

The inhabitation problem for non-idempotent intersection

Antonio Bucciarelli⁽¹⁾, Delia Kesner⁽¹⁾, and Simona Ronchi Della Rocca⁽²⁾

⁽¹⁾Univ Paris Diderot, Sorbonne Paris Cité, PPS, UMR 7126, CNRS, F-75205 Paris, France

⁽²⁾Dipartimento di Informatica, Università di Torino, Italia

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 [5, 7]. Intersection enjoys associativity, commutativity, and in particular idempotency ($A \wedge A = A$). Given a type A and a typing environment Γ , the problem of deciding whether there exists a term t such that $\Gamma \vdash t : A$ 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 [10]. Van Bakel [12] 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 [11], so proving that relevance has nothing to do with the hardness of the inhabitation problem.

We consider the type assignment system $\mathcal{M}$, which is a variant of $\mathcal{S}_r$, where the idempotency of the 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 solvable terms. i.e., 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 t such that $\Gamma \vdash t : \sigma$ can be generated.

In the literature, various restrictions of the classical intersection types system have been shown to have decidable inhabitation [8, 3]. 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 $(\wedge I)$ and $(\wedge E)$, so that the corresponding type assignment does not characterize interesting classes of terms, anymore.

$\frac{x : A \in \Gamma}{\Gamma \vdash x : A} (\text{var})$	$\frac{}{\Gamma \vdash t : \omega} (\omega)$	$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x. t : A \rightarrow B} (\rightarrow I)$	
$\frac{\Gamma \vdash t : A \rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash tu : B} (\rightarrow E)$	$\frac{\Gamma \vdash t : A \quad \Gamma \vdash t : B}{\Gamma \vdash t : A \wedge B} (\wedge I)$	$\frac{\Gamma \vdash t : A_1 \wedge A_2}{\Gamma \vdash t : A_i} (\wedge E)$	Types: $A ::= \alpha \mid \omega \mid A \rightarrow$
$A \mid A \wedge A$	Typing environments: $\Gamma ::= \emptyset \mid \Gamma, x : A \quad x \notin \text{dom}(\Gamma)$		

Figure 1: System $\mathcal{C}$

$\frac{x : A \in \Gamma \quad \sigma \in A}{\Gamma \vdash x : \sigma} (\text{var})$	$\frac{}{x : \{\sigma\} \vdash x : \sigma} (\text{var}_r)$
$\frac{\Gamma, x : A \vdash t : \tau}{\Gamma \vdash \lambda x. t : A \rightarrow \tau} (\rightarrow I)$	$\frac{\Gamma, x : A \vdash t : \tau}{\Gamma \vdash \lambda x. t : A \rightarrow \tau} (\rightarrow I)$
$\frac{\Gamma \vdash t : \{\sigma_i\}_{i \in I} \rightarrow \tau \quad (\Gamma \vdash u : \sigma_i)_{i \in I}}{\Gamma \vdash tu : \tau} (\rightarrow E)$	$\frac{\Gamma \vdash t : \{\sigma_i\}_{i \in I} \rightarrow \tau \quad (\Delta_i \vdash u : \sigma_i)_{i \in I}}{\Gamma \cup_{i \in I} \Delta_i \vdash tu : \tau} (\rightarrow E_r)$
Types: $\sigma ::= a \mid A \rightarrow \sigma$ (strict types) $A ::= \emptyset \mid \{\sigma\} \mid A \cup A$ (set types)	Typing environments: $\Gamma ::= \emptyset \mid \Gamma, x : A \quad x \notin \text{dom}(\Gamma)$ $(\Gamma \cup \Delta)(x) = \Gamma(x) \cup \Delta(x)$

Figure 2: Systems $\mathcal{S}$ and $\mathcal{S}_r$

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 of the λ -calculus are generated by the following grammar:

$$t, u, v ::= x \mid \lambda x. t \mid tu$$

The β -reduction is given by the rule $(\lambda x. t)u \rightarrow_\beta t\{u/x\}$ where $t\{u/x\}$ denotes the capture free replacement of x by u in t .

Definition 1.

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

$$\begin{aligned} \sigma, \tau, \rho &::= \alpha \mid A \rightarrow \tau & (\text{types}) \\ A &::= [\sigma_i]_{i \in I} & (\text{multiset types}) \end{aligned}$$

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

2. Typing environments, 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 $\text{dom}(\Gamma)$, is the set of variables whose image is different from $[\]$. Given the environments $\{\Gamma_i\}_{i \in I}$, the environment $+_{i \in I} \Gamma_i$ maps x to $\uplus_{i \in I} \Gamma_i(x)$, where $\uplus$ denotes multiset union. If $I = \emptyset$, the resulting environment is the one having empty domain. $\Gamma + \Delta$, and by abuse of notations $\Gamma +_{i \in I} \Delta_i$ are defined accordingly. $\Gamma \setminus x$ is the environment assigning $[\]$ to x , and acting as Γ otherwise; $x_1 : A_1, \dots, x_n : A_n$ is the environment assigning A_i to x_i , for $1 \leq i \leq n$, and $[\]$ to any other variable.
3. A typing judgement is a triple of the form $\Gamma \vdash t : A$. The type system $\mathcal{M}$ is given in Fig. 3. The measure of a derivation Π , written $\text{meas}(\Pi)$, is the number of rule applications in it. $\Pi \triangleright \Gamma \vdash t : \sigma$ denotes a derivation proving $\Gamma \vdash t : \sigma$; by abuse of notation, $\Gamma \vdash t : \sigma$ denotes the existence of such a derivation.

Let us recall the notion of head-normal forms (**hnf**), which is the syntactical counter part of the well known notion of solvability for the λ -calculus. A λ -term is in **hnf** if it is generated by the following grammar:

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

$$\begin{array}{c}
\frac{}{\mathbf{x} : [\rho] \vdash \mathbf{x} : \rho} \text{ (var)} \quad \frac{\Gamma \vdash \mathbf{t} : \rho}{\Gamma \setminus \mathbf{x} \vdash \lambda \mathbf{x}. \mathbf{t} : \Gamma(\mathbf{x}) \rightarrow \rho} (\rightarrow \text{I}) \\
\\
\frac{\Gamma \vdash \mathbf{t} : [\sigma_i]_{i \in I} \rightarrow \rho \quad (\Delta_i \vdash \mathbf{u} : \sigma_i)_{i \in I}}{\Gamma +_{i \in I} \Delta_i \vdash \mathbf{t} \mathbf{u} : \rho} (\rightarrow \text{E})
\end{array}$$

Figure 3: The type assignment system $\mathcal{M}$

A term *has hnf* if it β -reduces to a **hnf**.

Rule $(\rightarrow \text{E})$ enables the typability of non strongly normalizing terms, when $I = \emptyset$. For example $\mathbf{x} : [[] \rightarrow \alpha] \vdash \mathbf{x}((\lambda \mathbf{y}. \mathbf{y} \mathbf{y})(\lambda \mathbf{y}. \mathbf{y} \mathbf{y})) : \alpha$, 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 $\Pi \triangleright \Gamma \vdash \mathbf{t} : \sigma$ and $\mathbf{t} \rightarrow_\beta \mathbf{u}$, then $\Pi' \triangleright \Gamma \vdash \mathbf{u} : \sigma$. What is peculiar to $\mathcal{M}$ is the fact that the size of Π' is strictly smaller than that of Π , whenever the reduction $\mathbf{t} \rightarrow_\beta \mathbf{u}$ takes place in an occurrence of $\mathbf{t}$ which is *typed in* Π . This property, stated in Theorem 2.1 below, allows for an original, combinatorial proof of the fact that typed terms do have **hnf**.

We do not provide here the formal definition of what a typed occurrence is. Roughly speaking, given a derivation $\Pi \triangleright \Gamma \vdash \mathbf{t} : \sigma$, an occurrence of a subterm $\mathbf{u}$ of $\mathbf{t}$ is typed in Π if $\mathbf{u}$ at this occurrence is the subject of a subderivation of Π .

Theorem 2.

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

Proof. For 1 see [9] and for 2 see [4]. □

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 $(\rightarrow \text{E})$ 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:

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

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

Let $\mathcal{A}(\mathbf{t}) = \{\mathbf{a} \mid \exists \mathbf{u} \mathbf{t} \rightarrow_\beta^* \mathbf{u} \text{ and } \mathbf{a} \leq \mathbf{u}\}$ be the set of *approximants* of the λ -term $\mathbf{t}$, and let $\bigvee$ denote the least upper bound w.r.t. $\leq$. We write $\uparrow_{i \in I} \mathbf{a}_i$ to denote the fact that $\bigvee \{\mathbf{a}_i\}_{i \in I}$ does exist. It is easy to check that, for every $\mathbf{t}$ and $\mathbf{a}_1, \dots, \mathbf{a}_n \in \mathcal{A}(\mathbf{t})$, $\uparrow_{i \in \{1, \dots, n\}} \mathbf{a}_i$.

An approximate normal form $\mathbf{a}$ is a *head subterm* of $\mathbf{b}$ if either $\mathbf{b} = \mathbf{a}$ or $\mathbf{b} = \mathbf{c} \mathbf{c}'$ and $\mathbf{a}$ is a head subterm of $\mathbf{c}$.

The type assignment system $\mathcal{M}$ gives type to approximate normal forms, also, simply assuming that no type can be assigned to the constant Ω . It is easy to check that, if $\Gamma \vdash \mathbf{a} : \sigma$ and $\mathbf{a} \leq \mathbf{b}$ (resp. $\mathbf{a} \leq \mathbf{t}$) then $\Gamma \vdash \mathbf{b} : \sigma$ (resp. $\Gamma \vdash \mathbf{t} : \sigma$). Given $\Pi \triangleright \Gamma \vdash \mathbf{t} : \tau$, where $\mathbf{t}$ is in **hnf**, we denote by $\mathcal{A}(\Pi)$ the minimal approximant $\mathbf{b}$ of $\mathbf{t}$ such that $\Pi \triangleright \Gamma \vdash \mathbf{b} : \tau$. Formally,

$$\begin{array}{c}
\frac{\mathbf{a} \in \mathbf{T}(\Gamma + (\mathbf{x} : \mathbf{A}), \tau) \quad \mathbf{x} \notin \text{dom}(\Gamma)}{\lambda \mathbf{x}. \mathbf{a} \in \mathbf{T}(\Gamma, \mathbf{A} \rightarrow \tau)} \text{ (Abs)} \\
\\
\frac{(\mathbf{a}_i \in \mathbf{T}(\Gamma_i, \sigma_i))_{i \in I} \quad \uparrow_{i \in I} \mathbf{a}_i}{\bigvee_{i \in I} \mathbf{a}_i \in \mathbf{TI}(+_{i \in I} \Gamma_i, [\sigma_i]_{i \in I})} \text{ (Union)} \\
\\
\frac{\mathbf{a} \in \mathbf{H}_x^{x:[\sigma]}(\Gamma, \sigma) \triangleright \tau}{\mathbf{a} \in \mathbf{T}(\Gamma + (\mathbf{x} : [\sigma]), \tau)} \text{ (Head)} \quad \frac{\sigma = \tau}{\mathbf{a} \in \mathbf{H}_a^\Delta(\emptyset, \sigma) \triangleright \tau} \text{ (Final)} \\
\\
\frac{\Gamma = \Gamma_0 + \Gamma_1 \quad \mathbf{b} \in \mathbf{TI}(\Gamma_0, \mathbf{A}) \quad \mathbf{a} \in \mathbf{H}_{cb}^{\mathbf{A} + \Gamma_0}(\Gamma_1, \sigma) \triangleright \tau}{\mathbf{a} \in \mathbf{H}_c^\Delta(\Gamma, \mathbf{A} \rightarrow \sigma) \triangleright \tau} \text{ (Prefix)}
\end{array}$$

Figure 4: The inhabitation algorithm

Definition 3. Given $\Pi \triangleright \Gamma \vdash \mathbf{t} : \sigma$, where $\mathbf{t}$ is in **hnf**, $\mathcal{A}(\Pi) \in \mathcal{A}(\mathbf{t})$ is defined by induction on $\text{meas}(\Pi)$ as follows:

- If $\Pi \triangleright \Gamma \vdash \mathbf{x} : \rho$, then $\mathcal{A}(\Pi) = \mathbf{x}$.
- If $\Pi \triangleright \Gamma \vdash \lambda \mathbf{x}. \mathbf{t} : \mathbf{A} \rightarrow \rho$ follows from $\Pi' \triangleright \Gamma, \mathbf{x} : \mathbf{A} \vdash \mathbf{t} : \rho$, then $\mathcal{A}(\Pi) = \lambda \mathbf{x}. \mathcal{A}(\Pi')$, $\mathbf{t}$ being in **hnf**.
- If $\Pi \triangleright \Gamma \vdash \mathbf{t} \mathbf{u} : \rho$ follows from $\Pi' \triangleright \Gamma \vdash \mathbf{t} : [] \rightarrow \rho$, then $\mathcal{A}(\Pi) = \mathcal{A}(\Pi')\Omega$, $\mathbf{t}$ being in **hnf**.
- If $\Pi \triangleright \Gamma = \Gamma' +_{i \in I} \Delta_i \vdash \mathbf{t} \mathbf{u} : \rho$ follows from $\Pi' \triangleright \Gamma \vdash \mathbf{t} : [\sigma_i]_{i \in I} \rightarrow \rho$ and $(\Pi_i \triangleright \Delta_i \vdash \mathbf{u} : \sigma_i)_{i \in I}$, for some $I \neq \emptyset$, then by Theorem 2.2, we know that $\mathbf{u}$ reduces to a **hnf** $\mathbf{u}'$ and, by Theorem 2.1, we have $\Pi'_i \triangleright \Delta_i \vdash \mathbf{u}' : \sigma_i$, and $\text{meas}(\Pi'_i) \leq \text{meas}(\Pi_i)$, for all $i \in I$. Moreover, $\uparrow_{i \in I} \mathcal{A}(\Pi'_i)$, since $\mathcal{A}(\Pi'_i) \in \mathcal{A}(\mathbf{u}')$, for all $i \in I$. Then $\mathcal{A}(\Pi) = \mathcal{A}(\Pi')(\bigvee_{i \in I} \mathcal{A}(\Pi'_i))$.

Proposition 4. If $\Pi \triangleright \Gamma \vdash \mathbf{t} : \sigma$ then $\Pi \triangleright \Gamma \vdash \mathcal{A}(\Pi) : \sigma$.

Proof. Induction on $\text{meas}(\Pi)$. □

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 $\mathbf{T}(\Gamma, \sigma)$ containing *all* the approximate normal forms $\mathbf{a}$ such that there exists a derivation $\Pi \triangleright \Gamma \vdash \mathbf{a} : \sigma$, with $\mathbf{a} = \mathcal{A}(\Pi)$, then stops. The algorithm uses two auxiliary predicates, namely $\mathbf{TI}(\Gamma, [\sigma_i]_{i \in I})$ and $\mathbf{H}_a^\Delta(\Gamma, \sigma) \triangleright \tau$.

$\mathbf{TI}(\Gamma, [\sigma_i]_{i \in I})$ contains all the approximate normal forms $\mathbf{a} = \bigvee_{i \in I} \mathbf{a}_i$ such that $\Gamma = +_{i \in I} \Gamma_i$, $\mathbf{a}_i \in \mathbf{T}(\Gamma_i, \sigma_i)$ for all $i \in I$, and $\uparrow_{i \in I} \mathbf{a}_i$.

Finally, $\mathbf{H}_a^\Delta(\Gamma, \sigma) \triangleright \tau$ contains all the approximate normal forms $\mathbf{b}$ such that $\mathbf{a}$ is a head subterm of $\mathbf{b}$, and such that if $\mathbf{a} \in \mathbf{T}(\Delta, \sigma)$ then $\mathbf{b} \in \mathbf{T}(\Gamma + \Delta, \tau)$.

Rule **Union**, in the particular case $I = \emptyset$, gives $\Omega \in \mathbf{TI}(\emptyset, [])$, where $\emptyset$ denotes the environment having empty domain. The algorithm is not an obvious extension of the classical inhabitation algorithm for simple types [2, 6]. In particular, given an environment Γ and a type σ it reconstructs all the terms $\mathbf{t}$ in normal form such that $\Gamma \vdash \mathbf{t} : \sigma$, while the algorithm for simple types reconstructs just the long η -normal forms. The algorithm exhibits a non deterministic behaviour, as showed in the following example.

Example 5. 1. Let $\Gamma = \emptyset$ and $\sigma = [[\alpha] \rightarrow \alpha] \rightarrow [\alpha] \rightarrow \alpha$. With input (Γ, σ) , the algorithm can have the following two behaviours:

Choosing the sequence of rules: (Abs), (Abs), (Head), (Prefix), (Final) the final approximant is $\lambda xy.xy$;

Choosing the sequence of rules: (Abs), (Head), (Final) the final approximant is $\lambda x.x$.

2. Let $\Gamma = \emptyset$ and $\sigma = [[] \rightarrow \alpha] \rightarrow \alpha$. With input (Γ, σ) , the algorithm applies the sequence of rules: (Abs), (Head), (Prefix), (Final) and the final approximant is $\lambda x.x\Omega$;

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

Theorem 7. 1. (Soundness) If $\mathbf{a} \in \mathbf{T}(\Gamma, \sigma)$ then, for all $\mathbf{t}$ such that $\mathbf{a} \leq \mathbf{t}$, $\Gamma \vdash \mathbf{t} : \sigma$.
 2. (Completeness) If $\Pi \triangleright \Gamma \vdash \mathbf{t} : \sigma$ and $\mathbf{t}$ is in hnf, then $\mathcal{A}(\Pi) \in \mathbf{T}(\Gamma, \sigma)$.

References

- [1] Henk 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] Martin W. Bunder. The inhabitation problem for intersection types. In CATS, volume 77 of CRPIT, pages 7–14. Australian Computer Society, 2008.
- [4] Daniel De Carvalho. Sémantique de la logique linéaire et temps de calcul. PhD thesis, Université de la Méditerranée Aix-Marseille 2, 2007.
- [5] 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.
- [6] J. Roger Hindley. Basic Simple Type Theory. Cambridge Tracts in Theoretical Comp. Scie. Cambridge University Press, Amsterdam, 2008.
- [7] Jean Louis Krivine. Lambda-Calculus, Types and Models. Masson, Paris, and Ellis Horwood, Hemel Hempstead, 1993.
- [8] Toshihiko Kurata and Masako Takahashi. Decidable properties of intersection type systems. In TLCA'95, volume 902 of LNCS, pages 297–311, 1995.
- [9] Luca Paolini, Mauro Piccolo, and Simona Ronchi Della Rocca. Logical relational lambda-models. Accepted for publication in MSCS, 2013.
- [10] Pawel Urzyczyn. The emptiness problem for intersection types. Journal of Symbolic Logic, 64(3):1195–1215, 1999.
- [11] Pawel Urzyczyn. personal communication, 2014.
- [12] Steffen van Bakel. Complete restrictions of the intersection type discipline. Theor. Comput. Sci., 102:135–163, 1992.