The Inhabitation Problem for Non-idempotent Intersection Types

Antonio Bucciarelli, Delia Kesner, Simona Ronchi Della Rocca

The inhabitation problem for non-idempotent intersection types

Antonio Bucciarelli[1], Delia Kesner[1], and Simona Ronchi Della Rocca[2]

[1] Univ Paris Diderot, Sorbonne Paris Cit, PPS, UMR 7126, CNRS, Paris, France
[2] Dipartimento di Informatica, Università di Torino, Italy

Abstract. The inhabitation problem for intersection types is known to be undecidable. We study the problem in the case of non-idempotent intersection, and we prove decidability through a sound and complete algorithm. We then consider the inhabitation problem for an extended system typing the λ-calculus with pairs, and we prove the decidability in this case too. The extended system is interesting in its own, since it allows to characterize solvable terms in the λ-calculus with pairs.

1 Introduction

Intersection types have been presented in the literature in many variants. Historically, one of the first versions is the one characterizing solvable terms, that we call system $\mathcal{C}$, shown in Fig. 1 [6, 15]. Intersection enjoys associativity, commutativity, and in particular idempotency ($\mathtt{A} \wedge \mathtt{A} = \mathtt{A}$). Given a type $\mathtt{A}$ and a typing environment Γ, the problem of deciding whether there exists a term $\mathtt{t}$ such that $\Gamma \vdash \mathtt{t} : \mathtt{A}$ is provable, is known in the literature both as *emptiness problem* and as *inhabitation problem*. The inhabitation problem for system $\mathcal{C}$ has been proved to be undecidable by Urzyczyn [21]. Van Bakel [23] simplified the system, using strict types, where intersection is not allowed on the right side of the arrow; his system $\mathcal{S}$ is presented on the left part of Fig. 2, where intersection is naturally represented through set formation, and the universal type ω by the empty set. The right part of the figure presents the relevant version of $\mathcal{S}$, $\mathcal{S}_r$, a system being relevant if and only if, in its provable judgments, typing environments only contain the consumed premises. The systems $\mathcal{C}$, $\mathcal{S}$ and $\mathcal{S}_r$ are equivalent with respect to the typability power (neglecting the universal type ω). Urzyczyn's proof of undecidability of the inhabitation problem for system $\mathcal{C}$ can be easily adapted to system $\mathcal{S}$. Moreover, the inhabitation problem for $\mathcal{C}$ seems to reduce to that for $\mathcal{S}_r$, proving that the latter is undecidable too [22], so that relevance has nothing to do with the hardness of the inhabitation problem.

$$\frac{\mathtt{x}:\mathtt{A}\in\Gamma}{\Gamma\vdash\mathtt{x}:\mathtt{A}}(\mathtt{var}) \qquad \frac{}{\Gamma\vdash\mathtt{t}:\omega}(\omega) \qquad \frac{\Gamma,\mathtt{x}:\mathtt{A}\vdash\mathtt{t}:\mathtt{B}}{\Gamma\vdash\lambda\mathtt{x}.\mathtt{t}:\mathtt{A}\to\mathtt{B}}(\to\mathtt{I})$$

$$\frac{\Gamma\vdash\mathtt{t}:\mathtt{A}\to\mathtt{B}\quad\Gamma\vdash\mathtt{u}:\mathtt{A}}{\Gamma\vdash\mathtt{tu}:\mathtt{B}}(\to\mathtt{E}) \quad \frac{\Gamma\vdash\mathtt{t}:\mathtt{A}\quad\Gamma\vdash\mathtt{t}:\mathtt{B}}{\Gamma\vdash\mathtt{t}:\mathtt{A}\wedge\mathtt{B}}(\wedge\mathtt{I}) \quad \frac{\Gamma\vdash\mathtt{t}:\mathtt{A}_1\wedge\mathtt{A}_2}{\Gamma\vdash\mathtt{t}:\mathtt{A}_i\ (i=1,2)}(\wedge\mathtt{E})$$

Types: $\mathtt{A}::=\alpha\mid\omega\mid\mathtt{A}\to\mathtt{A}\mid\mathtt{A}\wedge\mathtt{A}$ Typing environments: $\Gamma::=\emptyset\mid\Gamma,\mathtt{x}:\mathtt{A}\quad\mathtt{x}\notin\mathtt{dom}(\Gamma)$

Fig. 1: System $\mathcal{C}$

$$\frac{\mathtt{x}:\mathtt{A}\in\Gamma\qquad\sigma\in\mathtt{A}}{\Gamma\vdash\mathtt{x}:\sigma}(\mathtt{var}) \qquad \frac{}{\mathtt{x}:\{\sigma\}\vdash\mathtt{x}:\sigma}(\mathtt{var_r})$$

$$\frac{\Gamma,\mathtt{x}:\mathtt{A}\vdash\mathtt{t}:\tau}{\Gamma\vdash\lambda\mathtt{x}.\mathtt{t}:\mathtt{A}\to\tau}(\to\mathtt{I}) \qquad \frac{\Gamma,\mathtt{x}:\mathtt{A}\vdash\mathtt{t}:\tau}{\Gamma\vdash\lambda\mathtt{x}.\mathtt{t}:\mathtt{A}\to\tau}(\to\mathtt{I})$$

$$\frac{\Gamma\vdash\mathtt{t}:\{\sigma_i\}_{i\in I}\to\tau\quad(\Gamma\vdash\mathtt{u}:\sigma_i)_{i\in I}}{\Gamma\vdash\mathtt{tu}:\tau}(\to\mathtt{E}) \quad \frac{\Gamma\vdash\mathtt{t}:\{\sigma_i\}_{i\in I}\to\tau\quad(\Delta_i\vdash\mathtt{u}:\sigma_i)_{i\in I}}{\Gamma\cup_{i\in I}\Delta_i\vdash\mathtt{tu}:\tau}(\to\mathtt{E_r})$$

Types:
$\sigma::=a\mid\mathtt{A}\to\sigma$ (strict types)
$\mathtt{A}::=\emptyset\mid\{\sigma\}\mid\mathtt{A}\cup\mathtt{A}$ (set types)

Typing environments:
$\Gamma::=\emptyset\mid\Gamma,\mathtt{x}:\mathtt{A}\quad\mathtt{x}\notin\mathtt{dom}(\Gamma)$
$(\Gamma\cup\Delta)(\mathtt{x})=\Gamma(\mathtt{x})\cup\Delta(\mathtt{x})$

Fig. 2: Systems $\mathcal{S}$ and $\mathcal{S}_r$

In this paper we consider the type assignment system $\mathcal{M}$, which is a variant of $\mathcal{S}_r$, where idempotency of intersection has been removed, and we prove that its inhabitation is decidable, by exploiting the fact that in this case types keep track faithfully of the different uses of variables in terms. System $\mathcal{M}$ characterizes terms having head normal form, so we design a sound and complete algorithm, that, given a typing environment Γ and a type σ, builds a set of approximate normal forms from which all and only the head normal forms $\mathtt{t}$ such that $\Gamma\vdash\mathtt{t}:\sigma$ can be generated. Then we extend the system, and consequently the language, in order to consider pairs and projections. We obtain a new system $\mathcal{P}$ which characterizes the (suitably defined) solvability in the extended calculus Λ_π, and we prove that inhabitation is decidable also for this extension.

In the last years, growing interest has been devoted to non idempotent intersection types, since they allow to reason about quantitative properties of terms, both from a syntactical and a semantic point of view. In fact, system $\mathcal{M}$ is not new: it is the well known system R of De Carvalho [5], and it is an instance of the class of the essential λ-models defined in [20], which supplies a logical description of the strongly linear relational λ-models. Some other type assignment systems with non-idempotent intersection have been studied in the literature, for various purposes: to

compute a bound for the normalization time of terms [7]), to supply new characterizations of strong normalization [3, 12], to study type inference [14, 17], to study linearity [13], to characterize solvability in the resource λ-calculus [19, 18]. Moreover intersection without idempotency, commutativity nor associativity, has been used to study the game semantics of a typed λ-calculus [8]. A unified model-theoretic approach covering both the relevant and non-relevant cases, and unveiling the relations between them, is presented in [9]. Returning to the inhabitation problem, various restrictions of the classical intersection types system have been shown to have decidable inhabitation [16, 4]. The approach is substantially different from that used in this work, since in all cases intersection is idempotent, and the decidability is obtained by restricting the use of rules ($\wedge\mathtt{I}$) and ($\wedge\mathtt{E}$), so that the corresponding type assignment does not characterize interesting classes of terms, anymore.

2 The type assignment system $\mathcal{M}$

In this section we consider a relevant type system for the λ-calculus having strict intersection types that enjoy associativity and commutativity, but not idempotency. In order to emphasize this last property we represent intersections as multisets of types.

We recall that terms and contexts of the λ-calculus are generated by the following grammars, respectively:

$$\mathtt{t}, \mathtt{u}, \mathtt{v} ::= \mathtt{x} \mid \lambda\mathtt{x}.\mathtt{t} \mid \mathtt{t}\mathtt{u} \qquad\qquad \mathtt{C} ::= \Box \mid \lambda\mathtt{x}.\mathtt{C} \mid \mathtt{C}\mathtt{t} \mid \mathtt{t}\mathtt{C}$$

Given a context $\mathtt{C}$ and a term $\mathtt{t}$, $\mathtt{C}[\mathtt{t}]$ denotes the term obtained by replacing the unique occurrence of $\Box$ in $\mathtt{C}$ by $\mathtt{t}$, allowing the capture of free variables of $\mathtt{t}$. The β-reduction is given by the rule $(\lambda\mathtt{x}.\mathtt{t})\mathtt{u} \to_\beta \mathtt{t}\{\mathtt{u}/\mathtt{x}\}$ where $\mathtt{t}\{\mathtt{u}/\mathtt{x}\}$ denotes the capture free replacement of $\mathtt{x}$ by $\mathtt{u}$ in $\mathtt{t}$.

Let us recall the notion of head-normal forms ($\mathtt{hnf}$), which is the syntactical counter part of the well known notion of solvability for the λ-calculus. A λ-term *is in* $\mathtt{hnf}$ if it is generated by the following grammar $\mathcal{J}$, it *has* $\mathtt{hnf}$ if it β-reduces to a $\mathtt{hnf}$.

$$\mathcal{J} ::= \lambda\mathtt{x}.\mathcal{J} \mid \mathcal{K} \qquad\qquad \mathcal{K} ::= \mathtt{x} \mid \mathcal{K}\mathtt{t}$$

Definition 1.

1. *The set $\mathcal{TM}$ of types is defined by the following grammar:*

$$\begin{array}{ll} \sigma, \tau, \rho ::= \alpha \mid \mathtt{A} \to \tau & (\textit{types}) \\ \mathtt{A} \quad ::= [\sigma_i]_{i \in I} & (\textit{multiset types}) \end{array}$$

where α ranges over a countable set of base types and I is a finite, possibly empty, set of indices.

2. *Typing environements, or simply environments, written Γ, Δ, 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 $[\,]$.*
3. *A typing judgement is a triple of the form $\Gamma \vdash \mathtt{t} : \mathtt{A}$. The type system $\mathcal{M}$ is given in Fig. 3. If Π is derivation with conclusion $\Gamma \vdash \mathtt{t} : \sigma$ we write $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$, and call $\mathtt{t}$ the subject of Π. The measure of a derivation Π, written $\mathtt{meas}(\Pi)$, is the number of rule applications in Π. By abuse of notation, $\Gamma \vdash \mathtt{t} : \sigma$ also denotes the existence of some derivation with conclusion $\Gamma \vdash \mathtt{t} : \sigma$.*
4. *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 $\Sigma' \triangleright \Delta +_{i\in I} \Delta_i \vdash \mathtt{uv} : \tau$ and Σ' is a left-subtree of Σ.*

Given $\{\Gamma_i\}_{i\in I}$, $+_{i\in I}\Gamma_i$ is the environment mapping $\mathtt{x}$ to $\uplus_{i\in I}\Gamma_i(\mathtt{x})$, where $\uplus$ denotes multiset union, where the resulting environment is the one having empty domain for $I = \emptyset$. The notations $\Gamma + \Delta$ and $\Gamma +_{i\in I} \Delta_i$ are just particular cases of the previous one. $\Gamma \setminus \mathtt{x}$ is the environment assigning $[\,]$ to $\mathtt{x}$, and acting as Γ otherwise; $\mathtt{x}_1 : \mathtt{A}_1, \ldots, \mathtt{x}_n : \mathtt{A}_n$ is the environment assigning $\mathtt{A}_i$ to $\mathtt{x}_i$, for $1 \leq i \leq n$, and $[\,]$ to any other variable.

$$\frac{}{\mathtt{x} : [\rho] \vdash \mathtt{x} : \rho}\ (\mathtt{var}) \qquad \frac{\Gamma \vdash \mathtt{t} : \rho}{\Gamma \setminus \mathtt{x} \vdash \lambda\mathtt{x}.\mathtt{t} : \Gamma(\mathtt{x}) \to \rho}\ (\to \mathtt{I})$$

$$\frac{\Gamma \vdash \mathtt{t} : [\sigma_i]_{i\in I} \to \rho \qquad (\Delta_i \vdash \mathtt{u} : \sigma_i)_{i\in I}}{\Gamma +_{i\in I} \Delta_i \vdash \mathtt{tu} : \rho}\ (\to \mathtt{E})$$

Fig. 3: The type assignment system $\mathcal{M}$ for the λ-calculus

Rule $(\to \mathtt{E})$ enables the typability of non strongly normalizing terms, when $I = \emptyset$. For example $\mathtt{x} : [[\,] \to \alpha] \vdash \mathtt{x}((\lambda\mathtt{y}.\mathtt{yy})(\lambda\mathtt{y}.\mathtt{yy})) : \alpha$, where the outermost application is typed neglecting its unsolvable argument. This feature is shared by all the intersection type systems characterizing solvability. The same holds for the fundamental *subject reduction* property: if $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$ and $\mathtt{t} \to_\beta \mathtt{u}$, then $\Pi' \triangleright \Gamma \vdash \mathtt{u} : \sigma$. What is peculiar to $\mathcal{M}$ is the fact that the size of Π' is strictly smaller than that of Π, whenever the reduction $\mathtt{t} \to_\beta \mathtt{u}$ takes place in an occurrence of $\mathtt{t}$ which is *typed*

in Π. This property, stated in Thm. 1.1 below, allows for an original, combinatorial proof of the fact that typed terms do have $\mathtt{hnf}$.

Definition 2.

- *The set* $\mathtt{o}(t)$ *of occurrences of* $\mathtt{t}$ *is the set of contexts* $\mathtt{C}$ *such that there exists a term* $\mathtt{u}$ *verifying* $\mathtt{C}[\mathtt{u}] = \mathtt{t}$, $\mathtt{u}$ *being the subterm of* $\mathtt{t}$ *at the occurrence* $\mathtt{C}$.
- *Given* $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$, *the set* $\mathtt{to}(\Pi) \subseteq \mathtt{o}(\mathtt{t})$ *of typed occurrences of* $\mathtt{t}$ *in* Π *is defined by induction on* $\mathtt{meas}(\Pi)$ *as follows:*
  - $\mathtt{to}(\Pi) = \{\Box\}$ *if* Π *is an istance of the axiom.*
  - $\mathtt{to}(\Pi) = \{\Box\} \cup \{\lambda\mathtt{x}.\mathtt{C} \mid \mathtt{C} \in \mathtt{to}(\Pi')\}$ *if the subject of* Π *is* $\lambda\mathtt{x}.\mathtt{t}$ *and* Π' *is the subderivation of* Π *typing* $\mathtt{t}$.
  - $\mathtt{to}(\Pi) = \{\Box\} \cup \{\mathtt{Cu} \mid \mathtt{C} \in \mathtt{to}(\Pi')\} \cup (\bigcup_{i\in I}\{\mathtt{tC} \mid \mathtt{C} \in \mathtt{to}(\Pi'_i)\})$ *if the subject of* Π *is* $\mathtt{tu}$, Π' *is the subderivation of* Π *typing* $\mathtt{t}$, *and* Π'_i, *for* $i \in I$, *are the sub-derivations of* Π *typing* $\mathtt{u}$.
- *Given* $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$, *we say that* $\mathtt{t}$ *is in* Π*-normal form if for all* $\mathtt{C} \in \mathtt{to}(\Pi)$, *the subterm of* $\mathtt{t}$ *at the occurrence* $\mathtt{C}$ *is not a redex.*

Theorem 1.

1. *(Subject reduction)* $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$ *and* $\mathtt{t} \to_\beta \mathtt{u}$ *imply* $\Pi' \triangleright \Gamma \vdash \sigma$ *where* $\mathtt{meas}(\Pi') \leq \mathtt{meas}(\Pi)$. *Moreover, if the reduced redex is typed in* Π, *then* $\mathtt{meas}(\Pi') < \mathtt{meas}(\Pi)$.
2. *(Characterization)* $\Gamma \vdash \mathtt{t} : \sigma$ *if and only if* $\mathtt{t}$ *has* $\mathtt{hnf}$.

Proof. For 1 see [20] and for 2 see [5].

3 Inhabitation for system $\mathcal{M}$

The system $\mathcal{M}$ allows to type a term without giving types to all its subterms through the rule $(\to \mathtt{E})$ in case $I = \emptyset$. So in order to reconstruct the subject of a derivation we need a notation for these untyped subterms. We will use the standard notion of *approximate normal form* [1], which can be defined through the following grammar:

$$\mathtt{a}, \mathtt{b}, \mathtt{c} \ ::= \ \Omega \mid \mathcal{N} \qquad \mathcal{N} \ ::= \ \lambda\mathtt{x}.\mathcal{N} \mid \mathcal{L} \qquad \mathcal{L} \ ::= \ \mathtt{x} \mid \mathcal{L}\mathtt{a}$$

Approximate normal forms are ordered by the smallest contextual order $\leq$ such that $\Omega \leq \mathtt{a}$, for all $\mathtt{a}$. We write $\mathtt{a} \leq \mathtt{t}$ when the term $\mathtt{t}$ is obtained from $\mathtt{a}$ by replacing all the occurrences of Ω by terms.

Let $\mathcal{A}(\mathtt{t}) = \{\mathtt{a} \mid \exists \mathtt{u}\ \mathtt{t} \to_\beta^* \mathtt{u} \text{ and } \mathtt{a} \leq \mathtt{u}\}$ be the set of *approximants* of the λ-term $\mathtt{t}$, and let $\bigvee$ denote the least upper bound w.r.t. $\leq$. We write $\uparrow_{i\in I} \mathtt{a}_i$ to denote that $\bigvee\{\mathtt{a}_i\}_{i\in I}$ does exist. 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 approximate normal form $\mathtt{a}$ is a *head subterm* of $\mathtt{b}$ if either $\mathtt{b} = \mathtt{a}$ or $\mathtt{b} = \mathtt{cc}'$ and $\mathtt{a}$ is a head subterm of $\mathtt{c}$. System $\mathcal{M}$ gives types to approximate normal forms, by simply assuming that no type can be assigned to the constant Ω. 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 $\Pi \triangleright \Gamma \vdash \mathtt{t} : \tau$, where $\mathtt{t}$ is in Π-normal form, we denote by $\mathcal{A}(\Pi)$ the minimal approximant $\mathtt{b}$ of $\mathtt{t}$ such that $\Pi \triangleright \Gamma \vdash \mathtt{b} : \tau$. Formally,

Definition 3. *Given $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$, where $\mathtt{t}$ is in Π-normal form, $\mathcal{A}(\Pi) \in \mathcal{A}(\mathtt{t})$ is defined by induction on $\mathtt{meas}(\Pi)$ as follows:*

- *If $\Pi \triangleright \Gamma \vdash \mathtt{x} : \rho$, then $\mathcal{A}(\Pi) = \mathtt{x}$.*
- *If $\Pi \triangleright \Gamma \vdash \lambda\mathtt{x}.\mathtt{t} : \mathtt{A} \to \rho$ follows from $\Pi' \triangleright \Gamma, \mathtt{x} : \mathtt{A} \vdash \mathtt{t} : \rho$, then $\mathcal{A}(\Pi) = \lambda\mathtt{x}.\mathcal{A}(\Pi')$, $\mathtt{t}$ being in Π'-normal form.*
- *If $\Pi \triangleright \Gamma = \Gamma' +_{i\in I} \Delta_i \vdash \mathtt{tu} : \rho$ follows from $\Pi' \triangleright \Gamma \vdash \mathtt{t} : [\sigma_i]_{i\in I} \to \rho$ and $(\Pi'_i \triangleright \Delta_i \vdash \mathtt{u} : \sigma_i)_{i\in I}$, then $\mathcal{A}(\Pi) = \mathcal{A}(\Pi')(\bigvee_{i\in I} \mathcal{A}(\Pi'_i))$ (remark that $\mathtt{t}$ is in Π'-normal form, $\mathtt{u}$ if in Π'_i-normal form, for all $i \in I$, and that $\uparrow_{i\in I} \mathcal{A}(\Pi'_i)$, since $\mathcal{A}(\Pi'_i) \in \mathcal{A}(\mathtt{u})$, for all $i \in I$).*

Remark that, in the final item of the definition above, the approximate normal form corresponding to the case $I = \emptyset$ is $\mathcal{A}(\Pi')\Omega$.

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

Proposition 1. *If $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$ and $\mathtt{t}$ is in Π-normal form, then $\Pi \triangleright \Gamma \vdash \mathcal{A}(\Pi) : \sigma$.*

3.1 The inhabitation algorithm

The inhabitation rules are given in Fig. 4. The algorithm, given an environment Γ and a type σ, builds the set $\mathtt{T}(\Gamma, \sigma)$ containing *all* the approximate normal forms $\mathtt{a}$ such that there exists a derivation $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$, with $\mathtt{a} = \mathcal{A}(\Pi)$, then stops. The algorithm uses two auxiliary predicates, namely $\mathtt{TI}(\Gamma, [\sigma_i]_{i\in I})$ and $\mathtt{H}^{\Delta}_{\mathtt{a}}(\Gamma, \sigma) \triangleright \tau$. The set $\mathtt{TI}(\Gamma, [\sigma_i]_{i\in I})$ contains all the approximate normal forms $\mathtt{a} = \bigvee_{i\in I} \mathtt{a}_i$ such that $\Gamma = +_{i\in I}\Gamma_i$, $\mathtt{a}_i \in \mathtt{T}(\Gamma_i, \sigma_i)$ for all $i \in I$, and $\uparrow_{i\in I} \mathtt{a}_i$. Finally, $\mathtt{H}^{\Delta}_{\mathtt{a}}(\Gamma, \sigma) \triangleright \tau$ contains all the approximate normal forms $\mathtt{b}$ such that $\mathtt{a}$ is a head subterm of $\mathtt{b}$, and such that if $\mathtt{a} \in \mathtt{T}(\Delta, \sigma)$ then $\mathtt{b} \in \mathtt{T}(\Gamma + \Delta, \tau)$.

Notice the particular case $I = \emptyset$ in (`Union`), which gives $\Omega \in \mathtt{TI}(\emptyset, [\,])$, where $\emptyset$ denotes the environement having empty domain. The algorithm is

$$\frac{\mathtt{a} \in \mathtt{T}(\Gamma + (\mathtt{x} : \mathtt{A}), \tau) \qquad \mathtt{x} \notin \mathtt{dom}(\Gamma)}{\lambda \mathtt{x}.\mathtt{a} \in \mathtt{T}(\Gamma, \mathtt{A} \to \tau)}\ (\mathtt{Abs})$$

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

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

$$\frac{\Gamma = \Gamma_0 + \Gamma_1 \qquad \mathtt{b} \in \mathtt{TI}(\Gamma_0, \mathtt{A}) \qquad \mathtt{a} \in \mathtt{H}_{\mathtt{cb}}^{\Delta + \Gamma_0}(\Gamma_1, \sigma) \triangleright \tau}{\mathtt{a} \in \mathtt{H}_{\mathtt{c}}^{\Delta}(\Gamma, \mathtt{A} \to \sigma) \triangleright \tau}\ (\mathtt{Prefix})$$

Fig. 4: The inhabitation algorithm for the λ-calculus

not an obvious extension of the classical inhabitation algorithm for simple types [2, 10]. In particular, when restricted to simple types, it reconstructs all the normal forms inhabiting a given type, while the original algorithm reconstructs just the *long η-normal forms*. This is achieved thanks to a non deterministic behaviour, illustrated in the Example 1.1 below.

Example 1.

1. Let $\Gamma = \emptyset$ and $\sigma = [[\alpha] \to \alpha] \to [\alpha] \to \alpha$. Given input (Γ, σ), the algorithm can have the following two behaviours:
   (1) Choosing the sequence of rules: $(\mathtt{Abs}), (\mathtt{Abs}), (\mathtt{Head}), (\mathtt{Prefix}), (\mathtt{Final})$ the final approximant is $\lambda \mathtt{xy}.\mathtt{xy}$;
   (2) Choosing the sequence of rules: $(\mathtt{Abs}), (\mathtt{Head}), (\mathtt{Final})$ the final approximant is $\lambda \mathtt{x}.\mathtt{x}$.
2. Let $\Gamma = \emptyset$ and $\sigma = [[] \to \alpha] \to \alpha$. Given input (Γ, σ), by the sequence of rules: $(\mathtt{Abs}), (\mathtt{Head}), (\mathtt{Prefix}), ((\mathtt{Union}), (\mathtt{Final}))$ we obtain $\lambda \mathtt{x}.\mathtt{x}\Omega$.

Definition 4. *In order to show that the inhabitation algorithm terminates, we define a measure on types and environments, as follows:*

$$\#(\alpha) = 1 \qquad \#([\sigma_i]_{i \in I}) = \textstyle\sum_{i \in I} \#(\sigma_i) + 1$$
$$\#(\mathtt{A} \to \rho) = \#(\mathtt{A}) + \#(\rho) + 1 \qquad \#(\Gamma) = \textstyle\sum_{\mathtt{x} \in \mathtt{dom}(\Gamma)} \#(\Gamma(\mathtt{x}))$$

The measure is then be extended to the judgements of the algorithm:

$$\#(\mathtt{T}(\Gamma, \rho)) = \#(\mathtt{H}_{\mathtt{b}}^{\Delta}(\Gamma, \rho) \triangleright \tau) = \#(\Gamma) + \#(\rho)$$
$$\#(\mathtt{TI}(\Gamma, \mathtt{A})) = \#(\Gamma) + \#(A)$$

Lemma 1 (Termination). *The inhabitation algorithm terminates.*

Proof Hint. To each call $C \in \{\mathtt{T}(\_, \_), \mathtt{TI}(\_, \_), \mathtt{H}\_(\_, \_) \triangleright \_\}$ of the algorithm, we associate a tree T_C as follows: nodes are labeled with elements of C.

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, all possible runs of C are encoded in the tree T_C, which is finitely branching. Moreover, it is easy to see that the measure $\#()$ strictly decreases along the branches of T_C, so that every branch has finite depth. Hence, T_C is finite by König's Lemma, *i.e.* the algorithm terminates.

Soundness and completeness of the inhabitation algorithm follow from the following Lemma, relating typings of approximate normal forms in system $\mathcal{M}$ and runs of the algorithm:

Lemma 2. $\mathtt{a} \in \mathtt{T}(\Gamma, \sigma) \Leftrightarrow \exists \Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$ *such that* $\mathtt{a} = \mathcal{A}(\Pi)$.

Proof. ($\Rightarrow$): We prove the following statements, by induction on the rules in Fig. 4:

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) $\mathtt{a} \in \mathtt{TI}(+_{i\in I}\Gamma_i, [\sigma_i]_{i\in I}) \Rightarrow \exists (\Pi_i \triangleright \Gamma_i \vdash \mathtt{a}_i : \sigma_i)_{i\in I}$ such that $\mathtt{a}_i = \mathcal{A}(\Pi_i)$, for $i \in I$, $\uparrow_{i\in I} \mathtt{a}_i$ and $\mathtt{a} = \bigvee_{i\in I} \mathtt{a}_i$.
c) $\mathtt{a} \in \mathtt{H}^{\Delta}_{\mathtt{b}}(\Gamma, \sigma) \triangleright \tau \Rightarrow$ there exists a function F associating to each derivation $\Sigma \triangleright \Delta \vdash \mathtt{b} : \sigma$ such that $\mathtt{b} = \mathcal{A}(\Sigma)$, a derivation $\Pi \triangleright \Gamma + \Delta \vdash \mathtt{a} : \tau$ such that $\mathtt{a} = \mathcal{A}(\Pi)$.

If $\lambda \mathtt{x}.\mathtt{a} \in \mathtt{T}(\Gamma, \mathtt{A} \to \tau)$ follows from $\mathtt{a} \in \mathtt{T}(\Gamma + (\mathtt{x} : \mathtt{A}), \tau)$ by (`Abs`), then we conclude by the *i.h.*(a) and by an application of $(\to \mathtt{I})$.

If $\bigvee_{i\in I} \mathtt{a}_i \in \mathtt{TI}(+_{i\in I}\Gamma_i, [\sigma_i]_{i\in I})$ follows from $(\mathtt{a}_i \in \mathtt{T}(\Gamma_i, \sigma_i))_{i\in I}$ and $\uparrow_{i\in I} \mathtt{a}_i$ by (`Union`), then by *i.h.*(a), there exist $(\Pi_i \triangleright \Gamma_i \vdash \mathtt{a}_i : \sigma_i)_{i\in I}$ such that for all $i \in I$, $\mathtt{a}_i = \mathcal{A}(\Pi_i)$, and we are done.

If $\mathtt{a} \in \mathtt{T}(\Gamma + (\mathtt{x} : [\sigma]), \tau)$ follows from $\mathtt{a} \in \mathtt{H}^{\mathtt{x}:[\sigma]}_{\mathtt{x}}(\Gamma, \sigma) \triangleright \tau$ by (`Head`), then the i. h. (c) provides a function associating to each derivation $\Sigma \triangleright \mathtt{x} : [\sigma] \vdash \mathtt{x} : \sigma$ a derivation $\Pi \triangleright \Gamma + \mathtt{x} : [\sigma] \vdash \mathtt{a} : \tau$ such that $\mathtt{a} = \mathcal{A}(\Pi)$, since $\mathtt{x} = \mathcal{A}(\Sigma)$. Applying this function to the unique derivation of $\mathtt{x} : [\sigma] \vdash \mathtt{x} : \sigma$, we get the suitable typing of $\mathtt{a}$.

If $\mathtt{a} \in \mathtt{H}^{\Delta}_{\mathtt{a}}(\emptyset, \sigma) \triangleright \tau$ follows from $\sigma = \tau$ by (`Final`), then the identity function satisfies the requirements of (c).

Finally, if $\mathtt{a} \in \mathtt{H}^{\Delta}_{\mathtt{c}}(\Gamma_0 + \Gamma_1, \mathtt{A} \to \sigma) \triangleright \tau$ follows from $\mathtt{b} \in \mathtt{TI}(\Gamma_0, \mathtt{A})$ and $\mathtt{a} \in \mathtt{H}^{\Delta+\Gamma_0}_{\mathtt{cb}}(\Gamma_1, \sigma) \triangleright \tau$ by (`Prefix`), then we have to provide a function F associating to each derivation $\Sigma \triangleright \Delta \vdash \mathtt{c} : \mathtt{A} \to \sigma$ such that $\mathtt{c} = \mathcal{A}(\Sigma)$, a derivation $\Pi \triangleright \Delta + \Gamma_0 + \Gamma_1 \vdash \mathtt{a} : \tau$ such that $\mathtt{a} = \mathcal{A}(\Pi)$. To begin with, the *i.h.*(b) applied to $\mathtt{b} \in \mathtt{TI}(\Gamma_0, \mathtt{A})$ provides a family of derivations $(\Pi_i \triangleright \Gamma_0^i \vdash \mathtt{b}_i : \sigma_i)_{i\in I}$ such that $\mathtt{b}_i = \mathcal{A}(\Pi_i)$, $\Gamma_0 = +_{i\in I}\Gamma_0^i$, $A = [\sigma_i]_{i\in I}$, $\uparrow_{i\in I} \mathtt{b}_i$ and $\mathtt{b} = \bigvee_{i\in I} \mathtt{b}_i$. Rule $(\to \mathtt{E})$ with premises Σ and $\{\Pi_i\}_{i\in I}$, gives

a type derivation $\Pi' \triangleright \Delta + \Gamma_0 \vdash \mathtt{cb} : \sigma$, such that $\mathtt{cb} = \mathcal{A}(\Pi')$. Then, the *i.h.*(c) applied to $\mathtt{a} \in \mathtt{H}^{\Delta+\Gamma_0}_{\mathtt{cb}}(\Gamma_1, \sigma) \triangleright \tau$ provides a function F' such that $F'(\Pi') \triangleright \Delta + \Gamma_0 + \Gamma_1 \vdash \mathtt{a} : \tau$ and $\mathtt{a} = \mathcal{A}(F'(\Pi'))$.To conclude, we set $F(\Sigma) = F'(\Pi')$.

($\Leftarrow$): We prove the following statements, by induction on the definition of $\mathcal{A}(\Pi)$ (Def. 3):

1. Given $\Sigma \triangleright \Delta \vdash \mathtt{b} : \tau$ and $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$, if $\mathtt{b} = \mathcal{A}(\Sigma)$ and $\mathtt{a} = \mathcal{A}(\Pi)$ are $\mathcal{L}$-approximate normal forms, and Σ is a left-subtree of Π, then there exists Γ' s.t. $\Gamma = \Gamma' + \Delta$ and $\mathtt{H}^{\Delta+\Gamma'}_{\mathtt{a}}(\Theta, \sigma) \triangleright \rho \subseteq \mathtt{H}^{\Delta}_{\mathtt{b}}(\Theta + \Gamma', \tau) \triangleright \rho$.
2. $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$ and $\mathtt{a} = \mathcal{A}(\Pi)$ imply $\mathtt{a} \in \mathtt{T}(\Gamma, \sigma)$.

1. If $\mathtt{a} = \mathtt{x}$, then Π is an instance of the axiom (`var`); Σ being a left subtree of Π, we get $\Sigma = \Pi$, $\mathtt{b} = \mathtt{x}$, $\Gamma' = \emptyset$, $\sigma = \tau$ and the inclusion $\mathtt{H}^{\Delta+\Gamma'}_{\mathtt{b}}(\Theta, \sigma) \triangleright \rho \subseteq \mathtt{H}^{\Delta}_{\mathtt{a}}(\Theta + \Gamma', \tau) \triangleright \rho$ holds trivially.

If $\mathtt{a} = \mathtt{ca}'$, $\mathtt{c}$ being a $\mathcal{L}$-approximate normal form, then the last rule of Π is an instance of ($\to \mathtt{E}$), with premises $\Pi' \triangleright \Gamma'' \vdash \mathtt{c} : [\sigma_i]_{i\in I} \to \sigma$ and $(\Pi_i \triangleright \Gamma_i \vdash \mathtt{a}' : \sigma_i)_{i\in I}$, so that $\Gamma = \Gamma'' +_{i\in I} \Gamma_i$; moreover $\Sigma \triangleright \Delta \vdash \mathtt{b} : \tau$ is also a left-subtree of Π'. We have in this case $\mathtt{a}' = \bigvee_{i\in I} \mathcal{A}(\Pi_i)$, where by the *i.h.*(2), $\mathcal{A}(\Pi_i) \in \mathtt{T}(\Gamma_i, \sigma_i)$. Then $\mathtt{H}^{\Delta+\Gamma''+_{i\in I}\Gamma_i}_{\mathtt{ca}'}(\Theta, \sigma) \triangleright \rho \subseteq_{(\mathtt{Prefix})} \mathtt{H}^{\Delta+\Gamma''}_{\mathtt{c}}(\Theta +_{i\in I} \Gamma_i, [\sigma_i]_{i\in I} \to \sigma) \triangleright \rho \subseteq_{i.h.(1)} \mathtt{H}^{\Delta}_{\mathtt{b}}(\Theta + \Gamma'' +_{i\in I} \Gamma_i, \sigma) \triangleright \rho$.

2. If $\mathtt{a}$ is a $\mathcal{L}$-approximate normal form, then $\exists \tau$ s.t. $\Gamma = \Gamma_0 + (\mathtt{x} : [\tau])$ and the type derivation $\mathtt{x} : [\tau] \vdash \mathtt{x} : \tau$ is a left subtree of $\Gamma_0 + (\mathtt{x} : [\tau]) \vdash \mathtt{a} : \sigma$. Then we have $\mathtt{a} \in_{(\mathtt{Final})} \mathtt{H}^{\Gamma}_{\mathtt{a}}(\emptyset, \sigma) \triangleright \sigma \subseteq_{\mathrm{Point}(1)} \mathtt{H}^{\mathtt{x}:[\tau]}_{\mathtt{x}}(\Gamma_0, \tau) \triangleright \sigma \subseteq_{(\mathtt{Head})} \mathtt{T}(\Gamma_0 + \{\mathtt{x} : [\tau]\}, \sigma)$. Otherwise, $\mathtt{a} = \lambda \mathtt{x}.\mathtt{a}'$, and we conclude by the *i.h.*(2) on $\mathtt{a}'$.

Theorem 2 (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'*-normal form, and* $\mathcal{A}(\Pi') \in \mathtt{T}(\Gamma, \sigma)$.

Proof. Soundness follows from Lem. 2 ($\Rightarrow$) and the remark that, if $\Gamma \vdash \mathtt{a} : \sigma$ and $\mathtt{a} \leq \mathtt{t}$, then $\Gamma \vdash \mathtt{t} : \sigma$. Completeness follows from Thm. 1.1, ensuring that given $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$, there exists $\Pi' \triangleright \Gamma \vdash \mathtt{t}' : \sigma$ such that $\mathtt{t}'$ is in Π'-normal form, then from Prop. 1 and Lem. 2 ($\Leftarrow$).

4 Adding pairs and projections

The language Λ_π is an extension of λ-calculus with pairs and projections. Its terms and contexts are defined by the following grammars:

$$\begin{array}{ll} \mathtt{t},\mathtt{u},\mathtt{v} ::= & \mathtt{x} \mid \lambda\mathtt{x}.\mathtt{t} \mid \mathtt{t}\mathtt{u} \mid \pi_1\mathtt{t} \mid \pi_2\mathtt{t} \mid \langle \mathtt{t},\mathtt{u}\rangle \\ \mathtt{C} ::= & \Box \mid \lambda\mathtt{x}.\mathtt{C} \mid \mathtt{C}\mathtt{t} \mid \mathtt{t}\mathtt{C} \mid \pi_1\mathtt{C} \mid \pi_2\mathtt{C} \mid \langle \mathtt{t},\mathtt{C}\rangle \mid \langle \mathtt{C},\mathtt{t}\rangle \end{array}$$

The reduction relation, also denoted by $\to$, is the contextual closure of the following rules:

$(\lambda\mathtt{x}.\mathtt{t})\mathtt{u} \to_\beta \mathtt{t}\{\mathtt{u}/\mathtt{x}\} \qquad \pi_1\langle \mathtt{t},\mathtt{u}\rangle \to_\pi \mathtt{t} \qquad \pi_2\langle \mathtt{t},\mathtt{u}\rangle \to_\pi \mathtt{u}$

As usual, $\to^*$ denotes the reflexive-transitive closure of $\to$. We write $\overline{\mathtt{t}}$ and $\overline{\pi}$ (resp. $\overline{\mathtt{t}}_n$ and $\overline{\pi}_n$) to denote a possibly empty sequence (resp. a sequence of length n) of terms and projections, respectively.

Λ_π inherits from the λ-calculus important properties, such as confluence. Solvability is defined as usual, but pairs are solvable independently from their content, since we want to consider them as lazy data structures:

Definition 5.

1. *A head context is a context of the shape:* $(\lambda\overline{\mathtt{x}}.\Box)\overline{\mathtt{t}}$*;*
2. *A term* $\mathtt{t}$ *is solvable if and only if there is a head context* $\mathtt{C}$ *such that* $\mathtt{C}[\mathtt{t}] \to^* \mathtt{u}$*, where* $\mathtt{u}$ *is either a pair* $\langle \mathtt{u}_1,\mathtt{u}_2\rangle$ *or the identity* $\mathtt{I}$*.*

We will prove that solvability can be syntactically characterized by the notion of $\mathtt{hnf}$, defined by the following grammar:

$$\mathcal{J} ::= \lambda\mathtt{x}.\mathcal{J} \mid \langle t,t\rangle \mid \mathcal{P} \qquad\qquad \mathcal{P} ::= \mathtt{x} \mid \mathcal{P}\mathtt{t} \mid \pi_1\mathcal{P} \mid \pi_2\mathcal{P}$$

The *head variable* of a term in $\mathcal{P}$ is defined by: $\mathtt{x}$ is the head variable of $\mathtt{x}$ and $\mathtt{x}$ is the head variable of $\mathtt{u}\mathtt{t}$ (resp. $\pi_i\mathtt{u}$, $i = 1,2$) if $\mathtt{x}$ is the head variable of $\mathtt{u}$. A term *has* $\mathtt{hnf}$ if it reduces to a $\mathtt{hnf}$.

We will prove now that if a term has $\mathtt{hnf}$ then it is solvable. The converse will be proved through a suitable type assignment system, which will be introduced in the next subsection.

Lemma 3. *It* $\mathtt{t}$ *has* $\mathtt{hnf}$ *then it is solvable.*

Proof. Let us define, for every sequence of projections $\overline{\pi}$ and every term $\mathtt{t}$, the term $\mathtt{P}_{\overline{\pi}}(\mathtt{t})$ such that $\overline{\pi}\mathtt{P}_{\overline{\pi}}(\mathtt{t}) \to^* \mathtt{t}$. If $\overline{\pi}$ is the empty sequence, $\mathtt{P}_{\overline{\pi}}(\mathtt{t}) = \mathtt{t}$, if $\overline{\pi} = \overline{\pi'}\pi_1$ (resp. $\overline{\pi'}\pi_2$), then $\mathtt{P}_{\overline{\pi}}(\mathtt{t}) = \langle \mathtt{P}_{\overline{\pi'}}(\mathtt{t}), I\rangle$ (resp. $\langle I, \mathtt{P}_{\overline{\pi'}}(\mathtt{t})\rangle$). Moreover, let Π and J be two lists of same lenght, the first containing sequences of projections and the second one natural numbers. The term $\nabla(\Pi, J)$ is defined inductively as follows:
if Π and J are empty, then $\nabla(\Pi, J) = \mathtt{I}$ else if $\Pi = \overline{\pi} :: \Pi'$ and $J = j :: J'$ then $\nabla(\Pi, J) = \mathtt{P}_{\overline{\pi}}(\lambda\overline{\mathtt{y}}_j.\nabla(\Pi', J'))$.
Now we prove that, if $\mathtt{t}$ has a $\mathcal{P}$-$\mathtt{hnf}$ and $\mathtt{x}$ is its head variable, then for all Π, J there exists a term $O^{\mathtt{t}}_{\nabla(\Pi,J)}$ such that $\mathtt{t}\{O^{\mathtt{t}}_{\nabla(\Pi,J)}/\mathtt{x}\} \to^* \nabla(\Pi, J)$. If $\mathtt{t} = \mathtt{x}\overline{\mathtt{t}}_n$, then $O^{\mathtt{t}}_{\nabla(\Pi,J)} = \lambda\overline{\mathtt{y}}_n.\nabla(\Pi, J)$.

If $\mathtt{t} = \overline{\pi}\mathtt{u}\mathtt{t}_n$, then $O^{\mathtt{t}}_{\nabla(\Pi,J)} = O^{\mathtt{u}}_{\nabla(\overline{\pi}::\Pi,n::J)}$. In fact, $\overline{\pi}\mathtt{u}\mathtt{t}_n\{O^{\mathtt{u}}_{\nabla(\overline{\pi}::\Pi,n::J)}/\mathtt{x}\}$ $\to^* \overline{\pi}(\mathtt{u}\{O^{\mathtt{u}}_{\nabla(\overline{\pi}::\Pi,n::J)}/\mathtt{x}\})\mathtt{t}'_n \to^*_{(i.h.)} \overline{\pi}\nabla(\overline{\pi} :: \Pi, n :: J)\overline{\mathtt{t}'}_n$ which reduces to $\overline{\pi}\mathtt{P}_{\overline{\pi}}(\lambda\overline{\mathtt{y}}_n.\nabla(\Pi, J))\overline{\mathtt{t}'}_n \to^* \nabla(\Pi, J)$, where $\overline{\mathtt{t}'}_n = \overline{\mathtt{t}}_n\{O^{\mathtt{u}}_{\nabla(\mathtt{P}_{\overline{\pi}}::\Pi,n::J)}/\mathtt{x}\}$. Now, to show the statement of the lemma we proceed by cases.

If $\mathtt{t} \in \mathcal{P}$, then let $\mathtt{x}$ be the head of $\mathtt{t}$ and let $\mathtt{C} = (\lambda\mathtt{x}.\Box)O^{\mathtt{t}}_{\nabla(\epsilon,\epsilon)}$. By the previous point we have $\mathtt{C}[\mathtt{t}] = (\lambda\mathtt{x}.\mathtt{t})O^{\mathtt{t}}_{\nabla(\epsilon,\epsilon)} \to \mathtt{t}\{O^{\mathtt{t}}_{\nabla(\epsilon,\epsilon)}/\mathtt{x}\} \to^* \nabla(\epsilon, \epsilon) = \mathtt{I}$. If $\mathtt{t} = \lambda\overline{\mathtt{y}}_n.\langle\mathtt{u}, \mathtt{v}\rangle$, then let $\mathtt{C} = \Box\overline{\mathtt{I}}_n$. Then $\mathtt{C}[\mathtt{t}] \to^* \langle\mathtt{u}, \mathtt{v}\rangle$. If $\mathtt{t} = \lambda\overline{\mathtt{y}}_n.\mathtt{u}$, where $\mathtt{u} \in \mathcal{P}$, then let $\mathtt{x}$ be the head of $\mathtt{u}$. If $\mathtt{x} \notin \overline{\mathtt{y}}_n$, we let $\mathtt{C} = (\lambda\mathtt{x}.\Box)O^{\mathtt{u}}_{\nabla(\epsilon,\epsilon)}\overline{\mathtt{I}}_n$. Then $\mathtt{C}[\mathtt{t}] \to^* (\lambda\overline{\mathtt{y}}_n.\mathtt{u}\{O^{\mathtt{u}}_{\nabla(\epsilon,\epsilon)}/\mathtt{x}\})\overline{\mathtt{I}}_n \to^* (\lambda\overline{\mathtt{y}}_n.\mathtt{I})\overline{\mathtt{I}}_n \to^* \mathtt{I}$. If $\mathtt{x} = \mathtt{y}_i$, we let $\mathtt{C} = (\Box)\mathtt{I}_{i-1}O^{\mathtt{u}}_{\nabla(\epsilon,\epsilon)}\overline{\mathtt{I}}_{n-i}$. Then $\mathtt{C}[\mathtt{t}] \to^* \mathtt{u}\{O^{\mathtt{u}}_{\nabla(\epsilon,\epsilon)}/\mathtt{x}\}\{\overline{\mathtt{I}}_{n-1}/\overline{\mathtt{y}}_{n-1}\} \to^* \mathtt{I}\{\overline{\mathtt{I}}_{n-1}/\overline{\mathtt{y}}_{n-1}\} = \mathtt{I}$.

4.1 The type assignment system $\mathcal{P}$

We now present the system $\mathcal{P}$, an extension of system $\mathcal{M}$ which assigns types to Λ_π-terms in such a way that typability coincides with solvability.

Definition 6.

1. *The set $\mathcal{TP}$ of types is extended using the following grammar:*

$$\begin{array}{lll} \sigma, \tau, \rho ::= \omega_\times \mid \alpha \mid \mathtt{A} \to \tau \mid \times_1(\tau) \mid \times_2(\tau) & & (\textit{types}) \\ \mathtt{A} \quad ::= [\sigma_i]_{i\in I} & & (\textit{multiset types}) \end{array}$$

   where $\omega_\times$ is a type constant.
2. *A typing judgement is a triple of the form $\Gamma \vdash \mathtt{t} : \mathtt{A}$, where Γ is a typing environment defined as in Def. 1. The type system $\mathcal{P}$ is obtained by adding to the system $\mathcal{M}$ the rules given in Fig. 5.*
3. *The definition of left-subtree of a derivation Σ is as Def. 1.4, with the the following additional cases:*
   - *a derivation $\Pi \rhd \Delta \vdash \mathtt{t} : \sigma$ is a left subtree of a derivation Σ if Π is the premise of $\Sigma' \rhd \Delta \vdash \langle\mathtt{t}, \mathtt{u}\rangle : \times_1(\sigma)$ and Σ' is a left subtree of Σ (and similarly for the case $\times_2(\sigma)$).*
   - *a derivation $\Pi \rhd \Delta \vdash \mathtt{t} : \times_i(\sigma)$ is a left subtree of a derivation Σ if Π is the premise of $\Sigma' \rhd \Delta \vdash \pi_i(\mathtt{t}) : \sigma$ and Σ' is a left subtree of Σ, for $i = 1, 2$.*

The constant $\omega_\times$ is a universal type for pairs, making all pairs typable. The system is relevant, and it enjoys both subject reduction (in a weighted version) and subject expansion. The notions of occurrences and typed occurrences are extended as expected from those of Def. 2.

$$\frac{}{\vdash \langle \mathtt{t},\mathtt{u}\rangle : \omega_\times}\ (\mathtt{emptypair})$$

$$\frac{\Gamma \vdash \mathtt{t} : \sigma}{\Gamma \vdash \langle \mathtt{t},\mathtt{u}\rangle : \times_1(\sigma)}\ (\mathtt{pair1}) \qquad \frac{\Gamma \vdash \mathtt{u} : \tau}{\Gamma \vdash \langle \mathtt{t},\mathtt{u}\rangle : \times_2(\tau)}\ (\mathtt{pair2})$$

$$\frac{\Gamma \vdash \mathtt{t} : \times_1(\sigma)}{\Gamma \vdash \pi_1\mathtt{t} : \sigma}\ (\mathtt{proj1}) \qquad \frac{\Gamma \vdash \mathtt{t} : \times_2(\sigma)}{\Gamma \vdash \pi_2\mathtt{t} : \sigma}\ (\mathtt{proj2})$$

Fig. 5: Typing rules for pairs in $\mathcal{P}$

Lemma 4.

1. *(Subject reduction)* $\Pi \rhd \Gamma \vdash \mathtt{t} : \sigma$ *and* $\mathtt{t} \to \mathtt{u}$ *imply* $\Pi' \rhd \Gamma \vdash \mathtt{u} : \sigma$ *where* $\mathtt{meas}(\Pi') \leq \mathtt{meas}(\Pi)$*. In particular, if the reduced redex is typed, then* $\mathtt{meas}(\Pi') < \mathtt{meas}(\Pi)$*.*
2. *(Subject expansion)* $\Gamma \vdash \mathtt{u} : \sigma$ *and* $\mathtt{t} \to \mathtt{u}$ *imply* $\Gamma \to \mathtt{t} : \sigma$*.*

Lemma 5. *Let* $\Pi \rhd \Gamma \vdash \mathtt{t} : \sigma$*. Then* $\mathtt{t}$ *has* $\mathtt{hnf}$*.*

Proof. By induction on $\mathtt{meas}(\Pi)$ using Lem. 4.1.

The solvability characterization is proved in the next theorem.

Theorem 3. *The following statements are equivalent: (1)* $\mathtt{t}$ *is solvable; (2)* $\mathtt{t}$ *has* $\mathtt{hnf}$*; (3)* $\mathtt{t}$ *is typable in system* $\mathcal{P}$*.*

Proof. $2 \Rightarrow 1$ holds by Lem. 3 and $3 \Rightarrow 2$ holds by Lem. 5. We now show $1 \Rightarrow 3$: $\mathtt{t}$ solvable implies, by definition, the existence of a context $\mathtt{C}$ such that $\mathtt{C} = (\lambda\overline{\mathtt{x}}.\Box)\overline{\mathtt{v}}$ and either $\mathtt{C}[\mathtt{t}] \to^* \mathtt{I}$ or $\mathtt{C}[\mathtt{t}] \to^* \langle \mathtt{u}_1, \mathtt{u}_2\rangle$, for some $\mathtt{u}_1, \mathtt{u}_2$. Both $\mathtt{I}$ and $\langle \mathtt{u}_1, \mathtt{u}_2\rangle$ can be typed, so $\mathtt{C}[\mathtt{t}]$ is typed, by Lem. 4.2. Note that $\mathtt{C}[\mathtt{t}] = (\lambda\overline{\mathtt{x}}.\mathtt{t})\overline{\mathtt{v}}$, so for typing $\mathtt{C}[\mathtt{t}]$ we need to type $\mathtt{t}$.

5 Inhabitation for system $\mathcal{P}$

We extend approximate normal forms (*cf.* Sec. 3) as follows:

$$\mathtt{a}, \mathtt{b}, \mathtt{c} \ ::= \ \Omega \mid \mathcal{N} \qquad \mathcal{N} \ ::= \ \lambda\mathtt{x}.\mathcal{N} \mid \langle \mathtt{a}, \mathtt{a}\rangle \mid \mathcal{L} \qquad \mathcal{L} \ ::= \ \mathtt{x} \mid \mathcal{L}\mathtt{a} \mid \pi_i\mathcal{L}$$

The order relation $\leq$ and the sets $\mathcal{A}(\mathtt{t})$ are defined as in the case of the pure λ-calculus. The type assignment system $\mathcal{P}$ for the Λ_π-calculus is extended to approximate normal forms, assuming as before that no type can be assigned to the constant Ω. Given $\Pi \rhd \Gamma \vdash \mathtt{t} : \tau$, where $\mathtt{t}$ is in $\mathtt{hnf}$, we extend the definition of minimal approximant of $\mathtt{t}$.

Definition 7. *Given $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$, where $\mathtt{t}$ is in Π-normal form, $\mathcal{A}(\Pi) \in \mathcal{A}(\mathtt{t})$ is defined by extending Def. 3 with the the following additional cases:*

- *If $\Pi \triangleright \ \vdash \langle \mathtt{t}, \mathtt{u} \rangle : \omega_\times$, then $\mathcal{A}(\Pi) = \langle \Omega, \Omega \rangle$.*
- *If $\Pi \triangleright \Gamma \vdash \langle \mathtt{t}_1, \mathtt{t}_2 \rangle : \times_i(\tau)$ follows from $\Pi_i \triangleright \Gamma \vdash \mathtt{t}_i : \tau$, then $i = 1$ implies $\mathcal{A}(\Pi) = \langle \mathcal{A}(\Pi_1), \Omega \rangle$ and $i = 2$ implies $\mathcal{A}(\Pi) = \langle \Omega, \mathcal{A}(\Pi_2) \rangle$.*
- *If $\Pi \triangleright \Gamma \vdash \pi_i \mathtt{t} : \tau$ follows from $\Pi' \triangleright \Gamma \vdash \mathtt{t} : \times_i(\tau)$, then $\mathcal{A}(\Pi) = \pi_i \mathcal{A}(\Pi')$.*

5.1 The inhabitation algorithm

The algorithm in Fig. 4 is extended with the additional rules in Fig 6.

$$\frac{\mathtt{a} \in \mathtt{H}^{\Delta}_{\pi_i(\mathtt{b})}(\Gamma, \sigma) \triangleright \tau}{\mathtt{a} \in \mathtt{H}^{\Delta}_{\mathtt{b}}(\Gamma, \times_i(\sigma)) \triangleright \tau}\ (\mathtt{Proj}) \qquad \frac{}{\langle \Omega, \Omega \rangle \in \mathtt{T}(\emptyset, \omega_\times)}\ (\mathtt{Pair})$$

$$\frac{\mathtt{a} \in \mathtt{T}(\Gamma, \tau)}{\langle \mathtt{a}, \Omega \rangle \in \mathtt{T}(\Gamma, \times_1(\tau))}\ (\mathtt{Prod1}) \qquad \frac{\mathtt{a} \in \mathtt{T}(\Gamma, \tau)}{\langle \Omega, \mathtt{a} \rangle \in \mathtt{T}(\Gamma, \times_2(\tau))}\ (\mathtt{Prod2})$$

Fig. 6: The inhabitation algorithm for the λ-calculus with products

Example 2. Let $\Gamma = \emptyset$ and $\tau = \times_1([\alpha] \to \alpha)$ and $\sigma = [\tau] \to \tau$. Then given the input (Γ, σ), the algorithm can have the following two behaviours:

Choosing the sequence of rules: $(\mathtt{Abs}), (\mathtt{Prod1}), (\mathtt{Abs}), (\mathtt{Head}), (\mathtt{Proj}), (\mathtt{Prefix}), (\mathtt{Final})$ the final approximant is $\lambda \mathtt{y}.\langle \lambda \mathtt{x}.\pi_1 \mathtt{yx}, \Omega \rangle$;

Choosing the sequence of rules: $(\mathtt{Abs}), (\mathtt{Head}), (\mathtt{Final})$ the final approximant is $\lambda \mathtt{y}.\mathtt{y}$;

To show that the inhabitation algorithm terminates, we extend the measure given in Sec. 2 by adding $\#(\omega_\times) = 1$ and $\#(\times_i(\tau)) = \#(\tau) + 1$. Termination and soundness hold, and the extension is straightforward.

Lemma 6 (Termination). *The inhabitation algorithm terminates.*

Proof. As for Lem. 1, using the extended measure above.

Lemma 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. We follow the proof of Lem. 2, with the suitable additional cases:

($\Rightarrow$): Let $\mathtt{a} \in \mathtt{H}^{\Delta}_{\mathtt{b}}(\Gamma, \times_i(\sigma)) \triangleright \tau$ follow from $\mathtt{a} \in \mathtt{H}^{\Delta}_{\pi_i(\mathtt{b})}(\Gamma, \sigma) \triangleright \tau$ by $(\mathtt{Proj})$. Suppose $\Delta \vdash \mathtt{b} : \times_i(\sigma)$ $(i = 1, 2)$. Then $\Delta \vdash \pi_i(\mathtt{b}) : \sigma$. By the *i.h.* (c) we get $\Pi \triangleright \Gamma + \Delta \vdash \mathtt{a} : \tau$, where $\mathtt{a} = \mathcal{A}(\Pi)$ and we are done.

If $\langle \Omega, \Omega \rangle \in \mathtt{T}(\emptyset, \omega_\times)$ follows by (`Pair`); then $\Pi \rhd \vdash \langle \Omega, \Omega \rangle : \omega_\times$, and $\langle \Omega, \Omega \rangle = \mathcal{A}(\Pi)$.

If $\langle \mathtt{a}, \Omega \rangle \in \mathtt{T}(\Gamma, \times_1(\tau))$ follows from $\mathtt{a} \in \mathtt{T}(\Gamma, \tau)$ by (`Prod1`), then by the *i.h.* (a) $\Gamma \vdash \mathtt{a} : \tau$. Then by (`pair1`), $\exists \Pi \rhd \Gamma \vdash \langle \mathtt{a}, \Omega \rangle : \times_1(\tau)$, and we are done, since $\langle \mathtt{a}, \Omega \rangle = \mathcal{A}(\Pi)$. Analougsly for (`Prod2`).

($\Leftarrow$): 1. Let $\mathtt{a} = \pi_i \mathtt{a}'$ $(i = 1, 2)$. By construction Π ends by an application of the rule (`proji`) with premise $\Pi' \rhd \Gamma \vdash \mathtt{a}' : \times_i(\sigma)$, where by definition $\mathtt{a}' = \mathcal{A}(\Pi')$. Moreover, $\Sigma \rhd \Delta \vdash \mathtt{b} : \tau$ is also a left-subtree of Π' and $\mathcal{A}(\Pi) = \pi_i \mathtt{a}'$ by definition, thus $\mathtt{H}^{\Delta+\Gamma'}_{\pi_i \mathtt{a}'}(\Theta, \sigma) \rhd \pi \subseteq_{(\mathtt{Proj})} \mathtt{H}^{\Delta+\Gamma'}_{\mathtt{a}'}(\Theta, \times_i(\sigma)) \rhd \pi \subseteq_{i.h.(1)} \mathtt{H}^{\Delta}_{\mathtt{b}}(\Theta + \Gamma', \tau) \rhd \pi$.

2. If $\mathtt{a}$ is a pair then there are three cases to consider. If $\Pi \rhd \vdash \mathtt{a} : \omega_\times$, then $\mathtt{a} = \langle \Omega, \Omega \rangle \in_{(\mathtt{Pair})} \mathtt{T}(\emptyset, \omega_\times)$. If $\Pi \rhd \Gamma \vdash \mathtt{a} : \times_1(\tau)$ follows from $\Pi' \rhd \Gamma \vdash \mathtt{a}' : \tau$ by (`proj1`), then $\mathtt{a} = \langle \mathtt{a}', \Omega \rangle$. We have $\mathtt{a}' \in_{i.h.(2)} \mathtt{T}(\Gamma, \tau)$, so that $\mathtt{a} \in_{(\mathtt{Prod1})} \mathtt{T}(\Gamma, \times_1(\tau))$. The case $\Pi \rhd \Gamma \vdash \mathtt{a} : \times_2(\tau)$ is similar.

Theorem 4 (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* $\Pi \rhd \Gamma \vdash \mathtt{t} : \sigma$ *then there exists* $\Pi' \rhd \Gamma \vdash \mathtt{t}' : \sigma$ *such that* $\mathtt{t}'$ *is in* Π'*-normal form, and* $\mathcal{A}(\Pi') \in \mathtt{T}(\Gamma, \sigma)$.

Proof. As in the proof of Thm. 2, but using Lem. 7 ($\Rightarrow$) for soundness, Thm. 4.1 and Lem. 7 ($\Leftarrow$) for completeness.

6 Conclusion

We proved that the inhabitation problem is decidable for the types systems $\mathcal{M}$ and $\mathcal{P}$, based on non-idempotent intersection types, and characterizing solvability for the λ-calculus and for its extension with pairs and projections, that we call Λ_π, respectively. To the best of our knowledge, solvability in Λ_π had not been studied before. Our result is a first step towards the study of further extensions of Λ_π, including *patterns* [11].

In fact, the logical characterization of solvability is related to the inhabitation problem of the underlying type system. While this relation is implicit for the λ-calculus, it can become crucial for extensions lacking a syntactical characterization of solvability.

Concerning denotational models, it is well known that system $\mathcal{M}$ induces a relational model of Λ. We aim to complete the picture studying the semantics of Λ_π through the system $\mathcal{P}$.

References

1. H. Barendregt. The Lambda Calculus: Its Syntax and Semantics. North-Holland, Amsterdam, revised edition, 1984.
2. C. Ben-Yelles. Type-assignment in the lambda-calculus; syntax and semantics. PhD thesis, University of Wales Swansea, 1979.
3. A. Bernadet and S. Lengrand. Non-idempotent intersection types and strong normalisation. *Logical Methods in Computer Science*, 9(4), 2013.
4. M. W. Bunder. The inhabitation problem for intersection types. In *CATS'08*, *CRPIT* 77, pages 7–14. Wollongong, NSW, Australia, 2008.
5. D. D. Carvalho. Sèmantique de la logique linéaire et temps de calcul. PhD thesis, Université de la Mediterranée Aix-Marseille 2, 2007.
6. M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame J. Form. Log.*, 21(4):685–693, 1980.
7. E. De Benedetti and S. Ronchi Della Rocca. Bounding normalization time through intersection types. In *ITRS'12*, EPTCS 121, pages 48–57, 2013.
8. P. Di Gianantonio, F. Honsell, and M. Lenisa. A type assignment system for game semantics. *Theoretical Computer Science*, 398:150–169, 2008.
9. T. Ehrhard. The Scott model of linear logic is the extensional collapse of its relational model. *Theoretical Computer Science*, 424:20–45, 2012.
10. J. R. Hindley. Basic Simple Type Theory. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Amsterdam, 2008.
11. C. B. Jay and D. Kesner. First-class patterns. *Journal of Functional Programming*, 19(2):191–225, 2009.
12. D. Kesner and D. Ventura. Quantitative types for the linear substitution calculus. In *TCS*, LNCS, Rome, Italy, 2014.
13. A. J. Kfoury. A Linearization of the Lambda-Calculus and Consequences. *Journal Logic Comp.*, 10(3):411–436, 2000.
14. A. J. Kfoury and J. B. Wells. Principality and type inference for intersection types using expansion variables. *Theoretical Computer Science*, 311(1-3):1–70, 2004.
15. J. L. Krivine. Lambda-Calculus, Types and Models. Masson, Paris, and Ellis Horwood, Hemel Hempstead, 1993.
16. T. Kurata and M. Takahashi. Decidable properties of intersection type systems. In *TLCA'95*, *LNCS* 902, pages 297–311, 1995.
17. H. Mairson and P. M. Neergaard. Types, potency, and idempotency: why nonlinearity and amnesia make a type system work. In *ICFP'04*, pages 138–149, 2004.
18. M. Pagani and S. Ronchi Della Rocca. Linearity, non-determinism and solvability. *Fundamenta Informaticae*, 103:358–373, 2010.
19. M. Pagani and S. Ronchi Della Rocca. Solvability in resource lambda-calculus. In *FOSSACS'10*, *LNCS* 6014, pages 358–373, 2010.
20. L. Paolini, M. Piccolo, and S. Ronchi Della Rocca. Logical relational lambda-models. Accepted for publication in MSCS.
21. P. Urzyczyn. The emptiness problem for intersection types. *Journal of Symbolic Logic*, 64(3):1195–1215, 1999.
22. P. Urzyczyn. Personal communication, 2014.
23. S. van Bakel. Complete restrictions of the intersection type discipline. *Theoretical Computer Science*, 102:135–163, 1992.