Semantically Linear Programming Languages

Luca Paolini
Dipartimento di Informatica
Università di Torino
10149 Torino, Italy
paolini@di.unito.it

Mauro Piccolo*
Dipartimento di Informatica
Università di Torino
10149 Torino, Italy
piccolo@di.unito.it

ABSTRACT

We propose a paradigmatic programming language (called $\mathbb{S}\ell$PCF) which is linear in a semantic sense. $\mathbb{S}\ell$PCF is not syntactically linear, namely its programs can contain more than one occurrencies of the same variable. We give an interpretation of $\mathbb{S}\ell$PCF into a model of linear coherence spaces and we show that such semantics is fully abstract with respect to our language. Furthermore, we discuss the independence of new syntactical operators and we address the universality problem.

Keywords

Linear Functions, Coherence Spaces, PCF,
Continous, Stable and Strongly Stable Functions.

1. INTRODUCTION

Coherence Spaces [19] are the result of the deep analysis of stable semantics [7] done by Jean-Yves Girard. These spaces provide a pleasant decomposition of stable functions via linear functions and exponential domain constructors. Such a decomposition is patently reflected in linear logic syntax. Since its inception, linear logic has inspired the introduction of many formal languages with sometimes different goals: resource-conscious evaluation, categorical language, implicit computational complexity, and so on.

The starting point is the least full subcategory of coherence spaces, including the infinite flat domain (representing natural numbers) and the coherence spaces of linear functions between domains in the model itself. We avoided the use of exponential domain constructors, since we want focus on linearity. We sought a programming language able to program the functions (with respect to a standard interpretation) of a the considered category (playing the role of model).

*Also affiliated to Preuves, Programmes et Systèmes, Université Paris VII, CNRS UMR 7126, France

10th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming, 15-17 July 2008 Valencia, Spain.

Two programs are operationally equivalent whenever they are interchangeable "in all contexts" without affecting the observable outcome of the computation (this equivalence is also called contextual or observational). In contrast, according to a denotational semantics the meaning of a program lies in its denotation; hence, two programs are denotationally equivalent in a given model only when they have the same denotation in the model itself. If the denotational equivalence implies the operational one, then the model is *correct*. If the two equivalences coincide then the model is *fully abstract*.

We propose $\mathbb{S}\ell$PCF, acronym of $\mathbb{S}$emantically-ℓinear-PCF, a language based on a syntactic restriction of PCF togheter with a new operator. The considered linear model is indeed fully abstract for our language. It should be clear that the considered linear model is not correct (w.r.t. standard interpretations) for almost all languages inspired to linear logic, since they are able to programs all stable functions between coherence spaces. More precisely, the considered model is correct for the "linear core" of these languages. A preliminary version of $\mathbb{S}\ell$PCF, without full abstraction and independence properties, was presented in [17].

Linear functions are strict in all their arguments. Except under conditionals, high-order variables (having an arrow as type) occur just once in terms. The conditional requires the use of the same high-order variables in both its branches. So strictness for higher-order abstractions (terms abstracting an arrow-typed variable) can be assured by the operational evaluation. The case of first-order abstractions (terms abstracting a ground variable) is more complex. Let $\mathbf{N}$ be the infinite flat domain representing natural numbers. Let $\lambda\!\!\lambda x^{\mathbf{N}}.n$ be the non-strict function associating the natural number n to all possible inputs (also undefined ones), namely an erasing function. Linear endo-functions on a flat domain are *all the stable functions, except the erasing ones.* In other words, the union of the set $\{\lambda\!\!\lambda x^{\mathbf{N}}.n | n \in \mathbb{N}\}$ of erasing functions with the set of linear function $\mathbf{N} \multimap \mathbf{N}$ is one-to-one with the space $!\mathbf{N} \multimap \mathbf{N}$ of stable functions. In order to represent all functions in $\mathbf{N} \multimap \mathbf{N}$, in our language, we do not put any constraints on the occurrences of ground variables. They can occur more than once or zero times. The strictness on first-order abstractions (abstracting ground variables) is provided by applying a call-by-value parameter passing policy. Note that this liberality on the management of ground variables morally entails that we may also use a high-order variable more than once, provided that we always apply it to the same sequence of arguments. More precisely, the result of the evaluation of a ground term (possibly containing high-order subterms) can be either erased or duplicated We introduced a third kind of variables, namely stable variables. They are used only in order to can program a linear recursion. These key points behind $\mathcal{S}\ell$PCF imply the correctness of our purely linear model.

This analysis teaches us that linearity can be considered in many respects. Three main kinds of linearity emerge: syntactical, operational and denotational. The syntactical one claims a linear use of variables in terms. This kind of linearity is sometimes considered the computational counterpart of the linear logic linearity. $\mathcal{S}\ell$PCF is not linear in this syntactical sense. However, $\mathcal{S}\ell$PCF is linear in a denotational sense, since only purely linear functions can be defined in the language, without any kind of exponentiation. Finally, if redexes are not duplicated during the evaluation, then a notion of operational linearity arises. $\mathcal{S}\ell$PCF is not endowed with a linear operational evaluation, although the restriction $\mathcal{S}\ell$PCF to terms without fixpoints (its finitary fragment) is operationally linear. Operational linearity is related to the notion of *simple term* [24] in λ-calculus, suggesting a more general linearity than the operational one, unrelated from a specific strategy.

The main results of this paper are the very simple proof of full abstraction and the proof of independence of new construct of $\mathcal{S}\ell$PCF. A very general motivation behind our results is that linearity which is embedded in our language is more generous than the strict syntactical one, so it can be used in order to improve resource-conscious results related to Linear Logic. $\mathcal{S}\ell$PCF is Turing-Complete. However, type-respecting recursive functions in StPCF are strictly less than that of the languages StPCF [29] and PCF+H [26]. In other words, linear functions between two coherence spaces are less than stable functions between the same domains. Since the syntactical constraints of $\mathcal{S}\ell$PCF forbid useless duplications of redexes, we are utterly convinced that $\mathcal{S}\ell$PCF can be fruitfully exploited in research fields like optimal evaluation (see for instance [2, 31, 34]), implicit computational complexity (see for instance [4, 5, 14]), linear computation (see for instance [1, 13]). Further motivations are theoretical. Our study is relevant from an higher-type computability point of view, since we introduce new operators. In [30], we are studying processes behind our programming language by translating programs in calculus of Solos [25]. Such a translation is related to the description of game-semantics done in [6, 23, 35], by using the π-calculus [33]. We plan also to study correspondences between Ludics [20] and the linear model, considered above.

Outline of the paper.

Section 2 presents $\mathcal{S}\ell$PCF, its operational semantics. Moreover, we discuss the calculus behind the language. Section 3 tackles some remarks on `which`?, a syntactical operator of $\mathcal{S}\ell$PCF. Section 4 recalls basic notions on coherence spaces, together with the subsections proving interpretation, correctness and completeness. Section 5 discuss further extensions of $\mathcal{S}\ell$PCF which are correct for our linear model, conclusions and conjectures. In its subsections, we introduce strongly-stable models and a Boolean version of a second-order Gustave-like OR operator.

2. A SEMANTICALLY LINEAR PROGRAMMING LANGUAGE

$\mathcal{S}\ell$PCF is a PCF-like language with implicit truth-values which are coded on integers (zero codes "true" while any other numeral stands for "false").

Definition 1. The set $\mathbb{T}$ of *linear types* is defined as follows: $\sigma, \tau ::= \iota \mid (\sigma \multimap \tau)$ where ι is the only *ground* type and $\sigma, \tau, ...$ are metavariables ranging over types.

As customary $\multimap$ associates to right. Hence $\sigma_1 \multimap \sigma_2 \multimap \sigma_3$ is an abbreviation for $\sigma_1 \multimap (\sigma_2 \multimap \sigma_3)$. Furthermore, it is easy to see that all types τ have shape $\tau_1 \multimap ... \multimap \tau_n \multimap \iota$, for some type $\tau_1, ..., \tau_n$ where $n \geq 0$.
Let $\mathrm{Var}^\sigma, \mathrm{SVar}^\sigma$ be numerable sets of variables of type σ. The set of *high-order* variables is $\mathrm{HVar} = \bigcup_{\sigma,\tau\in\mathbb{T}} \mathrm{Var}^{\sigma\multimap\tau}$ and the whole set of variables is $\mathrm{Var} = \mathrm{Var}^\iota \cup \mathrm{HVar} \cup \mathrm{SVar}$. Letters $\mathtt{x}^\sigma, \mathtt{y}^\sigma, \mathtt{z}^\sigma, \ldots$ range over variables in Var^σ while $\mathcal{F}^\sigma_0, \mathcal{F}^\sigma_1, \mathcal{F}^\sigma_2, \ldots$ ranges over stable variables, namely variables in SVar^σ. Last, $\varkappa$ will denote any kind of variables. Latin letters $\mathtt{M}, \mathtt{N}, \mathtt{L}, \ldots$ range over terms.

Definition 2. Let $\Gamma \subseteq \mathrm{HVar}$. Typed *terms* of $\mathcal{S}\ell$PCF are defined by using a type assignment proving judgments of the shape $\Gamma \vdash \mathtt{M} : \sigma$, in Table 1.

Note that both ground and stable variables can occur freely in our language. They belong to distinct kinds only for sake of simplicity. Except for the ℓ`if` construction, high-order variables are treated linearly.

Sometimes types will be omitted when they are clear from the context or uninteresting (note that given types of all variables of a term $\mathtt{M}$, there is a unique σ such that $\mathtt{M}^\sigma$). Sometimes, parentheses are omitted, always by respecting the following conventions: application associates to the left and application binds more tightly than abstraction, i.e. $\lambda\mathtt{x}.\mathtt{MNL} = (\lambda\mathtt{x}.((\mathtt{MN})\mathtt{L}))$. Free variables of terms are defined as expected. A term $\mathtt{M}$ is *closed* if and only if $\mathrm{FV}(\mathtt{M}) = \emptyset$, otherwise $\mathtt{M}$ is *open*. Terms are considered up to α-equivalence, namely a bound variable can be renamed provided no free variable is captured. Moreover, as usual, $\mathtt{M}[\mathtt{N}/\varkappa]$ denotes the capture-free substitution of all free occurrences of $\varkappa$ in $\mathtt{M}$ by $\mathtt{N}$. We will write $\underline{\mathtt{n}}$ for $\mathtt{succ}(\cdots(\mathtt{succ}(\underline{\mathtt{0}}))\cdots)$ where $\mathtt{succ}$ is applied n-times to $\underline{\mathtt{0}}$. Let $\mathcal{P} = \{\mathtt{M}^\iota \in \mathcal{S}\ell\mathrm{PCF} \mid \mathrm{FV}(\mathtt{M}^\iota) = \emptyset\}$ be the set of *programs* and let $\mathcal{N} = \{\underline{\mathtt{0}}, \ldots, \underline{\mathtt{n}}, \ldots\}$ be the set of *numerals*.

In order to define the evaluation of the language, we need pairing and projections operators on natural numbers. If $m, n \in \mathbb{N}$ then $2^m(2n+1) - 1$ is our pairing function. Projections from $z \in \mathbb{N}$ can be defined by functions

$$\min_{x\leq z}[(\exists y)_{\leq z}(z =< x, y >)]$$
$$\min_{y\leq z}[(\exists x)_{\leq z}(z =< x, y >)].$$

Such functions induce a bijection, details can be found either, in p.41,p.73 of [12] or in p.47 of [15].

$$\frac{}{\_ \vdash \underline{\mathtt{n}} : \iota} \qquad \frac{}{\_ \vdash \mathtt{succ} : \iota \multimap \iota} \qquad \frac{}{\_ \vdash \mathtt{pred} : \iota \multimap \iota} \qquad \frac{}{\_ \vdash \mathtt{which?} : ((\iota \multimap \iota) \multimap \iota) \multimap \iota} \qquad \frac{\Gamma, \mathtt{f}^\sigma \vdash \mathtt{M} : \tau}{\Gamma \vdash \lambda \mathtt{f}^\sigma.\mathtt{M} : \sigma \multimap \tau}$$

$$\frac{\mathtt{x} \in \mathrm{Var}^\iota}{\_ \vdash \mathtt{x} : \iota} \qquad \frac{}{\mathtt{f}^{\sigma \multimap \tau} \vdash \mathtt{f} : \sigma \multimap \tau} \qquad \frac{\digamma \in \mathrm{SVar}^\sigma}{\_ \vdash \digamma : \sigma} \qquad \frac{\_ \vdash \mathtt{M} : \sigma \quad \digamma \in \mathrm{SVar}^\sigma}{\_ \vdash \mu\digamma.\mathtt{M} : \sigma} \qquad \frac{\Gamma \vdash \mathtt{M} : \tau}{\Gamma \vdash \lambda \mathtt{x}^\iota.\mathtt{M} : \iota \multimap \tau}$$

$$\frac{\Gamma_\mathtt{M} \cap \Gamma = \emptyset \quad \Gamma_\mathtt{M} \vdash \mathtt{M} : \iota \quad \Gamma \vdash \mathtt{L} : \iota \quad \Gamma \vdash \mathtt{R} : \iota}{\Gamma_\mathtt{M} \cup \Gamma \vdash \ell\mathtt{if}\ \mathtt{M}^\iota\ \mathtt{L}^\iota\ \mathtt{R}^\iota : \iota} \qquad \frac{\Gamma_\mathtt{M} \cap \Gamma_\mathtt{N} = \emptyset \quad \Gamma_\mathtt{M} \vdash \mathtt{M} : \sigma \multimap \tau \quad \Gamma_\mathtt{N} \vdash \mathtt{N} : \sigma}{\Gamma_\mathtt{M} \cup \Gamma_\mathtt{N} \vdash \mathtt{MN} : \tau}$$

Table 1: $\mathcal{S}\ell$PCF: $\mathcal{S}$emantically-ℓinear PCF.

Definition 3. The *evaluation relation* $\Downarrow \subseteq \mathcal{P} \times \mathcal{N}$ is the effective relation inductively defined by the rules of Table 2. If there exists a numeral $\underline{\mathtt{n}}$ such that $\mathtt{M} \Downarrow \underline{\mathtt{n}}$ then we say that $\mathtt{M}$ *converges*, and we write $\mathtt{M} \Downarrow$, otherwise we say that it *diverges*, and we write $\mathtt{M} \Uparrow$.

A restriction of $\mathcal{S}\ell$PCF endowed with an operational semantics, has been presented in [17]. The relation $\Downarrow$ implements a *call-by-value* parameter passing policy in the ground case and *call-by-name* parameter passing policy in the high-order case.

Syntax deserves some discussion. We remark that $\mathtt{pred}$ is a partial operator, namely $\mathtt{pred}\,\underline{\mathtt{0}}$ diverges. The evaluation of $\ell\mathtt{if}\ \mathtt{M}^\iota\ \mathtt{L}^\iota\ \mathtt{R}^\iota$ asks to evaluate exactly one subterm between $\mathtt{L}^\iota$ and $\mathtt{R}^\iota$. The evaluation of $\mathtt{which?}\,\mathtt{M}$ is a pattern for infinite rules, similarly to rules for the operator $\exists$ of Plotkin [32]. Note that $\mathtt{which?}\,\mathtt{M} \Downarrow$ if and only if $\mathtt{M}(\lambda \mathtt{x}^\iota.\mathtt{x}) \Downarrow$; moreover, linearity assures that $\mathtt{M}$ applies its argument $\lambda \mathtt{x}^\iota.\mathtt{x}$ exactly to a unique numeral $\underline{\mathtt{k}}$, in Table 2. A deterministic evaluation can be formalized likewise that of the operator $\mathtt{strict?}$ presented in [29]. Since first-order strict stable functions are linear, we need to add a fixpoint operator to our language. Unfortunately, the least fixpoint of a linear function is always the bottom of the considered domain, because strictness. In order to overcome this problem we recall the fixpoint theorem [22].

THEOREM 1. *Let D be a cpo. If $f : D \to D$ is continuous then it has a least fixed point $\mathtt{fix}(f) \in D$.*

We introduced a special kind of variables, named stable variables. Those variables are used in order to program continuous (w.r.t. stable order) non-strict functions from a linear coherence space L to itself, so their fixpoints will be elements of L. Syntactically, a stable variable will be used without linear constraints. We do not permit to λ-abstract those variables, since they will be used only in order to obtain fixpoints. Since Turing-Completeness was proved for a restriction of $\mathcal{S}\ell$PCF (without $\mathtt{which?}$) in [17], we can program pairing and projections in the remaining language as done in Table 2.

LEMMA 1. *Let $\mathtt{M}^\tau, \mathtt{N}^\sigma \in \mathcal{S}\ell$PCF.*

1. *If* $\mathrm{HVar} \cap \mathrm{FV}(\mathtt{M}^\tau) \cap \mathrm{FV}(\mathtt{N}^\sigma) = \emptyset$ *and* $\mathtt{x}^\sigma \in \mathrm{HFV}(\mathtt{M}^\tau)$ *then* $\mathtt{M}^\tau[\mathtt{N}^\sigma/\mathtt{x}^\sigma] \in \mathcal{S}\ell$PCF.
2. *If* $\mathrm{FV}(\mathtt{N}^\sigma) \subseteq \mathrm{SVar} \cup \mathrm{Var}^\iota$ *then* $\mathtt{M}^\tau[\mathtt{N}^\sigma/\digamma^\sigma] \in \mathcal{S}\ell$PCF.
3. *If* $\mathtt{N}$ *does not contain high-order free variables and* $\varkappa_1, \ldots, \varkappa_n \in \mathrm{Var}$ *then* $\mathtt{M}[\mathtt{N}/\varkappa_1, \ldots, \mathtt{N}/\varkappa_n] \in \mathcal{S}\ell$PCF.

PROOF. Easy, by induction on terms. □

Lemma 1 implies that evaluation is well-defined.

Definition 4. Let $[^\sigma]$ be a *special constant* of type σ. The set of σ-*context* Ctx_σ is generated by the following grammar:

$$\begin{array}{lll} \mathtt{C}[^\sigma] & ::= & [^\sigma] \mid \varkappa^\tau \mid \digamma^\tau \mid \underline{\mathtt{0}} \mid \mathtt{succ} \mid \mathtt{pred} \mid \mathtt{which?} \\ & \mid & \ell\mathtt{if}\ \mathtt{C}[^\iota]\ \mathtt{C}[^\iota]\ \mathtt{C}[^\iota] \mid (\lambda\varkappa.\mathtt{C}[^\sigma]) \mid (\mathtt{C}[^\sigma]\mathtt{C}[^\sigma]) \mid \mu\digamma.\mathtt{C}[^\sigma] \end{array}$$

$\mathtt{C}[\mathtt{N}^\sigma]$ denotes the result obtained by replacing all the occurrences of $[^\sigma]$ in the context $\mathtt{C}[^\sigma]$ by the term $\mathtt{N}^\sigma$ and by allowing the capture of its free variables.

Clearly, $\mathtt{N}^\sigma \in \mathcal{S}\ell$PCF and $\mathtt{C}[^\sigma] \in \mathrm{Ctx}_\sigma$ doesn't imply that $\mathtt{C}[\mathtt{N}^\sigma] \in \mathcal{S}\ell$PCF.

Definition 5. (OPERATIONAL EQUIVALENCE). Let $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathcal{S}\ell$PCF.

1. $\mathtt{M} \lessapprox_\sigma \mathtt{N}$ whenever, for all $\mathtt{C}[^\sigma]$ s.t. $\mathtt{C}[\mathtt{M}], \mathtt{C}[\mathtt{N}] \in \mathcal{P}$, if $\mathtt{C}[\mathtt{M}] \Downarrow \underline{\mathtt{n}}$ then $\mathtt{C}[\mathtt{N}] \Downarrow \underline{\mathtt{n}}$.
2. $\mathtt{M} \approx_\sigma \mathtt{N}$ if and only if $\mathtt{M} \lessapprox_\sigma \mathtt{N}$ and $\mathtt{N} \lessapprox_\sigma \mathtt{M}$.

It is easy to verify that $\lessapprox_\sigma$ is a preorder while $\approx_\sigma$ is a congruence. It is useful to name some terms. We put $\Omega^\iota = \mu\digamma^\iota.\digamma^\iota$. Moreover, if $\sigma_0 = \mu_1 \multimap \ldots \multimap \mu_\mathtt{m} \multimap \iota$ for some $\mathtt{m} \in \mathbb{N}$, then

$$\Omega^{\sigma_0 \multimap \ldots \multimap \sigma_\mathtt{n} \multimap \iota} = \lambda \mathtt{x}_0^{\sigma_0} \ldots \mathtt{x}_\mathtt{n}^{\sigma_\mathtt{n}}. \\ \ell\mathtt{if}(\Omega^{\sigma_1 \multimap \ldots \multimap \sigma_\mathtt{n} \multimap \iota}\mathtt{x}_1^{\sigma_1} \ldots \mathtt{x}_\mathtt{n}^{\sigma_\mathtt{n}})(\mathtt{x}_0\Omega^{\mu_1} \ldots \Omega^{\mu_\mathtt{m}})(\mathtt{x}_0\Omega^{\mu_1} \ldots \Omega^{\mu_\mathtt{m}}).$$

By using Ω^σ it is possible to define approximants of a term having shape $\mu\digamma.\mathtt{M}^\sigma$ as follows,

$$\mu^0\digamma.\mathtt{M}^\sigma = \Omega^\sigma, \qquad \mu^{n+1}\digamma.\mathtt{M}^\sigma = \mathtt{M}[\mu^n\digamma.\mathtt{M}/\digamma].$$

LEMMA 2. *Let $\mathtt{M}_0^{\sigma_0}, \ldots, \mathtt{M}_\mathtt{m}^{\sigma_\mathtt{m}}$ be a sequence of closed terms ($m \geq 0$).*

$$\frac{}{\underline{0} \Downarrow \underline{0}}\ (0) \qquad \frac{\mathtt{M} \Downarrow \underline{\mathtt{n}}}{\mathtt{succ\,M} \Downarrow \mathtt{succ}\,\underline{\mathtt{n}}}\ (s) \qquad \frac{\mathtt{M[N/x]P_1 \cdots P_i} \Downarrow \underline{\mathtt{n}}}{(\lambda \mathtt{x}^{\sigma \multimap \tau}.\mathtt{M})\mathtt{NP_1 \cdots P_i} \Downarrow \underline{\mathtt{n}}}\ (\lambda^{\multimap}) \qquad \frac{\mathtt{N} \Downarrow \underline{\mathtt{m}} \quad \mathtt{M}[\underline{\mathtt{m}}/\mathtt{x}]\mathtt{P_1 \cdots P_i} \Downarrow \underline{\mathtt{n}}}{(\lambda \mathtt{x}^{\iota}.\mathtt{M})\mathtt{NP_1 \cdots P_i} \Downarrow \underline{\mathtt{n}}}\ (\lambda^{\iota}) \qquad \frac{\mathtt{M}[\mu \mathcal{F}.\mathtt{M}/\mathcal{F}]\mathtt{P_1 \cdots P_i} \Downarrow \underline{\mathtt{n}}}{(\mu \mathcal{F}.\mathtt{M})\mathtt{P_1 \cdots P_i} \Downarrow \underline{\mathtt{n}}}\ (\mu)$$

$$\frac{\mathtt{M} \Downarrow \underline{0} \quad \mathtt{L} \Downarrow \underline{\mathtt{m}}}{\ell\mathtt{if\ M\ L\ R} \Downarrow \underline{\mathtt{m}}}\ (\mathrm{if}_l) \qquad \frac{\mathtt{M} \Downarrow \mathtt{succ}(\underline{\mathtt{n}}) \quad \mathtt{R} \Downarrow \underline{\mathtt{m}}}{\ell\mathtt{if\ M\ L\ R} \Downarrow \underline{\mathtt{m}}}\ (\mathrm{if}_r) \qquad \frac{\mathtt{M} \Downarrow \mathtt{succ}\,\underline{\mathtt{n}}}{\mathtt{pred\,M} \Downarrow \underline{\mathtt{n}}}\ (p_n) \qquad \frac{\mathtt{M}(\lambda \mathtt{x}^{\iota}.\ell\mathtt{if}\,(\overbrace{\mathtt{pred} \ldots \mathtt{pred}}^{\underline{\mathtt{k}}}\ \mathtt{x})\ \underline{\mathtt{k}}\ \Omega^{\iota}) \Downarrow \underline{\mathtt{n}} \quad \ulcorner \underline{\mathtt{n}}, \underline{\mathtt{k}} \urcorner \Downarrow \underline{\mathtt{r}}^{\dagger}}{\mathtt{which?}\,\mathtt{M}^{(\iota \multimap \iota) \multimap \iota} \Downarrow \underline{\mathtt{r}}}\ (\mathtt{w}^{\underline{\mathtt{k}}})$$

† We write $\ulcorner \underline{\mathtt{n}}, \underline{\mathtt{m}} \urcorner$ as an abbreviation, for the application of a term coding the pairing function to $\underline{\mathtt{n}}$ and $\underline{\mathtt{m}}$. Clearly, $\mathtt{Sum} \equiv \mu \mathcal{F}.\lambda \mathtt{x}^{\iota}\mathtt{y}^{\iota}.\ell\mathtt{if\ x\ y}\ (\mathcal{F}\ (\mathtt{pred\,x})\ (\mathtt{succ\,y}))$ defines the addition, $\mathtt{Prod} \equiv \mu \mathcal{F}.\lambda \mathtt{x}^{\iota}\mathtt{y}^{\iota}.\ell\mathtt{if\ x}\ \underline{0}\ (\mathtt{Sum\,y}\,(\mathcal{F}\ (\mathtt{pred\,x})\ \mathtt{y}))$ defines the multiplication and $\mathtt{Exp2} \equiv \mu \mathcal{F}.\lambda \mathtt{x}^{\iota}.\ell\mathtt{if\ x}\ \underline{1}\ (\mathtt{Prod}\,\underline{2}\,(\mathcal{F}\ (\mathtt{pred\,x})))$ defines the exponentiation of base 2. Hence, $\lambda \mathtt{x}^{\iota}\mathtt{y}^{\iota}.\mathtt{pred}\big(\mathtt{Prod}\,(\mathtt{Exp2\,x})(\mathtt{succ}(\mathtt{Prod}\,\underline{2}\,\mathtt{y}))\big)$ defines the pairing function as syntactic sugar.

Table 2: Operational Semantics of $\mathcal{S}\ell$PCF.

1. *$\Omega^{\sigma_0 \multimap \cdots \multimap \sigma_m \multimap \iota}\mathtt{M_0 \ldots M_m}$ is a program and $\Omega^{\sigma_0 \multimap \cdots \multimap \sigma_m \multimap \iota}\mathtt{M_0 \ldots M_m} \Uparrow_{\mathtt{e}}$.*
2. *Let $(\mu \mathcal{F}.\mathtt{P}^{\sigma})\mathtt{M_0 \ldots M_m}$ be a program. $(\mu \mathcal{F}.\mathtt{P}^{\sigma})\mathtt{M_0 \ldots M_m} \Downarrow \underline{\mathtt{n}}$ if and only if $(\mu^{k+1}\mathcal{F}.\mathtt{P}^{\sigma})\mathtt{M_0 \ldots M_m} \Downarrow \underline{\mathtt{n}}$, for some $k \in \mathbb{N}$.*

2.1 A Semantically Linear Lambda-Calculus

It is good to make a clear distinction between reduction systems and programming languages, as remarked in [28, pp.283]. A reduction system is a (formal) language together with some rewriting rules. A programming language may be regarded as a reduction system with some additional features. It is an essential feature of a programming language to be equipped with a specified reduction strategy.

In this Section, we present the calculus behind $\mathcal{S}\ell$PCF. Let $\mathrm{HFV}(\mathtt{M}) = \mathrm{FV}(\mathtt{M}) \cap \mathrm{HVar}$. The language of $\mathcal{S}\ell$PCF and their sets of free variables (we use FV as notation) are mutually (re-)defined in Table 3.

In Definition 4, we defined the notion of context. In what follows, we need some restricted notions of context. We call **μ-lazy context**, contexts without holes under μ-abstractions. Moreover, we define **ℓif-lazy context**, ranging over $\mathtt{W}[^{\sigma}]$, as follows

$$\begin{aligned}\mathtt{W}[^{\sigma}] ::= \ & [^{\sigma}] \mid \varkappa^{\tau} \mid \mathcal{F}^{\tau} \mid \underline{0} \mid \mathtt{succ} \mid \mathtt{pred} \mid \mathtt{which?} \\ & \mid \ell\mathtt{if}\ \mathtt{W}[^{\sigma}]\ \mathtt{L\ R} \mid (\lambda \varkappa.\mathtt{W}[^{\sigma}]) \mid (\mathtt{W}[^{\sigma}]\mathtt{W}[^{\sigma}]) \mid \mu \mathcal{F}.\mathtt{W}[^{\sigma}]\ .\end{aligned}$$

Definition 6. (Reduction Rules) We denote $\to_{\mathcal{S}\ell}$ the firing everywhere in any context, of one of the following rules

$$\begin{array}{ll} (\lambda \mathtt{x}^{\sigma \multimap \tau}.\mathtt{M})\mathtt{N} \rightsquigarrow_{\beta} \mathtt{M[N/x]} & (\lambda \mathtt{x}^{\iota}.\mathtt{M})\underline{\mathtt{n}} \rightsquigarrow_{\beta_v} \mathtt{M}[\underline{\mathtt{n}}/\mathtt{x}] \\ \mu \mathcal{F}.\mathtt{M} \rightsquigarrow_{\mathtt{Y}} \mathtt{M}[\mu \mathcal{F}.\mathtt{M}/\mathcal{F}] & \mathtt{pred}\ (\mathtt{succ}\ \underline{\mathtt{n}}) \rightsquigarrow_{\delta} \underline{\mathtt{n}} \\ \ell\mathtt{if}\ \underline{0}\ \mathtt{L\ R} \rightsquigarrow_{\delta} \mathtt{L} & \ell\mathtt{if}\ \underline{\mathtt{n+1}}\ \mathtt{L\ R} \rightsquigarrow_{\delta} \mathtt{R} \\ \mathtt{which?}\,(\lambda \mathtt{f}^{\iota \multimap \iota}.\mathtt{W}[\mathtt{f}^{\iota \multimap \iota}\underline{\mathtt{k}}]) \rightsquigarrow_{\delta} \ulcorner \mathtt{W}[\underline{\mathtt{k}}], \underline{\mathtt{k}} \urcorner\ , & \\ & \text{where } \mathtt{W}[^{\iota}] \text{ is a } \ell\mathtt{if}\text{-lazy context.} \end{array}$$

We denote $\rightsquigarrow$ the binary relation on terms obtained as union of the relation-rules $\rightsquigarrow_{\beta}, \rightsquigarrow_{\beta_v}, \rightsquigarrow_{\mathtt{Y}}, \rightsquigarrow_{\delta}$ (without any context-closure). Moreover, we denote $\to^*_{\mathcal{S}\ell}$ and $=_{\mathcal{S}\ell}$ respectively, the reflexive and transitive closure of $\to_{\mathcal{S}\ell}$ and the reflexive, symmetric and transitive closure of $\to_{\mathcal{S}\ell}$. Redexes of our calculus are the left-sides of relation-rules defined above.

Clearly, only well-typed ℓ`if`-lazy contexts play a role in the definition of `which?`-rule. Always in the `which?`-rule, the first-order variable `f` abstracted is the same variable filling the hole of the context $\mathtt{W}[^{\iota}]$: therefore, we mean that `f` cannot bounded from the context. Whence, we can restrict $\mathtt{W}[^{\iota}]$ to be μ-lazy, since the μ-constraints of $\mathcal{S}\ell$PCF prevents a μ-abstraction on the hole. Last but not least, we remark that the linearity on `f` assure that the hole is unique.

As done in [8], it is easy to prove properties as the post-position of δ-rules in a sequence of reductions, the confluence and a standardization theorem.

Lemma 3. *Let $\mathtt{M}^{\sigma}, \mathtt{N}^{\sigma}$ be terms of $\mathcal{S}\ell$PCF such that $\mathtt{M} \rightsquigarrow \mathtt{N}$ and $\mathtt{C[M]}, \mathtt{C[N]} \in \mathcal{P}$, for a context $\mathtt{C}[^{\sigma}]$. If $\mathtt{C[N]} \Downarrow \underline{\mathtt{n}}$ then $\mathtt{C[M]} \Downarrow \underline{\mathtt{n}}$.*

Proof. The proof can be done by induction on the shape of contexts, by considering in each case the last rule applied on the derivation proving $\mathtt{C[N]} \Downarrow \underline{\mathtt{n}}$. □

Let $\mathtt{M} \in \mathcal{P}$ and $\underline{\mathtt{n}}$ be a numeral, if $\mathtt{M} \to^*_{\mathcal{S}\ell} \underline{\mathtt{n}}$ according to the leftmost strategy then $\mathtt{M} \Downarrow \underline{\mathtt{n}}$. Something stronger holds, since the confluence of our calculus.

Theorem 2. *$\mathtt{M} \Downarrow \underline{\mathtt{n}}$ if and only if $\mathtt{M} \to^*_{\mathcal{S}\ell} \underline{\mathtt{n}}$, for all term $\mathtt{M}$.*

Proof. $\mathtt{M} \Downarrow \underline{\mathtt{n}}$ implies $\mathtt{M} \to^*_{\mathcal{S}\ell} \underline{\mathtt{n}}$ is trivial. The other direction follows by induction on the number of reduction steps of $\mathtt{M} \to^*_{\mathcal{S}\ell} \underline{\mathtt{n}}$. The case of zero steps is immediate. The inductive case follows by Lemma 3. □

Following the suggestion presented in the Introducttion, we remark that $\mathcal{S}\ell$PCF is not syntactically linear but it is denotationally linear, moreover its finitary fragment (i.e. terms avoiding use of the recursion operator) is operationally linear. A term `M` is *simple* (see [24]), if in no reduction of `M`, a redex is multiplied. Although $\mathcal{S}\ell$PCF is operationally linear in the finitary fragment, there are $\mathcal{S}\ell$PCF-terms non simple. For example, $\mathtt{M} = (\lambda \mathtt{f}.\ell\mathtt{if\ x\ f}\ \underline{3}\ \mathtt{f}\ \underline{5})\mathtt{N}$ and $\mathtt{N} = (\lambda \mathtt{x}^{\iota}\lambda \mathtt{y}^{\iota}.\mathtt{x})\underline{9}$ are both $\mathcal{S}\ell$PCF-term, but `M` is not simple. In fact $\mathtt{M} \to_{\mathcal{S}\ell} \ell\mathtt{if\ x}\ (\mathtt{N}\,\underline{3})\ (\mathtt{N}\,\underline{5})$. We note that *simple terms* of λ-calculus

Term	*Condition*	Free Variables
$\underline{0}^\iota$		$FV(\underline{0}^\iota)=\emptyset$
$\mathtt{succ}^{\iota\multimap\iota}$		$FV(\mathtt{succ}^{\iota\multimap\iota})=\emptyset$
$\mathtt{pred}^{\iota\multimap\iota}$		$FV(\mathtt{pred}^{\iota\multimap\iota})=\emptyset$
$\mathtt{which?}^{(((\iota\multimap\iota)\multimap\iota)\multimap\iota)}$		$FV(\mathtt{which?})=\emptyset$
$\varkappa^\sigma$	if $\varkappa^\sigma\in \mathrm{Var}^\sigma\cup \mathrm{SVar}^\sigma$	$FV(\varkappa^\sigma)=\{\varkappa^\sigma\}$
$(\lambda \mathtt{x}^\sigma.\mathtt{M}^\tau)^{\sigma\multimap\tau}$	if $\mathtt{x}^\sigma\in \mathrm{Var}^\iota$ or $\mathtt{x}^\sigma\in HFV(\mathtt{M}^\tau)$	$FV((\lambda \mathtt{x}^\sigma.\mathtt{M}^\tau)^{\sigma\multimap\tau})=FV(\mathtt{M}^\tau)-\{\mathtt{x}^\sigma\}$
$(\mathtt{M}^{\sigma\multimap\tau}\mathtt{N}^\sigma)^\tau$	if $HFV(\mathtt{M}^{\sigma\multimap\tau})\cap HFV(\mathtt{N}^\sigma)=\emptyset$	$FV((\mathtt{M}^{\sigma\multimap\tau}\mathtt{N}^\sigma)^\tau)$ $=FV(\mathtt{M}^{\sigma\multimap\tau})\cup FV(\mathtt{N}^\sigma)$
$(\ell\mathtt{if}\ \mathtt{M}^\iota\ \mathtt{L}^\iota\ \mathtt{R}^\iota)^\iota$	if $HFV(\mathtt{L}^\iota)=HFV(\mathtt{R}^\iota)$, $HFV(\mathtt{M}^\iota)\cap HFV(\mathtt{R}^\iota)=\emptyset$ and	$FV((\ell\mathtt{if}\ \mathtt{M}^\iota\ \mathtt{L}^\iota\ \mathtt{R}^\iota)^\iota)$ $=FV(\mathtt{M}^\iota)\cup FV(\mathtt{L}^\iota)\cup FV(\mathtt{R}^\iota)$
$(\mu F^\sigma.M^\sigma)^\sigma$	if $F^\sigma\in SFV(\mathtt{M}^\sigma)$ and $HFV(\mathtt{M}^\sigma)=\emptyset$	$FV((\mu F^\sigma.\mathtt{M}^\sigma)^\sigma)=FV(\mathtt{M}^\sigma)\backslash\{F^\sigma\}$

Table 3: Syntax of $\mathbb{S}$emantically-ℓinear PCF.

suggest a fourth kind of linearity, namely linear reduction (i.e. all terms of the considered calculus are simple).

In [30], we introduce ℓinSolos, i.e. a typed process calculus based on the calculus of solos [25]. We use ℓinSolos in order to express computational processes generated by the calculus presented here. We define an interpretation of $\mathbb{S}\ell$PCF on ℓinSolos and we discuss how to process redexes of $\mathbb{S}\ell$PCF in a parallel way. More precisely, we show how to simulate any (single) reduction of $\mathbb{S}\ell$PCF in ℓinSolos (no reduction strategy is forced in the interpretation). Afterward, we prove that a suitable observational equivalence between processes is correct w.r.t the operational semantics of $\mathbb{S}\ell$PCF, via our interpretation. Results in [30] are deeply related to that presented in [6, 23, 35].

3. THE WHICH? OPERATOR

A novelty of $\mathbb{S}\ell$PCF is which?. This higher order operator deserves some discussions.

3.1 Programming with which?

The operator which? of $\mathbb{S}\ell$PCF plays a role similar to strict? of StPCF [29]. Likewise, we can use which? to define some useful primitives in $\mathbb{S}\ell$PCF in order to deal with a simple kind of exceptions in a linear setting.
First of all, let us introduce the following operator called @wh? : $((\iota\multimap\iota)\multimap\iota)\multimap(\iota\multimap\iota)\multimap\iota$ with the following operational semantics.

$$\frac{\mathtt{M}(\lambda\mathtt{x}^\iota.\ell\mathtt{if}(\overbrace{\mathtt{pred}\dots\mathtt{pred}}^{\underline{k}}\ \mathtt{x})\ (\mathtt{N}\underline{k})\ \Omega^\iota)\Downarrow\underline{n}\quad \lceil\underline{n},\underline{k}\rceil\Downarrow\underline{r}}{@\mathtt{wh?}\,\mathtt{M}^{(\iota\multimap\iota)\multimap\iota}\mathtt{N}^{\iota\multimap\iota}\Downarrow\underline{r}}\ (@\mathtt{wh?}^{\underline{k}})$$

Note that @wh? $\mathtt{M}\ \mathtt{N}\Downarrow$ if and only if $\mathtt{M}\,\mathtt{N}\Downarrow$. This control operator gives back the result of the application of the functional M with the function N, together with the unique numeral passed by M as argument to N.
If $\mathtt{M}^{(\iota\multimap\iota)\multimap\iota}$ is a term then @wh? M $(\lambda\mathtt{x}^\iota.\mathtt{x})$ behaves in the same way as which? M. So @wh? may appear more powerful than which?. However the term

$$\lambda\mathtt{f}^{(\iota\multimap\iota)\multimap\iota}.\lambda\mathtt{g}^{\iota\multimap\iota}.\mathtt{which?}\,(\lambda\mathtt{h}^{\iota\multimap\iota}.\mathtt{f}(\lambda\mathtt{x}^\iota.\mathtt{g}(\mathtt{h}\,\mathtt{x})))$$

behaves in the same way as @wh?. Therefore, by replacing which? with @wh? in $\mathbb{S}\ell$PCF, we program the same linear functions.

3.2 Adding which? to PCF-like languages

If we add which? together with the operational rule of Table 2 to PCF, then which? may return a non-deterministic result when applied to non-linear functions. For instance, if $\mathtt{R}=\lambda\mathtt{f}^{\iota\to\iota}.\mathtt{if}\ (\mathtt{f}\underline{3})\ \Omega^\iota\ (\mathtt{f}\underline{5})$ then both which? R $\Downarrow\lceil\underline{3},\underline{5}\rceil$ and which? R $\Downarrow\lceil\underline{5},\underline{5}\rceil$. However, we can add the operator dwh? to PCF together the same operational rule of which? in Table 2 with the following additional hypothesis,

$$\forall h\leq k\quad \mathtt{M}(\lambda\mathtt{x}^\iota.\ell\mathtt{if}(\overbrace{\mathtt{pred}\dots\mathtt{pred}}^{h}\ \mathtt{x})\ \underline{k}\ \Omega^\iota)\Uparrow$$

This hypothesis asks simply, for the minimum k satisfying the rule of which?, so the determinism is recovered. Such an hypothesis is inane for which? in $\mathbb{S}\ell$PCF. Unfortunately, the given evaluation rule for dwh? is not effective. However, an effective evaluation of dwh? can be formalized likewise that of the operator strict? [29].

It is easy to verify that dwh? can be programmed in PCF + strict? and so in StPCF [29]. Recall that $\mathsf{strict?}^{(\iota\to\iota)\to\iota}\mathtt{M}^{\iota\to\iota}$ checks whether the function $\mathtt{M}^{\iota\to\iota}$ is strict. If $\mathtt{M}^{\iota\to\iota}$ is strict then the evaluation of strict? M returns $\underline{0}$, while if $\mathtt{M}^{\iota\to\iota}$ is not strict but $\mathtt{M}^{\iota\to\iota}\underline{0}$ converges then strict? M returns $\underline{1}$. Recall that dwh? M $\Downarrow$ if and only if $\mathtt{M}(\lambda\mathtt{x}^\iota.\mathtt{x})\Downarrow$. Let $\doteq$ be a program simulating the ground equality (a formal definition is at the begin of Subsection 4.4), it is easy to check that

$$\mathtt{T}=\lambda\mathtt{f}^{(\iota\to\iota)\to\iota}\mathtt{x}^\iota.\mathsf{strict?}^{\iota\to\iota}(\lambda\mathtt{y}^\iota.\mathtt{f}(\lambda\mathtt{z}^\iota.\mathtt{if}\ (\mathtt{x}\doteq\mathtt{z})\ \mathtt{z}\ (\mathtt{if}\ \mathtt{y}\ \mathtt{z}\ \mathtt{z})))$$

tests whether f passes $\mathtt{x}^\iota$ as argument of $\lambda\mathtt{z}^\iota.\mathtt{z}$. Thus dwh? can be translated in the following term of PCF + strict?.

$$\lambda\mathtt{f}^{(\iota\to\iota)\to\iota}.\Big(\mu F^{\iota\to\iota}\lambda\mathtt{x}^\iota.\mathtt{if}\big(\mathtt{T}\,\mathtt{f}\,\mathtt{x}\big)\big(\lceil\mathtt{x},\mathtt{f}(\lambda\mathtt{x}^\iota.\mathtt{x})\rceil\big)\big(F\ (\mathtt{succ}(\mathtt{x}))\big)\Big)\ \underline{0}$$

Note that dwh? is stable and also strongly stable, since PCF + strict? is (see [29]).

We can also compare dwh? with the catch operator, which is present in the language SPCF [11]. catch $\mathtt{x}^\iota$ in $\mathtt{M}^\iota$ binds the occurrences of x in M. Informally, it asks the evaluation of M, if the computation of $\mathtt{M}^\iota$ asks the evaluation of the variable $\mathtt{x}^\iota$ then the computation of catch $\mathtt{x}^\iota$ in $\mathtt{M}^\iota$ terminates giving $\underline{0}$. Otherwise, if the computation of $\mathtt{M}^\iota$ terminates on a numeral $\underline{n}$ without using x, then catch $\mathtt{x}^\iota$ in $\mathtt{M}^\iota$ returns succ($\underline{n}$). We can write a term of SPCF having the same behavior of dwh? using catch, in the following way. Clearly the term $\mathtt{T}'=$ catch y in $\mathtt{f}(\lambda\mathtt{z}^\iota.\mathtt{if}\ (\mathtt{x}\doteq\mathtt{z})\ \mathtt{z}\ \mathtt{y})$ evaluates to $\underline{0}$ (i.e.

causes an error) if `f` does not pass `x` as argument of $\lambda \mathtt{z}^\iota.\mathtt{z}$. Otherwise it evaluates to the successor of the result of the application of `f` to $\lambda \mathtt{z}^\iota.\mathtt{z}$. So the term

$$\lambda \mathtt{f}^{(\iota\to\iota)\to\iota}.\left(\mu F^{\iota\to\iota}.\lambda \mathtt{x}^\iota.\left(\lambda \mathtt{w}^\iota.\mathtt{if\ w}\ \begin{matrix} F\ \mathtt{succ(x)} \\ \lceil \mathtt{x,pred(w)} \rceil \end{matrix}\right)\ \mathtt{T}'\right)\ \underline{\mathtt{0}}$$

behaves like `dwh?`.

We can conclude that `which?` can be used in a fruitful way, in order to manage some kind of "linear" exceptions in $\mathcal{S}\ell$PCF. Note that `which?` is able to perform observations that are subtly linear: in fact `which?M` gives back at once the parameter passed by `M` during its evaluation with the identity function together with the result of the evaluation, in a unique evaluation of the such an application. Thus, the evaluation is done by respecting operational linearity.

3.3 Independence of `which?`

We will show that `which?` can be interpreted in a linear function, so as a corollary, we will obtain that `which?` is also a Scott-continuous and stable function (see [3, pp. 29]). Now, we ask ourselves if there is a program of PCF having the same semantics of our `dwh?`. The answer is negative! We sketch the proof.

If `M` and `N` are programs then $\mathtt{M} \doteq \mathtt{N}$ is an abbreviation for the application of the following term to `M` and `N`:

$$\mu F^{\iota\to\iota\to\iota}.\lambda \mathtt{x}^\iota \mathtt{y}^\iota.\mathtt{if\ x}\ (\mathtt{if\ y}\,\underline{\mathtt{0}}\,\underline{\mathtt{1}})\ \big(\mathtt{if\ y}\ \underline{\mathtt{1}}\ (F\,(\mathtt{pred\,x})(\mathtt{pred\,y}))\big).$$

Namely, $\doteq$ simulates the equality between numerals. Let

$$\mathtt{F}^\sigma_{\underline{\mathtt{k}}} = \lambda \mathtt{f}^{\iota\to\iota}.\mathtt{if}\ ((\mathtt{f}\underline{\mathtt{k}}) \doteq \underline{\mathtt{k}})\ \underline{\mathtt{0}}\ \Omega^\iota\ , \qquad \mathtt{M}_1 = \lambda \mathtt{w}^{\sigma\to\iota}.\Omega^\iota\ ,$$

$$\mathtt{M}_2 = \lambda \mathtt{w}^{\sigma\to\iota}.\mathtt{if}\ \big(((\mathtt{w}\,\mathtt{F}^\sigma_{\underline{\mathtt{0}}}) \doteq \lceil \underline{\mathtt{0}},\underline{\mathtt{0}} \rceil)\ \mathtt{and}\ ((\mathtt{w}\,\mathtt{F}^\sigma_{\underline{\mathtt{1}}}) \doteq \lceil \underline{\mathtt{0}},\underline{\mathtt{1}} \rceil)\big)\ \underline{\mathtt{9}}\ \Omega^\iota$$

be PCF terms. Clearly the evaluation of $\mathtt{M}_1$`dwh?` should diverge, while $\mathtt{M}_2$`dwh?` should return $\underline{9}$.

First, we prove that $\mathtt{M}_1, \mathtt{M}_2$ are operationally equivalent in PCF extended with `por` (see [22, pp.181]). We use notation and Scott-Continuous model of PCF presented in [22]. We recall only that $\sqsubseteq$ denotes the extensional order. Let $\mathtt{F}^\sigma_* = \lambda \mathtt{f}^{\iota\to\iota}.\mathtt{if}\ \big(\mathtt{por}\,((\mathtt{f}\underline{\mathtt{0}}) \doteq \underline{\mathtt{0}})((\mathtt{f}\underline{\mathtt{1}}) \doteq \underline{\mathtt{1}})\big)\ \underline{\mathtt{0}}\ \Omega^\iota$, it is easy to check that $\mathcal{C}[\![\mathtt{F}^\sigma_{\underline{\mathtt{0}}}]\!], \mathcal{C}[\![\mathtt{F}^\sigma_{\underline{\mathtt{1}}}]\!] \sqsubseteq \mathcal{C}[\![\mathtt{F}^\sigma_*]\!]$, thus $\mathcal{C}[\![\mathtt{F}^\sigma_{\underline{\mathtt{0}}}]\!] \sqcup \mathcal{C}[\![\mathtt{F}^\sigma_{\underline{\mathtt{1}}}]\!] \sqsubseteq \mathcal{C}[\![\mathtt{F}^\sigma_*]\!]$.

For all d element of the Scott-domain $\mathcal{C}[\![((\mathbb{N}_\perp \to \mathbb{N}_\perp) \to \mathbb{N}_\perp) \to \mathbb{N}_\perp]\!]$, if $d \circ \mathcal{C}[\![\mathtt{F}^\sigma_{\underline{\mathtt{0}}}]\!] = 0$ and $d \circ \mathcal{C}[\![\mathtt{F}^\sigma_{\underline{\mathtt{1}}}]\!] = 1$ then $0, 1 \sqsubseteq d \circ \mathcal{C}[\![\mathtt{F}^\sigma_*]\!]$ by monotonicity. Thus we can state that, such an element d cannot exist. Easily, it follows that $\mathtt{M}_1, \mathtt{M}_2$ are equivalent in the standard Scott-continuous model. The correctness of this model implies that $\mathtt{M}_1, \mathtt{M}_2$ are operationally equivalent in PCF + `por`.

PROPOSITION 1. *There exists no program in* PCF + `por` *(and so in* PCF*) having the same semantics of* `dwh?`.

Note that programs of $\mathcal{S}\ell$PCF that do not use `which?` are program of PCF (by writing `if` in place of ℓ`if`). Thus the above proposition implies the following corollary.

COROLLARY 1. `which?` *is not syntactic sugar for* $\mathcal{S}\ell$PCF.

4. LINEAR MODEL

We are interested in the least full subcategory of coherence spaces, including **N** and the coherence spaces of linear functions between domains in the category itself. After the introduction coherence spaces, we prove that the considered model is fully abstract with respect to $\mathcal{S}\ell$PCF.

4.1 Coherence Spaces

Coherence spaces are a simple framework for Berry's stable functions [7], developed by Girard [19]. Proof details can be found in [21].

Definition 7. A *coherence space* X is a pair $\langle |X|, \frown\!\!\!\!\smile_X \rangle$ where $|X|$ is a set of *tokens* called the *web* of X and $\frown\!\!\!\!\smile_X$ is a reflexive and symmetric relation between tokens of $|X|$ called the *coherence relation* [1] on X. A clique x of X is a subset of $|X|$ containing pairwise coherent tokens. The set of cliques of X is denoted $\mathcal{C}l(X)$.

If X is a coherence space then $\mathcal{C}l(X)$ form a cpo with respect to the set-theoretical inclusion. In particular,

- $\emptyset \in \mathcal{C}l(X)$ and $\{a\} \in \mathcal{C}l(X)$, for each $a \in |X|$,
- if $y \subseteq x$ and $x \in \mathcal{C}l(X)$ then $y \in \mathcal{C}l(X)$,
- if $D \subseteq \mathcal{C}l(X)$ is directed then $\cup D \in \mathcal{C}l(X)$.

Definition 8. Let X and Y be coherence spaces and $f : \mathcal{C}l(X) \longrightarrow \mathcal{C}l(Y)$ be a monotone function.

- f is *continuous* whenever $\forall x \in \mathcal{C}l(X)\ \forall b \in f(x)\ \exists x_0 \subseteq_{fin} x$ s.t. $b \in f(x_0)$.
- f is *stable* whenever $\forall x \in \mathcal{C}l(X)\ \forall b \in f(x)\ \exists x_0 \subseteq_{fin} x$ such that $b \in f(x_0)$ and $\forall x' \subseteq x$, if $b \in f(x')$ then $x_0 \subseteq x'$.
- f is *linear* whenever $\forall x \in \mathcal{C}l(X), \forall b \in f(x)\ \exists! a \in x$ s.t. $b \in f(\{a\})$.

Some well-known characterizations hold for these functions.

LEMMA 4. *Let X and Y be coherence spaces.*

- *If $f : \mathcal{C}l(X) \longrightarrow \mathcal{C}l(Y)$ is a monotone function then f is continuous if and only if $f(\cup D) = \cup\{f(x)/x \in D\}$, for each directed $D \subseteq \mathcal{C}l(X)$.*
- *If $f : \mathcal{C}l(X) \longrightarrow \mathcal{C}l(Y)$ is a continuous function then f is stable if and only if $\forall x, x' \in \mathcal{C}l(X)$, $x \cup x' \in \mathcal{C}l(X)$ implies $f(x \cap x') = f(x) \cap f(x')$.*

[1] The strict incoherence $\smile_X$ is the complementary relation of $\frown\!\!\!\!\smile_X$; the incoherence $\asymp_X$ is the union of relations $\smile_X$ and $=$; the strict coherence $\frown_X$ is the complementary relation of $\asymp_X$.

- *If $f : \mathcal{C}l(X) \longrightarrow \mathcal{C}l(Y)$ is a stable function then f is linear if and only if $f(x) = \cup_{a \in x} f(\{a\})$ and $f(\emptyset) = \emptyset$.*

Linear functions can be represented as cliques.

Definition 9. Let X and Y be coherence spaces. $X \multimap Y$ is the coherence space having $|X \multimap Y| = |X| \times |Y|$ as web, while $(a, b) \frown_{X \multimap Y} (a', b')$ is defined as,

$$a =_X a' \text{ implies } b =_Y b' \text{ and } a \frown_X a' \text{ implies } b \frown_Y b'$$

Let X and Y be coherence spaces. The *trace* of a linear function $f : \mathcal{C}l(X) \to \mathcal{C}l(Y)$ is the set defined as follows:

$$\mathcal{T}r(f) = \{(a, b) \in |X| \times |Y| \mid b \in f(\{a\})\} \tag{1}$$

Let X and Y be coherence spaces, $t \in \mathcal{C}l(X \multimap Y)$ and $x \in \mathcal{C}l(X)$. Let us define the map $\mathcal{F}(t) : \mathcal{C}l(X) \to \mathcal{C}l(Y)$ to be the function such that

$$\mathcal{F}(t)(x) = \{b \in |Y| \mid \exists a \in x, (a, b) \in t\} \tag{2}$$

Lemma 5. *If $f : \mathcal{C}l(X) \to \mathcal{C}l(Y)$ is a linear function then $\mathcal{T}r(f) \in \mathcal{C}l(X \multimap Y)$. If $t \in \mathcal{C}l(X \multimap Y)$ then $\mathcal{F}(t) : \mathcal{C}l(X) \to \mathcal{C}l(Y)$ is a linear function.*

The basis of our model is the infinite flat domain. Let $\mathbf{N}$ denotes the space of natural numbers, namely $(|\mathbf{N}|, \frown_{\mathbf{N}})$ such that $|\mathbf{N}| = \mathbb{N}$ and $m \frown_{\mathbf{N}} n$ if and only if $m = n$, for all $m, n \in |\mathbf{N}|$. Thus $\mathcal{C}l(\mathbf{N}) = \{\emptyset\} \cup \{\ \{n\} \mid n \in |\mathbf{N}|\ \}$ endowed with set-theoretical inclusion form the infinite flat domain.

4.2 Interpretation

Emphatic brackets will be used as notation in order to formalize both correspondences between types and coherence spaces and between terms and cliques, in particular $[\![\iota]\!] = \mathbf{N}$ and $[\![\sigma \multimap \tau]\!] = [\![\sigma]\!] \multimap [\![\tau]\!]$. Interpretation is *standard* (see [32]), since ground types are interpreted on flat posets. Let $[\![\tau_0 \multimap \ldots \multimap \tau_m \multimap \iota]\!]$ be a coherence space; for sake of simplicity, its tokens will be wrote as $(a_1, ..., a_m, b)$ where $a_i \in |[\![\tau_i]\!]|$ for each $i \leq m$ and $b \in |\mathbf{N}|$.

An *environment* ρ is a function that associates to each variable $\varkappa^\sigma$ a clique in $\mathcal{C}l([\![\sigma]\!])$. The set of environments is denoted by Env. If $d \in \mathcal{C}l([\![\sigma]\!])$ and $\varkappa^\sigma \in \text{Var}$ then $\rho[\varkappa := d]$ is the environment such that, $\rho[\varkappa := d](\varkappa) = d$, but if $\varkappa' \neq \varkappa$ then $\rho[\varkappa := d](\varkappa') = \rho(\varkappa')$.

Definition 10. Let $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathcal{S}\ell\text{PCF}$ and $\rho \in \text{Env}$. The interpretation $[\![\mathtt{M}^\sigma]\!] : \text{Env} \to \mathcal{C}l([\![\sigma]\!])$ is defined [2] in Table 4.

$\mathtt{M}^\sigma \backsim_\sigma \mathtt{N}^\sigma$ (*denotationally equivalent*) if and only if $[\![\mathtt{M}^\sigma]\!]\rho = [\![\mathtt{N}^\sigma]\!]\rho$ for each ρ. The interpretation of closed terms is invariant with respect to environments, thus in such cases the environment can be omitted. The basic properties of a lambda-model are recalled in Lemma 6.

[2] Note that $\mathcal{F}$ is defined in Equation 2 and `fix` is defined in Theorem 1.

Lemma 6. *Let $\mathtt{M}^\sigma, \mathtt{N}^\tau \in \mathcal{S}\ell\text{PCF}$ and $\rho, \rho' \in \text{Env}$.*

1. *If $\rho(\varkappa) \subseteq \rho'(\varkappa)$ for each $\varkappa \in \text{FV}(\mathtt{M})$, then $[\![\mathtt{M}]\!]\rho \subseteq [\![\mathtt{M}]\!]\rho'$.*
2. *If $\mathtt{M}^\sigma[\mathtt{N}/\varkappa^\tau] \in \mathcal{S}\ell\text{PCF}$ then $[\![\mathtt{M}^\sigma[\mathtt{N}/\varkappa^\tau]]\!]\rho = [\![\mathtt{M}]\!]\rho[\varkappa^\tau := [\![\mathtt{N}]\!]\rho]$.*
3. *If $\sigma = \tau$, $C[^\sigma] \in \text{Ctx}_\sigma$, $C[\mathtt{M}], C[\mathtt{N}] \in \mathcal{P}$ and $[\![\mathtt{M}]\!]\rho = [\![\mathtt{N}]\!]\rho$ then $[\![C[\mathtt{M}]]\!] = [\![C[\mathtt{N}]]\!]$.*

Proof. *1.* and *2.* are easy, by induction on the structure of $\mathtt{M}^\sigma$. *3.* follows by induction on the structure of $\mathtt{C}[^\sigma]$. □

Lemma 7. *$[\![\mu F.\mathtt{M}^\sigma]\!]\rho = \bigcup_{n \in \mathbb{N}} [\![\mu^n F.\mathtt{M}^\sigma]\!]\rho$, for all $\sigma \in \mathbb{T}$.*

Proof. Since $[\![\mu^{n+1} F.\mathtt{M}^\sigma]\!]\rho = [\![\mathtt{M}^\sigma]\!]\rho[F := [\![\mu^n F.\mathtt{M}^\sigma]\!]\rho]$, the proof is easy. □

Lemma 8. *Let $\mathtt{M} \in \mathcal{P}$. If $\mathtt{M} \Downarrow \underline{\mathtt{n}}$ then $[\![\mathtt{M}]\!] = [\![\underline{\mathtt{n}}]\!]$.*

Proof. The proof can e done by induction on the derivation proving $\mathtt{M} \Downarrow \underline{\mathtt{n}}$, likewise to similar proofs in [29, 32]. □

4.3 Adequacy and Correctness

The denotational semantics is said to be *adequate* when $[\![\mathtt{M}]\!] = [\![\underline{\mathtt{n}}]\!]$ and $\mathtt{M} \Downarrow \underline{\mathtt{n}}$ are logically equivalent for any program $\mathtt{M}$, numeral $\underline{\mathtt{n}}$. We straightforward adapt a proof of Plotkin [32] for Scott-continuous domains, based on a computability argument in Tait style.

Definition 11. The "computability predicate" is defined by the following cases.

- Case $\text{FV}(\mathtt{M}^\sigma) = \emptyset$.
  - Subcase $\sigma = \iota$. $\text{Comp}(\mathtt{M}^\iota)$ if and only if $[\![\mathtt{M}]\!]\rho = [\![\underline{\mathtt{n}}]\!]\rho$ implies $\mathtt{M} \Downarrow \underline{\mathtt{n}}$.
  - Subcase $\sigma = \mu \multimap \tau$. $\text{Comp}(\mathtt{M}^{\mu \multimap \tau})$ if and only if $\text{Comp}(\mathtt{M}^{\mu \multimap \tau}\mathtt{N}^\mu)$ for each closed $\mathtt{N}^\mu$ such that $\text{Comp}(\mathtt{N}^\mu)$.
- Case $\text{FV}(\mathtt{M}^\sigma) = \{\varkappa_1^{\tau_1}, \ldots, \varkappa_n^{\tau_n}\}$, for some $n \geq 1$. $\text{Comp}(\mathtt{M}^\sigma)$ if and only if $\text{Comp}(\mathtt{M}[\mathtt{N}_1/\varkappa_1, \ldots, \mathtt{N}_n/\varkappa_n])$ for each closed $\mathtt{N}_i^{\tau_i}$ such that $\text{Comp}(\mathtt{N}_i^{\tau_i})$.

Lemma 9 states an equivalent formulation of computability predicate.

Lemma 9. *Let $\mathtt{M}^{\tau_1 \multimap \ldots \multimap \tau_m \multimap \iota} \in \mathcal{S}\ell\text{PCF}$ and $\text{FV}(\mathtt{M}) = \{\varkappa_1^{\mu_1}, ..., \varkappa_n^{\mu_n}\}$ $(n, m \in \mathbb{N})$. $\text{Comp}(\mathtt{M})$ if and only if $[\![\mathtt{M}[\mathtt{N}_1/\varkappa_1, \ldots, \mathtt{N}_n/\varkappa_n]\mathtt{P}_1 \ldots \mathtt{P}_m]\!]\rho = [\![\underline{\mathtt{n}}]\!]$ implies $\mathtt{M}[\mathtt{N}_1/\varkappa_1, \ldots, \mathtt{N}_n/\varkappa_n]\mathtt{P}_1 \ldots \mathtt{P}_m \Downarrow \underline{\mathtt{n}}$ for each closed $\mathtt{N}_i^{\mu_i}$ and $\mathtt{P}_j^{\tau_j}$ such that $\text{Comp}(\mathtt{N}_i)$ and $\text{Comp}(\mathtt{P}_j)$ where $i \leq n, j \leq m$.*

Lemma 10. *If $\mathtt{M}^\sigma \in \mathcal{S}\ell\text{PCF}$ then $\text{Comp}(\mathtt{M}^\sigma)$.*

$$\llbracket \underline{\mathtt{0}}^{\iota} \rrbracket\rho = \{0\} \qquad \llbracket \mathtt{succ}^{\iota \multimap \iota} \rrbracket\rho = \{(n, n+1) \mid n \in \mathbb{N}\} \qquad \llbracket \mathtt{pred}^{\iota \multimap \iota} \rrbracket\rho = \{(n, n-1) \mid n \in \mathbb{N}\}$$

$$\llbracket \mathtt{which?}^{((\iota \multimap \iota) \multimap \iota) \multimap \iota} \rrbracket\rho = \{(((n,n), r), \lceil n, r \rceil) \mid n, r \in \mathbb{N}\}$$

$$\llbracket (\ell\mathtt{if}\ \mathtt{M}^{\iota}\ \mathtt{N}^{\iota}\ \mathtt{L}^{\iota})^{\iota} \rrbracket\rho = \begin{array}{l} \{n \in \mathbb{N} \mid \llbracket \mathtt{M}^{\iota} \rrbracket\rho = \{0\} \wedge \llbracket \mathtt{N}^{\iota} \rrbracket\rho = \{n\}\} \cup \\ \{n \in \mathbb{N} \mid \llbracket \mathtt{M}^{\iota} \rrbracket\rho = \{m+1\} \wedge \llbracket \mathtt{L}^{\iota} \rrbracket\rho = \{n\}, m \in \mathbb{N}\} \end{array}$$

$$\llbracket \varkappa^{\sigma} \rrbracket\rho = \rho(\varkappa^{\sigma}) \qquad \llbracket \lambda \mathtt{x}^{\sigma}.\mathtt{M}^{\tau} \rrbracket\rho = \{(a_0, b) \in |\llbracket\sigma\rrbracket| \times |\llbracket\tau\rrbracket| \mid b \in \llbracket \mathtt{M}^{\tau} \rrbracket\rho[\mathtt{x}^{\sigma} := \{a_0\}]\}$$

$$\llbracket \mathtt{M}^{\sigma \multimap \tau} \mathtt{N}^{\sigma} \rrbracket\rho = \mathcal{F}(\llbracket \mathtt{M}^{\sigma \multimap \tau} \rrbracket\rho)\llbracket \mathtt{N}^{\sigma} \rrbracket\rho \qquad \llbracket (\mu F^{\sigma}.\mathtt{M}^{\sigma})^{\sigma} \rrbracket\rho = \mathtt{fix}(\lambda x.\llbracket \mathtt{M} \rrbracket\rho[F := x])$$

Table 4: Interpretation Map.

Proof. The proof is done by induction on the "untyped syntax shape" of terms. We consider only some cases, the other are easier (see [29, 32]).

- $\mathtt{M} = \varkappa$. Let $\sigma = \tau_1 \multimap \cdots \multimap \tau_m \multimap \iota$, where $m \in \mathbb{N}$. Let $\mathtt{P}^{\sigma}$ and $\mathtt{N}_i^{\tau_i}$ for $1 \leq i \leq m$ be closed terms such that $\mathrm{Comp}(\mathtt{P}^{\sigma})$ and $\mathrm{Comp}(\mathtt{N}_i^{\tau_i})$. By definition, $\mathrm{Comp}(\mathtt{P}^{\sigma})$ imply that, if $\llbracket \mathtt{P}\mathtt{N}_1 \cdots \mathtt{N}_m \rrbracket\rho = \llbracket \underline{\mathtt{n}} \rrbracket\rho$ then $\mathtt{P}\mathtt{N}_1 \cdots \mathtt{N}_m \Downarrow \underline{\mathtt{n}}$.
- $\mathtt{M} = \mathtt{NP}$. Assume $\mathtt{N}^{\tau \multimap \sigma}$ and $\mathtt{P}^{\tau}$ for types σ and τ. By induction hypothesis $\mathrm{Comp}(\mathtt{N}^{\tau \multimap \sigma})$ and $\mathrm{Comp}(\mathtt{P}^{\tau})$ and the proof follows by computability definition.
- $\mathtt{M} = \lambda \mathtt{x}.\mathtt{Q}$. Assume $\mathtt{x}^{\mu}$ and $\mathtt{Q}^{\tau}$ for types μ and τ. Let $\mathrm{FV}(\mathtt{M}) = \{\varkappa_1^{\mu_1}, \ldots, \varkappa_k^{\mu_k}\}$ for $k \geq 0$ and $\tau = \tau_1 \multimap \cdots \multimap \tau_h \multimap \iota$, where $h \geq 0$. Let $\mathtt{N}_1^{\mu_1}, \ldots, \mathtt{N}_k^{\mu_k}, \mathtt{P}_0^{\mu}, \mathtt{P}_1^{\tau_1}, \ldots, \mathtt{P}_h^{\tau_h}$ be closed terms such that $\mathrm{Comp}(\mathtt{N}_i)$ and $\mathrm{Comp}(\mathtt{P}_j)$ for $1 \leq i \leq k$ and $0 \leq j \leq h$ respectively. Thus $\mathrm{Comp}(\mathtt{Q}^{\tau}[\mathtt{P}_0/\mathtt{x}][\mathtt{N}_1/\varkappa_1, \ldots, \mathtt{N}_k/\varkappa_k]\mathtt{P}_1 \ldots \mathtt{P}_h)$, since $\mathrm{Comp}(\mathtt{Q}^{\tau})$ holds by induction hypothesis. Consider the case $\mu \neq \iota$ and suppose
$\llbracket (\lambda \mathtt{x}^{\mu}.\mathtt{Q}^{\tau})[\mathtt{N}_1/\varkappa_1, \ldots, \mathtt{N}_k/\varkappa_k]\mathtt{P}_0 \ldots \mathtt{P}_h \rrbracket\rho = \llbracket \underline{\mathtt{n}} \rrbracket$. Thus
$\llbracket \mathtt{Q}^{\tau}[\mathtt{P}_0/\mathtt{x}][\mathtt{N}_1/\varkappa_1, \ldots, \mathtt{N}_k/\varkappa_k]\mathtt{P}_1 \ldots \mathtt{P}_h \rrbracket\rho = \llbracket \underline{\mathtt{n}} \rrbracket$ by Lemma 8. Therefore $\mathtt{Q}^{\tau}[\mathtt{P}_0/\mathtt{x}][\mathtt{N}_1/\varkappa_1, \ldots, \mathtt{N}_k/\varkappa_k]\mathtt{P}_1 \ldots \mathtt{P}_h \Downarrow \underline{\mathtt{n}}$ by induction hypothesis.
So, $(\lambda \mathtt{x}^{\mu}.\mathtt{Q}^{\tau})[\mathtt{N}_1/\varkappa_1, \ldots, \mathtt{N}_k/\varkappa_k]\mathtt{P}_0 \ldots \mathtt{P}_h \Downarrow \underline{\mathtt{n}}$ by the evaluation rule $(\lambda^{\multimap})$. Now, suppose $\mu = \iota$ and
$\llbracket (\lambda \mathtt{x}^{\mu}.\mathtt{Q}^{\tau})[\mathtt{N}_1/\varkappa_1, \ldots, \mathtt{N}_k/\varkappa_k]\mathtt{P}_0 \ldots \mathtt{P}_h \rrbracket\rho = \llbracket \underline{\mathtt{n}} \rrbracket$, thus $\llbracket \mathtt{P}_0 \rrbracket = \llbracket \underline{\mathtt{m}} \rrbracket$ for some $\underline{\mathtt{m}}$. But $\mathrm{Comp}(\mathtt{P}_0^{\mu})$ and $\llbracket \mathtt{P}_0 \rrbracket = \llbracket \underline{\mathtt{m}} \rrbracket$ imply $\mathtt{P}_0 \Downarrow \underline{\mathtt{m}}$. Hence $\llbracket \mathtt{Q}^{\tau}[\underline{\mathtt{m}}/\mathtt{x}][\mathtt{N}_1/\varkappa_1, \ldots, \mathtt{N}_k/\varkappa_k]\mathtt{P}_1 \ldots \mathtt{P}_h \rrbracket\rho = \llbracket \underline{\mathtt{n}} \rrbracket$, by Lemma 6. Therefore
$\mathtt{Q}^{\tau}[\underline{\mathtt{m}}/\mathtt{x}][\mathtt{N}_1/\varkappa_1, \ldots, \mathtt{N}_k/\varkappa_k]\mathtt{P}_1 \ldots \mathtt{P}_h \Downarrow \underline{\mathtt{n}}$ by induction hypothesis. The proof follows by applying the evaluation rule (λ^{ι}).
- $\mathtt{M} = \mu F.\mathtt{N}$. Assume $\mathtt{M}^{\sigma}, \mathtt{N}^{\sigma}$ for some type σ. Let $\mathrm{FV}(\mathtt{M}) = \{\varkappa_1^{\mu_1}, \ldots, \varkappa_k^{\mu_k}\}$ for $k \geq 0$ and $\sigma = \tau_1 \multimap \cdots \multimap \tau_h \multimap \iota$, where $h \geq 0$. By induction on h, likewise to the corresponding proof of [32]. The case $h = 0$ is trivial, so assume $h \geq 1$. Assume $\mathtt{N}_1^{\mu_1}, \ldots, \mathtt{N}_k^{\mu_k}$ and $\mathtt{P}_1^{\tau_1}, \ldots, \mathtt{P}_h^{\tau_h}$ be closed terms such that $\mathrm{Comp}(\mathtt{N}_i)$ and $\mathrm{Comp}(\mathtt{P}_j)$ for $1 \leq i \leq k$ and $1 \leq j \leq h$ respectively.
Let $\llbracket (\mu F.\mathtt{N}^{\sigma}[\mathtt{N}_1/\varkappa_1, \ldots, \mathtt{N}_k/\varkappa_k])\mathtt{P}_1 \ldots \mathtt{P}_h \rrbracket\rho = \llbracket \underline{\mathtt{n}} \rrbracket$. As remarked just before this lemma
$$\begin{aligned} &\llbracket (\mu F.\mathtt{N}^{\sigma}[\mathtt{N}_1/\varkappa_1, \ldots, \mathtt{N}_k/\varkappa_k])\mathtt{P}_1 \ldots \mathtt{P}_h \rrbracket\rho = \\ &\qquad \llbracket (\mu^{k} F.\mathtt{N}^{\sigma}[\mathtt{N}_1/\varkappa_1, \ldots, \mathtt{N}_k/\varkappa_k])\mathtt{P}_1 \ldots \mathtt{P}_h \rrbracket\rho \end{aligned}$$
for some $k \in \mathbb{N}$. Thus, $\mu^{k} F.\mathtt{N}^{\sigma}[\mathtt{Q}'/\mathtt{v}_1, \ldots, \mathtt{Q}'/\mathtt{v}_m]\mathtt{P}_1 \ldots \mathtt{P}_h \Downarrow \underline{\mathtt{n}}$, by the previous points of this lemma. The proof follows by Lemma 2. $\square$

As usual the adequacy implies the correctness. Note that correctness implies that our terms are strict in all arguments, for all orders.

Theorem 3. *The linear interpretation is correct for* $\mathcal{S}\ell$PCF.

Proof. Let $\mathtt{M}^{\sigma}$ and $\mathtt{N}^{\sigma}$ such that $\llbracket \mathtt{M} \rrbracket_{\rho} = \llbracket \mathtt{N} \rrbracket_{\rho}$, for each environment $\rho \in \mathrm{Env}$. Let $\mathtt{C}[^{\sigma}]$ such that $\mathtt{C}[\mathtt{M}], \mathtt{C}[\mathtt{N}] \in \mathcal{P}$. If $\mathtt{C}[\mathtt{M}] \Downarrow \underline{\mathtt{n}}$ for some value $\underline{\mathtt{n}}$, then $\llbracket \mathtt{C}[\mathtt{M}] \rrbracket = \llbracket \underline{\mathtt{n}} \rrbracket$ by Lemma 8. Since $\llbracket \mathtt{C}[\mathtt{N}] \rrbracket = \llbracket \mathtt{C}[\mathtt{M}] \rrbracket = \llbracket \underline{\mathtt{n}} \rrbracket$ by Lemma 6, $\mathtt{C}[\mathtt{N}] \Downarrow \underline{\mathtt{n}}$ by adequacy. By definition of operational equivalence the proof is done. $\square$

4.4 Completeness and Full Abstraction

If $\sigma \in \mathbb{T}$ then, as usual, its order $\mathrm{ORDER}(\sigma)$ is defined by $\mathrm{ORDER}(\iota) = 0$ and $\mathrm{ORDER}(\sigma_1 \multimap \ldots \multimap \sigma_k \multimap \iota) = 1 + \max\{\mathrm{ORDER}(\sigma_i) | i \in [1, k]\}$.

We show that, all tokens of coherence spaces are definable. If $\mathtt{M}$ and $\mathtt{N}$ are closed ground terms then $\mathtt{M} \doteq \mathtt{N}$ is an abbreviation for the application of

$$\mu F^{\iota \multimap \iota \multimap \iota}.\lambda \mathtt{x}^{\iota}\mathtt{y}^{\iota}.\ell\mathtt{if}\ \mathtt{x}\ (\ell\mathtt{if}\ \mathtt{y}\ \underline{\mathtt{0}}\ \underline{\mathtt{1}})\ \big(\ell\mathtt{if}\ \mathtt{y}\ \underline{\mathtt{1}}\ (F\,(\mathtt{pred}\,\mathtt{x})(\mathtt{pred}\,\mathtt{y}))\big)$$

to $\mathtt{M}$ and $\mathtt{N}$. It is easy to check that

$$\llbracket \mathtt{M} \doteq \mathtt{N} \rrbracket = \begin{cases} 0 & \llbracket \mathtt{M} \rrbracket = m = \llbracket \mathtt{N} \rrbracket, \\ 1 & \llbracket \mathtt{M} \rrbracket = m \neq n = \llbracket \mathtt{N} \rrbracket, \\ \emptyset & \text{otherwise.} \end{cases}$$

Let $\mathtt{N}_0$ and $\mathtt{N}_1$ be an abbreviation for the term

$$\ell\mathtt{if}\ \mathtt{N}_0\ (\ell\mathtt{if}\ \mathtt{N}_1\ \underline{\mathtt{0}}\ \underline{\mathtt{1}})\ (\ell\mathtt{if}\ \mathtt{N}_1\ \underline{\mathtt{1}}\ \underline{\mathtt{1}}).$$

Lemma 11 (Token Definability). *Let* $\sigma \in \mathbb{T}$. *If* $a \in |\llbracket\sigma\rrbracket|$ *then there is a closed term* $\mathtt{M}^{\sigma}$ *such that* $\llbracket \mathtt{M}^{\sigma} \rrbracket = \{a\}$.

Proof. The proof is done by induction on the order of σ. The case $\mathrm{ORDER}(\sigma) = 0$ is trivial. Assume $\mathrm{ORDER}(\sigma) = 1$, thus $\sigma = \iota_1 \multimap \ldots \multimap \iota_k \multimap \iota$. Hence, given $a \in |\llbracket\sigma\rrbracket|$, a has the shape $(n_1, \ldots, n_k, n)$ where $n_1, \ldots, n_k, n \in \mathbb{N}$. If

$$\begin{aligned} &\mathtt{M}^{\sigma} = \lambda \mathtt{x}_1^{\iota} \ldots \mathtt{x}_{\mathtt{k}}^{\iota}. \\ &\ell\mathtt{if}\ (\ (\ \mathtt{x}_1 \doteq \underline{\mathtt{n}}_1)\ \mathtt{and}\ (\mathtt{x}_2 \doteq \underline{\mathtt{n}}_2)\ \mathtt{and}\ \ldots\ \mathtt{and}\ (\mathtt{x}_{\mathtt{k}} \doteq \underline{\mathtt{n}}_k))\ \underline{\mathtt{n}}\ \Omega^{\iota} \end{aligned}$$

then $\llbracket \mathtt{M}^{\sigma} \rrbracket_{\rho} = \{a\}$, for all ρ.
Finally, assume $\sigma = \sigma_1 \multimap \ldots \multimap \sigma_k \multimap \iota$ where $\mathrm{ORDER}(\sigma_i) > 1$ and $\sigma_i = \tau_1^i \multimap \ldots \multimap \tau_{h_i}^i \multimap \iota$ for $1 \leq i \leq k$. If $a \in \llbracket\sigma\rrbracket$

then a has shape $((a^1_1, \ldots, a^1_{h_1}, n_1), \ldots, (a^k_1, \ldots, a^k_{h_k}, n_k), n')$ where $n_1, \ldots, n_k, n' \in \mathbb{N}$ and $a^i_j \in |[\![\tau^i_j]\!]|$, for each $i \in [1, k]$ and $j \in [1, h_i]$. By inductive hypothesis's, for all $i \in [1, k]$ and $j \in [1, h_i]$, there exists a closed term $\mathtt{M}^{\tau^i_j}$ such that $[\![\mathtt{M}^{\tau^i_j}]\!] = \{a^i_j\}$. Thus,

$$\lambda \mathtt{x}^{\sigma_1}_1 \ldots \lambda \mathtt{x}^{\sigma_k}_k . \ell\mathtt{if} \left(\begin{array}{c} (\mathtt{x}_1 \mathtt{M}^{\tau^1_1} \ldots \mathtt{M}^{\tau^1_{h_1}} \doteq \underline{\mathtt{n}}_1) \\ \mathtt{and} \ \ldots \ \mathtt{and} \\ (\mathtt{x}_k \mathtt{M}^{\tau^k_1} \ldots \mathtt{M}^{\tau^k_{h_k}} \doteq \underline{\mathtt{n}}_k)) \end{array} \right) \underline{\mathtt{n}}' \ \Omega^\iota$$

is the the closed term defining the considered token. $\square$

Lemma 12 (Separability). *Let $\sigma \in \mathbb{T}$. For all distinct $f, g \in \mathcal{C}l([\![\sigma]\!])$ there exists a closed term $\mathtt{M}^{\sigma \multimap \iota} \in$ SℓPCF such that $\mathcal{F}([\![\mathtt{M}^{\sigma \multimap \iota}]\!])(f) \neq \mathcal{F}([\![\mathtt{M}^{\sigma \multimap \iota}]\!])(g)$.*

Proof. Let $\sigma = \sigma_1 \multimap \ldots \sigma_k \multimap \iota$. Since $f \neq g$ there exists a such that $a \in f$ but $a \notin g$. Assume $a = (a_1, \ldots, a_k, n)$ (where $a_1 \in |[\![\sigma_1]\!]|, \ldots, a_k \in |[\![\sigma_k]\!]|, n \in \mathbb{N}$). By lemma 11, for each $i \in [1, k]$ there is $\mathtt{N}^{\sigma_i} \in$ SℓPCF such that $[\![\mathtt{N}^{\sigma_i}]\!] = a_i$. By choosing $\mathtt{M}^{\sigma \multimap \iota}$ as $\lambda x^\sigma . \ell\mathtt{if}$ (($\mathtt{x}\,\mathtt{N}^{\sigma_1} \ldots \mathtt{N}^{\sigma_k}) \doteq \underline{\mathtt{n}}$) $\underline{0}$ $\underline{1}$, the proof follows. $\square$

Note that in [29, 32] the separability needs the definability of finite elements, while in our setting the token definability is sufficient. The Separability Lemma implies completeness and full abstraction, as in [29, 32]. Clearly `which`? is not necessary in order to obtain the full abstraction, but it is necessary in order to define all finite elements of the considered model.

Theorem 4 (Completness).
If $\mathtt{N}_1 \approx_\sigma \mathtt{N}_2$ then $[\![\mathtt{N}^\sigma_1]\!] = [\![\mathtt{N}^\sigma_2]\!]$.

Proof. Let us prove the contrapposition: let us assume $[\![\mathtt{N}^\sigma_1]\!]_\rho \neq [\![\mathtt{N}^\sigma_2]\!]_\rho$, for any environment ρ. By definable separability, there exists $\mathtt{M}^{\sigma \multimap \iota}$ such that $\mathcal{F}([\![\mathtt{M}^{\sigma \multimap \iota}]\!]_\rho)([\![\mathtt{N}_1]\!]_\rho) = n_1 \neq \mathcal{F}([\![\mathtt{M}^{\sigma \multimap \iota}]\!]_\rho)([\![\mathtt{N}_2]\!]_\rho) = n_2$. By adequacy, $\mathtt{M}^{\sigma \multimap \iota} \mathtt{N}^\sigma_1 \Downarrow \underline{\mathtt{n}}_1$ and $\mathtt{M}^{\sigma \multimap \iota} \mathtt{N}^\sigma_2 \Downarrow \underline{\mathtt{n}}_2$. So $\mathtt{N}_1 \not\approx_\sigma \mathtt{N}_2$. $\square$

Corollary 2 (Full abstraction).
$\mathtt{N}_1 \approx_\sigma \mathtt{N}_2$ *if and only if* $[\![\mathtt{N}^\sigma_1]\!] = [\![\mathtt{N}^\sigma_2]\!]$.

5. LINEAR EXTENSIONS OF SℓPCF

Denotational semantics is usually proposed as a tool to study the equivalence of programs. However, a further use of denotational models is as an abstract tool, assisting us in the comparison and analysis of whole programming languages endowed with different type-respecting computational power (such approach, was first suggested in [26] and pursued in [29]). Such approach, already used in Subsection 3.3, is crucial also in what follows.

Definition 12. A model is *universal* for a language when every effective element (of domains the interpretation of types) is definable by a closed term of the language [27].

Linear functions are also Scott-continuous with respect to extensional order, see [3, pp. 29]. In order to explore the possible linear extensions of SℓPCF we start by considering some well-known extensions of PCF.

It has been proved in [32] that Scott-continuous domains are fully abstract and universal for PCF^{++}, namely PCF extended with a parallel conditional `pif` and an existential operator $\exists$.
It has been shown that stable domains (dI-domains, qualitative domains, coherence spaces) give a fully abstract model [29] for StPCF, a syntactical extension of PCF. The language StPCF is obtained by extending PCF with two operators: `gor` and `strict`?. `gor` corresponds to a Gustave-like `or` function, while `strict`? corresponds to a non extensional-monotone function.
However `strict`? does not respect the Scott-continuity, while `pif` and $\exists$ do not respect the stability. Whence, they cannot belong to our language. On the other hand, `gor` respect both Scott-continuity and stability, but it's not strict, and therefore it's not linear.

In [26] PCF has been extended with the operator `H`. PCF+`H` is universal for the strongly stable model [9]. Note that `strict`? is strongly stable [29], thus it can be defined in PCF+ `H`. Thus `H` does not respect the extensionality and, then, the linearity of our model. Moreover, since `dwh`? can be defined in PCF + `strict`? our language can program only strongly stable functions.

Unfortunately, SℓPCF does not enjoy of the universality w.r.t. our model. Actually, it is not sufficient in order to obtain the definability of finite cliques.

$$\left\{ \begin{array}{l} ((0,0)\,,\,(1,0)\,,\,(0,1)\,,\,0) \\ ((0,1)\,,\,(0,0)\,,\,(1,0)\,,\,1) \\ ((1,0)\,,\,(0,1)\,,\,(0,0)\,,\,2) \\ ((1,1)\,,\,(1,1)\,,\,(1,1)\,,\,3) \end{array} \right\}$$

in $(\iota \multimap \iota) \multimap (\iota \multimap \iota) \multimap (\iota \multimap \iota) \multimap \iota$ is linear, but we will show that it cannot be defined in SℓPCF. We can add $\overset{2}{\mathtt{Gor}} : (\iota \multimap \iota) \multimap (\iota \multimap \iota) \multimap (\iota \multimap \iota) \multimap \iota$ to SℓPCF with the operational semantics described in Table 5. It is easy to check that the interpretation of $\overset{2}{\mathtt{Gor}}$ is the trace given above and that the full abstraction hold again for this extension of SℓPCF.

Next two Subsections show that the $\overset{2}{\mathtt{Gor}}$ is not strongly stable in the sense of Antonio Bucciarelli and Thomas Ehrhard [9, 10, 16]. For sake of simplicity, we will prove simply that a boolean version of $\overset{2}{\mathtt{Gor}}$ is not a strongly stable function.

Since $\overset{2}{\mathtt{Gor}}$ is not strongly stable, $\overset{2}{\mathtt{Gor}}$ is not syntactic sugar for SℓPCF, it is independent. Second, SℓPCF is not universal for the considered model. We conclude with some conjectures. We conjecture that the language SℓPCF + $\overset{2}{\mathtt{Gor}}$ is not universal w.r.t. our model. Worst, it does not make us able to define finite cliques of our linear model. We are proving that the definability of finite cliques and the universality, can be obtained by using SℓPCF (including `which?`) and a family of operator making us able to define $\overset{2}{\mathtt{Gor}}$. We are proving that S$\ell$PCF is fully-abstract and universal for the linear strongly stable (so, sequential [16]) considered above.

$$
\frac{\mathtt{P}_1\underline{0} \Downarrow \underline{0} \quad \mathtt{P}_1\underline{1} \Downarrow \underline{0} \quad \mathtt{P}_2\underline{0} \Downarrow \underline{1}}{\overset{2}{\mathtt{Gor}}\ \mathtt{P}_0\mathtt{P}_1\mathtt{P}_2 \Downarrow \underline{0}}\ (\overset{2}{\mathtt{Gor}}_0)
\qquad
\frac{\mathtt{P}_1\underline{0} \Downarrow \underline{1} \quad \mathtt{P}_1\underline{0} \Downarrow \underline{0} \quad \mathtt{P}_2\underline{1} \Downarrow \underline{0}}{\overset{2}{\mathtt{Gor}}\ \mathtt{P}_0\mathtt{P}_1\mathtt{P}_2 \Downarrow \underline{1}}\ (\overset{2}{\mathtt{Gor}}_1)
$$

$$
\frac{\mathtt{P}_1\underline{1} \Downarrow \underline{0} \quad \mathtt{P}_1\underline{0} \Downarrow \underline{1} \quad \mathtt{P}_2\underline{0} \Downarrow \underline{0}}{\overset{2}{\mathtt{Gor}}\ \mathtt{P}_0\mathtt{P}_1\mathtt{P}_2 \Downarrow \underline{2}}\ (\overset{2}{\mathtt{Gor}}_2)
\qquad
\frac{\mathtt{P}_1\underline{1} \Downarrow \underline{1} \quad \mathtt{P}_1\underline{1} \Downarrow \underline{1} \quad \mathtt{P}_2\underline{1} \Downarrow \underline{1}}{\overset{2}{\mathtt{Gor}}\ \mathtt{P}_0\mathtt{P}_1\mathtt{P}_2 \Downarrow \underline{3}}\ (\overset{2}{\mathtt{Gor}}_3)
$$

Table 5: A Second Order Gustave OR.

$$
\mathtt{b}\overset{2}{\mathtt{Gor}}_{k_1,k_2,k_3,k_4}(f_1,f_2,f_3) = \begin{cases} k_1 & \text{if } f_1(\mathbf{tt}) = \mathbf{tt}, f_2(\mathbf{ff}) = \mathbf{tt}, f_3(\mathbf{tt}) = \mathbf{ff} \\ k_2 & \text{if } f_1(\mathbf{tt}) = \mathbf{ff}, f_2(\mathbf{tt}) = \mathbf{tt}, f_3(\mathbf{ff}) = \mathbf{tt} \\ k_3 & \text{if } f_1(\mathbf{ff}) = \mathbf{tt}, f_2(\mathbf{tt}) = \mathbf{ff}, f_3(\mathbf{tt}) = \mathbf{tt} \\ k_4 & \text{if } f_1(\mathbf{ff}) = \mathbf{ff}, f_2(\mathbf{ff}) = \mathbf{ff}, f_3(\mathbf{ff}) = \mathbf{ff} \\ \bot & \text{otherwise} \end{cases}
$$
$$\text{where } k_1, k_2, k_3, k_4 \in \{\mathbf{tt}, \mathbf{ff}\}$$

Table 6: A boolean version of a Second Order Gustave-OR.

5.1 Strongly stable model

Definition 13. A *qualitative domain* [18] is a pair $\langle |Q|, Q\rangle$ where $|Q|$ is a set (called the *web*) and Q is a subset of $\wp(|Q|)$ satisfying the following conditions:

- $\emptyset \in Q$ and, if $a \in |Q|$ then $\{a\} \in Q$
- if $x \in Q$ and if $y \subseteq x$ then $y \in Q$
- if $D \subseteq Q$ is directed with respect to inclusion, the $\bigcup D \in Q$

The elements of Q are called states of the qualitative domain, and the qualitative domain itself will also be denoted Q.

Property 1. A qualitative domain Q is a coherence space when for all $u \subseteq |Q|$, if for all $a, b \in u, \{a, b\} \in Q$ then $u \in Q$.

If $\langle D, \leq\rangle$ is a poset and $A, B \subseteq D$ then we say that A is Egli-Milner smaller than B (written $A \sqsubseteq_{EM} B$) if

$$\forall x \in A.\exists y \in B.x \leq y \quad \text{and} \quad \forall y \in B.\exists x \in A.x \leq y$$

Definition 14. (Qualitative domain with coherence). A qualitative domain with coherence is a pair $\langle Q, \mathcal{C}(Q)\rangle$ where Q is a qualitative domain while $\mathcal{C}(Q)$ is a subset of $\wp_{fin}(Q)$ such that

- if $x \in Q$ then $\{x\} \in \mathcal{C}(Q)$
- if $A \in \mathcal{C}(Q)$ and $B \sqsubseteq_{EM} A$ then $B \in \mathcal{C}(Q)$
- if $D_1, \ldots, D_n$ is a family of directed subset of Q such that for any $d_1 \in D_1, \ldots d_n \in D_n$ the set $\{d_1, \ldots, d_n\} \in \mathcal{C}(Q)$ then $\{\bigvee D_1, \ldots, \bigvee D_n\} \in \mathcal{C}(Q)$

Intuitively, the notion of coherence corresponds to the notion of linear coherence in a sequential structure, see [10, 16].

Example 1. Let us consider the qualitative domain with coherence $\langle \mathbf{B}, \mathcal{C}(\mathbf{B})\rangle$ of booleans,

- $|\mathbf{B}| = \{\mathbf{tt}, \mathbf{ff}\}$
- $\mathbf{B} = \{\emptyset, \{\mathbf{tt}\}, \{\mathbf{ff}\}\}$
- $\mathcal{C}(\mathbf{B}) = \left\{ \begin{array}{l} \{\emptyset\}, \{\{\mathbf{tt}\}\}, \{\{\mathbf{ff}\}\}, \{\emptyset, \{\mathbf{tt}\}\}, \{\emptyset, \{\mathbf{ff}\}\}, \\ \{\emptyset, \{\mathbf{tt}\}, \{\mathbf{ff}\}\} \end{array} \right\}$

Morphisms between qualitative domains adapts the definition for coherence space. Let X and Y be qualitative domains.

- If $f : X \longrightarrow Y$ is a monotone function then f is continuous if and only if $f(\bigvee D) = \bigvee\{f(x)/x \in D\}$, for each directed $D \subseteq X$.
- If $f : X \longrightarrow Y$ is a continuous function then f is stable if and only if $\forall x, x' \in X$, $x \vee x' \in X$ implies $f(x \wedge x') = f(x) \wedge f(x')$.
- If $f : X \longrightarrow Y$ is a stable function then f is linear if and only if $f(\bot) = \bot$ and $\forall x, x' \in X$ bounded $f(x \vee x') = f(x) \vee f(x')$.

Definition 15. (Strongly stable function). A function $f : Q \to R$ between two qualitative domain with coherence is strongly stable if for all $X \in \mathcal{C}(Q)$ then $f(X) \in \mathcal{C}(R)$ and $f(\bigwedge X) = \bigwedge_{a \in X} f(a)$.

The strongly stable functions are similar to stable functions, but they have to preserve coherence as well as intersections of coherent sets of states and not just of bounded ones. The category obtained taking as objects qualitative domains with coherence and as morphisms the strongly stable functions is cartesian closed.

Definition 16. (Product). Let Q and R two qualitative domain with coherence. Their product $Q \times R$ is $\langle Q \& R, \mathcal{C}(Q \times$

$R)\rangle$ where $Q\&R$ is the product of beneath qualitative domains and, $X \in \mathcal{C}(Q \times R)$ iff $\pi_1(X) \in \mathcal{C}(Q)$ and $\pi_2(X) \in \mathcal{C}(R)$.

A set $X \subseteq A \times B$ is a *pairing* if $\pi_1(X) = A$ and $\pi_2(X) = B$. Recall that the set of strongly stable function between from Q to R is a qualitative domain [16].

Definition 17. (EXPONENTIAL OBJECT). Let Q and R two qualitative domains with coherence. Let R^Q be the qualitative domain of strongly stable function between from Q to R. The exponential object $Q \to R$ is $\langle R^Q, \mathcal{C}(Q \to R)\rangle$ where, for all $F \subseteq R^Q$, for all $X \in \mathcal{C}(Q)$, for all pairing $P \subseteq X \times F$, F belongs to $\mathcal{C}(Q \to R)$ iff by assuming $\mathsf{eval}(P) = \{f(x) | \langle x, f\rangle \in P\}$, the following two conditions hold

- $\mathsf{eval}(P) \in \mathcal{C}(R)$
- $\bigwedge \mathsf{eval}(P) = (\bigwedge F)(\bigwedge X)$

Example 2. Let us consider the qualitative domain $\mathbf{B} \to \mathbf{B}$ of strongly stable function between booleans. Let $a, b \in \{\mathtt{tt}, \mathtt{ff}\}$ we define some functions in $\mathbf{B} \to \mathbf{B}$ as follows

$$b^a = \begin{cases} b & \text{if } x = a \\ \bot & \text{otherwise} \end{cases}$$

We noted $\bot$ the bottom of qualitative domains, corresponding to emptyset in case of coherence spaces.

We sketch the proof that $F = \{\mathtt{tt}^{\mathtt{tt}}, \mathtt{ff}^{\mathtt{tt}}, \mathtt{tt}^{\mathtt{ff}}\}$ belong to $\mathcal{C}(\mathbf{B} \to \mathbf{B})$. For all $X \in \mathcal{C}(\mathbf{B})$ there are two cases. [3]

- If $\emptyset \in X$ then for every pairing $P \subseteq X \times F$, so we have $\emptyset \in \mathsf{eval}(P)$, since the considered functions are strict. Thus $\mathsf{eval}(P) \in \mathcal{C}(\mathbf{B})$ (see Example 1).
- If $\emptyset \notin X$ then X is singleton (see Example 1) and easily we can check that for every pairing $P \subseteq X \times F$, so we have $\mathsf{eval}(P) \in \mathcal{C}(\mathbf{B})$.

The second condition of Definition 17, about the commutation of greatest lower bound, can be easily checked to be true.

5.2 Second Order Gustave Or is not strongly stable

For sake of simplicity, we prove simply that a boolean version of $\mathtt{G}\overset{2}{\mathtt{o}}\mathtt{r}$ is not a strongly stable function. We replace everywhere $\underline{0}$ by $\mathtt{tt}$ and $\mathtt{succ}\,\underline{\mathtt{n}}$ by $\mathtt{ff}$, respecting our notations. Let us consider the second-order Gustave-Or defined in Table 6. It is straightforward to check that the definition of $\mathtt{bG}\overset{2}{\mathtt{o}}\mathtt{r}$ is well given and it is a boolean generalization of a Second Order Gustave Or.

[3]For sake of conciseness, we use the exponential notation in order to denote pairs of booleans: exponent (base) corresponds respectively to first (second) projection.

We prove that $\mathtt{bG}\overset{2}{\mathtt{o}}\mathtt{r}_{k_1,k_2,k_3,k_4}$ is not a strongly stable function (see Definition 15) for all $k_1, k_2, k_3, k_4 \in \{\mathtt{tt}, \mathtt{ff}\}$. Let us consider the functions in Example 2 above. We take 3 elements of $(\mathbf{B} \multimap \mathbf{B}) \times (\mathbf{B} \multimap \mathbf{B}) \times (\mathbf{B} \multimap \mathbf{B})$, namely $t_1 = \langle \mathtt{tt}^{\mathtt{tt}}, \mathtt{tt}^{\mathtt{ff}}, \mathtt{ff}^{\mathtt{tt}}\rangle$, $t_2 = \langle \mathtt{ff}^{\mathtt{tt}}, \mathtt{tt}^{\mathtt{tt}}, \mathtt{tt}^{\mathtt{ff}}\rangle$ and $t_3 = \langle \mathtt{tt}^{\mathtt{ff}}, \mathtt{ff}^{\mathtt{tt}}, \mathtt{tt}^{\mathtt{tt}}\rangle$. The set $I = \{t_1, t_2, t_3\}$ is in the coherence set of the domain corresponding to $(\mathbf{B} \multimap \mathbf{B}) \times (\mathbf{B} \multimap \mathbf{B}) \times (\mathbf{B} \multimap \mathbf{B})$, since the projections are in the coherence set of $\mathbf{B} \multimap \mathbf{B}$. We have two cases.

- If $k_1 = k_2 = k_3$ then $\bigwedge \mathtt{bG}\overset{2}{\mathtt{o}}\mathtt{r}_{k_1,k_2,k_3,k_4}(I) = k_1$, but $\mathtt{bG}\overset{2}{\mathtt{o}}\mathtt{r}_{k_1,k_2,k_3,k_4}(\bigwedge I) = \mathtt{bG}\overset{2}{\mathtt{o}}\mathtt{r}_{k_1,k_2,k_3,k_4}(\emptyset) = \emptyset$.
- Otherwise, there exists $i, j \in \{1, 2, 3\}$ such that $k_i \neq k_j$, thus
$$\mathtt{bG}\overset{2}{\mathtt{o}}\mathtt{r}_{k_1,k_2,k_3,k_4}(I) \notin \mathcal{C}(\mathbf{B})$$
since
$$\{k_i, k_j\} \subseteq \mathtt{bG}\overset{2}{\mathtt{o}}\mathtt{r}_{k_1,k_2,k_3,k_4}(I)$$
and $\emptyset \notin \mathtt{bG}\overset{2}{\mathtt{o}}\mathtt{r}_{k_1,k_2,k_3,k_4}(I)$.

Therefore $\mathtt{bG}\overset{2}{\mathtt{o}}\mathtt{r}_{k_1,k_2,k_3,k_4}$ is not strongly stable, for all k_1, k_2, k_3, k_4.

Acknowledgments

We wish to express our gratitude to Antonio Bucciarelli and Marco Gaboardi for some useful discussions about topics considered in this paper.

6. REFERENCES

[1] S. Alves, M. Fernández, M. Florido, and I. Mackie. The power of linear functions. In Z. Ésik, editor, *Proceedings of the 20th International Workshop on Computer Science Logic*, volume 4207 of *Lecture Notes in Computer Science*, pages 119–134. Springer-Verlag, 2006.

[2] A. Asperti and S. Guerrini. *The optimal implementation of functional programming languages*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK, 1998.

[3] A. Asperti and G. Longo. *Categories, Types, and Structures: An Introduction to Category Theory for the Working Computer Scientist*. Foundations of Computing Series. The MIT Press, Cambridge, MA, 1991.

[4] A. Asperti and L. Roversi. Intuitionistic light affine logic. *ACM Transactions on Computational Logic*, 3(1):1–39, 2002.

[5] S. Bellantoni, K. H. Niggl, and H. Schwichtenberg. Higher type recursion, ramification and polynomial time. *Annals of Pure and Applied Logic*, 104:17–30, 2000.

[6] M. Berger, K. Honda, and N. Yoshida. Sequentiality and the pi-calculus. In S. Abramsky, editor, *Proceedings of Typed Lambda Calculi and Applications, 5th International Conference, TLCA 2001, Krakow, Poland, May 2-5, 2001*, volume 2044 of *Lecture Notes in Computer Science*, pages 29–45. Springer-Verlag, 2001.

[7] G. Berry. Stable models of typed λ-calculi. In G. Ausiello and C. Böhm, editors, *Fifth International Colloquium on Automata, Languages and Programming - - ICALP'78, Udine, Italy, July 17-21, 1978*, volume 62 of *Lecture Notes in Computer Science*, pages 72–89. Springer-Verlag, 1978.

[8] G. Berry, P.-L. Curien, and J.-J. Lévy. Full abstraction for sequential languages: the state of the art. In M. Nivat and J. Reynolds, editors, *Algebraic Semantics*, pages 89–132. Cambridge University Press, 1985.

[9] A. Bucciarelli and T. Ehrhard. Sequentiality and strong stability. In *Proceedings of the Symposium on Logic in Computer Science - LICS'91*, pages 138–145, 1991.

[10] A. Bucciarelli and T. Ehrhard. Sequentiality in an extensional framework. *Information and Computation*, 110(2):265–296, 1994.

[11] R. Cartwright, P.-L. Curien, and M. Felleisen. Fully abstract semantics for observably sequential languages. *Information and Computation*, 111(2):297–401, 1994.

[12] N. Cutland. *Computability: An Introduction to Recursive Function Theory*. Cambridge University Press, 1980.

[13] U. Dal Lago. The geometry of linear higher-order recursion. In *Proceedings of 20th IEEE Symposium on Logic in Computer Science (LICS 2005), 26-29 June 2005, Chicago, IL, USA*, pages 366–375. IEEE Computer Society Press, 2005.

[14] U. Dal Lago, S. Martini, and L. Roversi. Higher-order linear ramified recurrence. In S. Berardi, M. Coppo, and F. Damiani, editors, *International Workshop Types for Proofs and Programs, TYPES 2003, Torino, Italy, April 30 - May 4, 2003*, volume 3085 of *Lecture Notes in Computer Science*, pages 178–193. Springer-Verlag, 2004.

[15] M. Davis and E. J. Weyuker. *Computability, Complexity and Languages*. Computer Science and Applied Mathematics. Academic Press, 1983.

[16] T. Ehrhard. Hypercoherences: A strongly stable model of linear logic. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Proceedings of the Workshop on Advances in Linear Logic*, number 222 in London Mathematical Society Lecture Note Series, pages 83–108. Cambridge University Press, Ithaca, New York, 1995.

[17] M. Gaboardi and L. Paolini. Syntactical, operational and denotational linearity. Presented at *Workshop on Linear Logic, Ludics, Implicit Complexity and Operator Algebras. Dedicated to Jean-Yves Girard on his 60th birthday*, Certosa di Pontignano, Siena, May 2007.

[18] J.-Y. Girard. The system F of variable types, fifteen years later. *Theoretical Computer Science*, 45(2):159–192, 1986.

[19] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[20] J.-Y. Girard. Locus solum: from the rules of logics to the logics of rules. *Mathematical Structures in Computer Science*, 11:301–506, 2001.

[21] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, 1989.

[22] C. A. Gunter. *Semantics of Programming Languages: Structures and Techniques*. Foundations of Computing Series. The MIT Press, Cambridge, MA, 1992.

[23] J. M. E. Hyland and L. C.-H. Ong. Pi-calculus, dialogue games and PCF. In *Proceedings of Conference on Functional Programming Languages and Computer Architecture FPCA95, La Jolla, CA, USA*, pages 96–107. ACM Press, 1995.

[24] J. W. Klop. New fix point combinators from old. In E. Barendsen, V. Capretta, H. Geuvers, and M. Niqui, editors, *Reflections on Type Theory*. Radboud University Nijmegen, 2007.

[25] C. Laneve and B. Victor. Solos in concert. *Mathematical Structures in Computer Science*, 13(5):657–683, 2003.

[26] J. R. Longley. The sequentially realizable functionals. *Annals of Pure and Applied Logic*, 117:1–93, 2002.

[27] A. Meyer. Semantical paradigms. In *Proceedings of the Third Annual Symposium on Logic in Computer Science - LICS*, pages 236–253, 1988. Notes for an invited lecture with two appendices by Stavros Cosmadakis.

[28] C.-H. L. Ong. Correspondence between operational and denotational semantics: the full abstraction for PCF. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 4, pages 269–356. Oxford University Press, 1995.

[29] L. Paolini. A stable programming language. *Information and Computation*, 204(3):339–375, 2006.

[30] L. Paolini and M. Piccolo. Processing linear programs. In preparation, 2008.

[31] L. Petersen, R. Harper, K. Crary, and F. Pfenning. A type theory for memory allocation and data layout. In *The 30th SIGPLAN-SIGACT Symposium on Principles of Programming Languages, New Orleans, Louisiana, January 15-17, 2003. ACM SIGPLAN Notices 38(1), January*, pages 172–184, 2003.

[32] G. D. Plotkin. LCF considerd as a programming language. *Theoretical Computer Science*, 5:225–255, 1977.

[33] D. Sangiorgi and D. Walker. *The π-calculus: a theory of mobile processes*. Cambridge University Press, Cambridge, 2001.

[34] D. N. Turner and P. Wadler. Operational interpretations of linear logic. *Theoretical Computer Science*, 227(1–2):231–248, 1999.

[35] N. Yoshida, M. Berger, and K. Honda. Strong normalisation in the PI-calculus. *Information and Computation*, 191(2):145–202, 2004.