Logical Methods in Computer Science
Volume 17, Issue 1, 2021, pp. 7:1–7:27
https://lmcs.episciences.org/

Submitted Dec. 17, 2018
Published Jan. 29, 2021

SOLVABILITY = TYPABILITY + INHABITATION

ANTONIO BUCCIARELLI [a], DELIA KESNER [b], AND SIMONA RONCHI DELLA ROCCA [c]

[a] Université de Paris, CNRS, IRIF, France
e-mail address: buccia@irif.fr

[b] Université de Paris, CNRS, IRIF and Institut Universitaire de France, France
e-mail address: kesner@irif.fr

[c] Dipartimento di Informatica, Università di Torino, Italy
e-mail address: ronchi@di.unito.it

Abstract. We extend the classical notion of solvability to a λ-calculus equipped with pattern matching. We prove that solvability can be characterized by means of *typability* and *inhabitation* in an *intersection type system* $\mathcal{P}$ based on *non-idempotent* types. We show first that the system $\mathcal{P}$ characterizes the set of terms having canonical form, *i.e.* that a term is typable if and only if it reduces to a canonical form. But the set of solvable terms is properly contained in the set of canonical forms. Thus, typability alone is not sufficient to characterize solvability, in contrast to the case for the λ-calculus. We then prove that typability, together with inhabitation, provides a full characterization of solvability, in the sense that a term is solvable if and only if it is typable and the types of all its arguments are inhabited. We complete the picture by providing an algorithm for the inhabitation problem of $\mathcal{P}$.

1. Introduction

In these last years there has been a growing interest in *pattern λ-calculi* [25, 20, 13, 21, 19, 24] which are used to model the pattern-matching primitives of functional programming languages (*e.g.* OCAML, ML, Haskell) and proof assistants (*e.g.* Coq, Isabelle). These calculi are extensions of the λ-calculus: abstractions are written as $\lambda\mathtt{p}.\mathtt{t}$, where $\mathtt{p}$ is a *pattern* specifying the expected structure of the argument. In this paper we restrict our attention to *pair* patterns, which are expressive enough to illustrate the challenging notion of solvability in the framework of pattern λ-calculi.

We define a calculus with *explicit pattern-matching* called $\Lambda_{\mathtt{p}}$. The use of explicit pattern-matching becomes very appropriate to implement different *evaluation strategies*, thus giving rise to different *programming languages* with pattern-matching [13, 14, 3]. In all of them, an application $(\lambda\mathtt{p}.\mathtt{t})\mathtt{u}$ reduces to $\mathtt{t}[\mathtt{p}/\mathtt{u}]$, where the constructor $[\mathtt{p}/\mathtt{u}]$ is an explicit matching, defined by means of suitable reduction rules, which are used to decide if the argument $\mathtt{u}$ matches the pattern $\mathtt{p}$. If the matching is possible, the evaluation proceeds

Key words and phrases: Pattern-calculus, Type-Assignement Systems, Non-idempotent Intersection Types, Solvability, Inhabitation.

LOGICAL METHODS IN COMPUTER SCIENCE DOI:10.23638/LMCS-17(1:7)2021

by computing a substitution which is applied to the body $\mathtt{t}$. Otherwise, two cases may arise: either a successful matching is not possible at all, and then the term $\mathtt{t}[\mathtt{p}/\mathtt{u}]$ reduces to a *failure*, denoted by the constant $\mathtt{fail}$, or pattern matching could potentially become possible after the application of some pertinent substitution to the argument $\mathtt{u}$, in which case the reduction is simply *blocked*. For example, reducing $(\lambda\langle \mathtt{z}_1, \mathtt{z}_2\rangle.\mathtt{z}_1)(\lambda \mathtt{y}.\mathtt{y})$ leads to a failure, while reducing $(\lambda\langle \mathtt{z}_1, \mathtt{z}_2\rangle.\mathtt{z}_1)\mathtt{y}$ leads to a blocking situation.

We aim to study *solvability* in the $\Lambda_{\mathtt{p}}$-calculus. Let us first recall this notion in the framework of the λ-calculus: a closed (*i.e.*, without free variables) λ-term $\mathtt{t}$ is solvable if there is $n \geq 0$ and there are terms $\mathtt{u}_1, ..., \mathtt{u}_n$ such that $\mathtt{t}\mathtt{u}_1...\mathtt{u}_n$ reduces to the identity function. Closed solvable terms represent meaningful programs: if $\mathtt{t}$ is closed and solvable, then $\mathtt{t}$ can produce any desired result when applied to a suitable sequence of arguments. The relation between solvability and meaningfulness is also evident in the semantics: it is sound to equate all unsolvable terms, as in Scott's original model D_∞ [26]. This notion can be easily extended to open terms, through the notion of *head context*, which does the job of both closing the term and then applying it to an appropriate sequence of arguments. Thus a λ-term $\mathtt{t}$ is solvable if there is a head context $\mathtt{H}$ such that, when $\mathtt{H}$ is filled by $\mathtt{t}$, then $\mathtt{H}[\mathtt{t}]$ is closed and reduces to the identity function.

In order to extend the notion of solvability to the $\Lambda_{\mathtt{p}}$-calculus, it is clear that pairs have to be taken into account. A relevant question is whether a pair should be considered as meaningful. At least two choices are possible: a *lazy* semantics considering any pair to be meaningful, or a *strict* one requiring both of its components to be meaningful. We chose a lazy approach, in fact in the operational semantics of $\Lambda_{\mathtt{p}}$ the constant $\mathtt{fail}$ is different from $\langle\mathtt{fail}, \mathtt{fail}\rangle$: if a term reduces to $\mathtt{fail}$ we do not have any information about its result, but if it reduces to $\langle\mathtt{fail}, \mathtt{fail}\rangle$ we know at least that it represents a pair. In fact, being a pair is already an observable property, which in particular is sufficient to unblock an explicit matching, independently from the solvability of its components. As a consequence, a term $\mathtt{t}$ is *defined* to be *solvable* iff there exists a head context $\mathtt{H}$ such that $\mathtt{H}[\mathtt{t}]$ is closed and reduces to a pair. Thus for example, the term $\langle\mathtt{t}, \mathtt{t}\rangle$ is always solvable, also when $\mathtt{t}$ is not solvable. Our notion of solvability turns out to be conservative with respect to the same notion for the λ-calculus (see Theorem 5.6).

In this paper we characterize solvability for the $\Lambda_{\mathtt{p}}$-calculus through two different and complementary notions related to a type assignment system with non-idempotent intersection types, called $\mathcal{P}$. The first one is *typability*, that gives the possibility to construct a typing derivation for a given term, and the second one is *inhabitation*, which gives the possibility to construct a term from a given typing. More precisely, we first supply a notion of *canonical form* such that reducing a term to some canonical form is a *necessary* but *not a sufficient* condition for being solvable. In fact, canonical forms may contain blocking explicit matchings, so that we need to guess whether or not there exists a substitution being able to *simultaneously* unblock all these blocked forms. Our type system $\mathcal{P}$ *characterizes* canonical forms: a term $\mathtt{t}$ has a canonical form if and only if it is typable in system $\mathcal{P}$ (Theorem 3.11). Types are of the shape $\mathtt{A}_1 \to \mathtt{A}_2 \to ... \to \mathtt{A}_n \to \sigma$, for $n \geq 0$, where $\mathtt{A}_i$ are multisets of types and σ is a type. The use of multisets to represent the non-idempotent intersection is standard, namely $[\sigma_1, ..., \sigma_m]$ is just a notation for $\sigma_1 \cap ... \cap \sigma_m$. By using the type system $\mathcal{P}$ we can supply the following *characterization* of solvability (Theorem 5.5): a closed term $\mathtt{t}$ in the $\Lambda_{\mathtt{p}}$-calculus is solvable if and only if $\mathtt{t}$ is typable in system $\mathcal{P}$, let say with a type of the shape $\mathtt{A}_1 \to \mathtt{A}_2 \to ... \to \mathtt{A}_n \to \sigma$ (where σ is a type derivable for a pair), and for all

$1 \leq i \leq n$ there is a term $\mathtt{t}_i$ inhabiting the type $\mathtt{A}_i$. In fact, if $\mathtt{u}_i$ inhabits the type $\mathtt{A}_i$, then $\mathtt{t}\mathtt{u}_1 ... \mathtt{u}_n$, resulting from plugging $\mathtt{t}$ into the head context $\Box\mathtt{u}_1 ... \mathtt{u}_n$, reduces to a pair. The extension of this notion to open terms is obtained by suitably adapting the notion of head context of the λ-calculus to our pattern calculus.

The property of being solvable in our calculus is clearly undecidable. More precisely, the property of having a canonical form is undecidable, since $\Lambda_{\mathtt{p}}$ extends the λ-calculus, the λ-terms having a $\Lambda_{\mathtt{p}}$-canonical form are exactly the solvable ones, and solvability of λ-terms is an undecidable property. But our characterization of solvability through the inhabitation property of $\mathcal{P}$ does not add a further level of undecidability: in fact we prove that inhabitation for system $\mathcal{P}$ is *decidable*, by designing a sound and complete inhabitation algorithm for it. The inhabitation algorithm presented here is a non trivial extension of the one given in [10, 11] for the λ-calculus, the difficulty of the extension being due to the explicit pattern matching.

Relation with λ-calculus. Let us recall the existing characterizations of solvability for the λ-calculus:

(1) : $\mathtt{H}[\mathtt{t}]$ reduces to the identity for an appropriate head context $\mathtt{H}$;
(2) : $\mathtt{t}$ has a head normal form;
(3) : $\mathtt{t}$ can be typed in a suitable intersection type system.

Statement **(1)** is the definition of solvability, Statement **(2)** (resp. **(3)**) is known as the *syntactical* (resp. *logical*) characterization of solvability. The syntactical characterization, *i.e.* **(2)** $\Leftrightarrow$ **(1)** has been proved in an untyped setting using the standardization theorem (see [6]). The logical characterization, *i.e.* **(3)** $\Leftrightarrow$ **(1)**, uses the syntactical one: it is performed by building an intersection type assignment system characterizing terms having head normal form (see for example [16]). Then the implication **(3)** $\Rightarrow$ **(2)** corresponds to the soundness of the type system (proved by means of a subject reduction property), while **(2)** $\Rightarrow$ **(3)** states its completeness (proved by subject expansion).

Traditional systems in the literature characterizing solvability for λ-calculus are for example [7, 22], where intersection is *idempotent*. Exactly the same results hold for *non-idempotent* intersection types, for example for the type system [15, 10], which is a restriction of $\mathcal{P}$ to λ-terms.

How does the particular case of the λ-calculus fit in the "solvability = typability + inhabitation" frame? We address this issue in the following digression. Let ι be a type which is peculiar to some subset of "solvable" terms, in the sense that any closed term of type ι reduces to a solvable term in that set (in the present work, such a subset contains all the pairs, in the case of the λ-calculus, it is the singleton containing only the identity). Then a type τ of the form $\mathtt{A}_1 \rightarrow \ldots \rightarrow \mathtt{A}_n \rightarrow \iota$ may be viewed as a certificate, establishing that, by applying a closed term $\mathtt{t} : \tau$ to a sequence of closed arguments $\mathtt{u}_i : \mathtt{A}_i$, one gets a term that reduces to a term in such a subset. This is summarized by the slogan "solvability = typability + inhabitation". In the case of the call-by-name λ-calculus, however, typability alone already guarantees solvability. The mismatch is only apparent, though: any closed, head normal term of the λ-calculus, *i.e.* any term of the shape $\lambda\mathtt{x}_1 \ldots \mathtt{x}_n.\mathtt{x}_j\mathtt{t}_1 \ldots \mathtt{t}_m$ $(n, m \geq 0)$, may be assigned a type of the form $\mathtt{A}_1 \rightarrow \ldots \rightarrow \mathtt{A}_n \rightarrow \iota$ where all the $\mathtt{A}_i$'s are empty except the one corresponding to the head variable $\mathtt{x}_j$, which is of the shape $[\underbrace{[\,] \rightarrow \ldots \rightarrow [\,]}_{m} \rightarrow \iota]$. The problems of finding inhabitants of the empty type and of $\underbrace{[\,] \rightarrow \ldots \rightarrow [\,]}_{m} \rightarrow \iota$ are both trivial.

Hence, "solvability = typability + inhabitation" does hold for the λ-calculus, too, but the "inhabitation" part is trivial in that particular case. This is due, of course, to the fact that the head normalizable terms of the λ-calculus coincide with both the solvable terms and the typable ones.

But in other settings, a term may be both typable and non solvable, the types of (some of) its arguments being non-inhabited (Theorem 5.1).

Related work. This work is an expanded and revised version of [12]. In particular:

- The reduction relation on $\Lambda_{\mathtt{p}}$-terms in this paper is smaller. In particular, the new reduction system uses *reduction at a distance* [1], implemented through the notion of list contexts.
- Accordingly, the type system $\mathcal{P}$ in this paper and the corresponding inhabitation algorithm are much simpler. In particular, the use of idempotent/persistent information on the structure of patterns is no more needed.

Non-idempotent intersection types are also used in [9] to derive strong normalization of a call-by-name calculus with constructors, pattern matching and fixpoints. A similar result can be found in [5], where the completeness proof of the (strong) call-by-need strategy in [4] is extended to the case of constructors. Based on [12], the type assignment system $\mathcal{P}$ was developed in [2] in order to supply a quantitative analysis (upper bounds and exact measures) for head reduction.

Organization of the paper. Section 2 introduces the pattern calculus and its main properties. Section 3 presents the type system and proves a characterization of terms having canonical forms by means of typability. Section 4 presents a sound and complete algorithm for the inhabitation problem associated with our typing system. Section 5 shows a complete characterization of solvability using the inhabitation result and the typability notion. Section 6 concludes by discussing some future work.

2. The Pair Pattern Calculus

We now introduce the $\Lambda_{\mathtt{p}}$-calculus, a generalization of the λ-calculus where abstraction is extended to *patterns* and terms to *pairs*. Pattern matching is specified by means of an *explicit* operation. Reduction is performed only if the argument matches the abstracted pattern.

Terms and **contexts** of the $\Lambda_{\mathtt{p}}$-calculus are defined by means of the following grammars:

$$
\begin{array}{llcl}
(\textbf{Patterns}) & \mathtt{p},\mathtt{q} & ::= & \mathtt{x} \mid \langle \mathtt{p},\mathtt{q}\rangle \\
(\textbf{Terms}) & \mathtt{t},\mathtt{u},\mathtt{v} & ::= & \mathtt{x} \mid \lambda \mathtt{p}.\mathtt{t} \mid \langle \mathtt{t},\mathtt{u}\rangle \mid \mathtt{tu} \mid \mathtt{t}[\mathtt{p}/\mathtt{u}] \mid \mathtt{fail} \\
(\textbf{List Contexts}) & \mathtt{L} & ::= & \Box \mid \mathtt{L}[\mathtt{p}/\mathtt{t}] \\
(\textbf{Term Contexts}) & \mathtt{C} & ::= & \Box \mid \lambda \mathtt{p}.\mathtt{C} \mid \langle \mathtt{C},\mathtt{t}\rangle \mid \langle \mathtt{t},\mathtt{C}\rangle \mid \mathtt{Ct} \mid \mathtt{tC} \mid \mathtt{C}[\mathtt{p}/\mathtt{t}] \mid \mathtt{t}[\mathtt{p}/\mathtt{C}] \\
(\textbf{Head Contexts}) & \mathtt{H} & ::= & \Box \mid \lambda \mathtt{p}.\mathtt{H} \mid \mathtt{Ht} \mid \mathtt{H}[\mathtt{p}/\mathtt{t}]
\end{array}
$$

where $\mathtt{x},\mathtt{y},\mathtt{z}$ range over a countable set of variables, and every pattern $\mathtt{p}$ is *linear*, *i.e.* every variable appears at most once in $\mathtt{p}$. We denote by $\mathtt{I}$ the **identity function** $\lambda\mathtt{x}.\mathtt{x}$ and by δ the auto applicative function $\lambda\mathtt{x}.\mathtt{xx}$. As usual we use the abbreviation $\lambda\mathtt{p}_1 \ldots \mathtt{p}_n.\mathtt{t}_1 \ldots \mathtt{t}_m$ for $\lambda\mathtt{p}_1(\ldots(\lambda\mathtt{p}_n.((\mathtt{t}_1\mathtt{t}_2)\ldots\mathtt{t}_m))\ldots)$, $n \geq 0$, $m \geq 1$. Remark that every λ-term is in particular a $\Lambda_{\mathtt{p}}$-term.

The operator $[\mathtt{p}/\mathtt{t}]$ is called an **explicit matching**. The constant $\mathtt{fail}$ denotes the failure of the matching operation. The sets of **free** and **bound** variables of a term $\mathtt{t}$, denoted respectively by $\mathtt{fv}(\mathtt{t})$ and $\mathtt{bv}(\mathtt{t})$, are defined as expected, in particular $\mathtt{fv}(\lambda\mathtt{p}.\mathtt{t}) := \mathtt{fv}(\mathtt{t}) \setminus \mathtt{fv}(\mathtt{p})$ and $\mathtt{fv}(\mathtt{t}[\mathtt{p}/\mathtt{u}]) := (\mathtt{fv}(\mathtt{t}) \setminus \mathtt{fv}(\mathtt{p})) \cup \mathtt{fv}(\mathtt{u})$. A term $\mathtt{t}$ is **closed** if $\mathtt{fv}(\mathtt{t}) = \emptyset$. We write $\mathtt{p}\#\mathtt{q}$ iff $\mathtt{fv}(\mathtt{p}) \cap \mathtt{fv}(\mathtt{q}) = \emptyset$. As usual, terms are considered modulo α-conversion. Given a term (resp. list) context $\mathtt{C}$ (resp. $\mathtt{L}$) and a term $\mathtt{t}$, $\mathtt{C}[\mathtt{t}]$ (resp. $\mathtt{L}[\mathtt{t}]$) denotes the term obtained by replacing the unique occurrence of $\Box$ in $\mathtt{C}$ (resp. $\mathtt{L}$) by $\mathtt{t}$, thus possibly capturing some free variables of $\mathtt{t}$. In this paper, an occurrence of a subterm $\mathtt{u}$ in a term $\mathtt{t}$ is understood as the unique context $\mathtt{C}$ such that $\mathtt{t} = \mathtt{C}[\mathtt{u}]$.

The **reduction relation** of the $\Lambda_{\mathtt{p}}$-calculus, denoted by $\to$, is the $\mathtt{C}$-contextual closure of the following rewriting rules:

$$\begin{array}{lll}
(\mathtt{dB}) & \mathtt{L}[\lambda\mathtt{p}.\mathtt{t}]\mathtt{u} & \mapsto \mathtt{L}[\mathtt{t}[\mathtt{p}/\mathtt{u}]] \\
(\mathtt{subs}) & \mathtt{t}[\mathtt{x}/\mathtt{u}] & \mapsto \mathtt{t}\{\mathtt{x}/\mathtt{u}\} \\
(\mathtt{match}_s) & \mathtt{t}[\langle\mathtt{p}_1,\mathtt{p}_2\rangle/\mathtt{L}[\langle\mathtt{u}_1,\mathtt{u}_2\rangle]] & \mapsto \mathtt{L}[\mathtt{t}[\mathtt{p}_1/\mathtt{u}_1][\mathtt{p}_2/\mathtt{u}_2]] \\
(\mathtt{match}_f) & \mathtt{t}[\langle\mathtt{p}_1,\mathtt{p}_2\rangle/\mathtt{L}[\lambda\mathtt{q}.\mathtt{u}]] & \mapsto \mathtt{fail} \\
(\mathtt{app}_f) & \mathtt{L}[\langle\mathtt{t},\mathtt{u}\rangle]\mathtt{v} & \mapsto \mathtt{fail} \\
(\mathtt{rem}_f) & \mathtt{t}[\langle\mathtt{p}_1,\mathtt{p}_2\rangle/\mathtt{fail}] & \mapsto \mathtt{fail} \\
(\mathtt{lem}_f) & \mathtt{L}[\mathtt{fail}] & \mapsto \mathtt{fail} \\
(\mathtt{la}_f) & \mathtt{fail}\ \mathtt{t} & \mapsto \mathtt{fail} \\
(\mathtt{abs}_f) & \lambda\mathtt{p}.\mathtt{fail} & \mapsto \mathtt{fail}
\end{array}$$

where $\mathtt{t}\{\mathtt{x}/\mathtt{u}\}$ denotes the substitution of all the free occurrences of $\mathtt{x}$ in $\mathtt{t}$ by $\mathtt{u}$ and $\mathtt{L} \neq \Box$ in rule $(\mathtt{lem}_f)$. By α -conversion, and without loss of generality, no reduction rule captures free variables, so that in particular $\mathtt{bv}(\mathtt{L}) \cap \mathtt{fv}(\mathtt{u}) = \emptyset$ holds for rule $(\mathtt{dB})$ and $\mathtt{bv}(\mathtt{L}) \cap \mathtt{fv}(\mathtt{t}) = \emptyset$ holds for rule $(\mathtt{match}_s)$. The rule $(\mathtt{dB})$ triggers the pattern operation while rule $(\mathtt{subs})$ performs substitution, rules $(\mathtt{match}_s)$ and $(\mathtt{match}_f)$ implement (successful or unsuccessful) pattern matching. Rule $(\mathtt{app}_f)$ prevents bad applications and rules $(\mathtt{rem}_f)$, $(\mathtt{lem}_f)$, $(\mathtt{la}_f)$ and $(\mathtt{abs}_f)$ deal with propagation of failure in right/left explicit matchings, left applications and abstractions, respectively. A **redex** is a term having the form of the left-hand side of some rewriting rule $\mapsto$. The reflexive and transitive closure of $\to$ is written $\to^*$.

Lemma 2.1. *The reduction relation $\to$ is confluent.*

Proof. The proof is given in the next subsection. ☐

Normal forms $\mathcal{N}$ are terms without occurrences of redexes; they are formally defined by the following grammars:

$$\begin{array}{lll}
\mathcal{N} & ::= & \mathtt{fail} \mid \mathcal{O} \\
\mathcal{O} & ::= & \mathcal{M} \mid \lambda\mathtt{p}.\mathcal{O} \mid \langle\mathcal{N},\mathcal{N}\rangle \mid \mathcal{O}[\langle\mathtt{p},\mathtt{q}\rangle/\mathcal{M}] \\
\mathcal{M} & ::= & \mathtt{x} \mid \mathcal{M}\mathcal{O} \mid \mathcal{M}[\langle\mathtt{p},\mathtt{q}\rangle/\mathcal{M}]
\end{array}$$

Lemma 2.2. *A term $\mathtt{t}$ is an $\mathcal{N}$-normal form if and only if $\mathtt{t}$ is a $\to$-normal form, i.e. if no rewriting rule is applicable to any subterm of $\mathtt{t}$.*

We define a term to be **normalizing** if it reduces to a normal form.

Let us notice that in a language like $\Lambda_{\mathtt{p}}$, where there is an explicit notion of failure, normalizing terms are not interesting from a computation point of view, since $\mathtt{fail}$ is a normal form, but cannot be considered as the result of a computation. If we want to formalize the notion of programs yielding a result, terms reducing to $\mathtt{fail}$ cannot be taken into consideration. Remark however that $\langle\mathtt{fail},\mathtt{fail}\rangle$ is not operationally equivalent to $\mathtt{fail}$, according to the idea that a pair can always be observed, and so it can be considered as a result of a computation. This suggests a notion of reduction which is **lazy w.r.t. pairs**, *i.e.* that never reduces inside pairs. Indeed:

Definition 2.3. A term $\mathtt{t}$ is **solvable** if there exists a head context $\mathtt{H}$ such that $\mathtt{H}[\mathtt{t}]$ is closed and $\mathtt{H}[\mathtt{t}] \to^* \langle\mathtt{u},\mathtt{v}\rangle$, for some terms $\mathtt{u}$ and $\mathtt{v}$.

Therefore, a syntactical class of terms which is particularly interesting from an operational point of view is that of canonical forms. **Canonical forms** $\mathcal{J}$ (resp. **pure canonical forms** $\mathcal{J}'$) can be formalized by the following grammar:

$$\begin{array}{ll} \mathcal{J} ::= \lambda\mathtt{p}.\mathcal{J} \mid \langle\mathtt{t},\mathtt{t}\rangle \mid \mathcal{K} \mid \mathcal{J}[\langle\mathtt{p},\mathtt{q}\rangle/\mathcal{K}] & \mathcal{J}' ::= \lambda\mathtt{p}.\mathcal{J}' \mid \langle\mathtt{t},\mathtt{t}\rangle \mid \mathcal{K}' \mid \mathcal{J}'[\langle\mathtt{p},\mathtt{q}\rangle/\mathcal{K}'] \\ \mathcal{K} ::= \mathtt{x} \mid \mathcal{K}\mathtt{t} \mid \mathcal{K}[\langle\mathtt{p},\mathtt{q}\rangle/\mathcal{K}] & \mathcal{K}' ::= \mathtt{x} \mid \mathcal{K}'\mathtt{t} \end{array}$$

where the notion of pure canonical form, *i.e.* of canonical form without nested matchings, is a technical notion that will be useful in the sequel. A term $\mathtt{t}$ **is in canonical form** (or it is **canonical**), written $\mathtt{cf}$, if it is generated by $\mathcal{J}$, and it **has a canonical form** if it reduces to a term in $\mathtt{cf}$. Note that $\mathcal{K}$-canonical forms cannot be closed. Also, remark that the $\mathtt{cf}$ of a term is not unique, *e.g.* both $\langle\mathtt{I},\mathtt{I}\ \mathtt{I}\rangle$ and $\langle\mathtt{I},\mathtt{I}\rangle$ are $\mathtt{cf}$s of $(\lambda\mathtt{xy}.\langle\mathtt{x},\mathtt{y}\rangle)\ \mathtt{I}\ (\mathtt{I}\ \mathtt{I})$. It is worth noticing that $\mathcal{N}\cap\mathcal{J}\neq\emptyset$ but neither $\mathcal{N}\subset\mathcal{J}$ nor $\mathcal{J}\subset\mathcal{N}$. Latter, we will prove that solvable terms are strictly contained in the canonical ones.

Example 2.4.

- The term $\langle\mathtt{fail},\mathtt{fail}\rangle$ is both in normal and canonical form.
- The term $\mathtt{fail}$ is in normal form, but not in canonical form.
- The term $\langle\delta\delta,\delta\delta\rangle$ is in canonical form, but not in normal form.
- The term $\lambda\langle\mathtt{x},\mathtt{y}\rangle.\mathtt{I}[\langle\mathtt{z}_1,\mathtt{z}_2\rangle/\mathtt{y}\mathtt{I}[\langle\mathtt{y}_1,\mathtt{y}_2\rangle/\mathtt{z}]]$ is in canonical form, but not in pure canonical form.
- The term $\lambda\langle\mathtt{x},\mathtt{y}\rangle.\mathtt{I}[\langle\mathtt{z}_1,\mathtt{z}_2\rangle/\mathtt{y}\mathtt{I}]$ is in pure canonical form.

We end this section by stating a lemma about $(\mathtt{subs})$-reduction that will be useful in next Section.

Lemma 2.5. *Every infinite $\to$-reduction sequence contains an infinite number of $(\mathtt{subs})$-reduction steps.*

Proof. It is sufficient to show that the reduction system without the rule $(\mathtt{subs})$, that we call $\mathcal{A}_0$, is terminating. Indeed, remark that $\mathtt{t}\to_{\mathcal{A}_0}\mathtt{t}'$ implies $\nu(\mathtt{t})>\nu(\mathtt{t}')$, where $\nu(\mathtt{t})$ is a pair whose first component is the number of applications of the form $\mathtt{L}[\mathtt{u}_1]\mathtt{u}_2$ in $\mathtt{t}$ (rules $\mathtt{dB},\mathtt{la}_f,\mathtt{app}_f$), and whose second component is the sum of the sizes of patterns in $\mathtt{t}$ (rules $\mathtt{match}_s,\mathtt{match}_f,\mathtt{rem}_f,\mathtt{abs}_f,\mathtt{lem}_f$). These pairs are ordered lexicographically. □

2.1. **The Confluence Proof.** In order to show confluence of our reduction system $\to$ we first simplify the system by erasing just one rule s_0 in such a way that confluence of $\to$ holds if confluence of $\to$ deprived from $\to_{s_0}$ holds. This last statement is proved by applying the decreasing diagram technique [28]. We just change the name/order of the rules to make easier the application of the decreasing technique.

$$\begin{array}{llll}
(s_0) & \mathtt{t}[\mathtt{x}/\mathtt{u}] & \mapsto & \mathtt{t}\{\mathtt{x}/\mathtt{u}\} & \\
(s_1) & \mathtt{L}[\mathtt{fail}] & \mapsto & \mathtt{fail} & (\mathtt{L}\text{ non-empty}) \\
(s_2) & \mathtt{fail}\ \mathtt{t} & \mapsto & \mathtt{fail} & \\
(s_3) & \lambda\mathtt{p}.\mathtt{fail} & \mapsto & \mathtt{fail} & \\
(s_4) & \mathtt{L}[\langle\mathtt{t},\mathtt{u}\rangle]\mathtt{v} & \mapsto & \mathtt{fail} & \\
(s_5) & \mathtt{t}[\langle\mathtt{p}_1,\mathtt{p}_2\rangle/\mathtt{fail}] & \mapsto & \mathtt{fail} & \\
(s_6) & \mathtt{t}[\langle\mathtt{p}_1,\mathtt{p}_2\rangle/\mathtt{L}[\lambda\mathtt{q}.\mathtt{u}]] & \mapsto & \mathtt{fail} & \\
(s_7) & \mathtt{t}[\langle\mathtt{p}_1,\mathtt{p}_2\rangle/\mathtt{L}[\langle\mathtt{u}_1,\mathtt{u}_2\rangle]] & \mapsto & \mathtt{L}[\mathtt{t}[\mathtt{p}_1/\mathtt{u}_1][\mathtt{p}_2/\mathtt{u}_2]] & \\
(s_8) & \mathtt{L}[\lambda\mathtt{p}.\mathtt{t}]\mathtt{u} & \mapsto & \mathtt{L}[\mathtt{t}[\mathtt{p}/\mathtt{u}]] &
\end{array}$$

We define $\mathcal{A}_0 :=\to \backslash s_0$. We write $\mathtt{t} \overset{=}{\to}_{s_0} \mathtt{t}'$ iff $\mathtt{t} \to_{s_0} \mathtt{t}'$ or $\mathtt{t} = \mathtt{t}'$.

Lemma 2.6. *For all* $\mathtt{t}_0, \mathtt{t}_1, \mathtt{t}_2$*, if* $\mathtt{t}_0 \to_{s_0} \mathtt{t}_1$ *and* $\mathtt{t}_0 \to_{\mathcal{A}_0} \mathtt{t}_2$*, then there exists* $\mathtt{t}_3$ *s.t.* $\mathtt{t}_1 \to^*_{\mathcal{A}_0} \mathtt{t}_3$ *and* $\mathtt{t}_2 \overset{=}{\to}_{s_0} \mathtt{t}_3$*.*

Proof. By induction on $\mathtt{t}_0 \to_{s_0} \mathtt{t}_1$, we only show the most significant cases:

$$\begin{array}{ccc}
\mathtt{t}[\mathtt{x}/\mathtt{u}] & \to_{s_0} & \mathtt{t}\{\mathtt{x}/\mathtt{u}\} \\
\mathcal{A}_0 \downarrow & & \mathcal{A}_0 \downarrow \\
\mathtt{t}'[\mathtt{x}/\mathtt{u}] & \to_{s_0} & \mathtt{t}'\{\mathtt{x}/\mathtt{u}\}
\end{array}
\qquad\qquad
\begin{array}{ccc}
\mathtt{t}[\mathtt{x}/\mathtt{u}] & \to_{s_0} & \mathtt{t}\{\mathtt{x}/\mathtt{u}\} \\
\mathcal{A}_0 \downarrow & & \mathcal{A}_0 \downarrow_* \\
\mathtt{t}[\mathtt{x}/\mathtt{u}'] & \to_{s_0} & \mathtt{t}\{\mathtt{x}/\mathtt{u}'\}
\end{array}$$

$$\begin{array}{ccc}
\mathtt{L}_2[\mathtt{L}_1[\mathtt{fail}][\mathtt{x}/\mathtt{u}]] & \to_{s_0} & \mathtt{L}_2[\mathtt{L}_1[\mathtt{fail}]\{\mathtt{x}/\mathtt{u}\}] \\
s_1 \downarrow & & s_1 \downarrow_= \\
\mathtt{fail} & = & \mathtt{fail}
\end{array}$$

∎

The following lemma can be found for example in [6].

Lemma 2.7. *Let* $\to_{\mathcal{R}_1}$ *and* $\to_{\mathcal{R}_2}$ *be two reduction relations. Suppose for any* $\mathtt{t}_0, \mathtt{t}_1, \mathtt{t}_2$ *such that* $\mathtt{t}_0 \to_{\mathcal{R}_2} \mathtt{t}_1$ *and* $\mathtt{t}_0 \to_{\mathcal{R}_1} \mathtt{t}_2$*, there exists* $\mathtt{t}_3$ *verifying* $\mathtt{t}_1 \to^*_{\mathcal{R}_1} \mathtt{t}_3$ *and* $\mathtt{t}_2 \overset{=}{\to}_{\mathcal{R}_2} \mathtt{t}_3$*. Then* $\to_{\mathcal{R}_2}$ *and* $\to_{\mathcal{R}_1}$ **commute***, i.e.* $\forall \mathtt{t}_0, \mathtt{t}_1, \mathtt{t}_2$ *if* $\mathtt{t}_0 \to^*_{\mathcal{R}_2} \mathtt{t}_1$ *and* $\mathtt{t}_0 \to^*_{\mathcal{R}_1} \mathtt{t}_2$*,* $\exists \mathtt{t}_3$ *s.t.* $\mathtt{t}_1 \to^*_{\mathcal{R}_1} \mathtt{t}_3$ *and* $\mathtt{t}_2 \to^*_{\mathcal{R}_2} \mathtt{t}_3$*.*

By Lemma 2.7 and 2.6 we obtain:

Corollary 2.8. *The reduction relations* $\to_{\mathcal{A}_0}$ *and* $\to_{s_0}$ *commute.*

Lemma 2.9. *The reduction relation* $\to_{\mathcal{A}_0}$ *is confluent.*

Proof. We use the decreasing diagram technique [28]. For that, we first order the reduction rules of the system $\mathcal{A}_0$ by letting $s_i < s_j$ iff $i < j$. We write $\mathtt{t} \to_i \mathtt{u}$ if $\mathtt{t} \to_{s_i} \mathtt{u}$. Given a set $\mathcal{I}$ of natural numbers we write $\mathtt{t} \to^*_{\triangledown\mathcal{I}} \mathtt{u}$ if every $\to_j$-reduction step in the sequence $\mathtt{t} \to^*_{\triangledown\mathcal{I}} \mathtt{u}$ verifies $j < \mathcal{I}$, *i.e.* if for every $\to_j$-reduction step in the sequence $\exists i \in \mathcal{I}$ such that $j < i$. The system $\to_{\mathcal{A}_0}$ is said to be decreasing iff for any $\mathtt{t}_0, \mathtt{t}_1, \mathtt{t}_2$ such that $\mathtt{t}_0 \to_l \mathtt{t}_1$ and $\mathtt{t}_0 \to_m \mathtt{t}_2$, there exists $\mathtt{t}_3$ such that $\mathtt{t}_1 \to^*_{\triangledown\{l\}} \overset{=}{\to}_m \to^*_{\triangledown\{l,m\}} \mathtt{t}_3$ and $\mathtt{t}_2 \to^*_{\triangledown\{m\}} \overset{=}{\to}_l \to^*_{\triangledown\{l,m\}} \mathtt{t}_3$, where $\mathtt{t} \overset{=}{\to}_l \mathtt{t}'$ means $\mathtt{t} \to_l \mathtt{t}'$ or $\mathtt{t} = \mathtt{t}'$.

We now show that the system $\to_{\mathcal{A}_0}$ is decreasing. As a matter of notation, we write for example $\mathtt{t} \to_{3=,5} \mathtt{t}'$ to denote a rewriting sequence of length 2 or 1, composed respectively by a $\to_3$-step followed by a $\to_5$-step or by a single $\to_5$-step.

- We consider the cases $\mathtt{t}_0 \to_1 \mathtt{t}_1$ and $\mathtt{t}_0 \to_i \mathtt{t}_2$ ($i = 1 \ldots 8$). We only show the interesting ones.

$$\begin{array}{ccc}
\mathtt{L}_2[\mathtt{L}_1[\mathtt{fail}][\langle \mathtt{p}_1, \mathtt{p}_2\rangle/\mathtt{fail}]] & \to_1 & \mathtt{fail} \\
{}_5\downarrow & & = \\
\mathtt{L}_2[\mathtt{fail}] & \overset{=}{\to}_1 & \mathtt{fail}
\end{array}$$

$$\begin{array}{ccc}
\mathtt{L}_2[\mathtt{L}_1[\mathtt{fail}][\langle \mathtt{p}_1, \mathtt{p}_2\rangle/\mathtt{L}[\langle \mathtt{u}_1, \mathtt{u}_2\rangle]]] & \to_1 & \mathtt{fail} \\
{}_7\downarrow & & = \\
\mathtt{L}_2[\mathtt{L}[\mathtt{L}_1[\mathtt{fail}][\mathtt{p}_1/\mathtt{u}_1][\mathtt{p}_2/\mathtt{u}_2]]] & \overset{=}{\to}_1 & \mathtt{fail}
\end{array}$$

All of them are decreasing diagrams as required.
- The cases $\mathtt{t}_0 \to_2 \mathtt{t}_1$ and $\mathtt{t}_0 \to_i \mathtt{t}_2$ ($i = 2 \ldots 8$) are straightforward.
- The interesting case $\mathtt{t}_0 \to_3 \mathtt{t}_1$ and $\mathtt{t}_0 \to_i \mathtt{t}_2$ ($i = 3 \ldots 8$) is the following.

$$\begin{array}{ccc}
\mathtt{L}[\lambda \mathtt{p}.\mathtt{fail}]\mathtt{u} & \to_3 & \mathtt{L}[\mathtt{fail}]\mathtt{u} \\
{}_8\downarrow & & {}_{1^=,2}\downarrow \\
\mathtt{L}[\mathtt{fail}[\mathtt{p}/\mathtt{u}]] & \to_1 & \mathtt{fail}
\end{array}$$

- The interesting cases $\mathtt{t}_0 \to_4 \mathtt{t}_1$ and $\mathtt{t}_0 \to_i \mathtt{t}_2$ ($i = 4 \ldots 8$) are straightforward.
- The interesting cases $\mathtt{t}_0 \to_5 \mathtt{t}_1$ and $\mathtt{t}_0 \to_i \mathtt{t}_2$ ($i = 5 \ldots 8$) are the following.

$$\begin{array}{ccc}
\mathtt{t}[\langle \mathtt{p}_1, \mathtt{p}_2\rangle/\mathtt{L}_1[\mathtt{L}_2[\langle \mathtt{u}_1, \mathtt{u}_2\rangle][\langle \mathtt{q}_1, \mathtt{q}_2\rangle/\mathtt{fail}]]] & \to_5 & \mathtt{t}[\langle \mathtt{p}_1, \mathtt{p}_2\rangle/\mathtt{L}_1[\mathtt{fail}]] \\
{}_7\downarrow & & {}_{1^=,5}\downarrow \\
\mathtt{L}_1[\mathtt{L}_2[\mathtt{t}[\mathtt{p}_1/\mathtt{u}_1][\mathtt{p}_2/\mathtt{u}_2]][\langle \mathtt{q}_1, \mathtt{q}_2\rangle/\mathtt{fail}]] & \to_5 & \mathtt{L}_1[\mathtt{fail}]
\end{array}$$

$$\begin{array}{ccc}
\mathtt{L}_1[\mathtt{L}_2[\lambda \mathtt{p}.\mathtt{t}][\langle \mathtt{q}_1, \mathtt{q}_2\rangle/\mathtt{fail}]]\mathtt{u} & \to_5 & \mathtt{L}_1[\mathtt{fail}]\mathtt{u} \\
{}_8\downarrow & & {}_{1^=,2}\downarrow \\
\mathtt{L}_1[\mathtt{L}_2[\mathtt{t}[\mathtt{p}/\mathtt{u}]][\langle \mathtt{q}_1, \mathtt{q}_2\rangle/\mathtt{fail}]] & \to_{5,1^=} & \mathtt{fail}
\end{array}$$

- The cases for $\mathtt{t}_0 \to_6 \mathtt{t}_1$ and $\mathtt{t}_0 \to_{s_i} \mathtt{t}_2$ ($i = 6 \ldots 8$) are similar.
- The cases for $\mathtt{t}_0 \to_7 \mathtt{t}_1$ and $\mathtt{t}_0 \to_{s_i} \mathtt{t}_2$ ($i = 7 \ldots 8$) have the following reduction scheme:

$$\begin{array}{ccc}
\cdot & \to_7 & \cdot \\
{}_7\downarrow & & {}_7\downarrow \\
\cdot & \to_7 & \cdot
\end{array}
\qquad\qquad
\begin{array}{ccc}
\cdot & \to_7 & \cdot \\
{}_8\downarrow & & {}_8\downarrow \\
\cdot & \to_7 & \cdot
\end{array}$$

- There is no other case. □

Lemma 2.10 (Hindley-Rosen). *Let $\to_{\mathcal{R}_1}$ and $\to_{\mathcal{R}_2}$ be two confluent reduction relations which commute. Then $\to_{\mathcal{R}_1 \cup \mathcal{R}_2}$ is confluent.*

Lemma 2.1. The reduction relation $\to$ is confluent.

Proof. Since $\to_{s_0}$ is trivially confluent, $\to_{\mathcal{A}_0}$ is confluent by Lemma 2.9 and $\to_{\mathcal{A}_0}$ and $\to_{s_0}$ commute by Corollary 2.8, then $\to_{\mathcal{A}_0} \cup \to_{s_0} = \to$ turns out to be confluent by Lemma 2.10, which concludes the proof. □

3. The Type System $\mathcal{P}$

In this section we present a type system for the $\Lambda_{\mathtt{p}}$-calculus, and we show that it characterizes terms having canonical form, *i.e.* that a term $\mathtt{t}$ is typable if and only if $\mathtt{t}$ has canonical form. **The set $\mathcal{T}$ of types** is generated by the following grammar:

$$\begin{array}{llll} \text{(Types)} & \sigma, \tau & ::= & \alpha \mid \pi \mid \mathtt{A} \to \sigma \\ \text{(Product Types)} & \pi & ::= & \times(\mathtt{A}, \mathtt{B}) \\ \text{(Multiset Types)} & \mathtt{A}, \mathtt{B} & ::= & [\sigma_k]_{k \in K} \end{array}$$

where α ranges over a countable set of constants, K is a (possibly empty) finite set of indices, and a multiset is an unordered list of (not necessarily different) elements. The arrow type constructor $\to$ is right associative.

Typing environments, written Γ, Δ, Λ, are functions from variables to multiset types, assigning the empty multiset to almost all the variables. The **domain** of Γ, written $\mathtt{dom}(\Gamma)$, is the set of variables whose image is different from $[\,]$. We may write $\Gamma\#\Delta$ iff $\mathtt{dom}(\Gamma) \cap \mathtt{dom}(\Delta) = \emptyset$.

Notation 3.1. Sometimes we will use symbols μ, ν to range over the union of types and multiset types. We abbreviate by the constant o the product type $\times([\,],[\,])$. We write $\sqcup$ to denote multiset union and $\sqsubseteq$ multiset inclusion; these operations take multiplicities into account. Moreover, abusing the notation we will use $\in$ to denote both set and multiset membership.

Given typing environments $\{\Gamma_i\}_{i \in I}$, we write $+_{i \in I}\Gamma_i$ for the environment which maps $\mathtt{x}$ to $\sqcup_{i \in I}\Gamma_i(\mathtt{x})$. If $I = \emptyset$, the resulting environment is the one having an empty domain. Note that $\Gamma + \Delta$ and $\Gamma +_{i \in I} \Delta_i$ are just particular cases of the previous general definition. When $\Gamma\#\Delta$ we may write $\Gamma; \Delta$ instead of $\Gamma + \Delta$. The notation $\Gamma \setminus \Delta$ is used for the environment whose domain is $\mathtt{dom}(\Gamma) \setminus \mathtt{dom}(\Delta)$, defined as expected; $\mathtt{x}_1 : \mathtt{A}_1; \ldots; \mathtt{x}_n : \mathtt{A}_n$ is the environment assigning $\mathtt{A}_i$ to $\mathtt{x}_i$, for $1 \leq i \leq n$, and $[\,]$ to any other variable; $\Gamma|_{\mathtt{p}}$ denotes the environment such that $\Gamma|_{\mathtt{p}}(\mathtt{x}) = \Gamma(\mathtt{x})$, if $\mathtt{x} \in \mathtt{fv}(\mathtt{p})$, $[\,]$ otherwise. We also assume that $\Gamma; \mathtt{x} : [\,]$ is identical to Γ. Finally, $\Gamma \sqsubseteq \Delta$ means that $\mathtt{x} \in \mathtt{dom}(\Gamma)$ implies $\mathtt{x} \in \mathtt{dom}(\Delta)$ and $\Gamma(\mathtt{x}) \sqsubseteq \Delta(\mathtt{x})$.

The **type assignment system** $\mathcal{P}$ (*cf.* Figure 1) is a set of typing rules assigning both types and multiset types of $\mathcal{T}$ to terms of $\Lambda_{\mathtt{p}}$. It uses an auxiliary system assigning multiset types to patterns. We write $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$ (resp. $\Pi \triangleright \Gamma \vdash \mathtt{t} : \mathtt{A}$ and $\Pi \triangleright \Gamma \Vdash \mathtt{p} : \mathtt{A}$) to denote a **typing derivation** ending in the sequent $\Gamma \vdash \mathtt{t} : \sigma$ (resp. $\Gamma \vdash \mathtt{t} : \mathtt{A}$ and $\Gamma \Vdash \mathtt{p} : \mathtt{A}$), in which case $\mathtt{t}$ (resp. $\mathtt{p}$) is called the **subject** of Π and σ or $\mathtt{A}$ its **object**. By abuse of notation, $\Gamma \vdash \mathtt{t} : \sigma$ (resp. $\Gamma \vdash \mathtt{t} : \mathtt{A}$ and $\Gamma \Vdash \mathtt{p} : \mathtt{A}$) also denotes the existence of some typing derivation ending with this sequent. A derivation $\Pi \triangleright \Gamma \vdash \mathtt{t} : \mu$ is **meaningful** if μ is either a type or a multiset $\neq [\,]$. A pattern $\mathtt{p}$ is **typable** if there is a derivation whose subject is $\mathtt{p}$; a term $\mathtt{t}$ is **typable** if there is derivation whose subject is $\mathtt{t}$ and whose object is a type, or equivalently if there is a a meaningful derivation whose subject is $\mathtt{t}$. We will prove later that every pattern is in fact typable (*cf.* Corollary 3.6). The **measure** of a typing derivation Π, written $\mathtt{meas}(\Pi)$, is the number of all the typing rules in Π except $(\mathtt{many})$.

Rules $(\mathtt{ax})$ and $(\mathtt{app})$ are those used for the λ-calculus in [10, 15]. Rule $(\mathtt{abs})$ is the natural extension to patterns of the standard rule for abstraction used in the λ-calculus. Linearity of patterns is guaranteed by the clause $\mathtt{p}\#\mathtt{q}$ in rule $(\mathtt{pairpat})$. Rule $(\mathtt{varpat})$ with $\mathtt{A} = [\,]$ is essential to type *erasing* functions such as for example $\lambda\mathtt{x}.\mathtt{I}$. The rule $(\mathtt{many})$ allows to assign multiset types to terms, and it cannot be iterated; in particular note that every term can be assigned with this typing rule the empty multiset type by setting $K = \emptyset$. The

$$\frac{}{\mathtt{x} : \mathtt{A} \Vdash \mathtt{x} : \mathtt{A}}\ (\mathtt{varpat}) \qquad \frac{\Gamma \Vdash \mathtt{p} : \mathtt{A} \quad \Delta \Vdash \mathtt{q} : \mathtt{B} \quad \mathtt{p}\#\mathtt{q}}{\Gamma + \Delta \Vdash \langle \mathtt{p}, \mathtt{q} \rangle : [\times(\mathtt{A}, \mathtt{B})]}\ (\mathtt{pairpat})$$

Patterns

$$\frac{}{\mathtt{x} : [\sigma] \vdash \mathtt{x} : \sigma}\ (\mathtt{ax}) \qquad \frac{(\Gamma_k \vdash \mathtt{t} : \sigma_k)_{k \in K}}{+_{k\in K}\Gamma_k \vdash \mathtt{t} : [\sigma_k]_{k\in K}}\ (\mathtt{many})$$

$$\frac{\Gamma \vdash \mathtt{t} : \sigma \quad \Gamma|_{\mathtt{p}} \Vdash \mathtt{p} : \mathtt{A}}{\Gamma \setminus \Gamma|_{\mathtt{p}} \vdash \lambda \mathtt{p}.\mathtt{t} : \mathtt{A} \to \sigma}\ (\mathtt{abs}) \qquad \frac{\Gamma \vdash \mathtt{t} : \mathtt{A} \to \sigma \qquad \Delta \vdash \mathtt{u} : \mathtt{A}}{\Gamma + \Delta \vdash \mathtt{t}\,\mathtt{u} : \sigma}\ (\mathtt{app})$$

$$\frac{\Gamma \vdash \mathtt{t} : \mathtt{A} \quad \Delta \vdash \mathtt{u} : \mathtt{B}}{\Gamma + \Delta \vdash \langle \mathtt{t}, \mathtt{u} \rangle : \times(\mathtt{A}, \mathtt{B})}\ (\mathtt{pair}) \qquad \frac{\Gamma \vdash \mathtt{t} : \sigma \quad \Gamma|_{\mathtt{p}} \Vdash \mathtt{p} : \mathtt{A} \quad \Delta \vdash \mathtt{u} : \mathtt{A}}{(\Gamma \setminus \Gamma|_{\mathtt{p}}) + \Delta \vdash \mathtt{t}[\mathtt{p}/\mathtt{u}] : \sigma}\ (\mathtt{sub})$$

Terms

Figure 1: The Type Assignment System $\mathcal{P}$.

rule ($\mathtt{pair}$) is self explanatory; note that, in case both the objects of its premises are empty multisets, it allows to type a pair like $\langle \delta\delta, \delta\delta \rangle$ without assigning types to its components, thus $(\lambda\langle \mathtt{x}, \mathtt{y}\rangle.\mathtt{I})\langle \delta\delta, \delta\delta \rangle$ will be typable too, whereas $\delta\delta$ will not. This choice reflects the fact that every pair is canonical, so any kind of pair needs to be typed. Rule ($\mathtt{sub}$) is the more subtle one: in order to type $\mathtt{t}[\mathtt{p}/\mathtt{u}]$, on one hand we need to type $\mathtt{t}$, and on the other one, we need to check that $\mathtt{p}$ and $\mathtt{u}$ can be assigned the same types.

The system is relevant, in the sense that only the used premises are registered in the typing environments. This property, formally stated in the following lemma, will be an important technical tool used to develop the inhabitation algorithm.

Lemma 3.2 (Relevance)**.**

- *If* $\Gamma \Vdash \mathtt{p} : \mathtt{A}$*, then* $\mathtt{dom}(\Gamma) \subseteq \mathtt{fv}(\mathtt{p})$.
- *If* $\Gamma \vdash \mathtt{t} : \sigma$*, then* $\mathtt{dom}(\Gamma) \subseteq \mathtt{fv}(\mathtt{t})$.

Proof. By induction on the typing derivations. □

A first elementary property of the type system is that head occurrences are always typed:

Lemma 3.3. *If* $\mathtt{H}[\mathtt{t}]$ *is typable, then t is typable.*

Proof. Let $\Pi \triangleright \Gamma \vdash \mathtt{H}[\mathtt{t}] : \mu$ be meaningful, then it is easy to prove, by induction on $\mathtt{H}$, that the occurrence of $\mathtt{t}$ filling the hole of $\mathtt{H}$ is always typed. □

3.1. **On the Typing of Patterns.** The system $\mathcal{P}$ features two kinds of typings: those of the form $\Gamma \vdash \mathtt{t} : \mu$, for terms, and those of the form $\Gamma \Vdash \mathtt{p} : \mathtt{A}$, for patterns. These are, of course, fundamentally dissymetric notions: $\vdash$ is undecidable and non deterministic (a given term may have several types in a given environment), whereas $\Vdash$ is decidable and deterministic. As a matter of fact the *unique* type $\mathtt{A}$ such that $\Gamma \Vdash \mathtt{p} : \mathtt{A}$ could have been denoted by $\Gamma(\mathtt{p})$, as we do for variables. However, we decided to keep the typing judgements $\Gamma \Vdash \mathtt{p} : \mathtt{A}$ since they allow for a clearer formulation of the typing rules of $\mathcal{P}$. Some preliminary definitions are given below to prove the uniqueness of the typing of patterns.

Given two patterns $\mathtt{p}$ and $\mathtt{q}$, we say that $\mathtt{p}$ **occurs** in $\mathtt{q}$ if

- either $\mathtt{p} = \mathtt{q}$
- or $\mathtt{q} = \langle \mathtt{q}_1, \mathtt{q}_2 \rangle$ and either $\mathtt{p}$ occurs in $\mathtt{q}_1$ or $\mathtt{p}$ occurs occurs in $\mathtt{q}_2$.

Remark that, by linearity of patterns, at most one of the conditions in the second item above may hold.

If $\mathtt{p}$ occurs in $\mathtt{q}$, the multiset type $\mathtt{A}^{\mathtt{q}}_{\mathtt{p}}$ is defined as follows:

- $\mathtt{A}^{\mathtt{p}}_{\mathtt{p}} = \mathtt{A}$
- $[\times(\mathtt{A},\mathtt{B})]^{\langle \mathtt{q}_1,\mathtt{q}_2\rangle}_{\mathtt{p}} = \begin{cases} \mathtt{A}^{\mathtt{q}_1}_{\mathtt{p}} & \text{if } \mathtt{p} \text{ occurs in } \mathtt{q}_1 \\ \mathtt{B}^{\mathtt{q}_2}_{\mathtt{p}} & \text{if } \mathtt{p} \text{ occurs in } \mathtt{q}_2 \end{cases}$
- $\mathtt{A}^{\langle \mathtt{q}_1,\mathtt{q}_2\rangle}_{\mathtt{p}}$ is undefined if $\mathtt{A}$ is not of the shape $[\times(\mathtt{C},\mathtt{B})]$, for some $\mathtt{C},\mathtt{B}$.

Typings of patterns can be characterized as follows:

Lemma 3.4. *For every environment Γ and pattern $\mathtt{p}$, $\Gamma \Vdash \mathtt{p} : \mathtt{A}$ if and only if $\mathtt{dom}(\Gamma) \subseteq \mathtt{fv}(\mathtt{p})$ and for all $\mathtt{q}$ occurring in $\mathtt{p}$, $\mathtt{A}^{\mathtt{p}}_{\mathtt{q}}$ is defined and $\Gamma|_{\mathtt{q}} \Vdash \mathtt{q} : \mathtt{A}^{\mathtt{p}}_{\mathtt{q}}$.*

Proof. ($\Rightarrow$): If $\Gamma \Vdash \mathtt{p} : \mathtt{A}$, then $\mathtt{dom}(\Gamma) \subseteq \mathtt{fv}(\mathtt{p})$ by Lemma 3.2. The proof is by induction on $\mathtt{p}$. If $\mathtt{p} = \mathtt{x}$ then either $\mathtt{x} \notin \mathtt{dom}(\Gamma)$ and $\mathtt{A} = [\,]$, or $\Gamma(\mathtt{x}) = \mathtt{B} \neq [\,]$ and $\mathtt{A} = \mathtt{B}^{\mathtt{x}}_{\mathtt{x}} = \mathtt{B}$. If the considered pattern is $\langle \mathtt{p}, \mathtt{q} \rangle$, then the last rule of its type derivation is

$$\frac{\Gamma \Vdash \mathtt{p} : \mathtt{B} \quad \Delta \Vdash \mathtt{q} : \mathtt{C} \quad \mathtt{p}\#\mathtt{q}}{\Gamma + \Delta \Vdash \langle \mathtt{p}, \mathtt{q} \rangle : [\times(\mathtt{B},\mathtt{C})]}$$

where $\mathtt{A} = [\times(\mathtt{B},\mathtt{C})]$. By the induction hypothesis $\mathtt{dom}(\Gamma) \subseteq \mathtt{fv}(\mathtt{p})$, $\mathtt{dom}(\Delta) \subseteq \mathtt{fv}(\mathtt{q})$ and for every $\mathtt{p}', \mathtt{q}'$ occurring respectively in $\mathtt{p}$ and $\mathtt{q}$, $\Gamma|_{\mathtt{p}'} \Vdash \mathtt{p}' : \mathtt{B}^{\mathtt{p}}_{\mathtt{p}'}$ and $\Delta|_{\mathtt{q}'} \Vdash \mathtt{q}' : \mathtt{B}^{\mathtt{q}}_{\mathtt{q}'}$. Note that every pattern occurring in $\langle \mathtt{p}, \mathtt{q} \rangle$ is either $\langle \mathtt{p}, \mathtt{q} \rangle$ itself or it occurs in exactly one of the two components of the pair. In the first case the proof is trivial. In the second one, since the domains of Γ and Δ are disjoint, by linearity of $\langle \mathtt{p}, \mathtt{q} \rangle$, $\Gamma|_{\mathtt{p}'} + \Delta|_{\mathtt{p}'} = (\Gamma + \Delta)|_{\mathtt{p}'}$ and $\Gamma|_{\mathtt{q}'} + \Delta|_{\mathtt{q}'} = (\Gamma + \Delta)|_{\mathtt{q}'}$, and the proof follows by induction.

($\Leftarrow$): The proof is again by induction on $\mathtt{p}$. □

Example 3.5. Let $\mathtt{p} = \langle \langle \mathtt{x}, \mathtt{y} \rangle, \mathtt{w} \rangle$ and $\Gamma = \mathtt{x} : [\alpha, \beta], \mathtt{y} : [\gamma]$. The (sub)patterns occurring in $\mathtt{p}$ are $\mathtt{x}$, $\mathtt{y}$, $\mathtt{w}$, $\langle \mathtt{x}, \mathtt{y} \rangle$ and $\mathtt{p}$. In the typing environment Γ restricted to its free variables, each (sub)pattern can be typed by a unique multiset:

$$\begin{array}{lcllcl} \Gamma|_{\mathtt{x}} & = & \mathtt{x} : [\alpha,\beta] \Vdash \mathtt{x} : [\alpha,\beta] & \Gamma|_{\langle \mathtt{x},\mathtt{y}\rangle} & = & \Gamma \Vdash \langle \mathtt{x},\mathtt{y}\rangle : [\times([\alpha,\beta],[\gamma])] \\ \Gamma|_{\mathtt{y}} & = & \mathtt{y} : [\gamma] \Vdash \mathtt{y} : [\gamma] & \Gamma|_{\mathtt{p}} & = & \Gamma \Vdash \mathtt{p} : [\times([\times([\alpha,\beta],[\gamma])],[\,])] \\ \Gamma|_{\mathtt{w}} & = & \emptyset \Vdash \mathtt{w} : [\,] & & & \end{array}$$

If we denote by $\mathtt{A}$ the type $[\times([\times([\alpha,\beta],[\gamma])],[\,])]$, then it is easy to verify that:

$$\begin{array}{lcl@{\qquad\qquad}lcl} \mathtt{A}^{\mathtt{p}}_{\mathtt{x}} & = & [\alpha,\beta] & \mathtt{A}^{\mathtt{p}}_{\langle\mathtt{x},\mathtt{y}\rangle} & = & [\times([\alpha,\beta],[\gamma])] \\ \mathtt{A}^{\mathtt{p}}_{\mathtt{y}} & = & [\gamma] & \mathtt{A}^{\mathtt{p}}_{\mathtt{p}} & = & \mathtt{A} \\ \mathtt{A}^{\mathtt{p}}_{\mathtt{w}} & = & [\,] & & & \end{array}$$

The following corollary follows immediately.

Corollary 3.6. *For every pattern* $\mathtt{p}$ *and every environment* Γ *such that* $\mathtt{dom}(\Gamma) \subseteq \mathtt{fv}(\mathtt{p})$, *there exists a unique multiset* $\mathtt{A}$ *such that* $\Gamma \Vdash \mathtt{p} : \mathtt{A}$. *Moreover if* $\mathtt{p} = \langle \mathtt{p}_1, \mathtt{p}_2\rangle$ *then* $\mathtt{A} = [\times(\mathtt{B},\mathtt{C})]$, *for some* $\mathtt{B},\mathtt{C}$.

3.2. **Main Properties of system $\mathcal{P}$.** We are going to define the notion of *typed occurrences* of a typing derivation, which plays an essential role in the rest of this paper: indeed, thanks to the use of non-idempotent intersection types, a combinatorial argument based on a measure on typing derivations (*cf.* Lemma 3.10(1)), allows to prove the termination of reduction of redexes occurring in typed occurrences of their respective typing derivations.

Given a typing derivation $\Pi \triangleright \Gamma \vdash \mathtt{C}[\mathtt{u}] : \sigma$, the occurrence of $\mathtt{u}$ in the hole of $\mathtt{C}$ is a typed occurrence of Π if and only if $\mathtt{u}$ is the subject of a meaningful subderivation of Π. More precisely:

Definition 3.7. Given a type derivation Π, the set of **typed occurrences** of Π, written $\mathtt{toc}(\Pi)$, is the set of contexts defined by induction on Π as follows.

- If Π ends with $(\mathtt{ax})$, then $\mathtt{toc}(\Pi) := \{\Box\}$.
- If Π ends with $(\mathtt{pair})$ with subject $\langle\mathtt{u},\mathtt{v}\rangle$ and premises Π_i ($i = \{1,2\}$) then $\mathtt{toc}(\Pi) := \{\Box\} \cup \{\langle\mathtt{C},\mathtt{v}\rangle \mid \mathtt{C} \in \mathtt{toc}(\Pi_1)\} \cup \{\langle\mathtt{u},\mathtt{C}\rangle \mid \mathtt{C} \in \mathtt{toc}(\Pi_2)\}$.
- If Π ends with $(\mathtt{abs})$ with subject $\lambda\mathtt{p}.\mathtt{u}$ and premise Π' then $\mathtt{toc}(\Pi) := \{\Box\} \cup \{\lambda\mathtt{p}.\mathtt{C} \mid \mathtt{C} \in \mathtt{toc}(\Pi')\}$.
- If Π ends with $(\mathtt{app})$ with subject $\mathtt{tu}$ and premises Π_1 and Π_2 with subjects $\mathtt{t}$ and $\mathtt{u}$ respectively, then $\mathtt{toc}(\Pi) := \{\Box\} \cup \{\mathtt{Cu} \mid \mathtt{C} \in \mathtt{toc}(\Pi_1)\} \cup \{\mathtt{tC} \mid \mathtt{C} \in \mathtt{toc}(\Pi_2)\}$.
- If Π ends with $(\mathtt{sub})$ with subject $\mathtt{t}[\mathtt{p}/\mathtt{u}]$ and premises Π_1 and Π_2 with subjects $\mathtt{t}$ and $\mathtt{u}$ respectively, then $\mathtt{toc}(\Pi) := \{\Box\} \cup \{\mathtt{C}[\mathtt{p}/\mathtt{u}] \mid \mathtt{C} \in \mathtt{toc}(\Pi_1)\} \cup \{\mathtt{t}[\mathtt{p}/\mathtt{C}] \mid \mathtt{C} \in \mathtt{toc}(\Pi_2)\}$.
- If Π ends with $(\mathtt{many})$, with premises Π_k ($k \in K$), then $\mathtt{toc}(\Pi) := \cup_{k\in K}\mathtt{toc}(\Pi_k)$.

Example 3.8. Given the following derivations Π and Π', the occurrences $\Box$ and $\Box\mathtt{y}$ belong to both $\mathtt{toc}(\Pi)$ and $\mathtt{toc}(\Pi')$ while $\mathtt{x}\Box$ belongs to $\mathtt{toc}(\Pi)$ but not to $\mathtt{toc}(\Pi')$.

$$\Pi \triangleright \dfrac{\dfrac{}{\mathtt{x} : [[\tau] \to \tau] \vdash \mathtt{x} : [\tau] \to \tau} \qquad \dfrac{\dfrac{}{\mathtt{y} : [\tau] \vdash \mathtt{y} : \tau}}{\mathtt{y} : [\tau] \vdash \mathtt{y} : [\tau]}}{\mathtt{x} : [[\tau] \to \tau], \mathtt{y} : [\tau] \vdash \mathtt{xy} : \tau}$$

$$\Pi' \triangleright \dfrac{\dfrac{}{\mathtt{x} : [[\,] \to \tau] \vdash \mathtt{x} : [\,] \to \tau} \qquad \dfrac{}{\emptyset \vdash \mathtt{y} : [\,]}}{\mathtt{x} : [[\,] \to \tau] \vdash \mathtt{xy} : \tau}$$

The type assignment system $\mathcal{P}$ enjoys the fundamental properties of subject reduction and subject expansion, based respectively on substitution and anti-substitution properties whose proofs can be found in [2].

Lemma 3.9 **(Substitution/Anti-Substitution Lemma).**

(1) *If* $\Pi_{\mathtt{t}} \triangleright \Gamma_{\mathtt{t}}; \mathtt{x} : \mathtt{A} \vdash \mathtt{t} : \tau$ *and* $\Pi_{\mathtt{u}} \triangleright \Gamma_{\mathtt{u}} \vdash \mathtt{u} : \mathtt{A}$*, then there exists* $\Pi \triangleright \Gamma_{\mathtt{t}} + \Gamma_{\mathtt{u}} \triangleright \mathtt{t}\{\mathtt{x}/\mathtt{u}\} : \tau$ *such that* $\mathtt{meas}(\Pi) \leq \mathtt{meas}(\Pi_{\mathtt{t}}) + \mathtt{meas}(\Pi_{\mathtt{u}})$*.*

(2) *If* $\Pi \triangleright \Gamma \vdash \mathtt{t}\{\mathtt{x}/\mathtt{u}\} : \tau$*, then there exist derivations* Π_t *and* $\Pi_{\mathtt{u}}$*, environments* $\Gamma_{\mathtt{t}}, \Gamma_{\mathtt{u}}$ *and multiset* $\mathtt{A}$ *such that* $\Pi_{\mathtt{t}} \triangleright \Gamma_{\mathtt{t}}; \mathtt{x} : \mathtt{A} \vdash \mathtt{t} : \tau$*,* $\Pi_{\mathtt{u}} \triangleright \Gamma_{\mathtt{u}} \vdash \mathtt{u} : \mathtt{A}$ *and* $\Gamma = \Gamma_{\mathtt{t}} + \Gamma_{\mathtt{u}}$*.*

On the other hand, the measure of any typing derivation is not increasing by reduction. Moreover, the measure strictly decreases for the reduction steps that are *typed*. This property makes easier the proof of the “only if” part of Theorem 3.11.

Lemma 3.10.

(1) **(Weighted Subject Reduction)** *If* $\Pi \triangleright \Gamma \vdash \mathtt{t} : \tau$ *and* $\mathtt{t} \to \mathtt{t}'$*, then* $\Pi' \triangleright \Gamma \vdash \mathtt{t}' : \tau$ *and* $\mathtt{meas}(\Pi') \leq \mathtt{meas}(\Pi)$*. Moreover, if the reduced redex occurs in a typed occurrence of* Π*, then* $\mathtt{meas}(\Pi') < \mathtt{meas}(\Pi)$*.*

(2) **(Subject Expansion)** *If* $\Pi' \triangleright \Gamma' \vdash \mathtt{t}' : \sigma$ *and* $\mathtt{t} \to \mathtt{t}'$*, then* $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$*.*

Proof. Both proofs are by induction on the reduction relation $\mathtt{t} \to \mathtt{t}'$. For the base cases (reduction at the root position): the rules $(\mathtt{dB}), (\mathtt{subs}), (\mathtt{match}_s)$ are treated exactly as in [2], and the rules $(\mathtt{match}_f), (\mathtt{app}_f), (\mathtt{rem}_f), (\mathtt{lem}_f), (\mathtt{la}_f), (\mathtt{abs}_f)$ do not apply since the term $\mathtt{t}$ (resp. $\mathtt{t}'$) would not be typable. All the inductive cases are straightforward, and in particular, when the reduction occurs in an untyped position, then the measures of Π and Π' are equal. □

Given $\Pi \triangleright \Gamma \vdash \mathtt{t} : \tau$, the term $\mathtt{t}$ is said to be in **Π-normal form**, also written **Π-nf**, if for every typed occurrence $\mathtt{C} \in \mathtt{toc}(\Pi)$ such that $\mathtt{t} = \mathtt{C}[\mathtt{u}]$, the subterm $\mathtt{u}$ is not a redex.

We are now ready to provide the logical characterization of terms having canonical form. This proof has been already given in [2], based on the property that, if a term has a canonical form, then there is a **head** reduction strategy reaching this normal form. But in order to make this paper self contained, we reformulate here the proof, using (for the completeness proof) a more general approach, namely that every reduction strategy choosing at least all the typed redexes reaches a canonical form.

Theorem 3.11 (Characterizing Canonicity)**.** *A term* $\mathtt{t}$ *is typable iff* $\mathtt{t}$ *has a canonical form.*

Proof.

- (if) We reason by induction on the grammar defining the canonical forms.

  We first consider $\mathcal{K}$-canonical forms, for which we prove a stronger property, namely that for every $\mathtt{t} \in \mathcal{K}$, for every σ there is an environment Γ such that $\Gamma \vdash \mathtt{t} : \sigma$. We reason by induction on the grammar defining $\mathcal{K}$.

  If $\mathtt{t} = \mathtt{x}$, the proof is straightforward. If $\mathtt{t} = \mathtt{vu}$, then by the *i.h.* there is a typing derivation $\Gamma \vdash \mathtt{v} : [\,] \to \sigma$. Since to every term the multiset $[\,]$ can be assigned by rule $(\mathtt{many})$ with an empty premise, the result follows by application of rule $(\mathtt{app})$.

  Let $\mathtt{t} = \mathtt{u}[\langle \mathtt{p}, \mathtt{q} \rangle/\mathtt{v}]$. By the *i.h.* for every σ, there is Γ such that $\Gamma \vdash \mathtt{u} : \sigma$. By Corollary 3.6, $\Gamma|_{\langle \mathtt{p},\mathtt{q} \rangle} \Vdash \langle \mathtt{p}, \mathtt{q} \rangle : [\times(\mathtt{A}, \mathtt{B})]$ for some $\mathtt{A}, \mathtt{B}$. Then by the *i.h.* again there is Δ such that $\Delta \vdash \mathtt{v} : \times(\mathtt{A}, \mathtt{B})$, and so, by rule $(\mathtt{many})$, $\Delta \vdash \mathtt{v} : [\times(\mathtt{A}, \mathtt{B})]$. Then, by applying rule $(\mathtt{sub})$, we get $\Gamma \setminus \Gamma|_{\langle \mathtt{p},\mathtt{q} \rangle} + \Delta \vdash \mathtt{u}[\langle \mathtt{p}, \mathtt{q} \rangle/\mathtt{v}] : \sigma$.

  Now, let $\mathtt{t}$ be a $\mathcal{J}$-canonical form. If $\mathtt{t} = \langle \mathtt{u}, \mathtt{v} \rangle$ then by rules $(\mathtt{many})$ and $(\mathtt{pair})$ $\vdash \langle \mathtt{u}, \mathtt{v} \rangle : \times([\,], [\,])$. If $\mathtt{t} = \lambda \mathtt{p}.\mathtt{u}$, then $\mathtt{u}$ can be typed by the *i.h.* so that let $\Gamma \vdash \mathtt{u} : \sigma$. If $\mathtt{p} = \mathtt{x}$, for some $\mathtt{x}$, then by applying rule $(\mathtt{abs})$, $\Gamma \setminus \Gamma|_{\mathtt{x}} \vdash \lambda \mathtt{x}.\mathtt{u} : \Gamma(\mathtt{x}) \to \sigma$, otherwise, by Corollary 3.6, $\Gamma|_{\mathtt{p}} \Vdash \mathtt{p} : [\times(\mathtt{A}, \mathtt{B})]$, for some $\mathtt{A}, \mathtt{B}$, and then $\Gamma \setminus \Gamma|_{\mathtt{p}} \vdash \lambda \mathtt{x}.\mathtt{u} : [\times(\mathtt{A}, \mathtt{B})] \to \sigma$, always by rule ($\mathtt{abs}$). Let $\mathtt{t} = \mathtt{t}'[\langle \mathtt{p}, \mathtt{q}\rangle/\mathtt{v}]$, where $\mathtt{t}'$ (resp. $\mathtt{v}$) is a $\mathcal{J}$ (resp. $\mathcal{K}$) canonical form. By the *i.h.* there are Γ, σ such that $\Gamma \vdash \mathtt{t}' : \sigma$. Moreover, Corollary 3.6 gives $\Gamma|_{\langle \mathtt{p},\mathtt{q}\rangle} \Vdash \langle \mathtt{p}, \mathtt{q}\rangle : [\times(\mathtt{A}, \mathtt{B})]$ for some $\mathtt{A}, \mathtt{B}$. Since $\mathtt{v}$ is a $\mathcal{K}$-canonical form, then $\Delta \vdash \mathtt{v} : \times(\mathtt{A}, \mathtt{B})$ as shown above, and then $\Delta \vdash \mathtt{v} : [\times(\mathtt{A}, \mathtt{B})]$, by rule ($\mathtt{many}$). Thus $\Gamma + \Delta \vdash \mathtt{t}'[\langle \mathtt{p}, \mathtt{q}\rangle/\mathtt{v}] : \sigma$ by applying rule ($\mathtt{sub}$).

- (only if) Let $\mathtt{t}$ be a typable term, *i.e.* $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$. Consider a reduction strategy $\mathcal{ST}$ that always chooses a *typed* redex occurrence. By Lemma 3.10(1) and Lemma 2.5 the strategy $\mathcal{ST}$ always terminates. Let $\mathtt{t}'$ be a normal-form of $\mathtt{t}$ for the strategy $\mathcal{ST}$, *i.e.* $\mathtt{t}$ reduces to $\mathtt{t}'$ using $\mathcal{ST}$, and $\mathtt{t}'$ has no typed redex occurrence. We know that $\Pi' \triangleright \Gamma \vdash \mathtt{t}' : \sigma$ by Lemma 3.10(1). We now proceed by induction on Π', by taking into account the notion of typed occurrence of Π'.

  If Π' ends with ($\mathtt{ax}$), then its subject is $\mathtt{x}$, which is canonical. If Π' ends with ($\mathtt{abs}$) with subject $\lambda\mathtt{p}.\mathtt{u}$ and premise Π'' with subject $\mathtt{u}$, then $\mathtt{u}$ has no typed redex occurrences, so it is canonical by the *i.h.* We conclude that $\lambda\mathtt{p}.\mathtt{u}$ is canonical too, by definition of canonical form. If Π ends with ($\mathtt{app}$) with subject $\mathtt{tu}$ and premises Π_1 and Π_2 having subjects $\mathtt{t}$ and $\mathtt{u}$ respectively, then $\mathtt{t}$, which is also typable, has no typed redex occurrences, so that it is canonical by the *i.h.* Moreover $\Box \in \mathtt{toc}(\Pi)$, so $\mathtt{tu}$ cannot be a redex. This implies that $\mathtt{t}$ cannot be an abstraction, so it is a $\mathcal{K}$ canonical form, and consequently $\mathtt{tu}$ is a $\mathcal{K}$ canonical form too. Suppose Π ends with ($\mathtt{sub}$) with subject $\mathtt{t}[\mathtt{p}/\mathtt{u}]$ and premises Π_1 and Π_2 with subjects $\mathtt{t}$ and $\mathtt{u}$ respectively. The term $\mathtt{t}$, which is typable, has no typed redex occurrences, so it is canonical by the *i.h.* Moreover, $\mathtt{p}$ cannot be a variable, otherwise the term would have a typed $\mathtt{subs}$-redex occurrence, so, by Corollary 3.6, $\Gamma|_{\mathtt{p}} \Vdash \mathtt{p} : [\times(\mathtt{A}, \mathtt{B})]$, for some $\mathtt{A}, \mathtt{B}$, where Γ is the typing environment of $\mathtt{t}$. The term $\mathtt{u}$ is also typed. Since $\mathtt{u}$ cannot be neither an abstraction (rule $\mathtt{match}_f$) nor a pair (rule $\mathtt{match}_s$), it is necessarily a $\mathcal{K}$ canonical form, and consequently $\mathtt{t}[\mathtt{p}/\mathtt{u}]$ is canonical. Finally, if Π' ends with ($\mathtt{pair}$), then its subject is a pair, which is a canonical form. This concludes the proof. $\Box$

4. Inhabitation for System $\mathcal{P}$

Given μ, a type or a multiset type, the inhabitation problem consists in finding a closed term $\mathtt{t}$ such that $\vdash \mathtt{t} : \mu$ is derivable. These notions will naturally be generalized later to non-closed terms. Since system $\mathcal{P}$ characterizes canonicity, it is natural to look for inhabitants in *canonical form*. The next Lemma proves that the problem can be simplified, namely that it is sufficient to look for inhabitants in *pure canonical form*, *i.e.* without nested substitution (we postpone the proof of this lemma to Section 4.1).

Lemma 4.1. *Let* $\mathtt{t}$ *be a canonical form. If* $\Pi \triangleright \Gamma \vdash \mathtt{t} : \mu$ *is derivable, then there is some type derivation* Π' *and some pure canonical form* $\mathtt{t}'$ *such that* $\Pi' \triangleright \Gamma \vdash \mathtt{t}' : \mu$ *is derivable.*

We already noticed that the system $\mathcal{P}$ allows to assign the multiset $[\,]$ to terms through the rule ($\mathtt{many}$). As a consequence, a typed term may contain untyped subterms. In order to identify inhabitants in such cases we introduce a term constant Ω to denote a generic untyped subterm. Accordingly, the type system $\mathcal{P}$ is extended to the new grammar of terms possibly containing Ω, which can only be typed using a particular case of the ($\mathtt{many}$) rule:

$$\frac{}{\vdash \Omega : [\,]}\ (\mathtt{many})$$

So the inhabitation algorithm should produce **approximate normal forms** (denoted $\mathtt{a},\mathtt{b},\mathtt{c}$), also written anf, defined as follows:

$$\begin{array}{lcl} \mathtt{a},\mathtt{b},\mathtt{c} & ::= & \Omega \mid \mathcal{N} \\ \mathcal{N} & ::= & \lambda\mathtt{p}.\mathcal{N} \mid \langle \mathtt{a},\mathtt{b}\rangle \mid \mathcal{L} \mid \mathcal{N}[\langle \mathtt{p},\mathtt{q}\rangle/\mathcal{L}] \\ \mathcal{L} & ::= & \mathtt{x} \mid \mathcal{L}\mathtt{a} \end{array}$$

The grammar defining anf*s* is similar to that of pure canonical forms, starting, besides variables, also from Ω. The notion of typed occurrences in the new extended system is straightforward. Moreover, an anf does not contain any redex, differently from canonical forms. Roughly speaking, an anf can be seen as a representation of an infinite set of pure canonical forms, obtained by replacing each occurrence of Ω by any term.

Example 4.2. The term $\lambda\langle \mathtt{x},\mathtt{y}\rangle.(\mathtt{x}(\mathtt{II}))[\langle \mathtt{z}_1,\mathtt{z}_2\rangle/\mathtt{yI}]$ is (pure) canonical but not an anf, while $\lambda\langle \mathtt{x},\mathtt{y}\rangle.(\mathtt{x}\Omega)[\langle \mathtt{z}_1,\mathtt{z}_2\rangle/\mathtt{yI}]$ is an anf.

Anf*s* are ordered by the smallest contextual order $\leq$ such that $\Omega \leq \mathtt{a}$, for any $\mathtt{a}$. We also write $\mathtt{a} \leq \mathtt{t}$ when the term $\mathtt{t}$ is obtained from $\mathtt{a}$ by replacing each occurrence of Ω by a term of $\Lambda_{\mathtt{p}}$. Thus for example $\mathtt{x}\Omega\Omega \leq \mathtt{x}(\mathtt{I}\delta)(\delta\delta)$ is obtained by replacing the first (resp. second) occurrence of Ω by $\mathtt{I}\delta$ (resp. $\delta\delta$).

Let $\mathcal{A}(\mathtt{t}) = \{\mathtt{a} \mid \exists \mathtt{u}\ \mathtt{t} \to^* \mathtt{u} \text{ and } \mathtt{a} \leq \mathtt{u}\}$ be the set of **approximants** of the term $\mathtt{t}$, and let $\bigvee$ denote the least upper bound with respect to $\leq$. We write $\uparrow_{i\in I} \mathtt{a}_i$ to denote the fact that $\bigvee\{\mathtt{a}_i\}_{i\in I}$ does exist. Note that $I = \emptyset$ implies $\bigvee\{\mathtt{a}_i\}_{i\in I} = \Omega$. It is easy to check that, for every $\mathtt{t}$ and $\mathtt{a}_1,\ldots \mathtt{a}_n \in \mathcal{A}(\mathtt{t})$, $\uparrow_{i\in\{1,\ldots,n\}} \mathtt{a}_i$. An anf $\mathtt{a}$ is a **head subterm** of $\mathtt{b}$ if either $\mathtt{b} = \mathtt{a}$ or $\mathtt{b} = \mathtt{c}\mathtt{c}'$ and $\mathtt{a}$ is a head subterm of $\mathtt{c}$. It is easy to check that, if $\Gamma \vdash \mathtt{a} : \sigma$ and $\mathtt{a} \leq \mathtt{b}$ (resp. $\mathtt{a} \leq \mathtt{t}$) then $\Gamma \vdash \mathtt{b} : \sigma$ (resp. $\Gamma \vdash \mathtt{t} : \sigma$).

Given $\Pi \rhd \Gamma \vdash \mathtt{t} : \mu$, where $\mathtt{t}$ is in Π-nf (*cf.* Section 3), the **minimal approximant of** Π, written $\mathcal{A}(\Pi)$, is defined by induction on Π as follows:

- $\mathcal{A}(\Gamma \vdash \mathtt{x} : \rho) = \mathtt{x}$.
- If $\Pi \rhd \Gamma \vdash \lambda\mathtt{p}.\mathtt{t} : \mathtt{A} \to \rho$ follows from $\Pi' \rhd \Gamma' \vdash \mathtt{t} : \rho$, then $\mathcal{A}(\Pi) = \lambda\mathtt{p}.\mathcal{A}(\Pi')$, $\mathtt{t}$ being in Π'-nf.
- If $\Pi \rhd \Gamma \vdash \langle \mathtt{t},\mathtt{u}\rangle : \times(\mathtt{A},\mathtt{B})$ follows from $\Pi_1 \rhd \Gamma_1 \vdash \mathtt{t} : \mathtt{A}$ and $\Pi_2 \rhd \Gamma \vdash \mathtt{u} : \mathtt{B}$, then $\mathcal{A}(\Pi) = \langle \mathcal{A}(\Pi_1), \mathcal{A}(\Pi_2)\rangle$.
- If $\Pi \rhd \Gamma = \Gamma' + \Delta \vdash \mathtt{tu} : \rho$ follows from $\Pi_1 \rhd \Gamma' \vdash \mathtt{t} : \mathtt{A} \to \rho$ and $\Pi_2 \rhd \Delta \vdash \mathtt{u} : \mathtt{A}$, then $\mathcal{A}(\Pi) = \mathcal{A}(\Pi_1)(\mathcal{A}(\Pi_2))$.
- If $\Pi \rhd \Gamma = \Gamma' + \Delta \vdash \mathtt{t}[\mathtt{p}/\mathtt{u}] : \tau$ follows from $\Pi' \rhd \Gamma'' \vdash \mathtt{t} : \tau$ and $\Psi \rhd \Delta \vdash \mathtt{u} : \mathtt{A}$, then $\mathcal{A}(\Pi) = \mathcal{A}(\Pi')[\mathtt{p}/\mathcal{A}(\Psi)]$.
- If $\Pi \rhd +_{i\in I}\Gamma_i \vdash \mathtt{t} : [\sigma_i]_{i\in I}$ follows from $(\Pi_i \rhd \Gamma_i \vdash \mathtt{t} : \sigma_i)_{i\in I}$, then $\mathcal{A}(\Pi) = \bigvee_{i\in I} \mathcal{A}(\Pi_i)$.

Example 4.3. Consider the following derivation Π (remember that o is an abbreviation for $\times([\,],[\,])$), built upon the subderivations Π_1 and Π_2 below:

$$\dfrac{\dfrac{\Pi_1 \qquad \dfrac{}{\Vdash \langle \mathtt{z}_1,\mathtt{z}_2\rangle : [o]} \qquad \Pi_2}{\mathtt{x} : [[\,] \to o], \mathtt{y} : [[\,] \to o] \vdash \mathtt{y}(\delta\delta)[\langle \mathtt{z}_1,\mathtt{z}_2\rangle/\mathtt{xI}] : o}}{\vdash \lambda\mathtt{xy}.\mathtt{y}(\delta\delta)[\langle \mathtt{z}_1,\mathtt{z}_2\rangle/\mathtt{xI}] : [[\,] \to o] \to [[\,] \to o] \to o}$$

$$\Pi_1 = \frac{\overline{\mathtt{y} : [[\,] \to o] \vdash \mathtt{y} : [\,] \to o} \qquad \overline{\emptyset \vdash \delta\delta : [\,]}}{\mathtt{y} : [[\,] \to o] \vdash \mathtt{y}(\delta\delta) : o} \qquad \Pi_2 = \frac{\dfrac{\overline{\mathtt{x} : [[\,] \to o] \vdash \mathtt{x} : [\,] \to o} \qquad \overline{\emptyset \vdash \mathtt{I} : [\,]}}{\mathtt{x} : [[\,] \to o] \vdash \mathtt{xI} : o}}{\mathtt{x} : [[\,] \to o] \vdash \mathtt{xI} : [o]}$$

The minimal approximant of Π is $\lambda \mathtt{xy.y}\Omega[\langle \mathtt{z}_1, \mathtt{z}_2\rangle/\mathtt{x}\Omega]$.

A simple induction on $\mathtt{meas}(\Pi)$ allows to show the following:

Lemma 4.4. *If $\Pi \rhd \Gamma \vdash \mathtt{t} : \mu$ and $\mathtt{t}$ is in Π-nf, then $\Pi \rhd \Gamma \vdash \mathcal{A}(\Pi) : \mu$.*

4.1. **From Canonical Forms to Pure Canonical Forms.** In this section we prove that, when a giving typing is inhabited, then it is necessarily inhabited by a pure canonical form. This property turns out to be essential to prove the completeness property of our algorithm (Theorem 4.8), since the algorithm only builds pure canonical forms.

Lemma 4.1. *Let $\mathtt{t}$ be a canonical form. If $\Pi \rhd \Gamma \vdash \mathtt{t} : \mu$ is derivable, then there is some type derivation Π' and some pure canonical form $\mathtt{t}'$ such that $\Pi' \rhd \Gamma \vdash \mathtt{t}' : \mu$ is derivable.*

Proof. By induction on $\Pi \rhd \Gamma \vdash \mathtt{t} : \mu$. The only interesting case is when $\mathtt{t} = \mathtt{t}_0[\mathtt{p}/\mathtt{t}_1]$, where $\mathtt{p}$ is some pair pattern. By construction of Π, there is a type derivation of the following form:

$$\frac{\Gamma \vdash \mathtt{t}_0 : \tau \qquad \Gamma|_{\mathtt{p}} \Vdash \mathtt{p} : [\times(\mathtt{A},\mathtt{B})] \qquad \dfrac{\Delta \vdash \mathtt{t}_1 : \times(\mathtt{A},\mathtt{B})}{\Delta \vdash \mathtt{t}_1 : [\times(\mathtt{A},\mathtt{B})]}}{\Gamma \setminus \Gamma|_{\mathtt{p}} + \Delta \vdash \mathtt{t}_0[\mathtt{p}/\mathtt{t}_1] : \tau}$$

where the shape of the type for $\mathtt{p}$ comes from Corollary 3.6. By the *i.h.* there are pure canonical terms $\mathtt{t}'_0, \mathtt{t}'_1$ such that $\Gamma \vdash \mathtt{t}'_0 : \tau$ and $\Delta \vdash \mathtt{t}'_1 : \times(\mathtt{A},\mathtt{B})$. If $\mathtt{t}'_0[\mathtt{p}/\mathtt{t}'_1]$ is a pure canonical term, then we conclude with a derivation of $\Gamma \setminus \Gamma|_{\mathtt{p}} + \Delta \vdash \mathtt{t}'_0[\mathtt{p}/\mathtt{t}'_1] : \tau$. If $\mathtt{t}'_0[\mathtt{p}/\mathtt{t}'_1]$ is not a pure canonical term, then necessarily $\mathtt{t}'_1 = \mathtt{u}[\mathtt{q}/\mathtt{v}]$, $\mathtt{q}$ being a pair and $\mathtt{u}, \mathtt{v}$ being pure canonical terms. Then there is necessarily a derivation of the following form:

$$\frac{\Delta' \vdash \mathtt{u} : \times(\mathtt{A},\mathtt{B}) \qquad \Delta'|_{\mathtt{q}} \Vdash \mathtt{q} : [\times(\mathtt{C},\mathtt{E})] \qquad \dfrac{\Gamma' \vdash \mathtt{v} : \times(\mathtt{C},\mathtt{E})}{\Gamma' \vdash \mathtt{v} : [\times(\mathtt{C},\mathtt{E})]}}{\Delta = \Delta' \setminus \Delta'|_{\mathtt{q}} + \Gamma' \vdash \mathtt{u}[\mathtt{q}/\mathtt{v}] : \times(\mathtt{A},\mathtt{B})}$$

We can then build the following derivation:

$$\frac{\Gamma \vdash \mathtt{t}'_0 : \tau \qquad \Gamma|_{\mathtt{p}} \Vdash \mathtt{p} : [\times(\mathtt{A},\mathtt{B})] \qquad \dfrac{\Delta' \vdash \mathtt{u} : \times(\mathtt{A},\mathtt{B})}{\Delta' \vdash \mathtt{u} : [\times(\mathtt{A},\mathtt{B})]}}{\Gamma \setminus \Gamma|_{\mathtt{p}} + \Delta' \vdash \mathtt{t}'_0[\mathtt{p}/\mathtt{u}] : \tau}$$

Note that $(\Gamma \setminus \Gamma|_{\mathtt{p}} + \Delta')|_{\mathtt{q}} = \Delta'|_{\mathtt{q}}$, since we can always choose $\mathtt{fv}(\mathtt{q}) \cap \Gamma = \emptyset$, by α conversion. So the following derivation can be built:

$$\frac{\Gamma \setminus \Gamma|_{\mathtt{p}} + \Delta' \vdash \mathtt{t}'_0[\mathtt{p}/\mathtt{u}] : \tau \qquad \Delta'|_{\mathtt{q}} \Vdash \mathtt{q} : [\times(\mathtt{C},\mathtt{E})] \qquad \dfrac{\Gamma' \vdash \mathtt{v} : \times(\mathtt{C},\mathtt{E})}{\Gamma' \vdash \mathtt{v} : [\times(\mathtt{C},\mathtt{E})]}}{(\Gamma \setminus \Gamma|_{\mathtt{p}} + \Delta') \setminus \Delta'|_{\mathtt{q}} + \Gamma' \vdash \mathtt{t}'_0[\mathtt{p}/\mathtt{u}][\mathtt{q}/\mathtt{v}] : \tau}$$

Since $\Gamma \setminus \Gamma|_{\mathtt{p}} \setminus \Delta'|_{\mathtt{q}} = \Gamma \setminus \Gamma|_{\mathtt{p}}$ and $\Delta' \setminus \Delta'|_{\mathtt{q}} + \Gamma' = \Delta$, the proof is given. ∎

4.2. **The Inhabitation Algorithm.** We now show a sound and complete algorithm to solve the inhabitation problem for System $\mathcal{P}$. The algorithm is presented in Figure 2. As usual, in order to solve the problem for closed terms, it is necessary to extend the algorithm to open ones, so, given an environment Γ and a type σ, the algorithm builds the set $\mathtt{T}(\Gamma,\sigma)$ containing *all* the anf*s* $\mathtt{a}$ such that there exists a derivation $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$, with $\mathtt{a} = \mathcal{A}(\Pi)$, then stops[1]. Thus, our algorithm is not an extension of the classical inhabitation algorithm for simple types [8, 18]. In particular, when restricted to simple types, it constructs all the anf*s* inhabiting a given type, while the original algorithm reconstructs just the *long η-normal forms.* The algorithm uses three auxiliary predicates, namely

- $\mathtt{P}_{\mathcal{V}}(\mathtt{A})$, where $\mathcal{V}$ is a finite set of variables, contains the pairs $(\Gamma,\mathtt{p})$ such that (i) $\Gamma \Vdash \mathtt{p} : \mathtt{A}$, and (ii) $\mathtt{p}$ does not contain any variable in $\mathcal{V}$.
- $\mathtt{M}(\Gamma,[\sigma_i]_{i\in I})$, contains all the anf*s* $\mathtt{a} = \bigvee_{i\in I} \mathtt{a}_i$ such that $\Gamma = +_{i\in I}\Gamma_i$, $\mathtt{a}_i \in \mathtt{T}(\Gamma_i,\sigma_i)$ for all $i \in I$, and $\uparrow_{i\in I} \mathtt{a}_i$.
- $\mathtt{H}^{\Delta}_{\mathtt{b}}(\Gamma,\sigma) \triangleright \tau$ contains all the anf*s* $\mathtt{a}$ such that $\mathtt{b}$ is a head subterm of $\mathtt{a}$, and such that if $\mathtt{b} \in \mathtt{T}(\Delta,\sigma)$ then $\mathtt{a} \in \mathtt{T}(\Gamma+\Delta,\tau)$. As a particular case, notice that $\mathtt{b} \in \mathtt{H}^{\Delta}_{\mathtt{b}}(\emptyset,\sigma) \triangleright \sigma$, for all $\mathtt{b} \in \mathcal{L}$, environment Δ and type σ.

Note that a special case of rule $(\mathtt{Many})$ with $I = \emptyset$ is $\dfrac{}{\Omega \in \mathtt{M}(\emptyset,[\,])}$. Note also that the algorithm has different kinds of non-deterministic behaviours, *i.e.* different choices of rules can produce different results. Indeed, given an input $(\Gamma,\mathtt{A}\to\sigma)$, the algorithm may apply a rule like $(\mathtt{Abs})$ in order to decrease the type σ, or a rule like $(\mathtt{Head})$ in order to decrease the environment Γ. Moreover, every rule (R) which is based on some decomposition of the environment and/or the type, like $(\mathtt{Subs})$, admits different applications. In what follows we illustrate the non-deterministic behaviour of the algorithm. For that, we represent a **run of the algorithm** as a tree whose nodes are labeled with the name of the rule being applied.

Example 4.5. We consider different inputs of the form $(\emptyset,\sigma)$, for different types σ. For every such input, we give an output and the corresponding run.

(1) $\sigma = [[\alpha]\to\alpha]\to[\alpha]\to\alpha$.
   (a) output: $\lambda\mathtt{xy.xy}$, run:
   $\mathtt{Abs}(\mathtt{Abs}(\mathtt{Head}(\mathtt{Prefix}(\mathtt{Many}(\mathtt{Head}(\mathtt{Final})),\mathtt{Final})),\mathtt{Varp}),\mathtt{Varp})$.
   (b) output: $\lambda\mathtt{x.x}$, run:
   $\mathtt{Abs}(\mathtt{Head}(\mathtt{Final}),\mathtt{Varp})$.
(2) $\sigma = [[\,]\to\alpha]\to\alpha$. output: $\lambda\mathtt{x.x}\Omega$, run: $\mathtt{Abs}(\mathtt{Head}(\mathtt{Prefix}(\mathtt{Many},\mathtt{Final})),\mathtt{Varp})$.
(3) $\sigma = [[o]\to o, o]\to o$.
   (a) output: $\lambda\mathtt{x.xx}$, run: $\mathtt{Abs}(\mathtt{Head}(\mathtt{Prefix}(\mathtt{Many}(\mathtt{Head}(\mathtt{Final})),\mathtt{Final})),\mathtt{Varp})$.
   (b) Explicit substitutions may be used to consume some, or all, the resources in $[[o]\to o, o]$. output: $\lambda\mathtt{x.x}[\langle\mathtt{y},\mathtt{z}\rangle/\mathtt{x}\langle\Omega,\Omega\rangle]$, run:
   $\mathtt{Abs}(\mathtt{Subs}(\mathtt{Prefix}(\mathtt{Many}(\mathtt{Prod}),\mathtt{Final}),\mathtt{Pairp}(\mathtt{Varp},\mathtt{Varp}),\mathtt{Head}(\mathtt{Final})),\mathtt{Varp})$.
   (c) There are four additional runs, producing the following outputs:
   $\lambda\mathtt{x.x}\langle\Omega,\Omega\rangle[\langle\mathtt{y},\mathtt{z}\rangle/\mathtt{x}]$,
   $\lambda\mathtt{x}.\langle\Omega,\Omega\rangle[\langle\mathtt{y},\mathtt{z}\rangle/\mathtt{xx}]$,
   $\lambda\mathtt{x}.\langle\Omega,\Omega\rangle[\langle\mathtt{y},\mathtt{z}\rangle/\mathtt{x}][\langle\mathtt{w},\mathtt{s}\rangle/\mathtt{x}\langle\Omega,\Omega\rangle]$,
   $\lambda\mathtt{x}.\langle\Omega,\Omega\rangle[\langle\mathtt{y},\mathtt{z}\rangle/\mathtt{x}\langle\Omega,\Omega\rangle][\langle\mathtt{w},\mathtt{s}\rangle/\mathtt{x}]$.

[1]It is worth noticing that, given Γ and σ, the set of anf*s* $\mathtt{a}$ such that there exists a derivation $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$ is possibly infinite. However, the subset of those verifying $\mathtt{a} = \mathcal{A}(\Pi)$ is finite.This is proved in Corollary 4.12.

$$\frac{\mathtt{x} \notin \mathcal{V}}{(\mathtt{x} : \mathtt{A}, \mathtt{x}) \in \mathtt{P}_{\mathcal{V}}(\mathtt{A})}\ (\mathtt{Varp})$$

$$\frac{(\Gamma, \mathtt{p}) \in \mathtt{P}_{\mathcal{V}}(\mathtt{A}) \quad (\Delta, \mathtt{q}) \in \mathtt{P}_{\mathcal{V}}(\mathtt{B}) \quad \mathtt{p}\#\mathtt{q}}{(\Gamma + \Delta, \langle \mathtt{p}, \mathtt{q}\rangle) \in \mathtt{P}_{\mathcal{V}}([\times(\mathtt{A}, \mathtt{B})])}\ (\mathtt{Pairp})$$

$$\frac{\mathtt{a} \in \mathtt{T}(\Gamma + \Delta, \tau) \quad (\Delta, \mathtt{p}) \in \mathtt{P}_{\mathtt{dom}(\Gamma)}(\mathtt{A})}{\lambda \mathtt{p}.\mathtt{a} \in \mathtt{T}(\Gamma, \mathtt{A} \to \tau)}\ (\mathtt{Abs}) \qquad \frac{\mathtt{a} \in \mathtt{M}(\Gamma, \mathtt{A}) \quad \mathtt{b} \in \mathtt{M}(\Delta, \mathtt{B})}{\langle \mathtt{a}, \mathtt{b}\rangle \in \mathtt{T}(\Gamma + \Delta, \times(\mathtt{A}, \mathtt{B}))}\ (\mathtt{Prod})$$

$$\frac{(\mathtt{a}_i \in \mathtt{T}(\Gamma_i, \sigma_i))_{i\in I} \qquad \uparrow_{i\in I} \mathtt{a}_i}{\bigvee_{i\in I} \mathtt{a}_i \in \mathtt{M}(+_{i\in I}\Gamma_i, [\sigma_i]_{i\in I})}\ (\mathtt{Many})$$

$$\frac{\mathtt{a} \in \mathtt{H}^{\mathtt{x}:[\sigma]}_{\mathtt{x}}(\Gamma, \sigma) \rhd \tau}{\mathtt{a} \in \mathtt{T}(\Gamma + \mathtt{x} : [\sigma], \tau)}\ (\mathtt{Head}) \quad \frac{\sigma = \tau}{\mathtt{a} \in \mathtt{H}^{\Delta}_{\mathtt{a}}(\emptyset, \sigma) \rhd \tau}\ (\mathtt{Final})$$

$$\frac{\mathtt{b} \in \mathtt{M}(\Gamma, \mathtt{A}) \quad \mathtt{a} \in \mathtt{H}^{\Delta+\Gamma}_{\mathtt{cb}}(\Lambda, \sigma) \rhd \tau}{\mathtt{a} \in \mathtt{H}^{\Delta}_{\mathtt{c}}(\Gamma + \Lambda, \mathtt{A} \to \sigma) \rhd \tau}\ (\mathtt{Prefix})$$

$$\frac{\mathtt{c} \in \mathtt{H}^{\mathtt{x}:[\sigma]}_{\mathtt{x}}(\Gamma, \sigma) \rhd \mathtt{F}(\sigma)(*) \quad (\Delta, \langle \mathtt{p}, \mathtt{q}\rangle) \in \mathtt{P}_{\mathtt{dom}(\Gamma+\Lambda+(\mathtt{x}:[\sigma]))}([\mathtt{F}(\sigma)]) \quad \mathtt{b} \in \mathtt{T}(\Delta + \Lambda, \tau)}{\mathtt{b}[\langle \mathtt{p}, \mathtt{q}\rangle/\mathtt{c}] \in \mathtt{T}(\Gamma + \Lambda + \mathtt{x} : [\sigma], \tau)}\ (\mathtt{Subs})$$

(*) where the operator $\mathtt{F}()$ on types is defined as follows:

$$\begin{array}{lcl} \mathtt{F}(\alpha) & := & \alpha \\ \mathtt{F}(\pi) & := & \pi \\ \mathtt{F}(\mathtt{A} \to \tau) & := & \mathtt{F}(\tau) \end{array}$$

Figure 2: The inhabitation algorithm

Along the recursive calls of the inhabitation algorithm, the parameters (type and/or environment) decrease strictly, for a suitable notion of measure, so that every run is finite:

Theorem 4.6. *The inhabitation algorithm terminates.*

Proof. See Subsection 4.3. □

We now prove soundness and completeness of our inhabitation algorithm.

Lemma 4.7. $\mathtt{a} \in \mathtt{T}(\Gamma, \sigma) \Leftrightarrow \exists \Pi \rhd \Gamma \vdash \mathtt{a} : \sigma$ *such that* $\mathtt{a} = \mathcal{A}(\Pi)$*.*

Proof. The "only if" part is proved by induction on the rules in Figure 2, and the "if" part is proved by induction on the definition of $\mathcal{A}(\Pi)$ (see Section 4.3 for full details). In both parts, additional statements concerning the predicates of the inhabitation algorithm other than $\mathtt{T}$ are required, in order to strengthen the inductive hypothesis. □

Theorem 4.8 (Soundness and Completeness)**.**

(1) *If* $\mathtt{a} \in \mathtt{T}(\Gamma,\sigma)$ *then, for all* $\mathtt{t}$ *such that* $\mathtt{a} \leq \mathtt{t}$, $\Gamma \vdash \mathtt{t} : \sigma$.
(2) *If* $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$ *then there exists* $\Pi' \triangleright \Gamma \vdash \mathtt{t}' : \sigma$ *such that* $\mathtt{t}'$ *is in* Π'*-nf, and* $\mathcal{A}(\Pi') \in \mathtt{T}(\Gamma,\sigma)$.

Proof. Soundness follows from Lemma 4.7 ($\Rightarrow$) and the fact that $\Gamma \vdash \mathtt{a} : \sigma$ and $\mathtt{a} \leq \mathtt{t}$ imply $\Gamma \vdash \mathtt{t} : \sigma$. Concerning completeness: Theorem 3.11 and Lemma 3.10(1) ensures that $\mathtt{t}$ has a canonical form $\mathtt{t}_0$ such that $\Pi_0 \triangleright \Gamma \vdash \mathtt{t}_0 : \sigma$. Then, Lemma 4.1 guarantees the existence of a pure canonical form $\mathtt{t}'$ such that $\Pi' \triangleright \Gamma \vdash \mathtt{t}' : \sigma$ and $\mathtt{t}'$ is in Π'-nf. Then Lemma 4.4 and Lemma 4.7 ($\Leftarrow$) allow us to conclude. $\square$

4.3. **Properties of the Inhabitation Algorithm.** We prove several properties of the inhabitation algorithm, namely, termination, soundness and completeness.

Termination. Being the inhabitation algorithm non deterministic, proving its termination means to prove both that a single run terminates and that every input generates a finite number of runs. We will prove these two properties separately.

First, let us define the following **measure** on types and environments:

$$\begin{array}{lcl}
\#(\alpha) & := & 1 \\
\#([\sigma_i]_{i\in I}) & := & 1 +_{i\in I} \#(\sigma_i) \\
\#(\times(\mathtt{A},\mathtt{B})) & := & \#(\mathtt{A}) + \#(\mathtt{B}) + 1 \\
\#(\mathtt{A} \to \sigma) & := & \#(\mathtt{A}) + \#(\sigma) + 1 \\
\#(\Gamma) & := & \sum_{\mathtt{x} \in \mathtt{dom}(\Gamma)} \#(\Gamma(\mathtt{x}))
\end{array}$$

The measures are extended to predicates in the following way:

$$\begin{array}{lcl}
\#(\mathtt{T}(\Gamma,\sigma)) & := & \#(\Gamma) + \#(\sigma) \\
\#(\mathtt{H}^{\Delta}_{\mathtt{a}}(\Gamma,\sigma) \triangleright \tau) & := & \#(\Gamma) + \#(\sigma) \\
\#(\mathtt{M}(\Gamma,\mathtt{A})) & := & \#(\Gamma) + \#(\mathtt{A}) \\
\#(\mathtt{P}_{\mathcal{V}}(\mathtt{A})) & := & \#(\mathtt{A})
\end{array}$$

Notice that $\#(\Gamma) \leq \#(\Gamma + \Delta)$, for any Δ. Also, $\#(\Gamma + \Delta) \leq \#(\Gamma) + \#(\Delta)$, thus *e.g.* $\#(\mathtt{x} : [\alpha] + \mathtt{x} : [\alpha]) = \#(\mathtt{x} : [\alpha,\alpha]) = 3 \leq \#(\mathtt{x} : [\alpha]) + \#(\mathtt{x} : [\alpha]) = 4$. Notice also that $\#(\Delta_1) < \#(\Delta_2)$ does not imply $\#(\Delta_1 + \Lambda) < \#(\Delta_2 + \Lambda)$, *e.g.* when $\Delta_1 = \mathtt{x} : [\alpha]$, $\Delta_2 = \mathtt{y} : [\alpha,\alpha]$ and $\Lambda = \mathtt{y} : [\alpha]$. However, as a particular useful case, if $\#(\Delta) + 1 < \#(\mathtt{x} : \mathtt{A})$, then $\#(\Delta+\Lambda) < \#(\mathtt{x} : \mathtt{A}+\Lambda)$. Indeed, if $\mathtt{x} \notin \mathtt{dom}(\Lambda)$, then $\#(\Delta+\Lambda) \leq \#(\Delta)+\#(\Lambda) < \#(\mathtt{x} : \mathtt{A}) + \#(\Lambda) = \#(\mathtt{x} : \mathtt{A} + \Lambda)$; otherwise, $\#(\Delta + \Lambda) \leq \#(\Delta) + \#(\Lambda) < (\#(\mathtt{x} : \mathtt{A}) - 1) + \#(\Lambda) = \#(\mathtt{x} : \mathtt{A} + \Lambda)$.

The following property follows directly.

Property 4.9. Let $(\Gamma,\mathtt{p}) \in \mathtt{P}_{\mathtt{dom}(\mathcal{V})}(\mathtt{A})$. Then $\#(\Gamma) \leq \#(\mathtt{A})$. Moreover, $\mathtt{p} = \langle \mathtt{p}_1, \mathtt{p}_2 \rangle$ implies $\#(\Gamma) + 1 < \#(\mathtt{A})$.

We can now prove:

Lemma 4.10. *Every run of the algorithm terminates.*

Proof. We associate a tree $\mathcal{T}$ to each call of the algorithm, where the nodes are labeled with elements in the set $\{\mathtt{T}(\_,\_), \mathtt{M}(\_,\_), \mathtt{H}^{\_}_{\_}(\_,\_) \triangleright \_, \mathtt{P}_{\_}(\_)\}$. A node n' is a son of n iff there exists some instance of a rule having n as conclusion and n' as premise. Thus, a run of the algorithm is encoded in the tree $\mathcal{T}$, which turns to be finitely branching. We now prove that the measure $\#(\_)$ strictly decreases along all the branches of $\mathcal{T}$, so that every branch has finite depth. We proceed by induction on the rules of the algorithm. The only interesting cases are rules ($\mathtt{Abs}$) and ($\mathtt{Subs}$).

- Consider rule ($\mathtt{Abs}$), with conclusion $\lambda\mathtt{p}.\mathtt{a} \in \mathtt{T}(\Gamma, \mathtt{A} \to \tau)$ and premises $\mathtt{a} \in \mathtt{T}(\Gamma + \Delta, \tau)$ and $(\Delta, \mathtt{p}) \in \mathtt{P}_{\mathtt{dom}(\Gamma)}(\mathtt{A})$. By Property 4.9, $\#(\Delta) \leq \#(\mathtt{A})$, so that $\#(\mathtt{T}(\Gamma + \Delta, \tau)) \leq \#(\Gamma) + \#(\Delta) + \#(\tau) \leq \#(\Gamma) + \#(\mathtt{A}) + \#(\tau) < \#(\mathtt{T}(\Gamma, \mathtt{A} \to \tau))$ and $\#(\mathtt{P}_{\mathtt{dom}(\Gamma)}(\mathtt{A})) = \#(\mathtt{A}) < \#(\mathtt{T}(\Gamma, \mathtt{A} \to \tau))$.
- Consider rule ($\mathtt{Subs}$), with conclusion $\mathtt{b}[\langle \mathtt{p}, \mathtt{q}\rangle/\mathtt{c}] \in \mathtt{T}(\Gamma + \Lambda + \mathtt{x} : [\sigma], \tau)$ and premises $\mathtt{c} \in \mathtt{H}^{\mathtt{x}:[\sigma]}_{\mathtt{x}}(\Gamma, \sigma) \triangleright \mathtt{F}(\sigma)$, $(\Delta, \langle \mathtt{p}, \mathtt{q}\rangle) \in \mathtt{P}_{\mathtt{dom}(\Gamma+\Lambda+(\mathtt{x}:[\sigma]))}([\mathtt{F}(\sigma)])$ and $\mathtt{b} \in \mathtt{T}(\Delta + \Lambda, \tau)$. Clearly $\#(\mathtt{H}^{\mathtt{x}:[\sigma]}_{\mathtt{x}}(\Gamma, \sigma) \triangleright \mathtt{F}(\sigma)) = \#(\Gamma) + \#(\sigma) < \#(\Gamma) + \#(\Lambda) + \#([\sigma]) + \#(\tau) = \#(\mathtt{T}(\Gamma + \Lambda + \mathtt{x} : [\sigma], \tau))$. Also, $\#(\mathtt{P}_{\mathtt{dom}(\Gamma+\Lambda+(\mathtt{x}:[\sigma]))}([\mathtt{F}(\sigma)])) = \#([\mathtt{F}(\sigma)]) \leq \#([\sigma]) < \#(\mathtt{T}(\Gamma + \Lambda + \mathtt{x} : [\sigma], \tau))$. Finally, by Property 4.9, $\#(\Delta) + 1 < \#([\mathtt{F}(\sigma)]) \leq \#([\sigma]) = \#(\mathtt{x} : [\sigma])$. So $\#(\mathtt{T}(\Delta + \Lambda, \tau)) = \#(\Delta + \Lambda) + \#(\tau) < \#(\mathtt{x} : [\sigma] + \Lambda) + \#(\tau) \leq \#(\Gamma + \mathtt{x} : [\sigma] + \Lambda) + \#(\tau) = \#(\mathtt{T}(\Gamma + \Lambda + \mathtt{x} : [\sigma], \tau))$.

So every branch has finite depth. Hence, $\mathcal{T}$ is finite by König's Lemma, *i.e.* the algorithm terminates. □

In order to complete the proof of termination we need to show that the number of different run of the algorithm on any given input is finite.

Let $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$, where, by α-conversion, we assume that $\mathtt{fv}(\mathtt{a}) \cap \mathtt{bv}(\mathtt{a}) = \emptyset$. We write $||\mathtt{a}||^{\Pi}_{\mathtt{x}}$ (resp $|\mathtt{a}|^{\Pi}_{\mathtt{x}}$) to denote the number of free (resp. bound) occurrences of $\mathtt{x}$ in $\mathtt{a}$ which are typed in Π. The following property holds[2].

Property 4.11. Let $\mathtt{a}$ be an approximate normal form. Let $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$. Then, for every variable $\mathtt{x}$ occurring in $\mathtt{a}$ we have $||\mathtt{a}||^{\Pi}_{\mathtt{x}} \leq \#(\Gamma(\mathtt{x}))$ and $|\mathtt{a}|^{\Pi}_{\mathtt{x}} \leq \#(\Gamma) + \#(\sigma)$.

Proof. If $\mathtt{x} \in \mathtt{fv}(\mathtt{a})$, then $\mathtt{x} : [\sigma_i]_{i \in I} \in \Gamma$, for some non-empty set I, and since every axiom corresponds to a free occurrence of $\mathtt{x}$ in $\mathtt{a}$ which is typed in Π, then the number of such occurrences is exactly the cardinality of I, which is trivially smaller than $\#(\Gamma(\mathtt{x}))$.

Let $\mathtt{x} \in \mathtt{bv}(\mathtt{a})$. The proof is by induction on Π.

- Let the last rule of Π be ($\mathtt{abs}$), with conclusion $\Pi \triangleright \Gamma' \setminus \Gamma'|_{\mathtt{p}} \vdash \lambda\mathtt{p}.\mathtt{b} : \mathtt{A} \to \tau$ and premises $\Pi' \triangleright \Gamma' \vdash \mathtt{b} : \tau$ and $\Gamma'|_{\mathtt{p}} \Vdash \mathtt{p} : \mathtt{A}$, where $\Gamma = \Gamma' \setminus \Gamma'|_{\mathtt{p}}$ and $\sigma = \mathtt{A} \to \tau$. Since $\mathtt{x}$ is bound in $\lambda\mathtt{p}.\mathtt{b}$, then either $\mathtt{x}$ is bound in $\mathtt{b}$ or $\mathtt{x}$ occurs in the pattern $\mathtt{p}$. If $\mathtt{x}$ is bound in $\mathtt{b}$, then the proof follows by induction. Otherwise, by Lemma 3.4, $\Gamma'|_{\mathtt{x}} \Vdash \mathtt{x} : \mathtt{A}^{\mathtt{p}}_{\mathtt{x}}$, *i.e.* $\Gamma'(\mathtt{x}) = \mathtt{A}^{\mathtt{p}}_{\mathtt{x}}$. Then the number of free occurrences of $\mathtt{x}$ in $\mathtt{b}$ typed in Π' is $||\mathtt{b}||^{\Pi'}_{\mathtt{x}} \leq \#(\mathtt{A}^{\mathtt{p}}_{\mathtt{x}})$, so the number of its bound occurrences in $\lambda\mathtt{p}.\mathtt{b}$ typed in Π is the same. Since $\#(\mathtt{A}^{\mathtt{p}}_{\mathtt{x}}) \leq \#(\mathtt{A})$ and $\#(\mathtt{A}) \leq \#(\mathtt{A} \to \tau) + \#(\Gamma)$, then we are done.
- Let the last rule of Π be ($\mathtt{sub}$), with conclusion $\Gamma' \setminus \Gamma'|_{\langle \mathtt{p},\mathtt{q}\rangle} + \Delta \vdash \mathtt{b}[\langle \mathtt{p}, \mathtt{q}\rangle/\mathtt{c}] : \sigma$ and premises $\Pi_{\mathtt{b}} \triangleright \Gamma' \vdash \mathtt{b} : \sigma$, $\Gamma'|_{\langle \mathtt{p},\mathtt{q}\rangle} \vdash \langle \mathtt{p}, \mathtt{q}\rangle : [\times(\mathtt{A}, \mathtt{B})]$, and $\Pi_{\mathtt{c}} \triangleright \Delta \vdash \mathtt{c} : [\times(\mathtt{A}, \mathtt{B})]$, where $\mathtt{a} = \mathtt{b}[\langle \mathtt{p}, \mathtt{q}\rangle/\mathtt{c}]$ and $\Gamma = \Gamma' \setminus \Gamma'|_{\langle \mathtt{p},\mathtt{q}\rangle} + \Delta$. Since $\mathtt{x}$ is bound in $\mathtt{b}[\langle \mathtt{p}, \mathtt{q}\rangle/\mathtt{c}]$, then either $\mathtt{x}$ is bound in $\mathtt{b}$ or $\mathtt{c}$, or $\mathtt{x}$ occurs in $\langle \mathtt{p}, \mathtt{q}\rangle$ and so $\mathtt{x}$ is free in $\mathtt{b}$. If $\mathtt{x}$ is bound in $\mathtt{b}$ or $\mathtt{c}$, then the proof follows by induction. Let $\mathtt{x}$ occur in $\langle \mathtt{p}, \mathtt{q}\rangle$. Since $\mathtt{x}$ occurs free in $\mathtt{b}$, by the *i.h.*, $||\mathtt{b}||^{\Pi_{\mathtt{b}}}_{\mathtt{x}} \leq \#(\Gamma'(\mathtt{x}))$. Notice that, by definition of approximant, $\mathtt{c}$ must be an $\mathcal{L}$ approximate normal form, so that let $\mathtt{c} = \mathtt{y}\mathtt{a}_1...\mathtt{a}_n$ ($n \geq 0$). The derivation $\Pi_{\mathtt{c}}$ has necessarily been obtained by applying rule ($\mathtt{many}$) to a derivation $\Pi'_{\mathtt{c}} \triangleright \Delta \vdash \mathtt{c} : \times(\mathtt{A}, \mathtt{B})$, so $\Delta(\mathtt{y})$ must be $\mathtt{C} + [\mathtt{A}_1 \to \ldots \mathtt{A}_n \to \times(\mathtt{A}, \mathtt{B})]$, for some $\mathtt{C}, \mathtt{A}_1, \ldots, \mathtt{A}_n$ such that $\Delta_i \vdash \mathtt{a}_i : \mathtt{A}_i$, and $\Delta = +_{i\in I}\Delta_i + \mathtt{y} : [\mathtt{A}_1 \to \ldots \mathtt{A}_n \to \times(\mathtt{A}, \mathtt{B})]$, where $(+_{i\in I}\Delta_i)(y) = \mathtt{C}$. So $\#(\Delta) \geq \#([\times(\mathtt{A}, \mathtt{B})])$. By the *i.h.* $||\mathtt{c}||^{\Pi'_{\mathtt{c}}}_{\mathtt{y}} = ||\mathtt{c}||^{\Pi_{\mathtt{c}}}_{\mathtt{y}} \leq \#(\Delta(\mathtt{y}))$.

[2]Tighter upper bounds than those provided below may be found, but this is inessential here.

Moreover, by Lemma 3.4 $\Gamma'(\mathtt{x}) = [\times(\mathtt{A},\mathtt{B})]_{\mathtt{x}}^{\langle \mathtt{p},\mathtt{q}\rangle}$. Then the number of free occurrences of $\mathtt{x}$ in $\mathtt{b}$ typed in $\Pi_{\mathtt{b}}$ is $||\mathtt{b}||_{\mathtt{x}}^{\Pi_{\mathtt{b}}} = \#([\times(\mathtt{A},\mathtt{B})]_{\mathtt{x}}^{\langle \mathtt{p},\mathtt{q}\rangle}) \leq \#([\times(\mathtt{A},\mathtt{B})]) \leq \#(\Delta) \leq \#(\Delta) + \#(\sigma)$. We conclude since $|\mathtt{a}|_{\mathtt{x}}^{\Pi} = ||\mathtt{b}||_{\mathtt{x}}^{\Pi_{\mathtt{b}}}$.

- All other cases follow easily by induction. □

This property has an important corollary.

Corollary 4.12. *Given a pair* (Γ,σ)*, the number of approximate normal forms* $\mathtt{a}$ *such that* $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$ *and* $\mathtt{a} = \mathcal{A}(\Pi)$ *is finite.*

Proof. Let $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$. By Property 4.11, the number of typed occurrences of every variable in Π is bounded by $\#(\Gamma) + \#(\sigma)$ (we suppose each variable to be either bound or free, but not both, in $\mathcal{A}(\Pi)$, by α-conversion). So the total number of typed occurrences of variables in Π is bounded by an integer, let say $\mathcal{B}$. By definition of $\mathcal{A}(\Pi)$, Ω is the only untyped subterm of $\mathtt{a}$, then $\mathcal{B}$ is an upper bound for the number of *all* occurrences of variables of $\mathtt{a}$, which turn out to be all typed occurrences of variables of $\mathtt{a}$ in Π.

It is easy to see that $\mathcal{B}$ is also a bound for the total number of axioms of each derivation $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$, so the number of such derivations is finite and the conclusion follows. □

Now we are able to complete the termination proof.

Theorem 4.6. *The inhabitation algorithm terminates.*

Proof. Lemma 4.7 ($\Rightarrow$) ensures that the outputs of the inhabitation algorithm, called on (Γ,σ), are all of the form $\mathtt{a} = \mathcal{A}(\Pi)$ for some $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$. By Corollary 4.12 there exist finitely many such $\mathtt{a}$'s, and by Lemma 4.10 producing any of these takes a finite number of steps. Altogether, the inhabitation algorithms always terminates. □

Soundness and Completeness. In order to show Lemma 4.7 we first introduce the following key notion. A derivation Π is a **left-subtree** of a derivation Σ if either $\Pi = \Sigma$, or $\Pi \triangleright \Delta \vdash \mathtt{u} : \sigma$ is the major premise of some derivation $\Sigma' \triangleright \Delta' \vdash \mathtt{uv} : \tau$, such that Σ' is a left-subtree of Σ.

Property 4.13. $(\Gamma,\mathtt{p}) \in \mathtt{P}_{\mathtt{dom}(\mathcal{V})}(\mathtt{A})$, if and only if there is a derivation $\Gamma \Vdash \mathtt{p} : \mathtt{A}$, such that $\mathtt{fv}(\mathtt{p}) \cap \mathcal{V} = \emptyset$.

Proof. Easy, by checking the rules. □

Lemma 4.7. $\mathtt{a} \in \mathtt{T}(\Gamma,\sigma) \Leftrightarrow \exists \Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$ such that $\mathtt{a} = \mathcal{A}(\Pi)$.

Proof. ($\Rightarrow$)**:** We prove by mutual induction the following statements:

a) $\mathtt{a} \in \mathtt{T}(\Gamma,\sigma) \Rightarrow \exists \Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$ such that $\mathtt{a} = \mathcal{A}(\Pi)$.
b) $\mathtt{a} \in \mathtt{M}(\Gamma,\mathtt{A}) \Rightarrow \exists \Pi \triangleright \Gamma \vdash \mathtt{a} : \mathtt{A}$ such that $\mathtt{a} = \mathcal{A}(\Pi)$.
c) $\mathtt{a} \in \mathtt{H}_{\mathtt{b}}^{\Delta}(\Gamma,\sigma) \triangleright \tau \Rightarrow$ if $\exists \Sigma \triangleright \Delta \vdash \mathtt{b} : \sigma$ such that $\mathtt{b} = \mathcal{A}(\Sigma)$, then $\exists \Pi \triangleright \Gamma + \Delta \vdash \mathtt{a} : \tau$ such that $\mathtt{a} = \mathcal{A}(\Pi)$.

Each statement is proved by induction on the rules in Figure 2.

a) • Let the last rule be ($\mathtt{Abs}$), with conclusion $\lambda \mathtt{p}.\mathtt{a} \in \mathtt{T}(\Gamma, \mathtt{A} \to \tau)$ and premises $\mathtt{a} \in \mathtt{T}(\Gamma + \Delta, \tau)$ and $(\Delta,\mathtt{p}) \in \mathtt{P}_{\mathtt{dom}(\Gamma)}(\mathtt{A})$. By Property 4.13, there is a derivation $\Delta \Vdash \mathtt{p} : \mathtt{A}$, and we conclude by the *i.h.* (a) on $\mathtt{a} \in \mathtt{T}(\Gamma + \Delta, \tau)$ and the typing rule ($\mathtt{abs}$).

- Let the last rule be $(\texttt{Head})$, with conclusion $\texttt{a} \in \texttt{T}(\Gamma' + \texttt{x} : [\tau], \sigma)$ and premise $\texttt{a} \in \texttt{H}_{\texttt{x}}^{\texttt{x}:[\tau]}(\Gamma', \tau) \triangleright \sigma$, where $\Gamma = \Gamma' + \texttt{x} : [\tau]$. Then consider the derivation $\Sigma \triangleright \texttt{x} : [\tau] \vdash \texttt{x} : \tau$ where $\texttt{x} = \mathcal{A}(\Sigma)$. The *i.h.* (c) provides $\Pi \triangleright \Gamma' + \texttt{x} : [\tau] \vdash \texttt{a} : \sigma$ such that $\texttt{a} = \mathcal{A}(\Pi)$.
- Let the last rule be $(\texttt{Prod})$, with conclusion $\langle \texttt{a}, \texttt{b} \rangle \in \texttt{T}(\Gamma_0 + \Gamma_1, \times(\texttt{A}, \texttt{B}))$ and premises $\texttt{a} \in \texttt{M}(\Gamma_0, \texttt{A})$ and $\texttt{b} \in \texttt{M}(\Gamma_1, \texttt{B})$, where $\Gamma = \Gamma_0 + \Gamma_1$ and $\sigma = \times(\texttt{A}, \texttt{B})$. Then we conclude by the *i.h.* (b) and the typing rule $(\texttt{pair})$.
- Let the last rule be $(\texttt{Subs})$, with conclusion $\texttt{b}[\langle \texttt{p}, \texttt{q} \rangle / \texttt{c}] \in \texttt{T}(\Gamma_0 + \Gamma_1 + \texttt{x} : [\tau], \sigma)$ and premises $\texttt{c} \in \texttt{H}_{\texttt{x}}^{\texttt{x}:[\tau]}(\Gamma_0, \tau) \triangleright \texttt{F}(\tau)$, $(\Delta, \langle \texttt{p}, \texttt{q} \rangle) \in \texttt{P}_{\texttt{dom}(\Gamma_0 + \Gamma_1 + x:[\tau])}([\texttt{F}(\tau)])$ and $\texttt{b} \in \texttt{T}(\Gamma_1 + \Delta, \sigma)$, where $\Gamma = \Gamma_0 + \Gamma_1 + \texttt{x} : [\tau]$. Since there is $\Sigma \triangleright \texttt{x} : [\tau] \vdash \texttt{x} : \tau$, by *i.h.* (c) on $\texttt{c} \in \texttt{H}_{\texttt{x}}^{\texttt{x}:[\tau]}(\Gamma_0, \tau) \triangleright \texttt{F}(\tau)$ there is Ψ' s.t. $\Psi' \triangleright \Gamma_0 + \texttt{x} : [\tau] \vdash \texttt{c} : \texttt{F}(\tau)$, and $\texttt{c} = \mathcal{A}(\Psi')$. Moreover, by rule $(\texttt{many})$ we obtain $\Psi \triangleright \Gamma_0 + \texttt{x} : [\tau] \vdash \texttt{c} : [\texttt{F}(\tau)]$. By Property 4.13, $(\Delta, \langle \texttt{p}, \texttt{q} \rangle) \in \texttt{P}_{\texttt{dom}(\Gamma_0 + \Gamma_1 + x:[\tau])}([\texttt{F}(\tau)])$ implies there is $\Psi'' \triangleright \Delta \Vdash \langle \texttt{p}, \texttt{q} \rangle : [\texttt{F}(\tau)]$ and $\texttt{fv}(\langle \texttt{p}, \texttt{q} \rangle) \cap \texttt{dom}(\Gamma_0 + \Gamma_1 + \texttt{x} : [\tau]) = \emptyset$. Now, by applying the *i.h.* (a) to $\texttt{b} \in \texttt{T}(\Gamma_1 + \Delta, \sigma)$, we get a derivation $\Pi' \triangleright \Gamma_1 + \Delta \vdash \texttt{b} : \sigma$ such that $\texttt{b} = \mathcal{A}(\Pi')$. We get the required proof Π by using the typing rule $(\texttt{sub})$ on the premises Ψ, Ψ'' and Π'.

b) Let the last rule be $(\texttt{Many})$ with conclusion $\texttt{a} \in \texttt{M}(+_{i \in I} \Gamma_i, [\sigma_i]_{i \in I})$ and premises $(\texttt{a}_i \in \texttt{T}(\Gamma_i, \sigma_i))_{i \in I}$ and $\uparrow_{i \in I} \texttt{a}_i$. The proof follows from the *i.h.* (a) and then the typing rule $(\texttt{many})$ or the (new) typing rule (Ω).

c)
- Let the last rule be $(\texttt{Final})$, with conclusion $\texttt{a} \in \texttt{H}_{\texttt{a}}^{\Delta}(\emptyset, \sigma) \triangleright \tau$ and premise $\sigma = \tau$. Suppose $\Sigma \triangleright \Delta \vdash \texttt{a} : \sigma$. The fact that the there exists a derivation $\Delta + \emptyset \vdash \texttt{a} : \sigma$ is then straightforward.
- Let the last rule be $(\texttt{Prefix})$, with conclusion $\texttt{a} \in \texttt{H}_{\texttt{c}}^{\Delta}(\Gamma_0 + \Gamma_1, \texttt{A} \to \sigma') \triangleright \tau$ and premises $\texttt{b} \in \texttt{M}(\Gamma_0, \texttt{A})$ and $\texttt{a} \in \texttt{H}_{\texttt{cb}}^{\Delta + \Gamma_0}(\Gamma_1, \sigma') \triangleright \tau$, where $\sigma = \texttt{A} \to \sigma'$. Suppose that there exists a derivation $\Sigma \triangleright \Delta \vdash \texttt{c} : \texttt{A} \to \sigma'$ such that $\texttt{c} = \mathcal{A}(\Sigma)$. The *i.h.* (b) applied to $\texttt{b} \in \texttt{M}(\Gamma_0, \texttt{A})$ provides a derivation $\Psi \triangleright \Gamma_0 \vdash \texttt{b} : \texttt{A}$, where $\texttt{b} = \mathcal{A}(\Psi)$. The typing rule $(\texttt{app})$ with premises Σ and Ψ gives a derivation $\Pi' \triangleright \Delta + \Gamma_0 \vdash \texttt{cb} : \sigma'$, such that $\texttt{cb} = \mathcal{A}(\Pi')$. Then, the *i.h.* (d) applied to $\texttt{a} \in \texttt{H}_{\texttt{cb}}^{\Delta + \Gamma_0}(\Gamma_1, \sigma') \triangleright \tau$ provides a derivation $\Pi \triangleright \Delta + \Gamma_0 + \Gamma_1 \vdash \texttt{a} : \tau$ such that $\texttt{a} = \mathcal{A}(\Pi)$, as required.

($\Leftarrow$): We prove by mutual induction the following statements:

a) Given $\Sigma \triangleright \Delta \vdash \texttt{b} : \tau$ and $\Pi \triangleright \Gamma \vdash \texttt{a} : \sigma$, if $\texttt{b} = \mathcal{A}(\Sigma)$ and $\texttt{a} = \mathcal{A}(\Pi)$ are $\mathcal{L}$-anfs, and Σ is a left-subtree of Π, then there exists Γ' s.t. $\Gamma = \Gamma' + \Delta$ and for every Θ, $\texttt{H}_{\texttt{a}}^{\Delta + \Gamma'}(\Theta, \sigma) \triangleright \rho \subseteq \texttt{H}_{\texttt{b}}^{\Delta}(\Theta + \Gamma', \tau) \triangleright \rho$.

b) $\Pi \triangleright \Gamma \vdash \texttt{a} : \sigma$ and $\texttt{a} = \mathcal{A}(\Pi)$ imply $\texttt{a} \in \texttt{T}(\Gamma, \sigma)$.

Each statement is proved by induction on the definition of approximate normal forms.

a)
- If $\texttt{a} = \texttt{x}$, then Π is an axiom $(\texttt{ax})$; Σ being a left subtree of Π, we get $\Sigma = \Pi$, $\texttt{b} = \texttt{x}$, $\Gamma' = \emptyset$, $\sigma = \tau$ and the inclusion $\texttt{H}_{\texttt{a}}^{\Delta + \Gamma'}(\Theta, \sigma) \triangleright \rho \subseteq \texttt{H}_{\texttt{b}}^{\Delta}(\Theta + \Gamma', \tau) \triangleright \rho$ trivially holds.
- If $\texttt{a} = \texttt{ca}'$, $\texttt{c}$ being an $\mathcal{L}$-anf, then the last rule of Π is an instance of $(\texttt{app})$, with premises $\Pi_1 \triangleright \Gamma_1 \vdash \texttt{c} : \texttt{A} \to \sigma$ and $\Pi_2 \triangleright \Gamma_2 \vdash \texttt{a}' : \texttt{A}$, so that $\Gamma' = \Gamma_1 + \Gamma_2$. Moreover, $\Sigma \triangleright \Delta \vdash \texttt{b} : \tau$ is also a left-subtree of Π_1 and Π_2 comes from $(\Pi_2^i \triangleright \Gamma_2^i \vdash \texttt{a}' : \sigma_i)_{i \in I}$, where $+_{i \in I} \Gamma_2^i = \Gamma_2$ and $[\sigma_i]_{i \in I} = \texttt{A}$. We have in

this case $\mathtt{a}' = \bigvee_{i\in I} \mathcal{A}(\Pi_2^i)$, where by the *i.h.* (b), $\mathcal{A}(\Pi_2^i) \in \mathtt{T}(\Gamma_2^i, \sigma_i)$. Then $\mathtt{H}_{\mathtt{ca}'}^{\Delta+\Gamma_1+\Gamma_2}(\Theta,\sigma)\triangleright\rho \subseteq_{(\mathtt{Prefix})} \mathtt{H}_{\mathtt{c}}^{\Delta+\Gamma_1}(\Theta+\Gamma_2, \mathtt{A}\to\sigma)\triangleright\rho \subseteq_{(a)} \mathtt{H}_{\mathtt{b}}^{\Delta}(\Theta+\Gamma_1+\Gamma_2,\tau)\triangleright\rho$.

- Since by hypothesis both $\mathtt{a}$ and $\mathtt{b}$ are $\mathcal{L}$-anf*s*, there are no other cases.

b)
- $\mathtt{a} = \Omega$ does not apply, since σ is not the empty multiset.
- If $\mathtt{a}$ is an $\mathcal{L}$-anf, then $\exists \mathtt{x}, \tau$ s.t. $\Gamma = \Gamma_0 + \mathtt{x} : [\tau]$ and the type derivation $\Sigma \triangleright \mathtt{x} : [\tau] \vdash \mathtt{x} : \tau$ is a left subtree of $\Pi \triangleright \Gamma_0 + \mathtt{x} : [\tau] \vdash \mathtt{a} : \sigma$. Then we have $\mathtt{a} \in \mathtt{H}_{\mathtt{a}}^{\Gamma}(\emptyset,\sigma)\triangleright\sigma$ by rule $(\mathtt{Final})$, $\mathtt{H}_{\mathtt{a}}^{\Gamma_0+\mathtt{x}:[\tau]}(\emptyset,\sigma)\triangleright\sigma \subseteq \mathtt{H}_{\mathtt{x}}^{\mathtt{x}:[\tau]}(\Gamma_0,\tau)\triangleright\sigma$ by Point (a) and $\mathtt{H}_{\mathtt{x}}^{\mathtt{x}:[\tau]}(\Gamma_0,\tau)\triangleright\sigma \subseteq \mathtt{T}(\Gamma_0+\mathtt{x}:[\tau],\sigma)$ by rule $(\mathtt{Head})$. We thus conclude $\mathtt{a} \in \mathtt{T}(\Gamma_0 + \mathtt{x} : [\tau], \sigma)$.
- Otherwise, we analyze all the other cases of $\mathcal{N}$-anf*s*.
  - If $\mathtt{a} = \lambda\mathtt{p}.\mathtt{b}$ (resp. $\mathtt{a} = \langle \mathtt{b}, \mathtt{c}\rangle$) then it is easy to conclude by induction, using rule $\mathtt{Abs}$ (resp. $\mathtt{Prod}$).
  - If $\mathtt{a} = \mathtt{c}[\langle \mathtt{p},\mathtt{q}\rangle/\mathtt{b}]$, then $\mathtt{c}$ (resp. $\mathtt{b}$) is an $\mathcal{N}$ (resp. $\mathcal{L}$)-approximate normal form. By construction, Π is of the following form:

$$\dfrac{\Pi' \triangleright \Gamma \vdash \mathtt{c} : \sigma \qquad \Gamma|_{\langle \mathtt{p},\mathtt{q}\rangle} \Vdash \langle \mathtt{p},\mathtt{q}\rangle : [\times(\mathtt{A}_1,\mathtt{A}_2)] \qquad \dfrac{\Psi' \triangleright \Delta \vdash \mathtt{b} : \times(\mathtt{A}_1,\mathtt{A}_2)}{\Psi \triangleright \Delta \vdash \mathtt{b} : [\times(\mathtt{A}_1,\mathtt{A}_2)]}}{\Gamma \setminus \Gamma|_{\langle \mathtt{p},\mathtt{q}\rangle} + \Delta \vdash \mathtt{c}[\langle \mathtt{p},\mathtt{q}\rangle/\mathtt{b}] : \sigma}$$

By definition $\mathcal{A}(\Pi) = \mathcal{A}(\Pi')[\langle \mathtt{p},\mathtt{q}\rangle/\mathcal{A}(\Psi')]$. By the *i.h.* (c) $\mathcal{A}(\Pi') \in \mathtt{T}(\Gamma,\sigma)$. Moreover, $\mathtt{b} = \mathtt{y}\mathtt{c}_1 \ldots \mathtt{c}_h$ $(h \geq 0)$, since it is an $\mathcal{L}$-canonical form, so that $\Delta = \Delta' + (\mathtt{y} : [\tau])$ where $\mathtt{F}(\tau) = \times(\mathtt{A}_1,\mathtt{A}_2)$ and $\Sigma \triangleright \mathtt{y} : [\tau] \vdash \mathtt{y} : \tau$ is a left prefix of $\Psi' \triangleright \Delta \vdash \mathtt{b} : \times(\mathtt{A}_1,\mathtt{A}_2)$. Therefore, $\mathcal{A}(\Psi) = \mathcal{A}(\Psi')$ belongs to the set $\mathtt{H}_{\mathcal{A}(\Psi')}^{\Delta}(\emptyset,\mathtt{F}(\tau))\triangleright\mathtt{F}(\tau)$, which is included in $\mathtt{H}_{\mathtt{y}}^{\mathtt{y}:[\tau]}(\Delta',\tau)\triangleright\mathtt{F}(\tau)$ by Point (a). By Property 4.13 we also have $(\Gamma|_{\langle \mathtt{p},\mathtt{q}\rangle}, \langle \mathtt{p},\mathtt{q}\rangle) \in \mathtt{P}_{\mathtt{dom}(\Gamma\setminus\Gamma|_{\langle \mathtt{p},\mathtt{q}\rangle}+\Delta)}([\mathtt{F}(\tau)])$. We thus obtain $\mathcal{A}(\Pi) = \mathcal{A}(\Pi')[\langle \mathtt{p},\mathtt{q}\rangle/\mathcal{A}(\Psi)] \in \mathtt{T}(\Gamma \setminus \Gamma|_{\langle \mathtt{p},\mathtt{q}\rangle} + \Delta' + (\mathtt{y} : [\tau]), \sigma) = \mathtt{T}(\Gamma \setminus \Gamma|_{\langle \mathtt{p},\mathtt{q}\rangle} + \Delta, \sigma)$ by rule $(\mathtt{Subs})$. □

5. Characterizing Solvability

We are now able to state the main result of this paper, *i.e.* the characterization of the solvability property for the pattern calculus $\Lambda_{\mathtt{p}}$.

The logical characterization of canonical forms given in Section 3 through the type assignment system $\mathcal{P}$ is a first step in this direction. In fact, the system $\mathcal{P}$ is complete with respect to solvability, but it is not sound, as shown in the next theorem.

Theorem 5.1. *The set of solvable terms is a proper subset of the set of terms having canonical forms.*

Proof.

- (Solvability implies canonicity) If $\mathtt{t}$ is solvable, then there is a head context $\mathtt{H}$ such that $\mathtt{H}[\mathtt{t}]$ is closed and reduces to $\langle \mathtt{u},\mathtt{v}\rangle$, for some $\mathtt{u}$ and $\mathtt{v}$. Since all pairs are typable, the term $\mathtt{H}[\mathtt{t}]$ is typable by Lemma 3.10(2), so that $\mathtt{t}$ is typable too by Lemma 3.3. We conclude that $\mathtt{t}$ has canonical form by Theorem 3.11.
- (Canonicity does not imply solvability) Let $\mathtt{t}_1 = \lambda\mathtt{x}.\mathtt{I}[\langle \mathtt{y},\mathtt{z}\rangle/\mathtt{x}][\langle \mathtt{y}',\mathtt{z}'\rangle/\mathtt{x}\mathtt{I}]$. the term $\mathtt{t}_1$ is canonical, hence typable by Theorem 3.11. However $\mathtt{t}_1$ is not solvable. In fact, it is

easy to see that there is no term $\mathtt{u}$ such that both $\mathtt{u}$ and $\mathtt{uI}$ reduce to pairs. Indeed, let $\mathtt{u} \to^* \langle \mathtt{v}_1, \mathtt{v}_2 \rangle$; then $\mathtt{uI} \to^* \langle \mathtt{v}_1, \mathtt{v}_2 \rangle \mathtt{I}$, which will reduce to $\mathtt{fail}$. □

However, as explained in the introduction, we can use inhabitation of system $\mathcal{P}$ to completely characterize solvability. The following lemma guarantees that the types reflect correctly the structure of the data.

Lemma 5.2. *Let* $\mathtt{t}$ *be a closed and typable term.*

- *If* $\mathtt{t}$ *has functional type, then* $\mathtt{t}$ *reduces to an abstraction.*
- *If* $\mathtt{t}$ *has product type, then it reduces to a pair.*

Proof. Let $\mathtt{t}$ be a closed and typable term. By Theorem 3.11 we know that $\mathtt{t}$ reduces to a (closed) canonical form in $\mathcal{J}$. The proof is by induction on the maximal length of such reduction sequences. If $\mathtt{t}$ is already a canonical form, we analyze all the cases.

- If $\mathtt{t}$ is a variable, then this gives a contradiction with $\mathtt{t}$ closed.
- If $\mathtt{t}$ is an abstraction, then the property trivially holds.
- If $\mathtt{t}$ is a pair, then the property trivially holds.
- If $\mathtt{t}$ is an application, then $\mathtt{t}$ necessarily has a head (free) variable which belongs to the set of free variables of $\mathtt{t}$, which leads to a contradiction with $\mathtt{t}$ closed.
- If $\mathtt{t} = \mathtt{u}[\langle \mathtt{p}_1, \mathtt{p}_2 \rangle / \mathtt{v}]$ is closed, then in particular $\mathtt{v}$ is closed, which leads to a contradiction with $\mathtt{t} \in \mathcal{J}$ implying $\mathtt{v} \in \mathcal{K}$. So this case is not possible.

Otherwise, there is a reduction sequence $\mathtt{t} \to \mathtt{t}' \to^* \mathtt{u}$, where $\mathtt{u}$ is in $\mathcal{J}$. The term $\mathtt{t}'$ is also closed and typable by Lemma 3.10(1), then the *i.h.* gives the desired result for $\mathtt{t}'$, so the property holds also for $\mathtt{t}$. □

The notion of inhabitation can easily be extended to typing environments, by defining Γ inhabited if $\mathtt{x} : \mathtt{C} \in \Gamma$ implies $\mathtt{C}$ is inhabited. The following lemma shows in particular that if the type of a pattern is inhabited, then its typing environment is also inhabited.

Lemma 5.3.

(1) *If* $\Pi \triangleright \Gamma \Vdash \mathtt{p} : \mathtt{A}$ *and* $\mathtt{A}$ *is inhabited, then* Γ *is also inhabited.*
(2) *If* $\Gamma \vdash \mathtt{t} : \mathtt{A}$ *and* Γ *is inhabited, then* $\mathtt{A}$ *is inhabited.*

Proof.

(1) The proof is by induction on $\mathtt{p}$.
   - If $\mathtt{p} = \mathtt{x}$ then Γ is $\mathtt{x} : \mathtt{A}$ with $\mathtt{A} \neq [\,]$ or it is $\emptyset$. In both cases the property is trivial.
   - If $\mathtt{p} = \langle \mathtt{p}_1, \mathtt{p}_2 \rangle$, then $\Pi_1 \triangleright \Gamma_1 \Vdash \mathtt{p}_1 : \mathtt{A}_1$ and $\Pi_2 \triangleright \Gamma_2 \Vdash \mathtt{p}_2 : \mathtt{A}_2$, where $\mathtt{A} = [\times(\mathtt{A}_1, \mathtt{A}_2)]$ and $\Gamma = \Gamma_1 + \Gamma_2$. Let us see that $\mathtt{A}_i$ $(i = 1, 2)$ is inhabited. Since $[\times(\mathtt{A}_1, \mathtt{A}_2)]$ is inhabited, so is $\times(\mathtt{A}_1, \mathtt{A}_2)$. By Lemma 5.2, the closed term $\mathtt{t}$ inhabiting $\times(\mathtt{A}_1, \mathtt{A}_2)$ reduces to a pair $\langle \mathtt{t}_1, \mathtt{t}_2 \rangle$. We know by Lemma 3.10(1) that $\vdash \langle \mathtt{t}_1, \mathtt{t}_2 \rangle : \times(\mathtt{A}_1, \mathtt{A}_2)$, and we conclude that $\vdash \mathtt{t}_i : A_i$ $(i = 1, 2)$. Now, by applying the *i.h.* to Π_1 (resp. Π_2) we have that for every $\mathtt{x} : \mathtt{A}' \in \Gamma_1$ (resp Γ_2), A' is inhabited. By linearity of $\mathtt{p}$, if $\mathtt{x} : \mathtt{A}' \in \Gamma$ then either $\mathtt{x} : \mathtt{A}' \in \Gamma_1$ or $\mathtt{x} : \mathtt{A}' \in \Gamma_2$ (otherwise stated: $\Gamma_1 + \Gamma_2 = \Gamma_1; \Gamma_2$). Hence we conclude that for every $\mathtt{x} : \mathtt{A}' \in \Gamma$, $\mathtt{A}'$ is inhabited.

(2) For all $\mathtt{x} : \mathtt{B} \in \Gamma$, let $\mathtt{u}_\mathtt{x}$ be a closed term inhabiting $\mathtt{B}$. By Lemma 3.9(1) the closed term obtained by replacing in $\mathtt{t}$ all occurrences of $\mathtt{x} \in \mathtt{dom}(\Gamma)$ by $\mathtt{u}_\mathtt{x}$ inhabits $\mathtt{A}$. □

In order to simplify the following proofs, let us introduce a new notation: let $\vec{\mathtt{A}}$ denote a sequence of multiset types $\mathtt{A}_1, ..., \mathtt{A}_n$, so that $\mathtt{A}_1 \to ... \to \mathtt{A}_n \to \sigma$ will be abbreviated by

$\vec{\mathtt{A}} \to \sigma$. Note that every type has this structure, for some multisets $\mathtt{A}_1, \ldots, \mathtt{A}_n$ ($n \geq 0$) and type σ. Moreover we will say that $\vec{\mathtt{A}}$ is inhabited if all its components are inhabited.

Lemma 5.4. *Let* $\mathtt{H}[\mathtt{t}]$ *be such that* $\Pi \triangleright \Gamma \vdash \mathtt{H}[\mathtt{t}] : \vec{\mathtt{A}} \to \pi$*, where* Γ *and* $\vec{\mathtt{A}}$ *are inhabited. Then there are* $\Pi', \Gamma', \vec{\mathtt{C}}$ *such that* $\Pi' \triangleright \Gamma' \vdash \mathtt{t} : \vec{\mathtt{C}} \to \pi$ *where* Γ' *and* $\vec{\mathtt{C}}$ *are inhabited.*

Proof. By induction on $\mathtt{H}$.

- If $\mathtt{H} = \square$, then the property trivially holds.
- If $\mathtt{H} = \mathtt{H}'\ \mathtt{u}$, then $\Gamma = \Gamma' + \Delta$ and Π is:

$$\dfrac{\Gamma' \vdash \mathtt{H}'[\mathtt{t}] : \mathtt{B} \to \vec{\mathtt{A}} \to \pi \qquad \Delta \vdash \mathtt{u} : \mathtt{B}}{\Gamma' + \Delta \vdash \mathtt{H}'[\mathtt{t}]\mathtt{u} : \vec{\mathtt{A}} \to \pi}\ (\mathtt{app})$$

  Γ' and Δ are inhabited, being sub-environments of Γ. By Lemma 5.3(2) $\mathtt{B}$ is inhabited. Then the proof follows by the *i.h.* on the major premise.
- If $\mathtt{H} = \lambda\mathtt{p}.\mathtt{H}'$ then $\Gamma = \Gamma' \setminus \Gamma'|_{\mathtt{p}}$, $\vec{\mathtt{A}} = \mathtt{A}_0, \vec{\mathtt{A}'}$ and Π is:

$$\dfrac{\Gamma' \vdash \mathtt{H}'[\mathtt{t}] : \vec{\mathtt{A}'} \to \pi \qquad \Gamma'|_{\mathtt{p}} \Vdash \mathtt{p} : \mathtt{A}_0}{\Gamma' \setminus \Gamma'|_{\mathtt{p}} \vdash \lambda\mathtt{p}.\mathtt{H}'[\mathtt{t}] : \mathtt{A}_0 \to \vec{\mathtt{A}'} \to \pi}\ (\mathtt{abs})$$

  Since $\mathtt{A}_0$ is inhabited, Lemma 5.3(1) ensures that $\Gamma'|_{\mathtt{p}}$ (and thus Γ') is inhabited, too. The proof follows by the *i.h.* on the major premise.
- If $\mathtt{H} = \mathtt{H}'[\mathtt{p}/\mathtt{u}]$ then $\Gamma = \Gamma' \setminus \Gamma'|_{\mathtt{p}} + \Delta$ and Π is:

$$\dfrac{\Gamma' \vdash \mathtt{H}'[\mathtt{t}] : \vec{\mathtt{A}} \to \pi \qquad \Gamma'|_{\mathtt{p}} \Vdash \mathtt{p} : \mathtt{B} \qquad \Delta \vdash \mathtt{u} : \mathtt{B}}{(\Gamma' \setminus \Gamma'|_{\mathtt{p}}) + \Delta \vdash \mathtt{H}'[\mathtt{t}][\mathtt{p}/\mathtt{u}] : \vec{\mathtt{A}} \to \pi}\ (\mathtt{sub})$$

  Δ is inhabited, being a sub-environment of Γ. By Lemma 5.3(2) $\mathtt{B}$ is inhabited. Hence by Lemma 5.3(1) $\Gamma'|_{\mathtt{p}}$ (and thus Γ') is inhabited. Then the proof follows by the *i.h.* on the major premise. □

Theorem 5.5 (Characterizing Solvability)**.** *A term* $\mathtt{t}$ *is solvable iff* $\Pi \triangleright \Gamma \vdash \mathtt{t} : \vec{\mathtt{C}} \to \sigma$*, where* σ *is a product type and* Γ *and* $\vec{\mathtt{C}}$ *are inhabited.*

Proof.

- (only if) If $\mathtt{t}$ is solvable, then there exists a head context $\mathtt{H}$ such that $\mathtt{H}[\mathtt{t}]$ is closed and $\mathtt{H}[\mathtt{t}] \to^* \langle \mathtt{u}, \mathtt{v} \rangle$. By subject expansion $\vdash \mathtt{H}[\mathtt{t}] : o$. Then Lemma 5.4 allows to conclude.
- (if) Let $\Gamma = \mathtt{x}_1 : \mathtt{A}_1, \ldots, \mathtt{x}_k : \mathtt{A}_k$ ($k \geq 0$) and $\vec{\mathtt{C}} = \mathtt{C}_1, \ldots, \mathtt{C}_m$ ($m \geq 0$). By hypothesis there exist closed terms $\mathtt{u}_1, \ldots \mathtt{u}_k, \mathtt{v}_1, \ldots \mathtt{v}_m$ such that $\emptyset \vdash \mathtt{u}_i : \mathtt{A}_i$ and $\emptyset \vdash \mathtt{v}_j : \mathtt{C}_j$ ($1 \leq i \leq k, 1 \leq j \leq m$). Let $\mathtt{H} = (\lambda\mathtt{x}_k \ldots ((\lambda\mathtt{x}_1.\square)\mathtt{u}_1) \ldots \mathtt{u}_k)\mathtt{v}_1 \ldots \mathtt{v}_m$ be a head context. We have $\mathtt{H}[\mathtt{t}]$ closed and $\emptyset \vdash \mathtt{H}[\mathtt{t}] : \sigma$, where σ is a product type. This in turn implies that $\mathtt{H}[\mathtt{t}]$ reduces to a pair, by Lemma 5.2. Then the term $\mathtt{t}$ is solvable by definition. □

Our notion of solvability is conservative with respect to that of the λ-calculus.

Theorem 5.6 (Conservativity)**.** *A* λ*-term* $\mathtt{t}$ *is solvable in the* λ*-calculus if and only if* $\mathtt{t}$ *is solvable in the* $\Lambda_{\mathtt{p}}$*-calculus.*

Proof.

- (if) Let $\mathtt{t}$ be a λ-term which is not solvable, *i.e.* which does not have head normal-form. Then $\mathtt{t}$ (seen as a term of our calculus) has no canonical form, and thus $\mathtt{t}$ is not typable by Theorem 3.11. It turns out that $\mathtt{t}$ is not solvable in $\Lambda_{\mathtt{p}}$ by Theorem 5.5.

- (only if) Let $\mathtt{t}$ be a solvable λ-term so that there exist a head context $\mathtt{H}$ such that $\mathtt{H}[\mathtt{t}]$ is closed and reduces to $\mathtt{I}$, then it is easy to construct a head context $\mathtt{H}'$ such that $\mathtt{H}'[\mathtt{t}]$ reduces to a pair (just take $\mathtt{H}' = \mathtt{H}\ \langle \mathtt{t}_1, \mathtt{t}_2 \rangle$ for some terms $\mathtt{t}_1, \mathtt{t}_2$). □

6. Conclusion and Further Work

We extend the classical notion of solvability, originally stated for the λ-calculus, to a pair pattern calculus. We provide a logical characterization of solvable terms by means of typability *and* inhabitation.

An interesting question concerns the consequences of changing non-idempotent types to idempotent ones in our typing system $\mathcal{P}$. Characterization of solvability will still need the two ingredients typability and inhabitation, however, inhabitation will become undecidable, in contrast to our decidable inhabitation problem for the non-idempotent system $\mathcal{P}$. This is consistent with the fact that the inhabitation problem for the λ-calculus is undecidable for idempotent types [27], but decidable for the non-idempotent ones [10].

Notice however that changing the meta-level substitution operator to explicit substitutions would not change neither the notion nor the characterization of solvability: all the explicit substitutions are fully computed in normal forms.

Further work will be developed in different directions. As we already discussed in Section 2, different definitions of solvability would be possible, as for example in [17]. We explored the one based on a lazy semantics, but it would be also interesting to obtain a full characterization based on a strict semantics.

On the semantical side, it is well known that non-idempotent intersection types can be used to supply a logical description of the relational semantics of λ-calculus [15, 23]. We would like to start from our type assignment system for building a denotational model of the pattern calculus. Last but not least, a challenging question is related to the characterization of solvability in a more general framework of pattern λ-calculi allowing the patterns to be dynamic [19].

Acknowledgment

We are grateful to Sandra Alves and Daniel Ventura for fruitful discussions. This work was partially done within the framework of ANR COCA HOLA (16-CE40-004-01).

References

[1] B. Accattoli and D. Kesner. The structural *lambda*-calculus. In A. Dawar and H. Veith, editors, *Proceedings of 24th EACSL Conference on Computer Science Logic*, volume 6247 of *LNCS*, pages 381–395. Springer, Aug. 2010.

[2] S. Alves, D. Kesner, and D. Ventura. A quantitative understanding of pattern matching, 2019. https://arxiv.org/abs/1912.01914.

[3] T. Balabonski. On the implementation of dynamic patterns. In E. Bonelli, editor, *HOR*, volume 49 of *EPTCS*, pages 16–30, 2010.

[4] T. Balabonski, P. Barenbaum, E. Bonelli, and D. Kesner. Foundations of strong call by need. *PACMPL*, 1(ICFP):20:1–20:29, 2017.

[5] P. Barenbaum, E. Bonelli, and K. Mohamed. Pattern matching and fixed points: Resource types and strong call-by-need: Extended abstract. In D. Sabel and P. Thiemann, editors, *Proceedings of the 20th International Symposium on Principles and Practice of Declarative Programming, PPDP 2018, Frankfurt am Main, Germany, September 03-05, 2018*, pages 6:1–6:12. ACM, 2018.

[6] H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in logic and the foundation of mathematics*. North-Holland, Amsterdam, revised edition, 1984.

[7] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *The Journal of Symbolic Logic*, 48(4):931–940, 1983.

[8] C. Ben-Yelles. *Type-assignment in the lambda-calculus; syntax and semantics*. PhD thesis, University of Wales Swansea, 1979.

[9] A. Bernadet. *Types intersections non-idempotents pour raffiner la normalisation forte avec des informations quantitatives*. PhD thesis, École Polytechnique, France, 2014.

[10] A. Bucciarelli, D. Kesner, and S. Ronchi Della Rocca. The inhabitation problem for non-idempotent intersection types. In *TCS*, volume 8705 of *LNCS*, pages 341–354. Springer, 2014.

[11] A. Bucciarelli, D. Kesner, and S. Ronchi Della Rocca. Inhabitation for non-idempotent intersection types. *Logical Methods in Computer Science*, 14(3), 2018.

[12] A. Bucciarelli, D. Kesner, and Simona Ronchi Della Rocca. Observability for pair pattern calculi. In *TLCA*, volume 38 of *LIPIcs*, pages 123–137, 2015.

[13] S. Cerrito and D. Kesner. Pattern matching as cut elimination. *Theoretical Computer Science*, 323(1-3):71–127, 2004.

[14] H. Cirstea, G. Faure, and C. Kirchner. A rho-calculus of explicit constraint application. *Higher-Order and Symbolic Computation*, 20(1-2):37–72, 2007.

[15] D. de Carvalho. Execution time of lambda-terms via denotational semantics and intersection types. *CoRR*, abs/0905.4251, 2009.

[16] M. Dezani-Ciancaglini and S. Ghilezan. Two behavioural lambda models. In H. Geuvers and F. Wiedijk, editors, *TYPES*, volume 2646 of *LNCS*, pages 127–147. Springer, 2002.

[17] Á. García-Pérez and P. Nogueira. No solvable lambda-value term left behind. *Logical Methods in Computer Science*, 12(2), 2016.

[18] J. R. Hindley. *Basic Simple Type Theory*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Amsterdam, 2008.

[19] B. Jay and D. Kesner. First-class patterns. *Journal of Functional Programming*, 19(2):191–225, 2009.

[20] W. Kahl. Basic pattern matching calculi: A fresh view on matching failure. In Y. Kameyama and P. Stuckey, editors, *FLOPS*, volume 2998 of *LNCS*, pages 276–290. Springer, 2004.

[21] J.-W. Klop, V. van Oostrom, and R. de Vrijer. Lambda calculus with patterns. *Theoretical Computer Science*, 398(1-3):16–31, 2008.

[22] J. L. Krivine. *Lambda-Calculus, Types and Models*. Masson, Paris, and Ellis Horwood, Hemel Hempstead, 1993.

[23] L. Paolini, M. Piccolo, and S. Ronchi Della Rocca. Essential and relational models. *Mathematical Structures in Computer Science*, 27(5):626–650, 2017.

[24] B. Petit. A polymorphic type system for the lambda-calculus with constructors. In *TLCA*, volume 5608 of *lncs*, pages 234–248. Springer, 2009.

[25] S. Peyton-Jones. *The Implementation of Functional Programming Languages*. Prentice-Hall, Inc., 1987.

[26] D. S. Scott. Outline of a mathematical theory of computation. In *Fourth Annual Princeton Conference on Information Sciences and Systems*, pages 169–176. Departement of Elctrical Engeneering, Princeton Univ., 1970.

[27] P. Urzyczyn. The emptiness problem for intersection types. *Journal of Symbolic Logic*, 64(3):1195–1215, 1999.

[28] V. van Oostrom. Confluence by decreasing diagrams. *Theoretical Computer Science*, 126(2):259–280, 1994.