Math. Struct. in Comp. Science (2016), *vol.* 26, *pp.* 829–867.
doi:10.1017/S0960129514000401 First published online 10 November 2014

On the reification of semantic linearity

MARCO GABOARDI[†], LUCA PAOLINI[‡] and MAURO PICCOLO[‡]

[†]*School of Computing, University of Dundee, Dundee, DD1 4HN, Scotland, U.K.*
[‡]*Dipartimento di Informatica, Università degli Studi di Torino, C.so Svizzera 185, 10149 Torino, Italy*
Email: paolini@di.unito.it

Received 21 April 2013; revised 19 November 2013

Linearity is a multi-faceted and ubiquitous notion in the analysis and development of programming language concepts. We study linearity in a denotational perspective by picking out programs that correspond to linear functions between domains.
We propose a PCF-like language imposing linear constraints on the use of variable to program only linear functions. To entail a full abstraction result, we introduce some higher-order operators related to exception handling and parallel evaluation. We study several notions of operational equivalence and show them to coincide with our language. Finally, we present a new operational evaluation of the language that provides the base for a real implementation. It exploits the denotational linearity to provide an efficient evaluation semantics SECD-like, that avoids the use of closures.

1. Introduction

Linearity is a key tool to support a conscious use of resources in programming languages. A non-exhaustive list of its uses includes garbage collection, memory management and aliasing control, description of digital circuits, process channels and messages management, languages for quantum computations, etc. A survey of several variants of linear type systems proposed in the literature is in Walker (2005). This broad spectrum of applications highlights the fact that linearity is a multi-faceted abstract concept which can be considered in different perspectives. For instance, notions of *syntactical linearity* can be considered when variables are used once (in suitable senses), e.g. Alves *et al.* (2007) and Hindley (1989). On the other hand, if redexes cannot be discarded or duplicated during reduction, like in Hindley (1989), then a kind of *operational linearity* is achieved. This is related to the notion of *simple term* in λ-calculus (Klop 2007), which suggests a kind of linearity on reductions, unrelated from a specific strategy.

Although ideas that can be tracked to linearity have been implicitly used in programming languages for many years, the introduction of linear logic (Girard 1987) is a redoubtable milestone in this setting. Linear logic arises from a sharp semantic analysis of stable domains where stable functions have been decomposed into linear functions and exponential domain constructors. Such a decomposition is patently reflected in the syntax of linear logic. Moreover, it suggests a new approach to linearity: *denotational linearity*. In a programming perspective, denotational linearity says that programs (i.e. closed terms) should correspond via a suitable interpretation to linear function on some specific domains, e.g. the linear models introduced in Bucciarelli *et al.* (2010), Bucciarelli and Ehrhard (1993), Girard (1987) and Huth (1993). By tackling this correspondence

minutely, some important contributions to the theory of programming can be obtained. For instance, if the intended domain includes all the computable functions, this analysis provides Turing-complete languages with weak syntactic linear constraints on variables, and new linear operators that, in a higher-type computability perspective (Longley 2000; Normann 2006), increase the expressivity of linear languages.

Following the denotational linearity approach, we study (recursive) linear functions in the *type theory* $\mathcal{L}$ of coherence spaces and linear functions introduced by Girard (1987). An attractive aspect of this type theory is the simplicity of its formulation. We present a language, named S$\not$PCF$_\star$ that is *fully abstract* for this type theory. S$\not$PCF$_\star$ can be described as the combination of the following ingredients.

— A core PCF-like language, named core-S$\not$PCF, characterized by a constrained management of variables and by an operational semantics based on a careful evaluation of program mixing call-by-name and call-by-value. This core language is Turing complete and can be soundly interpreted in $\mathcal{L}$.

— A higher-order operator, named `which?`, that corresponds to a primitive form of exception handling. `which?` permits to obtain besides the result of an evaluation also some information about the way the result is computed.

— A higher-order operator, named ℓ`et-`ℓ`or`, that permits to explore in parallel different program branches. A distinguished feature of ℓ`et-`ℓ`or` is that it permits to share some information between the different branches.

The combination of the core language with the two higher-order linear operators `which?` and ℓ`et-`ℓ`or` is essential to gain a *finite definability* result (Curien 2007), i.e. the definability of all the finite cliques of the considered linear model. The finite definability result is not only the key ingredient to obtain the full abstraction result, but it is also important to show the programming patterns that the language offers. The importance of the full abstraction comes from the fact that it allows us to reason about program in a compositional way. Moreover, full abstraction allows us also to prove the coincidence of three different definitions of operational equivalence. These definitions make the reasoning simpler on the equivalence between programs by permitting to consider restricted families of contexts.

An interesting remark is that S$\not$PCF$_\star$ is neither syntactically linear nor operationally linear in tight sense, albeit its finitary fragment (the set of programs which does not involve recursion) respects some syntactical and operational forms of linearity (see Remark 1). We present S$\not$PCF$_\star$ by means of a self-explanatory operational semantics that, quite inefficiently, requires for some operators an infinite branching search in the evaluation tree. Then, we show that S$\not$PCF$_\star$ programs can be evaluated in an efficient way by means of another abstract machine that traces out and records linear information to finitely prune the infinite branching search tree of the evaluation of S$\not$PCF$_\star$. The latter tracing operational machine is a very concrete by-product of our results, since it tailors on denotationally linear constraints a very efficient evaluation of a Turing-complete

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

programming language endowed of parallel features and exception-like mechanism, namely SℓPCF$_\star$.

The study of the relations between languages and models is a classic theme in denotational semantics (Curien 2007; Ong 1995). The full abstraction for PCF has led to the development of very sophisticated semantics techniques (as Abramsky *et al.* (2000) and Hyland and Ong (2000)) and revealed relevant programming principles and operational constructions. In particular, several works have studied how to extend PCF by different operators in order to achieve full abstraction compared to some classical models. For example, this approach has been followed by Plotkin (1977) for the continuous Scott model, by Berry and Curien (1982) for the sequential algorithms model, by Abramsky and McCusker, (1996) for a particular game model, by Longley (2002) for the strongly stable model and by Paolini (2006) for the stable model. Remark that all higher-type operators introduced in the works above cannot be directly interpreted in the linear model. This is another motivation for our search for new operators.

Many linear languages with different goals have been proposed so far in the literature. Recently, in the studies of syntactical linearity, Alves *et al.*, (2006; 2007; 2010) have proposed several syntactical linear languages in order to characterize different classes of computable functions. To explore their extension, higher-type semantically-motivated operators, is another interesting open problem.

Two PCF-like languages embedding linearity notions have been proposed in Bierman (2000) and Bierman *et al.* (2000). These languages are not denotationally linear, in the sense that not all their closed terms are in correspondence with linear functions of a suitable domain. In particular, they cannot be interpreted in the linear model considered here. Despite this fact, the authors of Bierman (2000) and Bierman *et al.* (2000) give some interesting results on the relations between several forms of operational reasoning, in the context of the linear decomposition. Our results on the coincidence of three operational equivalences can be viewed as further contributions in those topics.

The results we present in this paper extend and complete our previous works: Gaboardi and Paolini (2007), Gaboardi *et al.* (2011) and Paolini and Piccolo (2008). The core-SℓPCF$_\star$ language and the `which?` operator were first introduced by Gaboardi and Paolini (2007) and Paolini and Piccolo (2008), respectively. Paolini and Piccolo (2008) proved the language combining these two components, named SℓPCF, correct for the type theory $\mathcal{L}$. Moreover, they showed a full abstraction result for stable closed terms, i.e. terms not containing free occurrences of a particular kind of variables used for recursion. Gaboardi *et al.* (2011) extended SℓPCF by means of the ℓ`et-`ℓ`or` operator obtaining in this way the full abstraction for terms that can contain free occurrences of stable variables. The enhanced tracing operational machine we present in this paper greatly improves the efficiency of tracing operational machine defined in Gaboardi *et al.* (2011).

1.1. *Synopsis*

In Section 2, we introduce the core-SℓPCF$_\star$ language, the linear type theory $\mathcal{L}$ and the interpretation. We show that the interpretation is sound and that a full abstraction result holds for stable closed terms. In Section 3, we show that for the core language of

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

Section 2, a more general full abstraction result does not hold. In Section 4, we introduce the two operators `which?` and ℓ`et-`ℓ`or`, completing in this way the introduction of SℓPCF$_\star$. We provide some programming examples and we prove finite definability and full abstraction for it. In Section 5, we show the coincidence of three operational equivalences. In Section 6, we give an abstract machine for SℓPCF$_\star$ that traces the linear use of terms. This provides the base for an efficient implementation of our language.

2. The core of a linear programming language

As described in the introduction, following the denotational linearity approach, our aim is to design a programming language corresponding to a particular linear type theory $\mathcal{L}$. Here we start by presenting a core part of this language, named core-SℓPCF, then we present the type theory $\mathcal{L}$ and we show how core-SℓPCF can be soundly interpreted in $\mathcal{L}$ and how a limited form of full abstraction can be obtained.

The types of the language are defined as following:

$$\begin{array}{rcll} \sigma, \tau & ::= & \iota & \text{(natural numbers)} \\ & | & \sigma \multimap \tau & \text{(arrow or function type).} \end{array}$$

As usual, $\multimap$ associates to the right so $\sigma_1 \multimap \sigma_2 \multimap \sigma_3$ is read as $\sigma_1 \multimap (\sigma_2 \multimap \sigma_3)$.

Let Var^σ be a numerable set of variables of type σ. If $\sigma \neq \iota$ then, let SVar^σ be a numerable set of variables disjoint from Var^σ. Variables in Var^ι are named *ground variables*. Variables in $\ell\mathrm{Var} = \bigcup_{\sigma,\tau\in\mathbb{T}} \mathrm{Var}^{\sigma\multimap\tau}$ are named *linear variables*. Variables in $\mathrm{SVar} = \bigcup_{\sigma\neq\iota} \mathrm{SVar}^\sigma$ are named *stable variables*. Latin letters $\mathtt{x}^\sigma, \mathtt{y}^\sigma, \mathtt{f}^\sigma, \ldots$ denote variables in Var^σ. Moreover, $\boldsymbol{f}_0^\sigma, \boldsymbol{f}_1^\sigma, \boldsymbol{f}_2^\sigma, \ldots$ denote stable variables. Last, $\varkappa$ is a wild-card for all the variables.

Definitiona 1. Core-SℓPCF *pre-terms* are defined by the following grammar:

$$\begin{array}{rcll} \mathtt{M} & ::= & \varkappa^\tau & \text{variable} \\ & | & \mathbf{0} & \text{zero} \\ & | & \mathbf{s} & \text{successor} \\ & | & \mathbf{p} & \text{predecessor} \\ & | & \ell\mathtt{if\ M\ M\ M} & \text{conditional} \\ & | & \mathtt{MM} & \text{application} \\ & | & \lambda\mathtt{x}^\sigma.\mathtt{M} & \lambda\text{-abstraction} \\ & | & \mu\boldsymbol{f}^\sigma.\mathtt{M} & \mu\text{-abstraction} \end{array}$$

We write $\underline{\mathtt{n}}$ for $\mathbf{s}(\cdots(\mathbf{s0})\cdots)$ where $\mathbf{s}$ is applied n-times to $\mathbf{0}$, and we denote $\mathcal{N} = \{\mathbf{0}, \ldots, \underline{\mathtt{n}}, \ldots\}$ the set of *numerals*. When the type information is irrelevant, we will omit type labels. We partitioned variables in three main categories: $\mathrm{Var}^\iota, \ell\mathrm{Var}, \mathrm{SVar}$. Other choices are possible, but we preferred this one in order to emphasize their differences. The two kinds of abstraction act as binders in the standard way, λ-abstraction binds ground and linear variables (however, the first set of variables entails a call-by-value policy and the latter a call-by-name one), while μ-abstraction binds stable ones. Free variables of any kind (FV), free linear variables (ℓFV), free stable variables (SFV), closed

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

Table 1. *(a) Type system and (b) operational semantics.*

$$\frac{}{\vdash \mathbf{0} : \iota}\ (z) \qquad \frac{}{\vdash \mathbf{s} : \iota \multimap \iota}\ (s) \qquad \frac{}{\vdash \mathbf{p} : \iota \multimap \iota}\ (p) \qquad \frac{}{\varkappa^{\sigma} \vdash \varkappa : \sigma}\ (v) \qquad \frac{\Gamma \vdash \mathtt{M} : \tau}{\Gamma, \mathtt{x}^{\iota} \vdash \mathtt{M} : \tau}\ (gw) \qquad \frac{\Gamma \vdash \mathtt{M} : \tau}{\Gamma, \mathtt{f}^{\sigma} \vdash \mathtt{M} : \tau}\ (sw)$$

$$\frac{\Gamma, \mathtt{x}_1^{\iota}, \mathtt{x}_2^{\iota} \vdash \mathtt{M} : \tau}{\Gamma, \mathtt{x}^{\iota} \vdash \mathtt{M}[\mathtt{x}/\mathtt{x}_1, \mathtt{x}_2] : \tau}\ (gc) \qquad \frac{\Gamma, \mathtt{x}^{\sigma} \vdash \mathtt{M} : \tau}{\Gamma \vdash \lambda \mathtt{x}^{\sigma}.\mathtt{M} : \sigma \multimap \tau}\ (\lambda) \qquad \frac{\Gamma \vdash \mathtt{M} : \iota \quad \Delta \vdash \mathtt{L} : \iota \quad \Delta \vdash \mathtt{R} : \iota}{\Gamma, \Delta \vdash \ell\mathtt{if}\ \mathtt{M}\ \mathtt{L}\ \mathtt{R} : \iota}\ (lif)$$

$$\frac{\Gamma, \mathtt{f}_1^{\sigma}, \mathtt{f}_2^{\sigma} \vdash \mathtt{M} : \tau}{\Gamma, \mathtt{f}^{\sigma} \vdash \mathtt{M}[\mathtt{f}/\mathtt{f}_1, \mathtt{f}_2] : \tau}\ (sc) \qquad \frac{\Gamma, \mathtt{f}^{\sigma} \vdash \mathtt{M} : \sigma \quad \Gamma\restriction\ell = \varnothing}{\Gamma \vdash \mu\mathtt{f}^{\sigma}.\mathtt{M} : \sigma}\ (\mu) \qquad \frac{\Gamma \vdash \mathtt{M} : \sigma \multimap \tau \quad \Delta \vdash \mathtt{N} : \sigma}{\Gamma, \Delta \vdash \mathtt{MN} : \tau}\ (ap)$$

$$\frac{}{\mathbf{0} \Downarrow \mathbf{0}}\ (0) \qquad \frac{\mathtt{M} \Downarrow \underline{\mathtt{n}}}{\mathbf{s}\mathtt{M} \Downarrow \mathbf{s}\underline{\mathtt{n}}}\ (s) \qquad \frac{\mathtt{M} \Downarrow \mathbf{s}\underline{\mathtt{n}}}{\mathbf{p}\mathtt{M} \Downarrow \underline{\mathtt{n}}}\ (p) \qquad \frac{\mathtt{M} \Downarrow \mathbf{0} \quad \mathtt{L} \Downarrow \underline{\mathtt{m}}}{\ell\mathtt{if}\ \mathtt{M}\ \mathtt{L}\ \mathtt{R} \Downarrow \underline{\mathtt{m}}}\ (ifl) \qquad \frac{\mathtt{M} \Downarrow \mathbf{s}(\underline{\mathtt{n}}) \quad \mathtt{R} \Downarrow \underline{\mathtt{m}}}{\ell\mathtt{if}\ \mathtt{M}\ \mathtt{L}\ \mathtt{R} \Downarrow \underline{\mathtt{m}}}\ (ifr)$$

$$\frac{\mathtt{M}[\mathtt{N}/\mathtt{f}]\mathtt{P}_1 \cdots \mathtt{P}_i \Downarrow \underline{\mathtt{n}}}{(\lambda \mathtt{f}^{\sigma \multimap \tau}.\mathtt{M})\mathtt{N}\mathtt{P}_1 \cdots \mathtt{P}_i \Downarrow \underline{\mathtt{n}}}\ (\lambda^{\multimap}) \qquad \frac{\mathtt{M}[\mu\mathtt{f}.\mathtt{M}/\mathtt{f}]\mathtt{P}_1 \cdots \mathtt{P}_i \Downarrow \underline{\mathtt{n}}}{(\mu\mathtt{f}.\mathtt{M})\mathtt{P}_1 \cdots \mathtt{P}_i \Downarrow \underline{\mathtt{n}}}\ (\mu) \qquad \frac{\mathtt{N} \Downarrow \underline{\mathtt{m}} \quad \mathtt{M}[\underline{\mathtt{m}}/\mathtt{x}]\mathtt{P}_1 \cdots \mathtt{P}_i \Downarrow \underline{\mathtt{n}}}{(\lambda \mathtt{x}^{\iota}.\mathtt{M})\mathtt{N}\mathtt{P}_1 \cdots \mathtt{P}_i \Downarrow \underline{\mathtt{n}}}\ (\lambda^{\iota})$$

and open terms are defined as expected. $\mathtt{M}[\mathtt{N}/\varkappa]$ denotes the capture-free substitution of all free occurrences of $\varkappa$ in $\mathtt{M}$ by $\mathtt{N}$. As usual application associates to the left, i.e. $\mathtt{M}\mathtt{N}_1 \dots \mathtt{N}_k$ abbreviates $(\dots((\mathtt{M}\mathtt{N}_1)\mathtt{N}_2)\dots\mathtt{N}_k)$ and we routinely omit parenthesis as we safely can.

Core-SℓPCF *terms* are pre-terms that are well typed. We consider typing judgments of the shape $\Gamma \vdash \mathtt{M} : \sigma$ where $\mathtt{M}$ is a pre-term, σ is a linear type and Γ is a *basis*, that is a finite list of variables in $\mathrm{Var} \cup \mathrm{SVar}$, where each variable appears at most once. We denote $\Gamma\restriction\mathrm{S}$ (resp $\Gamma\restriction\iota$, $\Gamma\restriction\ell$) the restriction of the basis Γ containing only variables in SVar (resp. in Var^{ι}, $\ell\mathrm{Var}$). We denote Γ, Δ, $\Gamma \cup \Delta$ and $\Gamma \cap \Delta$ the disjoint union, the union and the intersection of two basis respectively. We write $\mathtt{M}^{\sigma}$ to signify that there is a basis Γ such that $\Gamma \vdash \mathtt{M} : \sigma$.

Definitiona 2. The pre-terms typable by using the type system in Table 1(a) are the *terms* of core-SℓPCF.

We remark that the type system contains *weakening* and *contraction* rules only for ground and stable variables (viz. (gw), (sw), (gc) and (sc)), while linear variables are managed by means of linear-typing rules (the rule $\ell\mathtt{if}$ is additive). Therefore, it is easy to check that the following rules are derivable rules in our system.

$$\frac{\Gamma_1 \vdash \mathtt{M} : \sigma \multimap \tau \quad \Gamma_2 \vdash \mathtt{N} : \sigma \qquad \Gamma_1\restriction\ell \cap \Gamma_2\restriction\ell = \varnothing.}{\Gamma_1 \cup \Gamma_2 \vdash \mathtt{M}\,\mathtt{N} : \sigma}$$

$$\frac{\Gamma_0 \vdash \mathtt{M} : \iota \qquad \Gamma_1 \vdash \mathtt{P} : \iota \qquad \Gamma_2 \vdash \mathtt{Q} : \iota \qquad \Gamma_1\restriction\ell = \Gamma_2\restriction\ell \qquad \Gamma_0\restriction\ell \cap \Gamma_1\restriction\ell = \varnothing.}{\Gamma_0 \cup \Gamma_1 \cup \Gamma_2 \vdash \ell\mathtt{if}\ \mathtt{M}\ \mathtt{P}\ \mathtt{Q} : \iota}$$

A closed term of ground type is called program and P denotes the set of all programs.

Definitiona 3. The *evaluation relation* $\Downarrow$ between programs of core-SℓPCF and numerals is the smallest relation inductively satisfying the rules of Table 1(b). For any program $\mathtt{M}$, 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 $\mathtt{M} \Uparrow$.

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

We remark that the evaluation is carried out in a call-by-name fashion by the rule $\lambda^{-\circ}$, and in a call-by-value fashion by the rule $\lambda^{!}$. Patently all terms appearing in the evaluation rules are well typed. To be more precise, in the rule $\lambda^{-\circ}$, we are assuming that $\tau = \sigma_1 \multimap \ldots \multimap \sigma_i \multimap \iota$ where σ_k is the type of $\mathtt{P}_k$. Moreover, note that the evaluation of the program $\mathbf{p}\,\underline{0}$ diverges.

Remark 1. Note that the finitary fragment of S$\mathscr{A}$PCF (i.e. the language obtained by avoiding stable variables and μ-abstractions) contains terms like $\lambda \mathtt{f}^{!\multimap !}.\ell\mathtt{if}\ \underline{5}\ (\mathtt{f}\ \underline{0})\ (\mathtt{f}\ \underline{1})$ and $\lambda \mathtt{f}^{!\multimap !\multimap !}\mathtt{x}^{!}.\mathtt{fxx}$ which are not syntactically linear. Moreover, S$\mathscr{A}$PCF is not operationally linear because in the following term

$$(\lambda \mathtt{f}^{!\multimap !}.\ell\mathtt{if}\ \underline{5}\ (\mathtt{f}\ \underline{0})\ (\mathtt{f}\ \underline{1}))((\lambda \mathtt{x}^{!}.\lambda \mathtt{y}^{!}.\mathtt{y})\underline{5})$$

the redex $(\lambda \mathtt{x}^{!}.\lambda \mathtt{y}^{!}.\mathtt{y})\underline{5}$ is duplicated during the evaluation.

We name some terms. Let $\Omega^{\iota} := \mathbf{p}\,\underline{0}$ and inductively, if $\sigma_0 = \mu_1 \multimap \ldots \multimap \mu_{\mathtt{m}} \multimap \iota$ for some $\mathtt{m} \in \mathbb{N}$, then

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

If $\sigma \neq \iota$ then it is possible to define Ω^{σ} as $\mu_{\boldsymbol{F}}^{\sigma}.\boldsymbol{F}$ and, still, to satisfy the Lemma 1. Nevertheless, the crucial use of Ω^{σ} is to define the approximants, i.e. terms approximating μ-abstraction that avoids the use of (additional) μ-abstractions. In this way, we can prove the Lemma 5 by adapting the Tait technique used in Plotkin (1977). By using Ω^{σ}, it is possible to define approximants of a term having shape $\mu_{\boldsymbol{F}}.\mathtt{M}^{\sigma}$ as follows,

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

It is easy to check that the μ-abstractions in $\mu^n{}_{\boldsymbol{F}}.\mathtt{M}^{\sigma}$ are strictly less than that in $\mu_{\boldsymbol{F}}.\mathtt{M}^{\sigma}$.

Lemma 1. Let $\mathtt{M}_0^{\sigma_0}, \ldots, \mathtt{M}_{\mathtt{m}}^{\sigma_{\mathtt{m}}}$ be a sequence of closed terms ($m \geqslant 0$).

1. $\Omega^{\sigma_0 \multimap \ldots \multimap \sigma_{\mathtt{m}} \multimap \iota}\mathtt{M}_0 \ldots\ \mathtt{M}_{\mathtt{m}}$ is a program and $\Omega^{\sigma_0 \multimap \ldots \multimap \sigma_{\mathtt{m}} \multimap \iota}\mathtt{M}_0 \ldots \mathtt{M}_{\mathtt{m}} \Uparrow$.
2. Let $(\mu_{\boldsymbol{F}}.\mathtt{P}^{\sigma})\mathtt{M}_0 \ldots \mathtt{M}_{\mathtt{m}}$ be a program.
   $(\mu_{\boldsymbol{F}}.\mathtt{P}^{\sigma})\mathtt{M}_0 \ldots \mathtt{M}_{\mathtt{m}} \Downarrow \underline{\mathtt{n}}$ if and only if $(\mu^{k+1}{}_{\boldsymbol{F}}.\mathtt{P}^{\sigma})\mathtt{M}_0 \ldots \mathtt{M}_{\mathtt{m}} \Downarrow \underline{\mathtt{n}}$, for some $k \in \mathbb{N}$.

Proof. (1) The proof can be done by induction on m. (2) Both implications can be proved by induction on derivations proving the hypothesis. □

2.1. *Pairing*

In the following sections, we need pairing and projections operators on natural numbers. This can be defined as follows. If $m, n \in \mathbb{N}$ then $< m, n >= 2^m(2n+1) - 1$ is our pairing function. Projections from $z \in \mathbb{N}$ can be defined by functions

$$pi_1(z) = \min_{x \leqslant z}[(\exists y)_{\leqslant z}(z =< x, y >)]\ ; \qquad pi_2(z) = \min_{y \leqslant z}[(\exists x)_{\leqslant z}(z =< x, y >)]\ .$$

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

Such functions induce a bijection, details can be found either, in p.41, p.73 of Cutland (1980) or in p.47 of Davis and Weyuker (1983). The reader can easily verify that

$$\begin{aligned}
&\mathtt{Sum} = \mu_{\boldsymbol{F}}.\lambda \mathtt{x}^{\iota}\mathtt{y}^{\iota}.\ell\mathtt{if\ x\ y}\,(\boldsymbol{F}(\mathbf{p}\,\mathtt{x})(\mathbf{s}\,\mathtt{y}))\\
&\mathtt{Prod} \equiv \mu_{\boldsymbol{F}}.\lambda \mathtt{x}^{\iota}\mathtt{y}^{\iota}.\ell\mathtt{if\ x}\ \underline{0}\ (\mathtt{Sum\,y}\,(\boldsymbol{F}(\mathbf{p}\,\mathtt{x})\,\mathtt{y}))\\
&\mathtt{Exp2} \equiv \mu_{\boldsymbol{F}}.\lambda \mathtt{x}^{\iota}.\ell\mathtt{if\ x}\ \underline{1}\ (\mathtt{Prod}\,\underline{2}\,(\boldsymbol{F}(\mathbf{p}\,\mathtt{x})))
\end{aligned}$$

respectively define the addition, the multiplication and the exponentiation of base 2. Hence, the above function $< \_, \_ >$ is defined by the program

$$\lceil -, - \rceil \equiv \lambda \mathtt{x}^{\iota}\mathtt{y}^{\iota}.\mathbf{p}\big(\mathtt{Prod}\,(\mathtt{Exp2\,x})(\mathbf{s}(\mathtt{Prod}\,\underline{2}\,\mathtt{y}))\big)\ .$$

Clearly, our pairing-program is total and correct i.e. $\lceil \underline{\mathtt{n}}, \underline{\mathtt{m}} \rceil \Downarrow \underline{< \mathtt{n}, \mathtt{m} >}$ for all numerals $\underline{\mathtt{n}}, \underline{\mathtt{m}}$. So, we will write $\mathtt{M} \Downarrow \lceil \underline{\mathtt{n}}, \underline{\mathtt{m}} \rceil$, for some $\mathtt{M}$, has a shorthand for $\exists \underline{\mathtt{k}}$, $\mathtt{M} \Downarrow \underline{\mathtt{k}}$ and $\lceil \underline{\mathtt{n}}, \underline{\mathtt{m}} \rceil \Downarrow \underline{\mathtt{k}}$.

The definition of the terms $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ corresponding to projection (i.e. which are such that $\boldsymbol{\pi}_1 \lceil \underline{\mathtt{n}}, \underline{\mathtt{m}} \rceil \Downarrow \underline{\mathtt{n}}$ and $\boldsymbol{\pi}_2 \lceil \underline{\mathtt{n}}, \underline{\mathtt{m}} \rceil \Downarrow \underline{\mathtt{m}}$) turns out to be an easy exercise and it is left to the reader. Again, we write $\mathtt{M} \Downarrow \boldsymbol{\pi}_i\ \underline{\mathtt{n}}$ ($i \in [1,2]$) to mean $\exists \underline{\mathtt{k}}$, $\mathtt{M} \Downarrow \underline{\mathtt{k}}$ and $\boldsymbol{\pi}_i\ \underline{\mathtt{n}} \Downarrow \underline{\mathtt{k}}$.

2.2. *Operational equivalence*

There is a notion of program equivalence which programmers understand well: two program fragments are equivalent if they can always be *interchanged* without affecting the visible or observable outcome of the computation.

The set of σ-*context* Ctx_σ is defined as:

$$\mathtt{C}[\sigma] ::= \ [\sigma] \mid \varkappa^{\tau} \mid \mathbf{0} \mid \mathbf{s} \mid \mathbf{p} \mid \ell\mathtt{if\ C}[\sigma]\ \mathtt{C}[\sigma]\ \mathtt{C}[\sigma] \mid (\mathtt{C}[\sigma]\mathtt{C}[\sigma]) \mid (\lambda \mathtt{x}^{\sigma}.\mathtt{C}[\sigma]) \mid \mu_{\boldsymbol{F}}.\mathtt{C}[\sigma].$$

$\mathtt{C}[\mathtt{N}^{\sigma}]$ 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.

Definitiona 4 (standard operational equivalence). Let $\mathtt{M}^{\sigma}, \mathtt{N}^{\sigma}$ be terms.

1. $\mathtt{M} \lessapprox_{\sigma} \mathtt{N}$ whenever, for all $\mathtt{C}[\sigma]$ s.t. $\mathtt{C}[\mathtt{M}], \mathtt{C}[\mathtt{N}] \in \mathrm{P}$, 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 $\mathtt{N} \lessapprox_{\sigma} \mathtt{M}$.

The above definition is standard, however, we introduce an alternative equivalence definition that will allow to study the formal relationship between the two binders we introduced.

Definitiona 5 (fix-point operational equivalence). Let $\mathtt{M}^{\sigma}, \mathtt{N}^{\sigma}$ be such that $\mathrm{SFV}(\mathtt{M}), \mathrm{SFV}(\mathtt{N}) \subseteq \{\boldsymbol{F}_1^{\sigma_1}, \ldots, \boldsymbol{F}_n^{\sigma_n}\}$.

1. $\mathtt{M} \lesssim_{\sigma} \mathtt{N}$ whenever, for all $\mathtt{P}_1^{\sigma_1}, \ldots \mathtt{P}_{\mathtt{n}}^{\sigma_{\mathtt{n}}}$, for all $\mathtt{C}[\sigma]$ s.t. $\mathtt{C}[\mathtt{M}[\mathtt{P}_1/\boldsymbol{F}_1, \ldots, \mathtt{P}_{\mathtt{n}}/\boldsymbol{F}_{\mathtt{n}}]], \mathtt{C}[\mathtt{N}[\mathtt{P}_1/\boldsymbol{F}_1, \ldots, \mathtt{P}_{\mathtt{n}}/\boldsymbol{F}_{\mathtt{n}}]] \in \mathrm{P}$ if $\mathtt{C}[\mathtt{M}[\mathtt{P}_1/\boldsymbol{F}_1, \ldots, \mathtt{P}_{\mathtt{n}}/\boldsymbol{F}_{\mathtt{n}}]] \Downarrow \underline{\mathtt{n}}$ then $\mathtt{C}[\mathtt{N}[\mathtt{P}_1/\boldsymbol{F}_1, \ldots, \mathtt{P}_{\mathtt{n}}/\boldsymbol{F}_{\mathtt{n}}]] \Downarrow \underline{\mathtt{n}}$.
2. $\mathtt{M} \sim_{\sigma} \mathtt{N}$ if and only if $\mathtt{M} \lesssim_{\sigma} \mathtt{N}$ and $\mathtt{N} \lesssim_{\sigma} \mathtt{M}$.

It is easy to verify that $\lesssim_{\sigma}$ is a preorder and $\sim_{\sigma}$ is an equivalence. Note that the comparison between the fix-point operational equivalence and the standard one is not immediate, since there is no obvious way to implement substitution of a stable variable with an arbitrary term. This issue will be discussed in detail in Section 6.

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

2.3. *Coherence spaces*

We are interested in the least full subcategory $\mathcal{L}$ of coherence spaces, including the flat domain of natural numbers and the coherence spaces of linear functions between domains in the category itself. Coherence spaces are a simple framework for Berry's stable functions (Berry 1978), developed by Girard (1987). This section does not aim to provide an exhaustive presentation of these arguments, it just aims to make the paper self-contained. More details can be found in Girard (1986, 1987) and Girard *et al.* (1989).

A *coherence space* X is a pair $\langle |X|, \frown_X \rangle$, where $|X|$ is a set of *tokens* called the *web* of X and $\frown_X$ is a reflexive and symmetric relation between tokens of $|X|$ called the *coherence relation* on X. The *strict incoherence* $\smile_X$ is the complementary relation of $\frown_X$; the *incoherence* $\asymp_X$ is the union of relations $\smile_X$ and $=$; the *strict coherence* $\frown_X$ is the complementary relation of $\asymp_X$. A *clique* x of X is a subset of $|X|$ containing pairwise coherent tokens. The set of cliques of X is denoted $\mathcal{C}l(X)$, while the set of finite cliques is denoted $\mathcal{C}l_{fin}(X)$. Two cliques $x, y \in \mathcal{C}l(X)$ are *compatible* when $x \cup y \in \mathcal{C}l(X)$.

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

— $\varnothing \in \mathcal{C}l(X)$ is the bottom element 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 $\bigcup D \in \mathcal{C}l(X)$,
— the set of compact elements is $\mathcal{C}l_{fin}(X)$.

Definitiona 6. Let X and Y be coherence spaces.

1. The *linear implication* $X \multimap Y$ is the coherence space having $|X \multimap Y| = |X| \times |Y|$ as web, while $(a, b)\ \Pi_{X \multimap Y}\ (a', b')$ iff $a\ \Xi_X\ a'$ implies $b\ \Pi_Y\ b'$.
2. A *linear morphism* between X and Y is an element of $\mathcal{C}l(X \multimap Y)$.
3. The *tensor product* $X \otimes Y$ is the coherence space having $|X \otimes Y| = |X| \times |Y|$ as web, while $(a, b)\ \Xi_{X \otimes Y}\ (a', b')$ if $a\ \Xi_X\ a'$ and $b\ \Xi_Y\ b'$.

We denote with **LinCoh** the category whose objects are coherence spaces and whose morphisms are linear morphisms between coherence spaces. $(\mathbf{LinCoh}, \otimes, \mathbf{1}, \multimap)$ is a symmetric monoidal closed category, where $\mathbf{1}$ is the coherence space having a singleton set as web.

Definitiona 7. Let X, Y be two coherence spaces. A function $f : \mathcal{C}l(X) \to \mathcal{C}l(Y)$ is *continuous* whenever, it is monotone and it commutes with the union of directed sets of cliques. A continuous function $f : \mathcal{C}l(X) \to \mathcal{C}l(Y)$ is *stable* when $\forall x, x' \in \mathcal{C}l(X)$, x and x' compatible implies $f(x \cap x') = f(x) \cap f(x')$. A function $f : \mathcal{C}l(X) \to \mathcal{C}l(Y)$ is *linear* when it is stable (so, continuous), $f(\varnothing) = \varnothing$ and for all $x, x' \in \mathcal{C}l(X)$, if x and x' are compatible then $f(x \cup x') = f(x) \cup f(x')$.

Linear morphisms are in bijection with linear functions between cliques.

The *trace* of a linear function $f : \mathcal{C}l(X) \to \mathcal{C}l(Y)$ is $\mathcal{T}r(f) = \{(a, b) \mid a \in |X|, b \in f(\{a\})\}$. Given $t \in \mathcal{C}l(X \multimap Y)$ and $x \in \mathcal{C}l(X)$, let us define the map $\mathcal{F}(t) : \mathcal{C}l(X) \to \mathcal{C}l(Y)$ as

$$\mathcal{F}(t)(x) = \{b \in |Y| \mid \exists a \in x, (a, b) \in t\}. \tag{1}$$

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

Proposition 1.

1. If $f : \mathcal{C}l(X) \to \mathcal{C}l(Y)$ is a linear function then $\mathcal{T}r(f) \in \mathcal{C}l(X \multimap Y)$.
2. 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.
3. $\mathcal{T}r(\mathcal{F}(t)) = t$, for all $t : \mathcal{C}l(X \multimap Y)$.
4. $\mathcal{F}(\mathcal{T}r(f)) = f$, for all $f : \mathcal{C}l(X) \to \mathcal{C}l(Y)$.

The basis of our model is the countable flat domain. Let $\mathbf{N}$ denotes the space of natural numbers, namely $(|\mathbf{N}|, \frown_{\mathbf{N}})$ such that $|\mathbf{N}| := \mathbb{N}$ (the latter is the set of natural numbers) and $m \frown_{\mathbf{N}} n$ if and only if $m = n$, for all $m, n \in |\mathbf{N}|$.

Definitiona 8 (linear model). The *linear model* $\mathcal{L}$ is the type structure generated by the coherence space $\mathbf{N}$ and the arrow $\multimap$.

To provide the interpretation of the recursion in core-S$\mathscr{P}$CF, we discuss the fix-point operator on coherence domain. More precisely, if X is a coherence space then we look for $\mathtt{fix}_X : (\mathcal{C}l(X) \to \mathcal{C}l(X)) \to \mathcal{C}l(X)$ such that $\mathtt{fix}_X(f) = f(\mathtt{fix}_X(f))$ for all stable functions $f : \mathcal{C}l(X) \to \mathcal{C}l(X)$. Given any coherence spaces X, Y, we remind that the set of stable functions between $\mathcal{C}l(X)$ and $\mathcal{C}l(Y)$ together with stable ordering[†] form a Scott domain. Thus we define a family of maps[‡] $\mathtt{fix}^n_X : (\mathcal{C}l(X) \to \mathcal{C}l(X)) \to \mathcal{C}l(X)$ by induction on $n \in \mathbb{N}$:

$$\mathtt{fix}^0_X = \lambda\!\!\lambda f : \mathcal{C}l(X) \to \mathcal{C}l(X).\varnothing \qquad \mathtt{fix}^{n+1}_X = \lambda\!\!\lambda f : \mathcal{C}l(X) \to \mathcal{C}l(X).f(\mathtt{fix}^n_X(f)).$$

By using Knaster–Tarski fix point theorem, we can define $\mathtt{fix}_X = \bigvee_n \mathtt{fix}^n_X$. In the sequel, we will omit the subscript X on $\mathtt{fix}_X$ when it is clear from the context or uninteresting.

Proposition 2. Let X be a coherence space and let $f : \mathcal{C}l(X) \to \mathcal{C}l(X)$, $g : \mathcal{C}l(X) \to \mathcal{C}l(\mathbf{N})$ be two stable functions and let $p \in \mathbb{N}$. If $g(\mathtt{fix}(f)) = \{p\}$, then there is $k \in \mathbb{N}$ such that $g(\mathtt{fix}^k(f)) = \{p\}$.

We stress the fact that $\mathtt{fix}$ is a stable function which is not linear. We use fix in a restricted way, it is applied only to stable endo-functions in $\mathcal{L}$, and it gives back traces of linear functions. The Kleisli category on the finite-clique comonad $\mathbf{LinCoh}_!$ is equivalent to the category of coherence spaces and stable functions. This category is Cartesian closed and it admits a least fix point operator, which corresponds to the above defined $\mathtt{fix}$.

2.4. *Linear interpretation*

We define a standard interpretation (Plotkin 1977) such that $[\![\iota]\!] = \mathbf{N}$ and $[\![\sigma \multimap \tau]\!] = [\![\sigma]\!] \multimap [\![\tau]\!]$. An *environment* $\rho \in \mathrm{Env}$ is a total function mapping a variable $\mathtt{x}^\sigma$ in a token $a \in |[\![\sigma]\!]|$ and a stable variable $\mathtt{F}^\sigma$ in a finite clique $x \in \mathcal{C}l_{fin}([\![\sigma]\!])$. We remark that

[†] Let $f, g : \mathcal{C}l(X) \to \mathcal{C}l(Y)$ two stable functions. The *stable ordering* $f \sqsubseteq g$ is defined as $\forall x, y \in \mathcal{C}l(X)$, $x \subseteq y$ implies $f(x) = f(y) \cap g(x)$.

[‡] Let A and B be two sets, let x be an element of A and let $E(x)$ be an expression that may contain x and corresponding to an element of B. $\lambda\!\!\lambda x : A.E(x)$ is the function from A to B mapping any element $x \in A$ into the element $E(x) \in B$, that is the function $f : A \to B$ such that $f(x) = E(x)$. In p. 49 of Gunter (1992), the notation $x \mapsto E(x)$ is used with the same meaning of $\lambda\!\!\lambda x.E(x)$.

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

Table 2. *Linear interpretation map.*

$$[\![\mathbf{0}^{\iota}]\!]\rho = \{0\} \qquad [\![\mathbf{s}^{\iota\multimap\iota}]\!]\rho = \{(n,n+1) \mid n \in \mathbb{N}\} \qquad [\![\mathbf{p}^{\iota\multimap\iota}]\!]\rho = \{(n+1,n) \mid n \in \mathbb{N}\}$$

$$[\![\mathtt{x}^{\sigma}]\!]\rho = \{\rho(\mathtt{x}^{\sigma})\} \qquad [\![\mathtt{F}^{\sigma}]\!]\rho = \rho(\mathtt{F}^{\sigma}) \qquad [\![\mathtt{M}^{\sigma\multimap\tau}\mathtt{N}^{\sigma}]\!]\rho = \mathcal{F}([\![\mathtt{M}]\!]\rho)[\![\mathtt{N}]\!]\rho$$

$$[\![(\ell\mathtt{if}\ \mathtt{M}^{\iota}\ \mathtt{N}^{\iota}\ \mathtt{L}^{\iota})^{\iota}]\!]\rho = \{n \in \mathbb{N} \mid [\![\mathtt{M}^{\iota}]\!]\rho = \{0\}; [\![\mathtt{N}^{\iota}]\!]\rho = \{n\}\} \cup \{n \in \mathbb{N} \mid [\![\mathtt{M}^{\iota}]\!]\rho = \{m+1\}; [\![\mathtt{L}^{\iota}]\!]\rho = \{n\}, m \in \mathbb{N}\}$$

$$[\![\lambda\mathtt{x}^{\sigma}.\mathtt{M}^{\tau}]\!]\rho = \{(a_0,b) \ \in |[\![\sigma \multimap \tau]\!]| \mid b \in [\![\mathtt{M}]\!]\rho[\mathtt{x}^{\sigma} := a_0]\}$$

$$[\![(\mu_{\mathtt{F}^{\sigma}}.\mathtt{M}^{\sigma})^{\sigma}]\!]\rho = \mathtt{fix}(\mathbb{\lambda} x \in \mathcal{C}l([\![\sigma]\!]).\bigcup_{x' \subseteq_{fin} x} [\![\mathtt{M}]\!]\rho[\mathtt{F} := x'])$$

the interpretation of x^{ι} cannot be associated to an empty cliques, so it is subjected to restrictions (see cases 4 and 5 of Lemma 2). The set of environments is denoted by Env. Let $\vec{a}$ be a sequence of tokens of a coherence space, let $\vec{\mathtt{x}}$ be a sequence of non-stable variables of the same length of $\vec{a}$; $\rho[\vec{\mathtt{x}} := \vec{a}]$ is the environment such that $\rho[\vec{\mathtt{x}} := \vec{a}](\mathtt{x}') = a_i$ in case $\mathtt{x}'$ is the ith element of $\vec{\mathtt{x}}$, otherwise $\rho[\vec{\mathtt{x}} := \vec{a}](\mathtt{x}') = \rho(\mathtt{x}')$. If $\vec{x}$ is a sequence of finite cliques and $\vec{\mathtt{F}}$ is a sequence of stable variables of the same length then $\rho[\vec{\mathtt{F}} := \vec{x}]$ is defined likewise. We will write $x \subseteq_{fin} y$ when $x \subseteq y$ with x finite.

Definitiona 9. Let $\mathtt{M}^{\sigma}, \mathtt{N}^{\sigma}$ and $\rho \in$ Env. The *linear interpretation* $[\![\mathtt{M}^{\sigma}]\!] : \text{Env} \to \mathcal{C}l([\![\sigma]\!])$ is defined in Table 2 using $\mathcal{F}$ as defined in *Equation 1* and `fix` which is the least fix point operator.

Looking at the interpretation, we justify the introduction of 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 fix-points will be always an element of L. Notice that, syntactically, a stable variable will be used without linear constraints. We do not permit to λ-abstract stable variables, we use them only in order to obtain fix-points. Permitting the λ-abstraction of stable variable would produce terms defining non-linear functions, which is against our wishes.

Two terms $\mathtt{M}^{\sigma}$ and $\mathtt{N}^{\sigma}$ are *denotationally equivalent* if and only if $[\![\mathtt{M}^{\sigma}]\!]\rho = [\![\mathtt{N}^{\sigma}]\!]\rho$ for every ρ. The interpretation of closed terms is invariant with respect to environments, thus in such cases the environment can be omitted. Next lemmas are standard. The basic properties of a λ-model are recalled in Lemma 2.

Lemma 2. Let $\mathtt{M}^{\sigma}, \mathtt{N}^{\tau}$ and $\rho, \rho' \in$ Env.

1. If $\rho(\mathtt{F}) \subseteq \rho'(\mathtt{F})$ for each $\mathtt{F} \in \text{SFV}(\mathtt{M})$, then $[\![\mathtt{M}]\!]\rho \subseteq [\![\mathtt{M}]\!]\rho'$.
2. If $\mathtt{M}^{\sigma}[\mathtt{N}/\mathtt{F}^{\tau}] \in$ S⁄PCF then $[\![\mathtt{M}^{\sigma}[\mathtt{N}/\mathtt{F}^{\tau}]]\!]\rho = \bigcup_{x \subseteq_{fin} [\![\mathtt{N}]\!]_{\rho}} [\![\mathtt{M}]\!]\rho[\mathtt{F}^{\tau} := x]$.
3. If $\mathtt{M}^{\sigma}[\mathtt{N}/\mathtt{x}^{\tau}] \in$ S⁄PCF with $\tau \neq \iota$ then $[\![\mathtt{M}^{\sigma}[\mathtt{N}/\mathtt{x}^{\tau}]]\!]\rho = \bigcup_{a \in [\![\mathtt{N}]\!]_{\rho}} [\![\mathtt{M}]\!]\rho[\mathtt{x} := a]$.
4. If $\mathtt{M}^{\sigma}[\underline{\mathtt{n}}/\mathtt{x}^{\iota}] \in$ S⁄PCF then $[\![\mathtt{M}^{\sigma}[\underline{\mathtt{n}}/\mathtt{x}^{\tau}]]\!]\rho = [\![\mathtt{M}]\!]\rho[\mathtt{x} := n]$.
5. $[\![(\lambda\mathtt{x}^{\iota}.\mathtt{M})\underline{\mathtt{n}}]\!] = [\![\mathtt{M}[\underline{\mathtt{n}}/\mathtt{x}^{\iota}]]\!]$.
6. $[\![(\lambda\mathtt{f}^{\sigma}.\mathtt{M})\mathtt{N}]\!] = [\![\mathtt{M}[\mathtt{N}/\mathtt{f}^{\sigma}]]\!]$.
7. $[\![\ell\mathtt{if}\ \underline{0}\ \mathtt{L}\ \mathtt{R}\,]\!] = [\![\mathtt{L}]\!]$ and $[\![\ell\mathtt{if}\ \underline{\mathtt{n}+1}\ \mathtt{L}\ \mathtt{R}\,]\!] = [\![\mathtt{R}]\!]$.

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

8. $[\![\mu_{\mathtt{F}^\sigma}.\mathtt{M}]\!] = [\![\mathtt{M}[\mu_{\mathtt{F}^\sigma}.\mathtt{M}/\mathtt{F}^\sigma]]\!]$.
9. If $\sigma = \tau$, $\mathtt{C}[^\sigma] \in \mathrm{Ctx}_\sigma$, $\mathtt{C}[\mathtt{M}], \mathtt{C}[\mathtt{N}] \in \mathrm{P}$ and $[\![\mathtt{M}]\!]\rho = [\![\mathtt{N}]\!]\rho$ then $[\![\mathtt{C}[\mathtt{M}]]\!] = [\![\mathtt{C}[\mathtt{N}]]\!]$.

Proof. (1)–(4) follow by induction on the structure of $\mathtt{M}^\sigma$. The proofs of (5)–(8) follow by definition of interpretation and by points (2), (3) and (4). The proof of (9) follows by induction on the structure of $\mathtt{C}[^\sigma]$. □

As usual, an approximation theorem for finite fixpoint semantics holds.

Lemma 3.

1. $[\![\mu^k_{\mathtt{F}}.\mathtt{M}]\!]\rho = \mathtt{fix}^k(\lambda x \in \mathcal{C}l([\![\sigma]\!]).\bigcup_{x' \subseteq_{fin} x} [\![\mathtt{M}]\!]\rho[\mathtt{F} := x'])$.
2. $[\![\mu_{\mathtt{F}}.\mathtt{M}^\sigma]\!]\rho = \bigcup_{n\in\mathbb{N}} [\![\mu^n_{\mathtt{F}}.\mathtt{M}^\sigma]\!]\rho$, for all $\sigma \in \mathbb{T}$.

Proof. (1) Obvious. (2) Since $[\![\mu^{n+1}_{\mathtt{F}}.\mathtt{M}^\sigma]\!]\rho = [\![\mathtt{M}^\sigma]\!]\rho[\mathtt{F} := [\![\mu^n_{\mathtt{F}}.\mathtt{M}^\sigma]\!]\rho]$, the proof is easy. □

Theorem 1. If $\mathtt{M} \Downarrow \underline{\mathtt{n}}$ then $[\![\mathtt{M}]\!] = [\![\underline{\mathtt{n}}]\!]$.

Proof. By induction on the derivation of $\mathtt{M} \Downarrow \underline{\mathtt{n}}$. Lemma 2 is crucial to deal with the various inductive steps. □

2.5. *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 $\mathtt{M}$ and numeral $\underline{\mathtt{n}}$. We straightforward adapt a proof of Plotkin (1977) for Scott-domains, based on a computability argument in Tait style.

Definitiona 10. The 'computability predicate' is defined by the following cases.

— Case $\mathrm{FV}(\mathtt{M}^\sigma) = \varnothing$.
  - Subcase $\sigma = \iota$. $\mathrm{Comp}(\mathtt{M}^\iota)$ if and only if $\forall \underline{\mathtt{n}}$, $[\![\mathtt{M}]\!] = [\![\underline{\mathtt{n}}]\!]$ implies $\mathtt{M} \Downarrow \underline{\mathtt{n}}$.
  - Subcase $\sigma = \mu \multimap \tau$. $\mathrm{Comp}(\mathtt{M}^{\mu\multimap\tau})$ if and only if $\mathrm{Comp}(\mathtt{M}^{\mu\multimap\tau}\mathtt{N}^\mu)$ for each closed $\mathtt{N}^\mu$ such that $\mathrm{Comp}(\mathtt{N}^\mu)$.

— Case $\mathrm{FV}(\mathtt{M}^\sigma) = \{\varkappa_1^{\tau_1}, \ldots, \varkappa_n^{\tau_n}\}$, for some $n \geqslant 1$.
  $\mathrm{Comp}(\mathtt{M}^\sigma)$ if and only if $\mathrm{Comp}(\mathtt{M}[\mathtt{N}_1/\varkappa_1, \ldots, \mathtt{N}_n/\varkappa_n])$ for each closed $\mathtt{N}_i^{\tau_i}$ s.t. $\mathrm{Comp}(\mathtt{N}_i^{\tau_i})$.

Lemma 4 states a standard equivalent formulation of computability predicate.

Lemma 4. Let $\mathtt{M}^{\tau_1\multimap\cdots\multimap\tau_m\multimap\iota}$ and $\mathrm{FV}(\mathtt{M}) = \{\varkappa_1^{\mu_1}, \ldots, \varkappa_n^{\mu_n}\}$ $(n, m \in \mathbb{N})$. $\mathrm{Comp}(\mathtt{M})$ if and only if $[\![\mathtt{M}[\mathtt{N}_1/\varkappa_1, \ldots, \mathtt{N}_n/\varkappa_n]\mathtt{P}_1 \ldots \mathtt{P}_m]\!] = [\![\underline{\mathtt{n}}]\!]$ 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 $\mathrm{Comp}(\mathtt{N}_i)$ and $\mathrm{Comp}(\mathtt{P}_j)$ where $i \leqslant n, j \leqslant m$.

We remark that we consider substitution to all kinds of free variables.

Lemma 5. If $\mathtt{M}^\sigma$ is a term of core-S$\mathscr{O}$PCF then $\mathrm{Comp}(\mathtt{M}^\sigma)$.

Proof. By induction on the shape of $\mathtt{M}$. We detail the most interesting cases.

— $\mathtt{M} = \varkappa$. 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 $1 \leqslant i \leqslant m$ be closed terms such that $\mathrm{Comp}(\mathtt{P}^\sigma)$ and $\mathrm{Comp}(\mathtt{N}_i^{\tau_i})$. By definition, $\mathrm{Comp}(\mathtt{P}^\sigma)$ implies that, if $[\![\mathtt{P}\mathtt{N}_1 \cdots \mathtt{N}_m]\!] = [\![\underline{\mathtt{n}}]\!]$ then $\mathtt{P}\mathtt{N}_1 \cdots \mathtt{N}_m \Downarrow \underline{\mathtt{n}}$, by definition of the computability predicate.

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

— $\mathtt{M} = \mathtt{NP}$. 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 definition of the computability predicate.

— $\mathtt{M} = \lambda\mathtt{x}.\mathtt{Q}$. Assume $\mathtt{x}^{\mu}$ and $\mathtt{Q}^{\tau}$ for types μ and τ. Let $\mathrm{FV}(\mathtt{M}) = \{\varkappa_1^{\mu_1},\dots,\varkappa_k^{\mu_k}\}$ for $k \geqslant 0$ and $\tau = \tau_1 \multimap \cdots \multimap \tau_h \multimap \iota$, where $h \geqslant 0$. Let $\mathtt{N}_1^{\mu_1},\dots,\mathtt{N}_k^{\mu_k},\mathtt{P}_0^{\mu},\mathtt{P}_1^{\tau_1},\dots,\mathtt{P}_h^{\tau_h}$ be closed terms such that $\mathrm{Comp}(\mathtt{N}_i)$ and $\mathrm{Comp}(\mathtt{P}_j)$ for $1 \leqslant i \leqslant k$ and $0 \leqslant j \leqslant h$ respectively. Thus $\mathrm{Comp}(\mathtt{Q}^{\tau}[\mathtt{P}_0/\mathtt{x}][\mathtt{N}_1/\varkappa_1,\dots,\mathtt{N}_k/\varkappa_k]\mathtt{P}_1\dots\mathtt{P}_h)$, since $\mathrm{Comp}(\mathtt{Q}^{\tau})$ holds by induction hypothesis.
Consider the case $\mu \neq \iota$ and $[\![(\lambda\mathtt{x}^{\mu}.\mathtt{Q}^{\tau})[\mathtt{N}_1/\varkappa_1,\dots,\mathtt{N}_k/\varkappa_k]\mathtt{P}_0\dots\mathtt{P}_h]\!] = [\![\underline{\mathtt{n}}]\!]$. Clearly, $[\![\mathtt{Q}^{\tau}[\mathtt{P}_0/\mathtt{x}][\mathtt{N}_1/\varkappa_1,\dots,\mathtt{N}_k/\varkappa_k]\mathtt{P}_1\dots\mathtt{P}_h]\!] = [\![\underline{\mathtt{n}}]\!]$ by Lemma 2 point (6). Thus $\mathtt{Q}^{\tau}[\mathtt{P}_0/\mathtt{x}][\mathtt{N}_1/\varkappa_1,\dots,\mathtt{N}_k/\varkappa_k]\mathtt{P}_1\dots\mathtt{P}_h \Downarrow \underline{\mathtt{n}}$ by induction hypothesis. Therefore,

$$(\lambda\mathtt{x}^{\mu}.\mathtt{Q}^{\tau})[\mathtt{N}_1/\varkappa_1,\dots,\mathtt{N}_k/\varkappa_k]\mathtt{P}_0\dots\mathtt{P}_h \Downarrow \underline{\mathtt{n}}$$

by the evaluation rule $(\lambda^{\multimap})$.
Now, suppose $\mu = \iota$ and $[\![(\lambda\mathtt{x}^{\mu}.\mathtt{Q}^{\tau})[\mathtt{N}_1/\varkappa_1,\dots,\mathtt{N}_k/\varkappa_k]\mathtt{P}_0\dots\mathtt{P}_h]\!] = [\![\underline{\mathtt{n}}]\!]$. Since linear functions are strict, we must have $[\![\mathtt{P}_0]\!] = [\![\underline{\mathtt{m}}]\!]$ for some $\underline{\mathtt{m}}$. But $\mathrm{Comp}(\mathtt{P}_0^{\mu})$ by induction and $[\![\mathtt{P}_0]\!] = [\![\underline{\mathtt{m}}]\!]$ imply $\mathtt{P}_0 \Downarrow \underline{\mathtt{m}}$. Hence $[\![\mathtt{Q}^{\tau}[\underline{\mathtt{m}}/\mathtt{x}][\mathtt{N}_1/\varkappa_1,\dots,\mathtt{N}_k/\varkappa_k]\mathtt{P}_1\dots\mathtt{P}_h]\!] = [\![\underline{\mathtt{n}}]\!]$ by Lemma 2 point (5). $\mathtt{Q}^{\tau}[\underline{\mathtt{m}}/\mathtt{x}][\mathtt{N}_1/\varkappa_1,\dots,\mathtt{N}_k/\varkappa_k]\mathtt{P}_1\dots\mathtt{P}_h \Downarrow \underline{\mathtt{n}}$ by induction hypothesis. The proof follows by applying the evaluation rule (λ^{ι}).

— $\mathtt{M} = \ell\mathtt{if}\ \mathtt{M}_1\ \mathtt{M}_2\ \mathtt{M}_3$. Assume $\mathrm{FV}(\mathtt{M}) = \{\varkappa_1^{\sigma_1},\dots,\varkappa_n^{\sigma_n}\}$ and let $\mathtt{P}_1^{\sigma_1},\dots\mathtt{P}_n^{\sigma_n}$ be closed terms such that $\mathrm{Comp}(\mathtt{P}_i)$ for all i. Suppose that $[\![\mathtt{M}[\mathtt{P}_1/\varkappa_1,\dots,\mathtt{P}_n/\varkappa_n]]\!] = [\![\underline{\mathtt{m}}]\!]$. Then, by interpretation, one of the following cases holds:

  – $[\![\mathtt{M}_1[\mathtt{P}_1/\varkappa_1,\dots,\mathtt{P}_n/\varkappa_n]]\!] = [\![\underline{0}]\!]$ and $[\![\mathtt{M}_2[\mathtt{P}_1/\varkappa_1,\dots,\mathtt{P}_n/\varkappa_n]]\!] = [\![\underline{\mathtt{m}}]\!]$,
  – $[\![\mathtt{M}_1[\mathtt{P}_1/\varkappa_1,\dots,\mathtt{P}_n/\varkappa_n]]\!] = [\![\underline{\mathtt{k}+1}]\!]$ and $[\![\mathtt{M}_3[\mathtt{P}_1/\varkappa_1,\dots,\mathtt{P}_n/\varkappa_n]]\!] = [\![\underline{\mathtt{m}}]\!]$.

  In the first case, we have $\mathtt{M}_1[\mathtt{P}_1/\varkappa_1,\dots,\mathtt{P}_n/\varkappa_n] \Downarrow \underline{0}$ and $\mathtt{M}_2[\mathtt{P}_1/\varkappa_1,\dots,\mathtt{P}_n/\varkappa_n] \Downarrow \underline{\mathtt{m}}$ by inductive hypothesis; thus we conclude by the evaluation rule (ifl). For the other case, we conclude again by using inductive hypothesis and the evaluation rule (ifr).

— $\mathtt{M} = \mu_{\boldsymbol{f}}.\mathtt{N}$, so the typing constraints imply $\mathtt{M}^{\sigma},\mathtt{N}^{\sigma}$ for the same type $\sigma \neq \iota$. Let $\mathrm{FV}(\mathtt{M}) = \{\varkappa_1^{\mu_1},\dots,\varkappa_k^{\mu_k}\}$ for $k \geqslant 0$ and $\sigma = \tau_1 \multimap \cdots \multimap \tau_h \multimap \iota$, where $h \geqslant 0$. Assume $h \geqslant 1$ and, $\mathtt{N}_1^{\mu_1},\dots,\mathtt{N}_k^{\mu_k}$, $\mathtt{P}_1^{\tau_1},\dots,\mathtt{P}_h^{\tau_h}$ be closed terms such that $\mathrm{Comp}(\mathtt{N}_i)$ and $\mathrm{Comp}(\mathtt{P}_j)$ for $1 \leqslant i \leqslant k$ and $1 \leqslant j \leqslant h$ respectively. Let $[\![(\mu_{\boldsymbol{f}}.\mathtt{N}^{\sigma}[\mathtt{N}_1/\varkappa_1,\dots,\mathtt{N}_k/\varkappa_k])\mathtt{P}_1\dots\mathtt{P}_h]\!] = [\![\underline{\mathtt{n}}]\!]$. By Proposition 2, there is k such that

$$\mathcal{F}\left(\mathtt{fix}^{k}\left(\mathbb{\lambda}x \in \mathcal{C}l([\![\sigma]\!]).\ \bigcup_{x' \subseteq_{fin} x} [\![\mathtt{M}[\vec{\mathtt{N}}/\vec{\varkappa}]]\!]\rho[\boldsymbol{f} := x']\right)\right)([\![\mathtt{P}_1]\!]\rho)\dots([\![\mathtt{P}_h]\!]\rho) = [\![\underline{\mathtt{n}}]\!].$$

Hence $[\![(\mu_{\boldsymbol{f}}.\mathtt{N}^{\sigma}[\mathtt{N}_1/\varkappa_1,\dots,\mathtt{N}_k/\varkappa_k])\mathtt{P}_1\dots\mathtt{P}_h]\!] = [\![(\mu^{k}\boldsymbol{f}.\mathtt{N}^{\sigma}[\mathtt{N}_1/\varkappa_1,\dots,\mathtt{N}_k/\varkappa_k])\mathtt{P}_1\dots\mathtt{P}_h]\!]$ for some $k \in \mathbb{N}$, by Lemma 3. Thus, $\mu^{k}\boldsymbol{f}.\mathtt{N}^{\sigma}[\mathtt{Q}'/\mathtt{v}_1,\dots,\mathtt{Q}'/\mathtt{v}_m]\mathtt{P}_1\dots\mathtt{P}_h \Downarrow \underline{\mathtt{n}}$, by the previous points of this lemma. The proof follows by Lemma 1.

□

Corollary 1 (adequacy). For all $\mathtt{M} \in \mathrm{P}$, for all $\underline{\mathtt{n}}$, $[\![\mathtt{M}]\!]\rho = [\![\underline{\mathtt{n}}]\!]\rho$ if and only if $\mathtt{M} \Downarrow \underline{\mathtt{n}}$.

As usual the adequacy implies the correctness, i.e. the denotational equivalence implies the operational one. Note that correctness implies that our terms are strict in all arguments, for all orders.

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

Theorem 2 (correctness). Let $\mathtt{M}^{\sigma}, \mathtt{N}^{\sigma}$ be terms of core-S$\mathscr{\ell}$PCF. If $[\![\mathtt{M}]\!]_{\rho} = [\![\mathtt{N}]\!]_{\rho}$, for each $\rho \in \mathrm{Env}$, then $\mathtt{M} \approx^{\sigma} \mathtt{N}$.

Proof. Let $\mathtt{C}[\sigma]$ such that $\mathtt{C}[\mathtt{M}], \mathtt{C}[\mathtt{N}] \in \mathrm{P}$. If $\mathtt{C}[\mathtt{M}] \Downarrow \underline{\mathtt{n}}$ for some value $\underline{\mathtt{n}}$, then $[\![\mathtt{C}[\mathtt{M}]]\!] = [\![\underline{\mathtt{n}}]\!]$ by Theorem 1. Since $[\![\mathtt{C}[\mathtt{N}]]\!] = [\![\mathtt{C}[\mathtt{M}]]\!] = [\![\underline{\mathtt{n}}]\!]$ by Lemma 2, $\mathtt{C}[\mathtt{N}] \Downarrow \underline{\mathtt{n}}$ by adequacy. By definition of standard operational equivalence, the proof is done. □

We can reuse the same argument to prove the correctness for the fix-point operational equivalence (Definition 5).

Proposition 3. Let $\mathtt{M}^{\sigma}, \mathtt{N}^{\sigma} \in$ core-S$\mathscr{\ell}$PCF. If $[\![\mathtt{M}]\!]_{\rho} = [\![\mathtt{N}]\!]_{\rho}$, for each $\rho \in \mathrm{Env}$, then $\mathtt{M} \sim^{\sigma} \mathtt{N}$.

Proof. Suppose $\mathrm{SFV}(\mathtt{M}), \mathrm{SFV}(\mathtt{N}) \subseteq \{\mathtt{F}_1^{\tau_1}, \ldots, \mathtt{F}_{\mathtt{n}}^{\tau_{\mathtt{n}}}\}$. Let $\mathtt{P}_1^{\tau_1}, \ldots, \mathtt{P}_{\mathtt{n}}^{\tau_{\mathtt{n}}}$ be closed terms and let $\mathtt{C}[\sigma]$ be a context such that $\mathtt{C}[\mathtt{M}[\mathtt{P}_1/\mathtt{F}_1, \ldots, \mathtt{P}_{\mathtt{n}}/\mathtt{F}_{\mathtt{n}}]], \mathtt{C}[\mathtt{N}[\mathtt{P}_1/\mathtt{F}_1, \ldots, \mathtt{P}_{\mathtt{n}}/\mathtt{F}_{\mathtt{n}}]] \in \mathrm{P}$.
If $\mathtt{C}[\mathtt{M}[\mathtt{P}_1/\mathtt{F}_1, \ldots, \mathtt{P}_{\mathtt{n}}/\mathtt{F}_{\mathtt{n}}]] \Downarrow \underline{\mathtt{n}}$ then $[\![\mathtt{C}[\mathtt{M}[\mathtt{P}_1/\mathtt{F}_1, \ldots, \mathtt{P}_{\mathtt{n}}/\mathtt{F}_{\mathtt{n}}]]]\!] = [\![\underline{\mathtt{n}}]\!]$ by Theorem 1; but $[\![\mathtt{C}[\mathtt{N}[\mathtt{P}_1/\mathtt{F}_1, \ldots, \mathtt{P}_{\mathtt{n}}/\mathtt{F}_{\mathtt{n}}]]]\!] = [\![\underline{\mathtt{n}}]\!]$ by hypothesis, thus $\mathtt{C}[\mathtt{N}[\mathtt{P}_1/\mathtt{F}_1, \ldots, \mathtt{P}_{\mathtt{n}}/\mathtt{F}_{\mathtt{n}}]] \Downarrow \underline{\mathtt{n}}$ by adequacy. By definition of fix-point operational equivalence the proof is done. □

2.6. *Token definability and stable closed full abstraction*

Core-S$\mathscr{\ell}$PCF is sufficient to define all tokens (belonging to coherence spaces) of $\mathcal{L}$. First, we introduce some abbreviations that are useful in order to simplify the rest of the paper. We use $(\mathtt{M\ and\ N})$ to abbreviate $(\ell\mathtt{if\ M}\ (\ell\mathtt{if\ N}\ \underline{0}\ \underline{1})\ \underline{1})$. The equivalence among numerals, denoted $\doteq$ used in infix notation, is encoded as

$$\mu_{\mathtt{F}^{\iota\multimap\iota\multimap\iota}}.\lambda\mathtt{x}^{\iota}.\lambda\mathtt{y}^{\iota}.\ell\mathtt{if\ x}\ (\ell\mathtt{if\ y}\ \underline{0}\ \underline{1})\ \big(\ell\mathtt{if\ y}\ \underline{1}\ (\mathtt{F}(\mathbf{p}\mathtt{x})(\mathbf{p}\mathtt{y}))\big).$$

We can define an encoding $(\!|-|\!) : |[\![\sigma]\!]| \to \mathbb{N}$ from tokens of the coherence space $[\![\sigma]\!]$ to natural numbers as:

— $(\!|n|\!) = n$ if $\sigma = \iota$;
— $(\!|(a_1, a_2)|\!) =< (\!|a_1|\!), (\!|a_2|\!) >$ if $\sigma = \tau_1 \multimap \tau_2$.

This encoding provides an enumeration of tokens in $\mathcal{L}$. We define two indexed families of terms, namely $\mathtt{Sgl}_n^{\sigma} : \sigma$ and $\mathtt{Chk}_n^{(\sigma)} : \sigma \multimap \iota$, by mutual induction. The first is a term of type σ having the nth token (of $[\![\sigma]\!]$) as interpretation. The latter is a term that checks whether the nth token (of $[\![\sigma]\!]$) is included in the interpretation of its argument.

Definitiona 11. The terms $\mathtt{Sgl}_n^{\sigma} : \sigma$ and $\mathtt{Chk}_n^{(\sigma)} : \sigma \multimap \iota$ are defined by mutual induction on σ. If $\sigma = \iota$, $\mathtt{Sgl}_n^{\iota} = \underline{\mathtt{n}}$ and $\mathtt{Chk}_n^{(\iota)} = \lambda\mathtt{y}^{\iota}.\ell\mathtt{if}\ (\underline{\mathtt{n}} \doteq \mathtt{y})\ \underline{0}\ \Omega^{\iota}$. If $\sigma = \sigma_1 \multimap \sigma_2$ where $\sigma_2 = \tau_1 \multimap \cdots \multimap \tau_k \multimap \iota$ for some $k \geqslant 0$, then $\mathtt{Sgl}_n^{\sigma}$ is

$$\lambda\mathtt{f}^{\sigma_1}\lambda\mathtt{g}_1^{\tau_1}\ldots\lambda\mathtt{g}_k^{\tau_k}.\ell\mathtt{if}\,(\mathtt{Chk}_{pi_1(n)}^{(\sigma_1)}\mathtt{f})(\mathtt{Sgl}_{pi_2(n)}^{\sigma_2}\mathtt{g}_1\ldots\mathtt{g}_k)(\Omega^{\sigma_2}\mathtt{g}_1\ldots\mathtt{g}_k)$$

and $\mathtt{Chk}_n^{(\sigma)}$ is $\lambda\mathtt{f}^{\sigma}.\ell\mathtt{if}\ (\mathtt{Chk}_{pi_2(n)}^{(\sigma_2)}(\mathtt{f}\,\mathtt{Sgl}_{pi_1(n)}^{(\sigma_1)}))\ \underline{0}\ \Omega^{\iota}$.

In $\mathtt{Chk}_n^{(\sigma)}$, we use (σ) as a short for $\sigma \multimap \iota$. For sake of readability, sometimes we will abuse the notation by writing $\mathtt{Sgl}_{\underline{\mathtt{n}}}^{\sigma}$ and $\mathtt{Chk}_{\underline{\mathtt{n}}}^{(\sigma)}$ to meaning, respectively $\mathtt{Sgl}_n^{\sigma}$ and $\mathtt{Chk}_n^{(\sigma)}$ where n is the number denoted by the numeral $\underline{\mathtt{n}}$. As an instance, if $\underline{\mathtt{n}} = \lceil\underline{\mathtt{n}_1}, \underline{\mathtt{n}_2}\rceil$ then the

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

term $\mathtt{Sgl}_{\mathtt{n}}^{\iota\multimap\iota}$ is operationally equivalent to the term $\lambda \mathtt{x}^{\iota}.\ell\mathtt{if}\ (\mathtt{x} \doteq \underline{\mathtt{n}_1})\ \underline{\mathtt{n}_2}\ \Omega^{\iota}$, while the term $\mathtt{Chk}_{\mathtt{n}}^{(\iota\multimap\iota)}$ is operationally equivalent to the term $\lambda \mathtt{f}^{\iota\multimap\iota}.\ell\mathtt{if}\ (\mathtt{f}\ \underline{\mathtt{n}_1} \doteq \underline{\mathtt{n}_2})\ \underline{\mathtt{0}}\ \Omega^{\iota}$.

Lemma 6. Let us fix a type σ. If $a \in |[\![\sigma]\!]|$ and $n = (\!|a|\!)$ then,

1. $[\![\mathtt{Sgl}_n^{(\sigma)}]\!]\rho = \{a\}$;
2. if $[\![\mathtt{Chk}_n^{(\sigma)}\ \mathtt{N}]\!]\rho = [\![\underline{\mathtt{0}}]\!]\rho$ then $a \in [\![\mathtt{N}]\!]\rho$.

Proof. The proof is done by mutual induction on σ.
The case $\sigma = \iota$ is immediate. Let us develop the case $\sigma = \sigma_1 \multimap \sigma_2$.To prove (1), let $\sigma_2 = \tau_1 \multimap \cdots \multimap \tau_k \multimap \iota$ for some $k \geqslant 0$.

$$
\begin{aligned}
[\![\mathtt{Sgl}_{\mathtt{n}}^{(\sigma)}]\!]\rho &= \left\{ (a_1, a_2) \;\middle|\; \begin{array}{l} [\![\mathtt{Chk}_{pi_1\underline{\mathtt{n}}}^{(\sigma_1)}\ \mathtt{f}]\!]\rho[\mathtt{f} := \{a_1\}] = \{0\}, \\ a_2 = (b_1, \ldots, b_k, m), \\ m \in [\![\mathtt{Sgl}_{pi_2\underline{\mathtt{n}}}^{(\sigma_2)} g_1 \ldots g_k]\!]\rho[\vec{\mathtt{g}} := \vec{b}] \end{array} \right\} \\
&= \left\{ (a_1, a_2) \;\middle|\; \begin{array}{l} (\!|a_1|\!) = pi_1(n), \\ a_2 = (b_1, \ldots, b_k, m), \\ m \in [\![\mathtt{Sgl}_{pi_2x}^{(\sigma_2)}\ g_1 \ldots g_k]\!]\rho[\vec{\mathtt{g}} := \vec{b}] \end{array} \right\} \\
&= \left\{ (a_1, a_2) \;\middle|\; \begin{array}{l} (\!|a_1|\!) = pi_1 n, \\ (\!|a_2|\!) = pi_2 n \end{array} \right\} \\
&= \{(a_1, a_2) | n = < (\!|a_1|\!), (\!|a_2|\!) > \}
\end{aligned}
$$

where the first row follows by definition of interpretation, the second row follows by applying mutual induction on type σ_1, the third row follows by applying the inductive hypothesis on type σ_2.

To prove (2) suppose $[\![\mathtt{Chk}_n^{(\sigma)}\ \mathtt{N}]\!]\rho = [\![\underline{\mathtt{0}}]\!]\rho$ and let $a = (a_1, a_2)$, then by definition of interpretation, $[\![(\mathtt{Chk}_{pi_2\underline{\mathtt{n}}}^{(\sigma_2)}(\mathtt{N}(\mathtt{Sgl}_{pi_1\underline{\mathtt{n}}}^{(\sigma_1)})))]\!]\rho = \{0\}$. By inductive hypothesis, this implies that $a_2 \in [\![\mathtt{N}(\mathtt{Sgl}_{pi_1\underline{\mathtt{n}}}^{(\sigma_1)})]\!]\rho$. By mutual induction and by definition of interpretation, we have that $a_2 \in \mathcal{F}([\![\mathtt{N}]\!]\rho)(\{a_1\})$. This implies $(a_1, a_2) \in [\![\mathtt{M}]\!]\rho$ by definition of $\mathcal{F}$. □

Lemma 7.

1. Comp($\mathtt{Sgl}_n^{(\sigma)}$).
2. Let $\mathtt{N}^{\sigma}$ be a closed term such that Comp($\mathtt{N}$). If $[\![\mathtt{Chk}_{\mathtt{n}}^{(\sigma)}\ \mathtt{N}]\!] = [\![\underline{\mathtt{0}}]\!]$ then $\mathtt{Chk}_{\mathtt{n}}^{(\sigma)}\ \mathtt{N} \Downarrow \underline{\mathtt{0}}$.

Proof. The proof is immediate by Lemma 5. □

It is easy to build a program taking in input the numeral $\underline{\mathtt{n}}$ and giving back $\mathtt{Sgl}_n^{(\sigma)}$. Instead, it is not possible to extend $\mathtt{Chk}_n^{(\sigma)}$ to a decision program, i.e. a (total) program deciding the membership of the token encoded by n to the interpretation of the considered argument (already a total $\mathtt{Chk}_n^{(\iota)}$ should imply the definability of the halting program, by Lemma 7).

Theorem 3 (token definability).
If $u \in |[\![\sigma]\!]|$ then there exists a closed $\mathtt{M}^{\sigma} \in$ core-SℓPCF such that $[\![\mathtt{M}]\!] = \{u\}$.

Proof. Let $n = (\!|u|\!)$, so $\mathtt{M} = \mathtt{Sgl}_n^{\sigma}$ is our term, by Lemma 6. □

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

Token definability permits to define the separating terms used in the next lemma. This in contrast to what happens in Plotkin (1977) and Paolini (2006), where a finite definability is needed.

Lemma 8 (separability). Let $\sigma \in \mathbb{T}$. For all distinct $f, g \in \mathcal{C}l(\llbracket\sigma\rrbracket)$ there exists a closed term $\mathtt{P}^{\sigma\multimap\iota}$ such that $\mathcal{F}(\llbracket\mathtt{P}\rrbracket)(f) \neq \mathcal{F}(\llbracket\mathtt{P}\rrbracket)(g)$.

Proof. Let $\sigma = \sigma_1 \multimap \ldots \sigma_k \multimap \iota$. Since $f \neq g$, it must exists $a \in f$ such that $a \notin g$ (or vice versa). Assume $a = (a_1, \ldots, a_k, n)$ with $a_1 \in |\llbracket\sigma_1\rrbracket|, \ldots, a_k \in |\llbracket\sigma_k\rrbracket|$ and $n \in \mathbb{N}$. By Theorem 3, for each $i \in \{1, \ldots, k\}$ there is $\mathtt{N}^{\sigma_i} \in$ core-S$\mathscr{I}$PCF such that $\llbracket\mathtt{N}^{\sigma_i}\rrbracket = a_i$.
By choosing $\mathtt{M}^{\sigma\multimap\iota}$ as $\lambda\mathtt{x}^{\sigma}.\mathtt{if}\ (\ (\ \mathtt{x}\,\mathtt{N}^{\sigma_1}\ldots\mathtt{N}^{\sigma_k})\doteq\underline{\mathtt{n}})\ \underline{\mathtt{0}}\ \underline{\mathtt{1}}$, the proof follows. □

As a corollary, a limited form of full abstraction holds restricted to terms that do not contain free *stable* variables. Erroneously, Paolini and Piccolo (2008) claimed a more general result.

Theorem 4 (stable closed completeness). Let $\mathtt{M}^{\sigma}, \mathtt{N}^{\sigma} \in$ core-S$\mathscr{I}$PCF and $\mathrm{SFV}(\mathtt{M}^{\sigma}) = \mathrm{SFV}(\mathtt{N}^{\sigma}) = \varnothing$. If $\mathtt{M} \approx_{\sigma} \mathtt{N}$ then $\llbracket\mathtt{M}\rrbracket = \llbracket\mathtt{N}\rrbracket$.

Proof. We prove the contra-positive. Let us assume $\llbracket\mathtt{M}\rrbracket\rho \neq \llbracket\mathtt{N}\rrbracket\rho$, for $\rho \in \mathrm{Env}$. By Lemma 8, there exists a closed $\mathtt{P}^{\sigma\multimap\iota}$ such that $\mathcal{F}(\llbracket\mathtt{P}\rrbracket\rho)(\llbracket\mathtt{M}\rrbracket\rho) = n_1 \neq \mathcal{F}(\llbracket\mathtt{P}\rrbracket)(\llbracket\mathtt{N}\rrbracket\rho)$. Moreover, by Theorem 3 we can build a closing context $\mathtt{C}[\,']$ such that $\llbracket\mathtt{PM}\rrbracket\rho = \llbracket\mathtt{C}[\mathtt{PM}]\rrbracket\varnothing$ and $\llbracket\mathtt{PN}\rrbracket\rho = \llbracket\mathtt{C}[\mathtt{PN}]\rrbracket\varnothing$ where $\varnothing$ is the empty environment (notice that terms $\mathtt{M}$ and $\mathtt{N}$ may contain free ground or higher-order variables, so token definability is needed to build $\mathtt{C}[\,']$). By adequacy both $\mathtt{C}[\mathtt{PM}] \Downarrow \underline{\mathtt{n}_1}$ and $\mathtt{C}[\mathtt{PN}] \not\Downarrow \underline{\mathtt{n}_1}$. Thus, $\mathtt{M} \not\approx_{\sigma} \mathtt{N}$. □

In the above proof, we can build the context $\mathtt{C}[\,']$ only because we have assumed that the two terms have no free stable variables.

Corollary 2. Let $\mathtt{M}^{\sigma}, \mathtt{N}^{\sigma} \in$ core-S$\mathscr{I}$PCF and $\mathrm{SFV}(\mathtt{M}^{\sigma}) = \mathrm{SFV}(\mathtt{N}^{\sigma}) = \varnothing$. Then, $\mathtt{M} \approx_{\sigma} \mathtt{N}$ if and only if $\llbracket\mathtt{M}\rrbracket = \llbracket\mathtt{N}\rrbracket$.

This corollary holds also for fix-point operational equivalence in a trivial way, since the terms we are considering have no free occurrences of stable variables.

3. Lack of full abstraction

Core-S$\mathscr{I}$PCF does not enjoy the full abstraction extended to all terms. To discuss this issue, we start by proving that core-S$\mathscr{I}$PCF is not able to define all finite cliques of the model by presenting a linear function that is not definable in core-S$\mathscr{I}$PCF, since it is not strongly stable (Bucciarelli and Ehrhard 1991). We use this function to define two terms having free occurrences of stable variables which are operationally equivalent, albeit they are interpreted in two different linear functions.

3.1. *Hypercoherence spaces*

We recall some basic notion on hypercoherence spaces and strongly stable functions, following Bucciarelli and Ehrhard (1994) and Ehrhard (1995).

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

Definitiona 12 (qualitative domain). A *qualitative domain* (Girard 1986) 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:

— $\varnothing \in Q$ 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, then $\bigcup D \in Q$.

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 $u \in Q$. In this case, Q is the set of cliques.

Hypercoherence spaces are defined in Ehrhard (1995).

Definitiona 13 (hypercoherence). A hypercoherence is a pair $X = (|X|, \Gamma(X))$ where $|X|$ is an enumerable set and $\Gamma(X)$ is a subset of $\wp_{fin}(|X|)$ such that $\varnothing \notin \Gamma(X)$ and for any $a \in |X|$, $\{a\} \in \Gamma(X)$.

There is a canonical way to obtain a qualitative domain starting from a hypercoherence. Given a hypercoherence X, we can define the qualitative domain $qD(X)$ as follows.

$$qD(X) = \{x \subseteq |X| \mid \forall u \subseteq_{fin} |X|, (u \neq \varnothing \wedge u \subseteq x) \Rightarrow u \in \Gamma(X)\}.$$

Let X, Y be two sets and A as a subset of pairs having the first component in X and the second in Y (i.e. a Cartesian product). We denote $\pi_1^H(A) = \{x \in X \mid (x, y) \in A\}$ and $\pi_2^H(A) = \{y \in Y \mid (x, y) \in A\}$. Moreover, if X is a set then $|X|$ denote the number of its elements.

Definitiona 14. Let X and Y be hypercoherences.

— The *tensor product* $X \otimes Y$ is the hypercoherence whose web is $|X \otimes Y| = |X| \times |Y|$ and whose atomic coherence is defined by $w \in \Gamma(X \otimes Y)$ iff $\pi_1^H(w) \in \Gamma(X)$ and $\pi_2^H(w) \in \Gamma(Y)$.
— We call *linear implication* of X and Y and we note $X \multimap Y$ the hypercoherence defined by $|X \multimap Y| = |X| \times |Y|$ and $w \in \Gamma(X \multimap Y)$ if and only if w is a non-empty and finite subset of $X \multimap Y$ such that

$$\pi_1^H(w) \in \Gamma(X) \Rightarrow (\pi_2^H(w) \in \Gamma(Y) \wedge |\pi_2^H(w)| = 1 \Rightarrow |\pi_1^H(w)| = 1).$$

— A *linear morphism* between X and Y is an element of $qD(X \multimap Y)$.

We denote with **HLinCoh** the category whose objects are hypercoherences and whose morphisms are linear morphisms between hypercoherences. It is a routine (Bucciarelli and Ehrhard 1994; Ehrhard 1995) to check that the tensor product is commutative, and associative and that the hypercoherence $\mathbf{1} = (\{\star\}, \{\{\star\}\})$ is its neutral element. It is also routine to check that there is a natural isomorphism between $\mathbf{HLinCoh}(X \otimes Y, Z) \cong \mathbf{HLinCoh}(X, Y \multimap Z)$, which makes **HLinCoh** a symmetric monoidal closed category.

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

3.2. *Non-definible finite cliques*

It is important to remark that **HLinCoh** provides a correct model of core-S$\mathscr{A}$PCF under a standard interpretation. A proof can be easily obtained mutatis mutandis the proof of correctness for coherence spaces given in previous section. We show that there are linear functions of the model $\mathcal{L}$ that cannot be defined in core-S$\mathscr{A}$PCF by proving that these functions are not linear morphisms in **HLinCoh**. So, they cannot be defined in core-S$\mathscr{A}$PCF.

Let $k_1, k_2, k_3, k_4 \in \mathbb{N}$; we define $\mathtt{G\eth r}^{\underline{k}_1,\underline{k}_2,\underline{k}_3,\underline{k}_4} : (\iota \multimap \iota) \multimap (\iota \multimap \iota) \multimap (\iota \multimap \iota) \multimap \iota$ to be the following trace

$$\left\{ \begin{array}{l} \mathbf{(}(0,0)\,,\,(1,0)\,,\,(0,1)\,,\,k_1\mathbf{)}, \\ \mathbf{(}(0,1)\,,\,(0,0)\,,\,(1,0)\,,\,k_2\mathbf{)}, \\ \mathbf{(}(1,0)\,,\,(0,1)\,,\,(0,0)\,,\,k_3\mathbf{)}, \\ \mathbf{(}(1,1)\,,\,(1,1)\,,\,(1,1)\,,\,k_4\mathbf{)} \end{array} \right\}$$

where for sake of simplicity we omitted a full parenthesizing. It is immediate to check that $\mathtt{G\eth r}^{\underline{k}_1,\underline{k}_2,\underline{k}_3,\underline{k}_4}$ is a coherent trace (i.e. a clique) in $[\![(\iota \multimap \iota) \multimap (\iota \multimap \iota) \multimap (\iota \multimap \iota) \multimap \iota]\!]$. As instance, we note that $\mathtt{G\eth r}^{\underline{1},\underline{2},\underline{3},\underline{4}}$ is exactly the operator $\mathtt{G\eth r}$ defined in Paolini and Piccolo (2008)[†].

We prove that, for each $k_1, k_2, k_3, k_4 \in \mathbb{N}$, $\mathtt{G\eth r}^{\underline{k}_1,\underline{k}_2,\underline{k}_3,\underline{k}_4}$ does not correspond to a linear morphism in the category of hypercoherences. Let $w = \{(0,0),(0,1),(1,0),(1,1)\}$, we first observe that trivially $w \in \Gamma(\mathbf{N} \multimap \mathbf{N})$ since $\pi_1^H(w) = \{0,1\} \notin \Gamma(\mathbf{N})$. Thus, $\pi_1^H(\mathtt{G\eth r}^{\underline{k}_1,\underline{k}_2,\underline{k}_3,\underline{k}_4}) \in \Gamma((\mathbf{N} \multimap \mathbf{N}) \otimes (\mathbf{N} \multimap \mathbf{N}) \otimes (\mathbf{N} \multimap \mathbf{N}))$ because $\pi_1^H(\pi_1^H(\mathtt{G\eth r}^{\underline{k}_1,\underline{k}_2,\underline{k}_3,\underline{k}_4})) = \pi_2^H(\pi_1^H(\mathtt{G\eth r}^{\underline{k}_1,\underline{k}_2,\underline{k}_3,\underline{k}_4})) = \pi_3^H(\pi_1^H(\mathtt{G\eth r}^{\underline{k}_1,\underline{k}_2,\underline{k}_3,\underline{k}_4})) = w$ which is in $\Gamma(\mathbf{N} \multimap \mathbf{N})$.

We consider two cases:

— suppose $k_1 = k_2 = k_3 = k_4$: we have that $\pi_2^H(\mathtt{G\eth r}^{\underline{k}_1,\underline{k}_2,\underline{k}_3,\underline{k}_4}) = \{k_1\}$ is in $\Gamma(\mathbf{N})$ but $|\pi_2^H(\mathtt{G\eth r}^{\underline{k}_1,\underline{k}_2,\underline{k}_3,\underline{k}_4})| = 1$ while $|\pi_1^H(\mathtt{G\eth r}^{\underline{k}_1,\underline{k}_2,\underline{k}_3,\underline{k}_4})| \neq 1$.

— suppose there are i, j such that $k_i \neq k_j$: in this case we have that $\{k_i, k_j\} \subseteq \pi_2^H(\mathtt{G\eth r}^{\underline{k}_1,\underline{k}_2,\underline{k}_3,\underline{k}_4})$ which is not in $\Gamma(\mathbf{N})$ because $\{k_i, k_j\} \notin \Gamma(\mathbf{N})$.

Thus, $\mathtt{G\eth r}^{\underline{k}_1,\underline{k}_2,\underline{k}_3,\underline{k}_4}$ is not a linear morphism in **HLinCoh**. So it is not definable in core-S$\mathscr{A}$PCF.

Corollary 3. For all $k_1, k_2, k_3, k_4 \in \mathbb{N}$, $\mathtt{G\eth r}^{\underline{k}_1,\underline{k}_2,\underline{k}_3,\underline{k}_4}$ is not definable in core-S$\mathscr{A}$PCF.

Let $\mathtt{M} = \mathscr{F}\Omega^{\iota\multimap\iota}\Omega^{\iota\multimap\iota}\Omega^{\iota\multimap\iota}$ and

$$\mathtt{N} = \ell\mathtt{if} \left(\begin{array}{l} \mathscr{F}(\mathtt{Sgl}^{\iota\multimap\iota}_{<\underline{0},\underline{0}>})(\mathtt{Sgl}^{\iota\multimap\iota}_{<\underline{1},\underline{0}>})(\mathtt{Sgl}^{\iota\multimap\iota}_{<\underline{0},\underline{1}>})\ \mathtt{and} \\ \mathscr{F}(\mathtt{Sgl}^{\iota\multimap\iota}_{<\underline{0},\underline{1}>})(\mathtt{Sgl}^{\iota\multimap\iota}_{<\underline{0},\underline{0}>})(\mathtt{Sgl}^{\iota\multimap\iota}_{<\underline{1},\underline{0}>})\ \mathtt{and} \\ \mathscr{F}(\mathtt{Sgl}^{\iota\multimap\iota}_{<\underline{1},\underline{0}>})(\mathtt{Sgl}^{\iota\multimap\iota}_{<\underline{0},\underline{1}>})(\mathtt{Sgl}^{\iota\multimap\iota}_{<\underline{0},\underline{0}>})\ \mathtt{and} \\ \mathscr{F}(\mathtt{Sgl}^{\iota\multimap\iota}_{<\underline{1},\underline{1}>})(\mathtt{Sgl}^{\iota\multimap\iota}_{<\underline{1},\underline{1}>})(\mathtt{Sgl}^{\iota\multimap\iota}_{<\underline{1},\underline{1}>}) \end{array} \right) \underline{7}\ \Omega^{\iota}$$

be core-S$\mathscr{A}$PCF terms such that $\mathscr{F}^{(\iota\multimap\iota)\multimap(\iota\multimap\iota)\multimap(\iota\multimap\iota)\multimap\iota} \vdash \mathtt{M}, \mathtt{N} : \iota$. It is easy to see that $[\![\mathtt{M}]\!]\rho \neq [\![\mathtt{N}]\!]\rho$, by taking $\rho(\mathscr{F})$ be the trace of $\mathtt{G\eth r}^{\underline{1},\underline{2},\underline{3},\underline{4}}$. While $\mathtt{M}, \mathtt{N}$ are operationally

[†] The index 2 in the $\mathtt{G\eth r}$ notation is put to emphasize that it is a second-order operator. It is different from the $\mathtt{gor}$ operator defined in Paolini (2006), which is a first-order operator.

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

Table 3. *(a) Type system, (b) operational semantics and (c) linear interpretation for the* `which?` *and* `ℓet-ℓor` *operators.*

$$\frac{}{\vdash \texttt{which?} : ((\iota \multimap \iota) \multimap \iota) \multimap \iota}\ (\text{w})$$

$$\frac{\Gamma \cap \Delta = \varnothing \quad \Delta\restriction \ell = \texttt{f}_1^{\sigma_1},\ldots,\texttt{f}_k^{\sigma_k} \quad \Gamma_1 \vdash \texttt{N}_1 : \sigma_1 \ \ldots\ \Gamma_k \vdash \texttt{N}_k : \sigma_k \quad \Delta \vdash \texttt{M}_i : \iota\ (1\leqslant i\leqslant 3)}{\Delta\restriction S, \Delta\restriction \iota, \Gamma_1,\ldots,\Gamma_k \vdash \underline{\ell\texttt{et}}\ \texttt{f}_1 = \texttt{N}_1,\ldots,\texttt{f}_k = \texttt{N}_k\ \underline{\texttt{in}\ell\texttt{or}}\ \texttt{M}_1\ \texttt{M}_2\ \texttt{M}_3 : \iota}\ (\ell\text{et-}\ell\text{or})$$

$$\frac{\texttt{M}(\lambda x^{\iota}.\ell\texttt{if}(\overbrace{\texttt{p}\ldots\texttt{p}}^{\texttt{k}}\ \texttt{x})\ \underline{\texttt{k}}\ (\texttt{p0})) \Downarrow \underline{\texttt{n}}}{\texttt{which?}\,\texttt{M}^{(\iota\multimap\iota)\multimap\iota} \Downarrow \lceil \underline{\texttt{n}}, \underline{\texttt{k}} \rceil}\ (\text{w})$$

$$\frac{\texttt{M}_1[\texttt{Sgl}_{\texttt{n}_1}^{\sigma_1}/\texttt{f}_1,\ldots,\texttt{Sgl}_{\texttt{n}_k}^{\sigma_k}/\texttt{f}_k] \Downarrow \underline{0} \quad \texttt{M}_2[\texttt{Sgl}_{\texttt{n}_1}^{\sigma_1}/\texttt{f}_1,\ldots,\texttt{Sgl}_{\texttt{n}_k}^{\sigma_k}/\texttt{f}_k] \Downarrow \underline{\texttt{sm}} \quad \texttt{Chk}_{\texttt{n}_j}^{(\sigma_k)}\texttt{N}_j \Downarrow \underline{0} \quad (j\in\{1,\ldots,k\})}{\underline{\ell\texttt{et}}\ \texttt{f}_1^{\sigma_1} = \texttt{N}_1,\ldots,\texttt{f}_k^{\sigma_k} = \texttt{N}_k\ \underline{\texttt{in}\ell\texttt{or}}\ \texttt{M}_1\ \texttt{M}_2\ \texttt{M}_3 \Downarrow \underline{\texttt{m}}}\ (1\text{lgor})$$

$$\frac{\texttt{M}_2[\texttt{Sgl}_{\texttt{n}_1}^{\sigma_1}/\texttt{f}_1,\ldots,\texttt{Sgl}_{\texttt{n}_k}^{\sigma_k}/\texttt{f}_k] \Downarrow \underline{0} \quad \texttt{M}_3[\texttt{Sgl}_{\texttt{n}_1}^{\sigma_1}/\texttt{f}_1,\ldots,\texttt{Sgl}_{\texttt{n}_k}^{\sigma_k}/\texttt{f}_k] \Downarrow \underline{\texttt{sm}} \quad \texttt{Chk}_{\texttt{n}_j}^{(\sigma_k)}\texttt{N}_j \Downarrow \underline{0} \quad (j\in\{1,\ldots,k\})}{\underline{\ell\texttt{et}}\ \texttt{f}_1^{\sigma_1} = \texttt{N}_1,\ldots,\texttt{f}_k^{\sigma_k} = \texttt{N}_k\ \underline{\texttt{in}\ell\texttt{or}}\ \texttt{M}_1\ \texttt{M}_2\ \texttt{M}_3 \Downarrow \underline{\texttt{m}}}\ (2\text{lgor})$$

$$\frac{\texttt{M}_3[\texttt{Sgl}_{\texttt{n}_1}^{\sigma_1}/\texttt{f}_1,\ldots,\texttt{Sgl}_{\texttt{n}_k}^{\sigma_k}/\texttt{f}_k] \Downarrow \underline{0} \quad \texttt{M}_1[\texttt{Sgl}_{\texttt{n}_1}^{\sigma_1}/\texttt{f}_1,\ldots,\texttt{Sgl}_{\texttt{n}_k}^{\sigma_k}/\texttt{f}_k] \Downarrow \underline{\texttt{sm}} \quad \texttt{Chk}_{\texttt{n}_j}^{(\sigma_k)}\texttt{N}_j \Downarrow \underline{0} \quad (j\in\{1,\ldots,k\})}{\underline{\ell\texttt{et}}\ \texttt{f}_1^{\sigma_1} = \texttt{N}_1,\ldots,\texttt{f}_k^{\sigma_k} = \texttt{N}_k\ \underline{\texttt{in}\ell\texttt{or}}\ \texttt{M}_1\ \texttt{M}_2\ \texttt{M}_3 \Downarrow \underline{\texttt{m}}}\ (3\text{lgor})$$

$$[\![\texttt{which?}]\!]\rho = \{(((k,k),n),\lceil n,k\rceil) \mid k,n\in\mathbb{N}\}$$

$$[\![\underline{\ell\texttt{et}}\ \vec{\texttt{f}} = \vec{\texttt{N}}\ \underline{\texttt{in}\ell\texttt{or}}\ \texttt{M}_1\ \texttt{M}_2\ \texttt{M}_3]\!]\rho = \left\{ n\in\mathbb{N} \,\middle|\, \exists \vec{a}\in\overrightarrow{[\![\texttt{N}]\!]\rho}\ \begin{array}{l}[\![\texttt{M}_3]\!]\rho[\vec{\texttt{f}} := \vec{a}] = \{0\}\wedge \\ [\![\texttt{M}_1]\!]\rho[\vec{\texttt{f}} := \vec{a}] = \{n+1\}\end{array} \right\}$$
$$\cup \left\{ n\in\mathbb{N} \,\middle|\, \exists \vec{a}\in\overrightarrow{[\![\texttt{N}]\!]\rho}\ \begin{array}{l}[\![\texttt{M}_1]\!]\rho[\vec{\texttt{f}} := \vec{a}] = \{0\}\wedge \\ [\![\texttt{M}_2]\!]\rho[\vec{\texttt{f}} := \vec{a}] = \{n+1\}\end{array} \right\}$$
$$\cup \left\{ n\in\mathbb{N} \,\middle|\, \exists \vec{a}\in\overrightarrow{[\![\texttt{N}]\!]\rho}\ \begin{array}{l}[\![\texttt{M}_2]\!]\rho[\vec{\texttt{f}} := \vec{a}] = \{0\}\wedge \\ [\![\texttt{M}_3]\!]\rho[\vec{\texttt{f}} := \vec{a}] = \{n+1\}\end{array} \right\}$$

equivalent, since to separate the two terms, a term behaving like $\texttt{G}\mathring{\texttt{o}}\texttt{r}^{\underline{1},\underline{2},\underline{3},\underline{4}}$ is necessary. But $\texttt{G}\mathring{\texttt{o}}\texttt{r}^{\underline{1},\underline{2},\underline{3},\underline{4}}$ is not definable in core-SℓPCF as shown by Corollary 3.

4. The language SℓPCF$_\star$

We extend the core-SℓPCF language by means of two new operators: `which?` and `ℓet-ℓor`. The resulting language SℓPCF$_\star$ is simply that presented in Gaboardi *et al.* (2011).

Definitiona 15. The SℓPCF$_\star$ *pre-terms* are defined by extending the grammar of Core-SℓPCF$_\star$ pre-terms as follows:

$$\texttt{M} ::= \cdots \mid \texttt{which?} \mid \underline{\ell\texttt{et}}\ \texttt{f}_1^{\sigma_1\multimap\tau_1} = \texttt{M},\ldots,\texttt{f}_k^{\sigma_k\multimap\tau_k} = \texttt{M}\ \underline{\texttt{in}\ell\texttt{or}}\ \texttt{M}\ \texttt{M}\ \texttt{M}$$

The *terms* of SℓPCF$_\star$ are the pre-terms typable by using the type system in Table 1(a) extended by the rules in Table 3(a).

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

The which? operator corresponds to a primitive control operator that permits to obtain besides the result of an evaluation also the information on which part of the input has been used during the evaluation.

The ℓet-ℓor generalizes the behaviour of the G$\ddot{\text{o}}$r operator presented in the previous section by forcing a linear evaluation. It is worth noting that the contexts of the terms $\mathtt{M_i}$ in the (ℓet-ℓor) rule are managed in an additive way, contracting common (ground, linear and stable) variables. However, note that a ℓet-ℓor binds all the linear variables in its three branches. So, a form of syntactic linearity *by slice* for linear variables is preserved, see Gaboardi *et al.* (2011) for more details.

In order to deal with the which? and the ℓet-ℓor operators, we need a careful evaluation of terms. In particular, the challenge is to evaluate the ℓet-ℓor operator without violating the denotational linearity. To obtain this, we present specific evaluation rules that are described using the terms $\mathtt{Sgl}_n^\sigma$ and $\mathtt{Chk}_n^{(\sigma)}$ introduced in the previous section.

Definitiona 16. The *evaluation relation* $\Downarrow \subseteq \mathrm{P} \times \mathcal{N}$ for S$\mathscr{L}$PCF$_\star$ programs is the smallest relation satisfying the rules in Table 1(b) extended by the rules in Table 3(b).

The evaluation rules for which? and ℓet-ℓor are patterns for infinite rules, likewise to that of the existential operator of Plotkin (1977).

The result of the evaluation of which? applied to a term M consists of a pair $\lceil \underline{\mathtt{n}}, \underline{\mathtt{k}} \rceil$ where $\underline{\mathtt{n}}$ is the result of the evaluation of the term $\mathtt{M}(\lambda \mathtt{x}^\iota.\mathtt{x})$ while $\underline{\mathtt{k}}$ is the unique numeral that M gives as argument to $\lambda \mathtt{x}^\iota.\mathtt{x}$. The fact that there exists a unique such $\underline{\mathtt{k}}$ follows from the fact that M represents a linear function. In this sense, which? can be seen as a primitive control operator.

The evaluation of the ℓet-ℓor operator is obtained by three distinct rules that explore pair-wisely the ℓet-ℓor branches. The evaluation of a ℓet-ℓor can be performed only in the case two between $\mathtt{M_1}, \mathtt{M_2}$ and $\mathtt{M_3}$ evaluate to the values $\underline{0}$ and $\underline{\mathtt{m}+1}$ respectively by replacing to $\mathtt{f}_1, \ldots, \mathtt{f}_k$ a (same) single-traced terms denotationally included in the trace of $\mathtt{N}_1, \ldots, \mathtt{N}_k$ respectively. For instance, the (1lgor) can be applied in the case $\mathtt{M_1}[\mathtt{N_1}/\mathtt{f_1}, \ldots, \mathtt{N_k}/\mathtt{f_k}] \Downarrow \underline{0}$ and $\mathtt{M_2}[\mathtt{N_1}/\mathtt{f_1}, \ldots, \mathtt{N_k}/\mathtt{f_k}] \Downarrow \mathbf{s}\underline{\mathtt{m}}$ and the same *single* information of each $\mathtt{N}_j$ coded by numerals $\underline{\mathtt{n}}_j$ is used in both evaluations. Albeit ex-ante we do not know what is the right pair of ℓet-ℓor branches to evaluate, they are established during the course of the evaluation, so ex-post only one of the three rules can converge.

The interpretation of which? and ℓet-ℓor reflects such ideas.

Definitiona 17. Let $\mathtt{M}^\sigma$ be a term of S$\mathscr{L}$PCF$_\star$ and $\rho \in \mathrm{Env}$. The *linear interpretation* $[\![\mathtt{M}^\sigma]\!]\rho \in \mathcal{C}l([\![\sigma]\!])$ is defined by the equations in Table 1(c) extended by the ones for the which? and the ℓet-ℓor operators in Table 3(c).

We can now extend the adequacy and correctness results of Section 2.4 to the whole S$\mathscr{L}$PCF$_\star$ language. The first step is to extend Lemma 5.

Lemma 9. Let $\mathtt{M}^\sigma$ be a term of S$\mathscr{L}$PCF$_\star$. Then Comp($\mathtt{M}^\sigma$).

Proof. By induction on the shape of M, similarly to the proof of Lemma 5. We detail only the two new cases.

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

— $\mathtt{M} = \mathtt{which?}$. Let $\mathtt{N}^{(\iota\multimap\iota)\multimap\iota}$ be a term such that Comp(N) and suppose $[\![\mathtt{which?\,N}]\!] = [\![\lceil \underline{\mathtt{n}}, \underline{\mathtt{k}} \rceil]\!]$. This means that $\mathcal{F}([\![\mathtt{N}]\!]\rho)(\{(k,k)\}) = \mathcal{F}([\![\mathtt{N}]\!]\rho)([\![\lambda \mathtt{x}^{\iota}.\ell\mathtt{if}(\overbrace{\mathtt{p}\ldots\mathtt{p}}^{\underline{\mathtt{k}}}\ \mathtt{x})\ \underline{\mathtt{k}}\ \Omega^{\iota}]\!]\rho) = \{n\} = [\![\underline{\mathtt{n}}]\!]\rho$, by definition of interpretation. Then $\mathtt{M}(\lambda \mathtt{x}^{\iota}.\ell\mathtt{if}(\overbrace{\mathtt{p}\ldots\mathtt{p}}^{\underline{\mathtt{k}}}\ \mathtt{x})\ \underline{\mathtt{k}}\ \Omega^{\iota}) \Downarrow \underline{\mathtt{n}}$ by definition of Comp. Then we conclude by applying the evaluation rule (w).

— $\mathtt{M} = \underline{\ell\mathtt{et}}\ \mathtt{f}_1^{\sigma_1} = \mathtt{N}_1, \ldots, \mathtt{f}_{\mathtt{k}}^{\sigma_{\mathtt{k}}} = \mathtt{N}_{\mathtt{k}}\ \underline{\mathtt{in}\ell\mathtt{or}}\ \mathtt{M}_1\ \mathtt{M}_2\ \mathtt{M}_3$. Let $\mathrm{FV}(\mathtt{M}) = \{\varkappa_1^{\mu_1}, \ldots, \varkappa_h^{\mu_h}\}$ for $h \geqslant 0$. Assume $\mathtt{P}_1^{\mu_1}, \ldots, \mathtt{P}_h^{\mu_h}$ to be closed terms such that $\mathrm{Comp}(\mathtt{P}_i)$ for $1 \leqslant i \leqslant h$. Let $[\![(\ell\mathtt{et}\ \mathtt{f}_1^{\sigma_1} = \mathtt{N}_1, \ldots, \mathtt{f}_{\mathtt{k}}^{\sigma_{\mathtt{k}}} = \mathtt{N}_{\mathtt{k}}\ \underline{\mathtt{in}\ell\mathtt{or}}\ \mathtt{M}_1\ \mathtt{M}_2\ \mathtt{M}_3)[\mathtt{P}_1/\varkappa_1, \ldots, \mathtt{P}_{\mathtt{h}}/\varkappa_{\mathtt{h}}]]\!]\rho = [\![\underline{\mathtt{n}}]\!]\rho$. There are three cases.

1. There exist $\vec{a} \in \overrightarrow{[\![\mathtt{N}[\mathtt{P}_1/\varkappa_1, \ldots, \mathtt{P}_{\mathtt{h}}/\varkappa_{\mathtt{h}}]]\!]\rho}$ such that $[\![\mathtt{M}_3[\mathtt{P}_1/\varkappa_1, \ldots, \mathtt{P}_{\mathtt{h}}/\varkappa_{\mathtt{h}}]]\!]\rho[\vec{\mathtt{f}} := \vec{a}] = \{0\}$ and $[\![\mathtt{M}_1[\mathtt{P}_1/\varkappa_1, \ldots, \mathtt{P}_{\mathtt{h}}/\varkappa_{\mathtt{h}}]]\!]\rho[\vec{\mathtt{f}} := \vec{a}] = \{n+1\}$.
2. There exist $\vec{a} \in \overrightarrow{[\![\mathtt{N}[\mathtt{P}_1/\varkappa_1, \ldots, \mathtt{P}_{\mathtt{h}}/\varkappa_{\mathtt{h}}]]\!]\rho}$ such that $[\![\mathtt{M}_1[\mathtt{P}_1/\varkappa_1, \ldots, \mathtt{P}_{\mathtt{h}}/\varkappa_{\mathtt{h}}]]\!]\rho[\vec{\mathtt{f}} := \vec{a}] = \{0\}$ and $[\![\mathtt{M}_2[\mathtt{P}_1/\varkappa_1, \ldots, \mathtt{P}_{\mathtt{h}}/\varkappa_{\mathtt{h}}]]\!]\rho[\vec{\mathtt{f}} := \vec{a}] = \{n+1\}$.
3. There exist $\vec{a} \in \overrightarrow{[\![\mathtt{N}[\mathtt{P}_1/\varkappa_1, \ldots, \mathtt{P}_{\mathtt{h}}/\varkappa_{\mathtt{h}}]]\!]\rho}$ such that $[\![\mathtt{M}_2[\mathtt{P}_1/\varkappa_1, \ldots, \mathtt{P}_{\mathtt{h}}/\varkappa_{\mathtt{h}}]]\!]\rho[\vec{\mathtt{f}} := \vec{a}] = \{0\}$ and $[\![\mathtt{M}_3[\mathtt{P}_1/\varkappa_1, \ldots, \mathtt{P}_{\mathtt{h}}/\varkappa_{\mathtt{h}}]]\!]\rho[\vec{\mathtt{f}} := \vec{a}] = \{n+1\}$.

We shorten the list $\mathtt{f}_1, \ldots, \mathtt{f}_{\mathtt{k}}$ by $\vec{\mathtt{f}}$. We develop only the first case: the others are similar. Suppose that $\vec{m}$ is a sequence of natural number of the same length of $\vec{a}$ such that each m_i is the encoding of the token a_i (for $1 \leqslant i \leqslant k$). By Lemma 6.1 $a_i = [\![\mathtt{Sgl}_{\underline{\mathtt{m}_{\mathtt{i}}}}^{(\sigma_{\mathtt{i}})}]\!]\rho$ and, moreover, by applying Lemma 2.3, it follows

$$[\![\mathtt{M}_3[\mathtt{P}_1/\varkappa_1, \ldots, \mathtt{P}_{\mathtt{h}}/\varkappa_{\mathtt{h}}, \mathtt{Sgl}_{\underline{\mathtt{m}_1}}^{(\sigma_1)}/\mathtt{f}_1, \ldots, \mathtt{Sgl}_{\underline{\mathtt{m}_{\mathtt{k}}}}^{(\sigma_{\mathtt{k}})}/\mathtt{f}_{\mathtt{k}}]]\!]\rho = [\![\underline{0}]\!]\rho \ \text{ and,}$$
$$[\![\mathtt{M}_1[\mathtt{P}_1/\varkappa_1, \ldots, \mathtt{P}_{\mathtt{h}}/\varkappa_{\mathtt{h}}, \mathtt{Sgl}_{\underline{\mathtt{m}_1}}^{(\sigma_1)}/\mathtt{f}_1, \ldots, \mathtt{Sgl}_{\underline{\mathtt{m}_{\mathtt{k}}}}^{(\sigma_{\mathtt{k}})}/\mathtt{f}_{\mathtt{k}}]]\!]\rho = [\![\underline{\mathtt{n}+1}]\!]\rho.$$

By Lemma 7.2 and by induction, we obtain

$$\mathtt{M}_3[\mathtt{P}_1/\varkappa_1, \ldots, \mathtt{P}_{\mathtt{h}}/\varkappa_{\mathtt{h}}, \mathtt{Sgl}_{\underline{\mathtt{m}_1}}^{(\sigma_1)}/\mathtt{f}_1, \ldots, \mathtt{Sgl}_{\underline{\mathtt{m}_{\mathtt{k}}}}^{(\sigma_{\mathtt{k}})}/\mathtt{f}_{\mathtt{k}}] \Downarrow \underline{0} \ \text{ and,}$$
$$\mathtt{M}_1[\mathtt{P}_1/\varkappa_1, \ldots, \mathtt{P}_{\mathtt{h}}/\varkappa_{\mathtt{h}}, \mathtt{Sgl}_{\underline{\mathtt{m}_1}}^{(\sigma_1)}/\mathtt{f}_1, \ldots, \mathtt{Sgl}_{\underline{\mathtt{m}_{\mathtt{k}}}}^{(\sigma_{\mathtt{k}})}/\mathtt{f}_{\mathtt{k}}] \Downarrow \underline{\mathtt{n}+1}.$$

Furthermore, $[\![\mathtt{Chk}_{\underline{\mathtt{m}_{\mathtt{i}}}}^{(\sigma_{\mathtt{i}})}(\mathtt{N}_{\mathtt{i}}[\mathtt{P}_1/\varkappa_1, \ldots, \mathtt{P}_{\mathtt{h}}/\varkappa_{\mathtt{h}}])]\!]\rho = [\![\underline{0}]\!]\rho$ by Lemma 6.2 and by definition of interpretation. Then, by Lemma 7.2, we have that $\mathtt{Chk}_{\underline{\mathtt{m}_{\mathtt{i}}}}^{(\sigma_{\mathtt{i}})}\ \mathtt{N}_{\mathtt{i}}[\mathtt{P}_1/\varkappa_1, \ldots, \mathtt{P}_{\mathtt{h}}/\varkappa_{\mathtt{h}}] \Downarrow \underline{0}$. So, we can conclude by induction and by applying the evaluation rule (l1gor).

□

Corollary 4 (adequacy). For all $\mathtt{M} \in \mathrm{P}$, for all $\underline{\mathtt{n}}$, $[\![\mathtt{M}]\!]\rho = [\![\underline{\mathtt{n}}]\!]\rho$ if and only if $\mathtt{M} \Downarrow \underline{\mathtt{n}}$.

Theorem 5 (correctness). Let $\mathtt{M}^{\sigma}, \mathtt{N}^{\sigma}$ be terms of S⁄PCF$_\star$.
If $[\![\mathtt{M}]\!]_\rho = [\![\mathtt{N}]\!]_\rho$, for each $\rho \in \mathrm{Env}$, then $\mathtt{M} \approx^{\sigma} \mathtt{N}$.

4.1. S⁄PCF$_\star$ *program examples*

We discuss some examples of the use of the which? and of the ℓet-ℓor operators. Let us start with two examples for which?. The first example show how to use it in order to define a kind of linear exception. Consider the term

$$\mathtt{Try}_{\mathtt{n}} = \lambda \mathtt{f}^{(\iota\multimap\iota)\multimap\iota}\mathtt{g}^{\iota\multimap\iota}.\big(\lambda \mathtt{x}^{\iota}.\ell\mathtt{if}\ ((\boldsymbol{\pi}_2\ \mathtt{x}) \doteq \mathtt{n})\ \underline{0}\ \mathbf{s}(\boldsymbol{\pi}_1\ \mathtt{x})\big)\big(\mathtt{which?}\,\lambda \mathtt{h}.\mathtt{f}(\lambda \mathtt{x}^{\iota}.\mathtt{g}(\mathtt{hx}))\big)$$

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

$\mathtt{Try}_{\mathtt{n}}$ takes in input a functional $\mathtt{f}^{(\iota\multimap\iota)\multimap\iota}$ and a function $\mathtt{g}^{\iota\multimap\iota}$. It tries to evaluate the term $\mathtt{f\,g}$ and it raises an exception (returning 0) if during the evaluation, $\mathtt{f}$ observes $\mathtt{g}$ on the particular value $\mathtt{n}$. Otherwise, it returns the successor of the result of the evaluation.

The second example is more involved, and it shows how to use the $\mathtt{which?}$ operator to define a family of programming constructs ($@\mathtt{wh}?^{\sigma}_{\tau}$) useful to collect precise run-time information. Let us introduce the operators $@\mathtt{wh}?^{\sigma}_{\tau} : (\sigma \multimap \tau) \multimap \sigma \multimap \tau$, with the following operational semantics:

$$\frac{\mathtt{M}^{\sigma\multimap\tau}\mathtt{Sgl}^{(\sigma)}_{\mathtt{m}}\mathtt{P}_1\ldots\mathtt{P}_{\mathtt{k}} \Downarrow \underline{\mathtt{n}} \qquad \mathtt{Chk}^{(\sigma)}_{\mathtt{m}}\ \mathtt{N} \Downarrow \underline{0}.}{@\mathtt{wh}?^{\sigma}_{\tau}\,\mathtt{M}^{\sigma\multimap\tau}\,\mathtt{N}^{\sigma}\mathtt{P}_1\ldots\mathtt{P}_{\mathtt{k}} \Downarrow \lceil\underline{\mathtt{n}},\underline{\mathtt{m}}\rceil}$$

Observe that $(@\mathtt{wh}?^{\sigma}_{\tau}\,\mathtt{M\,N})\mathtt{P}_1\ldots\mathtt{P}_{\mathtt{k}} \Downarrow$ if and only if $\mathtt{M\,N\,P}_1\ldots\mathtt{P}_{\mathtt{k}} \Downarrow$. This control operator gives back the result $\underline{\mathtt{n}}$ of the evaluation of $\mathtt{M\,N\,P}_1\ldots\mathtt{P}_{\mathtt{k}}$ together with the numeral $\underline{\mathtt{m}}$ encoding the part of the trace of $\mathtt{N}$ used for the evaluation. This information will be essential in the proof of the finite definability Theorem 7.

The definition of $@\mathtt{wh}?^{\sigma}_{\tau}$ in $\mathcal{S}$PCF$_\star$ is slightly involved, so we give first some examples to understand how this can be done.

First, observe that the term $\lambda\mathtt{f}^{\iota\multimap\iota}.\lambda\mathtt{x}^{\iota}.\,\mathtt{which?}(\lambda\mathtt{h}^{\iota\multimap\iota}.\mathtt{f}\ (\mathtt{h}\ \mathtt{x}))$ implements $@\mathtt{wh}?^{\iota}_{\iota}$. Informally, the idea of this program is that $\mathtt{which?}$ can use the variable $\mathtt{h}^{\iota\multimap\iota}$ as an *observer*, to retrieve the value associated to $\mathtt{x}$ during a specific evaluation. Suppose now that we want to build a term behaving as $@\mathtt{wh}?^{\iota\multimap\iota}_{\iota}$. Assume $\mathtt{M}^{(\iota\multimap\iota)\multimap\iota}$ and $\mathtt{N}^{\iota\multimap\iota}$ to be two terms such that $\mathtt{M\ N} \Downarrow \underline{\mathtt{n}}$. This evaluation is driven by $\mathtt{M}$: it inputs $\mathtt{N}$, it applies $\mathtt{N}$ to a (unique) suitable ground argument (we name this argument $\underline{\mathtt{i}}^{\mathtt{N}}$ and we name $\underline{\mathtt{o}}^{\mathtt{N}}$ the output of the evaluation of $\mathtt{N}\,\underline{\mathtt{i}}^{\mathtt{N}} \Downarrow$) and, last, it computes the result $\underline{\mathtt{n}}$ (possibly, by using $\underline{\mathtt{o}}^{\mathtt{N}}$). To reach our purpose, we need to find a way to bring in the output information about $\underline{\mathtt{i}}^{\mathtt{N}}$ and $\underline{\mathtt{o}}^{\mathtt{N}}$. Assume $\mathtt{I}$ to denote $\lambda\mathtt{y}^{\iota}.\mathtt{y}$, it is easy to understand that $\mathtt{M}(\lambda\mathtt{x}^{\iota}.(\mathtt{I}(\mathtt{N}(\mathtt{Ix})))) \Downarrow \underline{\mathtt{n}}$ by harmless eta-expansions and two additional copies of the identity. Patently, the rightmost identity forwards $\underline{\mathtt{i}}^{\mathtt{N}}$ and the leftmost identity forwards $\underline{\mathtt{o}}^{\mathtt{N}}$. By a sharp use of $\mathtt{which?}$, we replace the rightmost identity, so that $\mathtt{which?}(\lambda\mathtt{h}_1.\mathtt{M}(\lambda\mathtt{x}^{\iota}.(\mathtt{I}(\mathtt{N}(\mathtt{h}_1\mathtt{x}))))) \Downarrow \lceil\underline{\mathtt{n}},\underline{\mathtt{i}}^{\mathtt{N}}\rceil$. Re-applying this trick $\mathtt{which?}(\lambda\mathtt{h}_0.\mathtt{which?}(\lambda\mathtt{h}_1.\mathtt{M}(\lambda\mathtt{x}^{\iota}.(\mathtt{h}_0(\mathtt{N}(\mathtt{h}_1\mathtt{x})))))) \Downarrow \lceil\lceil\underline{\mathtt{n}},\underline{\mathtt{i}}^{\mathtt{N}}\rceil,\underline{\mathtt{o}}^{\mathtt{N}}\rceil$ which is almost what we are looking for. In order to conclude, it is sufficient to project the three numerals and suitably re-compose them.

The proof of the following theorem generalizes the above technique.

Theorem 6. $@\mathtt{wh}?^{\sigma}_{\tau}$ is definable in $\mathcal{S}$PCF$_\star$.

Proof. In the following, we show that, for each σ there is a term in $\mathcal{S}$PCF behaving as $@\mathtt{wh}?^{\sigma}_{\tau} : (\sigma \multimap \tau) \multimap \sigma \multimap \tau$, i.e. respecting its operational rule: $@\mathtt{wh}?^{\sigma}_{\tau}\,\mathtt{M}^{\sigma\multimap\tau}\,\mathtt{N}^{\sigma}\mathtt{P}_1\ldots\mathtt{P}_{\mathtt{k}} \Downarrow \lceil\underline{\mathtt{n}},\underline{\mathtt{m}}\rceil$ whenever $\mathtt{M}^{\sigma\multimap\tau}\mathtt{Sgl}^{(\sigma)}_{\mathtt{m}}\mathtt{P}_1\ldots\mathtt{P}_{\mathtt{k}} \Downarrow \underline{\mathtt{n}}$ and $\mathtt{Chk}^{(\sigma)}_{\mathtt{m}}\ \mathtt{N} \Downarrow \underline{0}$.

Let $\tau = \tau_1 \multimap \ldots \multimap \tau_k \multimap \iota$, the proof is by induction on σ.

Case $\sigma = \iota$. Let $\mathtt{L}$ be the following term:

$$\lambda\mathtt{f}^{\iota\multimap\tau}.\lambda\mathtt{x}^{\iota}.\lambda\mathtt{g}_1^{\tau_1}\ldots\mathtt{g}_{\mathtt{k}}^{\tau_{\mathtt{k}}}.\,\mathtt{which?}(\lambda\mathtt{h}^{\iota\multimap\iota}.\mathtt{f}\ (\mathtt{h}\ \mathtt{x})\ \mathtt{g}_1\ldots\mathtt{g}_{\mathtt{k}}).$$

Clearly, $\mathtt{L\ M\ N\ P}_1\ldots\mathtt{P}_{\mathtt{k}} \Downarrow \lceil\underline{\mathtt{n}},\underline{\mathtt{m}}\rceil$ if and only if $\mathtt{N} \Downarrow \underline{\mathtt{m}}$ and $\mathtt{M}\ \underline{\mathtt{m}}\ \mathtt{P}_1\ldots\mathtt{P}_{\mathtt{k}} \Downarrow \underline{\mathtt{n}}$ iff and only if $\mathtt{M}^{\sigma\multimap\tau}\mathtt{Sgl}^{(\sigma)}_{\mathtt{m}}\mathtt{P}_1\ldots\mathtt{P}_{\mathtt{k}}$ and $\mathtt{Chk}^{(\sigma)}_{\mathtt{m}}\ \mathtt{N} \Downarrow \underline{0}$. Therefore, $@\mathtt{wh}?^{\iota}_{\tau}$ is implemented by $\mathtt{L}$.

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

Case $\sigma = \sigma_1 \multimap \sigma_2$**.** Suppose $\sigma_2 = \mu_1 \multimap \cdots \multimap \mu_p \multimap \iota$ and, without loss of generality, assume $\mathtt{M}\,\mathtt{N}\,\mathtt{P}_1 \ldots \mathtt{P}_k \Downarrow \underline{n}$. Clearly $\mathtt{M}(\lambda \mathtt{z}_1 \ldots \mathtt{z}_p.\mathtt{I}(\mathtt{N}(\mathtt{Iz}_1)\ldots(\mathtt{Iz}_p)))\,\mathtt{P}_1 \ldots \mathtt{P}_k \Downarrow \underline{n}$, where the leftmost identity (typed $\iota \multimap \iota$) forwards the result produced by the use of $\mathtt{N}$ and the remaining copies of identities (typed $\mu_j \multimap \mu_j$) forward its j-th argument ($1 \leqslant j \leqslant p$). Let $\mathtt{L}$ be $(\lambda \mathtt{z}_1 \ldots \mathtt{z}_p.\mathtt{x}_0(\mathtt{N}(\mathtt{x}_1\mathtt{z}_1)\ldots(\mathtt{x}_p\mathtt{z}_p)))$ i.e. the above term where we replaced identities by variables $\mathtt{x}_j$, for $0 \leqslant j \leqslant p$.

We build a sequence of terms $\mathtt{L}_0, \mathtt{L}_p, \ldots, \mathtt{L}_1$ by abstracting one more free variable in each of them. Let $\mathtt{L}_0$ be $@\mathtt{wh}?^{\iota}_{\iota}(\lambda \mathtt{x}_0^{\iota\multimap\iota}.\mathtt{M}\,\mathtt{L}\,\mathtt{P}_1 \ldots \mathtt{P}_k)\,\mathtt{I}^{\iota\multimap\iota}$, thus the evaluation of $\mathtt{L}_0$ after substituting its free variables with (conveniently typed) identities gives back a pair $\lceil \underline{n}, \underline{m}_0 \rceil$.Let $\mathtt{L}_p$ be $@\mathtt{wh}?^{\mu_p}_{\mu_p}(\lambda \mathtt{x}_p^{\mu_p\multimap\mu_p}.\mathtt{L}_0)\mathtt{I}^{\mu_p\multimap\mu_p}$, let $\mathtt{L}_{p-1}$ be $@\mathtt{wh}?^{\mu_{p-1}}_{\mu_{p-1}}(\lambda \mathtt{x}_{p-1}^{\mu_{p-1}\multimap\mu_{p-1}}.\mathtt{L}_p)\mathtt{I}^{\mu_{p-1}\multimap\mu_{p-1}}$ and so on, until $\mathtt{L}_1$ is the closed term $@\mathtt{wh}?^{\mu_1}_{\mu_1}(\lambda \mathtt{x}_1^{\mu_1\multimap\mu_1}.\mathtt{L}_2)\mathtt{I}^{\mu_1\multimap\mu_1}$. It is easy to check that $\mathtt{L}_1 \Downarrow \lceil\lceil\lceil\lceil \underline{n}, \underline{m}_0\rceil, \underline{m}_p\rceil \ldots, \underline{m}_2\rceil, \underline{m}_1\rceil$ such that $\mathtt{N}\,\mathtt{Sgl}^{(\mu_1)}_{\mathtt{m}_1} \ldots \mathtt{Sgl}^{(\mu_p)}_{\mathtt{m}_p} \Downarrow \underline{m}_0$ and, moreover, $\mathtt{M}\,\mathtt{Sgl}^{(\sigma_2)}_{\mathtt{r}}\,\mathtt{P}_1 \ldots \mathtt{P}_k \Downarrow \underline{n}$ where $\mathtt{r} = \lceil \underline{m}_1, \lceil \underline{m}_2, \ldots \lceil \underline{m}_p, \underline{m}_0\rceil\rceil\rceil$. It is boring, but easy, to find a term $\mathtt{R} : \iota \multimap \iota$ such that $\mathtt{R}\lceil\lceil\lceil\lceil \underline{n}, \underline{m}_0\rceil, \underline{m}_p\rceil \ldots, \underline{m}_2\rceil, \underline{m}_1\rceil \Downarrow \lceil \underline{n}, \lceil \underline{m}_1, \lceil \underline{m}_2, \ldots \lceil \underline{m}_p, \underline{m}_0\rceil\rceil\rceil\rceil$.

Explicitly rewriting $\mathtt{L}_1 = @\mathtt{wh}?^{\mu_1}_{\mu_1}(\lambda \mathtt{x}_1^{\mu_1\multimap\mu_1}. \ldots @\mathtt{wh}?^{\iota}_{\iota}(\lambda \mathtt{x}_0^{\iota\multimap\iota}.\mathtt{M}\,\mathtt{L}\,\mathtt{P}_1 \ldots \mathtt{P}_k)\,\mathtt{I}^{\iota\multimap\iota} \ldots)\mathtt{I}^{\mu_1\multimap\mu_1}$ where $\mathtt{L}$ is $(\lambda \mathtt{z}_1 \ldots \mathtt{z}_p.\mathtt{x}_0(\mathtt{N}(\mathtt{x}_1\mathtt{z}_1)\ldots(\mathtt{x}_p\mathtt{z}_p)))$. We replace $\mathtt{M}, \mathtt{N}, \mathtt{P}_1, \ldots, \mathtt{P}_k$ by using some fresh variables, let $\mathtt{W}$ be the term

$$@\mathtt{wh}?^{\mu_1}_{\mu_1}(\lambda \mathtt{x}_1^{\mu_1\multimap\mu_1}. \ldots @\mathtt{wh}?^{\iota}_{\iota}(\lambda \mathtt{x}_0^{\iota\multimap\iota}.\mathtt{f}^{\sigma\multimap\tau}\,\mathtt{L}'\,\mathtt{y}_1^{\tau_1} \ldots \mathtt{y}_k^{\tau_k})\,\mathtt{I}^{\iota\multimap\iota} \ldots)\mathtt{I}^{\mu_1\multimap\mu_1}$$

where $\mathtt{L}'$ is $(\lambda \mathtt{z}_1 \ldots \mathtt{z}_p.\mathtt{x}_0(\mathtt{g}^{\sigma}(\mathtt{x}_1\mathtt{z}_1)\ldots(\mathtt{x}_p\mathtt{z}_p)))$. The term behaving as $@\mathtt{wh}?^{\iota}_{\tau}$ is defined as $\lambda \mathtt{f}^{\sigma\multimap\tau}\mathtt{g}^{\sigma}\mathtt{y}_1^{\tau_1} \ldots \mathtt{y}_k^{\tau_k}.\mathtt{RW}$. □

As an example of the use of the ℓet-ℓor operator, we show how to use it in order to program in SℓPCF$_\star$ the operator $\mathtt{G\eth r}$ defined in Paolini and Piccolo (2008). Indeed, we show how to define $\mathtt{G\eth r}^{\underline{k}_1,\underline{k}_2,\underline{k}_3,\underline{k}_4}$ for all $k_1, k_2, k_3, k_4 \in \mathbb{N}$. In order to proceed in a modular way, we introduce a notation useful to consider the restrictions of $\mathtt{G\eth r}^{\mathtt{L}}$ that use only a subset of the four rules. Precisely, a $\bullet$ in the list $\mathtt{L}$ is used to denote the operator obtained omitting the corresponding rules. For instance, $\mathtt{G\eth r}^{\mathtt{L}_0}$ where $\mathtt{L}_0 = \underline{0}, \underline{2}, \bullet, \bullet$ is defined as

$$\lambda \mathtt{f}_1^{\iota\multimap\iota}\mathtt{f}_2^{\iota\multimap\iota}\mathtt{f}_3^{\iota\multimap\iota}.\\ \begin{pmatrix}\lambda \mathtt{w}\ell\mathtt{if}\ \mathtt{w} \doteq \underline{0}\big(\ell\mathtt{if}\ (\mathtt{f}_2\underline{1} \doteq \underline{0}\ \mathtt{and}\ \mathtt{f}_3\underline{0} \doteq \underline{1})\ \underline{0}\ \Omega^{\iota}\big)\\ \quad\big(\ell\mathtt{if}\ \mathtt{w} \doteq \underline{1}(\ell\mathtt{if}\ (\mathtt{f}_2\underline{0} \doteq \underline{0}\ \mathtt{and}\ \mathtt{f}_3\underline{1} \doteq \underline{0})\ \underline{2}\ \Omega^{\iota})\ \Omega^{\iota}\big)\end{pmatrix}(\mathtt{f}_1\underline{0})$$

All the $\mathtt{G\eth r}^{\mathtt{L}}$ defined by just two rules, i.e. the ones with $\mathtt{L}$ containing two occurrences of $\bullet$, can be defined likewise. Thus, $\mathtt{G\eth r}^{\mathtt{L}_1}$ with parameter $\mathtt{L}_1 = \underline{1}, \bullet, \underline{0}, \underline{0}$ can be defined using the ℓet-ℓor operator as

$$\lambda \mathtt{f}_1\mathtt{f}_2\mathtt{f}_3.\,\underline{\ell\mathtt{et}}\ \ \mathtt{g}_1 = \mathtt{f}_1, \mathtt{g}_2 = \mathtt{f}_2, \mathtt{g}_3 = \mathtt{f}_3\ \underline{\mathtt{in}\ell\mathtt{or}}\\ (\mathtt{G\eth r}^{\underline{0},\bullet,\underline{1},\bullet}\ \mathtt{g}_1\mathtt{g}_2\mathtt{g}_3)(\mathtt{G\eth r}^{\underline{2},\bullet,\bullet,\underline{0}}\ \mathtt{g}_1\mathtt{g}_2\mathtt{g}_3)(\mathtt{G\eth r}^{\bullet,\bullet,\underline{0},\underline{1}}\ \mathtt{g}_1\mathtt{g}_2\mathtt{g}_3)$$

and $\mathtt{G\eth r}^{\mathtt{L}_2}$ with parameter $\mathtt{L}_2 = \bullet, \underline{0}, \underline{3}, \underline{4}$ can be defined as

$$\lambda \mathtt{f}_1\mathtt{f}_2\mathtt{f}_3.\,\underline{\ell\mathtt{et}}\ \ \mathtt{g}_1 = \mathtt{f}_1, \mathtt{g}_2 = \mathtt{f}_2, \mathtt{g}_3 = \mathtt{f}_3\ \underline{\mathtt{in}\ell\mathtt{or}}\\ (\mathtt{G\eth r}^{\bullet,\underline{0},\underline{4},\bullet}\ \mathtt{g}_1\mathtt{g}_2\mathtt{g}_3)(\mathtt{G\eth r}^{\bullet,\underline{1},\bullet,\underline{0}}\ \mathtt{g}_1\mathtt{g}_2\mathtt{g}_3)(\mathtt{G\eth r}^{\bullet,\bullet,\underline{0},\underline{5}}\ \mathtt{g}_1\mathtt{g}_2\mathtt{g}_3).$$

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

Finally, $\mathtt{G\eth r}^{\mathtt{L}_3}$ with parameter $\mathtt{L}_3 = \underline{0}, \underline{1}, \underline{2}, \underline{3}$ can be defined as

$$\lambda \mathtt{f}_1^{\iota\multimap\iota}\mathtt{f}_2^{\iota\multimap\iota}\mathtt{f}_3^{\iota\multimap\iota}.\,\underline{\ell\mathtt{et}}\ \mathtt{g}_1 = \mathtt{f}_1, \mathtt{g}_2 = \mathtt{f}_2, \mathtt{g}_3 = \mathtt{f}_3 \\ \underline{\mathtt{in}\ell\mathtt{or}}(\mathtt{G\eth r}^{\mathtt{L}_0}\ \mathtt{g}_1\mathtt{g}_2\mathtt{g}_3)(\mathtt{G\eth r}^{\mathtt{L}_1}\ \mathtt{g}_1\mathtt{g}_2\mathtt{g}_3)(\mathtt{G\eth r}^{\mathtt{L}_2}\ \mathtt{g}_1\mathtt{g}_2\mathtt{g}_3).$$

For every parameter $\mathtt{L}$, all the $\mathtt{G\eth r}^{\mathtt{L}}$ can be built analogously. As expected, the ℓet-ℓor operator is fundamental for the above construction.

5. Finite definability and full abstraction

In this section, we prove that the linear model $\mathcal{L}$ is *fully abstract* with respect to SℓPCF$_\star$. This result relies on the *completeness* of the linear interpretation with respect to the operational semantics and on the *definability* of all the finite cliques by means of SℓPCF$_\star$ terms. In particular, we start by proving the completeness with respect to the fix-point equivalence then, in next section (Proposition 4 and Theorem 9), we then prove that the fix-point and the standard operational equivalences coincide. From this, full abstraction holds also with respect to the latter.

Finite definability asserts that all finite cliques can be defined by means of SℓPCF$_\star$ terms. We follow a standard scheme for proofs of this kind, e.g. Plotkin (1977) and Paolini (2006). Non-trivial uses of @wh? and ℓet-ℓor constructors are needed in inductive steps.

Definitiona 18. Let u be a finite clique of a coherence space in $\mathcal{L}$. A term $\mathtt{M}$ *defines* u if and only if $[\![\mathtt{M}]\!] = u$. The class of closed terms having u as interpretation is denoted by $\lceil u \rfloor$, i.e. $\lceil u \rfloor = \{\mathtt{M} \mid [\![\mathtt{M}]\!] = u\}$. Moreover, $\lceil a^1, \ldots, a^k \rfloor$ is used to abbreviate $\lceil \{a^1, \ldots, a^k\} \rfloor$ and, $\lceil u \rfloor = \mathtt{M}$ is used to abbreviate $\mathtt{M} \in \lceil u \rfloor$.

By abuse of notation, in the following we denote $\lceil u \rfloor$ a term $\mathtt{M}$ such that $\mathtt{M} = \lceil u \rfloor$. In the following, if $i \in \{1, 2\}$ and $k \leqslant 0$ we denote $\pi_i^k(\underline{\mathtt{n}})$ to be π_i applied k times to $\underline{\mathtt{n}}$. To prove definability, we use the following auxiliary lemma.

Lemma 10. Let $(a_0, \ldots, a_n, a), (b_0, \ldots, b_n, b) \in |[\![\tau_0 \multimap \cdots \multimap \tau_n \multimap \iota]\!]|$. Then:

1. $(a_0, \ldots, a_n, a)\ \Pi\ (b_0, \ldots, b_n, b)$ if and only if $\exists k \leqslant n\colon a_k\ \Upsilon\ b_k$;
2. $(a_0, \ldots, a_n, a)\ \Sigma\ (b_0, \ldots, b_n, b)$ if and only if $\forall k \leqslant n\colon a_k\ \Xi\ b_k$.

Some measures are needed in the next theorem: the *cardinality* of a clique u is denoted $\|u\|$; the RK of a type is inductively defined as: $\mathrm{RK}(\iota) = 1$; $\mathrm{RK}(\sigma \multimap \tau) = \mathrm{RK}(\sigma) + \mathrm{RK}(\tau)$.

Theorem 7 (finite definability). If $u \in \mathcal{C}l_{fin}([\![\sigma]\!])$ then there exists a closed $\mathtt{M} \in$ SℓPCF$_\star$ such that $\mathtt{M} = \lceil \mathtt{u} \rfloor$.

Proof. Let $\sigma = \tau_1 \multimap \cdots \multimap \tau_k \multimap \iota$ for some $k \geqslant 0$. The proof is by induction on the triple $\langle \mathrm{RK}(\sigma), k, \|u\| \rangle$ ordered in a lexicographic way. The cases $\mathrm{RK}(\sigma) = 1$ and $\mathrm{RK}(\sigma) = 2$ are easy.

- Consider $\mathrm{RK}(\sigma) = 1$, then $\sigma = \iota$ and $[\![\sigma]\!] = \mathbf{N}$. Thus, Ω^ι and numerals define all possible finite cliques, since $\mathcal{C}l_{fin}(\mathbf{N}) = \{\varnothing\} \cup \{\{n\} / n \in |\mathbf{N}|\}$.

- Consider $\mathrm{RK}(\sigma) = 2$, then $\sigma = \iota \multimap \iota$.

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

— If $\|u\| = 0$ then $u = \varnothing$ is defined by $\Omega^{\iota \multimap \iota}$.
— If $\|u\| \geqslant 1$, then $u = u' \cup \{(a,b)\}$ for $a, b \in \mathbb{N}$. By induction hypothesis we have $\lceil u' \rfloor$, hence $\lceil u \rfloor = \lambda \mathtt{z}.\ell\mathtt{if}\ (\mathtt{z} \doteq \lceil a \rfloor)\ \lceil b \rfloor\ (\lceil u' \rfloor \mathtt{z})$.

- Consider $\mathrm{RK}(\sigma) \geqslant 3$ and $k = 1$, then $\sigma = \tau \multimap \iota$ with $\tau = \sigma_1 \multimap \cdots \sigma_r \multimap \iota$.

— If $\|u\| = 0$ then $u = \varnothing$ is defined by $\Omega^{\tau \multimap \iota}$.
— If $\|u\| = 1$, then $u = \{(a,b)\}$ for $a \in [\![\tau]\!]$ and $b \in \mathbb{N}$. Suppose $a = (a_1, \ldots, a_r, c)$ where $a_i \in [\![\sigma_i]\!]$ and $c \in \mathbb{N}$ $(1 \leqslant i \leqslant r)$. By induction hypothesis, we have $\lceil a_1 \rfloor, \ldots, \lceil a_r \rfloor, \lceil c \rfloor$ and $\lceil b \rfloor$, hence: $\lceil u \rfloor = \lambda \mathtt{f}.\ell\mathtt{if}\ (\mathtt{f} \lceil a_1 \rfloor \cdots \lceil a_r \rfloor \doteq \lceil c \rfloor)\ \lceil b \rfloor\ \Omega^{\iota}$.
— If $\|u\| > 1$, then $u = \{(a^0, b^0), \ldots, (a^m, b^m)\}$ for $a^i \in [\![\tau]\!]$ and $b^i \in \mathbb{N}$ $(0 \leqslant i \leqslant m)$. Suppose $a^i = (a^i_1, \ldots, a^i_r, c^i)$ where $a^i_j \in [\![\sigma_j]\!]$ and $c^i \in \mathbb{N}$ $(1 \leqslant j \leqslant r)$. By Lemma 10.1, we have $a^h \smile a^k$ $(0 \leqslant h \neq k \leqslant m)$, so by Lemma 10.2 $a^h_j \supset\!\!\!\subset a^k_j$ $(1 \leqslant j \leqslant r)$. Hence, take $v_j = \{a^i_j \mid 1 \leqslant i \leqslant m\}$. Moreover, for sake of simplicity, we write just $\underline{\mathtt{a}^{\mathtt{i}}_{\mathtt{j}}}$ in place of $(\!|a^i_j|\!)$, i.e. the natural number encoding a^i_j.
By induction hypothesis, we have $\lceil v_j \rfloor$, $\lceil c^i \rfloor$ for every $0 \leqslant i \leqslant m$, $1 \leqslant j \leqslant r$ and $1 \leqslant k \leqslant s$. So, we can define:

$$\begin{aligned}
&\lceil u \rfloor = \lambda \mathtt{F}^{\tau}.\Big(\lambda \mathtt{z}^{\mathtt{i}}.\\
&\ell\mathtt{if}\ (\boldsymbol{\pi}^r_1(z) \doteq \underline{\mathtt{c}^0}\ \mathtt{and}\ \boldsymbol{\pi}^{r-1}_1(\boldsymbol{\pi}_2(z)) \doteq \underline{\mathtt{a}^0_1}\ \ldots\ \boldsymbol{\pi}_2(z) \doteq \underline{\mathtt{a}^0_{\mathtt{r}}})\ \underline{\mathtt{b}^0}\\
&\ell\mathtt{if}\ (\boldsymbol{\pi}^r_1(z) \doteq \underline{\mathtt{c}^1}\ \mathtt{and}\ \boldsymbol{\pi}^{r-1}_1(\boldsymbol{\pi}_2(z)) \doteq \underline{\mathtt{a}^1_1}\ \ldots\ \boldsymbol{\pi}_2(z) \doteq \underline{\mathtt{a}^1_{\mathtt{r}}})\ \underline{\mathtt{b}^1}\\
&\vdots\\
&\ell\mathtt{if}\ (\boldsymbol{\pi}^r_1(z) \doteq \underline{\mathtt{c}^{\mathtt{m}}}\ \mathtt{and}\ \boldsymbol{\pi}^{r-1}_1(\boldsymbol{\pi}_2(z)) \doteq \underline{\mathtt{a}^{\mathtt{m}}_1}\ \ldots\ \boldsymbol{\pi}_2(z) \doteq \underline{\mathtt{a}^{\mathtt{m}}_{\mathtt{r}}})\ \underline{\mathtt{b}^{\mathtt{m}}}\ \Big)\\
&\big(@\mathtt{wh}?^{\sigma_{\mathtt{r}}}_{\iota}(\ldots(@\mathtt{wh}?^{\sigma_1}_{\sigma_2 \multimap \cdots \sigma_{\mathtt{r}} \multimap \iota}\mathtt{F}\lceil \mathtt{v}_1 \rfloor)\ldots)\lceil \mathtt{v}_{\mathtt{r}} \rfloor\big)
\end{aligned}$$

- Consider $\mathrm{RK}(\sigma) \geqslant 3$ and $k > 1$.

— If $\|u\| = 0$, then $u = \varnothing$ is defined by $\Omega^{\tau_1 \multimap \cdots \multimap \tau_k \multimap \iota}$.
— If $\|u\| = 1$, then $u = \{(a_1, \ldots a_k, b)\}$ where $a_i \in [\![\tau_i]\!]$ $(1 \leqslant i \leqslant k)$ and $b \in \mathbb{N}$. Thus,

$$\lceil u \rfloor = \lambda \mathtt{f}_1 \ldots \mathtt{f}_k.\ell\mathtt{if}\ ((\mathtt{Chk}^{(\sigma_1)}_{\underline{\mathtt{a}_1}}\mathtt{f}_1)\ \mathtt{and} \cdots \mathtt{and}\ (\mathtt{Chk}^{(\sigma_k)}_{\underline{\mathtt{a}_k}}\mathtt{f}_{\mathtt{k}}))\ \lceil \mathtt{b} \rfloor\ \Omega^{\iota}.$$

— If $\|u\| = 2$ then $u = \{(a^1_1, \ldots, a^1_k, b^1), (a^2_1, \ldots, a^2_k, b^2)\}$. We know that there is $i \in [1,k]$ such that $a^1_i \curlyvee a^2_i$. If $\tau_i = \iota$ then a^1_i and a^2_i are two different numbers: thus the term defining u is the following

$$\begin{aligned}
\lceil u \rfloor = \lambda \mathtt{f}_1 \ldots \mathtt{f}_{\mathtt{k}}.\ell\mathtt{if}\ (\mathtt{f}_{\mathtt{i}} \doteq \lceil a^1_i \rfloor)\Big(&\ell\mathtt{if}\ (\mathtt{Chk}^{(\sigma_1)}_{\underline{\mathtt{a}^1_1}}\mathtt{f}_1)\ \mathtt{and} \cdots\\
&\mathtt{and}\ (\mathtt{Chk}^{(\sigma_k)}_{\underline{\mathtt{a}^1_k}}\mathtt{f}_{\mathtt{k}})\ \lceil b^1 \rfloor\ \Omega\Big)\Big(\ell\mathtt{if}\big((\mathtt{f}_{\mathtt{i}} \doteq \lceil a^2_i \rfloor)\ \mathtt{and}\\
&(\mathtt{Chk}^{(\sigma_1)}_{\underline{\mathtt{a}^2_1}}\mathtt{f}_1)\ \mathtt{and} \cdots \mathtt{and}\ ((\mathtt{Chk}^{(\sigma_1)}_{\underline{\mathtt{a}^2_k}}\mathtt{f}_{\mathtt{k}}))\ \lceil b^2 \rfloor\ \Omega\Big).
\end{aligned}$$

—If $\tau = v_1 \multimap \cdots v_l \multimap \iota$ then we have that $a^1_i = (e^1_1, \ldots, e^1_l, c^1)$ and $a^2_i = (e^2_1, \ldots, e^2_l, c^2)$. Since $a^1_i \curlyvee a^2_i$, we have that for all $j \in [1,l]$ the sets $\{e^1_j, e^2_j\}$ are cliques of lower rank. Thus, by inductive hypothesis we have terms $\mathtt{N}_{\mathtt{j}}$ defining them. Thus, the term defining

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

u is the following

$$\lceil u\rfloor = \lambda \mathtt{f}_1.\ldots\mathtt{f}_k.\Big(\lambda \mathtt{x}^{\iota}.\ell\mathtt{if}\ \Big(\boldsymbol{\pi}_1^1\mathtt{x} \doteq \underline{\mathtt{c}^1}\ \mathtt{and}\ \boldsymbol{\pi}_1^{1-1}(\boldsymbol{\pi}_2\mathtt{x}) \doteq \underline{\mathtt{e}_1^1}$$
$$\mathtt{and}\ldots\boldsymbol{\pi}_2\mathtt{x} \doteq \underline{e_l^1}\ \Big)\Big(\ell\mathtt{if}\big(\mathtt{Chk}^{(\tau_1)}_{\underline{\mathtt{a}_1^1}}\mathtt{f}_1\ \mathtt{and}\ \ldots$$
$$\mathtt{Chk}^{(\tau_{i-1})}_{\underline{\mathtt{a}_{i-1}^1}}\mathtt{f}_{i-1}\ \mathtt{and}\ \mathtt{Chk}^{(\tau_{i+1})}_{\underline{\mathtt{a}_{i+1}^1}}\mathtt{f}_{i+1}\ \mathtt{and}\ \ldots\mathtt{Chk}^{(\tau_k)}_{\underline{\mathtt{a}_k^1}}\mathtt{f}_k\big)\ \lceil b^1\rfloor\ \Omega\Big)$$
$$\Big(\ell\mathtt{if}\big(\boldsymbol{\pi}_1^l\mathtt{x} \doteq \underline{c^2}\ \mathtt{and}\ \boldsymbol{\pi}_1^{l-1}(\boldsymbol{\pi}_2\mathtt{x}) \doteq \underline{e_1^2}\ \mathtt{and}\ldots\boldsymbol{\pi}_2\mathtt{x} \doteq \underline{e_l^2}\ \mathtt{and}$$
$$\mathtt{Chk}^{(\tau_1)}_{\underline{\mathtt{a}_1^2}}\mathtt{f}_1\ \mathtt{and}\ \ldots\mathtt{Chk}^{(\tau_{i-1})}_{\underline{\mathtt{a}_{i-1}^2}}\mathtt{f}_{i-1}\ \mathtt{and}$$
$$\mathtt{Chk}^{(\tau_{i+1})}_{\underline{\mathtt{a}_{i+1}^2}}\mathtt{f}_{i+1}\ \mathtt{and}\ldots\mathtt{Chk}^{(\tau_k)}_{\underline{\mathtt{a}_k^2}}\mathtt{f}_k\big)\ \lceil b^2\rfloor\ \Omega\Big)$$
$$\Big)\Big(@\mathtt{wh}?^{\nu_l}_{\iota}(\ldots(@\mathtt{wh}?^{\nu_1}_{\nu_2\multimap\ldots\nu_l\multimap\iota}(\mathtt{f}_i)(\mathtt{N}_1))\ldots)\mathtt{N}_1\Big).$$

— If $\|u\| > 2$, then $u = \{d^1,\ldots,d^m\}$ where $d^j = (a_1^j,\ldots,a_k^j,b^j)$, $a_i^j \in [\![\tau_i]\!]$ and $b^j \in \mathbb{N}$ $(1 \leqslant j \leqslant m,\ 1 \leqslant i \leqslant k)$. We denote by $d^j[b]$ the token $(a_1^j,\ldots,a_k^j,b)$. By Lemma 10.1 there exists $1 \leqslant h \leqslant k$ such that $a_h^1 \frown a_h^2$, so we can build the following finite cliques:

$$w_1 = \{d^1[0], d^2[b^2+1]\ \}$$
$$w_2 = \{d^1[b^1+1]\} \cup \{d^r[0] \mid 2 < r \leqslant m\ \}$$
$$w_3 = \{d^2[0]\} \cup \{d^r[b^r+1] \mid 2 < r \leqslant m\ \}.$$

Note that $\|w_s\| < \|u\|$ for $s = 1,2,3$. So, by induction hypothesis we have $\lceil w_1\rfloor, \lceil w_2\rfloor$ and $\lceil w_3\rfloor$. Hence, we conclude the proof by letting $\lceil u\rfloor$ be

$$\lambda \mathtt{f}_1\ldots\mathtt{f}_k.\underline{\ell\mathtt{et}}\ \mathtt{g}_1 = \mathtt{f}_1,\ldots,\mathtt{g}_k = \mathtt{f}_k\ \ \underline{\mathtt{in}\ell\mathtt{or}}\ \ \big(\lceil w_1\rfloor \mathtt{g}_1\cdots\mathtt{g}_k\big)\big(\lceil w_2\rfloor \mathtt{g}_1\cdots\mathtt{g}_k\big)\big(\lceil w_3\rfloor \mathtt{g}_1\cdots\mathtt{g}_k\big).$$

□

The definability of finite cliques is the key ingredient to extend the stable closed completeness (i.e. Theorem 4) to all the terms of $\mathcal{S}\ell$PCF$_\star$ as follows.

Theorem 8 (completeness). If $\mathtt{M} \sim_\sigma \mathtt{N}$ then $[\![\mathtt{M}]\!]\rho = [\![\mathtt{N}]\!]\rho$, for all $\rho \in \mathrm{Env}$.

Proof. Let $\Gamma \vdash \mathtt{M},\mathtt{N} : \sigma$ with $\Gamma\restriction \mathrm{S} = \{\boldsymbol{f}_1^{\tau_1},\ldots,\boldsymbol{f}_n^{\tau_n}\}$ and $\Gamma\restriction\ell, \Gamma\restriction\iota = \{\mathtt{x}_1{}^{\sigma_1},\ldots,\mathtt{x}_m{}^{\sigma_m}\}$. Assume that there exists ρ such that $[\![\mathtt{M}]\!]\rho \neq [\![\mathtt{N}]\!]\rho$. By the Lemma 8, there exists a closed term $\mathtt{P}^{\sigma\multimap\iota}$ such that $\mathcal{F}([\![\mathtt{P}]\!])([\![\mathtt{M}]\!]\rho) \neq \mathcal{F}([\![\mathtt{P}]\!])([\![\mathtt{N}]\!]\rho)$. By the Theorem 7, for all $\boldsymbol{f}_i$ in $\Gamma\restriction\mathrm{S}$ there is a term $\mathtt{P}_i = \lceil\rho(\boldsymbol{f}_i)\rfloor$ and for all $\mathtt{x}_i$ in $\Gamma\restriction\ell, \Gamma\restriction\iota$ there is a term $\mathtt{N}_i = \lceil\rho(\mathtt{x}_i)\rfloor$. So, we can build $\mathtt{C} = \mathtt{P}(\lambda \mathtt{x}_1^{\sigma_1}\ldots\mathtt{x}_m^{\sigma_m}.[\cdot^{\sigma}]\mathtt{N}_1\cdots\mathtt{N}_m)$. Without loss of generality, let us assume $\mathcal{F}([\![\mathtt{P}]\!])([\![\mathtt{M}]\!]\rho) = \{k\}$. By adequacy we have $\mathtt{C}[\mathtt{M}[\mathtt{P}_1/\boldsymbol{f}_1,\ldots,\mathtt{P}_n/\boldsymbol{f}_n]] \Downarrow \underline{\mathtt{k}}$ but $\mathtt{C}[\mathtt{N}[\mathtt{P}_1/\boldsymbol{f}_1,\ldots,\mathtt{P}_n/\boldsymbol{f}_n]] \not\Downarrow \underline{\mathtt{k}}$. This concludes the proof. □

By soundness and completeness the full abstraction follows.

Corollary 5 (full abstraction). $\mathtt{M} \sim_\sigma \mathtt{N}$ if and only if $[\![\mathtt{M}]\!]\rho = [\![\mathtt{N}]\!]\rho$, for all $\rho \in \mathrm{Env}$.

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

6. Coincidence of operational equivalences

In this section, we prove the coincidence of the standard operational equivalence (i.e. $\approx_\sigma$, see Definition 4) and the fix-point operational equivalence (i.e. $\sim_\sigma$, see Definition 5). Therefore, the full abstraction (Corollary 5) holds also for the standard operational equivalence and a *compositional* theory of program equivalence can be effectively defined.

One direction follows easily by the fact that the fix-point equivalence coincides with the denotational equivalence by Corollary 5 and by the correctness of the denotational semantics w.r.t the standard operational equivalence.

Proposition 4. Let $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathscr{S}\mathrm{PCF}_\star$. If $\mathtt{M} \sim_\sigma \mathtt{N}$ then $M \approx_\sigma N$.

Proof. $[\![\mathtt{M}]\!] = [\![\mathtt{N}]\!]$, by Theorem 8. Thus we conclude, by Theorem 5. □

The opposite direction is more difficult and it requires semantic reasoning. First, we prove an auxiliary result (Corollary 6) that claims that in a coherence space $X \in \mathcal{L}$ (i.e. in our type structure) different from $\mathbf{N}$, finite cliques are never maximal w.r.t. set-theoretical inclusion[†]. Then, we use this fact, together with adequacy (Theorem 4) and finite definability (Theorem 7) to prove the coincidence-result in the ground case (Lemma 14), which implies the general result (Theorem 9).

In the next lemmas we show that, if $X \in \mathcal{L}$ is coherence space different from $\mathbf{N}$ then, a finite clique of X is never maximal w.r.t. set-theoretical inclusion.

Lemma 11. Let x be a non-empty finite set of tokens in $|[\![\sigma]\!]|$ satisfying

$$\exists a \in x,\ \forall b \in x,\ a\ \Sigma\ b. \tag{2}$$

Then $\exists a' \notin x,\ \forall b \in x,\ a'\ \Upsilon\ b$.

Proof. Remark that x is not required to be a clique. The proof is by induction on the structure of σ. The case $\sigma = \iota$ is immediate: since x is finite, it suffices to choose a number in the set $\mathbb{N} \setminus x$ to obtain the result. For the inductive case $\sigma = \sigma_1 \multimap \sigma_2$, let $a = (a_1, a_2) \in x$ be an element satisfying (2), viz. $\forall (b_1, b_2) \in x,\ a_1\ \Xi\ b_1 \wedge a_2\ \Sigma\ b_2$. So it is possible to build the set $x_2 = \{b_2 \mid (b_1, b_2) \in x \textit{ and } b_2\ \Sigma\ a_2\} = \{b_2 \mid (b_1, b_2) \in x\}$. By inductive hypothesis there is $a_2' \notin x_2$ such that for all $b_2 \in x_2$ we have that $a_2'\ \smile\ b_2$. Let $a' = (a_1, a_2')$; it is not difficult to check that for all $b \in x$ we have that $a'\ \Upsilon\ b$. □

Lemma 12. Let x be a non-empty finite set of tokens in $|[\![\sigma]\!]|$ ($\sigma \neq \iota$) satisfying

$$\exists a \in x,\ \forall b \in x,\ a\ \Xi\ b. \tag{3}$$

Then $\exists a' \notin x,\ \forall b \in x,\ a'\ \Pi\ b$.

Proof. Remark that x is not required to be a clique, albeit all finite cliques satisfy (3). The proof is by induction on the structure of σ. The base case $\sigma = \iota$ is vacuously true. For the inductive case $\sigma = \sigma_1 \multimap \sigma_2$, let $a = (a_1, a_2) \in x$ be an element satisfying (3). Let $x_1 = \{b_1 \mid (b_1, b_2) \in x \textit{ and } b_1\ \Sigma\ a_1\}$ and $x_2 = \{b_2 \mid (b_1, b_2) \in x \textit{ and } b_2\ \Xi\ a_2\}$. It is easy to

[†] Observe that this fact is not true in the general case of stable functions: for example in the coherence space $!\mathbf{N} \multimap \mathbf{N}$, the finite clique $\{(\varnothing, 0)\}$ is maximal.

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

see that $\{b_2 | (b_1, b_2) \in x \text{ and } b_1 \mathrel{\Xi} a_1\} \subseteq x_2$, because (3). By Lemma 11, there is $a_1' \notin x_1$ such that for all $c_1 \in x_1$ we have $a_1' \frown c_1$. There are two cases.

1. If $\sigma_2 = \iota$ then $x_1 = \{b_1 \mid (b_1, b_2) \in x\}$ and we can set $a' = (a_1', k)$ where k is a randomly chosen natural number.
2. If $\sigma_2 \neq \iota$ then we apply the inductive hypothesis: there is $a_2' \notin x_2$ such that for all $c_2 \in x_2$ we have $a_2' \mathrel{\Pi} c_2$. We set $a' = (a_1', a_2')$.

In both cases it is not difficult to check that for all $b \in x$ we have that $a' \mathrel{\Pi} b$. □

Corollary 6. Let $x \in \mathcal{C}l_{fin}(|[\![\sigma]\!]|)$ with $\sigma \neq \iota$. Then, there is $a \in |[\![\sigma]\!]|$ such that $a \notin x$ and $x \cup \{a\} \in \mathcal{C}l(|[\![\sigma]\!]|)$.

Given an environment ρ, a term $\mathtt{M}$, a stable variable $\boldsymbol{F}^{\sigma}$ and an infinite clique $x \in \mathcal{C}l(|[\![\sigma]\!]|)$, with a slight abuse of notation in the sequel we write $[\![\mathtt{M}]\!]\rho[\boldsymbol{F} := x]$ to denote $\bigcup_{y \subseteq_{fin} x} [\![\mathtt{M}]\!]\rho[\boldsymbol{F} := y]$.

Lemma 13. If $\mathtt{M}^{\sigma}[\mathtt{N}/\boldsymbol{F}^{\tau}] \in \mathcal{S}\mathrm{PCF}_{\star}$ then $[\![\mathtt{M}^{\sigma}[\mathtt{N}/\boldsymbol{F}^{\tau}]]\!]\rho = \bigcup_{x \subseteq_{fin} |[\![\mathtt{N}]\!]_{\rho}} [\![\mathtt{M}]\!]\rho[\boldsymbol{F}^{\tau} := x]$.

Proof. The proof is straightforward by induction on $\mathtt{M}^{\sigma}$. □

Now, we show that by using fixpoints, it is possible to build contexts that allow us to discriminate as much as we can do by using substitutions.

Lemma 14. Let $\Gamma \vdash \mathtt{M}, \mathtt{N} : \iota$ with $\Gamma\restriction_{\ell} = \varnothing$. If $\mathtt{M} \approx_{\iota} \mathtt{N}$ then $\mathtt{M} \sim_{\iota} \mathtt{N}$.

Proof. We prove the contrapositive. Let $\mathrm{SFV}(\mathtt{M}), \mathrm{SFV}(\mathtt{N}) \subseteq \{\vec{\boldsymbol{F}}\}$ and let $\mathtt{C}[\,]$ be a context and $\vec{\mathtt{P}}$ be closed terms such that $\mathtt{C}[\mathtt{M}[\vec{\mathtt{P}}/\vec{\boldsymbol{F}}]], \mathtt{C}[\mathtt{N}[\vec{\mathtt{P}}/\vec{\boldsymbol{F}}]] \in \mathrm{P}$ $\mathtt{C}[\mathtt{M}[\vec{\mathtt{P}}/\vec{\boldsymbol{F}}]] \Downarrow \underline{\mathtt{n}}$ and $\mathtt{C}[\mathtt{N}[\vec{\mathtt{P}}/\vec{\boldsymbol{F}}]] \not\Downarrow \underline{\mathtt{n}}$. By induction on $|\vec{\boldsymbol{F}}|$ that there is a $\mathtt{C}'[\,]$ such that $\mathtt{C}'[\mathtt{M}] \Downarrow \underline{\mathtt{n}}'$ and $\mathtt{C}'[\mathtt{M}] \not\Downarrow \underline{\mathtt{n}}'$.

Base case. Namely the two terms have no free occurrence of stable variables, thus we can take $\mathtt{C}'[\,] = \mathtt{C}[\,]$.

Inductive case. The fix-point (in)equivalence implies that there is a context $\mathtt{C}[\,]$ and there are closed terms $\mathtt{P}_1, \ldots, \mathtt{P}_{m+1}$, such that $\mathtt{C}[\mathtt{M}[\mathtt{P}_1/\boldsymbol{F}_1^{\sigma_1}, \ldots, \mathtt{P}_{m+1}/\boldsymbol{F}_{m+1}^{\sigma_{m+1}}]] \Downarrow \underline{\mathtt{n}}$ but $\mathtt{C}[\mathtt{N}[\mathtt{P}_1/\boldsymbol{F}_1^{\sigma_1}, \ldots, \mathtt{P}_{m+1}/\boldsymbol{F}_{m+1}^{\sigma_{m+1}}]] \not\Downarrow \underline{\mathtt{n}}$. Thus, by correctness, there exists a ρ such that

$$[\![\mathtt{M}]\!]\rho[\boldsymbol{F}_1^{\sigma_1} := [\![\mathtt{P}_1]\!]\rho, \ldots, \boldsymbol{F}_{m+1}^{\sigma_{m+1}} := [\![\mathtt{P}_{m+1}]\!]\rho] \neq [\![\mathtt{N}]\!]\rho[\boldsymbol{F}_1^{\sigma_1} := [\![\mathtt{P}_1]\!]\rho, \ldots, \boldsymbol{F}_{m+1}^{\sigma_{m+1}} := [\![\mathtt{P}_{m+1}]\!]\rho].$$

In the following, we define $\rho_1 = \rho[\boldsymbol{F}_1^{\sigma_1} := [\![\mathtt{P}_1]\!]\rho, \ldots, \boldsymbol{F}_m^{\sigma_m} := [\![\mathtt{P}_m]\!]\rho]$. Since $\Gamma \vdash \mathtt{M}, \mathtt{N} : \iota$, both $[\![\mathtt{M}]\!]\rho_1[\boldsymbol{F}_{m+1}^{\sigma_{m+1}} := [\![\mathtt{P}_{m+1}]\!]\rho]$ and $[\![\mathtt{N}]\!]\rho_1[\boldsymbol{F}_{m+1}^{\sigma_{m+1}} := [\![\mathtt{P}_{m+1}]\!]\rho]$ are finite sets (in particular, they have at most one element and they cannot be both empty). Thus, without loss of generality, we assume that $[\![\mathtt{M}]\!]\rho_1[\boldsymbol{F}_{m+1}^{\sigma_{m+1}} := [\![\mathtt{P}_{m+1}]\!]\rho] = \{k\}$ with $k \in \mathbb{N}$. So by Lemma 13 there exists $x \subseteq_{fin} [\![\mathtt{P}_{m+1}]\!]\rho$ such that

$$[\![\mathtt{M}]\!]\rho_1[\boldsymbol{F}_{m+1} := [\![\mathtt{P}_{m+1}]\!]\rho] = [\![\mathtt{M}]\!]\rho_1[\boldsymbol{F}_{m+1} := x] \neq [\![\mathtt{N}]\!]\rho_1[\boldsymbol{F}_{m+1} := x] \subseteq [\![\mathtt{N}]\!]\rho_1[\boldsymbol{F}_{m+1} := [\![\mathtt{P}_{m+1}]\!]\rho].$$

Since stable variables are never of ground type, we can let $\sigma_{m+1} = \tau_1 \multimap \cdots \multimap \tau_l \multimap \iota$ and $l \geqslant 1$. Let $x = \{(a_1^1, \ldots, a_l^1, h^1), \ldots, (a_1^p, \ldots, a_l^p, h^p)\}$ with $p \geqslant 0$ and $a_1^i \in |[\![\tau_1]\!]|, \ldots, a_l^i \in |[\![\tau_l]\!]|, h^i \in \mathbb{N}$ for all $i \in [1, p]$. Furthermore, by Corollary 6 there is a token $(a_1^*, \ldots, a_l^*, h^*) \notin x$ such that $x^* = x \cup \{(a_1^*, \ldots, a_l^*, h^*)\}$ is still a clique. Observe that it is not restrictive to

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

assume $h^* \notin \{h_1, \ldots, h_p\}$, since changing of outputs preserves the coherence of tokens. By finite definability, there is a term $\mathtt{P}$ defining the clique x^*. Now, let us consider the context

$$\mathtt{D}[\,^{l}] = \mu_{\mathcal{F}m+1}.\lambda \mathtt{f}_1^{\tau_1} \ldots \mathtt{f}_m^{\tau_l}.$$
$$\left(\lambda \mathtt{x}'.\ell\mathtt{if}\ (\mathtt{x} \doteq \underline{\mathtt{h}}^*)\ [\,^{l}]\ \left(\ell\mathtt{if}\ (\mathtt{x} \doteq \underline{\mathtt{h}}_1)\ \underline{\mathtt{h}}_1\ \left(\ldots\ldots \ell\mathtt{if}\ (\mathtt{x} \doteq \underline{\mathtt{h}}_\mathtt{p})\underline{\mathtt{h}}_\mathtt{p}\ \Omega^{l} \ldots\right)\right)\right)(\mathtt{P}\ \mathtt{f}_1\ \ldots \mathtt{f}_1).$$

Observe that $\mathtt{D}[\mathtt{M}], \mathtt{D}[\mathtt{N}] \in \mathcal{S}\mathrm{PCF}_\star$ because we assumed that $\mathtt{M}, \mathtt{N}$ have no free occurrences of linear variables. If $q_1 = (\!|a_1^*|\!), \ldots, q_m = (\!|a_l^*|\!)$, it is not difficult to see that

$$[\![\mathtt{D}[\mathtt{M}]\ \mathtt{Sgl}_{\mathtt{q}_1}^{(\tau_1)}\ \ldots\ \mathtt{Sgl}_{\mathtt{q}_1}^{(\tau_l)}]\!]\rho_1 = [\![\mathtt{M}]\!]\rho_1[\mathcal{F}_{m+1} := [\![\mathtt{P}_{\mathtt{m}+1}]\!]\rho] = \{k\} \quad \text{and,}$$
$$[\![\mathtt{D}[\mathtt{N}]\ \mathtt{Sgl}_{\mathtt{q}_1}^{(\tau_1)}\ \ldots\ \mathtt{Sgl}_{\mathtt{q}_1}^{(\tau_l)}]\!]\rho_1 \subseteq [\![\mathtt{N}]\!]\rho_1[\mathcal{F}_{m+1} := [\![\mathtt{P}_{\mathtt{m}+1}]\!]\rho]$$

which is either empty or it is a singleton $\{k'\}$ different from k. By Lemma 13, adequacy and fix-point equivalence: $\mathtt{D}[\![\mathtt{M}[\mathtt{P}_1/\mathcal{F}_1^{\sigma_1}, \ldots, \mathtt{P}_\mathtt{m}/\mathcal{F}_m^{\sigma_m}]]\!]\ \mathtt{Sgl}_{\mathtt{q}_1}^{(\tau_1)}\ \ldots\ \mathtt{Sgl}_{\mathtt{q}_1}^{(\tau_l)} \Downarrow \underline{\mathtt{k}}$ and, $\mathtt{D}[\![\mathtt{N}[\mathtt{P}_1/\mathcal{F}_1^{\sigma_1}, \ldots, \mathtt{P}_{\mathtt{m}+1}/\mathcal{F}_m^{\sigma_m}]]\!]\ \mathtt{Sgl}_{\mathtt{q}_1}^{(\tau_1)}\ \ldots\ \mathtt{Sgl}_{\mathtt{q}_1}^{(\tau_l)}$ diverges or converges to $\underline{\mathtt{k}}'$. Thus, we can apply inductive hypothesis (since the number of free stable variables has decreased) to conclude the proof. □

The above lemma can be used to prove the general case.

Theorem 9. Let $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathcal{S}\mathrm{PCF}_\star$. If $\mathtt{M} \approx_\sigma \mathtt{N}$ then $\mathtt{M} \sim_\sigma \mathtt{N}$.

Proof. We prove the contrapositive statement. Let us consider $\mathcal{F}_1^{\sigma_1}, \ldots, \mathcal{F}_n^{\sigma_n}$ such that $\mathrm{SFV}(\mathtt{M}), \mathrm{SFV}(\mathtt{N}) \subseteq \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$. Let $\mathtt{C}[^\sigma]$ be a context and $\vec{\mathtt{P}}$ be closed terms such that $\mathtt{C}[\mathtt{M}[\vec{\mathtt{P}}/\vec{\mathcal{F}}]] \Downarrow \underline{\mathtt{n}}$ and $\mathtt{C}[\mathtt{N}[\vec{\mathtt{P}}/\vec{\mathcal{F}}]] \not\Downarrow \underline{\mathtt{n}}$ for some numeral $\underline{\mathtt{n}}$. Namely, $\mathtt{C}[\mathtt{M}[\vec{\mathtt{P}}/\vec{\mathcal{F}}]], \mathtt{C}[\mathtt{N}[\vec{\mathtt{P}}/\vec{\mathcal{F}}]] \in \mathrm{P}$, $\mathtt{C}[\mathtt{M}] \not\sim_\iota \mathtt{C}[\mathtt{N}]$ and, both $\mathtt{C}[\mathtt{M}]$, $\mathtt{C}[\mathtt{N}]$ have no free occurrence of linear variables. Thus, by Lemma 14 there is a context $\mathtt{D}[\,^{l}]$ such that $\mathtt{D}[\mathtt{C}[\mathtt{M}]] \Downarrow \underline{\mathtt{m}}$ and $\mathtt{D}[\mathtt{C}[\mathtt{M}]] \not\Downarrow \underline{\mathtt{m}}$. So, the proof is done. □

Corollary 7. The equivalence $\sim_\sigma$ is a congruence.

6.1. *Applicative operational equivalence*

We conclude the section by defining an applicative operational equivalence obtained by considering only special kinds of contexts (*applicative contexts*) to test the equality of terms.

Definitiona 19 (applicative operational equivalence). Let $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathcal{S}\mathrm{PCF}_\star$ such that $\mathrm{SFV}(\mathtt{M}), \mathrm{SFV}(\mathtt{N}) \subseteq \{\mathcal{F}_1^{\sigma_1}, \ldots, \mathcal{F}_n^{\sigma_n}\}$.

- $\mathtt{M} \lesssim_\sigma^{\mathtt{A}} \mathtt{N}$ whenever, for all context $\mathtt{C}$ of the form $(\lambda\vec{\mathtt{f}}.[\cdot^\sigma])\ \mathtt{P}_1 \ldots \mathtt{P}_\mathtt{m}$ and for all closed terms $\mathtt{L}_1^{\sigma_1}, \ldots, \mathtt{L}_n^{\sigma_1}$, if $\mathtt{C}[\mathtt{M}[\vec{\mathtt{L}}/\vec{\mathcal{F}}]] \Downarrow \underline{\mathtt{n}}$ then $\mathtt{C}[\mathtt{N}[\vec{\mathtt{L}}/\vec{\mathcal{F}}]] \Downarrow \underline{\mathtt{n}}$.
- $\mathtt{M} \sim_\sigma^{\mathtt{A}} \mathtt{N}$ iff $\mathtt{M} \lesssim_\sigma^{\mathtt{A}} \mathtt{N}$ and $\mathtt{N} \lesssim_\sigma^{\mathtt{A}} \mathtt{M}$.

Theorem 10. Let $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathcal{S}\mathrm{PCF}_\star$. If $\mathtt{M} \sim_\sigma^{\mathtt{A}} \mathtt{N}$ then $[\![\mathtt{M}]\!] = [\![\mathtt{N}]\!]$.

Proof. Just by observing that the context used in the proof of Theorem 8 is an applicative context. □

The applicative equivalence still coincides with the previous ones, so it provides a convenient tool for reasoning on programs.

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

Corollary 8. Let $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathcal{S}\ell\mathrm{PCF}_\star$. Then $\mathtt{M} \sim_\sigma \mathtt{N}$, $\mathtt{M} \approx_\sigma \mathtt{N}$ and $\mathtt{M} \sim_\sigma^{\mathtt{A}} \mathtt{N}$ coincide.

7. Enhanced tracing reduction machine

The operational semantics of $\mathcal{S}\ell\mathrm{PCF}_\star$ presented in Sections 2 and 4 is effective, but quite inefficient. The operational rules for `which?` and ℓ`et-`ℓ`or` non-deterministically face a potentially infinite number of evaluation branches, therefore an exhaustive search of the right branch among the infinite ones is needed. In this section, we introduce a *enhanced tracing evaluation semantics* which is able to drastically prune such infinite-branching search tree. Roughly speaking, denotational linearity provides the certainty that each term is applied to a unique sequence of arguments. Consequently, only one token in its interpretation is used. An efficient evaluation can be obtained by introducing a variable (a name) for each argument of a function, then the term is stored in correspondence of such name (in a suitable environment) and finally, during the evaluation the term is traced and the trace is stored in the environment. Summarizing, the idea is 'to recursively trace' the arguments supplied to functions (by means of an environment) that record trace-information along the evaluation tree. The enhanced tracing evaluation semantics improve tracing we presented in Gaboardi *et al.* (2011) avoiding its main computational defect (i.e. a free use of substitutions).

7.1. *The recursion-free fragment*

For sake of simplicity, we start by presenting the tracing evaluation of the *recursion-free fragment* of $\mathcal{S}\ell\mathrm{PCF}_\star$, i.e. terms that does neither contain stable variables nor μ-abstractions.

Following Barendregt, we say that a term respects the hygienic-condition when all variables have different names. More precisely, any two different bound variables have different names and, the name of one free variable is different from that of all bound ones. $\mathcal{S}\ell\mathrm{PCF}_\star$ endowed with the straightforward notion of reduction does not preserve hygienic-conditions by evaluation because it involves both recursive and additive features. For instance, $(\lambda \mathtt{f}^{\iota \multimap \iota}.\ell\mathtt{if\ 0\ f3\ f5})(\lambda \mathtt{x}^\iota.\mathtt{x}^\iota)$ respects the hygienic-conditions, but the evaluations involves both $(\lambda \mathtt{x}^\iota.\mathtt{x}^\iota)\underline{3}$ and $(\lambda \mathtt{x}^\iota.\mathtt{x}^\iota)\underline{5}$. However, in the recursion-free fragment of $\mathcal{S}\ell\mathrm{PCF}_\star$, we can state a remarkable property about the hygienic-compliance of the evaluation tree (in the case of a converging evaluation).

Property 2. Let $\mathtt{M}$ be a program in the recursion-free fragment of $\mathcal{S}\ell\mathrm{PCF}_\star$ and $\mathtt{M}$ respecting the hygienic-condition. Given a converging evaluation derivation $\mathcal{D}$ proving $\mathtt{M} \Downarrow \underline{\mathtt{m}}$, if $\mathcal{D}'$ is a sub-derivation of $\mathcal{D}$ with conclusion $\mathtt{N} \Downarrow \underline{\mathtt{n}}$ then,

— $\mathtt{N}$ respect the hygienic-condition provided that: in the premise of the operational rule (`w`) the (introduced) abstracted variable $\mathtt{x}^\iota$ is fresh and, in the premise of rules (`1lgor`) (`2lgor`), (`3lgor`), we rename variables (by using pairwise different fresh variables) in the duplicated terms;

— we can define a (finite) function from all the bound variables of $\mathtt{N}$ (they have different names by hygienic-conditions) to terms of the corresponding type as follows: to each $\mathtt{x}^\sigma$ we associate a $\mathtt{Sgl}_n^\sigma$ for a suitable $\underline{\mathtt{n}}$ such that $\mathtt{Chk}_n^{(\sigma)}(\mathtt{P}) = 0$ whenever $\mathtt{P}$ replaces $\mathtt{x}^\sigma$

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

in a sub-derivations $\mathcal{D}^*$ of $\mathcal{D}$; this condition endures that $\mathtt{Sgl}_n^\sigma$ can be used in place of $\mathtt{P}$.

Proof. The proof follows easily by straightforward induction on the evaluation rules. □

The above property suggests that the recursion-free fragment of the evaluation semantics can be rewritten by avoiding immediate substitution, we can use a global environment-list to record the term supposed to be substituted to a given variable and to use it, when the variable is encountered. This is essentially the tracing operational semantics we are going to propose. From now on, for sake of simplicity, we assume that we evaluate only terms respecting the hygienic-condition.

In order to collect the tracing information in an evaluation tree, we introduce a very simple notion of environment.

Definitiona 20. An *environment* $\mathfrak{L}$ is a finite list of entries that pairs variables (ground or linear) to either a term (named *argument*) or a numeral (named *trace*).

Let $\mathtt{x}^\sigma$ be a variable in the domain of $\mathfrak{L}$. If $\sigma = \iota$ then $\mathfrak{L}(\mathtt{x})$ is always the trace, i.e. a numeral. If σ is an arrow then either $\mathfrak{L}(\mathtt{x})$ is a term typed σ (the argument) or it is a trace typed ι.

We note [.], an empty environment and we use $[\mathtt{f}_1 := \mathtt{N}_1, \ldots, \mathtt{f}_k := \mathtt{N}_k] :: \mathfrak{L}$ to represent the environment list obtained by appending the k pairs in the first list in front of the pairs of list $\mathfrak{L}$.

During the evaluation, we use variable-names as hold-place, recording in the environment the corresponding term when it is encountered and recording the trace when it becomes available. Linearity gives us a notable benefit in the Landin-style evaluation that we are presenting, namely we avoid the use of closures (i.e. recursive data-structures) and we avoid the duplication of the environments on subterms. We remark that we need a generator of fresh variables, with two main purposes. First, in order to trace the evaluation of `which?` arguments (each `which?` argument is associated to a fresh variable to be traced), second in order to trace subterms.

The rule of the enhanced tracing machine will be driven by *states* (i.e. term in environment); we denote them by $\langle \mathtt{M} | \mathfrak{L} \rangle$, where $\mathfrak{L}$ involves all free variable of $\mathtt{M}$. When we evaluate a subterm we do not restrict the environment to its free variables, for sake of efficiency, thus sometimes $\mathfrak{L}$ will involve more than the free variables of $\mathtt{M}$. Anyway, for sake of clarity, we explicitly remove unused variable-names from environments.

If $\mathtt{M}'$ is a closed term then we can obtain its evaluation, by supplying the state $\langle \mathtt{M} | [.] \rangle$ to the tracing machine that we are introducing.

Definitiona 21. The *enhanced tracing evaluation* (for the recursion-free evaluation of $\mathcal{S}\mathcal{L}\mathrm{PCF}_\star$) is the effective relation $\Downarrow_{\mathtt{E}}$ from states (ground terms in environments) to states (numerals in environments) defined by the rules of Table 4. If $\langle \mathtt{M} | \mathfrak{L}_0 \rangle \Downarrow_{\mathtt{E}} \langle \underline{\mathtt{n}} | \mathfrak{L}_1 \rangle$ then we say that $\mathtt{M}$ *converges*, and we write simply $\langle \mathtt{M} | \mathfrak{L}_0 \rangle \Downarrow_{\mathtt{E}}$, otherwise we say that it *diverges*, and we write $\langle \mathtt{M} | \mathfrak{L}_0 \rangle \Uparrow_{\mathtt{E}}$.

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

Table 4. *Enhanced tracing evaluation for the recursion-free fragment of S∕PCF⋆.*

$$\frac{}{\langle \underline{0}|\mathcal{L}\rangle \Downarrow_{\mathrm{E}} \langle \underline{0}|\mathcal{L}\rangle}\ (0) \qquad \frac{\langle \mathtt{M}|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|\mathcal{L}_1\rangle}{\langle \mathbf{s}\,\mathtt{M}|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \mathbf{s}\,\underline{\mathtt{n}}|\mathcal{L}_1\rangle}\ (\mathrm{s}) \qquad \frac{\langle \mathtt{M}|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \mathbf{s}\,\underline{\mathtt{n}}|\mathcal{L}_1\rangle}{\langle \mathbf{p}\,\mathtt{M}|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|\mathcal{L}_1\rangle}\ (\mathrm{p})$$

$$\frac{\langle \mathtt{M}|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{0}|\mathcal{L}_1\rangle \quad \langle \mathtt{L}|\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{m}}|\mathcal{L}_2\rangle}{\langle \ell\mathtt{if}\ \mathtt{M}\ \mathtt{L}\ \mathtt{R}|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{m}}|\mathcal{L}_2\rangle}\ (\mathrm{ifl}) \qquad \frac{\langle \mathtt{M}|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \mathbf{s}\underline{\mathtt{n}}|\mathcal{L}_1\rangle \quad \langle \mathtt{R}|\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{m}}|\mathcal{L}_2\rangle}{\langle \ell\mathtt{if}\ \mathtt{M}\ \mathtt{L}\ \mathtt{R}|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{m}}|\mathcal{L}_2\rangle}\ (\mathrm{ifr})$$

$$\frac{\langle \mathtt{N}|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}'|\mathcal{L}_1\rangle \quad \langle \mathtt{MP}_1 \dots \mathtt{P}_k|[\mathtt{x} := \underline{\mathtt{n}}']::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|[\mathtt{x} := \underline{\mathtt{n}}']::\mathcal{L}_2\rangle}{\langle (\lambda\mathtt{x}^{\iota}.\mathtt{M})\mathtt{NP}_1 \dots \mathtt{P}_k|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|\mathcal{L}_2\rangle}\ (\lambda^{\iota})$$

$$\frac{\langle \mathtt{MP}_1 \dots \mathtt{P}_k|[\mathtt{f} := \mathtt{N}]::\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|[\mathtt{f} := \underline{\mathtt{t}}]::\mathcal{L}_1\rangle}{\langle (\lambda\mathtt{f}^{\sigma\multimap\tau}.\mathtt{M})\mathtt{NP}_1 \dots \mathtt{P}_k|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|\mathcal{L}_1\rangle}\ (\lambda^{\multimap})$$

$$\frac{\mathtt{h}^{(\iota\multimap\iota)\multimap\iota}\ \text{fresh}, \quad \langle \mathtt{hI}|[\mathtt{h} := \mathtt{M}]::\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|[\mathtt{h} := \underline{\mathtt{t}}]::\mathcal{L}_1\rangle \quad \underline{\mathtt{k}} = \pi_1\pi_1(\underline{\mathtt{t}})}{\langle \mathtt{which?}\,\mathtt{M}^{(\iota\multimap\iota)\multimap\iota}|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \lceil \underline{\mathtt{n}}, \underline{\mathtt{k}} \rceil|\mathcal{L}_1\rangle}\ (\mathtt{w})$$

$$\frac{\begin{array}{c}\langle \mathtt{M}_1|[\mathtt{f}_1 := \mathtt{N}_1, \dots, \mathtt{f}_k := \mathtt{N}_k]::\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{0}|[\mathtt{f}_1 := \underline{\mathtt{n}}_1, \dots, \mathtt{f}_k := \underline{\mathtt{n}}_k]::\mathcal{L}_1\rangle \\ \langle \mathtt{M}_2|[\mathtt{f}_1 := \mathtt{Sgl}^{\sigma_1}_{\underline{\mathtt{n}}_1}, \dots, \mathtt{f}_k := \mathtt{Sgl}^{\sigma_k}_{\underline{\mathtt{n}}_k}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \mathbf{s}\underline{\mathtt{m}}|[\mathtt{f}_1 := \underline{\mathtt{n}}_1, \dots, \mathtt{f}_k := \underline{\mathtt{n}}_k]::\mathcal{L}_2\rangle\end{array}}{\langle \underline{\ell\mathtt{et}}\ \mathtt{f}_1^{\sigma_1} = \mathtt{N}_1, \dots, \mathtt{f}_k^{\sigma_k} = \mathtt{N}_k\ \underline{\mathtt{in}\ell\mathtt{or}}\ \mathtt{M}_1\ \mathtt{M}_2\ \mathtt{M}_3|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{m}}|\mathcal{L}_2\rangle}\ (1\mathrm{lgor})$$

$$\frac{\begin{array}{c}\langle \mathtt{M}_2|[\mathtt{f}_1 := \mathtt{N}_1, \dots, \mathtt{f}_k := \mathtt{N}_k]::\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{0}|[\mathtt{f}_1 := \underline{\mathtt{n}}_1, \dots, \mathtt{f}_k := \underline{\mathtt{n}}_k]::\mathcal{L}_1\rangle \\ \langle \mathtt{M}_3|[\mathtt{f}_1 := \mathtt{Sgl}^{\sigma_1}_{\underline{\mathtt{n}}_1}, \dots, \mathtt{f}_k := \mathtt{Sgl}^{\sigma_k}_{\underline{\mathtt{n}}_k}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \mathbf{s}\underline{\mathtt{m}}|[\mathtt{f}_1 := \underline{\mathtt{n}}_1, \dots, \mathtt{f}_k := \underline{\mathtt{n}}_k]::\mathcal{L}_2\rangle\end{array}}{\langle \underline{\ell\mathtt{et}}\ \mathtt{f}_1^{\sigma_1} = \mathtt{N}_1, \dots, \mathtt{f}_k^{\sigma_k} = \mathtt{N}_k\ \underline{\mathtt{in}\ell\mathtt{or}}\ \mathtt{M}_1\ \mathtt{M}_2\ \mathtt{M}_3|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{m}}|\mathcal{L}_2\rangle}\ (2\mathrm{lgor})$$

$$\frac{\begin{array}{c}\langle \mathtt{M}_3|[\mathtt{f}_1 := \mathtt{N}_1, \dots, \mathtt{f}_k := \mathtt{N}_k]::\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{0}|[\mathtt{f}_1 := \underline{\mathtt{n}}_1, \dots, \mathtt{f}_k := \underline{\mathtt{n}}_k]::\mathcal{L}_1\rangle \\ \langle \mathtt{M}_0|[\mathtt{f}_1 := \mathtt{Sgl}^{\sigma_1}_{\underline{\mathtt{n}}_1}, \dots, \mathtt{f}_k := \mathtt{Sgl}^{\sigma_k}_{\underline{\mathtt{n}}_k}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \mathbf{s}\underline{\mathtt{m}}|[\mathtt{f}_1 := \underline{\mathtt{n}}_1, \dots, \mathtt{f}_k := \underline{\mathtt{n}}_k]::\mathcal{L}_2\rangle\end{array}}{\langle \underline{\ell\mathtt{et}}\ \mathtt{f}_1^{\sigma_1} = \mathtt{N}_1, \dots, \mathtt{f}_k^{\sigma_k} = \mathtt{N}_k\ \underline{\mathtt{in}\ell\mathtt{or}}\ \mathtt{M}_1\ \mathtt{M}_2\ \mathtt{M}_3|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{m}}|\mathcal{L}_2\rangle}\ (3\mathrm{lgor})$$

$$\frac{}{\langle \mathtt{x}^{\iota}|\mathcal{L}_0::[\mathtt{x} := \underline{\mathtt{n}}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|\mathcal{L}_0::[\mathtt{x} := \underline{\mathtt{n}}]::\mathcal{L}_1\rangle}\ (\mathrm{Hgvar})$$

$$\frac{\langle \mathtt{hP}_1 \dots \mathtt{P}_k|\mathcal{L}_0::[\mathtt{f} := \mathtt{h}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|\mathcal{L}'_0::[\mathtt{f} := \mathtt{h}]::\mathcal{L}'_{10}::[\mathtt{h} := \underline{\mathtt{t}}]::\mathcal{L}'_{11}\rangle}{\langle \mathtt{fP}_1 \dots \mathtt{P}_k|\mathcal{L}_0::[\mathtt{f} := \mathtt{h}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|\mathcal{L}'_0::[\mathtt{f} := \underline{\mathtt{t}}]::\mathcal{L}'_{10}::[\mathtt{h} := \underline{\mathtt{t}}]::\mathcal{L}'_{11}\rangle}\ (\mathrm{Hvar})$$

$$\frac{\mathtt{h}\ \text{fresh}, \quad \langle \mathtt{hNP}_1 \dots \mathtt{P}_k|[\mathtt{h} := \mathtt{M}]::\mathcal{L}_0::[\mathtt{f} := \mathtt{M}^{\sigma\multimap\tau}\mathtt{N}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|[\mathtt{h} := \underline{\mathtt{t}}]::\mathcal{L}_0::[\mathtt{f} := \mathtt{M}^{\sigma\multimap\tau}\mathtt{N}]::\mathcal{L}_1\rangle}{\langle \mathtt{fP}_1 \dots \mathtt{P}_k|\mathcal{L}_0::[\mathtt{f} := \mathtt{M}^{\sigma\multimap\tau}\mathtt{N}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|\mathcal{L}_0::[\mathtt{f} := \pi_2(\underline{\mathtt{t}})]::\mathcal{L}_1\rangle}\ (\mathrm{Happ})$$

$$\frac{\langle \mathtt{M}|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|\mathcal{L}_1\rangle}{\langle \mathtt{f}^{\iota\multimap\iota}\mathtt{M}|[\mathtt{f} := \mathbf{s}]::\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \mathbf{s}\underline{\mathtt{n}}|[\mathtt{f} := \lceil \underline{\mathtt{n}}, \mathbf{s}\underline{\mathtt{n}} \rceil]::\mathcal{L}_1\rangle}\ (\mathrm{Hs}) \qquad \frac{\langle \mathtt{M}|\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \mathbf{s}\,\underline{\mathtt{n}}|\mathcal{L}_1\rangle}{\langle \mathtt{f}^{\iota\multimap\iota}\mathtt{M}|[\mathtt{f} := \mathbf{p}]::\mathcal{L}_0\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|[\mathtt{f} := \lceil \mathbf{s}\underline{\mathtt{n}}, \underline{\mathtt{n}} \rceil]::\mathcal{L}_1\rangle}\ (\mathrm{Hp})$$

$$\frac{\mathtt{h}\ \text{fresh}, \quad \langle \mathtt{hI}|[\mathtt{h} := \mathtt{M}]::\mathcal{L}_0::[\mathtt{f} := \mathtt{which?}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|[\mathtt{h} := \underline{\mathtt{t}}]::\mathcal{L}_0::[\mathtt{f} := \mathtt{which?}]::\mathcal{L}_1\rangle \quad \underline{\mathtt{k}} = \pi_1(\pi_1(\underline{\mathtt{t}}))}{\langle \mathtt{fM}|\mathcal{L}_0::[\mathtt{f} := \mathtt{which?}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \lceil \lceil \underline{\mathtt{n}}, \underline{\mathtt{k}} \rceil, \underline{\mathtt{n}} \rceil, \lceil \underline{\mathtt{n}}, \underline{\mathtt{k}} \rceil \rceil|\mathcal{L}_0::[\mathtt{f} := \lceil \lceil \lceil \underline{\mathtt{k}}, \underline{\mathtt{k}} \rceil, \underline{\mathtt{n}} \rceil, \lceil \underline{\mathtt{n}}, \underline{\mathtt{k}} \rceil \rceil]::\mathcal{L}_1\rangle}\ (\mathrm{Hw})$$

$$\frac{\begin{array}{c}\langle \mathtt{P}|\mathcal{L}_0::[\mathtt{f} := \lambda\mathtt{x}^{\iota}.\mathtt{M}^{\iota}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{m}}|\mathcal{L}'_0::[\mathtt{f} := \lambda\mathtt{x}^{\iota}.\mathtt{M}^{\iota}]::\mathcal{L}'_1\rangle \\ \langle \mathtt{M}|[\mathtt{x}^{\iota} := \underline{\mathtt{m}}]::\mathcal{L}'_0::[\mathtt{f} := \lambda\mathtt{x}^{\iota}.\mathtt{M}^{\iota}]::\mathcal{L}'_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|[\mathtt{x}^{\iota} := \underline{\mathtt{m}}]::\mathcal{L}''_0::[\mathtt{f} := \lambda\mathtt{x}^{\iota}.\mathtt{M}^{\iota}]::\mathcal{L}''_1\rangle\end{array}}{\langle \mathtt{fP}|\mathcal{L}_0::[\mathtt{f} := \lambda\mathtt{x}^{\iota}.\mathtt{M}^{\iota}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|\mathcal{L}''_0::[\mathtt{f} := \lceil \underline{\mathtt{m}}, \underline{\mathtt{n}} \rceil]::\mathcal{L}''_1\rangle}\ (\mathrm{H}\lambda^{\iota}_{\iota})$$

$$\frac{\begin{array}{c}\mathtt{h}\ \text{fresh}, \quad \langle \mathtt{P}_1|\mathcal{L}_0::[\mathtt{f} := \lambda\mathtt{x}^{\iota}.\mathtt{M}^{\sigma\multimap\tau}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{m}}|\mathcal{L}'_0::[\mathtt{f} := \lambda\mathtt{x}^{\iota}.\mathtt{M}^{\sigma\multimap\tau}]::\mathcal{L}'_1\rangle \\ \langle \mathtt{hP}_2 \dots \mathtt{P}_k|[\mathtt{x} := \underline{\mathtt{m}}, \mathtt{h} := \mathtt{M}]::\mathcal{L}'_0::[\mathtt{f} := \lambda\mathtt{x}^{\iota}.\mathtt{M}^{\sigma\multimap\tau}]::\mathcal{L}'_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|[\mathtt{x} := \underline{\mathtt{m}}, \mathtt{h} := \underline{\mathtt{t}}]::\mathcal{L}''_0::[\mathtt{f} := \lambda\mathtt{x}^{\iota}.\mathtt{M}^{\sigma\multimap\tau}]::\mathcal{L}''_1\rangle\end{array}}{\langle \mathtt{fP}_1 \dots \mathtt{P}_k|\mathcal{L}_0::[\mathtt{f} := \lambda\mathtt{x}^{\iota}.\mathtt{M}^{\sigma\multimap\tau}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|\mathcal{L}''_0::[\mathtt{f} := \lceil \underline{\mathtt{m}}, \underline{\mathtt{t}} \rceil]::\mathcal{L}''_1\rangle}\ (\mathrm{H}\lambda^{\iota}_{\sigma\multimap\tau})$$

$$\frac{\langle \mathtt{M}|[\mathtt{g} := \mathtt{P}]::\mathcal{L}_0::[\mathtt{f} := \lambda\mathtt{g}^{\sigma\multimap\tau}.\mathtt{M}^{\iota}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|[\mathtt{g} := \underline{\mathtt{t}}]::\mathcal{L}'_0::[\mathtt{f} := \lambda\mathtt{g}^{\sigma\multimap\tau}.\mathtt{M}^{\iota}]::\mathcal{L}'_1\rangle}{\langle \mathtt{fP}|\mathcal{L}_0::[\mathtt{f} := \lambda\mathtt{g}^{\sigma\multimap\tau}.\mathtt{M}^{\iota}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|\mathcal{L}'_0::[\mathtt{f} := \lceil \underline{\mathtt{t}}, \underline{\mathtt{n}} \rceil]::\mathcal{L}'_1\rangle}\ (\mathrm{H}\lambda^{\multimap}_{\iota})$$

$$\frac{\begin{array}{c}\mathtt{h}\ \text{fresh}, \\ \langle \mathtt{hP}_2 \dots \mathtt{P}_k|[\mathtt{g} := \mathtt{P}_1, \mathtt{h} := \mathtt{M}]::\mathcal{L}_0::[\mathtt{f} := \lambda\mathtt{g}^{\sigma\multimap\sigma'}.\mathtt{M}^{\tau\multimap\tau'}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|[\mathtt{g} := \underline{\mathtt{t}}_1, \mathtt{h} := \underline{\mathtt{t}}_{\mathtt{M}}]::\mathcal{L}'_0::[\mathtt{f} := \lambda\mathtt{g}^{\sigma\multimap\sigma'}.\mathtt{M}^{\tau\multimap\tau'}]::\mathcal{L}'_1\rangle\end{array}}{\langle \mathtt{fP}_1 \dots \mathtt{P}_k|\mathcal{L}_0::[\mathtt{f} := \lambda\mathtt{g}^{\sigma\multimap\sigma'}.\mathtt{M}^{\tau\multimap\tau'}]::\mathcal{L}_1\rangle \Downarrow_{\mathrm{E}} \langle \underline{\mathtt{n}}|\mathcal{L}'_0::[\mathtt{f} := \lceil \underline{\mathtt{t}}_1, \underline{\mathtt{t}}_{\mathtt{M}} \rceil]::\mathcal{L}'_1\rangle}\ (\mathrm{H}\lambda^{\multimap}_{\multimap})$$

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

To help the reader, we discuss how the rule have been conceived. The left-hand side of a state is always a term of the shape $\xi \mathtt{M}_0 \dots \mathtt{M}_m : \iota$ for some $m \geqslant 0$, where ξ (the *head*) is either a variable, or a numeral, an abstraction, or a `which?`, or $\ell\mathtt{if}\ \mathtt{M}\ \mathtt{L}\ \mathtt{R}$ (for some $\mathtt{M}, \mathtt{L}, \mathtt{R}$), or $\underline{\ell\mathtt{et}}\ \mathtt{f}_1^{\sigma_1} = \mathtt{N}_1, \dots, \mathtt{f}_\mathtt{k}^{\sigma_\mathtt{k}} = \mathtt{N}_\mathtt{k}\ \underline{\mathtt{in}\ell\mathtt{or}}\ \mathtt{M}_1\ \mathtt{M}_2\ \mathtt{M}_3$. In the latter two cases ($\ell\mathtt{if}$ and $\ell\mathtt{et}$-$\ell\mathtt{or}$) and when ξ is a numeral the typing rules imply that $m = 0$.

The enhanced tracing evaluation presented in Table 4 is driven by the head ξ of the term $\xi \mathtt{M}_0 \dots \mathtt{M}_m : \iota$ in the state $\langle \xi \mathtt{M}_0 \dots \mathtt{M}_m | \mathfrak{L} \rangle$, which appears in the left-hand side of the conclusion. In order to facilitate the comprehension of the tracing semantics, we have devised its rules in two parts. The rules in the higher part take into account all possible shapes of the head ξ, but the case of a (head) variable. The latter case is tackled by the lower part of Table 4, they are driven by the shapes of terms associated to the (variable) ξ in the environment.

The higher part of Table 4 is quite easy to understand, so we comment only rules involving non-standard operators. The rule (`w`) extends the incoming environment $\mathfrak{L}_0$ by a fresh variable `h` associated to the term `M` that we plan to trace, and evaluates `h` applied to the identity $\mathtt{I} = \lambda\mathtt{x}.\mathtt{x}$ (recall that if $\mathtt{which?}(\mathtt{M})$ converges, by rules in the Table 1(c) , then `MI` also converges). The result of this evaluation gives back a state $\langle \underline{\mathtt{n}} | \mathfrak{L}_1 \rangle$, where $\mathfrak{L}_1$ reports the observed trace of `M` that allows us to build the expected output information. Likewise, each of the three $\ell\mathtt{et}$-$\ell\mathtt{or}$ rules start by evaluating a branch (non-deterministically). If a branch converges (in a rule) then, the reported trace is used in order to start the evaluation of the other branch, where we supply as arguments of the involved variables a term having (exactly) the behaviour of the reported trace (recall the Lemma 6). Therefore, a rule converges only in case the two (involved) branches do the same observations on the list of arguments supplied to a linear variable.

The lower part of Table 4 contains rules having in the left-hand side of conclusion, a state of the shape $\langle \varkappa \mathtt{M}_0 \dots \mathtt{M}_\mathtt{m} | \mathfrak{L} \rangle$ where $\varkappa$ is the head variable. These rules are driven by the shape of the term associated to $\varkappa$ in $\mathfrak{L}$. Because (well typed) $\ell\mathtt{et}$-$\ell\mathtt{or}$-terms, well-typed $\ell\mathtt{if}$-terms and ground variables are typed ι, we remark no $\mathfrak{L}$ can associates $\varkappa$ to one of them. Indeed, the call-by-value policy for ground sub-terms force their evaluations before storing them in the environment.

The rule (Hgvar) is easy. (Hvar) considers the case where the argument associated to the head variable is another variable-name, it forwards the evaluation by using the argument (this case can happen only with arrow-typed variables). (Happ) is a key rule. It applies in the case the argument associated to the head variable is an application `MN`. In this case, it shifts `N` in the term of the state driving the rules. Such approach allows us to collect sufficient information to report the needed traces. The rules (H**s**), (H**p**) are quite simple. Likewise to the rule (`w`) presented above, the rule (H`w`) is interesting: it uses the information gathered by the fresh variable `h` to produce the right outcome and the trace of the `which?` itself. Finally, the rules ($\mathrm{H}\lambda^{\iota}_{\iota}$), ($\mathrm{H}\lambda^{\iota}_{\circ}$), ($\mathrm{H}\lambda^{\neg\circ}_{\iota}$) and ($\mathrm{H}\lambda^{\neg\circ}_{\circ}$) consider the cases arising from a head variable associated to an abstraction in the environment. There are four rules depending from two types, i.e. the type of the body of the abstraction and the type of the abstracted variable: mnemonically, the rule's names use as superscript the type of the variable and as subscript the type of the body. The rules ($\mathrm{H}\lambda^{\iota}_{\iota}$) and ($\mathrm{H}\lambda^{\iota}_{\circ}$) need first to evaluate the ground argument in order to comply the call-by-value policy. All the rules

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

use trace-information in the (right-hand side of) premises to obtain the trace-information that they provide in the (right-hand side of) conclusion.

The tracing evaluation semantics is very concrete and it can be straightforwardly adapted to use the de Bruijn indexes (de Bruijn 1972) to avoid the needed for variable names in environments, since our environments are never reordered and grow only on the left-hand side. Therefore, this evaluation looks as a reasonable base for an implementation.

7.2. *Full evaluation*

We extend the evaluation provided in the previous sub-section by adding the evaluation rules for recursion. First of all, we can assume without loss of generality that all names of stable variables respect hygienic conditions, i.e. we assume that all 'identifiers' in programs are unique. A μ-abstraction $\mu_{\mathtt{F}}.\mathtt{M}$ is morally a ML-like expression $\underline{\mathsf{letrec}}\ \mathtt{F} = \mathtt{M}\ \underline{\mathsf{in}}\ \mathtt{M}$. A standard compilation technique to normalize the presence of $\underline{\mathsf{letrec}}$ in programs is the lambda-lifting (see Johnsson (1985)). Lambda-lifting allows us to transform programs so that all recursive-definitions are given in one giant $\underline{\mathsf{letrec}}$ at the top-level and the only free variables of definition-bodies are just stable ones.

We adapt the lifting algorithm to $\mathbb{S}\not{\mathbb{E}}\mathrm{PCF}_\star$. Let $\mathtt{M}$ be a term of $\mathbb{S}\not{\mathbb{E}}\mathrm{PCF}_\star$ and let assume that in $\mathtt{M}$ occur $k \geqslant 0$ (different) μ-abstractions.

We say that a μ-abstraction is *liftable* whenever it does not contain ground (and linear) free-variables (albeit, it can contain stable ones). We remind that a well-typed μ-abstraction cannot contain linear free variables. A not liftable μ-abstraction is said *unliftable*.

We describe our **lifting-algorithm**.

1. Let $\mathtt{X}$ be the term we are considering. If $\mathtt{X}$ only contains liftable μ-abstraction then we skip to step 3, otherwise we proceed to step 2.
2. Since $\mathtt{X}$ contains (at least) one unliftable μ-abstraction, we can always find a unliftable μ-abstraction $\mu_{\mathtt{F}\mathtt{B}}.\mathtt{B}$ (occurring in $\mathtt{X}$) that does not contain (as subterm) any other unliftable μ-abstraction. Let us assume that $\mathrm{FV}(\mu_{\mathtt{F}\mathtt{B}}.\mathtt{B})\restriction\iota = \{\mathtt{z}_1^{\sigma_1},\ldots,\mathtt{z}_h^{\sigma_h}\}$ for some $h \geqslant 1$; note that we are just looking to ground free variables. We replace the (unique) occurrence of $\mu_{\mathtt{F}\mathtt{B}}^{\sigma}.\mathtt{B}^{\sigma}$ in $\mathtt{X}$ by $(\mu_{\mathtt{F}\mathtt{A}}^{\sigma_1\to\ldots\to\sigma_h\to\sigma}.\mathtt{A}^{\sigma_1\to\ldots\to\sigma_h\to\sigma})\mathtt{z}_1^{\sigma_1}\ldots\mathtt{z}_h^{\sigma_h}$ where
$$\mathtt{A} = \lambda\mathtt{y}_1^{\sigma_1}\ldots\mathtt{y}_h^{\sigma_h}.\mathtt{B}[\mathtt{y}_1^{\sigma_1}/\mathtt{z}_1^{\sigma_1}\ldots\mathtt{y}_h^{\sigma_h}/\mathtt{z}_h^{\sigma_h},(\mathtt{F}_{\mathtt{A}}^{\sigma_1\to\ldots\to\sigma_h\to\sigma}\mathtt{y}_1^{\sigma_1}\ldots\mathtt{y}_h^{\sigma_h})/\mathtt{F}_{\mathtt{B}}^{\sigma}]$$
and $\mathtt{y}_1^{\sigma_1},\ldots,\mathtt{y}_h^{\sigma_h}$ are fresh variables. Then, we proceed by re-applying the Step 1, to the term built in this step. Patently, this iteration ends in finite steps, since there are finite unliftable μ-abstractions in $\mathtt{X}$ and Step 2 decrease it.
3. Let $\mathfrak{Def}$ be a finite function from stable variables to $\mathbb{S}\not{\mathbb{E}}\mathrm{PCF}_\star$-terms such that all (possible) free variables are stable ones. We start by assuming $\mathfrak{Def}$ be the empty function and that $\mathtt{X}$ is the term obtained by previous steps (containing only liftable μ-abstractions).
4. If $\mathtt{X}$ does not contain (liftable) μ-abstraction we concluded the lifting, otherwise we proceed to step 5.
5. Since $\mathtt{X}$ contains (at least) one liftable μ-abstraction, we can always find a liftable μ-abstraction $\mu_{\mathtt{F}\mathtt{B}}.\mathtt{B}$ (occurring in $\mathtt{X}$) that does not contain (as subterm) any other μ-abstraction (albeit it can contain free stable variables). We extend $\mathfrak{Def}$ by associating

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

Table 5. *XP-reduction machinary: recursion!*

$$\frac{\langle \mathfrak{Def}(\mathcal{F})^{\circledR} \mathtt{P}_1 \dots \mathtt{P}_k | \mathfrak{L}_0 \rangle \Downarrow_E \langle \underline{\mathtt{n}} | \mathfrak{L}_1 \rangle}{\langle \mathcal{F} \mathtt{P}_1 \dots \mathtt{P}_k | \mathfrak{L}_0 \rangle \Downarrow_E \langle \underline{\mathtt{n}} | \mathfrak{L}_1 \rangle} \ (\mathcal{F}) \qquad \frac{\langle \mathtt{f} \mathtt{P}_1 \dots \mathtt{P}_k | \mathfrak{L}_0 :: [\mathtt{f} := \mathfrak{Def}(\mathcal{F})^{\circledR}] :: \mathfrak{L}_1 \rangle \Downarrow_E \langle \underline{\mathtt{n}} | \mathfrak{L}_2 \rangle}{\langle \mathtt{f} \mathtt{P}_1 \dots \mathtt{P}_k | \mathfrak{L}_0 :: [\mathtt{f} := \mathcal{F}] :: \mathfrak{L}_1] \rangle \Downarrow_E \langle \underline{\mathtt{n}} | \mathfrak{L}_2 \rangle} \ (\mathrm{H}\mu)$$

B to $\mathcal{F}_B$ and we replace in X the (unique) occurrence of $\mu_{\mathcal{F}_B}$.B by $\mathcal{F}_B$.
We proceed by coming back to the step 4, with the term built in this step and the updated $\mathfrak{Def}$. Patently, this iteration ends in finite steps, since there is a finite number of μ-abstractions in X and we decrease it.

The lifting-algorithm gives back a function $\mathfrak{Def}$ together with a term X of S⁄PCF$_\star$ free of μ-abstractions. Following the standard literature, the algorithm we considered can be improved in many ways. As the Johnsson's algorithm, we first perform parameter lifting and then perform block floating (Augustsson 1987; Johnsson 1987). The main difference with respect to standard presentation is that we do not move recursive-definitions at the top-level of the term itself, but we move them out of the starting program (i.e. in $\mathfrak{Def}$). Soundness proofs are very standard but quite boring, see for instance Augustsson (1987) and Johnsson (1987), we omit them.

By applying the lifting to S⁄PCF$_\star$, we transform a program in a definition-function $\mathfrak{Def}$ together with a body B to be evaluated. Remark that both B and terms in the range of $\mathfrak{Def}$ satisfy the proviso that, all its free variables are stable variables. *From now on we assume that the lifting has been executed*, so we can focus our attention on the evaluation of terms that does not contain μ-abstractions but (possibly) contains free stable variables. Such terms are supplied together a definition function $\mathfrak{Def}$ associating to each stable variable a suitable term.

Multiple recursive calls of the same definition can cause clash of variable names, to avoid them we need a refreshing function: if M contains only free variables and no μ-abstraction then $\mathtt{M}^{\circledR}$ is obtained by replacing all bound variable names by means of fresh (i.e. never used in the evaluation) names. Clearly, M and $\mathtt{M}^{\circledR}$ are α-equivalent in the standard sense.

Definitiona 22. The full evaluation of S⁄PCF$_\star$ is thus obtained by adding to the rules of Tables 4 those of Table 5.

The only peculiarity in these new rules is the use of the refreshing function on new definition-instances. This kind of refreshing is sufficient since the linearity assures that no duplication can occur. The lack of linearity impedes the use of the proposed evaluation technique to PCF: first, closures must be duplicated on subterms; second, the considered refreshing is useless in presence of term duplication along the computation.

Note that the refreshing function can be easily implemented avoiding the exploration of involved terms at each call. To reach this goal, the idea is to collect the position of the bound variables in definitions and their positions along the compilation phase. Then, we record such information in the definition function. Because, we always use $\mathfrak{Def}$ in conjunction with ®, we can transform these two logical operations in a single, more efficient, operation. The latter, by using the collected information, can smoothly provide a

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

refreshed instance of the definition. Standard techniques of nominal binding can be used to reach this goal.

Summarizing, the proposed enhanced tracing operational evaluation prune the infinite-branching search tree that affects the evaluation $\Downarrow$ presented in previous sections. In this sense, it is similar to the tracing evaluation presented in Gaboardi *et al.* (2011). However, we go definitively further the latter, by avoiding its main defect (i.e. a free use of substitutions) which is replaced by a very limited and controlled form of refreshing. Concluding, we feel that the proposed evaluation is very efficient.

7.3. *Soundness*

In this section, we prove that the two presented operational evaluations (namely, $\Downarrow$ and $\Downarrow_{\mathtt{E}}$) are equivalent. It is useful to remark that the environment-list $\mathfrak{L}$ can be regarded as a stack with the top element on its left.

Definitiona 23. Let $\mathfrak{L} = [\mathtt{f}_1 := \mathtt{N}_1, \ldots, \mathtt{f}_k := \mathtt{N}_k]$ $(k \geqslant 0)$ be an environment such that all variables are pairwise different[†]. The set $\mathsf{Dom}(\mathfrak{L}) = \{\mathtt{f}_1, \ldots, \mathtt{f}_n\}$ is the *domain* of $\mathfrak{L}$.

If $\mathfrak{L}$ is an environment, the arrow-typed variables in $\mathsf{Dom}(\mathfrak{L})$ are partitioned in two subsets, traced and not-traced. A variable $\mathtt{x}^{\sigma\to\tau} \in \mathsf{Dom}(\mathfrak{L})$ is *traced* when $\mathfrak{L}$ associates it to a numeral and it is *not-traced* when $\mathfrak{L}$ associates it to a terms of type $\sigma \to \tau$.

Definitiona 24. Let $\mathfrak{L} = [\mathtt{f}_1 := \mathtt{N}_1, \ldots, \mathtt{f}_k := \mathtt{N}_k]$ $(k \geqslant 0)$. Its *not-traced domain* $\mathsf{uDom}(\mathfrak{L})$ is the restriction of $\mathsf{Dom}(\mathfrak{L})$ to not-traced variable (included ground variables).
$\mathfrak{L}$ is *acyclic* whenever, for each $i \leqslant k$, $W_i = \mathrm{FV}(\mathtt{N}_i) - \mathrm{SVar}$ (namely the set of all free variables in $\mathtt{N}_i$ but stable ones) satisfies $W_i \subseteq \mathsf{uDom}([\mathtt{f}_{i+1} := \mathtt{N}_{i+1}, \ldots, \mathtt{f}_k := \mathtt{N}_k])$.

We are interested only in acyclic environment. The empty-environment is trivially acyclic. Acyclicity arises from the fact that the environment works as a stack, where we push subterms that contains (not stable) free variables bounded in the stack itself. If $\mathfrak{L}$ contains traced variables then the evaluation of $\langle \mathtt{M} | \mathfrak{L} \rangle$ leaves them untouched and unused. Recorded traces are just reported-back to the 'caller' of the evaluation i.e. if we are considering a sub-derivation of a derivation then the derivation can use the reported trace conveniently.

Definitiona 25. Let $\mathfrak{L}$ be an acyclic environment. If $\langle \mathtt{M} | \mathfrak{L} \rangle$ is a state then $\mathfrak{L}$ induces a substitution on $\mathtt{M}$ defined as follows:

— if $\mathfrak{L} = [\,]$ then $\mathfrak{L}(\!|\mathtt{M}|\!) = \mathtt{M}$,
— if $\mathfrak{L} = [\mathtt{f} := \mathtt{N}] :: \mathfrak{L}'$ and $\mathtt{f}$ is traced then, $\mathfrak{L}(\!|\mathtt{M}|\!) = \mathfrak{L}'(\!|\mathtt{M}|\!)$,
— if $\mathfrak{L} = [\mathtt{f} := \mathtt{N}] :: \mathfrak{L}'$ and $\mathtt{f}$ is not-traced then, $\mathfrak{L}(\!|\mathtt{M}|\!) = \mathfrak{L}'(\!|\mathtt{M}[\mathtt{N}/\mathtt{f}]|\!)$.

$\langle \mathtt{M} | \mathfrak{L} \rangle$ is *well formed* whenever, $\mathfrak{L}$ is acyclic and $\mathfrak{L}(\!|\mathtt{M}|\!)$ is a closed term typed ι.

Since the not-traced part of a domain includes all ground variables, they are always substituted by the operator $(\!|.|\!)$. If $\langle \mathtt{M} | \mathfrak{L} \rangle$ is state and $\mathfrak{L}$ is acyclic and $(\mathrm{FV}(\mathtt{M}) - \mathrm{SVar}) \subseteq$

[†] The proviso holds, since we consider only evaluations of terms respecting hygienic conditions.

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

$\mathsf{uDom}(\mathfrak{L})$ then, it is easy to see that $\langle \mathtt{M}|\mathfrak{L}\rangle$ is well formed. Moreover, the evaluation of a well-formed state just involves well-formed states.

Lemma 15. If $\langle \mathtt{M}|\mathfrak{L}_0\rangle$ is a well-formed state and $\langle \mathtt{M}|\mathfrak{L}_0\rangle \Downarrow_{\mathtt{E}} \langle \underline{\mathtt{n}}|\mathfrak{L}_1\rangle$ then $\mathsf{Dom}(\mathfrak{L}_0) = \mathsf{Dom}(\mathfrak{L}_1)$ and all states involved in the evaluation are well formed. Moreover, if $\mathfrak{L}_0 = [\mathtt{f}_1 := \mathtt{N}_1, \ldots, \mathtt{f}_k := \mathtt{N}_k]$ $(k \geqslant 0)$ then $\mathfrak{L}_1 = [\mathtt{f}_1 := \mathtt{N}'_1, \ldots, \mathtt{f}_k := \mathtt{N}'_k]$ and, for each $1 \leqslant i \leqslant k$, one of the following is satisfied:

— if $\mathtt{f}_i$ is ground then $\mathtt{N}_i = \mathtt{N}'_i$ is also ground;
— if $\mathtt{f}_i$ is traced in $\langle \mathtt{M}|\mathfrak{L}_0\rangle$ then $\mathtt{N}_i = \mathtt{N}'_i$ is a numeral;
— if $\mathtt{f}_i$ is not-traced in $\langle \mathtt{M}|\mathfrak{L}_0\rangle$ then, either $\mathtt{N}_i = \mathtt{N}'_i$ (if the evaluation has not required the use of $\mathtt{f}_i$) or $\mathtt{N}'_i$ is a numeral $\underline{\mathtt{m}}$ where $\mathtt{Chk}^{(\sigma)}_{\mathtt{m}}(\mathtt{N}_i) = \underline{\mathtt{0}}$ (in other words $\mathtt{Sgl}^{(\sigma)}_{\mathtt{m}}$ corresponds to a trace in $\mathtt{N}_i$).

Proof. A straightforward induction on evaluation rules. □

Clearly, if $\langle \mathtt{M}|\mathfrak{L}_0\rangle$ is a well-formed state and $\langle \mathtt{M}|\mathfrak{L}_0\rangle \Downarrow_{\mathtt{E}} \langle \underline{\mathtt{n}}|\mathfrak{L}_1\rangle$ then $\mathsf{uDom}(\mathfrak{L}_1) \subseteq \mathsf{uDom}(\mathfrak{L}_0)$.

Definitiona 26. Let $\langle \mathtt{M}|\mathfrak{L}_0\rangle$ be a well-formed state such that $\langle \mathtt{M}|\mathfrak{L}_0\rangle \Downarrow_{\mathtt{E}} \langle \underline{\mathtt{n}}|\mathfrak{L}_1\rangle$. If $\mathfrak{L}_0 = [\mathtt{f}_1 := \mathtt{N}_1, \ldots, \mathtt{f}_k := \mathtt{N}_k]$ and $\mathfrak{L}_1 = [\mathtt{f}_1 := \mathtt{N}'_1, \ldots, \mathtt{f}_k := \mathtt{N}'_k]$ $(k \geqslant 0)$ then, we define $\mathsf{Reify}_{\mathfrak{L}_0}(\mathfrak{L}_1)$ be the environment $[\mathtt{f}_1 := \mathtt{N}''_1, \ldots, \mathtt{f}_k := \mathtt{N}''_k]$ such that, for all $1 \leqslant i \leqslant k$,

— if $\mathtt{N}'_i = \underline{\mathtt{m}}$ then $\mathtt{N}''_i$ is defined as $\mathtt{Sgl}^{(\sigma)}_{\mathtt{m}}$,
— otherwise, if $\mathtt{N}_i = \mathtt{N}'_i$ then $\mathtt{N}''_i$ is defined as $\mathtt{N}_i$.

The last definition allows us to express formally, the deep relation between $\mathfrak{L}_0$ and $\mathfrak{L}_1$.

Proposition 5. Let $\langle \mathtt{M}|\mathfrak{L}_0\rangle$ be a well-formed state.
If $\langle \mathtt{M}|\mathfrak{L}_0\rangle \Downarrow_{\mathtt{E}} \langle \underline{\mathtt{n}}|\mathfrak{L}_1\rangle$ and $\mathfrak{L}_2 = \mathsf{Reify}_{\mathfrak{L}_0}(\mathfrak{L}_1)$ then $\langle \mathtt{M}|\mathfrak{L}_2\rangle \Downarrow_{\mathtt{E}} \langle \underline{\mathtt{n}}|\mathfrak{L}_1\rangle$.

Proof. Easy, by induction on the derivation proving $\langle \mathtt{M}|\mathfrak{L}_0\rangle \Downarrow_{\mathtt{E}} \langle \underline{\mathtt{n}}|\mathfrak{L}_1\rangle$. □

Finally, we give the formal correspondence between the two operational evaluation.

Theorem 11. Let $\langle \mathtt{M}|\mathfrak{L}_0\rangle$ be well formed.

1. If $\mathfrak{L}_0(\!|\mathtt{M}|\!) \Downarrow \underline{\mathtt{n}}$ then $\langle \mathtt{M}|\mathfrak{L}_0\rangle \Downarrow_{\mathtt{E}} \langle \underline{\mathtt{n}}|\mathfrak{L}_1\rangle$.
2. If $\langle \mathtt{M}|\mathfrak{L}_0\rangle \Downarrow_{\mathtt{E}} \langle \underline{\mathtt{n}}|\mathfrak{L}_1\rangle$ then $\mathfrak{L}_0(\!|\mathtt{M}|\!) \Downarrow \underline{\mathtt{n}}$.

Proof. Both directions follow by induction on given derivations. □

8. Conclusions and future works

The results presented in this paper are part of a wider project (started with the works of Gaboardi and Paolini (2007) and Paolini and Piccolo (2008)), aiming to extend the expressive power of linear programming languages by means of the reification of linear functions between suitable domains. On the one hand, S$\not\partial$PCF$_\star$ provides an interesting programming language, because the efficient evaluation machine presented in this paper. On the other hand, in the usual reductionist mainstream of science, S$\not\partial$PCF$_\star$ can be considered as an over-simplified paradigmatic calculus that can be used to tackle complex

https://doi.org/10.1017/S0960129514000401 Published online by Cambridge University Press

issues that can (possibly) be extended to non-strictly linear settings. In this line, we leave as future works to look for a sharp way to compile the whole PCF language in S⁄PCF$_\star$. An interesting direction is to extend the results obtained in this paper in order to prove the universality of S⁄PCF$_\star$ with respect to the linear model, i.e. to find the language able to define all the *recursive cliques* of the model. Another interesting direction is the study of S⁄PCF$_\star$ semantics with respect to other model notions.

Acknowledgements

This work has been supported by the LINTEL project (grant number TO_Call1_2012_0085), funded by the Compagnia di San Paolo. Marco Gaboardi has been supported by the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement no. 272487.

References

Abramsky, S., Malacaria, P. and Jagadeesan, R. (2000) Full abstraction for PCF. *Information and Computation* **163** (2) 409–470. (An extended abstract can be found in Theoretical Aspects of Computer Software (Sendai, 1994), *Lecture Notes in Computer Science*, **789** 1–15, Springer-Verlag, Berlin, 1994.)

Abramsky, S. and McCusker, G. (1996) Linearity, sharing and state: A fully abstract game semantics for idealized algol with active expressions. *Electronic Notes in Theoretical Computer Science* **3** 2–14.

Alves, S., Fernández, M., Florido, M. and Mackie, I. (2006) The power of linear functions. In: Ésik, Z. (ed.) Proceedings of the 20th International Workshop on Computer Science Logic, 15th Annual Conference of the European Association for Computer Science Logic, Szeged, Hungary, 25–29 September. *Springer-Verlag Lecture Notes in Computer Science* **4207** 119–134.

Alves, S., Fernández, M., Florido, M. and Mackie, I. (2007) Linear recursive functions. In: Comon-Lundh, H., Kirchner, C. and Kirchner, H. (eds.) Rewriting, Computation and Proof, Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of His 60th Birthday. *Springer-Verlag Lecture Notes in Computer Science* **4600** 182–195.

Alves, S., Fernndez, M., Florido, M. and Mackie, I. (2010) Gödel's system T revisited. *Theoretical Computer Science* **411** (11–13) 1484–1500.

Augustsson, L. (1987) *Compiling Lazy Functional Languages, Part II*, Ph.D. thesis, Department of Computer Sciences, Chalmers University of Technology, Goteborg, Sweden.

Berry, G. (1978) Stable models of typed λ-calculi. In: Ausiello, G. and Böhm, C. (eds.) Fifth International Colloquium on Automata, Languages and Programming. (Udine, Italy, July 17–21). *Springer-Verlag Lecture Notes in Computer Science* **62** 72–89.

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

Bierman, G. M. (2000) Program equivalence in a linear functional language. *Journal of Functional Programming* **10** (2) 167–190.

Bierman, G. M., Pitts, A. M. and Russo, C. V. (2000) Operational properties of lily, a polymorphic linear lambda calculus with recursion. *Electronic Notes in Theoretical Computer Science* **41** (3) 70–88.

Bucciarelli, A., Carraro, A., Ehrhard, T. and Salibra, A. (2010) On linear information systems. In: LINEARITY'09, Satellite Workshop of Computer Science Logic 2009 *Electronic Proceedings in Theoretical Computer Science* **22** 38–48.

Bucciarelli, A. and Ehrhard, T. (1991) Sequentiality and strong stability. In: *Proceedings of the Symposium on Logic in Computer Science* 138–145.

Bucciarelli, A. and Ehrhard, T. (1993) A theory of sequentiality. *Theoretical Computer Science* **113** (2) 273–291.

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

Curien, P.-L. (2007) Definability and full abstraction. *Electronic Notes in Theoretical Computer Science* **172** 301–310.

Cutland, N. (1980) *Computability: An Introduction to Recursive Function Theory*, Cambridge University Press.

Davis, M. and Weyuker, E. J. (1983) *Computability, Complexity and Languages*, Computer Science and Applied Mathematics, Academic Press.

de Bruijn, N. G. (1972) Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church–Rosser theorem. *Indagationes Mathematicae (Proceedings)* **75** (5) 381–392.

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

Gaboardi, M. and Paolini, L. (2007) Syntactical, operational and denotational linearity. In: *Workshop on Linear Logic, Ludics, Implicit Complexity and Operator Algebras. Dedicated to Jean-Yves Girard on his 60th birthday*. Certosa di Pontignano, Siena.

Gaboardi, M., Paolini, L. and Piccolo, M. (2011) Linearity and PCF: a semantic insight! In: Chakravarty, M. M. T., Hu, Z. and Danvy, O. (eds.) *Proceeding of the 16th Association for Computing Machinery's Special interest Group on Programming Language, International Conference on Functional Programming* Tokyo, Japan 372–384.

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

Girard, J.-Y. (1987) Linear logic. *Theoretical Computer Science* **50** 1–102.

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

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

Hindley, J. R. (1989) BCK-combinators and linear λ-terms have types. *Theoretical Computer Science* **64** 97–105.

Huth, M. (1993) Linear domains and linear maps. In: Brookes, S. D., Main, M. G., Melton, A., Mislove, M. W. and Schmidt, D. A. (eds.) Proceedings of Mathematical Foundations of Programming Semantics, 9th International Conference, (New Orleans, LA, USA, April 7–10). *Springer Lecture Notes in Computer Science* **802** 438–453.

Hyland, J. M. E. and Ong, L. C.-H. (2000) On full abstraction for PCF: I, II, and III. *Information and Computation* **163** (2) 285–408.

Johnsson, T. (1985) Lambda lifting: Transforming programs to recursive equations. In: Jouannaud, J.-P. (ed), Functional Programming Languages and Computer Architecture. *Lecture Notes in Computer Science* **201** 190–203.

Johnsson, T. (1987) *Compiling Lazy Functional Languages*, Ph.D. thesis, Department of Computer Sciences, Chalmers University of Technology, Goteborg, Sweden.

Klop, J. W. (2007) New fix point combinators from old. In: Barendsen, E., Capretta, V., Geuvers, H. and Niqui, M. (eds.) *Reflections on Type Theory*, Radboud University Nijmegen.

Longley, J. R. (2000) Notions of computability at higher types I. In: Cori, R., Razborov, A., Todorcevic, S. and Wood, C. (eds.) Proceedings of the Annual European Summer Meeting of the Association for Symbolic Logic – Logic Colloquium. *Lecture Notes in Logic* **19** 32–142, Paris, France. Association for Symbolic Logic.

Longley, J. R. (2002) The sequentially realizable functionals. *Annals of Pure and Applied Logic* **117** 1–93.

Normann, D. (2006) Computing with functionals - computability theory or computer science? *Bulletin of Symbolic Logic* **12** (1) 43–59.

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

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

Paolini, L. and Piccolo, M. (2008) Semantically linear programming languages. In: Antoy, S. and Albert, E. (eds.) *Proceedings of the 10th International Association for Computing Machinery's Special interest Group on Programming Languages Conference on Principles and Practice of Declarative Programming*, Valencia, Spain, ACM 97–107.

Plotkin, G. D. (1977) LCF considered as a programming language. *Theoretical Computer Science* **5** 225–255.

Walker, D. (2005) Substructural type systems. In: *Advanced Topics in Types and Programming Languages*. The MIT Press.