_i \in \mathsf{SeqStr}^{\tau_i}$ such that $p[p_1, \ldots, p_k] = \bot$ and $p[q_1, \ldots, q_k] = n$. Then $q[p_1, \ldots, p_k] = \bot$ for all $q \supseteq p$.*

Proof. We consider the play for $p[p_1, \ldots, p_k] = \bot$. Three cases are possible:

1. The play is infinite. Then the play for $q[p_1, \ldots, p_k]$ is infinite too, since $q \supseteq p$. Hence $q[p_1, \ldots, p_k] = \bot$.
2. The play is finite and stops with player P in turn. Then the play for $p[q_1, \ldots, q_k]$ stops in the same situation, since $p_i \subseteq q_i$. Hence $p[q_1, \ldots, q_k] = \bot$ in contradiction to the assumption of $p[q_1, \ldots, q_k] = n$. So this case can not occur.
3. The play is finite and stops with player O in turn. Then the play for $q[p_1, \ldots, p_k]$ stops in the same situation, since $p \subseteq q$. Hence $q[p_1, \ldots, p_k] = \bot$. □

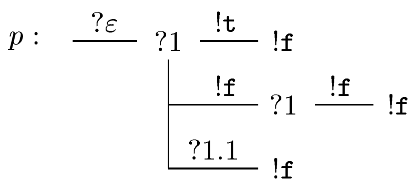
As an immediate consequence we get the following lemma.

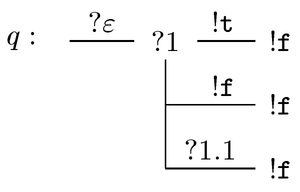
4.4.11 Lemma. *For all $F \sqsubseteq_s G \in \mathsf{HSF}^\tau$ and $x_i \sqsubseteq_s y_i \in \mathsf{HSF}^{\tau_i}$:*

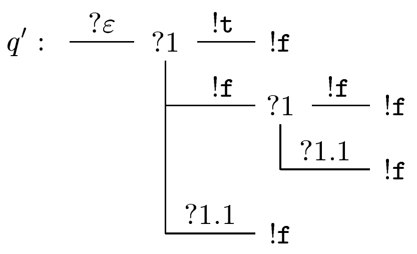
$$F(x_1, \ldots, x_k) = F(y_1, \ldots, y_k) \sqcap G(x_1, \ldots, x_k).$$ □

Let $p, q \in \mathsf{SeqStr}^\tau$ with $F_p \preceq_s F_q$. One could ask, if there always exist $p', q' \in \mathsf{SeqStr}^\tau$ with $F_p = F_{p'}$ and $F_q = F_{q'}$ such that $p' \subseteq q$ and $q' \supseteq p$. The next examples show that neither is the case.

4.4.12 Example. Let $\tau = ((\mathbf{B} \to \mathbf{B}) \to \mathbf{B})$, $F = [f \to f, [f \to f] \to f]$ and $G = [[f \to f] \to f, [f \to t] \to f]$. Let $p, q, q' \in SeqStr^\tau$ be given by the following decision trees:

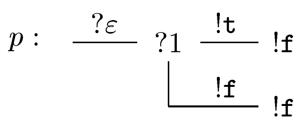


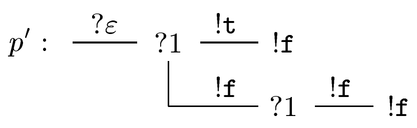


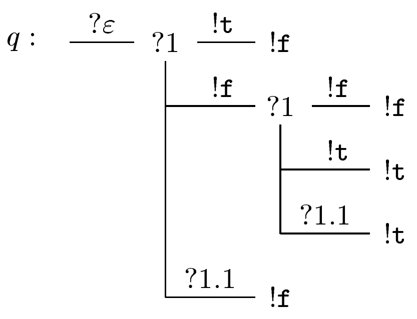


Then $F = F_p$, $G = F_q = F_{q'}$. Moreover $p \subset q'$ and hence $F \preceq_s G$. But there is no $p' \subset q$ with $F_{p'} = F$. $\Diamond$

4.4.13 Example. Let $\tau = ((\mathbf{B} \to \mathbf{B}) \to \mathbf{B})$, $F = [[\bot \to f] \to f, [\bot \to t] \to f]$ and $G = [[f \to f, t \to f] \to f, [f \to f, t \to t] \to t, [f \to t] \to f]$. Let $p, p', q \in SeqStr^\tau$ be given by the following decision trees:







Then $F = F_p = F_{p'}$, $G = F_q$. Moreover $p' \subset q$ and hence $F \preceq_s G$. For all $q' \supset p$ and all $H \in \mathsf{HSF}^{(\mathbf{B} \to \mathbf{B})}$ is $F_{q'}(H) \neq t$. But $G([f \to f, t \to t]) = t$, so there is no $q' \supset p$ with $F_{q'} = G$. $\Diamond$

4.4.14 Lemma. *Let $F, G, H \in \mathsf{HSF}^\tau$. If $F \sqsubseteq_e G \sqsubseteq_e H$ and $F \sqsubseteq_s H$, then $F \sqsubseteq_s G$.*

Proof. Let $F, G, H \in \mathsf{HSF}^\tau$. First we show that $F \preceq_s H$ implies $F \sqcap_e G \preceq_s H \sqcap_e G$. Let $p, p', q \in \mathsf{SeqStr}^\tau$ such that $F = F_p$, $H = F_{p'}$ and $p \subseteq p'$. By the construction in the proof of Lemma 4.4.2 we have $p\nabla q \subseteq p'\nabla q$. Moreover

$$\begin{array}{lclcl} F_{p\nabla q} & = & F_p \sqcap_e F_q & = & F \sqcap_e G, \quad \text{and} \\ F_{p'\nabla q} & = & F_{p'} \sqcap_e F_q & = & H \sqcap_e G. \end{array}$$

Hence $F \sqcap_e G \preceq_s H \sqcap_e G$.

Now suppose $F \sqsubseteq_s H$. Then there exist $F_1, \ldots, F_n \in \mathsf{HSF}^\tau$ with

$$F = F_1 \preceq_s F_2 \preceq_s \ldots \preceq_s F_n = H.$$

By the considerations above we get

$$F \sqcap_e G = F_1 \sqcap_e G \preceq_s F_2 \sqcap_e G \preceq_s \ldots \preceq_s F_n \sqcap_e G = H \sqcap_e G.$$

Hence $F \sqcap_e G \sqsubseteq_s H \sqcap_e G$.

If in addition $F \sqsubseteq_e G \sqsubseteq_e H$ then

$$F = F \sqcap_e G \sqsubseteq_s H \sqcap_e G = G. \qquad \square$$

4.4.15 Lemma. *Let $F, G \in \mathsf{HSF}^\tau$. If their least upper bound w.r.t. $\sqsubseteq_s$, i.e. $F \sqcup_s G$ exists, then the least upper bound w.r.t. $\sqsubseteq_e$ exists too and is the same.*

Proof. Let us assume $H = F \sqcup_s G$ and $F, G \sqsubseteq_e H' \sqsubset_e H$. By the preceding lemma we have $F, G \sqsubseteq_s H'$. Then $H \sqsubseteq_s H'$ since H is the $\sqsubseteq_s$-lub of F and G. By Lemma 4.4.8 $H \sqsubseteq_e H'$ in contradiction with $H' \sqsubset_e H$. $\square$

Next we show that the question whether $(\mathsf{HSF}^\tau, \sqsubseteq_e)$ is a cpo or not can be reduced to the question whether $(\mathsf{HSF}^\tau, \sqsubseteq_s)$ is a cpo or not.

4.4.16 Theorem. *If $(\mathsf{HSF}^\tau, \sqsubseteq_s)$ is a cpo, then $(\mathsf{HSF}^\tau, \sqsubseteq_e)$ is a cpo, too.*

Proof. Suppose $(\mathsf{HSF}^\tau, \sqsubseteq_s)$ is a cpo. Let

$$F_1 \sqsubseteq_e F_2 \sqsubseteq_e \ldots$$

be a chain in $(\mathsf{HSF}^\tau, \sqsubseteq_e)$. Let $F_i^{(n)}$ denote the projection of F_i on $\tau^{(n)}$. Since $\mathsf{HSF}^{\tau^{(n)}}$ is finite there exists an i_n such that $F_i^{(n)} = F_{i'}^{(n)}$ for all $i, i' \geq i_n$. We choose the i_n such that $i_{n+1} > i_n$ for all n.

Now we consider the functionals $F_{i_n}^{(n)}$ as functionals in HSF^τ. By the choice of i_n we get $F_{i_n}^{(n)} = F_{i_{n+1}}^{(n)}$ for all n. On the other hand we know that $F_{i_{n+1}}^{(n)} \sqsubseteq_s F_{i_{n+1}}^{(n+1)}$, hence $F_{i_n}^{(n)} \sqsubseteq_s F_{i_{n+1}}^{(n+1)}$ for all n. Now let $F = \bigsqcup_s \{F_{i_n}^{(n)} \mid n \in \mathbb{N}\}$ be the supremum of this chain in $(\mathsf{HSF}^\tau, \sqsubseteq_s)$. Then F is also the supremum of this chain in $(\mathsf{HSF}^\tau, \sqsubseteq_e)$, i.e. $F = \bigsqcup_e \{F^{(n)}_{i_n} \mid n \in \mathbb{N}\}$.

We conclude the proof by showing that the sets of upper bounds w.r.t. $\sqsubseteq_e$ of $\{F^{(n)}_{i_n} \mid n \in \mathbb{N}\}$ and of $\{F_i \mid i \in \mathbb{N}\}$ coincide. For one direction we have to show that for all n and all $\overline{G} \in \mathsf{HSF}^{\overline{\tau}}$ with $F^{(n)}_{i_n}(\overline{G}) = v$ exists an i such that $F_i(\overline{G}) = v$. Since $F^{(n)}_{i_n}$ is a restriction of F_{i_n} this is true for $i = i_n$. Hence each upper bound of $\{F_i \mid i \in \mathbb{N}\}$ is an upper bound of $\{F^{(n)}_{i_n} \mid n \in \mathbb{N}\}$. For the other direction we have to show that for all i and all $\overline{G} \in \mathsf{HSF}^{\overline{\tau}}$ with $F_i(\overline{G}) = v$ there exists an n such that $F^{(n)}_{i_n}(\overline{G}) = v$. Let $\overline{q}$ be sequential strategies for $\overline{G}$ and p a sequential strategy for F_i. Since $F_i(\overline{G}) = v$ the play t for $p[\overline{q}]$ is finite there exists an m such that t is a play in $\tau^{(m)}$. Hence $F^{(m)}_i(\overline{G}) = v$. Now let n be such that $n \geq m$ and $i_n \geq i$. Then $F^{(m)}_i \sqsubseteq_e F^{(m)}_{i_n} \sqsubseteq_e F^{(n)}_{i_n}$. So $F^{(n)}_{i_n}(\overline{G}) = v$. Hence each upper bound of $\{F^{(n)}_{i_n} \mid n \in \mathbb{N}\}$ is an upper bound of $\{F_i \mid i \in \mathbb{N}\}$. □

4.4.17 Definition. Let $\{F_i \mid i \in \mathbb{N}\}$ be a chain in $(\mathsf{HSF}^\tau, \sqsubseteq_s)$. We call it a *strong chain* iff there is a chain $\{p_i \mid i \in \mathbb{N}\}$ in $(\mathsf{SeqStr}^\tau, \subseteq)$ such that $F_{p_i} = F_i$ for all i. ◊

4.4.18 Lemma.

1. *Every strong chain has a supremum.*

2. *Every element $F \in \mathsf{HSF}^\tau$ is supremum of a strong chain of compact elements.*

Proof. Let $\{F_i \mid i \in \mathbb{N}\}$ be a strong chain in HSF^τ witnessed by $\{p_i \mid i \in \mathbb{N}\}$, i.e. for all i holds $F_i = F_{p_i}$ and $p_i \subseteq p_{i+1}$. Let $p = \bigcup\{p_i \mid i \in \mathbb{N}\}$. We show that $F_p = \bigsqcup_e \{F_{p_i} \mid i \in \mathbb{N}\}$. Let $\overline{q}$ be sequential strategies with $p[\overline{q}] = v$. Then the play for $p[\overline{q}]$ is finite hence there exists an i such that the play for $p_i[\overline{q}]$ is the same. Hence $p_i[\overline{q}] = v$. For the other direction let $p_i[\overline{q}] = v$ then the play for $p[\overline{q}]$ is the same as that of $p_i[\overline{q}]$, hence $p_[\overline{q}] = v$. So we have shown that F_p is the supremum, indeed the pointwise one, of $\{F_{p_i} \mid i \in \mathbb{N}\}$.

For the second part let $p \in \mathsf{SeqStr}^\tau$ be a strategy with $F_p = F$. Since SeqStr^τ is a *dI*-domain there exists a chain $p_0 \subseteq p_1 \subseteq \ldots$ of finite strategies with $p = \bigcup\{p_i \mid i \in \mathbb{N}\}$. Then $F_{p_0} \sqsubseteq_s F_{p_1} \sqsubseteq_s \ldots \sqsubseteq_s F_p$, and $F_{p_0} \sqsubseteq_e F_{p_1} \sqsubseteq_e \ldots \sqsubseteq_e F_p$ too, by Lemma 4.4.8. Moreover $F_p = \bigsqcup_e \{F_{p_i} \mid i \in \mathbb{N}\} = \bigsqcup_s \{F_{p_i} \mid i \in \mathbb{N}\}$. □

4.4.19 Corollary. *If every chain in $(\mathsf{HSF}^\tau, \sqsubseteq_s)$ is a strong chain, then $(\mathsf{HSF}^\tau, \sqsubseteq_s)$ and $(\mathsf{HSF}^\tau, \sqsubseteq_e)$ are cpo-s.*

4.4.20 Remark. If every chain in $(\mathsf{HSF}^\tau, \sqsubseteq_s)$ is a strong chain, then $\preceq_s$ is transitive, i.e. $\sqsubseteq_s = \preceq_s$. Hence the transitivity of $\preceq_s$ is a necessary condition for the assumption of the preceding corollary to hold. ◊

4.5 Full abstraction for PCF

In Section 3.4 we have seen that the sequential strategies form a cartesian closed category that is enriched over dI-domains. Since HSF is the extensional collapse of SeqStr the hereditarily sequential functionals form a cartesian closed category too.

Moreover we have shown that HSF has all least fixpoints and the functions interpreting the PCF-constants in the standard model are hereditarily sequential. Hence we get an adequate model for PCF based on HSF. For full abstraction it remains to show that the compact elements of any HSF^τ are PCF-definable. First of all it is clear that any compact element is definable in PCFC. Let $F \in \mathsf{HSF}^\tau$ be compact, then there exists a finite strategy p with $F_p = F$. Now $\mathsf{pcfc}(p)$ is a defining term for F in PCFC. To get a defining term in PCF we define a translation from PCFC-terms T to PCF terms T^* such that $T \Downarrow C$ iff $T^* \Downarrow C^*$.

$$\begin{array}{rcll}
T^* & = & T & \text{if } T \text{ is a } \mathsf{PCF}\text{-term,} \\
(T\,S)^* & = & (T^*\,S^*) & \\
(\lambda x.T)^* & = & \lambda x.T^* & \\
\mathsf{Y}(T)^* & = & \mathsf{Y}(T^*) & \\
\mathsf{case}(M, N_0 \ldots, N_n)^* & = & \mathsf{cond}(\mathsf{iszero} M^*)\, N_0^*\, (\ldots(\mathsf{cond}(\mathsf{iszero}(\mathsf{pred}^n M^*))\, N_n^*\, \Omega)\ldots) &
\end{array}$$

Chapter 5

Computability

In this chapter we give two definitions of those elements of HSF (SF) that should be considered as computable: the recursive elements of HSF (SF) being extensions of recursive sequential strategies and the Kleene-recursive functionals definable by Kleene's schemata (S1) – (S8), (S11) over HSF (over SF), (Section 5.1). We show that both notions define the same class of computable elements of HSF (SF). For that we proof in Section 5.2 that the recursive sequential functionals are closed under Kleene's schemata and in Section 5.3 that the recursive sequential functionals are definable by the schemata. From the equivalence proof we also get another characterization for the class SF.

5.1 Recursive sequential functionals

By a recursive sequential strategy we mean a sequential strategy that is recursive in the ordinary sense with respect to a suitable coding of views and moves.

5.1.1 Definition. An element $F \in \mathsf{HSF}$ ($\in \mathsf{SF}$) is *recursive*, if there exists a recursive sequential strategy p with $F = F_p$ (with $F = [\![p]\!]$).

We denote the set of recursive sequential functionals by HSF_R and SF_R respectively.

$\Diamond$

5.1.2 Lemma. *Let* $\tau = (\tau_1, \ldots, \tau_k \to \gamma)$*,* $F \in \mathsf{HSF}_R^{\tau}$ *and* $G \in \mathsf{HSF}_R^{\tau_1}$*, then* $F(G) \in \mathsf{HSF}_R^{(\tau_2, \ldots, \tau_k \to \gamma)}$.

Proof. Let p and p_1 be recursive strategies for F and G respectively. Then $q = p[p_1]$ is a recursive strategy for $F(G)$ since the construction of $q = p[p_1]$ in Lemma 3.3.1 is recursive in p and p_1. $\Box$

5.1.3 Definition. An element of HSF (of SF) is called *Kleene-recursive*, if it is definable by Kleene's schemata (S1)–(S8), (S11) over HSF (over SF). We denote the set of Kleene-recursive sequential functionals by HSF_K (SF_K). $\Diamond$

The schemata are listed and explained in Section 2.2.

5.2 Closure under Kleene's schemata

Now we show that each Kleene-recursive (hereditarily) sequential functional is recursive, i.e. $\mathsf{HSF}_K \subseteq \mathsf{HSF}_R$ and $\mathsf{SF}_K \subseteq \mathsf{SF}_R$.

5.2.1 Theorem. *The recursive (hereditarily) sequential functionals are closed under the schemata (S1)–(S8) and (S11).*

Proof. This follows from Lemma 5.1.2 and the fact, that all strategies defined in Examples 3.3.7-3.3.15 are recursive. $\square$

5.3 Definability by Kleene's schemata

Moreover, the reverse is also true, i.e. $\mathsf{HSF}_R \subseteq \mathsf{HSF}_K$ and $\mathsf{SF}_R \subseteq \mathsf{SF}_K$:

5.3.1 Theorem. *The recursive (hereditarily) sequential functionals are definable by the schemata (S1)–(S8) and (S11).*

Proof. We define a Kleene-recursive functional `eval` of type $(\gamma \to \gamma), \tau_1, \ldots, \tau_k \to \gamma$ that is universal for the (hereditarily) sequential functionals over $\tau = \tau_1, \ldots, \tau_k \to \gamma$, i.e. for all $p \in \mathsf{SeqStr}^\tau$ holds $\mathtt{eval}(p^{\ulcorner\urcorner}, \overline{X}) = F_p(\overline{X})$ if X_a ranges over C^{τ_a} and $\mathtt{eval}(p^{\ulcorner\urcorner}, \overline{X}) = [\![p]\!](\overline{X})$ if X_a ranges over HSF^{τ_a} where $p^{\ulcorner\urcorner}$ is p w.r.t. a suitable coding of the Game of Higher Types. So let $\ulcorner \cdot \urcorner$ be a coding for the Game of Higher Types such that there are primitive recursive functions working on the codes to handle with views, plays, moves and so on. Note that all primitive recursive functions (of type $\gamma, \ldots, \gamma \to \gamma$) are definable by the schemata (S1)–(S6).

The idea is to simulate the play with indeterminate opponent. To deal with the various instances of the oracles we need a notion of context where in some sense the enabled questions are stored. The problem is that there is no bound on the number of instances of the oracles. But the type is fixed and hence the number of occurrences a of odd length is finite. For each such occurrence a we define a functional $\mathtt{new\_context\_}a$ to append a new instance X_a to a context C of type τ_a. The empty context will be given by the undefined functional of type $(\gamma, \tau_{a.1}, \ldots, \tau_{a.l} \to \gamma)$

$$\texttt{new\_context\_}a(X_a, C, n, \overline{X}^a) = \begin{cases} X_a(\overline{X}^a), & \text{if } n = 0 \\ C((n-1), \overline{X}^a), & \text{if } n > 0. \end{cases}$$

X^a::C^a is short for $\lambda n, \overline{X}^a.\texttt{new\_context}_a(X^a, C^a, n, \overline{X}^a)$.

$$\texttt{getanswer}(p, C^1, \dots, C^b, \ulcorner \mathit{view} \urcorner) =$$

$$= \begin{cases} v, & \text{if } p(\ulcorner \mathit{view} \urcorner) = \ulcorner !v \urcorner \\ \texttt{getanswer}(p, C^1, \dots, C^b, \ulcorner \mathit{view} \cdot ?a \cdot !\varphi \urcorner), & \text{if } p(\ulcorner \mathit{view} \urcorner) = \ulcorner (?a, i) \urcorner \\ \bot & \text{otherwise,} \end{cases}$$

where

$$\varphi = \begin{cases} C^a(i), & \text{if } a \text{ is a leaf,} \\ \\ C^a\Big(i, \varphi_1, \dots, \varphi_l) & \text{if } a \text{ has } l \text{ successors,} \end{cases}$$

and for $1 \leq j \leq l$

$$\varphi_j = \begin{cases} \texttt{getanswer}(p, C^1, \dots, C^b, \ulcorner \mathit{view} \cdot ?a \cdot ?a.j \urcorner), & \text{if } a.j \text{ is a leaf,} \\ \\ \lambda \overline{X}^{a.j}.\texttt{getanswer}(p, C^1, \dots, X^{a.j.1}{::}C^{a.j.1}, \dots, X^{a.j.l_j}{::}C^{a.j.l_j}, \dots, C^b, \\ \qquad \ulcorner \mathit{moves} \cdot ?a \cdot ?a.j \urcorner), \\ \qquad \text{if } a.j \text{ has } l_j \text{ successors,} \end{cases}$$

Finally we set

$$\texttt{eval}(p, X_1, \dots, X_k) = \texttt{getanswer}(p, X_1{::}\Omega^1, \dots, X_k{::}\Omega^k, \dots, \Omega^b, \ulcorner ?\varepsilon \urcorner).$$

□

To summarize, we have proved: $\mathsf{HSF}_R = \mathsf{HSF}_K$ and $\mathsf{SF}_R = \mathsf{SF}_K$. If we analyse the last two proofs in the case of p being not necessarily recursive we get also the following result:

5.3.2 Corollary. *The sequential functionals SF are exactly those partial continuous functionals that are definable by (S1)–(S8) and (S11) over the partial continuous functionals C starting from all partial functions from ω to ω as base functions.* □

5.3.3 Remark. The last result corresponds to the characterization of the serial functionals in (Sazonov 1976a): "*In LCF + $D_{o\to o}$ there are expressible all serial functions and only they.*", where $D_{o\to o}$ stands for a set of constants for all partial functions from ω to ω. ◊

Chapter 6

Conclusion and Open Problems

In this work we have presented a new computation model for sequential computations in higher types.

This approach gives a new description of an order extensional fully abstract model of PCF. Whether it is isomorphic to Milner's model depends on the answer to the open question, if the hereditarily sequential functionals form a cpo at all types.

If the answer is negative, there are two possible reasons for this mismatch. If the additional elements are intuitively sequential then our model does not capture exactly the intuitive notion of sequentiality. Otherwise it would show that the requirement that all function spaces have to be cpo-s is too restrictive to capture sequentiality. We believe that this second possibility is equally sensible, since the degrees of parallelism (see Section 6.4) are not closed under extensional suprema:

6.0.4 Example. Let $\exists_n$ be the following approximations to the existential quantifier $\exists$ of type $\tau = ((\mathtt{N} \to \mathtt{B}) \to \mathtt{B})$ defined by

$$\exists_n(g) = \begin{cases} \mathtt{f} & \text{if } g(\bot) = \mathtt{f}, \\ \mathtt{t} & \text{if there is } i \leq n \text{ such that } g(i) = \mathtt{t}, \\ \bot & \text{otherwise.} \end{cases}$$

$$\exists(g) = \begin{cases} \mathtt{f} & \text{if } g(\bot) = \mathtt{f}, \\ \mathtt{t} & \text{if there is } i \in \mathbb{N} \text{ such that } g(i) = \mathtt{t}, \\ \bot & \text{otherwise.} \end{cases}$$

Then for all $n \geq 1$ the functionals $\exists_n$ belong to the same degree, moreover $\exists_n \sqsubseteq_e \exists_{n+1}$, but the functional $\exists = \bigsqcup_e \{\exists_n \mid n \geq 1\}$ belongs to a proper larger degree (Lichtenthäler 1996). $\Diamond$

Besides the question of completeness of HSF^τ there are many open problems connected with effectiveness. To finish we will state these open problems and discuss their relationship. Some of them have been firstly given in (Jung and Stoughton 1993) as a minimal condition for a solution of the full abstraction problem of PCF. These problems have also been discussed in the concluding discussion of the Workshop on "Full Abstraction of PCF and Related Languages" held from 18 to 28 April, 1995, in Aarhus.

6.1 Effectiveness

In the last chapter we have seen that the recursive strategies describe exactly the Kleene-recursive functionals over HSF. Another approach to computability is via effective numberings of the finite elements (Ershov 1973, 1975, 1977b, Weihrauch 1987).

6.1.1 Definition. A ω-algebraic cpo D is *effective* if there is a (surjective) numbering $\beta : \mathbb{N} \to D_0$ of the finite elements of D such that $\{\langle i,j\rangle \mid \beta(i) \sqsubseteq \beta(j)\}$ is r.e.

An element x of an effective cpo is *computable* if the set $\{i \mid \beta(i) \sqsubseteq X\}$ is r.e.

A continuous function $f : D \to E$ is *computable* if the set $\{\langle i,j\rangle \mid \beta^E(j) \sqsubseteq f(\beta^D(i))\}$ is r.e. $\Diamond$

6.1.2 Problem. Eff(PCF)*: Is* HSF^τ *effective for any* τ*?*

If the answer to this question is positive, this would give a third definition for a sequential functional to be computable, and the relationship to recursiveness and Kleene-recursiveness should be studied.

6.2 Counting problems

Since finite partial orders are by definition effective, we need a stronger criterion for effectiveness in the finite case:

6.2.1 Problem. Count(PCF_k)*: Is the number of elements in the fully abstract model of* PCF_k *recursive in* τ*, i.e. is the following set recursive?*

$$\{\langle \tau, n\rangle \mid n = \#\mathsf{HSF}^\tau\}$$

6.2.2 Problem. DCount(PCF_k)*: Is the number of definable elements in the standard model of* PCF_k *recursive in* τ*, i.e. is the following set recursive?*

$$\{\langle \tau, n\rangle \mid n = \#\mathsf{SF}^\tau\}$$

6.3 Definability problems

6.3.1 Problem. $\mathbf{Def}(\mathsf{PCF}_k)$: *Is definability of elements in the standard model decidable, i.e. is the following set recursive?*

$$\{F \in C^\tau \mid \exists p \in \mathsf{SeqStr}^\tau \; F = [\![p]\!]\}$$

6.3.2 Remark. The corresponding problem $\mathbf{Def}(\lambda k)$, i.e. for the simply typed lambda calculus over the full function hierarchy over a finite set with k elements is known as the "Plotkin-Statman-conjecture" (Statman 1982). Loader (1994) has shown that $\mathbf{Def}(\lambda k)$ fails for $k \geq 3$, he conjectures that it fails for $k = 2$ too. Finally, $\mathbf{Def}(\lambda 1)$ holds, since λ-definability in a model with only 1 element at ground type reduces to provability in propositional intuitionistic logic, and is thus decidable. $\Diamond$

6.3.3 Remark. Sieber's sequentiality relations (Sieber 1992) provide a characterization of definable elements up to types of level 2. Moreover this characterization is effective and Stoughton (1994) has implemented an algorithm that solves this definability problem up to level 2. $\Diamond$

In view of the full abstraction problem the following restricted definability problem is more interesting:

6.3.4 Problem. $\mathbf{RDef}(\mathsf{PCF}_k)$: *Is definability of functions restricted to definable arguments in the standard model decidable, i.e. is the following set recursive?*

$$\{F \in [\mathsf{HSF}^{\overline{\tau}} \to V_\perp \mid \exists p \in \mathsf{SeqStr}^\tau \; F = F_p\}$$

The definability problems are equivalent to the counting problems:

6.3.5 Theorem. *The problems* $\mathbf{Def}(\mathsf{PCF}_k)$ *and* $\mathbf{DCount}(\mathsf{PCF}_k)$ *are equivalent. The problems* $\mathbf{RDef}(\mathsf{PCF}_k)$ *and* $\mathbf{Count}(\mathsf{PCF}_k)$ *are equivalent.* $\Box$

6.4 Degrees of parallelism

A degree of parallelism is an equivalence class of continuous functionals that are relatively definable by each other with respect to PCF or equivalently by SeqStr. The notion of relative definability in the continuous type hierarchy is given by the following definition:

6.4.1 Definition. Let $F \in C^\tau$ and $G \in C^\sigma$. F is less parallel than G ($F \leq_{par} G$) if there exists a $p \in \mathsf{SeqStr}^{\sigma \Rightarrow \tau}$ with $F = [\![p]\!](G)$. $\Diamond$

The known facts about the degrees of parallelism can be found in (Sazonov 1976b, Bucciarelli 1996, Lichtenthäler 1996).

6.4.2 Problem. $\mathbf{Deg}(\mathsf{PCF}_k)$*: Is the relation* $\leq_{par}$ *of relative definability in the standard model decidable, i.e. is the following set recursive?*

$$\{\langle F, G\rangle \mid F \leq_{par} G\}$$

This problem is equivalent to $\mathbf{Def}(\mathsf{PCF}_k)$:

6.4.3 Theorem. *The problems* $\mathbf{Deg}(\mathsf{PCF}_k)$ *and* $\mathbf{Def}(\mathsf{PCF}_k)$ *are equivalent.*

Proof. "$\mathbf{Deg}(\mathsf{PCF}_k) \implies \mathbf{Def}(\mathsf{PCF}_k)$":

$$\begin{aligned} F \in \mathbf{Def}(\mathsf{PCF}_k) &\iff \exists p \in \mathsf{SeqStr}^{\tau}\ F = [\![p]\!] \\ &\iff \exists p' \in \mathsf{SeqStr}^{\gamma\Rightarrow\tau}\ F = [\![p']\!](\bot_\gamma) \\ &\iff \langle F, \bot_\gamma\rangle \in \mathbf{Deg}(\mathsf{PCF}_k). \end{aligned}$$

Hence the problem $\mathbf{Def}(\mathsf{PCF}_k)$ is reduced to the problem $\mathbf{Deg}(\mathsf{PCF}_k)$.

"$\mathbf{Def}(\mathsf{PCF}_k) \implies \mathbf{Deg}(\mathsf{PCF}_k)$": Let $F \in C^\tau$ and $G \in C^\sigma$. $F \leq_{par} G$ iff there exists a definable $H \in C^{\sigma\Rightarrow\tau}$ with $H(G) = F$. Since $C^{\sigma\Rightarrow\tau}$ is finite and it is decidable if $H(G) = F$ $\mathbf{Deg}(\mathsf{PCF}_k)$ follows from $\mathbf{Def}(\mathsf{PCF}_k)$. □

Bibliography

M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy
(1991) Explicit substitutions. *Journal of Functional Programming*, 1:375–416.

S. Abramsky and R. Jagadeesan
(1994) New foundations for the geometry of interaction. *Information and Computation.*

S. Abramsky, R. Jagadeesan, and P. Malacaria
(1994) Full abstraction for PCF (extended abstract). In *Theoretical Aspects of Computer Software: TACS'94, Sendai, Japan*, volume 789 of *Lecture Notes in Computer Science*, Springer-Verlag, Amsterdam, pages 1–15.

A. Asperti and G. Longo
(1991) *Categories, Types and Structures: An Introduction on Category Theory for the Working Computer Scientist.* Foundations of Computing Series. The MIT Press.

H. P. Barendregt
(1984) *The Lambda Calculus, Its Syntax and Semantics.* North-Holland, revised edition.

(1990) Functional programming and lambda calculus. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics.* Elsevier and MIT Press, pages 321–363.

U. Berger
(1990) *Totale Objekte und Mengen in der Bereichstheorie.* Dissertation, Ludwig-Maximilians-Universität München.

G. Berry
(1976) *Modèles complètement adéquats et stables des lambda-calculs typés.* Thèse de doctorat d'etat, Université Paris 7.

(1978) Stable models of typed lambda-calculi. In G. Ausiello and C. Böhm, editors, *Automata, Languages and Programming*, Springer-Verlag, Berlin, pages 72–89.

G. Berry and P.-L. Curien
(1982) Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20:265–321.

(1985) Theory and practice of sequential algorithms: the kernel of the applicative language CDS. In M. Nivat and J. C. Reynolds, editors, *Algebraic methods in semantics*. Cambridge University Press, pages 35–87.

G. Berry, P.-L. Curien, and J.-J. Lévy
(1985) Full abstraction for sequential languages: the state of art. In M. Nivat et al., editors, *Algebraic methods in semantics*. Cambridge University Press, pages 89–132.

S. Brookes and S. Geva
(1992) Stable and sequential functions on Scott domains, FM-domains and dI-domains. Technical report CMU-CS-92-121, Carnegie Mellon University, School of Computer Science.

A. Bucciarelli
(1993) *Sequential Models of PCF: Some Contributions to the Domain-Theoretic Approach to Full Abstraction.* PhD thesis, Università di Pisa-Genova-Udine, TD - 6/93.

(1996) Degrees of parallelism in the continuous type hierarchy. to appear in *Theoretical Computer Science*, ftp-able at `theory.doc.ic.ac.uk` in directory `papers/Bucciarelli`.

A. Bucciarelli and T. Ehrhard
(1991) Sequentiality and strong stability. In *Proc. 6th Annual IEEE Symp. Logic in Computer Science*. IEEE Computer Society Press, pages 138–145.

R. Cartwright, P.-L. Curien, and M. Felleisen
(1994) Fully abstract semantics for observably sequential languages. *Information and Computation*, 111:297–401.

R. Cartwright and M. Felleisen
(1992) Observable sequentiality and full abstraction. In *Nineteenth Annual ACM Symp. Principles of Programming Languages*. ACM Press, pages 328–342.

S. A. Cook
(1989) Computability and complexity of higher type functions. In Y. N. Moschovakis, editor, *Logic from Computer Science. Proc. of a Workshop held November 13-17, 1989*, Springer-Verlag, Berlin, pages 51–72.

S. A. Cook and B. M. Kapron
(1990) Characterizations of the basic feasible functionals of finite type. In S. R. Buss and P. J. Scott, editors, *Feasible Mathematics — A Mathematical Sciences Institute Workshop*, Birkhäuser, Boston; Basel; Berlin, pages 71–96.

S. A. Cook and A. Urquhart
(1989) Functional interpretations of feasibly constructive arithmetic. In *Procceedings of the Twenty First Annual ACM Symposium on Theory of Computing*. ACM Press, pages 107–112.

P.-L. Curien
(1986) Categorical combinators. *Information and Computation*, 69:188–254.

(1992) Observable algorithms on concrete data structures. In *Proc. 7th Annual IEEE Symp. Logic in Computer Science*. IEEE Computer Society Press, pages 432–443.

(1993) *Categorical Combinators, Sequential Algorithms, and Functional Programming.* Birkhäuser, second edition.

Y. L. Ershov

(1972a) Computable functionals of finite types. *Algebra and Logic (English translation)*, 11:203–242.

(1972b) Everywhere-defined continuous functionals. *Algebra and Logic (English translation)*, 11:363–368.

(1973) Theorie der Numerierungen I. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 19:289–388.

(1974) Maximal and everywhere defined functionals. *Algebra and Logic (English translation)*, 13:210–225.

(1975) Theorie der Numerierungen II. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 21:473–584.

(1976) Hereditarily effective operations. *Algebra and Logic (English translation)*, 15:400–409.

(1977a) Model **C** of partial continuous functionals. In R. O. Gandy and J. M. E. Hyland, editors, *Logic Colloquium '76*, North-Holland, Amsterdam, pages 455–467.

(1977b) Theorie der Numerierungen III. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 23:289–371.

S. Feferman

(1977) Inductive schemata and recursively continuous functionals. In R. O. Gandy and J. M. E. Hyland, editors, *Logic Colloquium '76*, North-Holland, Amsterdam, pages 373–392.

R. O. Gandy

(1967a) Computable functionals of finite type I. In J. N. Crossley, editor, *Sets, Models and Recursion Theory (Logic Colloquium '65)*, North-Holland, Amsterdam, pages 202–242.

(1967b) General recursive functionals of finite type and hierarchies of functions. *Universite Clermont-Ferrand/Faculte des Sciences: Annales*, 35:5–24.

P. Giannini and G. Longo

(1984) Effectively given domains and lambda-calculus models. *Information and Computation*, 62:36–63.

A. Grzegorczyk

(1954a) On the definition of computable functionals. *Fundamenta Mathematicae*, 42:332–239.

(1954b) Computable functionals. *Fundamenta Mathematicae*, 42:168–202.

(1957) On the definitions of computable real continuous functions. *Fundamenta Mathematicae*, 44:63–71.

(1959) Some approaches to constructive analysis. In A. Heyting, editor, *Constructivity in Mathematics*, North-Holland, Amsterdam, pages 43–61.

C. A. Gunter and D. S. Scott
(1990) Semantic domains. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics.* Elsevier and MIT Press, pages 633–674.

M. C. B. Hennessy and G. D. Plotkin
(1979) Full abstraction for a simple parallel programming language. In J. Bečvář, editor, *Mathematical Foundations of Computer Science 1979*, Springer-Verlag, Berlin, pages 108–120.

H. J. Hoover
(1990) Computational models for feasible real analysis. In S. R. Buss and P. J. Scott, editors, *Feasible Mathematics — A Mathematical Sciences Institute Workshop*, Birkhäuser, Boston; Basel; Berlin, pages 221–237.

J. M. E. Hyland and C.-H. L. Ong
(1994) On full abstraction for PCF: I, II and III. 130 pages, ftp-able at `theory.doc.ic.ac.uk` in directory `papers/Ong`.

A. Jung and A. Stoughton
(1993) Studying the fully abstract model of PCF within its continuous function model. In M. Bezem and J. F. Groote, editors, *Proc. Int. Conf. Typed Lambda Calculi and Applications*, volume 664 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pages 230–244.

A. Jung and J. Tiuryn
(1993) A new characterization of lambda definability. In M. Bezem and J. F. Groote, editors, *Proc. Int. Conf. Typed Lambda Calculi and Applications*, volume 664 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pages 245–257.

A. Jung, M. P. Fiore, E. Moggi, P. O'Hearn, J. Riecke, G. Rosolini, and I. Stark
(1996) Domains and denotational semantics: History, accomplishments and open problems. Technical Report CSR-96-2, University of Birmingham, School of Computer Science.

G. Kahn and G. D. Plotkin
(1978) Structures de données concrètes. Rapport 336, IRIA-LABORIA.

B. M. Kapron
(1991) Feasible computation in higher types. Technical Report 249/91, Department of Computer Science, University of Toronto.

B. M. Kapron and S. A. Cook
(1991) A new characterization of Mehlhorn's polynomial time functionals (extended abstract). In *Proc. 32nd Annual Symp. Foundations of Computer Science.* IEEE Computer Society Press, pages 342–347.

A. S. Kechris and Y. N. Moschovakis
(1977) Recursion in higher types. In J. Barwise, editor, *Handbook of Mathematical Logic.* North-Holland, Amsterdam, pages 681–737.

D. P. Kierstead
(1983) Syntax and semantics in higher-type recursion theory. *Transactions of the AMS*, 276:67–105.

S. C. Kleene

(1959) Recursive functionals and quantifiers of finite types I. *Transactions of the AMS*, 91:1–52.

(1962a) Turing-machine computable functionals of finite types I. In P. Suppes, editor, *Proc. 1960 Congress of Logic, Methodology and the Philosophy of Science*, North-Holland, Amsterdam, pages 38–45.

(1962b) Turing-machine computable functionals of finite types II. *Proceedings of the London Mathematical Society*, 12 (3rd series):245–258.

(1963) Recursive functionals and quantifiers of finite types II. *Transactions of the AMS*, 108:106–142.

(1978) Recursive functionals and quantifiers of finite types revisited I. In J. E. Fenstad, R. O. Gandy, and G. E. Sacks, editors, *Generalized Recursion Theory II*, North-Holland, Amsterdam, pages 185–222.

(1980) Recursive functionals and quantifiers of finite types revisited II. In J. Barwise, H. J. Keisler, and K. Kunen, editors, *The Kleene Symposium*, North-Holland, Amsterdam, pages 1–29.

(1982) Recursive functionals and quantifiers of finite types revisited III. In G. Metakides, editor, *Patras Logic Symposium*, North-Holland, Amsterdam, pages 1–40.

(1985) Unimonotone functions of finite types (Recursive functionals and quantifiers of finite types revisited IV). In A. Nerode and R. A. Shore, editors, *Recursion Theory*, American Mathematical Society, Providence, Rhode Island, pages 119–138.

G. Kreisel

(1971) Some reasons for generalizing recursion theory. In R. O. Gandy and C. M. E. Yates, editors, *Logic Colloquium '69*, North-Holland, Amsterdam, pages 139–198.

G. Kreisel, D. Lacombe, and J. R. Schoenfield

(1959) Partial recursive functionals and effective operations. In A. Heyting, editor, *Constructivity in Mathematics*, North-Holland, Amsterdam, pages 290–297.

C. Kreitz and K. Weihrauch

(1982) Complexity theory on real numbers and functions. In A. B. Cremers and H. P. Kriegel, editors, *Proc. 6th GI Conference on Theoretical Computer Science*, Springer-Verlag, Berlin, pages 165–174.

B. Lichtenthäler

(1996) Degrees of parallelism. Informatik Berichte 96-01, Universität - GH Siegen, Germany.

R. Loader

(1994) The undecidability of λ-definability. In M. Zeleny, editor, *The Church Festschrift*. University of Chicago Press.

G. Longo

(1984) Limits, higher type computability and type-free languages. In M. P. Chytil and V. Koubek, editors, *Mathematical Foundations of Computer Science 1984*, Springer-Verlag, Berlin, pages 96–114.

(1988) On Church's formal theory of functions and functionals. The λ-calculus: connections to higher type recursion theory, proof theory, category theory. *Annals of Pure and Applied Logic*, 40:93–133.

G. Longo and E. Moggi
(1984) The hereditary partial effective functionals and recursion theory in higher types. *Journal of Symbolic Logic*, 49:1319–1332.

K. Mehlhorn
(1976) Polynomial and abstract subrecursive classes. *Journal of Computer and System Science*, 12:147–178.

R. Milner
(1977) Fully abstract models of typed λ-calculi. *Theoretical Computer Science*, 4:1–22.

J. C. Mitchell
(1990) Type systems for programming languages. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics.* Elsevier and MIT Press, pages 365–458.

Y. N. Moschovakis
(1969a) Abstract first order computability. I, II. *Transactions of the AMS*, 138:427–464,465–504.

(1969b) Abstract computability and invariant definability. *Journal of Symbolic Logic*, 34:605–633.

(1971) Axioms for computation theories - First draft. In R. O. Gandy and C. M. E. Yates, editors, *Logic Colloquium '69*, North-Holland, Amsterdam, pages 199–255.

P. D. Mosses
(1990) Denotational semantics. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B: Formal Models and Semantics.* Elsevier and MIT Press, pages 575–631.

J. Myhill and J. C. Shepherdson
(1955) Effective operations on partial recursive functions. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 1:310–317.

H. Nickau
(1994) Hereditarily sequential functionals. In A. Nerode and Y. V. Matiyasevich, editors, *Proc. Symp. Logical Foundations of Computer Science: Logic at St. Petersburg*, volume 813 of *Lecture Notes in Computer Science.* Springer-Verlag, pages 253–264.

D. Normann
(1980) *Recursion on the Countable Functionals*, volume 811 of *Lecture Notes in Mathematics.* Springer-Verlag.

(1981) The continuous functionals: computations, recursions and degrees. *Annals of Mathematical Logic*, 21:1–26.

(1982) External and internal algorithms on the continuous functionals. In G. Metakides, editor, *Patras Logic Symposium*, North-Holland, Amsterdam, pages 137–144.

(1985) Aspects of the continuous functionals. In A. Nerode and R. A. Shore, editors, *Recursion Theory*, American Mathematical Society, Providence, Rhode Island, pages 171–176.

P. W. O'Hearn and J. G. Riecke
(1994) Kripke logical relations and PCF. Preprint.

C.-H. L. Ong
(1995) Correspondence between operational and denotational semantics. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science, Vol 4*. Oxford University Press, pages 269–356.

G. D. Plotkin
(1977) LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255.

(1980) Lambda-definability in the full type hierarchy. In J. R. Hindley and J. P. Seldin, editors, *To H. B. Curry, essays on combinatory logic, lambda-calculus and formalism*. Academic Press, London, pages 363–374.

V. Y. Sazonov
(1975) Sequentially and parallely computable functionals. In C. Böhm, editor, *Proc. Symp. Lambda-Calculus and Computer Science Theory*, Springer-Verlag, Berlin, pages 312–318.

(1976a) Expressibility of functions in D. Scott's LCF language. *Algebra and Logic (English translation)*, 15:192–206.

(1976b) Degrees of parallelism in computations. In A. Mazurkiewicz, editor, *Mathematical Foundations of Computer Science 1976*, Springer-Verlag, Berlin, pages 517–523.

D. S. Scott
(1975) Lambda calculus and recursion theory. In S. Kanger, editor, *Proc. 3rd Scandinavian Logic Symposium*, pages 154–193.

(1976) Data types as lattices. *SIAM Journal of Computing*, 5:522–587.

(1980a) Lambda calculus: Some models, some philosophy. In J. Barwise, H. J. Keisler, and K. Kunen, editors, *The Kleene Symposium*, North-Holland, Amsterdam, pages 223–265.

(1980b) Relating theories of the λ-calculus. In J. R. Hindley and J. P. Seldin, editors, *To H.B. Curry, essays on combinatory logic, lambda-calculus and formalism*, Academic Press, London, pages 403–450.

(1993) A type-theoretic alternative to ISWIM, CUCH, OWHY. *Theoretical Computer Science*, 121:411–440. Reprint of a manuscript written in 1969.

A. Seth
(1992) There is no recursive axiomatization for feasible functionals of type 2. In *Proc. 7th Annual IEEE Symp. Logic in Computer Science*. IEEE Computer Society Press, pages 286–295.

K. Sieber
(1992) Reasoning about sequential functions via logical relations. In M. P. Fourman, P. T. Johnstone, and A. M. Pitts, editors, *Applications of Categories in Computer Science*. Cambridge University Press, pages 258–269.

M. B. Smyth

(1977) Effectively given domains. *Theoretical Computer Science*, 5:257–274.

I. N. Soskov

(1991) Second order definability via enumerations. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 37:45–54.

R. Statman

(1982) Completeness, invariance and λ-definability. *Journal of Symbolic Logic*, 47:17–26.

(1985) Logical relations and the typed lambda-calculus. *Information and Computation*, 65:85–97.

(1990) Some models of Scott's theory LCF based on a notion of rate of convergence. In *Contemporary Mathematics.* AMS Press, pages 263–280.

A. Stoughton

(1988) *Fully Abstract Models of Programming Languages.* Pitman Publishing, London.

(1990) Equationally fully abstract models of PCF. In M. Main et al., editors, *Mathematical Foundations of Programming Language Semantics '89*, Springer-Verlag, Berlin, pages 271–283.

(1994) Mechanizing logical relations. In *Mathematical Foundations of Programming Language Semantics '93*, volume 802 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin.

M. B. Trakhtenbrot

(1975) On representation of sequential and parallel functions. In *Proc. 4th Symp. Mathematical Foundations of Computer Science.* Springer-Verlag, pages 411–417.

J. Vuillemin

(1974) Correct and optimal implementation of recursion in a simple programming language. *Journal of Computer and System Science*, 9:332–354.

K. Weihrauch

(1981) Recursion and complexity theory on CPO-s. In P. Deussen, editor, *Theoretical Computer Science. 5th GI-Conference, Karlsruhe, March 1981*, Springer-Verlag, Berlin, pages 195–202.

(1985) Type 2 recursion theory. *Theoretical Computer Science*, 38:17–33.

(1987) *Computability*, volume 9 of *EATCS Monographs on Theoretical Computer Science.* Springer-Verlag.

G. Winskel

(1987) Event structures. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency. Advances in Petri Nets 1986, Part II*, Springer-Verlag, Berlin, pages 325–392.

G. O. Zhang

(1991) *Logic of Domains.* Birkhäuser.

Index