Hereditarily Sequential Functionals

Hanno Nickau*
Theoretische Informatik, FB 6, Universität-GH-Siegen,
D-57068 Siegen, Germany
email: nickau@informatik.uni-siegen.de

Abstract

In order to define models of simply typed functional programming languages being closer to the operational semantics of these languages, the notions of sequentiality, stability and seriality were introduced. These works originated from the definability problem for PCF, posed in [Sco72], and the full abstraction problem for PCF, raised in [Plo77].

The presented computation model, forming the class of hereditarily sequential functionals, is based on a game in which each play describes the interaction between a functional and its arguments during a computation. This approach is influenced by the work of Kleene [Kle78], Gandy [Gan67], Kahn and Plotkin [KP78], Berry and Curien [BC82, Cur86, Cur92], and Cartwright and Felleisen [CF92].

We characterize the computable elements in this model in two different ways: (a) by recursiveness requirements for the game, and (b) as definability with the schemata (S1)–(S8), (S11), which is related to definability in PCF. It turns out that both definitions give the same class of computable functionals. So a robust notion of (sequential) computability on higher types is presented.

1 Introduction

In ordinary computability theory all reasonable computation models are equivalent leading to Church's Thesis that this is the right notion 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.

As there is no common notion of computable functional, up to now there are very few approaches to higher type complexity (e.g. [CK90, Wei85]). From our view of complexity, a notion of higher type computability is needed where (1) terminating computations are finite

*This research was supported in part by ESPRIT BRA 3230 and BRA 7232

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) Algorithms should be sequential in the sense that computations should proceed stepwise with one focus at a time. (3) A computed functional should only depend on the extension of its input not on a particular intension describing the input.

PCF, a typed lambda-calculus with constants for arithmetic and recursion on all types [Plo77], fulfills the above conditions as long as we allow only PCF definable inputs. Up to now there is no satisfactory model of PCF for our purposes. Here we mention only two. The "standard model" based on partial continuous functionals contains functions which are not sequential, e.g. parallel-or, hence it violates (2) (cf. [Plo77]). The model of Berry and Curien [BC85, Cur86] based on sequential algorithms gives a concrete description of the interaction between the program and the input, but the possible use of intensional aspects of the inputs violates (3). For example the algorithms left-strict-or and right-strict-or are distinguishable by a sequential algorithm but have the same input-output behaviour.

Starting from these observations and analysing Kleene's unimonotone functionals [Kle78] a new computation model has been developed which is as concrete as the sequential algorithm model but fulfills (3). This 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 which is not extensional w.r.t. application of sequential strategies but is extensional in the sense of (3).

To each sequential strategy p we relate a functional F_p describing the extension of p. We call the functionals F_p hereditarily sequential and the set of these functionals *HSF*. We give another interpretation for the sequential strategies as partial continuous functionals, denoted by $[\![p]\!]$. The functionals $[\![p]\!]$ are called sequential and the set of these functionals *SF*. The sequential functionals seem to be essentially the same as the serial functionals in [Saz76].

We give two definitions for elements of *HSF* (*SF*) to be computable. We have the recursive elements of *HSF* (*SF*) being extensions of recursive sequential strategies. They correspond to the effectively serial functionals in [Saz76]. Secondly we consider the class of Kleene-recursive functionals, i.e. definable by Kleene's schemata (S1) – (S8), (S11) [Kle59] over *HSF* (*SF*), which is strongly related to definability in PCF [Pla66].

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.

Because of the limited space the proofs are only sketched. A version of this paper including full proofs will be available as a technical report in the series "Informatik-Berichte, Fachgruppe Informatik, Universität-GH-Siegen".

2 Game of Higher Types

2.1 Types and Type Structures

We consider here only simple types over one groundtype γ. So the *types* are inductively given by two clauses. The groundtype γ is a type, and if $\tau_1, \ldots \tau_k$ are types, then $\tau_1, \ldots, \tau_k \to \gamma$ is also a type.

The *unary types* are defined by $\underline{0} = \gamma$ and $\underline{n{+}1} = \underline{n} \to \gamma$.

Each occurrence of γ in a type 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.

Example 2.1.1 The type $(\gamma, \gamma \to \gamma), \gamma \to \gamma$ considered as a tree and as a tree of occurrences:

$\diamond$

A type τ is unary iff all occurrences in τ are identified by sequences of ones.

The groundtype γ serves as a name for a fixed set of values V. In this paper we consider only the case of $V = \omega$, the set of natural numbers.

A *type structure* is a collection of sets $\{D^\tau \mid \tau \text{ a type}\}$ such that (i) $D^\gamma \supseteq V$ and (ii) 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.

Example 2.1.2 Examples of type structures are the hereditarily total functionals $(HT^\tau)_\tau$ given by $HT^\gamma = \omega$ and $HT^{\tau_1, \ldots, \tau_k \to \gamma}$ being the set of all (total) maps from $HT^{\tau_1} \times \cdots \times HT^{\tau_k}$ to ω, and the Scott-Ershov-hierarchy $(C^\tau)_\tau$ of the partial continuous functionals over the flat domain $C^\gamma = \omega_\perp$. $\diamond$

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.

In the sequel it will be useful to have a lambda notation for defining functions of type τ_a in some type structure $(D^\sigma)_\sigma$. For that purpose we assume a set of variables $\{X_a \mid a \text{ occurence in } \tau\}$, where X_a ranges over D^{τ_a}. By $\overline{X}^a$ we denote the sequence $X_{a.i}, \ldots, X_{a.l}$ if a has l successors and the empty sequence ε if a is a leaf. For a list of occurrences $A = [a_1, \ldots, a_m]$ let $\overline{X}^A = \overline{X}^{a_1}, \ldots, \overline{X}^{a_m}$. If in addition a is an occurrence, then $A :\cup: a = A$ for $a \in A$ and $A :\cup: a = [a_1, \ldots, a_m, a]$ for $a \notin A$. If t is a parameterized definition of an element in D^γ, then $\lambda \overline{X}^a.t = \lambda X_{a.1}, \ldots, X_{a.l}.t$ 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 have $\lambda \overline{X}^a.t = t$.

Example 2.1.3 Let $\tau = (\gamma, \gamma \to \gamma), \gamma \to \gamma$ be the type of example 2.1.1, then we have $\tau_1 = \gamma, \gamma \to \gamma$ and $\tau_2 = \gamma$ well corresponding to the notation $\tau = \tau_1, \tau_2 \to \gamma$ and $\tau_{1.1} = \tau_{1.2} = \gamma$. 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 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. 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 $!n$, where n is in the set of values.

The choice of moves is restricted by the following rules. Player P can only play questions $?a$ with $|a|$ odd and player O only those with $|a|$ even. The question $?a.i$ can only be played if question $?a$ is open at that moment. The last open occurrence of $?a$ is the *enabling for* $?a.i$. All moves between the enabling $?a$ and $?a.i$ are hidden for the opposing player until $?a.i$ gets answered.

An answer $!n$ answers the last open question $?a$. All moves between this question $?a$ and $!n$ are hidden for the opposing player. A question is *open* if it is played but not answered and not hidden.

Each play starts with the (not enabled and not repeatable) question $?\varepsilon$ of O. A play is *finished* if the initial question $?\varepsilon$ is answered. If this last answer is $!n$ then we call n the *result* of this play. If s is a not finished play, then we call the sequence of moves of s which are not hidden for the player in turn *view* of the play. If P is in turn we call it *P-view* otherwise *O-view*.

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 $!n$ 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.

2.3 Sequential Strategies

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. A *sequential strategy for P* (over τ) is a partial function from P-views (over τ) to allowed moves and a *sequential strategy for O* (over τ) is a partial function from O-views (over τ) to allowed moves.

Example 2.3.1 (a) We give an example of a strategy p for P over τ (τ as in example 2.1.1) representing the functional $\lambda gz.g(z,5)$:

$$\begin{array}{lcll}
p([?\varepsilon]) & = & ?1 & \\
p([?\varepsilon, ?1, !n]) & = & !n & (n \in \omega) \\
p([?\varepsilon, ?1, ?1.1]) & = & ?2 & \\
p([?\varepsilon, ?1, ?1.1, ?2, !m]) & = & !m & (m \in \omega) \\
p([?\varepsilon, ?1, ?1.2]) & = & !5 &
\end{array}$$

(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:

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

$\diamond$

Now we show how O-strategies can be defined from P-strategies of lower types. Let $\tau = \tau_1, \ldots, \tau_k \to \gamma$. For all O-views s over τ there exists an $i \leq k$ such that all questions but the first in s are of the form $?i.a$. In this case let $s^{(-i)}$ result from s by omitting the first question $?\varepsilon$ and by substituting each question $?i.a$ with $?a$. Then $s^{(-i)}$ is a P-view over τ_i. If the questions in s are of the form $?j.a$ with $j \neq i$ then $s^{(-i)}$ is undefined. Vice versa, if we start with a P-view s' over τ_i than we get an O-view $s'^{(i)}$ over τ by substituting each question $?a$ with $?i.a$ and by adding the question $?\varepsilon$ as the new first element. We have $(s'^{(i)})^{(-i)} = s'$ and $(s^{(-i)})^{(i)} = s$ if $s^{(-i)}$ is defined.

Example 2.3.2 Let s be the O-view $[?\varepsilon, ?1, ?1.2, !3, ?1.1, !5]$ then $s^{(-1)} = [?\varepsilon, ?2, !3, ?1, !5]$. $\diamond$

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

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

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

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

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 ($i = 1, \ldots, k$). In this case we write $q = (p_1, \ldots, p_k)^O$.

Example 2.3.3 The strategy q from example 2.3.1 (b) can be written as $q = (p_1, p_2)^O$ with:

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

◇

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 $\mathit{SeqStr} = \bigcup_\tau \mathit{SeqStr}^\tau$.

Let p be a sequential strategy over $\tau = \tau_1, \ldots, \tau_k \to \gamma$ and p_i $(i = 1, \ldots, k)$ sequential strategies over τ_i. If there exists a finished play between P with strategy p and O with strategy $(p_1, \ldots, p_k)^O$ with value n then we write $p[p_1, \ldots, p_k] = n$, otherwise $p[p_1, \ldots, p_k]$ is undefined $(= \bot)$. Note that the finished play is unique, if it exists. That means $p[\,\cdot\,]$ is a partial function from $\mathit{SeqStr}^{\tau_1} \times \cdots \times \mathit{SeqStr}^{\tau_k}$ to ω.

Example 2.3.4 Let p, p_1 and p_2 be the strategies from examples 2.3.1 and 2.3.3 respectively. The play for $p[p_1, p_2]$ proceeds as follows:

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

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

2.4 Decision trees

Now we consider another representation of the sequential strategies. In this representation, the form being similar to that used by Cartwright and Felleisen [CF92] for sequential algorithms and by Bucciarelli [Buc93] for Kleene's oracles, only the relevant part of the strategy is shown. For instance if for some strategy $p([?\varepsilon]) = ?1$ then in no play there will be a P-view starting with $?\varepsilon, ?2$. Hence a value of p on those views is irrelevant. More formally we define the *relevant part* $\tilde{p}$ of p to be the largest substrategy of p closed under the following two conditions:

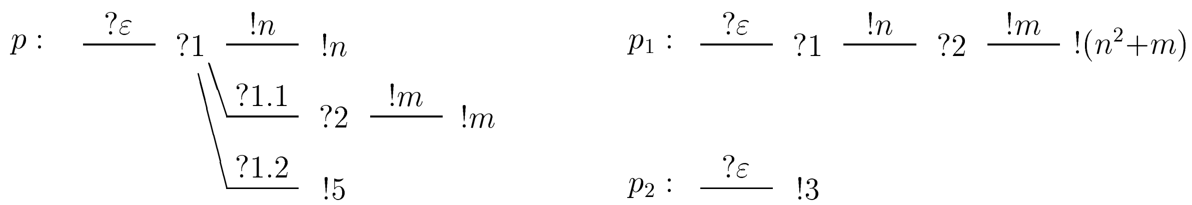
- $[?\varepsilon] \in \mathrm{dom}(\tilde{p})$,

- if $[?\varepsilon, \ldots, s_{n-2}, s_{n-1}, s_n] \in \mathrm{dom}(\tilde{p})$, then $\tilde{p}([?\varepsilon, \ldots, s_{n-2}]) = s_{n-1}$.

Obviously we have $\tilde{p}[\,\cdot\,] = p[\,\cdot\,]$.

The relevant part $\tilde{p}$ 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. So if $\tilde{p}([?\varepsilon, \ldots, s_{n-1}]) = s_n$ then there is a path labeled with $?\varepsilon, \ldots, s_{n-1}$ from the root to a vertex labeled with s_n.

Example 2.4.1 Let p, p_1 and p_2 be the strategies from examples 2.3.1 and 2.3.3 respectively. They contain no irrelevant information and the corresponding decision trees are the following:



In fact the trees for p and p_1 are infinite. For instance the notation $\xrightarrow{!n}\, !n$ means that there is a subtree of this form for each $n \in \omega$. $\diamond$

3 Hereditarily Sequential Functionals

3.1 Definition of *HSF*

Now we consider the input/output behaviour or extension of the sequential strategies. For each $p \in \mathit{SeqStr}^\tau$ we define a functional F_p. In this case we call p a *sequential strategy for* F_p. The functionals F which possess sequential strategies are called *hereditarily sequential*. We write HSF^τ for the set of hereditarily sequential functionals of type τ and $\mathit{HSF} = \bigcup_\tau \mathit{HSF}^\tau$.

For $\tau = \gamma$ there is exactly one P-view $[?\varepsilon]$ and no O-view. There is the undefined strategy, which is mapped to $\bot$, and for each $n \in \omega$ the strategy giving the final answer $!n$, which is mapped to n. So we have $\mathit{HSF}^\gamma = \omega_\bot$.

For $\tau = \tau_1, \ldots, \tau_k \to \gamma$ we define $F_p : \mathit{HSF}^{\tau_1} \times \cdots \times \mathit{HSF}^{\tau_k} \to \omega_\bot$ by $F_p(G_1, \ldots, G_k) = p[p_1, \ldots, p_k]$ with $F_{p_i} = G_i$ for all $i = 1, \ldots, k$. 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 lemma simultaneously with the definition of the hereditarily sequential functionals.

Lemma 3.1.1 *Let p be a sequential strategy over $\tau = \tau_1, \ldots, \tau_k \to \gamma$ and let p_i, q_i be sequential strategies over τ_i with $F_{p_i} = F_{q_i}$ for $i = 1, \ldots, k$. Then $p[p_1, \ldots, p_k] = p[q_1, \ldots, q_k]$.* $\square$

This is proved by a straightforward but tedious case analysis using the restriction rules in the game.

3.2 Definition of *SF*

In this subsection we consider another possible interpretation of the sequential strategies yielding partial continuous functionals which we call *sequential.* More formally, we define an interpretation map $[\![\,\cdot\,]\!] : \mathit{SeqStr}^\tau \to C^\tau$, where $(C^\tau)_\tau$ is the Scott-Ershov-hierarchy of the partial continuous functionals over the natural numbers. The image under $[\![\,\cdot\,]\!]$ we call SF^τ.

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

$$[\![p]\!] = \lambda\overline{X}.[\![d']\!] = \lambda X_1, \ldots, X_k.[\![d']\!],$$

where the semantics of a finite decision tree (for fixed τ) is inductively given by:

$$\begin{array}{rcl} [\![\emptyset]\!] & = & \bot, \\ [\![!n]\!] & = & n \qquad (n \in \omega), \end{array}$$

$$\left[\!\!\left[\begin{array}{lll} ?a & \xrightarrow{!v_1} & d_1 \\ & \vdots & \vdots \\ & \xrightarrow{!v_n} & d_n \\ & \xrightarrow{?a.1} & d'_1 \\ & \vdots & \vdots \\ & \xrightarrow{?a.l} & d'_l \end{array} \right]\!\!\right] = \begin{cases} [\![d_1]\!], & \text{if } X_a(\lambda\overline{X}^{a.1}[\![d'_1]\!], \ldots, \lambda\overline{X}^{a.l}[\![d'_l]\!]) = v_1, \\ \quad\vdots & \qquad\qquad\vdots \\ [\![d_n]\!], & \text{if } X_a(\lambda\overline{X}^{a.1}[\![d'_1]\!], \ldots, \lambda\overline{X}^{a.l}[\![d'_l]\!]) = v_n, \\ \bot & \text{otherwise.} \end{cases}$$

Here the X_a range over C^{τ_a}.

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

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

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

Note that the collection $\{\mathit{SF}^\tau \mid \tau \text{ a type}\}$ does not form a type structure as defined in 2.1. But the hereditarily sequential functionals and the sequential functionals are nicely related as the following lemma shows.

Lemma 3.2.1 *Let $p, p' \in \mathit{SeqStr}^\tau$. Then $[\![p]\!]^\tau = [\![p']\!]^\tau$ implies $F_p = F_{p'}$, and $F_p = F_{p'}$ if and only if $[\![p]\!]^\tau|_{SF} = [\![p']\!]^\tau|_{SF}$.* □

For a proof note that this follows from the fact $F_p(F_{p_1}, \ldots, F_{p_k}) = p[p_1, \ldots, p_k] = [\![p]\!]([\![p_1]\!], \ldots, [\![p_k]\!])$ for all sequential strategies $p \in \mathit{SeqStr}^\tau$, $p_i \in \mathit{SeqStr}^{\tau_i}$.

So each hereditarily sequential functional can be seen as the restriction of a sequential functional to (hereditarily) sequential arguments, in other words the relation $F_p \mapsto [\![p]\!]|_{SF}$ defines a bijection between HSF^τ and $\mathit{SF}^\tau|_{SF}$. The next example shows that the reverse of the first implication in Lemma 3.2.1 does not hold in general, hence the classes *SF* and *HSF* are in fact different.

Example 3.2.2 Let $p, p' \in \mathit{SeqStr}^{(\gamma,\gamma\to\gamma)\to\gamma}$ be given by the following decision trees:

$$p : \xrightarrow{?\varepsilon} ?1 \begin{array}{l} \xrightarrow{!0} ?1 \begin{array}{l} \xrightarrow{!0} !0 \\ \xrightarrow{?1.2} !0 \end{array} \\ \xrightarrow{?1.1} !0 \end{array} \qquad\qquad p' : \xrightarrow{?\varepsilon} ?1 \xrightarrow{!0} !0$$

Then $F_p = F_{p'}$ with $F_p(G) = 0$ iff $G(\bot,\bot) = 0$. But for H with $H(\bot, 0) = H(0,\bot) = 0$ and $H(\bot,\bot) = \bot$ we have $[\![p]\!](H) = 0$ and $[\![p']\!](H) = \bot$, hence $[\![p]\!] \neq [\![p']\!]$. Note that H is not sequential. $\diamond$

Remark 3.2.3 The class *SF* of sequential functionals seems to be essentially the same as the class of serial functionals defined in [Saz76]. The serial functionals are characterized in terms of a kind of oracle Turing machine where questions to the oracles are posed in form of applicative terms (PCF terms). It is beyond the scope of this paper to explore the connections between his approach and the presented one in more details. $\diamond$

3.3 The extensional and the sequential ordering

Each set HSF^τ is naturally ordered by two different orderings. One is the pointwise or extensional ordering $\sqsubseteq_e$, defined by:

$$F \sqsubseteq_e F' \quad\Longleftrightarrow\quad \forall (F_1,\ldots,F_k) \in \mathrm{dom}(F) : F(F_1,\ldots,F_k) = F'(F_1,\ldots,F_k).$$

Theorem 3.3.1 *$(\mathit{HSF}^\tau, \sqsubseteq_e)$ is a Scott-domain, i.e. an ω-algebraic, bounded complete cpo. The hereditarily sequential functionals are continuous w.r.t. $\sqsubseteq_e$.* $\square$

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

Example 3.3.2 Let $p \in \mathit{SeqStr}^{(\gamma,\gamma\to\gamma)\to\gamma}$ and $q, q', \tilde{q} \in \mathit{SeqStr}^{\gamma,\gamma\to\gamma}$ be given by the following decision trees:

$$p : \xrightarrow{?\varepsilon} ?1 \begin{array}{l} \xrightarrow{!0} !0 \\ \xrightarrow{?1.1} ?1 \begin{array}{l} \xrightarrow{!0} !0 \\ \xrightarrow{?1.1} !0 \end{array} \\ \xrightarrow{?1.2} ?1 \begin{array}{l} \xrightarrow{!0} !0 \\ \xrightarrow{?1.2} !0 \end{array} \end{array} \qquad\qquad \begin{array}{l} q : \xrightarrow{?\varepsilon} ?1 \xrightarrow{!0} !0 \\ q' : \xrightarrow{?\varepsilon} ?2 \xrightarrow{!0} !0 \\ \tilde{q} : \xrightarrow{?\varepsilon} ?1 \xrightarrow{!0} ?2 \xrightarrow{!0} !0 \end{array}$$

Then F_p, F_q, $F_{q'}$ and $F_{\tilde{q}}$ 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 $F_{\tilde{q}}(0,0) = 0$ respectively. Note that $F_q \sqcap_e F_{q'} = F_{\tilde{q}}$, hence we obtain

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

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

Remark 3.3.3 Note that $(HSF^\tau, \sqsubseteq_e)$ is isomorphic to $(C^\tau, \sqsubseteq_e)$ if and only if τ is a unary type. In that case they are also isomorphic to $(SF^\tau, \sqsubseteq_e)$. $\diamond$

The second ordering is more closely related to the definition of *HSF*, being the ordering inherited from the sequential strategies. It is called sequential ordering $\sqsubseteq_s$, defined by:

$$F \sqsubseteq_s F' \quad \Longleftrightarrow \quad \exists p, p' \in \mathit{SeqStr}^\tau \text{ with } p \subseteq p', F = F_p, \text{ and } F' = F_{p'}.$$

Theorem 3.3.4 *$(HSF^\tau, \sqsubseteq_s)$ is a dI-domain. The hereditarily sequential functionals are stable w.r.t. $\sqsubseteq_s$.* $\square$

For a proof note that (a) $(\mathit{SeqStr}^\tau, \subseteq)$ is a dI-domain, (b) if $F_p \sqsubseteq_s F_q$ then there exist $p' \subseteq q$ and $q' \supseteq p$ with $F_p = F_{p'}$ and $F_q = F_{q'}$, and (c) $p[\,\cdot\,]$ is a stable function from $(\mathit{SeqStr}^{\tau_1}, \subseteq) \times \cdots \times (\mathit{SeqStr}^{\tau_k}, \subseteq)$ to $(\omega_\perp, \sqsubseteq)$.

4 Computability on *HSF* and *SF*

In this section we give two definitions of those elements of *HSF* (*SF*) which 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*). We show that both notions define the same class of computable elements of *HSF* (*SF*). From the equivalence proof we get also another characterization for the class *SF*.

4.1 Recursive sequential functionals

An element $F \in HSF$ is *recursive*, if there exists a recursive sequential strategy p with $F = F_p$. By a recursive sequential strategy we mean a sequential strategy which is recursive in the ordinary sense with respect to a suitable coding of views and moves. We denote the set of recursive sequential functionals by HSF_R.

In the same way we call an element $F \in SF$ recursive, if there is a recursive sequential strategy p with $F = [\![p]\!]$. The recursive sequential functionals considered this way correspond to the effectively serial functionals defined in [Saz76] (cf. Remark 3.2.3).

4.2 Kleene-recursive sequential functionals

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*) [Kle59]. We denote the set of Kleene-recursive sequential functionals by HSF_K (SF_K).

The schemata (S1)–(S8) and (S11) are given below. The schemata define functionals F under the hypothesis that $F_1, F_2, \ldots, F_l$ have been defined before, the variables X_a range over HSF^{τ_a} (over SF^{τ_a}). The $\lambda\overline{X}^a$-notation is explained in section 2.1.

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

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

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

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

(S5) primitive recursion $F(\overline{X}) = \begin{cases} F_1(X_2, \ldots, X_k) & \text{if } X_1 = 0 \\ F_2(F(X_1 - 1, X_2, \ldots, X_k), \overline{X}) & \text{if } X_1 > 0 \end{cases}$

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

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

(S8) functional-application $F(\overline{X}) = X_1(\lambda\overline{X}^{1.1} F_1(\overline{X}, \overline{X}^{1.1}), \ldots, \lambda\overline{X}^{1.l} F_l(\overline{X}, \overline{X}^{1.l}))$
$(\tau_1 = \tau_{1.1}, \ldots, \tau_{1.l} \to \gamma)$

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

Now we show that each Kleene-recursive (hereditarily) sequential functional is recursive, i.e. $HSF_K \subseteq HSF_R$ and $SF_K \subseteq SF_R$.

Theorem 4.2.1 *The recursive (hereditarily) sequential functionals are closed under the schemata (S1)–(S8) and (S11).*

Proof. First we give recursive sequential strategies for the schemata (S1)–(S3) and (S7) not depending on other strategies:

(S1), "successor": A recursive sequential strategy p for $F(\overline{X}) = X_1 + 1$ $(\tau_1 = \gamma)$ is given by $p([?\varepsilon]) = ?1$ and $p([?\varepsilon, ?1, !n]) = !(n{+}1)$ for $n \in \omega$.

(S2), "constants": For each $c \in \omega$ a recursive sequential strategy p for $F(\overline{X}) = c$ is given by $p([?\varepsilon]) = !c$.

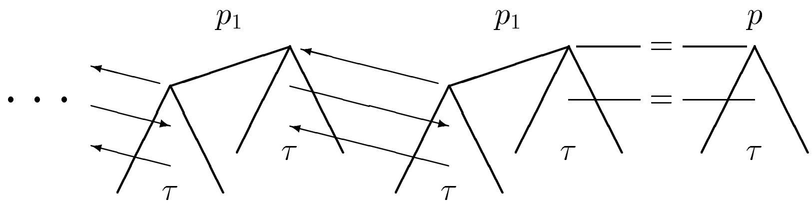
(S3) "identity": A recursive sequential strategy p for $F(\overline{X}) = X_1$ $(\tau_1 = \gamma)$ is given by $p([?\varepsilon]) = ?1$ and $p([?\varepsilon, ?1, !n]) = !n$ for $n \in \omega$.

(S7), "function-application": A recursive sequential strategy p for $F(\overline{X}) = X_1(X_2, \ldots, X_{l+1})$ $(\tau_i = \gamma$ for $i = 2, \ldots, l{+}1$, $\tau_1 = \gamma, \ldots, \gamma \to \gamma)$ is given by

$$p(s) = \begin{cases} ?1, & \text{if } s = [?\varepsilon] \\ ?i{+}1, & \text{if } s \text{ ends in } ?1.i \\ !n, & \text{if } s \text{ ends in } !n \end{cases}$$

For the other schemata we have to define a recursive sequential strategy p for F under the assumption that recursive sequential strategies $p_1, p_2, \ldots, p_l$ are given for $F_1, F_2, \ldots, F_l$ respectively.

We show how does it work for (S11), "recursion", $F(\overline{X}) = F_1(F, \overline{X})$, the other schemata are handled similarly. The idea for computing $p(s)$ is to simulate the strategy p_1, recursively if the first argument needs evaluation, where s is used as partial information about the arguments $\overline{X}$ of the outermost simulation of p_1. If all this information is "consumed" then the next reaction concerning the arguments $\overline{X}$ of the outermost simulation of p_1 is the outcome of $p(s)$. This could be visualized by the following picture:



□

Moreover, the reverse is also true, i.e. $HSF_R \subseteq HSF_K$ and $SF_R \subseteq SF_K$:

Theorem 4.2.2 *The recursive (hereditarily) sequential functionals are definable by the schemata (S1)–(S8) and (S11).*

Proof. We define a Kleene-recursive functional `eval` of type $(\gamma \to \gamma), \tau_1, \ldots, \tau_k \to \gamma$ which is universal for the (hereditarily) sequential functionals over $\tau = \tau_1, \ldots, \tau_k \to \gamma$, i.e. for all $p \in SeqStr^\tau$ 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 by a finite family of functionals $\mathtt{getanswer}^A(p, \overline{X}^A, moves)$ where the list of occurrences A contains all open opposing questions in the unfinished play $moves$. Hence if $?a$ is enabled in $moves$ then $X_a \in \overline{X}^A$.

$$
\begin{aligned}
&\mathtt{getanswer}^A(p, \overline{X}^A, moves)\\
&\quad = \begin{cases} v, & \text{if } p(\mathtt{view}(moves)) = \ulcorner !v \urcorner \\ \mathtt{getanswer}^A(p, \overline{X}^A, [moves, \ulcorner ?a \urcorner, \ulcorner !\varphi \urcorner]), & \text{if } p(\mathtt{view}(moves)) = \ulcorner ?a \urcorner \\ \bot & \text{otherwise,} \end{cases}
\end{aligned}
$$

where

$$
\varphi = \begin{cases} X_a, & \text{if } a \text{ is a leaf and } X_a \in \overline{X}^A, \\[2ex] X_a\Big(\lambda \overline{X}^{a.1}.\mathtt{getanswer}^{A{:}\cup{:}a.1}(p, \overline{X}^{A{:}\cup{:}a.1}, [moves, \ulcorner ?a \urcorner, \ulcorner ?a.1 \urcorner]), \\ \qquad\qquad \vdots \\ \qquad \lambda \overline{X}^{a.l}.\mathtt{getanswer}^{A{:}\cup{:}a.l}(p, \overline{X}^{A{:}\cup{:}a.l}, [moves, \ulcorner ?a \urcorner, \ulcorner ?a.l \urcorner])\Big), \\ & \text{if } a \text{ has } l \text{ successors and } X_a \in \overline{X}^A, \\[2ex] \bot, & \text{if } X_a \notin \overline{X}^A, \end{cases}
$$

Finally we set $\texttt{eval}(p, X_1, \ldots, X_k) = \texttt{getanswer}^{[\varepsilon]}(p, X_1, \ldots, X_k, \lceil [?\varepsilon] \rceil)$. $\square$

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

Corollary 4.2.3 *The sequential functionals SF are exactly those partial continuous functionals which 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.* $\square$

Remark 4.2.4 The last result corresponds to the characterization of the serial functionals in [Saz76]: "*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. $\diamond$

5 Conclusion

The goal of finding a new model for dealing with sequential computations in higher types — a model which is well suited to analyse complexity also — has been achieved, but much work remains to be done. Actually the relation to the fully abstract model of PCF has to be worked out in more detail. It is not difficult to extend the definition of *HSF* to get a cartesian closed category. As PCF and the schemata (S1)–(S8), (S11) are strongly related [Pla66], Theorem 4.2.1 shows that we get a model for PCF. Theorem 4.2.2 shows that all compact elements of this model are PCF definable. Hence by a theorem of Milner [Mil77] this model is isomorphic to the order extensional fully abstract model of PCF. This should give a new view to this model and perhaps there are deeper insights possible with the help of sequential strategies.

Moreover we are interested in the structure of the domains HSF^τ. Can we find some order theoretical conditions characterizing the domains in *HSF* like the conditions for concrete domains [KP78] for example? Is there a manageable notion of "sequential neighbourhood" in analogy to the stable neighbourhoods in [Zha91]?

As stated in the beginning, this computation model should be a good base to study complexity in higher types.

References

[BC82] G. Berry and P.-L. Curien. Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20:265–321, 1982.

[BC85] G. Berry and P.-L. Curien. 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*, pp. 35–87, Cambridge university press, 1985.

[Buc93] Antonio Bucciarelli. *Sequential Models of PCF: Some Contributions to the Domain-Theoretic Approach to Full Abstraction.* PhD thesis, Università di Pisa-Genova-Udine, TD - 6/93, 1993.

[CF92] R. Cartwright and M. Felleisen. Observable sequentiality and full abstraction. In *19th ACM Symposium on Principles of Programming Languages*, pp. 328–342, ACM Press, 1992.

[CK90] S. A. Cook and B. M. Kapron. 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*, pages 71–96, Birkhäuser, 1990.

[Cur86] P.-L. Curien. Categorical combinators. *Information and Computation*, 69:188–254, 1986.

[Cur92] P.-L. Curien. Observable algorithms on concrete data structures. In *7th IEEE Symposium on Logic in Computer Science*, pp. 432–443, IEEE Computer Society Press, 1992.

[Ers75] Yu. L. Ershov. Theorie der Numerierungen II. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 21:473–584, 1975.

[Gan67] R. O. Gandy. Computable functionals of finite type I. In J. N. Crossley, editor, *Sets, Models and Recursion Theory (Logic Colloquium '65)*, pp. 202–242, North-Holland, 1967.

[Kle59] S. C. Kleene. Recursive functionals and quantifiers of finite types I. *Transactions of the AMS*, 91:1–52, 1959.

[Kle78] S. C. Kleene. 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*, pp. 185–222, North-Holland, 1978.

[KP78] G. Kahn and G. D. Plotkin. Structures de données concrètes. Rapport 336, IRIA-LABORIA, 1978.

[Mil77] R. Milner. Fully abstract models of typed λ-calculi. *Theoretical Computer Science*, 4:1–22, 1977.

[Pla66] R. Platek. Foundations of Recursion Theory. PhD thesis, Stanford University, 1966.

[Plo77] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[Saz76] V. Yu. Sazonov. Expressibility of functions in D. Scott's LCF language. *Algebra and Logic (English translation)*, 15:192–206, 1976.

[Sco72] D. S. Scott. Continuous lattices. In F. W. Lawvere, editor, *Toposes, Algebraic Geometry and Logic*, pp. 98–136, Springer, 1972.

[Wei85] K. Weihrauch. Type 2 recursion theory. *Theoretical Computer Science*, 38:17–33, 1985.

[Zha91] G. O. Zhang. Logic of Domains. Birkhäuser, 1991.