Linearity and PCF: a Semantic Insight!

Marco Gaboardi
Dip. di Scienze dell'Informazione, Università di Bologna, INRIA Focus Team, Mura Anteo Zamboni 7, 40127 Bologna, Italy
gaboardi@cs.unibo.it

Luca Paolini
Dipartimento di Informatica, Università degli Studi di Torino - Corso Svizzera 185, 10149 Torino, Italy
paolini@di.unito.it

Mauro Piccolo
Dipartimento di Informatica, Università degli Studi di Torino - Corso Svizzera 185, 10149 Torino, Italy
piccolo@di.unito.it

Abstract

Linearity is a multi-faceted and ubiquitous notion in the analysis and the development of programming language concepts. We study linearity in a denotational perspective by picking out programs that correspond to linear functions between coherence spaces.

We introduce a language, named $\mathcal{SL}\mathrm{PCF}_\star$, that increases the higher-order expressivity of a linear core of PCF by means of new operators related to exception handling and parallel evaluation. $\mathcal{SL}\mathrm{PCF}_\star$ allows us to program all the finite elements of the model and, consequently, it entails a full abstraction result that makes the reasoning on the equivalence between programs simpler.

Denotational linearity provides also crucial information for the operational evaluation of programs. We formalize two evaluation machineries for the language. The first one is an abstract and concise operational semantics designed with the aim of explaining the new operators, and is based on an infinite-branching search of the evaluation space. The second one is more concrete and it prunes such a space, by exploiting the linear assumptions. This can also be regarded as a base for an implementation.

Categories and Subject Descriptors D.3.1 [*Programming Languages*]: Formal Definitions and Theory—Semantics, Syntax; D.3.3 [*Programming Languages*]: Language Constructs and Features—Control structures; F.3.2 [*Logics and meanings of programs*]: Semantics of programming languages—Denotational semantics, Operational semantics

General Terms Languages, Theory, Design

Keywords PCF, Linear Logic, Denotational Semantics, Operational Semantics

1. Introduction

Linearity is a key tool in order 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 literature is [34]. This broad spectrum of applications highlights the fact that linearity is a multifaceted 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. [4, 21]. On the other hand, if redexes cannot be discarded or duplicated during reduction [21] then a kind of **operational linearity** is achieved. This is related to the notion of *simple term* [24] in λ-calculus which suggests a kind of linearity on reductions, unrelated from a specific strategy.

Although some ideas that can be tracked to linearity have been implicitly used in programming languages for many years, the introduction of linear logic [19] 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 [13, 14, 19, 22]. By tackling this correspondence minutely some important contributions to the theory of programming can be obtained. If the considered domain includes all the computable functions (usually, they do it), then the analysis provides Turing-complete languages with weak syntactic linear constraints on variables, and new linear operators that, in a higher-type computability perspective [25, 27] increase the expressivity of linear languages.

To advance in this research line, we aim to pick out all and only (recursive) linear functions in a **linear model** built as the full subcategory of coherence spaces and linear functions [19] identified by the type structure of numerals and arrows. The language $\mathcal{SL}\mathrm{PCF}$ proposed in [30] is correct for this linear model, so it is denotationally linear. Moreover, it grasps a limited completeness, namely $\mathcal{SL}\mathrm{PCF}$ is sufficient to assure the definability of all the tokens ("prime elements" in domain terms) of the linear model. From this follows a restricted full abstraction result for terms without free stable variables (i.e. variables used for recursion). A more general result was erroneously claimed in [30, Corollary 3]; see Section 3 for more details.

In this paper we propose $\mathcal{SL}\mathrm{PCF}_\star$ a language extending $\mathcal{SL}\mathrm{PCF}$ by the operator ℓ`et-`ℓ`or`. This operator provides a linear counterpart of the `gor` operator introduced in [29], and increases the nondeterministic expressiveness of the language. Indeed, this extension allows us to gain a **finite definability** result [15], i.e. the definability of all the finite cliques of the considered linear model. In fact, a crucial role in the proof of our finite definability result is played not only by the ℓ`et-`ℓ`or` operator but also by the $\mathcal{SL}\mathrm{PCF}$ operator `which`?. In [30], the `which`? operator has been proposed as an

ICFP'11, September 19–21, 2011, Tokyo, Japan.

example of interesting higher-order linear operator providing runtime information, in order to give a flavor of our research line. However, it was not necessary for the token-definability and the limited full abstraction results. Here, we show how to use `which`? (together with the new ℓ`et`-ℓ`or` operator) in order to reach the finite definability.

By using the finite definability, we prove that $\mathcal{SL}\mathrm{PCF}_\star$ is **fully abstract** with respect to the considered linear model. That is, the *operational equivalence* coincides with the *denotational equivalence*. This result allows us to reason on programs in a compositional way. This is important because even if the operational equivalence is defined for closed ground contexts, since it relies on the operational formal machine evaluating programs, sometimes one wants just to replace a subterm with another one and preserve the equivalence. So, we need to consider also open sub-terms that form our programs. In our case, tackling the equivalence also when terms contain open variables is particularly challenging due to the presence of stable variables, used for recursion. Besides, we prove the coincidence of three different definitions of operational equivalence that make simpler the reasoning on the equivalence between programs by permitting to consider only contexts of a restricted shape. Moreover, the proof of this result uses non-trivial syntactical and denotational arguments that are to our knowledge new and of wider interest.

We remark that $\mathcal{SL}\mathrm{PCF}_\star$ is neither syntactically linear nor operationally linear in a tight sense, albeit its finitary fragment (the set of programs which does not involve recursion) enjoys some syntactical and operational forms of linearity. For instance, it is syntactically linear for slices, and when only slices are considered it can be evaluated without duplicating redexes. Besides, we conclude the paper by giving an operational semantics inducing an efficient evaluation of $\mathcal{SL}\mathrm{PCF}_\star$ terms. This operational semantics traces out and records linear information to drastically prune the infinite branching search tree of the evaluation of $\mathcal{SL}\mathrm{PCF}_\star$.

Outline In Section 2 we introduce in an informal way the contributions that will be technically presented in the rest of the paper. In Section 3 we introduce the background needed to understand the technical results of this paper. In Section 4 we show that $\mathcal{SL}\mathrm{PCF}$ lacks the full abstraction. In Section 5 we introduce the language $\mathcal{SL}\mathrm{PCF}_\star$ and we give some programming examples. In Section 6 we prove the finite definability and the full abstraction for $\mathcal{SL}\mathrm{PCF}_\star$. In Section 7 we show the coincidence of the three operational equivalence introduced. In Section 8 we give an abstract machine for $\mathcal{SL}\mathrm{PCF}_\star$ that traces the linear use of terms and it provides the base for an efficient implementation of our language.

2. Contributions: An Informal Account

In this paper, we propose an extension of the language $\mathcal{SL}\mathrm{PCF}$ introduced in [30]. $\mathcal{SL}\mathrm{PCF}$ is a PCF-like language enriched with the `which`? operator, that is correct for the considered linear model. In particular, linearity is obtained by means of some constraints on clever variable management. $\mathcal{SL}\mathrm{PCF}$ is based on three kinds of variables: ground $\mathtt{x}^\iota$ and stable variables $F^{\sigma\multimap\tau}$, that can be weakened and contracted, and linear ones $\mathtt{f}^{\sigma\multimap\tau}$ that cannot. Ground and linear variables can be λ-abstracted, stable variables cannot. Stable variables can be bound by a dedicated binder (the μ-abstraction) and they are used to define recursive functions. In $\mathcal{SL}\mathrm{PCF}$, an argument `N` supplied to $\lambda\mathtt{x}.\mathtt{M}$ is evaluated by using a call-by-value policy in case `x` is ground, a call-by-name policy otherwise. More details can be found in Section 3.2.

The denotational insights

The basic components of the model are *tokens*. A token is a tuple of natural numbers that complies with the structure of types. For instance

$$\begin{array}{c}((\iota\multimap\iota)\multimap\iota)\multimap(\iota\multimap\iota)\multimap\iota\\ \big(((0\ ,\ 1)\ ,\ 2)\ ,\ (3\ ,\ 4)\ ,\ 5\big)\end{array}$$

The second row of the formula above describes a token belonging to the coherence space corresponding to the type written in the first row (clearly, ι is the type of natural numbers while $\multimap$ is the linear arrow). $\mathcal{SL}\mathrm{PCF}$ is able to define all the tokens which belong to coherence spaces corresponding to the types [30]. For instance, the following term defines the token written above:

$$\lambda\mathtt{f}.\lambda\mathtt{g}.\ell\mathtt{if}\ \Big(\big(\mathtt{f}\big(\lambda\mathtt{x}.\ell\mathtt{if}\ (\mathtt{x}\doteq\underline{0})\ \underline{1}\ \Omega\big)\doteq\underline{2}\big)\ \mathtt{and}\ (\mathtt{g}\ \underline{3}\doteq\underline{4})\Big)\ \underline{5}\ \Omega\,.$$

To program a token means to verify that each input coincides with the one described by the token, in the example there are two input (i.e. $((0,1),2)$ and $(3,4)$).

The elements of the linear model as usual in coherence spaces are *cliques*, i.e. sets of *coherent* tokens. The coherence relation ensures that cliques describes only *traces* of functions (where the trace is an economic way to describe the graph of a function). In particular, coherence establishes when two tokens can coexists in the trace of a function. Consider a simple clique of a linear function:

$$\left\{\begin{array}{l}\big((0,0)\,,\,((5,7)\,,\,0)\,,\,0\big)\\ \big((2,0)\,,\,((3,9)\,,\,1)\,,\,1\big)\\ \big((2,1)\,,\,((3,9)\,,\,1)\,,\,2\big)\end{array}\right\}$$

Above three tokens have two input arguments, respectively of type $\iota\multimap\iota$ and $(\iota\multimap\iota)\multimap\iota$. Coherence ensures that, for each pair of tokens, there exists an input argument allowing to decide what is the unique token being (eventually) involved in the computation. For instance, by applying the first argument to $\underline{2}$ we can distinguish between the second and the third token, in fact the subtokens $(2,0)$ and $(2,1)$ are incoherent indeed they cannot coexist in the same function. Likewise, by applying the second argument to $\lambda\mathtt{x}^\iota.\ell\mathtt{if}\ (\mathtt{x}\doteq\underline{5})\ \underline{7}\ (\ell\mathtt{if}\ (\mathtt{x}\doteq\underline{3})\ \underline{9}\ \Omega)$, we can distinguish the first token from both the second and the third. Indeed $((3,9),1)$ and $((5,7),0)$ are incoherent and they cannot coexist in the same function.

These properties are important in order to program linear functions, however, linearity gives no information about how to locate such observations. In particular, when higher-order types are considered this becomes quite tricky, and parallelism is necessary.

Is $\mathcal{SL}\mathrm{PCF}$ able to program (at least) all the finite cliques?

Anticipating, we say immediately that the answer is negative: $\mathcal{SL}\mathrm{PCF}$ allows us to program some finite cliques but not all of them. The extension we propose in this paper however fills this gap. That is, $\mathcal{SL}\mathrm{PCF}_\star$ permits one to program all the finite cliques.

Let us now go in more details. $\mathcal{SL}\mathrm{PCF}$ is Turing-complete. This means that it is able to program all the first order (computable) cliques. For instance denoting with $*$ the multiplication between numerals, the following term

$$\mu F.\lambda\mathtt{x}.\ell\mathtt{if}\ (\mathtt{x}\doteq\mathtt{0})\ \mathtt{1}\ (\mathtt{x}*(F\,(\mathbf{p}\mathtt{x}^\iota)))$$

defines the clique $\{(n,\mathtt{factorial}(n))\mid n\in\mathbf{N}\}$. So at the first order, $\mathcal{SL}\mathrm{PCF}$ is able to program not only all the finite cliques but all the computable infinite ones. When higher-order functions are considered, however, the situation becomes more difficult. For instance, if we want to define the following clique

$$\left\{\begin{array}{l}\big(((0,3)\,,\,0)\,,\,1\big)\\ \big(((1,4)\,,\,0)\,,\,2\big)\end{array}\right\}$$

we cannot discriminate the two given tokens just by looking at the result of the evaluation of the argument applied to the term $\mathtt{N}=\lambda\mathtt{x}.\ell\mathtt{if}\ (\mathtt{x}\doteq\underline{0})\ \underline{3}\ (\ell\mathtt{if}\ (\mathtt{x}\doteq\underline{1})\ \underline{4}\ \Omega)$, because it corresponds to $\underline{0}$ for both tokens. In order to discriminate them, we need to grasp

some intensional aspect of the above evaluation, more specifically, we need to identify the value passed to $\mathtt{N}$ (i.e. $\underline{0}$ in the first token and $\underline{1}$ in the second). Information like this can be retrieved by an appropriate use of the `which`? operator, introduced in [30]. The `which`? operator corresponds to a primitive form of exception handling: it allows to obtain besides the result of an evaluation also the information on what has been used during that evaluation. More examples are in Section 5.1.

The `which`? operator increases the expressivity of the language. However, there are still some finite cliques that cannot be programmed in $\mathcal{S}\ell\mathrm{PCF}$. For instance,

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

The reason is that we cannot identify *a priori* which is the observation that should be done. In particular, we are not able to determine which argument we should observe first and what is the value we should supply to it. Suppose we want to realize the program defining the clique in Equation 1, writing a term of the following shape

$$\lambda \mathtt{f}_1.\lambda \mathtt{f}_2.\lambda \mathtt{f}_3.\Big(\lambda \mathtt{x}.\ell\mathtt{if}\ (\mathtt{x} \doteq \underline{0})\ \mathtt{P}_1\ \big(\ell\mathtt{if}\ (\mathtt{x} \doteq \underline{1})\ \mathtt{P}_2\ \mathtt{P}_3\big)\Big)(\mathtt{f}_1\underline{0})$$

This means that we start the observation from the first argument $\mathtt{f}_1$ and we apply $\underline{0}$ to it. This term diverges if we supply as first argument a term defining $(1,0)$ and any other term as second and third argument. However, this behavior can be admitted by the third token. Something similar happens if we start the observation from an other argument and/or if we choose to apply $\underline{1}$ instead of $\underline{0}$ to it. Such a behavior would be not desirable.

To solve this circularity, we need to make available a parallel operator. This kind of parallelism is well known in *stable domains*, where so called Gustave's functions exists [6, 8]. Following [29], we could add a *gustave-or* operator where linearity is carefully forced. That is, a gustave-or typed as follows:

$$\frac{\Gamma \vdash \mathtt{M}_1 : \iota \quad \Gamma \vdash \mathtt{M}_2 : \iota \quad \Gamma \vdash \mathtt{M}_3 : \iota}{\Gamma \vdash \mathtt{Gor}\ (\mathtt{M}_1, \mathtt{M}_2, \mathtt{M}_3) : \iota}$$

with the provision that the basis Γ does not contain linear variables. This operator can then be equipped by the following semantics:

$$\frac{\mathtt{M}_1 \Downarrow \underline{0} \quad \mathtt{M}_2 \Downarrow \underline{\mathtt{n}+1}}{\mathtt{Gor}\ (\mathtt{M}_1,\mathtt{M}_2,\mathtt{M}_3) \Downarrow \underline{\mathtt{n}}} \quad \frac{\mathtt{M}_2 \Downarrow \underline{0} \quad \mathtt{M}_3 \Downarrow \underline{\mathtt{n}+1}}{\mathtt{Gor}\ (\mathtt{M}_1,\mathtt{M}_2,\mathtt{M}_3) \Downarrow \underline{\mathtt{n}}} \quad \frac{\mathtt{M}_3 \Downarrow \underline{0} \quad \mathtt{M}_1 \Downarrow \underline{\mathtt{n}+1}}{\mathtt{Gor}\ (\mathtt{M}_1,\mathtt{M}_2,\mathtt{M}_3) \Downarrow \underline{\mathtt{n}}}$$

Adding it to $\mathcal{S}\ell\mathrm{PCF}$ however is not sufficient to program the clique defined in Equation 1 (actually, to extend $\mathcal{S}\ell\mathrm{PCF}$ by `Gor` does not add any linear function!). We need something enabling a limited form of contraction on linear variable to permit parallel observations of the same linear variable on different arguments. One way to do this is by introducing a further control operator, that can be described as a `let`-like operator:

$$\frac{\Gamma \vdash \mathtt{N} : \sigma \multimap \tau \qquad \mathit{F} : \sigma \multimap \tau, \Delta \vdash \mathtt{M} : \iota}{\Gamma, \Delta \vdash \mathtt{let}\ \mathit{F}^{\sigma\multimap\tau} = \mathtt{N}\ \mathtt{in}\ \mathtt{M} : \iota}$$

where the variable F can be weakened and contracted, i.e. it does not respect any occurrence constraint. The operational meaning of this `let`-like operator can be described by the following rule:

$$\frac{[\![\mathtt{P}]\!] = \{a\} \subseteq [\![\mathtt{N}]\!] \qquad \mathtt{M}[\mathtt{P}/\mathit{F}] \Downarrow \underline{\mathtt{n}} \qquad \mathtt{M}[\Omega/\mathit{F}] \Uparrow}{\mathtt{let}\ \mathit{F}^{\sigma\multimap\tau} = \mathtt{N}\ \mathtt{in}\ \mathtt{M} \Downarrow \underline{\mathtt{n}}}$$

Such an operator appears to have the flavor of the co-dereliction of Differential Linear Logic [16]. The idea motivating the above rule is that:

> *Denotational linearity* allow several observations of a *clique*, as long as they are performed on the *same token*.

Following this intuition, the side condition $[\![\mathtt{P}]\!] = \{a\} \subseteq [\![\mathtt{N}]\!]$ says that $\mathtt{P}$ is interpreted as the unique token of $\mathtt{N}$ that can be so observed several time in evaluating $\mathtt{M}[\mathtt{P}/\mathit{F}]$. The third condition $\mathtt{M}[\Omega/\mathit{F}] \Uparrow$ instead is here just to ensure that the token is actually really used.

A clever combination of the two above operators allows to program the clique described in Equation 1 (such combination will be detailed in Section 5.1). We remark some drawbacks of `let`.

1. Its operational rule given above is not effective, due to the presence of the third condition $\mathtt{M}[\Omega/\mathit{F}] \Uparrow$.
2. It limits the understanding of the program control flow. It can be checked only at run-time that the same token has been observed.

The first drawback can be solved by designing an ad-hoc operational semantics, the second one is more problematic since it could make the understanding of programs really problematic. For these two reasons, we extend $\mathcal{S}\ell\mathrm{PCF}$ by a single operator, combining together `Gor` and `let`. The ℓ`et`-ℓ`or` (considering only a linear variable `f`) is typed as follows:

$$\frac{\Gamma \vdash \mathtt{N} : \sigma \quad \mathtt{f} : \sigma, \Delta \vdash \mathtt{M}_1 : \iota \quad \mathtt{f} : \sigma, \Delta \vdash \mathtt{M}_2 : \iota \quad \mathtt{f} : \sigma, \Delta \vdash \mathtt{M}_3 : \iota}{\Gamma, \Delta \vdash \underline{\ell\mathtt{et}}\ \mathtt{f}^{\sigma} = \mathtt{N}\ \underline{\mathtt{in}\ell\mathtt{or}}\ \mathtt{M}_1\ \mathtt{M}_2\ \mathtt{M}_3 : \iota}$$

again with the provision that the basis Δ does not contain linear variables. Note that now the $\underline{\ell\mathtt{et}}$-bounded variable `f` is linear, albeit used in three program-branches. The ℓ`et`-ℓ`or` evaluation can now be described using rules as follows:

$$\frac{\mathtt{M}_1[\mathtt{P}/\mathtt{f}] \Downarrow \underline{0} \quad \mathtt{M}_2[\mathtt{P}/\mathtt{f}] \Downarrow \underline{\mathtt{n}+1} \quad [\![\mathtt{P}]\!] = \{a\} \subseteq [\![\mathtt{N}]\!]}{\underline{\ell\mathtt{et}}\ \mathtt{f}^{\sigma\multimap\tau} = \mathtt{N}\ \underline{\mathtt{in}\ell\mathtt{or}}\ \mathtt{M}_1\ \mathtt{M}_2\ \mathtt{M}_3 \Downarrow \underline{\mathtt{n}}}$$

(we give just one rule, the other two are analogous). Thanks to a generalized ℓ`et`-ℓ`or` operator, the resulting language $\mathcal{S}\ell\mathrm{PCF}_\star$ permits to program all the finite cliques. So, the *full abstraction* follows.

The operational Insights

The full abstraction result mentioned above ensures as usual that a *compositional* theory of program equivalence can be defined. That is, program equivalence is a congruence. Concretely, the full abstraction is proved with respect to a non-standard notion of contextual equivalence $\sim$, named *fix-point equivalence* defined as:

$$\mathtt{M} \sim \mathtt{N} \quad \text{iff} \quad \mathtt{C}[\mathtt{M}[\vec{\mathtt{P}^\sigma}/\vec{\mathit{F}^\sigma}]] \Downarrow \underline{\mathtt{n}} \iff \mathtt{C}[\mathtt{N}[\vec{\mathtt{P}^\sigma}/\vec{\mathit{F}^\sigma}]] \Downarrow \underline{\mathtt{n}}$$

where all the $\mathtt{P}_i^{\sigma_i}$ are closed terms. This notion of equivalence makes explicit the fact that for reasoning in a compositional way about programs we need to permit to substitute general terms to stable variables, even if such variables are only used for dealing with recursion. It is exactly to program the $\mathtt{P}_i$ that the definability of all the finite cliques is needed in order to get full abstraction. Indeed, in [30] a full abstraction result for terms without open stable variables was proved. Anyway, the definition of fix-point equivalence seems ad-hoc with respect to the usual notion of contextual equivalence:

$$\mathtt{M} \approx \mathtt{N} \quad \text{iff} \quad \mathtt{C}[\mathtt{M}] \Downarrow \underline{\mathtt{n}} \iff \mathtt{C}[\mathtt{N}] \Downarrow \underline{\mathtt{n}}$$

So, it is natural to consider the following question:

> Do the relations $\sim$ and $\approx$ coincide?

Anticipating, the answer to this question is positive. Although, proving this result is quite technical. Intuitively, the reason is that stable variables can only be μ-abstracted. More precisely, suppose $\mathtt{M} \not\sim \mathtt{N}$ and suppose they have a free stable variable F of type $\sigma_1 \multimap \ldots \multimap \sigma_k \multimap \iota$. We have that there is a context $\mathtt{C}$ and a term $\mathtt{P}$ such that $\mathtt{C}[\mathtt{M}[\mathtt{P}/\mathit{F}]] \Downarrow \underline{\mathtt{n}}$ and $\mathtt{C}[\mathtt{N}[\mathtt{P}/\mathit{F}]] \not\Downarrow \underline{\mathtt{n}}$. Intuitively, to prove that also $\mathtt{M} \not\approx \mathtt{N}$ holds, one could think to build a (pseudo)-context

$$\mathtt{C}' = \mathtt{C}[(\lambda \mathit{F}.[\,])\mathtt{P}]$$

such that $\mathtt{C}'[\mathtt{M}] \Downarrow \underline{\mathtt{n}}$ and $\mathtt{C}'[\mathtt{N}] \not\Downarrow \underline{\mathtt{n}}$. Unfortunately, this context cannot be built since the stable variable F cannot be λ-abstracted but can only μ-abstracted. So, the best that one can hope to get is a context

$$\mathtt{C}'' = \mathtt{C}_1[\mu F.\mathtt{C}_2[\,]]$$

acting similarly to $\mathtt{C}'$. We will show in Section 7 how to build a such context. Here, we propose an example. Suppose that M and N can be distinguished in the empty context $[.]$, by using P (for simplicity, we assume F, P typed $\iota \multimap \iota$, and $[\![\mathtt{P}]\!] = \{(n,m)\}$ with $n \neq 0$). So, we consider a term $\mathtt{P}' : \iota \multimap \iota$ defining the clique $\{(0,0),(n,m)\}$. We can define $\mathtt{C}''$ as

$$\mu F.\lambda \mathtt{y}.\Big(\big(\lambda \mathtt{x}^{\iota}.\ell\mathtt{if}\ (\mathtt{x} \doteq \underline{0})\ [.]\big(\ell\mathtt{if}\ (\mathtt{x} \doteq \underline{\mathtt{m}})\ \underline{\mathtt{m}}\ \Omega\big)\big)(\mathtt{P}'\ \mathtt{y})\Big)$$

Therefore, this context can be used to build the terms $\mathtt{C}''[\mathtt{N}]\underline{0}$ and $\mathtt{C}''[\mathtt{M}]\underline{0}$. After one recursion step, the terms $\mathtt{N}[\mathtt{C}''[\mathtt{N}]/F]$ and $\mathtt{M}[\mathtt{C}''[\mathtt{M}]/F]$ are obtained. However, in the next recursive step $\mathtt{C}''[\mathtt{N}]$ and $\mathtt{C}''[\mathtt{M}]$ will behave exactly as P and so the two terms can be distinguished. One can doubt that this construction cannot be always done. However, the properties of the linear model ensure that finite cliques are never *maximal* with respect to the set theoretical inclusion. This for instance does not happen in *stable models* in general. This means that given a finite clique x one can always find a *new coherent token* that can be added to it and that can be used to control the recursion. The proof just generalizes this example.

Besides the equivalence of programs, *linearity* provides also crucial information for the operational evaluation of programs. As instance, the side condition $[\![\mathtt{P}]\!] = \{a\} \subseteq [\![\mathtt{N}]\!]$ in the rule of ℓet-ℓor above assumes the existence of such a P, but it does not give any hints on its search. If we face a concrete implementation of $\mathcal{S}\ell\mathrm{PCF}_\star$ then this could be a problem. An exhaustive search of such a token a is intrinsically inefficient. So, it is natural to consider the following question:

> Is there a reasonable way to drive the evaluation of $\mathcal{S}\ell\mathrm{PCF}_\star$ programs?

The answer to this question as we will show in Section 8 is positive, and again *linearity* comes in our help. It ensures two important properties: first that such a token a exists, and second that it is unique. From these properties we can devise a finer implementation of the language. Consider again the rule

$$\frac{\mathtt{M}_1[\mathtt{P}/\mathtt{f}] \Downarrow \underline{0} \quad \mathtt{M}_2[\mathtt{P}/\mathtt{f}] \Downarrow \underline{\mathtt{n}+\mathtt{1}} \quad [\![\mathtt{P}]\!] = \{a\} \subseteq [\![\mathtt{N}]\!]}{\underline{\ell\mathtt{et}}\ \mathtt{f}^{\sigma\multimap\tau} = \mathtt{N}\ \underline{\mathtt{in}\ell\mathtt{or}}\ \mathtt{M}_1\ \mathtt{M}_2\ \mathtt{M}_3 \Downarrow \underline{\mathtt{n}}}$$

Instead of evaluating $\mathtt{M}_i[\mathtt{P}/\mathtt{f}]$ one can think to evaluate $\mathtt{M}_i$ in an environment e storing the information about variables. So, we associate f to the term N. When the term N is used, we record its observed token a (conveniently encoded by a term P). In particular, the above ℓet-ℓor rule becomes, roughly:

$$\frac{\langle \mathtt{M}_1 | e_0[\mathtt{f} := \mathtt{N}]\rangle \Downarrow_{\mathtt{T}} \langle \underline{0} | e_1[\mathtt{f} := \mathtt{P}]\rangle \quad \langle \mathtt{M}_2 | e_1[\mathtt{f} := \mathtt{P}]\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{m}} | e_2\rangle}{\langle \underline{\ell\mathtt{et}}\ \mathtt{f} = \mathtt{N}\ \underline{\mathtt{in}\ell\mathtt{or}}\ \mathtt{M}_1\ \mathtt{M}_2\ \mathtt{M}_3 | e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{m}} | e_2\rangle}$$

Note that the above reasoning relies on an effective tracing of all evaluated terms. For instance an evaluation of the term $(\lambda \mathtt{x}^{\iota}.\mathtt{M})\mathtt{Q}$ is done using a rule as

$$\frac{\langle \mathtt{Q} | e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{m}} | e_1\rangle \quad \langle \mathtt{M} | e_1[\mathtt{x} := \underline{\mathtt{m}}]\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}} | e_2\rangle}{\langle \mathtt{fQ} | e_0[\mathtt{f} := \lambda \mathtt{x}^{\iota}.\mathtt{M}']\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}} | e_2[\mathtt{f} := \langle \underline{\mathtt{m}}, \underline{\mathtt{n}}\rangle]\rangle}$$

that traces the information of the used token, i.e. it puts $\langle \underline{\mathtt{m}}, \underline{\mathtt{n}}\rangle$ into the environment. A tracing evaluation of $\mathcal{S}\ell\mathrm{PCF}_\star$ programs will be devised in Section 8. Note that this kind of evaluation can be considered a kind of linear call-by-need evaluation where only one evaluation is done and where further observations are used to check the consistency of the information.

Synopsis

In summary, the key contributions of this paper are:

- The definition of a new linear operator ℓet-ℓor (Definition 13) that permits to establish a bridge between denotational and syntactical linearity through a full abstraction result (Corollary 2).
- A finite definability result (Theorem 6). It gives evidence that $\mathcal{S}\ell\mathrm{PCF}_\star$ is able to program a broader class of linear programs than other linear programming languages.
- The coincidence of different operational equivalences (Corollary 5). This makes simpler the reasoning on the equivalence between linear programs.
- An efficient reduction semantics exploiting linear properties in order to provide a concrete running evaluation of $\mathcal{S}\ell\mathrm{PCF}_\star$ that avoids exhaustive evaluation searches, by tracing and recording explicitly the linear use of subterms.

3. Background

3.1 Coherence Spaces

Coherence spaces are a simple framework for Berry's stable functions [7], developed by Girard [19]. More details are in [20].

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 $\mathcal{C}l_{fin}(X)$.

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

Definition 1. *Let X and Y be coherence spaces and $f : \mathcal{C}l(X) \longrightarrow \mathcal{C}l(Y)$ be a monotone function. Then, f is* linear *whenever $\forall x \in \mathcal{C}l(X), \forall b \in f(x)\ \exists! a \in x$ s.t. $b \in f(\{a\})$.*

Linear functions can be represented as cliques.

Definition 2. *Let X and Y be coherence spaces. $X \multimap Y$ is the coherence space having $|X \multimap Y| = |X| \times |Y|$ as web, while $(a,b) \frown_{X\multimap Y} (a',b')$ iff $a \frown_X a'$ implies $b \frown_Y b'$.*

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

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

Lemma 1. *If $f : \mathcal{C}l(X) \to \mathcal{C}l(Y)$ is a linear function then $\mathcal{T}r(f) \in \mathcal{C}l(X \multimap Y)$. 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.*

Definition 3 (Linear Model)**.** *The* Linear Model $\mathcal{L}$ *is the type structure generated by the coherence space* $\mathbf{N}$ *and the arrow* $\multimap$.

3.2 The language $\mathcal{S}\ell\mathrm{PCF}$

$\mathcal{S}\ell\mathrm{PCF}$ has been introduced in [17, 30] to be the syntactical counterpart of the above mentioned linear model.

Definition 4. *The set $\mathbb{T}$ of* linear types *is defined by the grammar:* $\sigma, \tau \quad ::= \quad \iota \mid \sigma \multimap \tau$ *, where ι is the only* ground *type.*

For the sake of clearness we introduce three kinds of variables, in order to remark their different explicit use.

Definition 5. *Let Var^σ be numerable sets of variables of type σ. Let $\mathcal{S}\mathrm{Var}^\sigma$ ($\sigma \neq \iota$) be numerable sets of variables disjoint from*

$$\frac{}{\vdash \mathbf{0} : \iota}\ (z) \quad \frac{}{\vdash \mathbf{s} : \iota \multimap \iota}\ (s) \quad \frac{}{\vdash \mathbf{p} : \iota \multimap \iota}\ (p) \quad \frac{}{\varkappa^{\sigma} \vdash \varkappa : \sigma}\ (v) \quad \frac{}{\vdash \mathtt{which?} : ((\iota \multimap \iota) \multimap \iota) \multimap \iota}\ (w)$$

$$\frac{\Gamma \vdash \mathtt{M} : \tau}{\Gamma, \mathtt{x}^{\iota} \vdash \mathtt{M} : \tau}\ (gw) \quad \frac{\Gamma, \mathtt{x_1}^{\iota}, \mathtt{x_2}^{\iota} \vdash \mathtt{M} : \tau}{\Gamma, \mathtt{x}^{\iota} \vdash \mathtt{M}[\mathtt{x}/\mathtt{x_1}, \mathtt{x_2}] : \tau}\ (gc) \quad \frac{\Gamma, \mathtt{x}^{\sigma} \vdash \mathtt{M} : \tau}{\Gamma \vdash \lambda \mathtt{x}^{\sigma}.\mathtt{M} : \sigma \multimap \tau}\ (\lambda) \quad \frac{\Gamma \cap \Delta = \emptyset \quad \Gamma \vdash \mathtt{M} : \iota \quad \Delta \vdash \mathtt{L} : \iota \quad \Delta \vdash \mathtt{R} : \iota}{\Gamma, \Delta \vdash \ell\mathtt{if}\ \mathtt{M}\ \mathtt{L}\ \mathtt{R} : \iota}\ (lif)$$

$$\frac{\Gamma \vdash \mathtt{M} : \tau}{\Gamma, F^{\sigma} \vdash \mathtt{M} : \tau}\ (sw) \quad \frac{\Gamma, F_1^{\sigma}, F_2^{\sigma} \vdash \mathtt{M} : \tau}{\Gamma, F^{\sigma} \vdash \mathtt{M}[F/F_1, F_2] : \tau}\ (sc) \quad \frac{\Gamma, F^{\sigma} \vdash \mathtt{M} : \sigma \quad \Gamma \restriction \ell = \emptyset}{\Gamma \vdash \mu F.\mathtt{M} : \sigma}\ (\mu) \quad \frac{\Gamma \cap \Delta = \emptyset \quad \Gamma \vdash \mathtt{M} : \sigma \multimap \tau \quad \Delta \vdash \mathtt{N} : \sigma}{\Gamma, \Delta \vdash \mathtt{MN} : \tau}\ (ap)$$

$$\frac{}{\mathbf{0} \Downarrow \mathbf{0}}\ (0) \quad \frac{\mathtt{M} \Downarrow \underline{\mathtt{n}}}{\mathbf{s}\,\mathtt{M} \Downarrow \mathbf{s}\,\underline{\mathtt{n}}}\ (s) \quad \frac{\mathtt{M} \Downarrow \mathbf{s}\,\underline{\mathtt{n}}}{\mathbf{p}\,\mathtt{M} \Downarrow \underline{\mathtt{n}}}\ (p) \quad \frac{\mathtt{M} \Downarrow \mathbf{0} \quad \mathtt{L} \Downarrow \underline{\mathtt{m}}}{\ell\mathtt{if}\ \mathtt{M}\ \mathtt{L}\ \mathtt{R} \Downarrow \underline{\mathtt{m}}}\ (ifl) \quad \frac{\mathtt{M} \Downarrow \mathbf{s}(\underline{\mathtt{n}}) \quad \mathtt{R} \Downarrow \underline{\mathtt{m}}}{\ell\mathtt{if}\ \mathtt{M}\ \mathtt{L}\ \mathtt{R} \Downarrow \underline{\mathtt{m}}}\ (ifr) \quad \frac{\mathtt{M}[\mathtt{N}/\mathtt{f}]\mathtt{P_1} \cdots \mathtt{P_i} \Downarrow \underline{\mathtt{v}}}{(\lambda \mathtt{f}^{\sigma \multimap \tau}.\mathtt{M})\mathtt{NP_1} \cdots \mathtt{P_i} \Downarrow \underline{\mathtt{v}}}\ (\lambda^{\multimap})$$

$$\frac{\mathtt{M}[\mu F.\mathtt{M}/F]\mathtt{P_1} \cdots \mathtt{P_i} \Downarrow \underline{\mathtt{v}}}{(\mu F.\mathtt{M})\mathtt{P_1} \cdots \mathtt{P_i} \Downarrow \underline{\mathtt{v}}}\ (\mu) \quad \frac{\mathtt{N} \Downarrow \underline{\mathtt{m}} \quad \mathtt{M}[\underline{\mathtt{m}}/\mathtt{x}]\mathtt{P_1} \cdots \mathtt{P_i} \Downarrow \underline{\mathtt{v}}}{(\lambda \mathtt{x}^{\iota}.\mathtt{M})\mathtt{NP_1} \cdots \mathtt{P_i} \Downarrow \underline{\mathtt{v}}}\ (\lambda^{\iota}) \quad \frac{\mathtt{M}(\lambda \mathtt{x}^{\iota}.\ell\mathtt{if}(\overbrace{\mathbf{p} \ldots \mathbf{p}}^{\underline{\mathtt{k}}}\ \mathtt{x})\ \underline{\mathtt{k}}\ (\mathbf{p0})) \Downarrow \underline{\mathtt{n}}}{\mathtt{which?}\,\mathtt{M}^{(\iota \multimap \iota) \multimap \iota} \Downarrow \lceil \underline{\mathtt{n}}, \underline{\mathtt{k}} \rceil}\ (w)$$

$$[\![\mathbf{0}^{\iota}]\!]\rho = \{0\} \quad [\![\mathbf{s}^{\iota \multimap \iota}]\!]\rho = \{(n, n+1) \mid n \in \mathbb{N}\} \quad [\![\mathbf{p}^{\iota \multimap \iota}]\!]\rho = \{(0,0)\} \cup \{(n, n-1) \mid n \in \mathbb{N} \wedge n > 0\} \quad [\![\mathtt{x}^{\sigma}]\!]\rho = \{\rho(\mathtt{x}^{\sigma})\}$$

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

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

$$[\![(\mu F^{\sigma}.\mathtt{M}^{\sigma})^{\sigma}]\!]\rho = \mathtt{fix}(\mathbb{\lambda} x \in \mathcal{C}l([\![\sigma]\!]).\bigcup_{x' \subseteq_{fin} x} [\![\mathtt{M}]\!]\rho[F := x']) \quad [\![\mathtt{which?}]\!]\rho = \{(((n,n),r), \lceil r, n \rceil) \mid n, r \in \mathbb{N}\}$$

Table 1. (a) Type system, (b) operational semantics and (c) linear interpretation for $\mathcal{S}\ell\mathrm{PCF}$

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* $\mathcal{S}\mathrm{Var} = \bigcup_{\sigma \neq \iota} \mathcal{S}\mathrm{Var}^{\sigma}$ *are named* stable variables.

Note that there are no stable variables of type ι. Latin letters $\mathtt{x}^{\sigma}, \mathtt{y}^{\sigma}, \mathtt{f}^{\sigma}, \ldots$ denote variables in Var^{σ}. We use types to explicitly distinguish between ground, e.g. $\mathtt{x}^{\iota}$, and linear variables, e.g. $\mathtt{x}^{\sigma \multimap \tau}$. Moreover, $F^{\sigma}, F_1^{\sigma}, F_2^{\sigma}, \ldots$ denote stable variables. Last, $\varkappa$ is a wild-card for all the variables.

We define terms (Definition 7) as the pre-terms (Definition 6) that can be typed using the type system in Table 1.a.

Definition 6. Pre-terms *are defined by the grammar:*

$$\mathtt{M} ::= \varkappa^{\tau} \mid \mathbf{0} \mid \mathbf{s} \mid \mathbf{p} \mid \ell\mathtt{if}\ \mathtt{M}\ \mathtt{M}\ \mathtt{M} \mid (\mathtt{MM}) \mid (\lambda \mathtt{x}^{\sigma}.\mathtt{M}) \mid \mu F.\mathtt{M} \mid \mathtt{which?}$$

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*.

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 Var, where each variable appears at most once. We denote $\Gamma \restriction \mathcal{S}$ (resp $\Gamma \restriction \iota$, $\Gamma \restriction \ell$) the restriction of the basis Γ containing only variables in $\mathcal{S}\mathrm{Var}$ (resp. in Var^{ι}, $\ell\mathrm{Var}$). We denote Γ, Δ and $\Gamma \cap \Delta$ the union and the intersection of two basis respectively. We can now define $\mathcal{S}\ell\mathrm{PCF}$ terms.

Definition 7. *The* terms *of* $\mathcal{S}\ell\mathrm{PCF}$ *are the pre-terms typable by using the type system in Table 1.a.*

Sometimes, we write $\mathtt{M}^{\sigma}$ when, for some Γ, we have $\Gamma \vdash \mathtt{M} : \sigma$. Free variables of any kind (FV), free linear variables (ℓFV), free stable variables ($\mathcal{S}$FV), closed and open terms are defined as expected. We denote $\mathcal{P} = \{\mathtt{M}^{\iota} \in \mathcal{S}\ell\mathrm{PCF} \mid \mathrm{FV}(\mathtt{M}^{\iota}) = \emptyset\}$ the set of *programs*. As usual, $\mathtt{M}[\mathtt{N}/\varkappa]$ denotes the capture-free substitution of all free occurrences of $\varkappa$ in $\mathtt{M}$ by $\mathtt{N}$.

Lemma 2 (Substitution)**.** *Let* $\mathtt{M}^{\tau}, \mathtt{N}^{\sigma} \in \mathcal{S}\ell\mathrm{PCF}$.

- *If* $\ell\mathrm{FV}(\mathtt{M}^{\tau}) \cap \ell\mathrm{FV}(\mathtt{N}^{\sigma}) = \emptyset$ *and* $\mathtt{x}^{\sigma} \in \ell\mathrm{Var}$ *then* $\mathtt{M}^{\tau}[\mathtt{N}^{\sigma}/\mathtt{x}^{\sigma}] \in \mathcal{S}\ell\mathrm{PCF}$.
- *If* $\ell\mathrm{FV}(\mathtt{N}^{\tau}) = \emptyset$ *then* $\mathtt{M}^{\tau}[\mathtt{N}^{\sigma}/F^{\sigma}] \in \mathcal{S}\ell\mathrm{PCF}$.
- *If* $\ell\mathrm{FV}(\mathtt{N}^{\tau}) = \emptyset$ *then* $\mathtt{M}[\mathtt{N}/\varkappa_1, \ldots, \mathtt{N}/\varkappa_n] \in \mathcal{S}\ell\mathrm{PCF}$.

Pairing (i.e. mapping pair of numerals on one numeral) and *projections function* will be used everywhere in the paper. We will denote with $\lceil \underline{\mathtt{n}}, \underline{\mathtt{m}} \rceil$ the numeral $\underline{\mathtt{k}}$ encoding the ordered pair of $\underline{\mathtt{n}}$ and $\underline{\mathtt{m}}$ and we write $\pi_1(\underline{\mathtt{k}})$ for the numeral $\underline{\mathtt{n}}$ and $\pi_2(\underline{\mathtt{k}})$ for the numeral $\underline{\mathtt{m}}$. $\mathcal{S}\ell$PCF-terms defining them can be found in [30]. We avoided to explicitly use the product-types in syntax and semantics just for sake of simplicity, albeit a such extension can be pursued.

Definition 8. *The* evaluation relation $\Downarrow \subseteq \mathcal{P} \times \mathcal{N}$ *is the smallest relation inductively satisfying the rules of Table 1.b. 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* $\mathtt{M} \Uparrow$.

Remark that $\mathbf{p}$ is a partial operator, namely $\mathbf{p}\,\mathbf{0}$ diverges. Moreover, note that as stressed in Section 2, the result of the evaluation of the operator $\mathtt{which?}$ applied to a term $\mathtt{M}$ consists of a pair $\lceil \underline{\mathtt{n}}, \underline{\mathtt{k}} \rceil$ where $\underline{\mathtt{n}}$ is the result of the evaluation of $\mathtt{M}(\lambda \mathtt{x}^{\iota}.\mathtt{x})$ while $\underline{\mathtt{k}}$ is the unique (thanks to linearity) numeral that $\mathtt{M}$ gives as argument to $\lambda \mathtt{x}^{\iota}.\mathtt{x}$.

The set of *σ-context* Ctx_{σ} is defined as:

$$\begin{aligned} \mathtt{C}[\sigma] \quad ::= \quad & [\sigma] \mid \varkappa^{\tau} \mid \mathbf{0} \mid \mathbf{s} \mid \mathbf{p} \mid \ell\mathtt{if}\ \mathtt{C}[\sigma]\ \mathtt{C}[\sigma]\ \mathtt{C}[\sigma] \\ & \mid \mathtt{which?} \mid (\mathtt{C}[\sigma]\mathtt{C}[\sigma]) \mid (\lambda \mathtt{x}^{\sigma}.\mathtt{C}[\sigma]) \mid \mu F.\mathtt{C}[\sigma] \end{aligned}$$

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

Definition 9 (Standard Operational Equivalence)**.** *Let* $\mathtt{M}^{\sigma}, \mathtt{N}^{\sigma} \in \mathcal{S}\ell\mathrm{PCF}$.

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

We are interested in a language for which the linear model $\mathcal{L}$ is fully abstract under a *standard interpretation* $[\![-]\!]$, i.e. ground types are interpreted on flat posets (see [33]). The standard interpretation is such that $[\![\iota]\!] = \mathbf{N}$ and $[\![\sigma \multimap \tau]\!] = [\![\sigma]\!] \multimap [\![\tau]\!]$.

An *environment* $\rho \in \mathrm{Env}$ is a partial function mapping a variable $\mathtt{x}^{\sigma}$ in a token $a \in |[\![\sigma]\!]|$ and a stable variable F^{σ} in a finite clique $x \in \mathcal{C}l_{fin}([\![\sigma]\!])$. 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 i-th 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 $\vec{F}$ is a sequence of stable variables of the same length then $\rho[\vec{F} := \vec{x}]$ is defined likewise.

Definition 10. *Let* $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathcal{S}\ell\mathrm{PCF}$ *and* $\rho \in \mathrm{Env}$. *The* linear interpretation $[\![\mathtt{M}^\sigma]\!] : \mathrm{Env} \to \mathcal{C}l([\![\sigma]\!])$ *is defined in Table 1.c using* $\mathcal{F}$ *as defined in* Equation 2 *and* `fix` *which is the least fix point operator.*

3.3 Stable Closed Full Abstraction

We recall the main properties of $\mathcal{S}\ell\mathrm{PCF}$ [30, 32]. First, the linear interpretation is adequate and correct.

Theorem 1 ([30])**.** *Let* $\mathtt{M}^\iota \in \mathcal{P}$ *and* $\mathtt{N}^\sigma, \mathtt{L}^\sigma \in \mathcal{S}\ell\mathrm{PCF}$.

Adequacy: $\mathtt{M}^\iota \Downarrow \underline{\mathtt{n}} \iff [\![\mathtt{M}^\iota]\!] = [\![\underline{\mathtt{n}}]\!]$.
Correctness: $[\![\mathtt{N}^\sigma]\!] = [\![\mathtt{L}^\sigma]\!] \Rightarrow \mathtt{N} \approx^\sigma \mathtt{L}$

In $\mathcal{S}\ell\mathrm{PCF}$ all the tokens of the linear model $\mathcal{L}$ can be defined. Moreover, as shown in [30], this can be done without using the `which`? operator.

Theorem 2 (Token Definability)**.**
If $u \in |[\![\sigma]\!]|$ *then there exists a closed* $\mathtt{M}^\sigma \in \mathcal{S}\ell\mathrm{PCF}$ *such that* $[\![\mathtt{M}]\!] = \{u\}$.

Token definability permits to define the separating terms used in the next lemma. This in contrast to what happens in [29, 33] where a finite definability is needed.

Lemma 3 (Separability)**.** *Let* $\sigma \in \mathbb{T}$. *For all distinct* $f, g \in \mathcal{C}l([\![\sigma]\!])$ *there exists a closed term* $\mathtt{P}^{\sigma\multimap\iota}$ *such that* $\mathcal{F}([\![\mathtt{P}]\!])(f) \neq \mathcal{F}([\![\mathtt{P}]\!])(g)$.

From this follows that full abstraction holds for all the terms that do not contain free *stable* variables. We stress that such a result was erroneously claimed in [30] to be of more generality. Indeed, we show in the next section that the unrestricted full abstraction fails.

Theorem 3 (Stable Closed Completeness)**.** *Let* $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathcal{S}\ell\mathrm{PCF}$ *and* $\mathcal{S}\mathrm{FV}(\mathtt{M}^\sigma) = \mathcal{S}\mathrm{FV}(\mathtt{N}^\sigma) = \emptyset$. *Then:*

$$\mathtt{M} \approx_\sigma \mathtt{N} \Rightarrow [\![\mathtt{M}]\!] = [\![\mathtt{N}]\!]$$

Proof. Let us prove the contrapositive. Let us assume $[\![\mathtt{M}]\!]\rho \neq [\![\mathtt{N}]\!]\rho$, for an environment ρ. By Separability Lemma 3, there exists a closed $\mathtt{P}^{\sigma\multimap\iota}$ such that $\mathcal{F}([\![\mathtt{P}]\!]\rho)([\![\mathtt{M}]\!]\rho) = n_1 \neq \mathcal{F}([\![\mathtt{P}]\!])([\![\mathtt{N}]\!]\rho)$. Moreover, by Token Definability Theorem 2 we can build a context $\mathtt{C}[^\iota]$ such that $[\![\mathtt{P}\ \mathtt{M}]\!]\rho = [\![\mathtt{C}[\mathtt{P}\ \mathtt{M}]]\!]\emptyset$ and $[\![\mathtt{P}\ \mathtt{N}]\!]\rho = [\![\mathtt{C}[\mathtt{P}\ \mathtt{N}]]\!]\emptyset$ where $\emptyset$ is the empty environment. So, by adequacy we have $(\mathtt{C}[\mathtt{P}\ \mathtt{M}]) \Downarrow \underline{\mathtt{n}_1}$ and $(\mathtt{C}[\mathtt{P}\ \mathtt{N}]) \not\Downarrow \underline{\mathtt{n}_1}$. So $\mathtt{M} \not\approx_\sigma \mathtt{N}$. □

Note that, in the above proof the Token Definability Theorem 2 allows us to build the context $\mathtt{C}[^\iota]$ only because we have assumed that the two terms have no stable variables. In the next section, we prove that in order to relax this constraint the language must to be extended.

Corollary 1 (Stable Closed Full Abstraction)**.** *Let* $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathcal{S}\ell\mathrm{PCF}$ *and* $\mathcal{S}\mathrm{FV}(\mathtt{M}^\sigma) = \mathcal{S}\mathrm{FV}(\mathtt{N}^\sigma) = \emptyset$. *Then:*

$$\mathtt{M} \approx_\sigma \mathtt{N} \iff [\![\mathtt{M}]\!] = [\![\mathtt{N}]\!]$$

4. Lack of Full abstraction

$\mathcal{S}\ell\mathrm{PCF}$ does not enjoy the unrestricted full abstraction. In this section, we show that the Corollary 1 cannot be extended to terms having free occurrences of stable variables, since they are interpreted in finite cliques. We prove that $\mathcal{S}\ell\mathrm{PCF}$ is not able to define all finite cliques of the model by presenting a linear function that is not definable in $\mathcal{S}\ell\mathrm{PCF}$, since it is not strongly stable [12]. We use this function to define two terms having free occurrences of stable variables which are operationally equivalent, albeit they are interpreted into two different linear functions.

4.1 Token enumeration

First, we introduce some abbreviations that are useful in order to simplify the rest of the paper. We use $(\mathtt{M}\ \mathtt{and}\ \mathtt{N})$ to abbreviate $(\ell\mathtt{if}\ \mathtt{M}\ (\ell\mathtt{if}\ \mathtt{N}\ \underline{0}\ \underline{1})\ \underline{1})$. The equivalence among numerals, denoted $\doteq$ used in infix notation, is encoded as $\mu F^{\iota\multimap\iota\multimap\iota}.\lambda\mathtt{x}^\iota.\lambda\mathtt{y}^\iota.\ell\mathtt{if}\ \mathtt{x}\ (\ell\mathtt{if}\ \mathtt{y}\ \underline{0}\ \underline{1})\ \big(\ell\mathtt{if}\ \mathtt{y}\ \underline{1}\ (F\,(\mathtt{px})(\mathtt{py}))\big)$. Moreover, we define a family of diverging terms by induction on types, $\Omega^\iota = \mathbf{p}\underline{0}$ and, if $\sigma_0 = \mu_1 \multimap \ldots \multimap \mu_\mathtt{m} \multimap \iota$ for some $\mathtt{m} \in \mathbb{N}$ then $\Omega^{\sigma_0\multimap\ldots\multimap\sigma_\mathtt{n}\multimap\iota}$ is

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

Clearly, $\Omega^{\sigma_1\multimap\ldots\sigma_\mathtt{n}\multimap\iota}\mathtt{M}_1^{\sigma_1} \ldots \mathtt{M}_\mathtt{n}^{\sigma_\mathtt{n}} \Uparrow$ for each $\mathtt{M}_1^{\sigma_1}, \ldots, \mathtt{M}_n^{\sigma_n}$.

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

- $(\!|n|\!) = n$ if $\sigma = \iota$;
- $(\!|(a_1, a_2)|\!) = \lceil (\!|a_1|\!), (\!|a_2|\!) \rceil$ if $\sigma = \tau_1 \multimap \tau_2$.

It provides an enumeration of the tokens of our model. Remark that the Theorem 2 implies that there is a family of terms $\mathtt{Sgl}_\mathtt{n}^\sigma : \sigma$ (short for *singleton*) being an enumeration of terms that can be interpreted on (single) tokens. Concretely, we define $\mathtt{Sgl}_\mathtt{n}^\sigma : \sigma$ by mutual induction with terms $\mathtt{Chk}_\mathtt{n}^{(\sigma)} : \sigma \multimap \iota$ that checks whether a token is included in the operational behavior of a term typed σ.

Definition 11. *The terms* $\mathtt{Sgl}_\mathtt{n}^\sigma : \sigma$ *and* $\mathtt{Chk}_\mathtt{n}^{(\sigma)} : \sigma \multimap \iota$ *are defined by mutual induction on* σ.
If $\sigma = \iota$, $\mathtt{Sgl}_\mathtt{n}^\iota = \underline{\mathtt{n}}$ *and* $\mathtt{Chk}_\mathtt{n}^{(\iota)} = \lambda\mathtt{y}^\iota.\ell\mathtt{if}\ (\underline{\mathtt{n}} \doteq \mathtt{y})\ \underline{0}\ \Omega^\iota$.
If $\sigma = \sigma_1 \multimap \sigma_2$, *let* $\sigma_2 = \tau_1 \multimap \ldots \multimap \tau_k \multimap \iota$ *for some* k, *without loss of generality, then* $\mathtt{Sgl}_\mathtt{n}^\sigma$ *is*

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

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

In $\mathtt{Chk}_\mathtt{n}^{(\sigma)}$ we use (σ) as a short for $\sigma \multimap \iota$. As an instance, if $\underline{\mathtt{n}} = \lceil \underline{\mathtt{n}_1}, \underline{\mathtt{n}_2} \rceil$ then the 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 $\lambda\mathtt{f}^{\iota\multimap\iota}\ell\mathtt{if}\ (\mathtt{f}\ \underline{\mathtt{n}_1} \doteq \underline{\mathtt{n}_2})\ \underline{0}\ \Omega^\iota$.

Lemma 4. *Let us fix a type* σ. *If* $a \in |[\![\sigma]\!]|$ *and* $n = (\!|a|\!)$, *then:*

1. $[\![\mathtt{Sgl}_\mathtt{n}^{(\sigma)}]\!]\rho = \{a\}$, *for all* ρ;
2. *if* $[\![\mathtt{Chk}_\mathtt{n}^{(\sigma)}\ \mathtt{N}]\!]\rho = [\![\underline{0}]\!]\rho$ *then* $a \in [\![\mathtt{N}]\!]\rho$, *for all* ρ.

4.2 Fix-point operational equivalence

In order to simplify the reasoning about programs, we introduce a non-standard notion of operational equivalence.

Definition 12. *Let assume* $\mathtt{M}^\sigma, \mathtt{N}^\sigma$ *be terms of* $\mathcal{S}\ell\mathrm{PCF}$, *such that* $\mathcal{S}\mathrm{FV}(\mathtt{M}), \mathcal{S}\mathrm{FV}(\mathtt{N}) \subseteq \{F_1^{\sigma_1}, \ldots, F_n^{\sigma_n}\}$.

- $\mathtt{M} \lesssim_\sigma \mathtt{N}$ *whenever, for all* $\mathtt{P}_1^{\sigma_1}, \ldots \mathtt{P}_\mathtt{n}^{\sigma_\mathtt{n}}$, *for all* $\mathtt{C}[\sigma]$ *s.t.* $\mathtt{C}[\mathtt{M}[\mathtt{P}_1/F_1, \ldots, \mathtt{P}_\mathtt{n}/F_\mathtt{n}]], \mathtt{C}[\mathtt{N}[\mathtt{P}_1/F_1, \ldots, \mathtt{P}_\mathtt{n}/F_\mathtt{n}]] \in \mathcal{P}$ *if* $\mathtt{C}[\mathtt{M}[\mathtt{P}_1/F_1, \ldots, \mathtt{P}_\mathtt{n}/F_\mathtt{n}]] \Downarrow \underline{\mathtt{n}} \Rightarrow \mathtt{C}[\mathtt{N}[\mathtt{P}_1/F_1, \ldots, \mathtt{P}_\mathtt{n}/F_\mathtt{n}]] \Downarrow \underline{\mathtt{n}}$.
- $\mathtt{M} \sim_\sigma \mathtt{N}$ *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. Indeed, proving that the two coincide corresponds to prove that $\sim_\sigma$ is also a *congruence*. For instance, let us assume that both $\mathtt{M}^\iota, \mathtt{N}^\iota$ contain just one free variable F^σ. If $\mathtt{M} \not\sim_\sigma \mathtt{N}$ then it is not easy to build contexts $\mathtt{C}[\sigma]$ and $\mu F.\mathtt{C}'[\sigma]$ (depending from the common substitution to F) such that $\mathtt{C}[\mu F.\mathtt{C}'[\mathtt{M}]] \Downarrow$ and $\mathtt{C}[\mu F.\mathtt{C}'[\mathtt{N}]] \not\Downarrow$, i.e. $\mathtt{M} \not\approx_\sigma \mathtt{N}$.

The standard interpretation is correct with respect to the fix-point operational equivalence too.

Proposition 1. *Let* $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathcal{SL}\mathrm{PCF}$. $[\![\mathtt{M}]\!] = [\![\mathtt{N}]\!] \Rightarrow \mathtt{M} \sim^\sigma \mathtt{N}$.

For clarity, we anticipate that the fix-point equivalence coincide with the standard contextual equivalence (see Section 7).

4.3 Full Abstraction Counterexample

We use the second-order *gustave-or* operator $\mathtt{G\mathring{o}r}$: $(\iota \multimap \iota) \multimap (\iota \multimap \iota) \multimap (\iota \multimap \iota) \multimap \iota$, introduced in [30], and equipped with the following evaluation.

$$\frac{\mathtt{P}_0\underline{0} \Downarrow \underline{0} \quad \mathtt{P}_1\underline{1} \Downarrow \underline{0} \quad \mathtt{P}_2\underline{0} \Downarrow \underline{1}}{\mathtt{G\mathring{o}r}^{\mathtt{L}}\,\mathtt{P}_0\mathtt{P}_1\mathtt{P}_2 \Downarrow \underline{0}}\ (\mathtt{G}_0) \qquad \frac{\mathtt{P}_0\underline{0} \Downarrow \underline{1} \quad \mathtt{P}_1\underline{0} \Downarrow \underline{0} \quad \mathtt{P}_2\underline{1} \Downarrow \underline{0}}{\mathtt{G\mathring{o}r}^{\mathtt{L}}\,\mathtt{P}_0\mathtt{P}_1\mathtt{P}_2 \Downarrow \underline{1}}\ (\mathtt{G}_1)$$

$$\frac{\mathtt{P}_0\underline{1} \Downarrow \underline{0} \quad \mathtt{P}_1\underline{0} \Downarrow \underline{1} \quad \mathtt{P}_2\underline{0} \Downarrow \underline{0}}{\mathtt{G\mathring{o}r}^{\mathtt{L}}\,\mathtt{P}_0\mathtt{P}_1\mathtt{P}_2 \Downarrow \underline{2}}\ (\mathtt{G}_2) \qquad \frac{\mathtt{P}_0\underline{1} \Downarrow \underline{1} \quad \mathtt{P}_1\underline{1} \Downarrow \underline{1} \quad \mathtt{P}_2\underline{1} \Downarrow \underline{1}}{\mathtt{G\mathring{o}r}^{\mathtt{L}}\,\mathtt{P}_0\mathtt{P}_1\mathtt{P}_2 \Downarrow \underline{3}}\ (\mathtt{G}_3)$$

The operator $\mathtt{G\mathring{o}r}$, non-deterministically provides inputs to its three branches and it looks for their outputs (ex-ante, it is not possible to choose inputs). As soon as the evaluation terminates by using a rule, the other rules cannot terminate anywise, i.e. ex-post the evaluation determines a unique rule which converges. The operator $\mathtt{G\mathring{o}r}$ is the operational counterpart of the following clique

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

The following result has been proved in [30].

Theorem 4. $\mathtt{G\mathring{o}r}$ *is not* $\mathcal{SL}\mathrm{PCF}$*-definable*

Based on the statement above, we can build the desired counter-example. Here we make use of fix-point operational equivalence. Let $\mathtt{M} = F\,\Omega^{\iota\multimap\iota}\Omega^{\iota\multimap\iota}\Omega^{\iota\multimap\iota}$ and

$$\mathtt{N} = \ell\mathtt{if}\left(\begin{array}{l} F\,(\mathtt{Sgl}^{\iota\multimap\iota}_{[\underline{0},\underline{0}]})(\mathtt{Sgl}^{\iota\multimap\iota}_{[\underline{1},\underline{0}]})(\mathtt{Sgl}^{\iota\multimap\iota}_{[\underline{0},\underline{1}]}) \doteq \underline{0}\ \mathtt{and} \\ F\,(\mathtt{Sgl}^{\iota\multimap\iota}_{[\underline{0},\underline{1}]})(\mathtt{Sgl}^{\iota\multimap\iota}_{[\underline{0},\underline{0}]})(\mathtt{Sgl}^{\iota\multimap\iota}_{[\underline{1},\underline{0}]}) \doteq \underline{1}\ \mathtt{and} \\ F\,(\mathtt{Sgl}^{\iota\multimap\iota}_{[\underline{1},\underline{0}]})(\mathtt{Sgl}^{\iota\multimap\iota}_{[\underline{0},\underline{1}]})(\mathtt{Sgl}^{\iota\multimap\iota}_{[\underline{0},\underline{0}]}) \doteq \underline{2}\ \mathtt{and} \\ F\,(\mathtt{Sgl}^{\iota\multimap\iota}_{[\underline{1},\underline{1}]})(\mathtt{Sgl}^{\iota\multimap\iota}_{[\underline{1},\underline{1}]})(\mathtt{Sgl}^{\iota\multimap\iota}_{[\underline{1},\underline{1}]}) \doteq \underline{3} \end{array}\right)\underline{7}\,\Omega^\iota$$

be $\mathcal{SL}\mathrm{PCF}$ terms such that $F^{(\iota\multimap\iota)\multimap(\iota\multimap\iota)\multimap(\iota\multimap\iota)\multimap\iota} \vdash \mathtt{M}, \mathtt{N} : \iota$. It is easy to see that $[\![\mathtt{M}]\!]\rho \neq [\![\mathtt{N}]\!]\rho$, by taking $\rho(F) = g$ as defined in Equation 3. Instead, $\mathtt{M} \sim_\iota \mathtt{N}$ since to separate the two terms a term behaving like $\mathtt{G\mathring{o}r}$ is necessary. However, such a term is not definable in $\mathcal{SL}\mathrm{PCF}$ as shown by Theorem 4.

5. The extended language

To recover the problem presented in the previous section, $\mathcal{SL}\mathrm{PCF}_\star$ extends the $\mathcal{SL}\mathrm{PCF}$ language by means of a new $\ell\mathtt{et}$-$\ell\mathtt{or}$ operator.

Definition 13. *The* $\mathcal{SL}\mathrm{PCF}_\star$ pre-terms *are defined by extending the grammar of pre-terms as follows:*

$$\mathtt{M} ::= \cdots \mid \underline{\ell\mathtt{et}}\,\mathtt{f}_1^{\sigma_1\multimap\tau_1} = \mathtt{M}, \ldots, \mathtt{f}_\mathtt{k}^{\sigma_\mathtt{k}\multimap\tau_\mathtt{k}} = \mathtt{M}\,\underline{\mathtt{in}\ell\mathtt{or}}\ \mathtt{M}\,\mathtt{M}\,\mathtt{M}$$

The terms *of* $\mathcal{SL}\mathrm{PCF}_\star$ *are the pre-terms typable by using the type system in Table 1.a extended by the rules in Table 2.a.*

The $\ell\mathtt{et}$-$\ell\mathtt{or}$ generalizes the behavior of the $\mathtt{G\mathring{o}r}$ operator presented in the previous section by forcing a linear evaluation. It is worth noticing that the contexts of the terms $\mathtt{M}_\mathtt{i}$ in the $(\ell\mathtt{et}$-$\ell\mathtt{or})$ rule are managed in an additive way, contracting common (ground, linear and stable) variables. However, note that a $\ell\mathtt{et}$-$\ell\mathtt{or}$ binds all the linear variables in its three branches. So, a form of syntactic linearity (*by slice* [19]) for linear variables is preserved.

Definition 14. *A* slice *of a term* $\mathtt{M}$ *is a new term where:*

- *each subterm* $\ell\mathtt{if}\ \mathtt{P}\ \mathtt{M}_0\ \mathtt{M}_1$ *of* $\mathtt{M}$ *is replaced by, either* $(\lambda\mathtt{z}^\iota.\mathtt{M}_0)\mathtt{P}$ *or* $(\lambda\mathtt{z}^\iota.\mathtt{M}_1)\mathtt{P}$ *where* $\mathtt{z}$ *is a fresh variable;*
- *each subterm* $\underline{\ell\mathtt{et}}\,\mathtt{f}_1 = \mathtt{N}_1, \ldots, \mathtt{f}_k = \mathtt{N}_k\,\underline{\mathtt{in}\ell\mathtt{or}}\ \mathtt{M}_1\,\mathtt{M}_2\,\mathtt{M}_3$ *is replaced by,* $(\lambda\mathtt{f}_1 \ldots \mathtt{f}_k.\mathtt{M}_i)\mathtt{N}_1 \ldots \mathtt{N}_k$ *where* $1 \leq i \leq 3$.

Morally, a slice of a term chooses one branch of each $\ell\mathtt{if}$ and one branch of $\ell\mathtt{et}$-$\ell\mathtt{or}$. Indeed, a slice contains neither $\ell\mathtt{if}$ nor $\ell\mathtt{et}$-$\ell\mathtt{or}$. It is easy to verify that each linear variable $\mathtt{f}^{\sigma\multimap\tau}$ occurs in a slice of a term $\mathtt{M}$ at most once. This says that $\mathcal{SL}\mathrm{PCF}_\star$ is syntactically linear by slices. However, as expected, a term containing n $\ell\mathtt{if}$ and m $\ell\mathtt{et}$-$\ell\mathtt{or}$ has 2^n3^m slices which can eventually coincide.

In order to deal with the $\ell\mathtt{et}$-$\ell\mathtt{or}$ operator without violating the semantic linearity we need a careful evaluation of terms, this is described using the terms $\mathtt{Sgl}^\sigma_\mathtt{n}$ and $\mathtt{Chk}^{(\sigma)}_\mathtt{n}$ introduced in the previous section.

Definition 15. *The* evaluation relation $\Downarrow \subseteq \mathcal{P} \times \mathcal{N}$ *for* $\mathcal{SL}\mathrm{PCF}_\star$ *programs is the smallest relation satisfying the rules in Table 1.b extended by the rules in Table 2.b.*

The evaluation rules for $\ell\mathtt{et}$-$\ell\mathtt{or}$ are patterns for infinite rules, likewise the rule for $\mathtt{which?}$ (see previous sections) and for $\exists$ in [33]. The evaluation of the $\ell\mathtt{et}$-$\ell\mathtt{or}$ operator is obtained by three distinct rules that explore pairwisely the $\ell\mathtt{et}$-$\ell\mathtt{or}$ branches. The evaluation of a $\ell\mathtt{et}$-$\ell\mathtt{or}$ 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 using a *single* tuple (i.e., a token) of the traces in the $\ell\mathtt{et}$-$\ell\mathtt{or}$-argument $\mathtt{N}_\mathtt{i}$, for $i \leq k$. For instance, the $(1\mathtt{lgor})$ can be applied in the case $\mathtt{M}_1[\mathtt{N}_1/\mathtt{f}_1, \ldots, \mathtt{N}_\mathtt{k}/\mathtt{f}_\mathtt{k}] \Downarrow \underline{0}$ and $\mathtt{M}_2[\mathtt{N}_1/\mathtt{f}_1, \ldots, \mathtt{N}_\mathtt{k}/\mathtt{f}_\mathtt{k}] \Downarrow \underline{\mathtt{s}\mathtt{m}}$ and the same *single* information of each $\mathtt{N}_j$ coded on numerals n_j is used in both evaluations. The *check* of this constraints is the motivation for introducing the terms $\mathtt{Sgl}^\sigma_\mathtt{n}$ and $\mathtt{Chk}^{(\sigma)}_\mathtt{n}$ above. Albeit ex-ante we don't know what is the right pair of $\ell\mathtt{et}$-$\ell\mathtt{or}$ 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 the $\ell\mathtt{et}$-$\ell\mathtt{or}$ operator follows these ideas.

Definition 16. *Let* $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathcal{SL}\mathrm{PCF}_\star$ *and* $\rho \in \mathrm{Env}$. *The* linear interpretation $[\![\mathtt{M}^\sigma]\!] : \mathrm{Env} \to \mathcal{C}l([\![\sigma]\!])$ *is defined by the equations in Table 1.c extended by the ones for the* $\ell\mathtt{et}$-$\ell\mathtt{or}$ *operator in Table 2.c.*

It is not difficult to see that the correctness w.r.t. the linear model (Theorem 1) still holds for $\mathcal{SL}\mathrm{PCF}_\star$.

5.1 $\mathcal{SL}\mathrm{PCF}_\star$ program examples

We show how to use the $\ell\mathtt{et}$-$\ell\mathtt{or}$ operator in order to program in $\mathcal{SL}\mathrm{PCF}_\star$ the operator $\mathtt{G\mathring{o}r}$ presented in Section 4. This should so suggest how to recover the counterexample to full abstraction.

In fact, we show something more. We describe how to program a family of terms $\mathtt{G\mathring{o}r}^{\mathtt{L}}$ that generalize the behavior of the operator $\mathtt{G\mathring{o}r}$. The terms $\mathtt{G\mathring{o}r}^{\mathtt{L}}$, are parametrized over an ordered list $\mathtt{L}$ of four numerals $\underline{\mathtt{k}_0}, \underline{\mathtt{k}_1}, \underline{\mathtt{k}_2}, \underline{\mathtt{k}_3}$ representing the output returned in the case the assumptions of one of the rule $(\mathtt{G}_0)$, $(\mathtt{G}_1)$, $(\mathtt{G}_2)$ or $(\mathtt{G}_3)$ respectively, are satisfied. So, in particular $\mathtt{G\mathring{o}r}^{\mathtt{L}}$=$\mathtt{G\mathring{o}r}$ when $\mathtt{L} = \{\underline{0}, \underline{1}, \underline{2}, \underline{3}\}$ is considered. In order to proceed in a modular way, we introduce also a notation in order to consider restrictions of $\mathtt{G\mathring{o}r}^{\mathtt{L}}$ which 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, $\mathtt{G\mathring{o}r}^{\mathtt{L}_0}$ where $\mathtt{L}_0 = \underline{0}, \underline{2}, \bullet, \bullet$ is defined as

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

All the $\mathtt{G\mathring{o}r}^{\mathtt{L}}$ defined by just two rules, i.e. for $\mathtt{L}$ containing two occurrences of $\bullet$, can be defined likewise. Thus, $\mathtt{G\mathring{o}r}^{\mathtt{L}_1}$ with parameter

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

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

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

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

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

Table 2. (a) Type system, (b) operational semantics and (c) linear interpretation for the ℓet-ℓor operator.

$\mathtt{L}_1 = \underline{1}, \bullet, \underline{0}, \underline{0}$ can be defined using the ℓet-ℓor operator as

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

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

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

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

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

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

We give here also a flavor on how to use the which? operator to define programming constructs useful to collect run-time information. These programming constructs (called $@\mathtt{wh}?_\tau^\sigma$) are a generalization of the operators introduced in [30, Section 3.1] and will be used in the next Section to prove the Finite Definability Theorem 6. Let us introduce the operators $@\mathtt{wh}?_\tau^\sigma : (\sigma \multimap \tau) \multimap \sigma \multimap \tau$, with the following operational semantics:

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

Observe that $(@\mathtt{wh}?\ \mathtt{M}\,\mathtt{N})\mathtt{P}_1 \ldots \mathtt{P}_k \Downarrow$ if and only if $\mathtt{MNP}_1 \ldots \mathtt{P}_k \Downarrow$. This control operator gives back the result $\underline{\mathtt{n}}$ of the evaluation of $\mathtt{MN}\ \mathtt{P}_1 \ldots \mathtt{P}_k$ 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 6.

$@\mathtt{wh}?$ can be programmed in $\mathcal{S}\ell\mathrm{PCF}_\star$ using which?.

Theorem 5. *$@\mathtt{wh}?_\tau^\sigma$ is $\mathcal{S}\ell\mathrm{PCF}_\star$ programmable.*

Proof. The proof is by structural induction on σ. The base case is simple observing that the term
$\lambda \mathtt{f}^{\iota\multimap\tau}\lambda \mathtt{x}^\iota \lambda \mathtt{g}_1^{\tau_1}, \ldots, \mathtt{g}_k^{\tau_k}.\mathtt{which}?(\lambda \mathtt{h}^{\iota\multimap\iota}.\mathtt{f}(\mathtt{hx})\mathtt{g}_1 \ldots \mathtt{g}_k)$
behaves as $@\mathtt{wh}?_\tau^\iota$, with $\tau = \tau_1 \multimap \ldots \multimap \tau_k \multimap \iota$. The inductive case is a bit more complicated. The idea is to generalize the behavior of the term above where the variable $\mathtt{h}$ acts like an observer which retrieves the used trace. □

6. Finite Definability and Full Abstraction

In this section we prove that the linear model $\mathcal{L}$ is *fully abstract* with respect to $\mathcal{S}\ell\mathrm{PCF}_\star$. 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 $\mathcal{S}\ell\mathrm{PCF}_\star$ terms. In particular, we prove the completeness with respect to the Fix-Point equivalence. In the next section (Proposition 2 and Theorem 8) 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 $\mathcal{S}\ell\mathrm{PCF}_\star$ terms. Our proof follows a standard scheme for proofs of this kind, e.g. [33],[29]. Non trivial uses of $@\mathtt{wh}?$ and ℓet-ℓor constructors are needed in the inductive high-order steps.

Definition 17. *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 $\lceil u \rfloor$ [1], hence $\lceil u \rfloor = \{\mathtt{M} \mid [\![\mathtt{M}]\!] = u\}$. By abuse of notation, in the following we denote $\lceil x \rfloor$ a term $\mathtt{M}$ such that $\mathtt{M} = \lceil x \rfloor$.*

To prove definability, we use the following auxiliary lemma.

Lemma 5. *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. *$(a_0, \ldots, a_n, a) \frown (b_0, \ldots, b_n, b)$ iff $\exists k \leq n$: $a_k \smile b_k$.*
2. *$(a_0, \ldots, a_n, a) \asymp (b_0, \ldots, b_n, b)$ iff $\forall k \leq n$: $a_k \subset b_k$.*

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 6 (Finite Definability)**.** *If $u \in \mathcal{C}l_{fin}([\![\sigma]\!])$ then there exists a closed $\mathtt{M} \in \mathcal{S}\ell\mathrm{PCF}_\star$ such that $\mathtt{M} = \lceil \mathtt{u} \rfloor$.*

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

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

[1] In general, we use $\lceil a^1, \ldots, a^k \rfloor$ as an abbreviation for $\lceil \{a^1, \ldots, a^k\} \rfloor$.

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

  - If $\|u\| = 0$ then $u = \emptyset$ is defined by $\Omega^{\iota\multimap\iota}$.
  - If $\|u\| \geq 1$, 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 $\mathrm{RK}(\sigma) \geq 3$ and $k = 1$, then $\sigma = \tau \multimap \iota$ with $\tau = \sigma_1 \multimap \cdots \sigma_r \multimap \iota$.

  - If $\|u\| = 0$ then $u = \emptyset$ 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 $c \in \mathbb{N}$ $(1 \leq i \leq r)$. 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 $b^i \in \mathbb{N}$ $(0 \leq i \leq m)$. Suppose $a^i = (a^i_1, \ldots, a^i_r, c^i)$ where $a^i_j \in [\![\sigma_j]\!]$ and $c^i \in \mathbb{N}$ $(1 \leq j \leq r)$. By Lemma 5.1, we have $a^h \smile a^k$ $(0 \leq h \neq k \leq m)$, so by Lemma 5.2 $a^h_j \frown a^k_j$ $(1 \leq j \leq r)$. Hence, take $v_j = \{a^i_j \mid 1 \leq i \leq m\}$. 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 $0 \leq i \leq m$, $1 \leq j \leq r$ and $1 \leq k \leq s$. So, we can define:

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

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

  - If $\|u\| = 0$, then $u = \emptyset$ 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 $a_i \in [\![\tau_i]\!]$ $(1 \leq i \leq k)$ and $b \in \mathbb{N}$. Thus,

$$\begin{aligned}
&\lceil u \rfloor = \lambda \mathtt{f}_1 \ldots \mathtt{f}_k.\\
&\qquad \ell\mathtt{if}\ ((\mathtt{Chk}^{(\sigma_1)}_{\underline{\mathtt{a}^1_1}}\mathtt{f}_1)\ \mathtt{and}\cdots\mathtt{and}\ (\mathtt{Chk}^{(\sigma_k)}_{\underline{\mathtt{a}^1_k}}\mathtt{f}_k))\ \lceil \mathtt{b} \rfloor\ \Omega^{\iota}
\end{aligned}$$

  - 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 $a^1_i \smile a^2_i$. If $\tau_i = \iota$ then a^1_i and a^2_i are two different numbers: thus the term defining u is the following

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

    If $\tau = \nu_1 \multimap \ldots \nu_l \multimap \iota$ 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 $a^1_i \smile a^2_i$, 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 u is the following

$$\begin{aligned}
&\lceil u \rfloor = \lambda \mathtt{f}_1 \ldots \mathtt{f}_k.\Big(\lambda \mathtt{x}^{\iota}.\ell\mathtt{if}\ \Big(\boldsymbol{\pi}^1_1\mathtt{x} \doteq \underline{\mathtt{c}^1}\ \mathtt{and}\ \boldsymbol{\pi}^{1-1}_1(\boldsymbol{\pi}_2\mathtt{x}) \doteq \underline{\mathtt{e}^1_1}\\
&\qquad \mathtt{and}\ldots\boldsymbol{\pi}_2\mathtt{x} \doteq \underline{e^1_l}\Big)\Big(\ell\mathtt{if}\big(\mathtt{Chk}^{(\tau_1)}_{\underline{\mathtt{a}^1_1}}\mathtt{f}_1\ \mathtt{and}\ \ldots\\
&\mathtt{Chk}^{(\tau_{i-1})}_{\underline{\mathtt{a}^1_{i-1}}}\mathtt{f}_{i-1}\ \mathtt{and}\ \mathtt{Chk}^{(\tau_{i+1})}_{\underline{\mathtt{a}^1_{i+1}}}\mathtt{f}_{i+1}\ \mathtt{and}\ \ldots\mathtt{Chk}^{(\tau_k)}_{\underline{\mathtt{a}^1_k}}\mathtt{f}_k\big)\ \lceil \mathtt{b}^1 \rfloor\ \Omega\Big)\\
&\Big(\ell\mathtt{if}\big(\boldsymbol{\pi}^l_1\mathtt{x} \doteq \underline{c^2}\ \mathtt{and}\ \boldsymbol{\pi}^{l-1}_1(\boldsymbol{\pi}_2\mathtt{x}) \doteq \underline{e^2_1}\ \mathtt{and}\ldots\boldsymbol{\pi}_2\mathtt{x} \doteq \underline{e^2_l}\ \mathtt{and}\\
&\qquad \mathtt{Chk}^{(\tau_1)}_{\underline{\mathtt{a}^2_1}}\mathtt{f}_1\ \mathtt{and}\ \ldots\mathtt{Chk}^{(\tau_{i-1})}_{\underline{\mathtt{a}^2_{i-1}}}\mathtt{f}_{i-1}\ \mathtt{and}\\
&\qquad \mathtt{Chk}^{(\tau_{i+1})}_{\underline{\mathtt{a}^2_{i+1}}}\mathtt{f}_{i+1}\ \mathtt{and}\ldots\mathtt{Chk}^{(\tau_k)}_{\underline{\mathtt{a}^2_k}}\mathtt{f}_k\big)\ \lceil \mathtt{b}^2 \rfloor\ \Omega\Big)\\
&\Big)\Big(@\mathtt{wh}?\,{}^{\nu_l}_{\iota}(\ldots(@\mathtt{wh}?\,{}^{\nu_1}_{\nu_2\multimap\ldots\nu_l\multimap\iota}(\mathtt{f}_{\mathtt{i}})(\mathtt{N}_1))\ldots)\mathtt{N}_1\Big)
\end{aligned}$$

  - If $\|u\| > 2$, then $u = \{d^1, \ldots, d^m\}$ where $d^j = (a^j_1, \ldots, a^j_k, b^j)$, $a^j_i \in [\![\tau_i]\!]$ and $b^j \in \mathbb{N}$ $(1 \leq j \leq m,\ 1 \leq i \leq k)$. We denote by $d^j[b]$ the token $(a^j_1, \ldots, a^j_k, b)$. By Lemma 5.1 there exists $1 \leq h \leq k$ such that $a^1_h \frown a^2_h$, so we can build the following finite cliques:

$$\begin{aligned}
w_1 &= \{d^1[0], d^2[b^2 + 1]\ \}\\
w_2 &= \{d^1[b^1 + 1]\} \cup \{d^r[0] \mid 2 < r \leq m\ \}\\
w_3 &= \{d^2[0]\} \cup \{d^r[b^r + 1] \mid 2 < r \leq m\ \}
\end{aligned}$$

    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:

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

    and this concludes the proof. □

The definability of finite cliques is the key ingredient to extend the Stable Closed Completeness Theorem 3 to all the terms of $\mathcal{SL}\mathrm{PCF}_\star$ as follows.

Theorem 7 (Completeness)**.** *Let* $\mathtt{M}^{\sigma}, \mathtt{N}^{\sigma} \in \mathcal{SL}\mathrm{PCF}_\star$.

$$\mathtt{M} \sim_{\sigma} \mathtt{N} \Rightarrow [\![\mathtt{M}]\!] = [\![\mathtt{N}]\!]$$

Proof. Let $\Gamma \vdash \mathtt{M}, \mathtt{N} : \sigma$ with $\Gamma\restriction\mathcal{S} = \{F^{\tau_1}_1, \ldots, F^{\tau_n}_n\}$ and $\Gamma\restriction\ell, \Gamma\restriction\iota = \{\mathtt{x}_1{}^{\sigma_1}, \ldots, \mathtt{x}_{\mathtt{m}}{}^{\sigma_m}\}$. Assume $[\![\mathtt{M}]\!] \neq [\![\mathtt{N}]\!]$, then there exists ρ such that $[\![\mathtt{M}]\!]\rho \neq [\![\mathtt{N}]\!]\rho$. By the Separability Lemma 3, 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 Finite Definability Theorem 6, for all F_i in $\Gamma\restriction\mathcal{S}$ there is a term $\mathtt{P_i} = \lceil \rho(F_i) \rfloor$ 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 $\mathtt{C} = \mathtt{P}(\lambda \mathtt{x}^{\sigma_1}_1 \ldots \mathtt{x}^{\sigma_m}_m.[\cdot^{\sigma}]\mathtt{N}_1 \cdots \mathtt{N}_m)$. Without loss of generality, let us assume $\mathcal{F}([\![\mathtt{P}]\!])([\![\mathtt{M}]\!]\rho) = \{k\}$. By adequacy $\mathtt{C}[\mathtt{M}[\mathtt{P}_1/F_1, \ldots, \mathtt{P}_n/F_n]] \Downarrow \underline{\mathtt{k}}$ but $\mathtt{C}[\mathtt{N}[\mathtt{P}_1/F_1, \ldots, \mathtt{P}_n/F_n]] \not\Downarrow \underline{\mathtt{k}}$. This concludes the proof. □

By soundness and completeness the full abstraction follows.

Corollary 2 (Full Abstraction)**.** *Let* $\mathtt{M}^{\sigma}, \mathtt{N}^{\sigma} \in \mathcal{SL}\mathrm{PCF}_\star$.

$$\mathtt{M} \sim_{\sigma} \mathtt{N} \iff [\![\mathtt{M}]\!] = [\![\mathtt{N}]\!]$$

7. Coincidence of operational equivalences

In this section we prove the coincidence between the standard operational equivalence (Definition 9) and the fix-point operational equivalence (Definition 12). Therefore, the full abstraction holds also for the standard operational equivalence and a *compositional* theory of program equivalence can be effectively defined.

The fix-point equivalence coincides with the denotational equivalence by Corollary 2. The proof follows of the next proposition directly by the correctness of the denotational semantics w.r.t the standard operational equivalence.

Proposition 2. *Let* $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathcal{SL}\mathrm{PCF}_\star$. $\mathtt{M} \sim_\sigma \mathtt{N} \Rightarrow \mathtt{M} \approx_\sigma \mathtt{N}$.

The opposite direction is more difficult and it requires a semantic reasoning. First, we prove an auxiliary result claiming 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 (Corollary 3)[2]. Then, we use this fact, together with Adequacy (Theorem 1) and Finite Definability (Theorem 6) to prove the result in the ground case (Lemma 8), which implies the general result (Theorem 8).

The fact that finite cliques in a coherence space $X \in \mathcal{L}$ different from $\mathbf{N}$ are never maximal w.r.t. set-theoretical inclusion follows from the next lemmas.

Lemma 6.

1. *Let x be a non-empty finite set of tokens in $|[\![\sigma]\!]|$ s.t. $\exists a \in x \forall b \in x. a \asymp b$. Then $\exists a' \notin x \forall b \in x. a' \smile b$.*
2. *Let x be a non-empty finite set of tokens in $|[\![\sigma]\!]|$ ($\sigma \neq \iota$) s.t. $\exists a \in x \forall b \in x. a \supset b$. Then $\exists a' \notin x. \forall b \in x. a \frown b$.*

Corollary 3. *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 F^σ and an infinite clique $x \in \mathcal{C}l([\![\sigma]\!])$, with a slight abuse of notation in the sequel we write $[\![\mathtt{M}]\!]\rho[F := x]$ to denote $\bigcup_{y \subseteq_{fin} x} [\![\mathtt{M}]\!]\rho[F := y]$. The next auxiliary lemma will be useful in what follows.

Lemma 7. *Let $\mathtt{M}^\sigma, \mathtt{N}^\tau \in \mathcal{SL}\mathrm{PCF}_\star$ and $\rho \in$ Env. If $\mathtt{M}^\sigma[\mathtt{N}/F^\tau] \in \mathcal{SL}\mathrm{PCF}_\star$, then $[\![\mathtt{M}^\sigma[\mathtt{N}/F^\tau]]\!]\rho = \bigcup_{x \subseteq_{fin} [\![\mathtt{N}]\!]\rho} [\![\mathtt{M}]\!]\rho[F^\tau := x]$*

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 8. *If $\Gamma \vdash \mathtt{M}, \mathtt{N} : \iota$ with $\Gamma_{\restriction \ell} = \emptyset$. $\mathtt{M} \approx_\iota \mathtt{N} \Rightarrow \mathtt{M} \sim_\iota \mathtt{N}$.*

Proof. We prove the contrapositive. Let $F_1^{\sigma_1}, \ldots, F_n^{\sigma_n}$ be such that $\mathcal{SFV}(\mathtt{M}), \mathcal{SFV}(\mathtt{N}) \subseteq \{F_1, \ldots, F_n\}$. Let $\mathtt{C}[.^\iota]$ be a context and $\vec{\mathtt{P}}$ be closed terms such that $\mathtt{C}[\mathtt{M}[\vec{\mathtt{P}}/\vec{F}]] \Downarrow \underline{\mathtt{n}}$ and $\mathtt{C}[\mathtt{N}[\vec{\mathtt{P}}/\vec{F}]] \not\Downarrow \underline{\mathtt{n}}$. We prove by induction on n 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. Since $n = 0$ 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 $\mathtt{C}[.^\iota]$ and there are closed terms $\mathtt{P}_1, \ldots, \mathtt{P}_{n+1}$, such that $\mathtt{C}[\mathtt{M}[\mathtt{P}_1/F_1^{\sigma_1}, \ldots, \mathtt{P}_{n+1}/F_{n+1}^{\sigma_{n+1}}]] \Downarrow \underline{\mathtt{k}}$ but $\mathtt{C}[\mathtt{N}[\mathtt{P}_1/F_1^{\sigma_1}, \ldots, \mathtt{P}_{n+1}/F_{n+1}^{\sigma_{n+1}}]] \not\Downarrow \underline{\mathtt{k}}$. Thus, by Correctness, there exists a ρ such that

$$[\![\mathtt{M}]\!]\rho[F_1^{\sigma_1} := [\![\mathtt{P}_1]\!]\rho, \ldots, F_{n+1}^{\sigma_{n+1}} := [\![\mathtt{P}_{n+1}]\!]\rho] \neq [\![\mathtt{N}]\!]\rho[F_1^{\sigma_1} := [\![\mathtt{P}_1]\!]\rho, \ldots, F_{n+1}^{\sigma_{n+1}} := [\![\mathtt{P}_{n+1}]\!]\rho]$$

In the following, we define $\rho_1 = \rho[F_1^{\sigma_1} := [\![\mathtt{P}_1]\!]\rho, \ldots, F_n^{\sigma_n} := [\![\mathtt{P}_n]\!]\rho]$. Let us observe that both $[\![\mathtt{M}]\!]\rho_1[F_{n+1}^{\sigma_{n+1}} := [\![\mathtt{P}_{n+1}]\!]\rho]$ and $[\![\mathtt{N}]\!]\rho_1[F_{n+1}^{\sigma_{n+1}} := [\![\mathtt{P}_{n+1}]\!]\rho]$ 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 $[\![\mathtt{M}]\!]\rho_1[F_{n+1}^{\sigma_{n+1}} := [\![\mathtt{P}_{n+1}]\!]\rho] = \{k\}$. So by Lemma 7 there exists $x \subseteq_{fin} [\![\mathtt{P}_{n+1}]\!]\rho$ such that

$$[\![\mathtt{M}]\!]\rho_1[F_{n+1} := [\![\mathtt{P}_{n+1}]\!]\rho] = [\![\mathtt{M}]\!]\rho_1[F_{n+1} := x] \neq [\![\mathtt{N}]\!]\rho_1[F_{n+1} := x] \subseteq [\![\mathtt{N}]\!]\rho_1[F_{n+1} := [\![\mathtt{P}_{n+1}]\!]\rho]$$

Since stable variables are never of ground type, we can let $\sigma_{n+1} = \tau_1 \multimap \ldots \multimap \tau_m \multimap \iota$ and $m \geq 1$. Let $x = \{(a_1^1, \ldots, a_m^1, h^1), \ldots, (a_1^p, \ldots, a_m^p, h^p)\}$ with $p \geq 0$ and $a_1^i \in |[\![\tau_1]\!]|, \ldots, a_m^i \in |[\![\tau_m]\!]|, h^i \in \mathbb{N}$ for all $i \in [1, p]$. Furthermore, by Corollary 3 there is a token $(a_1^*, \ldots, a_m^*, h^*) \notin x$ such that $x^* = x \cup \{(a_1^*, \ldots, a_m^*, h^*)\}$ is still a clique. Observe that it is not restrictive to 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

$$\begin{aligned}\mathtt{D} = \mu F_{n+1}.\lambda \mathtt{f}_1^{\tau_1} \ldots \mathtt{f}_m^{\tau_m}.\Big(\lambda \mathtt{x}^\iota.\ & \ell\mathtt{if}\ (\mathtt{x} \doteq \underline{\mathtt{h}^*}) \quad [\cdot^\iota] \\ & \ell\mathtt{if}\ (\mathtt{x} \doteq \underline{\mathtt{h}_1}) \quad \underline{\mathtt{h}_1} \\ & \vdots \\ & \ell\mathtt{if}\ (\mathtt{x} \doteq \underline{\mathtt{h}_p}) \quad \underline{\mathtt{h}_p}\ \Omega^\iota\Big)\Big(\mathtt{P}\ \mathtt{f}_1\ \ldots \mathtt{f}_m\Big)\end{aligned}$$

Observe that $\mathtt{D}[\mathtt{M}], \mathtt{D}[\mathtt{N}] \in \mathcal{SL}\mathrm{PCF}_\star$ because we hypothesized that $\mathtt{M}, \mathtt{N}$ have no free occurrences of linear variables. If $q_1 = (\!|a_1^*|\!), \ldots, q_m = (\!|a_m^*|\!)$, it is not difficult to see that $[\![\mathtt{D}[\mathtt{M}]\ \mathtt{Sgl}_{\mathtt{q}_1}^{(\tau_1)}\ \ldots\ \mathtt{Sgl}_{\mathtt{q}_m}^{(\tau_m)}]\!]\rho_1 = [\![\mathtt{M}]\!]\rho_1[F_{n+1} := [\![\mathtt{P}_{n+1}]\!]\rho] = \{\mathtt{k}\}$ and $[\![\mathtt{D}[\mathtt{N}]\ \mathtt{Sgl}_{\mathtt{q}_1}^{(\tau_1)}\ \ldots\ \mathtt{Sgl}_{\mathtt{q}_m}^{(\tau_m)}]\!]\rho_1 \subseteq [\![\mathtt{N}]\!]\rho_1[F_{n+1} := [\![\mathtt{P}_{n+1}]\!]\rho]$ which is either empty or it is a singleton $\{k'\}$ different from k.
By Lemma 7, adequacy and definition of fix-point equivalence we have $\mathtt{D}[\mathtt{M}]\ \mathtt{Sgl}_{\mathtt{q}_1}^{(\tau_1)}\ \ldots\ \mathtt{Sgl}_{\mathtt{q}_m}^{(\tau_m)} \not\sim_\iota \mathtt{D}[\mathtt{N}]\ \mathtt{Sgl}_{\mathtt{q}_1}^{(\tau_1)}\ \ldots\ \mathtt{Sgl}_{\mathtt{q}_m}^{(\tau_m)}$. 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 8. *Let $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathcal{SL}\mathrm{PCF}_\star$. $\mathtt{M} \approx_\sigma \mathtt{N} \Rightarrow \mathtt{M} \sim_\sigma \mathtt{N}$.*

Proof. We prove the contrapositive statement. Let $F_1^{\sigma_1}, \ldots, F_n^{\sigma_n}$ be such that $\mathcal{SFV}(\mathtt{M}), \mathcal{SFV}(\mathtt{N}) \subseteq \{F_1, \ldots, F_n\}$. Let $\mathtt{C}$ be a context and $\vec{\mathtt{P}}$ be closed terms such that $\mathtt{C}[\mathtt{M}[\vec{\mathtt{P}}/\vec{F}]] \Downarrow \underline{\mathtt{n}}$ and $\mathtt{C}[\mathtt{N}[\vec{\mathtt{P}}/\vec{F}]] \not\Downarrow \underline{\mathtt{n}}$ for some numeral $\underline{\mathtt{n}}$. In particular, observe that $\mathtt{C}[\mathtt{M}] \not\sim_\iota \mathtt{C}[\mathtt{N}]$, by definition of Fix-Point Operational Equivalence. Moreover, by construction $\mathtt{C}[\mathtt{M}]$ and $\mathtt{C}[\mathtt{N}]$ have no free occurrence of linear variables. Thus, by Lemma 8 there is a context $\mathtt{D}$ 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 4. *The equivalence $\sim_\sigma$ is a congruence.*

7.1 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.

Definition 18 (Applicative Operational Equivalence)**.** *Let $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathcal{SL}\mathrm{PCF}_\star$ such that $\mathcal{SFV}(\mathtt{M}), \mathcal{SFV}(\mathtt{N}) \subseteq \{F_1^{\sigma_1}, \ldots, F_n^{\sigma_n}\}$.*

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

Theorem 9. *Let $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathcal{SL}\mathrm{PCF}_\star$. $\mathtt{M} \sim_\sigma^{\mathtt{A}} \mathtt{N} \Rightarrow [\![\mathtt{M}]\!] = [\![\mathtt{N}]\!]$.*

Proof. Just by observing that the context used in the proof of Theorem 7 is an applicative context. □

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

[2] 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 $\{(\emptyset, 0)\}$ is maximal.

Corollary 5 (Equivalences Coincidence)**.** *Let* $\mathtt{M}^\sigma, \mathtt{N}^\sigma \in \mathcal{S}\ell\mathrm{PCF}_\star$.

$$\mathtt{M} \sim_\sigma \mathtt{N} \iff \mathtt{M} \approx_\sigma \mathtt{N} \iff \mathtt{M} \sim_\sigma^{\mathtt{A}} \mathtt{N}$$

8. A Tracing Evaluation Semantics

The operational semantics of $\mathcal{S}\ell\mathrm{PCF}_\star$ presented in the previous sections is effective, but quite inefficient. Indeed, the rules for `which`? and ℓ`et-`ℓ`or` non-deterministically face a potentially infinite number of evaluation branches. Consequently, its bovine implementation should try an exhaustive search of the right branch among the infinite ones. In this section, we introduce a *tracing evaluation semantics* which is able to drastically prune such infinite-branching search tree.

Roughly, denotational linearity provides the certainty that each term is applied to a unique sequence of arguments. That is, only one token of the corresponding *trace* is used. So, an efficient evaluation can be obtained by storing an argument `N` in an environment e, and when the evaluation of `N` is done, by replacing the term in the environment by its trace. This way of evaluating programs is a kind of higher-order call-by-need evaluation where the trace of the term is stored instead of the value. In order to simplify the evaluation of the new operators, we record such sequence-information along the evaluation tree. The idea is "to recursively trace" the arguments supplied to functions. The so-obtained pruned search-tree is finite-branching and it induces a clever and more efficient evaluation.

A key role in order to trace all the information is played by the environment.

Definition 19. *An* **environment** e *is a function from a finite set of variables (ground or linear) to either a term (named* subject*) or a numeral (named* trace*).*
Let $\mathtt{x}^\sigma$ *be a variable in the domain of* e*. If* $\sigma = \iota$ *then* $e(\mathtt{x})$ *is always the trace. If* σ *is an arrow then either* $e(\mathtt{x})$ *is a term typed* σ *(the subject) or it is the trace typed* ι*.*
We note $e[\mathtt{z} := \mathtt{M}]$ *the new environment* e' *such that* $e'(\mathtt{z}) = \mathtt{M}$ *and* $e'(\mathtt{y}) = e(\mathtt{y})$*, for each* $\mathtt{y} \neq \mathtt{z}$*. Last, we note* $e\restriction_{\{\mathtt{x}\}}$ *the restricted environment obtained by deleting the variable* `x`*.*

Note that, the environments associate terms to type union, i.e. a variable could refer either to a term or to a numeral (encoding the trace). Moreover, in the case of a ground variable we just need to store a numeral because of the call-by-value policy that makes subjects and traces coincide.

The use of an environment to manage substitutions and the presence of a recursion operator raise the issue of clashes between variable-names. For sake of simplicity, we do not introduce closures. We overcame clashes by assuming a whole bunch of fresh variables and by doing the appropriate renaming at run-time in the reduction rules. Another reason for following that approach is that fresh variables are anyway necessary to trace the evaluation of `which`? arguments.

As expected, the rules of our machine will be driven by *states* (i.e. term in environment); they are denoted by $\langle \mathtt{M} | e \rangle$. Given a state, the environment contains as usual the subterms to be substituted to all the free variables and, sometimes, also some additional variables recording traces. Indeed, during the evaluation, sometimes an operational rule "fetches" an head redex and extends the environment with a fresh variable associated to a new subject. When arguments are supplied, the operational rules carefully substitute the subject by the convenient numeral encoding the trace. In this way, if a variable `x` contains a trace then, it has recorded the use done during the evaluation of the subject initially associated to the variable `x` itself. For sake of clarity, we explicitly remove not used variables from environments.

We note $\sharp$ the empty environment, so if `M` is a ground closed term then we can obtain its evaluation, by supplying the state $\langle \mathtt{M} | \sharp \rangle$ to the tracing machine.

Definition 20. *The* tracing evaluation $\Downarrow_{\mathtt{T}}$ *is the effective relation from states (ground terms in environments) to states (numerals in environments) defined by the rules of Table 3. If* $\langle \mathtt{M} | \sharp \rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}} | e_1 \rangle$ *then we say that* `M` converges*, and we write simply* $\mathtt{M} \Downarrow_{\mathtt{T}}$*, otherwise we say that it* diverges*, and we write* $\mathtt{M} \Uparrow_{\mathtt{T}}$.

We emphasize that all the states in rules of table 3 are driven by the head of a ground term. In order to facilitate the comprehension of the tracing machine we have devised its rules in two parts. The rules in the higher Part (a) take into account all possible shapes of the head of terms, except the case of a head variable that is tackled by the rules in the lower Part (b).

The rules in the higher part are quite easy to understand, so we comment only the rules involving non-standard operators. The rule $(\mathtt{w})$ extends the incoming environment e_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 `which`?`(M)` converges, by rules in the Table 1.c, then `MI` also converges). The result of this evaluation gives back a state $\langle \underline{\mathtt{n}} | e_1 \rangle$ where e_1 reports the observed trace of `M`. From this, the result is built. Likewise, the three ℓ`et-`ℓ`or` rules start by evaluating a branch. If such a branch converges, the reported traces are used in order to start the evaluation of the other branch. In this case we supply as subjects of the involved variables terms having (exactly) the behavior of the reported trace (recall the Lemma 4). Therefore, a rule converges only in case the two branches do the same observations on the list of arguments supplied to a linear variable.

The lower part of Table 3 is a bit more complex. All the rules fetch head-variables, so the discriminating factor that drives their behavior is found in the shape of the subject associated in the environment to the head variable itself. Indeed, it is easy to see that there is one rule for all possible shape of the "head-subject". We remark that ℓ`et-`ℓ`or` is not taken into account, since a term having a ℓ`et-`ℓ`or` as head is certainly typed ground and ground terms are evaluated before to be stored in the environment (because of the call-by-value policy).

The rule (Hgvar) is easy. (Hvar) considers the case where the subject associated to the head variable is another variable-name, it forwards the evaluation by using the subject. (Happ) is a key rule. It applies in the case the subject 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**) and (Hμ) are quite simple. Likewise to the rule $(\mathtt{w})$ presented above, the rule (H**w**) is interesting: it uses the information collected by the fresh variable `h` to produce the right outcome and the trace of the `which`? itself (since it is a subject of the incoming environment). Finally, the rules (Hλ^ι_ι), (H$\lambda^\iota_{_\circ}$), (H$\lambda^{_\circ}_\iota$) and (H$\lambda^{_\circ}_{_\circ}$) consider the cases arising from an 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 (Hλ^ι_ι) and (H$\lambda^\iota_{_\circ}$) need first to evaluate the ground argument in order to comply the call-by-value policy. All the rules compose the sub-traces information in the rule-premises to obtain the traces that they must provide in the rule-conclusion.

We now prove that the trace evaluation machine is correct and complete with respect to the evaluation semantics presented in Table 1 and Table 2. It is easy to check by induction that the rules of Table 3 preserve the following property: if $\mathrm{FV}(\mathtt{M}) = \{\varkappa_1, \ldots, \varkappa_n\}$ for some $n \geq 0$ then such variables are in the domain of e.

$$\frac{}{\langle \underline{0}|e\rangle \Downarrow_{\mathtt{T}} \langle \underline{0}|e\rangle}\ (0) \qquad \frac{\langle \mathtt{M}|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle}{\langle \mathtt{s}\,\mathtt{M}|e_0\rangle \Downarrow_{\mathtt{T}} \langle \mathtt{s}\,\underline{\mathtt{n}}|e_1\rangle}\ (\mathrm{s}) \qquad \frac{\langle \mathtt{M}|e_0\rangle \Downarrow_{\mathtt{T}} \langle \mathtt{s}\,\underline{\mathtt{n}}|e_1\rangle}{\langle \mathtt{p}\,\mathtt{M}|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle}\ (\mathrm{p}) \qquad \frac{\mathtt{f}' \text{ fresh}, \quad \langle \mathtt{M}[\mathtt{f}'/\mathtt{f}]\mathtt{P}_1 \dots \mathtt{P}_k|e_0[\mathtt{f}' := \mathtt{N}]\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle}{\langle (\lambda \mathtt{f}^{\sigma \multimap \tau}.\mathtt{M})\mathtt{N}\mathtt{P}_1 \dots \mathtt{P}_k|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\restriction_{\{\mathtt{f}'\}}\rangle}\ (\lambda^{\multimap})$$

$$\frac{\langle \mathtt{M}|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{0}|e_1\rangle \quad \langle \mathtt{L}|e_1\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{m}}|e_2\rangle}{\langle \ell\mathtt{if}\ \mathtt{M}\ \mathtt{L}\ \mathtt{R}|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{m}}|e_2\rangle}\ (\mathrm{ifl}) \qquad \frac{\langle \mathtt{M}|e_0\rangle \Downarrow_{\mathtt{T}} \langle \mathtt{s}\underline{\mathtt{n}}|e_1\rangle \quad \langle \mathtt{R}|e_1\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{m}}|e_2\rangle}{\langle \ell\mathtt{if}\ \mathtt{M}\ \mathtt{L}\ \mathtt{R}|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{m}}|e_2\rangle}\ (\mathrm{ifr}) \qquad \frac{\langle \mathtt{M}[\mu F.\mathtt{M}/F]\mathtt{P}_1 \dots \mathtt{P}_k|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle}{\langle (\mu F.\mathtt{M})\mathtt{P}_1 \dots \mathtt{P}_k|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle}\ (\mu)$$

$$\frac{\mathtt{x}' \text{ fresh}, \quad \langle \mathtt{N}|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}'|e_1\rangle \quad \langle \mathtt{M}[\mathtt{x}'/\mathtt{x}]\mathtt{P}_1 \dots \mathtt{P}_k|e_1[\mathtt{x}' := \underline{\mathtt{n}}']\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_2\rangle}{\langle (\lambda \mathtt{x}^{\iota}.\mathtt{M})\mathtt{N}\mathtt{P}_1 \dots \mathtt{P}_k|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_2\restriction_{\{\mathtt{x}'\}}\rangle}\ (\lambda^{\iota}) \qquad \frac{\mathtt{h}^{(\iota \multimap \iota) \multimap \iota} \text{ fresh}, \quad \langle \mathtt{hI}|e_0[\mathtt{h} := \mathtt{M}]\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle \quad \underline{\mathtt{k}} = \pi_1\pi_1 e_1(\mathtt{h})}{\langle \mathtt{which?}\,\mathtt{M}^{(\iota \multimap \iota) \multimap \iota}|e_0\rangle \Downarrow_{\mathtt{T}} \langle \lceil \underline{\mathtt{n}}, \underline{\mathtt{k}} \rceil | e_1\restriction_{\{\mathtt{h}\}}\rangle}\ (\mathbf{w})$$

$$\frac{\begin{array}{c}\mathtt{f}_i' \text{ fresh } (1 \le i \le k), \quad \langle \mathtt{M}_2[\mathtt{f}_1'/\mathtt{f}_1, \dots, \mathtt{f}_k'/\mathtt{f}_k]|e_1[\mathtt{f}_1' := \mathtt{Sgl}^{\sigma_1}_{e_1(\mathtt{f}_1')}, \dots, \mathtt{f}_k' := \mathtt{Sgl}^{\sigma_k}_{e_1(\mathtt{f}_k')}]\rangle \Downarrow_{\mathtt{T}} \langle \mathtt{s}\underline{\mathtt{m}}|e_2\rangle \\ \langle \mathtt{M}_1[\mathtt{f}_1'/\mathtt{f}_1, \dots, \mathtt{f}_k'/\mathtt{f}_k]|e_0[\mathtt{f}_1' := \mathtt{N}_1, \dots, \mathtt{f}_k' := \mathtt{N}_k]\rangle \Downarrow_{\mathtt{T}} \langle \underline{0}|e_1\rangle\end{array}}{\langle \underline{\ell\mathtt{et}}\ \mathtt{f}_1^{\sigma_1} = \mathtt{N}_1, \dots, \mathtt{f}_k^{\sigma_k} = \mathtt{N}_k\ \underline{\mathtt{in}\ell\mathtt{or}}\ \mathtt{M}_1\ \mathtt{M}_2\ \mathtt{M}_3|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{m}}|e_2\restriction_{\{\mathtt{f}_1', \dots, \mathtt{f}_k'\}}\rangle}\ (\mathtt{1lgor})$$

$$\frac{\begin{array}{c}\mathtt{f}_i' \text{ fresh } (1 \le i \le k), \quad \langle \mathtt{M}_3[\mathtt{f}_1'/\mathtt{f}_1, \dots, \mathtt{f}_k'/\mathtt{f}_k]|e_1[\mathtt{f}_1' := \mathtt{Sgl}^{\sigma_1}_{e_1(\mathtt{f}_1')}, \dots, \mathtt{f}_k' := \mathtt{Sgl}^{\sigma_k}_{e_1(\mathtt{f}_k')}]\rangle \Downarrow_{\mathtt{T}} \langle \mathtt{s}\underline{\mathtt{m}}|e_2\rangle \\ \langle \mathtt{M}_2[\mathtt{f}_1'/\mathtt{f}_1, \dots, \mathtt{f}_k'/\mathtt{f}_k]|e_0[\mathtt{f}_1' := \mathtt{N}_1, \dots, \mathtt{f}_k' := \mathtt{N}_k]\rangle \Downarrow_{\mathtt{T}} \langle \underline{0}|e_1\rangle\end{array}}{\langle \underline{\ell\mathtt{et}}\ \mathtt{f}_1^{\sigma_1} = \mathtt{N}_1, \dots, \mathtt{f}_k^{\sigma_k} = \mathtt{N}_k\ \underline{\mathtt{in}\ell\mathtt{or}}\ \mathtt{M}_1\ \mathtt{M}_2\ \mathtt{M}_3|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{m}}|e_2\restriction_{\{\mathtt{f}_1', \dots, \mathtt{f}_k'\}}\rangle}\ (\mathtt{2lgor})$$

$$\frac{\begin{array}{c}\mathtt{f}_i' \text{ fresh } (1 \le i \le k), \quad \langle \mathtt{M}_1[\mathtt{f}_1'/\mathtt{f}_1, \dots, \mathtt{f}_k'/\mathtt{f}_k]|e_1[\mathtt{f}_1' := \mathtt{Sgl}^{\sigma_1}_{e_1(\mathtt{f}_1')}, \dots, \mathtt{f}_k' := \mathtt{Sgl}^{\sigma_k}_{e_1(\mathtt{f}_k')}]\rangle \Downarrow_{\mathtt{T}} \langle \mathtt{s}\underline{\mathtt{m}}|e_2\rangle \\ \langle \mathtt{M}_3[\mathtt{f}_1'/\mathtt{f}_1, \dots, \mathtt{f}_k'/\mathtt{f}_k]|e_0[\mathtt{f}_1' := \mathtt{N}_1, \dots, \mathtt{f}_k' := \mathtt{N}_k]\rangle \Downarrow_{\mathtt{T}} \langle \underline{0}|e_1\rangle\end{array}}{\langle \underline{\ell\mathtt{et}}\ \mathtt{f}_1^{\sigma_1} = \mathtt{N}_1, \dots, \mathtt{f}_k^{\sigma_k} = \mathtt{N}_k\ \underline{\mathtt{in}\ell\mathtt{or}}\ \mathtt{M}_1\ \mathtt{M}_2\ \mathtt{M}_3|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{m}}|e_2\restriction_{\{\mathtt{f}_1', \dots, \mathtt{f}_k'\}}\rangle}\ (\mathtt{3lgor})$$

$$\frac{e(\mathtt{x}) = \underline{\mathtt{n}}}{\langle \mathtt{x}^{\iota}|e\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e\rangle}\ (\mathrm{Hgvar}) \qquad \frac{\langle \mathtt{hP}_1 \dots \mathtt{P}_k|e_0\restriction_{\{\mathtt{f}\}}\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle}{\langle \mathtt{fP}_1 \dots \mathtt{P}_k|e_0[\mathtt{f} := \mathtt{h}]\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1[\mathtt{f} := e_1(\mathtt{h})]\rangle}\ (\mathrm{Hvar}) \qquad \frac{\mathtt{h} \text{ fresh}, \quad \langle \mathtt{hNP}_1 \dots \mathtt{P}_k|e_0[\mathtt{h} := \mathtt{M}]\restriction_{\{\mathtt{f}\}}\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle}{\langle \mathtt{fP}_1 \dots \mathtt{P}_k|e_0[\mathtt{f} := \mathtt{M}^{\sigma \multimap \tau}\mathtt{N}]\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1[\mathtt{f} := (\pi_2 e_1(\mathtt{h}))]\restriction_{\{\mathtt{h}\}}\rangle}\ (\mathrm{Happ})$$

$$\frac{\langle \mathtt{M}|e_0\restriction_{\{\mathtt{f}\}}\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle}{\langle \mathtt{f}^{\iota \multimap \iota}\mathtt{M}|e_0[\mathtt{f} := \mathtt{s}]\rangle \Downarrow_{\mathtt{T}} \langle \mathtt{s}\underline{\mathtt{n}}|e_1[\mathtt{f} := \lceil \underline{\mathtt{n}}, \mathtt{s}\underline{\mathtt{n}} \rceil]\rangle}\ (\mathrm{Hs}) \qquad \frac{\langle \mathtt{M}|e_0\restriction_{\{\mathtt{f}\}}\rangle \Downarrow_{\mathtt{T}} \langle \mathtt{s}\,\underline{\mathtt{n}}|e_1\rangle}{\langle \mathtt{f}^{\iota \multimap \iota}\,\mathtt{M}|e_0[\mathtt{f} := \mathtt{p}]\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1[\mathtt{f} := \lceil \mathtt{s}\underline{\mathtt{n}}, \underline{\mathtt{n}} \rceil]\rangle}\ (\mathrm{Hp})$$

$$\frac{\langle \mathtt{fP}_1 \dots \mathtt{P}_k|e_0[\mathtt{f} := (\mathtt{M}[\mu F.\mathtt{M}/F])]\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle}{\langle \mathtt{fP}_1 \dots \mathtt{P}_k|e_0[\mathtt{f} := (\mu F.\mathtt{M})]\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle}\ (\mathrm{H}\mu) \qquad \frac{\mathtt{h} \text{ fresh}, \quad \langle \mathtt{hI}|e_0[\mathtt{h} := \mathtt{M}]\restriction_{\{\mathtt{f}\}}\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle \quad \underline{\mathtt{k}} = \pi_1\pi_1 e_1(\mathtt{h})}{\langle \mathtt{fM}|e_0[\mathtt{f} := \mathtt{which?}]\rangle \Downarrow_{\mathtt{T}} \langle \lceil \underline{\mathtt{n}}, \underline{\mathtt{k}} \rceil | e_1[\mathtt{f} := \lceil \lceil \lceil \underline{\mathtt{k}}, \underline{\mathtt{k}} \rceil, \underline{\mathtt{n}} \rceil, \lceil \underline{\mathtt{n}}, \underline{\mathtt{k}} \rceil \rceil \restriction_{\{\mathtt{h}\}}\rangle}\ (\mathrm{Hw})$$

$$\frac{\mathtt{z} \text{ fresh}, \quad \langle \mathtt{P}|e_0\restriction_{\{\mathtt{f}\}}\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{m}}|e_1\rangle \quad \langle \mathtt{M}[\mathtt{z}/\mathtt{x}]|e_1[\mathtt{z} := \underline{\mathtt{m}}]\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_2\rangle}{\langle \mathtt{fP}|e_0[\mathtt{f} := \lambda \mathtt{x}^{\iota}.\mathtt{M}^{\iota}]\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_2[\mathtt{f} := \lceil \underline{\mathtt{m}}, \underline{\mathtt{n}} \rceil]\restriction_{\{\mathtt{z}\}}\rangle}\ (\mathrm{H}\lambda_{\iota}^{\iota}) \qquad \frac{\mathtt{h,z} \text{ fresh}, \quad \langle \mathtt{P}_1|e_0\restriction_{\{\mathtt{f}\}}\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle \quad \langle \mathtt{hP}_2 \dots \mathtt{P}_k|e_1[\mathtt{z} := \underline{\mathtt{n}}, \mathtt{h} := \mathtt{M}[\mathtt{z}/\mathtt{x}]]\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_2\rangle}{\langle \mathtt{fP}_1 \dots \mathtt{P}_k|e_0[\mathtt{f} := \lambda \mathtt{x}^{\iota}.\mathtt{M}^{\sigma \multimap \tau}]\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_2[\mathtt{f} := \lceil e_2(\mathtt{z}), e_2(\mathtt{h}) \rceil]\restriction_{\{\mathtt{z,h}\}}\rangle}\ (\mathrm{H}\lambda_{\multimap}^{\iota})$$

$$\frac{\mathtt{g}' \text{ fresh}, \quad \langle \mathtt{M}[\mathtt{g}'/\mathtt{g}]|e_0[\mathtt{g}' := \mathtt{P}]\restriction_{\{\mathtt{f}\}}\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle}{\langle \mathtt{fP}|e_0[\mathtt{f} := \lambda \mathtt{g}^{\sigma \multimap \tau}.\mathtt{M}^{\iota}]\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1[\mathtt{f} := \lceil e_1(\mathtt{g}'), \underline{\mathtt{n}} \rceil]\restriction_{\{\mathtt{g}'\}}\rangle}\ (\mathrm{H}\lambda_{\iota}^{\multimap}) \qquad \frac{\mathtt{h,g}' \text{ fresh}, \quad \langle \mathtt{hP}_2 \dots \mathtt{P}_k|e_1[\mathtt{g}' := \mathtt{P}_1, \mathtt{h} := \mathtt{M}[\mathtt{g}'/\mathtt{g}]]\restriction_{\{\mathtt{f}\}}\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_2\rangle}{\langle \mathtt{fP}_1 \dots \mathtt{P}_k|e_0[\mathtt{f} := \lambda \mathtt{g}^{\sigma \multimap \sigma'}.\mathtt{M}^{\tau \multimap \tau'}]\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_2[\mathtt{f} := \lceil e_2(\mathtt{g}'), e_2(\mathtt{h}) \rceil]\restriction_{\{\mathtt{h,g}'\}}\rangle}\ (\mathrm{H}\lambda_{\multimap}^{\multimap})$$

Table 3. Tracing Semantics - Part (a) and Part (b).

Note that e associates to a variable $\varkappa$ a term $\mathtt{N}$ that can contain variables which are also in the domain of e. We show that we can avoid the use of cyclic environments. Let e_0 be an environment such that $dom(e_0) = \{\varkappa_1, \dots, \varkappa_n\}$, let $\langle \mathtt{M}|e\rangle$ be a state, let $\mathtt{N}$ be $\mathtt{M}[e(\varkappa_1)/\varkappa_1, \dots, e(\varkappa_n)/\varkappa_n]$. We define $\mathcal{T}$ to be the function from state to state, defined as follows: $\mathcal{T}(\langle \mathtt{M}|e\rangle) = \langle \mathtt{N}|e\restriction_{\{\mathrm{FV}(\mathtt{N})\}}\rangle$. We plan to transform a state $\langle \mathtt{M}|e\rangle$ in a closed ground term by iterating $\mathcal{T}$ until the environments becomes (eventually) empty. If such procedure is terminating then the state is said **acyclic**.

Lemma 9. *The evaluation of acyclic states only uses acyclic states.*

Proof. Since we add only fresh variables to the environments and the environments is always finite. □

From now on, we consider only acyclic states. Since $\langle \mathtt{M}|\sharp\rangle$ is trivially acyclic, we can define $\mathcal{T}_\infty(\langle \mathtt{M}|e\rangle)$ to be the closed ground term obtained by iterating $\mathcal{T}$ until the environment becomes empty.

Lemma 10. *Let* $\tau = \sigma_1 \multimap \dots \multimap \sigma_k \multimap \gamma$.
If $\langle \mathtt{f}^{\tau}\mathtt{P}_1 \dots \mathtt{P}_k|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle$ *then* $\mathtt{Chk}^{(\tau)}_{e_1(\mathtt{f})}(e_0(\mathtt{f})) \Downarrow \underline{0}$ *and* $\langle \mathtt{Sgl}_{e_1(\mathtt{f})}\mathtt{P}_1 \dots \mathtt{P}_k|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\restriction_{\{\mathtt{f}\}}\rangle$

Proof. Easy by induction on the tracing reduction rules. □

We can prove the trace evaluation machine to be correct and complete, by using the previous lemmas.

Theorem 10. *Let* $\mathcal{T}_\infty(\langle \mathtt{M}|e_0\rangle) = \mathtt{P}$.

Correctness. *If* $\langle \mathtt{M}|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle$ *then* $\mathtt{P} \Downarrow \underline{\mathtt{n}}$.
Completeness. *If* $\mathtt{P} \Downarrow \underline{\mathtt{n}}$ *then* $\langle \mathtt{M}|e_0\rangle \Downarrow_{\mathtt{T}} \langle \underline{\mathtt{n}}|e_1\rangle$.

Note that the tracing evaluation machine is a concrete improvement with respect to the evaluation machine presented in Table 1 and Table 2, in the sense that it prunes the infinite branching search trees of the evaluation rules for `which?` and ℓ`et-`ℓ`or`. So, it can be regarded as a guideline for a reasonable implementation.

Example Let us see the effectiveness of the tracing machine, by showing the evaluation of the term $\mathtt{which?}(\lambda \mathtt{f}^{\iota \multimap \iota}.\mathtt{s}(\mathtt{f}\underline{3}))$. In the following, we denote with e the environment $\sharp[\mathtt{f}' = \mathtt{I}]$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\langle \underline{3}|e\rangle \Downarrow_{\mathtt{T}} \langle \underline{3}|e\rangle} \quad \dfrac{}{\langle \mathtt{z}|e[\mathtt{z} = \underline{3}]\rangle \Downarrow_{\mathtt{T}} \langle \underline{3}|e[\mathtt{z} = \underline{3}]\rangle}\ (\mathbf{Hgvar})}{\langle \mathtt{f}'\underline{3}|e\rangle \Downarrow_{\mathtt{T}} \langle \underline{3}|\sharp[\mathtt{f}' = \lceil \underline{3}, \underline{3} \rceil]\rangle}\ (\mathbf{H}\lambda_{\iota}^{\iota})}{\langle \mathtt{s}(\mathtt{f}'\underline{3})|e\rangle \Downarrow_{\mathtt{T}} \langle \underline{4}|\sharp[\mathtt{f}' = \lceil \underline{3}, \underline{3} \rceil]\rangle}\ (\mathbf{s})}{\langle \mathtt{h\ I}|\sharp[\mathtt{h} = \lambda \mathtt{f}.\mathtt{s}(\mathtt{f}\underline{3})]\rangle \Downarrow_{\mathtt{T}} \langle \underline{3}, \sharp[\mathtt{h} = \lceil \lceil \underline{3}, \underline{3} \rceil, \underline{4} \rceil]\rangle}\ (\mathbf{H}\lambda_{\iota}^{\multimap})}{\langle \mathtt{which?}(\lambda \mathtt{f}^{\iota \multimap \iota}.\mathtt{s}(\mathtt{f}\underline{3}))|\sharp\rangle \Downarrow_{\mathtt{T}} \langle \lceil \underline{4}, \underline{3} \rceil | \sharp\rangle}\ (\mathbf{w})$$

It can be noted here that the rules $(\mathrm{H}\lambda_{\iota}^{\multimap})$ and $(\mathrm{H}\lambda_{\iota}^{\iota})$ are crucial to trace out the use of terms being substituted to the fresh head variables $\mathtt{h}$ and $\mathtt{f}'$. The rule $(\mathrm{H}\lambda_{\iota}^{\multimap})$ traces the term $\lambda \mathtt{f}.\mathtt{s}(\mathtt{f}\underline{3})$ applied to $\mathtt{I}$. The rule $(\mathrm{H}\lambda_{\iota}^{\iota})$ traces the term $\mathtt{I}$ applied to $\underline{3}$.

9. Related works

The study of the relations between languages and models is a classic theme in denotational semantics [15, 28]. The full abstraction for PCF has led to the development of very sophisticated semantics techniques (as [2, 23]) 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 [33] for the continuous Scott model, by Berry and Curien [9] for the sequential algorithms model, by Abramsky and McCusker [1] for a particular game model, by Longley [26] for the strongly stable model and by Paolini [29] for the stable model. Remark that all higher-type operators introduced in the works above [9, 26, 29, 33] cannot be directly interpreted in the linear model. This motivates 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. have proposed several syntactical linear languages in order to characterize different classes of computable functions [3–5]. Such languages are syntactically linear but do not have extra operators that are key ingredients of $\mathcal{SL}\mathrm{PCF}_\star$.

Two PCF-like languages embedding linearity notions have been proposed in [10, 11]. 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 [10, 11] 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.

10. Conclusions and Future Works

The results presented in this paper are part of a wider project (started with the works [17, 30]) aiming to extend the expressive power of linear programming languages. Our aim is to study particular denotational models embedding a notion of linearity, in order to extract programming languages that are fully abstract or fully complete (universal) with respect to the model, and that have new interesting operational features. Our study is also related to higher-type computability [25, 27], since the higher-type of new operators arising from these analysis.

A first interesting direction is to extend the results obtained in this paper in order to prove the universality of $\mathcal{SL}\mathrm{PCF}_\star$ 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 $\mathcal{SL}\mathrm{PCF}_\star$ semantics with respect to other model notions. In particular, we would pursue the study started in [31] about a model for $\mathcal{SL}\mathrm{PCF}_\star$ based on linear processes. Moreover, we would pursue the study about categorical models for $\mathcal{SL}\mathrm{PCF}_\star$ [18].

Last, we guess that an even more efficient evaluation of $\mathcal{SL}\mathrm{PCF}$ is possible. In particular, it would be interesting to study a new evaluation machine having an optimized memory management. That is, an evaluation machine that does not need neither fresh variables nor closures and where the heap will be used only for trace out sub-programs.

References

[1] S. Abramsky and G. McCusker. Linearity, sharing and state: a fully abstract game semantics for idealized algol with active expressions. *ENTCS*, 3, 1996.

[2] S. Abramsky, P. Malacaria, and R. Jagadeesan. Full abstraction for PCF. *I & C*, 163(2):409–470, 2000.

[3] S. Alves, M. Fernández, M. Florido, and I. Mackie. The power of linear functions. In *LNCS* 4207, pages 119–134. 2006.

[4] S. Alves, M. Fernández, M. Florido, and I. Mackie. Linear recursive functions. In *LNCS* 4600, pages 182–195. 2007.

[5] S. Alves, M. Fernndez, M. Florido, and I. Mackie. Gdel's system t revisited. *TCS*, 411(11-13):1484 – 1500, 2010.

[6] R. Amadio and P. L. Curien. *Domains and Lambda-Calculi* Cambridge University Press, 1998.

[7] G. Berry. Stable models of typed λ-calculi. In *LNCS* 62, pages 72–89. 1978.

[8] G. Berry. *Modèles completement adequats et stable du lambda-calcul typé*. PhD thesis, Université de Paris VII, France, 1979.

[9] G. Berry and P. L. Curien. Sequential algorithms on concrete data structures. *TCS*, 20:265–321, 1982.

[10] G. M. Bierman. Program equivalence in a linear functional language. *JFP*, 10(2):167–190, 2000.

[11] G. M. Bierman, A. M. Pitts, and C. V. Russo. Operational properties of lily, a polymorphic linear lambda calculus with recursion. *ENTCS*, 41(3), 2000.

[12] A. Bucciarelli and T. Ehrhard. Sequentiality and strong stability. In *Proc. LICS'91*, pages 138–145, 1991.

[13] A. Bucciarelli and T. Ehrhard. A theory of sequentiality. *TCS*, 113(2): 273–291, 7 June 1993.

[14] A. Bucciarelli, A. Carraro, T. Ehrhard, and A. Salibra. On linear information systems. In *EPTCS* 22, pages 38 – 48, 2010.

[15] P. L. Curien. Definability and full abstraction. *ENTCS*, 172:301–310, 2007.

[16] T. Ehrhard and L. Regnier. Differential interaction nets. *TCS*, 364(2): 166–195, 2006.

[17] M. Gaboardi and L. Paolini. Syntactical, operational and denotational linearity. In *Workshop LOGIC*. 2007.

[18] M. Gaboardi and M. Piccolo. Categorical models for a semantically linear λ-calculus. In *EPTCS* 22, pages 1 – 13, 2010.

[19] J. Y. Girard. Linear logic. *TCS*, 50:1–102, 1987.

[20] J. Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge University Press, 1989.

[21] J. R. Hindley. BCK-combinators and linear λ-terms have types. *TCS*, 64:97–105, 1989.

[22] M. Huth. Linear domains and linear maps. In *LNCS* 802, pages 438–453. 1993.

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

[24] J. W. Klop. New fix point combinators from old. In *Reflections on Type Theory*. Radboud University Nijmegen, 2007.

[25] J. R. Longley. Notions of computability at higher types I. In *Lecture Notes in Logic*, pages 32–142. 2000.

[26] J. R. Longley. The sequentially realizable functionals. *APAL*, 117: 1–93, 2002.

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

[28] C.-H. Luke Ong. Correspondence between operational and denotational semantics: the full abstraction for PCF. In *Handbook of Logic in Computer Science* pages 269–356. Oxford University Press, 1995.

[29] L. Paolini. A stable programming language. *I & C*, 204(3):339–375, 2006.

[30] L. Paolini and M. Piccolo. Semantically linear programming languages. In *PPDP'08*, pages 97–107. ACM, 2008.

[31] L. Paolini and M. Piccolo. A process-model for linear programs. In *LNCS* 5497, pages 289–305. 2009.

[32] M. Piccolo. *Linearity and Beyond in Denotational Semantics*. PhD thesis, Universitá di Torino/Université de Paris VII, 2009.

[33] G. D. Plotkin. LCF considerd as a programming language. *TCS*, 5: 225–255, 1977.

[34] D. Walker. *Advanced Topics in Types and Programming Languages*, chapter Substructural Type Systems. The MIT Press, 2005.