Realizability Models for a Linear Dependent PCF

Aloïs Brunel and Marco Gaboardi

1 LIPN - Paris XIII
2 University of Pennsylvania – Università di Bologna – Inria Epi Focus

Abstract. Recently, Dal Lago and Gaboardi have proposed a type system, named $d\ell\text{PCF}$ as a framework for implicit computational complexity. $d\ell\text{PCF}$ is a non-standard type system for PCF programs which is relatively complete with respect to complexity properties thanks to the use of linear types inspired by Bounded linear logic and dependent types à la Dependent ML. In this work, we adapt the framework of quantitative realizability in order to obtain a model for $d\ell\text{PCF}$. The quantitative realizability model aims at a better understanding of $d\ell\text{PCF}$ type decorations and at the same time giving an abstract semantical proof of intensional soundness.

1 Introduction

Implicit computational complexity and program resource consumption - The main goal of implicit computational complexity is to provide new characterizations of complexity classes that are abstract with respect to complexity measures and the underlying concrete model of computation. In the search towards this goal, many advancements obtained so far have largely contributed to the development of new tools to understand and analyze the resource consumption of programs.

Within this perspective, we can identify two approaches that have been fruitful thus far.

(i) Within the first approach one identifies some a priori restriction on the shape of programs, in the form of a syntactic criterion or typing relation, that ensures programs respecting the restriction to be in a particular complexity class.

(ii) Within the second approach one identifies some a posteriori criteria, in the form of a static program analysis, that ensures programs passing the criteria to be in a particular complexity class.

The two approaches have their pros and cons. On the one hand, by following the first approach one obtains constraints that are usually easy to check [4, 20, 14, 3]; these constraints however cut off many interesting programs and so this approach gives systems suffering from poor intensional expressivity [14].

For the second approach, once one has fixed the programming language (usually a first order language), one does not impose a priori restrictions on the
shape of programs that can be considered. That usually means these methods give characterizations of complexity classes with a better intensional expressivity even if they also cut some interesting program in this case [17, 21, 23]. However, the analysis used in this setting usually involves more complex verifications [1].

Relative completeness and $d\ell$-PCF - Recently, Dal Lago and Gaboardi in [8] proposed a type system, named $d\ell$-PCF, for a call-by-name version of PCF that combines the two approaches and pushes these ideas to the limit. $d\ell$-PCF can be considered a non-standard type system that unifies type inference and program analysis. This type system explores ideas combining linear types inspired by Bounded linear logic (BLL) [12, 9] with dependent types à la Dependent ML [26]. That is, the types contain index terms\(^3\) that are useful to reduce the problem of resource consumption of PCF terms to the problem of satisfying a set of inequalities. In order to achieve this goal, typing judgements in $d\ell$-PCF enrich PCF typing judgements by mean of additional information. An example of typing judgment in $d\ell$-PCF is the following:

$$\phi; \Phi; \Gamma \vdash \phi t : [a < J] \cdot \sigma \rightarrow \text{Nat}[K, H]$$

In particular, $\Phi$ is a set of inequalities over index terms which represent the side conditions under which the typing judgment is derivable, $\mathcal{E}$ is the equational theory giving meaning to index terms. The index term $I$ describes the weight associated with the proof proving the typing judgment, the weight can be seen as an abstraction of the complexity of the program. The notation $[a < J] \cdot \sigma$ is used instead of the more standard BLL notation $!_{a < J} \sigma$ and $K, H$ are index terms used to add additional information to the base type $\text{Nat}$ ensuring that the result of the computation is in the interval given by $K$ and $H$. Finally, $\phi$ contains the free variables of all the index terms appearing in the typing judgement.

The main property of $d\ell$-PCF is relative completeness. That means that the type system is able to analyze all the PCF terms with no a priori nor a posteriori restriction. This means that for every terminating PCF program there is a decoration in $d\ell$-PCF giving information about its complexity. The price to pay in order to obtain this property is obviously that the typability is no longer decidable. More concretely, $d\ell$-PCF reduces typability to constraint satisfiability for constraints that are in general non decidable. This suggest that $d\ell$-PCF should not be considered a type system but instead it should be considered a general framework useful to study resource consumption and to compare different approaches.

Besides relative completeness, $d\ell$-PCF is also intensionally sound. Anticipating on the next section, we can formulate the intensional soundness as follows.

**Theorem 1 (Intensional soundness).**

Let $\emptyset; \emptyset; \emptyset \vdash t : \text{Nat}[J, K]$ and $t \Downarrow^n m$. Then, $n \leq \|t\| \cdot (\|J\| + 1)$ and $\|J\| \leq m \leq \|K\|$.

\(^3\) Index terms are first-order terms drawn from a specific signature and whose meaning is given by means of an equational rewriting system.
The above theorem relates the typing judgments, and in particular the index terms, with both the intensional and the extensional semantics of the program. That is, given a program $t$ of PCF typable in $\text{d/PCF}$ with weight $I$ and type $\text{Nat}[J, K]$, we have that the weight $I$ bounds the number of evaluation steps while the index terms $J$ and $K$ give a bound on the value computed by $t$. The intensional soundness has been proved in [8] with respect to a call-by-name Krivine machine. That is, it has been shown that the reduction steps in the machine decrease a specific weight that can be obtained by considering typed machine configurations.

Quantitative realizability models - Starting from Kleene, the concept of realizability has been introduced in different forms and has proven very useful to prove properties of computational systems. In a series of recent works, Dal Lago and Hofmann have shown how to adapt Kleene Realizability to build quantitative models for subsystems of linear logic with restricted complexity. The models they propose contain a natural notion of resource (in the form of elements drawn from a resource monoid) that can be exploited to obtain semantical proofs of complexity soundness for these logics. Their models also permit to obtain a better understanding of the resource usage of the different logics. Starting from Dal Lago and Hofmann results, Brunel in [7] has extended the quantitative approach to Krivine’s Realizability, a version of realizability introduced by Krivine that aims at extending the proofs-as-programs correspondence to classical logic and set theory. He has also shown how the quantitative part relates to the notion of forcing.

Contributions - In this work, we adapt the framework of quantitative realizability to the case of $\text{d/PCF}$. In particular, we follow the approach developed by Brunel in [7] and we design a quantitative realizability model based on Krivine’s Realizability.

In order to deal with the generality of $\text{d/PCF}$ typing judgments, we need to define the realizability interpretation $|\sigma|_{E}^{\rho}$ of a type $\sigma$ as parametrized over an equational program $E$ and an assignment $\rho$ of index term variables to natural numbers. In this way, we can internalize the generality of the type system in the model.

The interpretation we define is sound with respect to $\text{d/PCF}$ typing judgments. Moreover, we show two forms of completeness of our class of quantitative models with respect to $\text{d/PCF}$: external completeness and internal completeness. External completeness corresponds to a bounded-time termination property of realizers. Internal completeness gives quantitative informations about the computed values and gives a concrete bound on the termination of realizers. Thanks to both external and internal completeness we give an abstract proof of $\text{d/PCF}$’s intensional soundness. Besides, using the relative completeness of $\text{d/PCF}$ we also prove that the type system and the realizability model carries exactly the same information when a universal equational program is considered. This last result represents a form of full abstraction and ensures that our model is an abstract tool that can be used in full generality to reason about $\text{d/PCF}$ programs.

The motivations for our study are twofold. On the one hand, we aim at better understanding the role of $\text{d/PCF}$ type decorations and the quantitative realizability
model is the most natural candidate. On the other hand what we aim to obtain is a framework allowing us to extend by means of quantitative information some recent results obtained in the framework of complexity preserving certification [15].

Outline - In Section 2, we recall the system dℓPCF and its main properties. In Section 3, we introduce our quantitative realizability model and we show that it is sound with respect to dℓPCF. In Section 4, we use the model defined in the previous section to give a semantical proof of dℓPCF’s intensional soundness; moreover we show that when a universal equational program is considered, realizability and typing coincide. Finally, in Section 5, we recast some related works and we conclude.

2 dℓPCF

The type system dℓPCF is a refinement of the type system for PCF by means of index terms whose semantics, in turn, is defined by an equational program over a signature Σ.

Formally, a signature Σ is a pair (S, α) where S is a finite set of function symbols and α : S → N assigns an arity to every function symbol. Index terms on a given signature Σ = (S, α) are generated by the following grammar:

\[ I, J, K ::= a \mid f(I_1, \ldots, I_{\alpha(f)}) \mid \sum_{a < 1} J \mid a^J \sum_{a < 1} K \]

where f ∈ S and a is a variable drawn from a set \( V \) of index variables. We assume the symbols 0, 1 (with arity 0) and +, ÷ (with arity 2) are always part of Σ. An index term in the form \( \sum_{a < 1} J \) is a bounded sum, while one in the form \( a^J \sum_{a < 1} K \) is a forest cardinality. For every natural number n, the index term n is just 1 + 1 + … + 1 n-times.

Index terms are meant to denote natural numbers, possibly depending on the (unknown) values of variables. Variables can be instantiated with other index terms, e.g. \( I(J/a) \). So, index terms can also act as first order functions. The meaning of the function symbols from Σ is induced by an equational program E. Formally, an equational program E over a signature Σ and a set of variables \( \mathcal{V} \) is a set of equations in the form \( t = s \) where both t and s are terms built from variables and the symbols in Σ. We are interested in equational programs guaranteeing that, whenever symbols in Σ are interpreted as partial functions over \( \mathbb{N} \) and 0, 1, + and ÷ are interpreted in the usual way, the semantics of any function symbol f can be uniquely determined from E. This can be guaranteed by, for example, taking E as an Herbrand-Gödel scheme or as an orthogonal constructor term rewriting system.

A bounded sum \( \sum_{a < 1} J \) is an index term whose value is simply the sum of all possible values of J with a taking the values from 0 up to I, excluded.

The forest cardinality will be used to describe function call trees. Informally, \( a^J \sum_{a < 1} K \) is an index term denoting the number of nodes in a forest composed of
J trees described using K. All the nodes in the forest are (uniquely) identified by natural numbers. These are obtained by consecutively visiting each tree in pre-order, starting from I. The term K has the role of describing the number of children of each forest node n by properly instantiating the variable a, e.g. the number of children of the node 0 is K{0/a}. Formally, the meaning of a forest cardinality is defined by the following two equations:

\[ K_a \cdot 0 = 0 \]
\[ K_a \cdot (\uparrow + 1) = (\uparrow + 1) + (I + \uparrow + 1) \cdot K_a \]

and it represents the number (i.e. the cardinality) of nodes in the forest described by I.

The expression \([I]_\rho^\Sigma\) denotes the meaning of I, defined by induction along the lines of the previous discussion, where \(\rho : V \rightarrow N\) is an assignment and \(\Sigma\) is an equational program giving meaning to the function symbols in I. Since \(\Sigma\) does not necessarily interpret such symbols as total functions, and moreover, the value of a forest cardinality can be undefined, \([I]_\rho^\Sigma\) can be undefined itself. A constraint is an inequality in the form I \(\leq J\). A constraint is true in an assignment \(\rho\) if \([I]_\rho^\Sigma\) and \([J]_\rho^\Sigma\) are both defined and the first one is smaller or equal than the second one. Now, for a subset \(\phi\) of \(V\), and for a set \(\Phi\) of constraints involving variables in \(\phi\), the expression \(\phi : \Phi \models^\Sigma I \leq J\) denotes the fact that the truth of I \(\leq J\) semantically follows from the truth of the constraints in \(\Phi\). Similarly, one can define the meaning of expressions like \(\phi : \Phi \models^\Sigma I = J\) or \(\phi : \Phi \models^\Sigma I \simeq J\), the latter standing for the equality of I and J in the sense of Kleene, i.e. I and J are equal if when I (resp. J) is defined then so is J (resp. I) and they are equal. When both \(\phi\) and \(\Phi\) are empty, such expressions can be written in a much more concise form, e.g. I \(\simeq J\) stands for \(\emptyset ; \emptyset \models^\Sigma I \simeq J\). The expression \(\phi : \Phi \models^\Sigma I = I\) indicates that (the semantics of) I is defined for the relevant values of the variables in \(\phi\) under the constraints in \(\Phi\); this is usually written as \(\phi : \Phi \models^\Sigma I \iff I\). Finally, we say that \(\rho\) satisfies \(\phi : \Phi\) and we note \(\rho \models^\Sigma \phi : \Phi\) if \(\phi \subseteq dom(\rho)\) and for each I \(\leq J \in \Phi\), we have \([I]_\rho^\Sigma \leq [J]_\rho^\Sigma\).

From now on, all the definitions will be parametric on an equational program \(\Sigma\) over a signature \(\Sigma\). For the sake of simplicity, we will often avoid to mention \(\Sigma\) explicitly. Terms are generated by the following grammar:

\[ t ::= x | n | s(t) | p(t) | \lambda x. t | tu | \text{if} t \text{ then } u \text{ else } v | \text{fix } x. t \]

where \(n\) ranges over natural numbers and \(x\) ranges over a set of variables. A notion of size \(|t|\) for a term \(t\) will be useful in the sequel. This can be defined as follows:

\[ |x| = |p| = 1; |s| = 2; |n| = n; |s(t)| = |s| + |t| + 1; |p(t)| = |p| + |t| + 1; |\lambda x. t| = |t| + 1; |tu| = |t| + |u| + 1; |\text{if} t \text{ then } u \text{ else } v| = |t| + |u| + |v| + 1; |\text{fix } x. t| = |t| + 1. \]
\( \text{dlPCF} \) can be seen as a refinement of PCF obtained by a decoration using linear types of its type derivation. Basic and modal types are defined as follows:

\[
\begin{align*}
\sigma, \tau &::= \text{Nat}[I, J] | A \rightarrow \sigma & \text{basic types} \\
A, B &::= [a < I] \cdot \sigma & \text{modal types}
\end{align*}
\]

where \( I, J \) range over index terms and \( a \) ranges over index variables. We will write the symbol \( \varsigma \) when we want to talk about types without distinguishing between basic and modal types. \( \text{Nat}[I] \) is syntactic sugar for \( \text{Nat}[I, I] \). We will use the convention that \( r \cdot a \cdot I \cdot s \) has precedence over \( \rightarrow \), e.g. \( [a < I] \cdot \sigma \rightarrow \tau \) stands for \( ([a < I] \cdot \sigma) \rightarrow \tau \).

In the typing rules, modal types need to be manipulated in an algebraic way. For this reason, two operations on modal types need to be introduced. The first one is a binary operation \( Z \) on modal types. Suppose that \( A = [a < I] \cdot \gamma \{0/c\} \) and that \( B = [b < J] \cdot \gamma \{1 + b/c\} \). In other words, \( A \) consists of the first \( I \) instances of \( \gamma \), i.e. \( \gamma \{0/c\}, \ldots, \gamma \{1-1/c\} \) while \( B \) consists of the next \( J \) instances of \( \gamma \), i.e. \( \gamma \{1+0/c\}, \ldots, \gamma \{1+J-1/c\} \). Their sum \( A \bowtie B \) is naturally defined as a modal type consisting of the first \( I + J \) instances of \( \gamma \), i.e. \( [c < 1+J] \cdot \gamma \). An operation of bounded sum on modal types can be defined by generalizing the idea above: suppose that

\[
A = [b < J] \cdot \sigma \{ \sum_{a < a} J \{d/a\} + b/c \}.
\]

Then its bounded sum \( \sum_{a < a} A \) is \( [c < \sum_{a < a} J] \cdot \sigma \).

Central to \( \text{dlPCF} \) is the notion of subtyping. An inequality relation \( \preceq \) between (basic and modal) types can be defined using the formal system in Figure 1. This relation corresponds to lifting index inequalities at the type level. The equality

\[
\phi; \Phi \vdash^\preceq K \leq I \\
\phi; \Phi \vdash^\preceq J \leq H
\]

\[
\phi; \Phi \vdash^\preceq \text{Nat}[I, J] \preceq \text{Nat}[K, H] \\
\phi; \Phi \vdash^\preceq A \rightarrow \sigma \preceq B \rightarrow \tau \\
\phi, a; \Phi, a < J \vdash^\preceq \sigma \preceq [a < J] \cdot \tau
\]

*Fig. 1.* The subtyping relation

\( \phi; \Phi \vdash \sigma \preceq \tau \) holds when both \( \phi; \Phi \vdash \sigma \preceq \tau \) and \( \phi; \Phi \vdash \tau \preceq \sigma \) can be derived from the rules in Figure 1.

*Typing judgements* of \( \text{dlPCF} \) are expressions in the form

\[
\phi; \Phi; \Gamma \vdash^\preceq t : \sigma
\]

where \( \Gamma \) is a *typing context*. That is, a set of term variable assignments of the shape \( x : A \) where each variable \( x \) occurs at most once. The expression (1) can be informally read as follows: for every values of the index variables in \( \phi \) satisfying \( \Phi \), \( t \) can be given type \( \sigma \) and cost \( I \) once its free term variables have types as in \( \Gamma \).
In proving this, equations from $\mathcal{E}$ can be used. Typing rules are in Figure 2, where binary and bounded sums are used in their natural generalization to contexts. A type derivation is nothing more than a tree built according to typing rules. A precise type derivation is a type derivation such that all premises of the form $\sigma \subseteq \tau$ (respectively, in the form $I \leq J$) are required to be in the form $\sigma \equiv \tau$ (respectively, $I = J$). As a last remark, note that each rule can be seen as a decoration of a rule of ordinary PCF.

Derivations in df/PCF enjoy substitution properties both at the level of terms and at the level of index terms.

**Lemma 1 (Substitution).** Let $\phi, a; \Phi, a < I; \emptyset \vdash^{\mathcal{E}}_K t : \sigma$ and $\phi; \Phi; x : [a < I] : \sigma, \Delta \vdash^{\mathcal{E}}_K u : \tau$. Then we have $\phi; \Phi; \Delta \vdash^{\mathcal{E}}_K u[t/x] : \tau$ with $\phi; \Phi \vdash^{\mathcal{E}}_K H \leq K + I + \sum_{a < 1} J$.

Thanks to the above Substitution Lemma we can prove, as expected, that reductions of df/PCF terms preserve the types. In particular, this holds with respect to a weak reduction with which df/PCF is equipped [8].

**Theorem 2 (Subject Reduction).** Let $\phi; \Phi; \emptyset \vdash^{\mathcal{E}}_K t : \sigma$ and $t \rightarrow u$. Then, $\phi; \Phi; \emptyset \vdash^{\mathcal{E}}_K u : \sigma$, where $\phi; \Phi \vdash^{\mathcal{E}}_K \leq I$.

Notice that the above theorem says something more than the usual type preservation theorems. Indeed, it says that the weight can change during the reduction but it also ensures that it cannot increase.
Besides, the index terms can be used to obtain precise information about the reduction of dℓPCF programs by means of the abstract machine KPCF inspired by Krivine’s Machine and defined in Figure 3.

The configurations of the machine KPCF, ranged over by \(C, D, \ldots\), are triples \(C = (t, \mu, \xi)\) where \(\mu\) and \(\xi\) are two additional constructions: \(\mu\) is an environment, that is a (possibly empty) finite sequence of closures; while \(\xi\) is a (possibly empty) stack of contexts. Stacks are ranged over by \(\xi, \theta, \ldots\). A closure, as usual, is a pair \(c = (t, \mu)\) where \(t\) is a term and \(\mu\) is an environment. A context is either a closure, a term \(s\), a term \(p\), or a triple \((u, v, \mu)\) where \(u, v\) are terms and \(\mu\) is an environment. In the sequel we will consider configurations that can be typed by means of PCF types (a precise definition is given in [8]) and we denote the set of such configurations as \(\text{Conf}_{\text{PCF}}\).

As usual, the symbol \(\Rightarrow^*\) denotes the reflexive and transitive closure of the transition relation \(\Rightarrow\). The relation \(\Rightarrow^*\) implements weak-head reduction. Weak-head normal form and the normal form coincide for programs. So the machine KPCF is a correct device to evaluate programs. For this reason, the notation \(t \Downarrow n\) can be used as a shorthand for \((p, t, \epsilon, \epsilon) \Rightarrow^* (p, n, \mu, \epsilon)\). Moreover, notations like \(C \Rightarrow^*_n\) will be used to stress that \(C\) reduces to an irreducible configuration in exactly \(n\) steps.

In the sequel we will also need to distinguish variable steps from the others. For this reason we write \(\Rightarrow_v\) for a reduction step obtained by applying the variable rule while we write \(\Rightarrow_r\) for a reduction step obtained by applying any other rule except the variable rule. Finally, if \(C\) is a configuration, we use the notation \(C \Downarrow^* (v, \mu, \epsilon)\) to denote the fact that \(C\) reduces to the value \(v\) and uses during the reduction \(n\) variable steps. We define \(C \Downarrow^* (v, \mu, \epsilon)\) such that \(C \Downarrow^*_n (v, \mu, \epsilon)\).

Index terms ensure that the typing judgements give precise information about the complexity of dℓPCF terms as stated by the following theorem.

**Theorem 3 (Intensional soundness).** Let \(\emptyset; \emptyset \vdash t : \text{Nat}[J, K]\) and \(t \Downarrow^n m\). Then, \(n \leq |t|: (|J| + 1)\) and \(|J| \leq m \leq |K|\).

In [8] the above result has been shown by means of syntactical techniques. In particular, the idea of the proof given there is to show that by performing a variable step the weight of a typing judgement can be decreased while every other step leaves the weight unchanged (but can increase the size of a configuration at most by a quantity equal to the size of the initial term. See [8] for details.). One of our goals in this article is to prove the above theorem by using semantics tools. Central to our proof is the following lemma that makes explicit what we described above.

**Lemma 2.** Let \(\emptyset; \emptyset \vdash t : \text{Nat}[J, K]\) such that \((t, \epsilon, \epsilon) \Downarrow^* r\). If \(t \Downarrow^n m\), then \(n \leq |t|: (r + 1)\).

The design of the index term decorations and of the dℓPCF type system has been motivated by the search for a relatively complete type system for complexity. This property can be proved by considering a universal equational program \(U\) (i.e. an equational program able to simulate all the equational programs including itself).
Theorem 4 (Relative Completeness for Programs). Let \( t \) be a PCF program such that \( t \downarrow^m \). Then, there exist two index terms \( I \) and \( J \) such that \( \|I\|_t \leq n \) and \( \|J\|_t = m \) and such that the term \( t \) is typable in \( d\ell \text{PCF} \) as \( \emptyset; \emptyset \vdash^t t : \text{Nat}[J] \).

Interestingly, this property does not only hold for programs but it also holds for functions.

Theorem 5 (Relative Completeness for Functions). Suppose that \( t \) is a PCF term such that \( \vdash t : \text{Nat} \rightarrow \text{Nat} \). Moreover, suppose that there are two (total and computable) functions \( f, g : \mathbb{N} \rightarrow \mathbb{N} \) such that \( t \downarrow^{g(n)} f(n) \), there are terms \( I, J, K \) with \( \|I + J\| \leq g \) and \( \|K\| = f \), such that
\[
\emptyset; \emptyset \vdash^t t : [b < J] \cdot \text{Nat}[a] \rightarrow \text{Nat}[K].
\]

\begin{center}
<table>
<thead>
<tr>
<th>Term</th>
<th>Environment</th>
<th>Stack</th>
<th>Term</th>
<th>Environment</th>
<th>Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>( tu )</td>
<td>( \mu )</td>
<td>( \xi )</td>
<td>( t )</td>
<td>( \mu )</td>
<td>( (u, \mu) \cdot \xi )</td>
</tr>
<tr>
<td>( \lambda x.t )</td>
<td>( \mu )</td>
<td>( c \cdot \xi )</td>
<td>( t )</td>
<td>( c \cdot \mu )</td>
<td>( \xi )</td>
</tr>
<tr>
<td>( x )</td>
<td>( (t_0, \mu_0) \cdots (t_\eta, \mu_\eta) )</td>
<td>( \xi )</td>
<td>( t_x )</td>
<td>( \mu_x )</td>
<td>( \xi )</td>
</tr>
<tr>
<td>if ( z ) then ( u ) else ( v )</td>
<td>( \mu )</td>
<td>( \xi )</td>
<td>( t )</td>
<td>( \mu )</td>
<td>( (u, v, \mu) \cdot \xi )</td>
</tr>
<tr>
<td>fix ( x.t )</td>
<td>( \mu )</td>
<td>( s \cdot \xi )</td>
<td>( t \cdot x.t, \mu \cdot \mu )</td>
<td>( \xi )</td>
<td></td>
</tr>
<tr>
<td>( n )</td>
<td>( \mu )</td>
<td>( \mu \cdot \xi )</td>
<td>( n )</td>
<td>( \mu \cdot \xi )</td>
<td></td>
</tr>
<tr>
<td>( 0 )</td>
<td>( \mu )</td>
<td>( (t, u, \nu \cdot \xi) \cdot \xi \rightarrow \mu \cdot \nu \cdot \xi )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n \cdot 1 )</td>
<td>( \mu )</td>
<td>( (t, u, \nu \cdot \xi) \rightarrow u \cdot \nu \cdot \xi )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( s(t) )</td>
<td>( \mu )</td>
<td>( \xi )</td>
<td>( t )</td>
<td>( \mu )</td>
<td>( s \cdot \xi )</td>
</tr>
<tr>
<td>( p(t) )</td>
<td>( \mu )</td>
<td>( \xi )</td>
<td>( t )</td>
<td>( \mu )</td>
<td>( p \cdot \xi )</td>
</tr>
</tbody>
</table>
\end{center}

Fig. 3. The \( K_{\text{PCF}} \) machine transition steps.

3 Quantitative realizability model for \( d\ell \text{PCF} \)

The realizability model we present in this section is inspired by Krivine’s realizability [18]. As in Krivine’s realizability, we build the model around the Krivine machine. Hence the machine \( K_{\text{PCF}} \) presented in the previous section becomes at the same time the evaluation medium for \( d\ell \text{PCF} \) terms and the basis of the realizability machinery. To define the interpretation and derive the properties we are interested in, we first need to extend the machine \( K_{\text{PCF}} \). We add to the set of closures a special closure \( \dagger \), named \textit{daimon}, well typed for every PCF type and which has no corresponding reduction rule in the machine \( K_{\text{PCF}} \). Hence, once in head position, the daimon blocks the computation. It is in a sense the dual of the empty stack. Whereas in ludics [11] the daimon empties the context, we choose it not to do so, not because it would break our model, but because we just do not need it. Even if it has no computational behavior, it will permit to evaluate under a \( \lambda \) binder. This is what says the following lemma.
Lemma 3. If \((t, \nu, \xi, \epsilon) \vdash^m\), then \((t, \nu, \epsilon) \vdash^n\) with \(n \leq m\).

Proof. Easy by induction on \(n\) and inspection of the \(K_{\text{PCF}}\) machine rules. \(\square\)

We can now start to define the realizability model. The core of biorientation-based models is the notion of orthogonality between closures and stacks. Here, we pair closures and stacks with index terms. Index terms are used in a similar way to monoid elements in [10] and [7].

Definition 1.

- A **weighted closure** is a pair \((c, I)\) where \(c\) is a closure and \(I\) is an index term. The set of weighted closures is denoted by \(\Lambda\).
- A **weighted stack** is a pair \((\xi, J)\) where \(\xi\) is a stack and \(J\) an index term. The set of weighted stacks is denoted by \(\Pi\).

Remark 1. Notice that in the previous definition, the index terms are possibly open.

Krivine’s realizability is usually parametric over a subset \(\perp\) of machine’s configurations. Different sets \(\perp’\) and \(\perp''\) represent different notions of computational correctness. Configurations alone are not sufficient to track quantitative informations. For this reason we extend the definition of \(\perp\) to include also an additional information represented by a natural number.

Definition 2. A **quantitative pole** is a set \(\perp \subseteq \text{Conf}_{\text{PCF}} \times \mathbb{N}\) such that:

- If \(n \leq m\) and \((C, n) \in \perp\), then \((C, m) \in \perp\).
- If \(C \rightarrow_c C’\) and \((C’, n) \in \perp\), then \((C, n+1) \in \perp\).
- If \(C \rightarrow_e C’\) and \((C’, n) \in \perp\), then \((C, n) \in \perp\).

Like the in the usual notion of pole [18], we ask \(\perp\) to be closed under anti-evaluation. However, it is worth noticing that only anti-evaluation on variable steps requires to increment the quantitative information. Indeed, we only want to count variable steps, as this information is sufficient as stressed by Lemma 2.

In the rest of this section, the definitions are parametric in the quantitative pole. In the next section, we will fix a particular quantitative pole \(\perp\) in order to derive the complexity results about typable dℓPCF terms.

A choice of quantitative pole induces a notion of orthogonality. In order to define it, we need to be able to obtain a natural number from an index term. This is achieved by using the interpretation function \([\_\_]^E_\rho\). For this reason, the quantitative notion of orthogonality is naturally parametrized over an equational program \(E\) and an assignment \(\rho\).

Definition 3. A **weighted closure** \((c, I)\) is \(E\)-\(\rho\)-orthogonal to a weighted stack \((\xi, J)\) iff:

- \(\text{FV}(I) \cup \text{FV}(J) \subseteq \text{dom}(\rho)\)
- \(((c, \xi), [I + J]^E_\rho) \in \perp\)
We use the notation \((c, I) \perp_{\rho} (\xi, J)\) to indicate that \((c, I)\) and \((\xi, J)\) are \(E\)-\(\rho\)-orthogonal.

Notice that the notion of \(E\)-\(\rho\)-orthogonality makes sense only when \([I + J]_{\rho}^E\) is defined. This is the case also for all the other notions that we introduce in the sequel of this section and the next one.

In what follows, we will in general assume that the equational program \(E\) is given so we will simply write \([I]_{\rho}\) for \(\rho\)-orthogonality. Substitution behaves well with respect to \(\rho\)-orthogonality.

**Lemma 4.** The following two properties are equivalent:

- \((c, I) \perp_{\rho[a\leftarrow n]} (\xi, J)\)
- \((c, I\{n/a\}) \perp_{\rho} (\xi, J\{n/a\})\)

**Proof.** It follows easily by the fact that for every index term \(I\) such that \(\text{FV}(I) \subseteq \text{dom}(\rho) \cup \{a\}\) we have \([I]_{\rho[a\leftarrow n]} = [I\{n/a\}]_{\rho}\). □

In order to define our quantitative realizability model we need to interpret types by sets of weighted closures. We then need additional operations for sets of weighted closures and weighted stacks.

**Definition 4.** If \(X\) is a set of weighted closures, we define its upward closure \(X^\rho\) as

\[
X^\rho = \{ (c, I) \mid \exists J, (c, J) \in X \land [J]_{\rho} \subseteq [I]_{\rho} \}
\]

Notice that the upward closure operator is monotonic, i.e. \(X \subseteq Y\) implies \(X^\rho \subseteq Y^\rho\) and idempotent, i.e. \(X^\rho = \overline{X^\rho}\). As usual, the orthogonality relation can be extended to sets of weighted closures and weighted stacks.

**Definition 5.**

- If \(X\) is a set of weighted closures, then its \(\rho\)-orthogonal set \(X^{\perp_{\rho}}\) is defined as:

\[
X^{\perp_{\rho}} = \{ (\xi, J) \mid \forall (c, I) \in X, (c, I) \perp_{\rho} (\xi, J) \}
\]

- If \(X\) is a set of weighted stacks, then its \(\rho\)-orthogonal set \(X^{\perp_{\rho}}\) is defined as:

\[
X^{\perp_{\rho}} = \{ (c, I) \mid \forall (\xi, J) \in X, (c, I) \perp_{\rho} (\xi, J) \}
\]

- A set of weighted closures \(X\) is a \(\rho\)-behavior if \(X^{\perp_{\rho}} = X\).

The extensions of \(\rho\)-orthogonality to sets of weighted closures and weighted stacks enjoy the usual properties of orthogonality. In particular we know that each set \(Y\), which is the orthogonal of a set \(X\), i.e. \(Y = X^{\perp_{\rho}}\), is also a \(\rho\)-behavior. Moreover, we have the following important property.

**Lemma 5.** Let \(\rho\) be a substitution and \(X\) be a \(\rho\)-behavior. If \((c, I) \in X\) and \(J\) is an index term such that \(\text{FV}(J) \subseteq \text{dom}(\rho)\) and \([I]_{\rho} \subseteq [J]_{\rho}\), then \((c, J) \in X\).
Proof. It follows easily by the definition of ρ-behavior (Definition 5) and by the definition of quantitative pole (Definition 2).

Now, we have all the components to define an interpretation of dℓPCF basic and modal types.

Definition 6 (Interpreting Types). Given a type ς and an assignment ρ such that FV(ς) ⊆ dom(ρ), the interpretation |ς|ρ of ς in ρ is defined by the rules in Figure 4.

Remark 2. Notice that we do not interpret modal types over ρ-behaviors but nonetheless when ς does not contain any positive occurrence of a modal type, |ς|ρ is a ρ-behavior. The reason why we don’t interpret modal types as behaviors (for instance by using a biorthogonality closure) is that it would be impossible to prove the soundness in the case of rules that manipulate modal types in the typing context. This is an issue related to call-by-name, and more can be found about it in [7].

The interpretation of types has some interesting properties with respect of the quantitative information. In particular, it inherits some properties of the orthogonality relation.

Lemma 6. Given a type ς and an assignment ρ, if (c, K) ∈ |ς|ρ(ς→ς) then (c, K{ς/a}) ∈ |ς|ρ(ς→ς).

Proof. By induction on the structure of ς. The cases where ς is either a type Nat[I, J] or a type A → τ follow easily by ρ-orthogonality definition (Definition 5) and by Lemma 4.

The case ς = [b < I].σ is more interesting. Consider (c, K) ∈ |[b < I].σ|ρ(ς→ς), we have two possible cases. If c = * then the conclusion follows immediately by the definition of upward closure (Definition 4). Otherwise, by definition of upward closure we know that |K|ρ(ς→ς) ⊆ |I + J|ρ(ς→ς) for some J such that (c, J) ∈ ∩m∈[ς→ς]|K|ρ(ς→ς,m). By induction hypothesis and since |[b < I].σ|ρ(ς→ς) we clearly have (c, J{n/a}) ∈ ∩m∈[ς→ς]|σ{n/a}|ρ(ς→ς). So, by definition of interpretation we have

\[\left( c, \sum_{b < I} J{n/a} + I{n/a} \right) \in |[b < I].σ|ρ(ς→ς)\]
Finally, by upward closure definition since
\[
\left[ \sum_{b \in I[n/a]} J[n/a] + I[n/a] \right]_{\rho} \leq \left[ K[n/a] \right]_{\rho}
\]
we can conclude \((c, K[n/a]) \in \left[ [b < 1], \sigma \right]\{n/a\}_{\rho}\).

The type system of d\(\text{PCF}\) uses the subtyping relation extensively. For this reason, an important milestone in showing that our realizability definition gives a model of d\(\text{PCF}\) is to show that it is sound with respect to the subtyping relation. In particular, as already stressed, we are interested in having this correspondence only when we are able to satisfy the given constraints. This is formally stated by requiring that the assignment \(\rho\) satisfies the constraints in \(\Phi\). The soundness with respect to the subtyping is given by the following theorem.

**Theorem 6 (Subtyping soundness).** Suppose \(\phi; \Phi \vdash \sigma \subseteq \tau\). Then \(\rho \models \phi; \Phi\) implies \([]\sigma\] \(\subseteq [\tau\]_{\rho}\).

**Proof.** By induction on the derivation with conclusion \(\phi; \Phi \vdash \sigma \subseteq \tau\). We show just the base case and the modal case. The linear arrow case follows as usual by the \(\rho\)-orthogonality properties.

**Case**
\[
\phi; \Phi \vdash K \leq I \quad \phi; \Phi \vdash J \leq H
\]

Because we have \([K]_{\rho} \leq [I]_{\rho}\) and \([J]_{\rho} \leq [H]_{\rho}\), it is immediate that if \([I]_{\rho} \leq n \leq [J]_{\rho}\), then \([K]_{\rho} \leq n \leq [H]_{\rho}\). By the orthogonality properties, we have \([\text{Nat}[I, J]]_{\rho} \leq [\text{Nat}[K, H]]_{\rho}\).

**Case**
\[
\phi, a; \Phi, a < J \vdash \sigma \subseteq \tau \quad \phi; \Phi \vdash J \leq I
\]

We want to prove that
\[
\{ (c, \sum_{a < I} K + I) \mid (c, K) \in \bigcap_{n < [I]} [\sigma|_{p(a \to n)}] \} \subseteq \{ (c, \sum_{a < J} K + J) \mid (c, K) \in \bigcap_{n < [J]} [\tau|_{p(a \to n)}] \}
\]

Then, the conclusion will follows by monotonicity and idempotency of the upward closure operator.

Consider \((c, K) \in \bigcap_{n < [I]} [\sigma|_{p(a \to n)}].\) By induction hypothesis we have \([\sigma|_{p(z \to n)}] \subseteq [\tau|_{p(z \to n)}]\) for each \(n < [I]_{\rho}\). Moreover, \(\rho \models \phi; \Phi\) clearly implies \([I]_{\rho} \subseteq [J]_{\rho}\).

So, we have \((c, K) \in \bigcap_{n < [I]} [\tau|_{p(a \to n)}].\) Hence, \((c, \sum_{a < J} K + J) \in [a < J] \cdot [\tau]_{\rho}\).

Since \(\sum_{a < J} K + J \leq \sum_{a < J} K + 1\), by monotonicity and idempotency of the upward closure operator we can conclude \((c, \sum_{a < 1} K + 1) \in [a < J] \cdot [\tau]_{\rho}\). □
Thanks to the soundness of the subtyping relation we can now prove that our realizability model is sound also with respect to the $\mathsf{d/PCF}$ type system.

**Theorem 7 (Soundness).** Suppose $\phi; \Phi; x_1 : A_1, \ldots, x_n : A_n \vdash^K t : \sigma$. Let $\rho \models \phi; \Phi$ and $(c_i, J_i) \in [A_i]_\rho$. Then

$$ (t, [x_1 := c_1, \ldots, x_n := c_n], K + \sum_{1 \leq i \leq n} J_i) \in [\sigma]_\rho $$

**Proof.** The proof is by generalized induction on the value of $[K]_\rho$ with further induction on the derivation proving $\phi; \Phi; x_1 : A_1, \ldots, x_n : A_n \vdash^K t : \sigma$. All the cases requires some manipulations of the index terms. We show few representative cases.

**Case $[K]_\rho = 0$**

**Subcase**

$$ \phi; \Phi \vdash^K [a < 1] \cdot \sigma \subseteq [a < 1] \cdot \tau $$

For simplicity, suppose $n = 0$. Consider $(c, H) \in [[a < 1] \cdot \sigma]_\rho$. The case $c = 1$ is easy. So, consider the case $c \neq 1$. By assumption and by Subtyping soundness Theorem 6, we know that $(c, H) \in [[a < 1] \cdot \tau]_\rho$. This means that there exists some $K$ such that $\sum_{a < 1} K + 1 = K[0/a] + 1 \leq H$ and $(c, K) \in [\tau]_{\rho[0 < a]}$. But by Lemma 6, that means $(c, K[0/a]) \in [\tau[0/a]]_\rho$. Hence, by anti-reduction, that means $(x, [x := c], K[0/a] + 1) \in [\tau[0/a]]_\rho$. But because $K[0/a] + 1 \leq H$, we finally obtain $(x, [x := c], H) \in [\tau[0/a]]_\rho$.

**Subcase**

$$ \phi; b; \Phi; b < L; \Gamma, x : [a < P] \cdot \sigma \vdash^K t : \tau $$

Without loss of generality and by definition of bounded sum we can consider the case where $\Gamma = y : [a < H] \cdot \delta \{a + \sum_{d < b} \mathsf{H}[d/a]/a\}$ and $\Sigma = y : [a < \sum_{b < L} H] \cdot \delta$. Consider $(c, N) \in [[a < \sum_{b < L} H] \cdot \delta]_\rho$. By the interpretation definition and by some manipulation of the index terms we have that for every $n < [\mathsf{L}]_\rho$:

$$ (c, \sum_{a < \mathsf{H}[n/b]} M[n/b] + \mathsf{H}[n/b]) \in [[a < H] : \delta \{a + \sum_{d < b} \mathsf{H}[d/a]/a\}]_\rho[\mathsf{L} \cdot n] $$

for some $M$ such that $\sum_{b < L} (\sum_{a < H} M + \mathsf{H}) \cong \mathsf{N}$. 

By using some manipulations of the indices, we can derive:

$$
\phi; \Phi; \Gamma \vdash t : [a < H\{0/b\}] \cdot \delta\{0/b\}, x : [a < P\{0/b\}] \cdot \sigma\{0/b\} \vdash^c_{[0/b]} \gamma
$$

Moreover, by assumption we have $||R^{-1} + \sum_{b < L} K||_\rho = 0$ and by definition of forest cardinality this implies that $||P\{0/b\}||_\rho = 0$.

By definition of interpretation, this implies that we have $\phi; x.t.[y := c], 0 \in [\sigma\{0/b\}]_\rho$. So, by induction hypothesis we have

$$(t, [y := c, x := (\text{fix } x.t, [y := c])], \sum_{a < H\{0/b\}} M\{0/b\} \cdot H\{0/b\} + K\{0/b\}) \in |\gamma|_\rho$$

and by antireduction we obtain:

$$(\text{fix } x.t, [y := c], \sum_{a < H\{0/b\}} M\{0/b\} \cdot H\{0/b\} + K\{0/b\}) \in |\gamma|_\rho$$

and by Lemma 5 since clearly

$$\left[ \sum_{a < H\{0/b\}} M\{0/b\} \cdot H\{0/b\} + K\{0/b\} \right] \leq \left[ N + K \right]$$

we have

$$(\text{fix } x.t, [y := c], N + K) \in |\gamma|_\rho$$

**Case $[K]_\rho = n + 1$**

**Subcase**

$$\phi; \Phi; \Gamma \vdash t : [a < I] \cdot \sigma \rightarrow \tau$$

$$\phi, a; \Phi, a < I; \Delta \vdash u : \sigma$$

$$\phi; \Phi \vdash \Sigma \subseteq \Gamma \uplus \sum_{a < I} \Delta$$

$$\phi; \Phi; \Sigma \vdash a + \sum_{a < I} K \vdash u : \tau$$

Without loss of generality and by definition of bounded sum we consider the case where $\Gamma = x : [b \in M] \cdot \gamma$, and $\Delta = x : [b \in H] \cdot \gamma\{M + b + \sum_{d < a} H(d/b)/b\}$ and $\Sigma = x : [b \in M + \sum_{a < I} H] \cdot \gamma\{0\}$. By the interpretation definition and by some manipulation of the index terms we have that $(c, \sum_{b < M} L + M) \in (b < M] \cdot \gamma\{0\}$ and for every $n \leq [I]_\rho$ we also have

$$(c, \sum_{b < M} L + H) \in (b < M] \cdot \gamma\{M + b + \sum_{d < a} H(d/b)/b\}_{b\in\{a+n\}}$$

for some $L$ such that $\sum_{b < M} L + M + \sum_{d < a} (\sum_{b < H} L + H) \geq N$.

So, by induction hypothesis we have $\phi; (t, [x := c], J + \sum_{b < M} L + M) \in [a < I] \cdot \sigma \rightarrow \tau\{0\}$. Also, by induction hypothesis and some transformation we have $\phi; (u, [x := c], J + \sum_{b < H} L + H + I) \in [a < I] \cdot \sigma\{0\}$. So by anti-reduction we have:

$$((tu, [x := c]), J + \sum_{b < M} L + \sum_{a < I} (K + \sum_{b < H} L + H)) \in [\tau\{0\}]$$
and since

\[ J + \sum_{b < M} L + M + (\sum_{a < I} (K + \sum_{b < H} L + H) + I) = J + \sum_{a < I} K + I + N \]

the conclusion follows.

**Subcase**

\[
\begin{align*}
\phi, b; \Phi, b &< L; \Gamma, x : [a < P] \cdot \sigma \vdash^\varepsilon K_t : \tau \\
\phi; \Phi &\vdash^\varepsilon \tau_{0/b} \subseteq \gamma \\
\phi, a, b; \Phi, a &< P, b < L \vdash^\varepsilon \tau_{\bigoplus_{b=0}^{b+1} a} [P + b + 1/b] \subseteq \sigma \\
\phi; \Phi &\vdash^\varepsilon \Sigma \subseteq \sum_{b < L} \Gamma \\
\phi; \Phi &\vdash^\varepsilon \bigoplus_{b=0}^{b+1} P \leq L, R \\
\phi; \Phi; \Sigma &\vdash^\varepsilon_{R+1} \sum_{n \leq L} K_{\text{fix}} x.t : \gamma - R
\end{align*}
\]

Without loss of generality and by definition of bounded sum we can consider the case where \( \Gamma = y : [a < H] \cdot \delta\{a + \sum_{d<b} H[d/a]/a\} \) and \( \Sigma = y : [a < \sum_{b<L} H] \cdot \delta \). Consider \( (c, N) \in \|a < \sum_{b<L} H\| \cdot \delta \). By the interpretation definition and by some manipulation of the index terms we have that for every \( n < \|L\|_\rho \):

\[
(\sum_{a < H[n/b]} M[n/b] + H[n/b]) \in [a < H] \cdot \delta\{a + \sum_{d<b} H[d/a]/a\}_{\rho[\text{fix}]} = N.
\]

for some \( M \) such that \( \sum_{b<L} (\sum_{a<H} M + H) \cong N \).

By using some manipulations of the indices, we can derive:

\[
\phi; \Phi; \Gamma_1, x : [a < P\{0/b\}] \cdot \sigma\{0/b\} \vdash^\varepsilon K_{\{0/b\}} t : \gamma
\]

Without loss of generality we can assume that \( \|P\{0/b\}\| > 0 \). The case where \( \|P\{0/b\}\| = 0 \) is similar to the base case. By further manipulation of the indices, we also have:

\[
\phi; \Phi, a < P\{0/b\}; \Sigma_1 \vdash^\varepsilon_{Q\{a/c\} \cong 1+\sum_{b<Q\{a/c\}} K\{U/b\}} \text{fix } x.t : \sigma\{0/b\}
\]

for \( Q = \bigoplus_{b=0}^{b+1} P + 1 + \bigoplus_{b=0}^{b+1} P \{b\} \) and \( U = 1 + b + \sum_{c<a} Q \) and \( \phi; \Phi, a < P\{0/b\} \vdash \Sigma_1 \subseteq \sum_{b<Q\{a/c\}} \tau\{U/b\} \). In particular, we can choose \( \Gamma_1 \) and \( \Sigma_1 \) such that:

\[
\phi; \Phi \vdash \Sigma \cong \sum_{a<P\{0/b\}} \Sigma_1 + \Gamma_1 \subseteq \sum_{a<P\{0/b\}} \sum_{b<\sum_{b<Q\{a/c\}}} \sum_{b<L} \tau\{U/b\} + \Gamma\{0/b\} \cong \sum_{b<L} \Gamma
\]

Since \( \|P\{0/b\}\| > 0 \) for every \( \rho' = \rho\{a := k\} \) with \( k < \|P\{0/b\}\| \) we have that

\[
\|Q\{a/c\} \cong 1+ \sum_{b<Q\{a/c\}} K\{U/b\}\| \rho' < \|R \cong 1+ \sum_{b<L} K\| \rho
\]

So by generalized induction hypothesis we have that

\[
(fix x.t, [y := c], J) \in |\gamma|_{\rho}
\]
for
\[ J = Q\{a/c\} + 1 + \sum_{b \in Q\{a/c\}} K\{U/b\} + \sum_{b \in Q\{a/c\}} (\sum_{a \in H\{U/b\}} M\{U/b\} + H\{U/b\}) \]
and so
\[(\text{fix} \; x.t.[y := c], \sum_{a < P\{0/b\}} J) \in \|a < [P\{0/b\}]\| \cdot \gamma |_\rho \]

So, by induction hypothesis we also have
\((t, [y := c], x := (\text{fix} \; x.t.[y := c]), K_2) \in | \gamma | |_\rho \)

where
\[ K_2 = K\{0/b\} + P\{0/b\} + \sum_{a < P\{0/b\}} \left( Q\{a/c\} + 1 + \sum_{b \in Q\{a/c\}} K\{U/b\} + \sum_{b \in Q\{a/c\}} (\sum_{a \in H\{U/b\}} M\{U/b\} + H\{U/b\}) \right) \]
\[ + \sum_{a < H\{0/b\}} M\{0/b\} + H\{0/b\} \]
\[ = R + 1 + \sum_{b < L} K + N \]

So, by antireduction we obtain
\[(\text{fix} \; x.t.[y := c], K_2) \in | \sigma | |_\rho \]

that is what we need to prove. \(\square\)

4 Quantitative reducibility candidates and \(d\ell\text{PCF}\) properties

In this section, we are interested in using the quantitative realizability model introduced in the previous section to prove properties about \(d\ell\text{PCF}\). In particular, we show how it can be used to give a new proof of the \(d\ell\text{PCF}\) intensional soundness theorem. Moreover, when a universal equational program is considered we can also show a correspondence between our model and the type system.

All along this section we will consider a fixed quantitative pole \(\perp\) defined as follows:
\[ \perp = \{ (C, n) \mid C \downarrow^m \land m \leq n \} \]

Checking that \(\perp\) is indeed a quantitative pole is easy since we take into account exactly the number of variable reduction steps. The choice of this quantitative pole ensures that we have some additional properties on \(\rho\)-behaviors. In particular, the following lemma shows that each behavior contains the weighted closure \((\mathfrak{X}, 0)\).
Lemma 7. For each $\rho$-behavior $X$, we have $(\mathfrak{X}, 0) \in X$.

Proof. If $(\xi, I) \in X^{\perp_{\rho}}$, then it is immediate that $(\mathfrak{X}, \xi) \downarrow^0$. Hence $(\mathfrak{X}, 0)^{\perp_{\rho}}(\xi, I)$ since $0 \leq [I]_{\rho}$. So, $(\mathfrak{X}, 0) \in X^{\perp_{\rho}} = X$. \hfill $\square$

The quantitative pole $\bot$ also permits to consider a natural quantitative extension of the usual notion of reducibility candidates.

Definition 7 (Quantitative reducibility candidates). The set of $\rho$-quantitative reducibility candidates, denoted by $\mathbb{QCR}_{\rho}$, is the set of all the $\rho$-behaviors $X$ such that $X \subseteq \{(\epsilon, 0)\}^{\perp_{\rho}}$.

The notion of quantitative reducibility candidate helps us to prove a first form of completeness for our model. We name here this first form external completeness in contrast to a second form of completeness that will be presented below and that is called internal completeness. We use the terminology completeness for this first theorem, since it is similar to the notion of completeness of phase semantics with respect to linear logic or of Kripke models with respect to intuitionistic logic for instance, but where termination is considered instead of typability.

Theorem 8 (External completeness). For every $\rho$ and all basic types $\sigma$ we have $[\sigma]_{\rho} \in \mathbb{QCR}_{\rho}$.

Proof. By induction on the structure of the basic type $\sigma$.

Case $\text{Nat}[I, J]$. It is easy to check that for each integer $n$ and each environment $\nu$, $(n, \nu, 0) \in \{(\epsilon, 0)\}^{\perp_{\rho}}$. Hence by monotonicity of the biorthogonality closure, we obtain $[\text{Nat}[I, J]]_{\rho} \subseteq \{(\epsilon, 0)\}^{\perp_{\rho}} = \{(\epsilon, 0)\}^{\perp_{\rho}}$.

Case $[a < I]\sigma \rightarrow \tau$. Suppose $(c, K) \in [(a < I]_{\sigma} \rightarrow \tau]_{\rho}$. Then because $(\mathfrak{X}, \epsilon, 0) \in [(a < I]_{\sigma}]_{\rho}$ and because $(\epsilon, 0) \in [\tau]_{\rho}$ (since $[\tau]_{\rho} \in \mathbb{QCR}_{\rho}$ by induction hypothesis), we obtain that $(c, \mathfrak{X}, \epsilon, [K]_{\rho}) \in \bot$. Using Lemma 3, we obtain $(c, \epsilon, [K]_{\rho}) \in \bot$. \hfill $\square$

The external completeness property stated above can be considered as a bounded-time termination property of realizers. However, this property alone is not sufficient to prove the intensional soundness result for $\mathsf{d/PCF}$.

Remark 3. Theorem 8 does not hold for any choice of pole. For instance, if we choose the pole $\bot = \{ (C, n) \mid C \text{ diverges} \}$, for each integer $n$ and $k \in \mathbb{N}$, the configuration $(\mathfrak{n}, \epsilon, k) \not\in \bot$. This shows that $[\text{Nat}[0, 0]]_{\rho} \not\in \mathbb{QCR}_{\rho}$.

To be able to prove the intensional soundness, we first need to prove an internal completeness result for the type $\text{Nat}[J, K]$. This property allows us to characterize the elements of $[\text{Nat}[J, K]]_{\rho}$. The proof will mainly use the fact that $\bot$ only contains safe configurations, and that stacks can discriminate two different natural numbers. It is similar to the internal completeness property of ludics [11, 25].

Theorem 9 (Internal completeness). Suppose $[I]_{\rho} \leq [K]_{\rho}$ and $\models I \downarrow$. If $(c, I) \in [\text{Nat}[J, K]]_{\rho}$ then $(c, \epsilon) \downarrow^m (n, \nu, \epsilon)$ with $m \leq [I]_{\rho}$ and $[J]_{\rho} \leq n \leq [K]_{\rho}$.
Proof. Let $\Omega$ be a PCF diverging term. In order to discriminate integers we will use the following particular families of stacks indexed by $k \in \mathbb{N}$:

$$\xi_k = s \cdot \underbrace{p \cdot p \ldots p}_{k \text{ times}} (\Omega, \emptyset, \epsilon) \cdot \epsilon \quad \theta_k = \underbrace{p \cdot p \ldots p}_{k \text{ times}} (\emptyset, \Omega, \epsilon) \cdot \epsilon$$

These families of stacks have the following properties:

- $\forall k \leq n, (\mathbf{n}, \epsilon, \xi_k) \sim_{k+1}^{\xi_n+1} \rho (\mathbf{n'}, \emptyset, (\Omega, \emptyset, \epsilon) \cdot \epsilon) \mapsto (\Omega, \emptyset, \epsilon)$,
- $\forall k > n, (\mathbf{n}, \epsilon, \xi_k)$ diverges,
- $\forall k \geq n, (\mathbf{n}, \epsilon, \theta_k) \sim_k (\emptyset, \Omega, \epsilon) \cdot \epsilon) \mapsto (\Omega, \emptyset, \epsilon)$,
- $\forall k < n, (\mathbf{n}, \epsilon, \theta_k)$ diverges.

Thanks to these properties we have:

$$\{ (\xi_k, 0) \mid k \leq \|J\| \} \cup \{ (\theta_k, 0) \mid \|K\| \leq k \}$$

$\subseteq \{ (\mathbf{n}, 0) \mid \|J\| \leq n \leq \|K\| \} \implies \|\text{Nat}[J, K]\|_{\rho}^{\epsilon}$

Now, suppose $(c, I) \in \|\text{Nat}[J, K]\|_{\rho}^{\epsilon}$. We clearly know that $(c, I) \in \{ (\xi_{\mathbf{n}}, 0), (\theta_{\mathbf{n}}, 0) \}^{\epsilon}$. But this says that there exists some integer $n \in \mathbb{N}$ such that $(c, I) \sim_{\mathbf{n}} (\emptyset, J, \epsilon)$ (by the safety property of the configurations in $\|J\|_{\rho}^{\epsilon}$) and by external completeness Theorem 8 that $m \leq \|J\|_{\rho}$.

Moreover, it tells us also that $\|J\| \leq n \leq \|K\|_{\rho}$ otherwise we would have a diverging computation, hence the result.

The internal completeness theorem above is the last ingredient we need to prove the intensional soundness.

**Theorem 10 (Intensional soundness).** Let $\emptyset; \emptyset t : \text{Nat}[J, K]$. Then, we have $\downarrow^m \mathbf{n}$ with $n \leq |t| \cdot (|\|J\|_{\rho}^{\epsilon} + 1)$ and $\|J\|_{\rho}^{\epsilon} \leq m \leq \|K\|_{\rho}^{\epsilon}$.

**Proof.** It follows directly by Theorem 7, by internal completeness 9 and by Lemma 2.

The type system of dfPCF has been conceived to be relatively complete with respect to the evaluation on Krivine’s machine. Relative completeness is obtained by considering a universal equational program $U$. Using relative completeness for programs and for functions respectively, we can show that on programs and functions realizability and typability coincide.

**Theorem 11 (Coincidence).** Let $\rho \vdash^U \phi; \Phi$. Then,

1. $\phi; \emptyset \vdash^U t : \text{Nat}[J, K] \iff ((t, \epsilon), I) \in \|\text{Nat}[J, K]\|_{\rho}^{\epsilon}$,
2. Moreover, if $\rho$ bounds the variable $a$, we have

   $\phi; a : \Phi \vdash^U t : [b < J] \text{Nat}[a] \rightarrow \text{Nat}[K] \iff ((t, \epsilon), I) \in \|b < J\| \text{Nat}[a] \rightarrow \text{Nat}[K]\|_{\rho}^{\epsilon}$

**Proof.** For both the points, the direction $\Rightarrow$ follows directly by Theorem 7. The direction $\Leftarrow$ follows instead by the internal completeness Theorem 9 and by Relative Completeness for programs (Theorem 4) and functions (Theorem 5), respectively.

The above theorem ensures that we can reason about the type system in an abstract way by using the realizability model.
5 Conclusions and Related works

Realizability techniques are nowadays a standard tool to reason about program behavior [24,6,5,15]. The gain in using such techniques is an approach to formal reasoning about programs that abstracts the language properties from the concrete syntax. An example is the semantical soundness proof we presented here. However, classical works on program behavior do not consider the quantitative aspects of programs.

Quantitative realizability models have been studied before in the context of linear logic. Hofmann and Scott in [13] have studied a realizability model for Bounded linear logic. This model has then been further revisited by Dal Lago and Hofmann in [10] and by Brunel in [7]. The main technical difference between our work and the previous ones is that our model, analogously to $d\ell$-PCF, is parametrized on an equational program and on an assignment. In this respect, our work can be understood not as a unique model but as a set of models that can be instantiated as needed. Furthermore, ours is also the first quantitative realizability model that can be used to reasoning about the full PCF language. It is interesting to note that the information given by $d\ell$PCF types are enough to reason about all the terminating recursive programs. This permits to avoid the use of extra-techniques like step-indexing [2] usually employed in this setting. An obvious limitation of our approach is that index terms, differently from step-indexing, cannot be used to reason about non-terminating programs.

Another motivation for our study is the possibility to internalize the notion of forcing in classical realizability models as shown by Krivine in [19] and Miquel in [22]. These works established connections between forcing conditions of logical principles and program transformations. Starting from these works, Brunel in [7] has shown that the quantitative part of a quantitative realizability model can be viewed as a forcing condition. Forcing can also be useful in an intuitionistic framework as shown in [16], where it can be used to generalize step-indexing. Since they also use forcing to account for term fixpoints, it would be interesting to explore the link between these frameworks.

References