Logical Methods in Computer Science
Vol. 14(3:7)2018, pp. 1–28
https://lmcs.episciences.org/

Submitted Dec. 12, 2017
Published Aug. 03, 2018

INHABITATION FOR NON-IDEMPOTENT INTERSECTION TYPES

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

[a,b] IRIF, CNRS and Univ Paris-Diderot, France
e-mail address: {buccia,kesner}@irif.fr

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

In honour of Furio Honsell, in occasion of his 60th birthday

Abstract. The inhabitation problem for intersection types in λ-calculus is known to be undecidable. We study the problem in the case of non-idempotent intersection, considering several type assignment systems, which characterize the solvable or the strongly normalizing λ-terms. We prove the decidability of the inhabitation problem for all the systems considered, by providing sound and complete inhabitation algorithms for them.

1. Introduction

Given a type assignment system associating types to terms of a given programming language, two problems naturally arise, namely the *typability* and the *inhabitation problem* (known in the literature also as *emptiness problem*). In the former, given a program, one wants to know if it is possible to assign a typing to it, in the latter, in some specular way, given a typing, one aims for a program to which the typing can be assigned. Since types are program specifications, the decidability of the first problem supplies tools for proving program correctness, whereas the decidability of the second one provides tools for program synthesis [29]. Considering the λ-calculus as a general paradigm for functional programming languages, a virtuous example is the simple type assignment system, for which both problems are decidable, and which is the basis of typed functional languages, like ML and Haskell. In this paper we study the inhabitation problem for an extended type assignment system, based on (non-idempotent) intersection types.

Intersection types have been introduced in order to increase the typability power of simple type assignment systems, but quite early they turned out to be a very powerful tool for characterizing semantic properties of λ-calculus, like solvability and strong normalization, and for describing models of λ-calculus in various settings. Intersection types have been presented in the literature in many variants. Historically, one of the first versions is the one characterizing solvable terms [10, 22], that we call system $\mathcal{C}$, shown in Figure 1. In such a system, α denotes any basic type and the universal type is denoted by the constant ω.

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

LOGICAL METHODS IN COMPUTER SCIENCE DOI:10.23638/LMCS-14(3:7)2018
© Antonio Bucciarelli, Delia Kesner, and Simona Ronchi Della Rocca
Creative Commons

Intersection enjoys associativity, commutativity, and idempotency ($\mathtt{A} \wedge \mathtt{A} = \mathtt{A}$). Typing environments Γ are functions from variables to types, we represent them as lists of pairs of the form $\mathtt{x} : \mathtt{A}$, where the comma symbol is used as set constructor. System $\mathcal{C}$ assigns types different from ω to all and only those terms having head-normal forms, which are the syntactical counterpart of the semantic notion of *solvability* [2]. Since it is undecidable to know if a given term is solvable [2], then typability in system $\mathcal{C}$ (with types different from ω) is undecidable too. Moreover, the inhabitation problem for system $\mathcal{C}$ has been proved to be undecidable by Urzyczyn [30]. Remark that system $\mathcal{C}$ is not syntax directed, so it is difficult to reason about it. Van Bakel [33] simplified system $\mathcal{C}$ by using strict types, where intersection is not allowed on the right-hand side of the arrow; his system $\mathcal{C}_B$ is presented in Figure 2, where we represent intersection through a set constructor, and so the universal type ω as the empty set. System $\mathcal{C}_B$ is syntax directed, *i.e.* there are just three typing rules, corresponding to the three different constructors of the λ-calculus. The systems $\mathcal{C}$ and $\mathcal{C}_B$ have the same typability power, neglecting the universal type ω, in the sense that a term is typable in $\mathcal{C}$ by a type different form ω if and only if it is typable in $\mathcal{C}_B$. In particular, Urzyczyn's proof of undecidability of the inhabitation problem for system $\mathcal{C}$ can be easily adapted to system $\mathcal{C}_B$, proving that the latter is undecidable too [32].

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

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

Types:
$\mathtt{A} ::= \alpha \mid \omega \mid \mathtt{A} \to \mathtt{A} \mid \mathtt{A} \wedge \mathtt{A}$

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

Figure 1: System $\mathcal{C}$

$$\frac{\mathtt{x} : \mathtt{A} \in \Gamma \quad \sigma \in \mathtt{A}}{\Gamma \vdash \mathtt{x} : \sigma}(\mathtt{var}_w) \qquad \frac{\Gamma, \mathtt{x} : \mathtt{A} \vdash \mathtt{t} : \tau}{\Gamma \vdash \lambda \mathtt{x}.\mathtt{t} : \mathtt{A} \to \tau}(\to_{\mathtt{I}})$$

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

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

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

Figure 2: System $\mathcal{C}_B$

In this paper we study the inhabitation problem for *non-idempotent intersection types*, *i.e.* considering intersection modulo commutativity and associativity but not idempotence. It is possible to design various type assignment systems using non idempotent intersection; we start from a system, that we call $\mathcal{H}$, which has been introduced in [15] and further used by De Carvalho [9], for the purpose of studying the complexity of reduction. This system is particularly interesting since it induces a denotational model of λ-calculus, in the relational semantics setting [28]. Moreover, it enjoys relevance and has a quantitative flavor. We prove that the inhabitation problem is decidable by exploiting the fact that types keep track faithfully of the different uses of variables in terms, thanks to the relevance of the system. System $\mathcal{H}$ characterizes terms having head normal form. An important result for defining the algorithm solving the inhabitation problem for system $\mathcal{H}$ is an approximation theorem, proved in section 3.1, saying that a term can be assigned all and only the types that can be assigned to its approximants. We solve the inhabitation problem for system $\mathcal{H}$ in a constructive way, by designing 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 with a weakening rule, yielding system $\mathcal{H}_w$, and we prove that inhabitation remains decidable for such an extension.

In the second part of this paper we present some non-idempotent intersection types systems characterizing strong normalization, and we show that the inhabitation problem is decidable for each of them. We take into account different systems present in literature, also when they do not enjoy good properties like subject reduction, with the aim of performing a complete analysis.

Interest of the problem. As said before, inhabitation has been used for program synthesis (see for example [29]), and it is a technical problem interesting by itself. The results of this paper show in particular that the two classical problems about type assignment systems, namely typability and inhabitation, are unrelated problems, giving a first example of a system for which the first one is undecidable while the second one is decidable. This fact suggests interesting questions about the relation between program correctness and program synthesis.

Related work. Various restrictions of the standard intersection type system have been shown to have decidable inhabitation problems [23, 31]. The approach is substantially different from the one 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 system does not characterize interesting classes of terms, anymore. Recently a new notion of intersection type system with bounded dimension has been introduced [13], it is based on decorations of terms called elaborations, that remember some information about their typing derivation, and allow a stratification of typed terms. It has been proved that inhabitation (where inhabitants are decorated terms) is decidable, and its complexity is EXPSPACE complete. Previously it had been proved in [24] that the inhabitation of rank two intersection types is EXPTIME hard.

A preliminary version of our inhabitation algorithm for system $\mathcal{H}$ has been presented in [6]. The present paper simplifies this first algorithm (see discussion in Section 3.2) and extends inhabitation to other systems, as explained before.

Non-idempotent intersection types. 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 $\mathcal{H}$ is not new: it is the system of Gardner [15] and de Carvalho [9], and it is an instance of the class of the essential λ-models defined in [28], 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 [11]), to supply new characterizations of strong normalization [4, 18], to study type inference [21, 25], to study linearity [20], to characterize solvability in the resource λ-calculus [27, 26], to characterize the set of hereditary head-normalizing infinite λ-terms [34]. Moreover intersection without idempotency, commutativity nor associativity, has been used to study the game semantics of a typed λ-calculus [12]. Non-idempotent types are also useful to prove observational equivalence of programming languages [17, 1]. A unified model-theoretical approach covering both the relevant and non-relevant cases, and unveiling the relations between them, is presented in [14]. Non-idempotent (intersection and union) types have also been proposed [19] to characterize different operational properties of the $\lambda\mu$-calculus, a computational interpretation of classical logic in natural deduction style. A survey on non-idempotent intersection type assignement systems for the λ-calculus and the proof techniques for them can be found in [7].

Organization of the paper. Section 2 contains preliminary notions about the λ-calculus; Section 3 presents the system $\mathcal{H}$ and its inhabitation algorithm; Section 4 presents the systems $\mathcal{H}_{e,w}$ and $\mathcal{S}_w$, both characterizing strong normalisation, and their respective inhabitation algorithms. Finally, Section 5 proposes some conclusions.

2. Preliminaries

The λ-calculus. Terms and contexts of the λ-calculus are generated by the following grammars, respectively:

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

where $\mathtt{x}$ ranges over a countable set of variables. As usual, the application symbol associates to the left; to avoid ambiguities in the notation we may use parentheses as *e.g.* in $\mathtt{x}(\mathtt{xy})$. We use the notation $\mathtt{I}$ for the identity function $\lambda\mathtt{x}.\mathtt{x}$ and $\mathtt{D}_{\mathtt{up}}$ for $\lambda\mathtt{x}.\mathtt{xx}$, which duplicates its argument. We write $\mathtt{fv}(\mathtt{t})$ to denote the set of free variable of $\mathtt{t}$ and $=$ for the syntactical equality on terms, modulo renaming of bound variables. The notation $\lambda\mathtt{xy}.\mathtt{t}$ is used as an abbreviation for $\lambda\mathtt{x}.\lambda\mathtt{y}.\mathtt{t}$. We assume an hygiene condition on variables, *i.e.* free and bound variables have different names, as well as variables bound by different binders. 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 $\mathtt{t}$, thus potentially allowing the capture of free variables of $\mathtt{t}$. A context $\mathtt{C}$ is closing for $\mathtt{t}$ if $\mathtt{C}[\mathtt{t}]$ is closed, *i.e.* if $\mathtt{fv}(\mathtt{C}[\mathtt{t}]) = \emptyset$.

The β-reduction, denoted by $\to_\beta$, is the contextual closure of the rule:

$$(\lambda\mathtt{x}.\mathtt{t})\mathtt{u} \to \mathtt{t}\{\mathtt{u}/\mathtt{x}\}$$

where $\mathtt{t}\{\mathtt{u}/\mathtt{x}\}$ denotes the capture-free replacement of $\mathtt{x}$ by $\mathtt{u}$ in $\mathtt{t}$. A term of the form $(\lambda\mathtt{x}.\mathtt{t})\mathtt{u}$ is called a β-redex. We use $\to_\beta^*$ to denote the reflexive and transitive closure of $\to_\beta$, and $=_\beta$ the transitive, reflexive and symmetric closure of $\to_\beta$.

Normal forms. A term $\mathtt{t}$ *is in β-normal form*, or just in normal form ($\mathtt{nf}$), when it does not contain any β-redex, it *has normal form* if it can be reduced to a term in normal form, it is *strongly normalizing*, written $\mathtt{t} \in SN$, if every β-reduction sequence starting from it eventually stops. Terms in normal form ($\mathcal{F}$) are generated by the following grammar:

$$\mathcal{F} ::= \lambda \mathtt{x}.\mathcal{F} \mid \mathcal{H} \qquad\qquad \mathcal{H} ::= \mathtt{x} \mid \mathcal{H}\mathcal{F}$$

Head-normal forms. The notion of head-normal form ($\mathtt{hnf}$) is the syntactical counter part of the well-known notion of solvability for the λ-calculus, being a term $\mathtt{t}$ solvable iff there is a closing context $\mathtt{C}$ for $\mathtt{t}$ of the shape $(\lambda \mathtt{x}_1 ... \mathtt{x}_n.\Box)\mathtt{t}_1...\mathtt{t}_m$ (called a head-context) such that $\mathtt{C}[\mathtt{t}] =_\beta \mathtt{I}$. 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 term which is in $\mathtt{hnf}$.

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

For example, $\mathtt{t} = (\lambda \mathtt{y}.\mathtt{y})\mathtt{x}\mathtt{D}_{\mathtt{up}}$ is not in $\mathtt{hnf}$ but β-reduces to $\mathtt{x}\mathtt{D}_{\mathtt{up}}$ which is in $\mathtt{hnf}$, thus $\mathtt{t}$ has a $\mathtt{hnf}$. The term $\mathtt{I}$ is solvable while $\mathtt{D}_{\mathtt{up}}\mathtt{D}_{\mathtt{up}}$ is not.

Approximate normal forms. Approximate normal forms [2] are normal forms in an extended calculus. Let $\Lambda\Omega$ be the λ-calculus enriched with a constant Ω, and let $\rightarrow_{\beta\omega}$ be the contextual closure of the β-reduction plus the following two reduction rules:

$$\Omega\mathtt{t} \rightarrow \Omega \qquad\qquad \lambda \mathtt{x}.\Omega \rightarrow \Omega$$

Normal forms of $\Lambda\Omega$ with respect to $\rightarrow_{\beta\omega}$ are defined through the following grammar:

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

Elements generated by the three grammar rules above are called approximate normal forms, $\mathcal{N}$ approximate normal forms and $\mathcal{L}$ approximate normal forms, respectively. E.g. both $\lambda \mathtt{xy}.\mathtt{x}\Omega(\lambda z.\mathtt{yz}\Omega)$ and $\lambda \mathtt{xy}.\mathtt{x}\Omega\Omega$ are approximate normal forms, while $\lambda \mathtt{x}.\Omega$ and $\lambda \mathtt{x}.\mathtt{xy}(\Omega \mathtt{y})$ are not, since they reduce respectively to Ω and $\lambda \mathtt{x}.\mathtt{xy}\Omega$.

Approximants of a term. Approximate normal forms can be ordered by the smallest contextual order $\leq$ such that $\Omega \leq \mathtt{a}$, for all $\mathtt{a}$. By abuse of notation, we write $\mathtt{a} \leq \mathtt{t}$ to compare an approximate normal form $\mathtt{a}$ with a term $\mathtt{t}$ when $\mathtt{t}$ is obtained from $\mathtt{a}$ by replacing all the occurrences of Ω by arbitrary terms. Let $\bigvee$ denote the least upper bound w.r.t. $\leq$. We use the predicate $\uparrow_{i \in I} \mathtt{a}_i$ to state that $\bigvee \{\mathtt{a}_i\}_{i \in I}$ does exist.

The *set of approximants of a term* $\mathtt{t}$ is given by:

$$\mathcal{A}(\mathtt{t}) = \{\mathtt{a} \mid \exists \mathtt{u}\ \mathtt{t} \rightarrow^*_\beta \mathtt{u} \text{ and } \mathtt{a} \leq \mathtt{u}\}$$

It is easy to check that, for every $\mathtt{t}$ and $\mathtt{a}_1, \ldots \mathtt{a}_n \in \mathcal{A}(\mathtt{t})$, $\uparrow_{i \in \{1,\ldots,n\}} \mathtt{a}_i$. Thus for example, $\mathcal{A}(\lambda \mathtt{x}.\mathtt{D}_{\mathtt{up}}(\mathtt{II})) = \{\Omega, \lambda \mathtt{x}.\Omega, \lambda \mathtt{xy}.\Omega, \lambda \mathtt{xy}.\mathtt{y}\}$ and $\bigvee \mathcal{A}(\lambda \mathtt{x}.\mathtt{D}_{\mathtt{up}}(\mathtt{II})) = \lambda \mathtt{xy}.\mathtt{y}$.

In many well-behaved models, it is possible to relate the interpretation of a term to the interpretation of its approximants, the first example being in [35]. We are going to show such a property, known as the *approximation theorem*, for the type assignment system $\mathcal{H}$ presented below.

3. Systems characterizing head normalization

In this section we consider the inhabitation problem with respect to an intersection type system for the λ-calculus that characterizes head-normal forms. The essential feature of this type system that makes its inhabitation problem decidable is the non-idempotency of the intersection. The typing rules are relevant (*i.e.* the weakening is not allowed), and we represent non-idempotent intersections as multisets of types. Moreover, in order to work with a syntax-directed system, we restrict the set of types to the *strict ones* [33].

Definition 3.1.

(1) The set of *types* is defined by the following grammar:

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

where α ranges over a countable set of base types. Multiset types are associated to a finite, possibly empty, set I of indices; the *empty multiset* corresponds to the case $I = \emptyset$ and is simply denoted by $[\,]$. To avoid an excessive number of parentheses, we use for example $\mathtt{A} \to \mathtt{B} \to \sigma$ instead of $\mathtt{A} \to (\mathtt{B} \to \sigma)$.

(2) *Typing environments*, which are also simply called environments, written Γ, Δ, are functions from variables to multiset types, assigning the empty multiset to almost all the variables. We use the symbol $\emptyset$ to denote the empty typing environment. The domain of Γ, written $\mathtt{dom}(\Gamma)$, is the set of variables whose image is different from $[\,]$. Given environments Γ and Δ, $\Gamma + \Delta$ is the environment mapping $\mathtt{x}$ to $\Gamma(\mathtt{x}) \uplus \Delta(\mathtt{x})$, where $\uplus$ denotes multiset union; $+_{i\in I}\Delta_i$ denotes its obvious extension to a non-binary case, where the resulting environment has empty domain in the case $I = \emptyset$. We write $\Gamma \setminus \mathtt{x}$ for 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 $1 \le i \le n$, and $[\,]$ to any other variable.

(3) A *typing judgement* is a triple either of the form $\Gamma \vdash \mathtt{t} : \sigma$ or $\Gamma \vdash \mathtt{t} : \mathtt{A}$. The type system $\mathcal{H}$ is given in Figure 3.

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

$$\frac{(\Delta_i \vdash \mathtt{t} : \sigma_i)_{i\in I}}{+_{i\in I}\Delta_i \vdash \mathtt{t} : [\sigma_i]_{i\in I}}\ (\mathtt{m}) \qquad \frac{\Gamma \vdash \mathtt{t} : \mathtt{A} \to \tau \quad \Delta \vdash \mathtt{u} : \mathtt{A}}{\Gamma + \Delta \vdash \mathtt{tu} : \tau}\ (\to_{\mathtt{E}})$$

Figure 3: The type assignment system $\mathcal{H}$ for the λ-calculus

Rules $(\mathtt{var})$ and $(\to_{\mathtt{I}})$ are self explanatory. Rule $(\mathtt{m})$ can be considered as an auxiliary rule, *i.e.* it has no logical meaning but just collects together different typings for the same term; remark that it cannot be iterated. Rule $(\to_{\mathtt{E}})$ has two premises, the one for $\mathtt{t}$ (resp. $\mathtt{u}$) is called the major (resp. minor) premise. In the case $\mathtt{A} = [\,]$, this rule allows to type a term without giving types to all its subterms, and in particular it allows to type an application whose argument is unsolvable. For example, the judgement $\mathtt{x} : [[\,] \to \alpha] \vdash \mathtt{x}(\mathtt{D_{up}}\mathtt{D_{up}}) : \alpha$ turns to be derivable by taking $I = \emptyset$ in rule $(\mathtt{m})$:

$$\dfrac{\dfrac{}{\mathtt{x} : [[\,] \to \alpha] \vdash \mathtt{x} : [\,] \to \alpha}\,(\mathtt{var}) \qquad \dfrac{}{\emptyset \vdash \mathtt{D_{up}D_{up}} : [\,]}\,(\mathtt{m})}{\mathtt{x} : [[\,] \to \alpha] \vdash \mathtt{x}(\mathtt{D_{up}D_{up}}) : \alpha}\,(\to_{\mathtt{E}})$$

This feature is shared by all the intersection type systems characterizing solvability.

Notation 3.2. We write $\Pi \triangleright \Gamma \vdash_{\mathcal{H}} \mathtt{t} : \sigma$, or simply $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$ when $\mathcal{H}$ is clear from the context, to denote a type derivation Π in system $\mathcal{H}$ with conclusion $\Gamma \vdash \mathtt{t} : \sigma$. We call $\mathtt{t}$ the *subject* of Π. A term $\mathtt{t}$ is said to be $\mathcal{H}$-typable if there exists a derivation $\Pi \triangleright \Gamma \vdash_{\mathcal{H}} \mathtt{t} : \sigma$. By abuse of notation, we omit sometimes the name of the derivation by writing simply $\Gamma \vdash \mathtt{t} : \sigma$ to denote the existence of some derivation with conclusion $\Gamma \vdash \mathtt{t} : \sigma$. We extend these notations to judgements of the shape $\Gamma \vdash \mathtt{t} : \mathtt{A}$, when we want to reason by induction on the structure of a proof.

One important tool is going to be used in the proofs:

Definition 3.3. The *measure* of a derivation Π, written $\mathtt{meas}(\Pi)$, is the number of rule applications in Π, except rule $(\mathtt{m})$.

Note that the definition of the measure of a type derivation reflects the fact that $(\mathtt{m})$ is an auxiliary rule. The notion of measure of a derivation provides an original, combinatorial proof of the fact that typed terms do have $\mathtt{hnf}$. In fact the fundamental *subject reduction* property holds as follows: if $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$ and $\mathtt{t} \to_\beta \mathtt{u}$, then $\Pi' \triangleright \Gamma \vdash \mathtt{u} : \sigma$, with the peculiarity that the measure of Π' is strictly smaller than that of Π whenever the reduction $\mathtt{t} \to_\beta \mathtt{u}$ takes place in a subterm of $\mathtt{t}$ which is *typed in* Π. A formal definition of typed positions follows.

Definition 3.4.

- The set $\mathtt{o}(t)$ of positions 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 position* $\mathtt{C}$.
- Given $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$, the *set* $\mathtt{to}(\Pi) \subseteq \mathtt{o}(\mathtt{t})$ *of typed positions of* $\mathtt{t}$ *in* Π is defined by induction on the structure of Π as follows:
  - $\mathtt{to}(\Pi) = \{\Box\}$ if Π is an instance of the axiom.
  - $\mathtt{to}(\Pi) = \{\Box\} \cup \{\lambda \mathtt{x}.\mathtt{C} \mid \mathtt{C} \in \mathtt{to}(\Pi')\}$ if the last rule of Π is $(\to_{\mathtt{I}})$, its subject is $\lambda \mathtt{x}.\mathtt{u}$ and its premise is Π'.
  - $\mathtt{to}(\Pi) = \{\Box\} \cup \{\mathtt{Cv} \mid \mathtt{C} \in \mathtt{to}(\Pi')\} \cup \{\mathtt{uC} \mid \mathtt{C} \in \mathtt{to}(\Delta)\}$ if the last rule of Π is $(\to_{\mathtt{E}})$, its subject is $\mathtt{uv}$, Π' and Δ are the major and minor premises of Π respectively.
  - $\mathtt{to}(\Pi) = \bigcup_{i \in I} \{\mathtt{C} \mid \mathtt{C} \in \mathtt{to}(\Pi'_i)\}$ if the last rule of Π is $(\mathtt{m})$, and $(\Pi'_i)_{i \in I}$ are its premises. A particular case is when $I = \emptyset$, in which case $\mathtt{to}(\Pi) = \emptyset$. This coincides with the fact that terms typed by an empty $(\mathtt{m})$ rule are semantically untyped.

  We say that *the subterm of* $\mathtt{t}$ *at position* $\mathtt{C}$ *is typed in* Π if $\mathtt{C}$ is a typed position of $\mathtt{t}$ in Π.
- Given $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$ (resp. $\Pi \triangleright \Gamma \vdash \mathtt{t} : \mathtt{A}$), we say that $\mathtt{t}$ is *in* Π*-normal form*, written $\Pi \triangleright_{\mathtt{nf}} \Gamma \vdash \mathtt{t} : \sigma$ (resp. $\Pi \triangleright_{\mathtt{nf}} \Gamma \vdash \mathtt{t} : \mathtt{A}$), if for all $\mathtt{C} \in \mathtt{to}(\Pi)$, $\mathtt{t} = \mathtt{C}[\mathtt{u}]$ implies $\mathtt{u}$ is not a redex.

Example 3.5. Consider the following derivation Π, where $\sigma := [\alpha_0, \alpha_1] \to [\,] \to \tau$:

$$\dfrac{\dfrac{}{\mathtt{x}:[\sigma]\vdash\mathtt{x}:\sigma}\,(\mathtt{var})\quad \dfrac{\dfrac{}{\mathtt{y}:[\alpha_0]\vdash\mathtt{y}:\alpha_0}\,(\mathtt{var})\quad\dfrac{}{\mathtt{y}:[\alpha_1]\vdash\mathtt{y}:\alpha_1}\,(\mathtt{var})}{\mathtt{y}:[\alpha_0,\alpha_1]\vdash\mathtt{y}:[\alpha_0,\alpha_1]}\,(\mathtt{m})}{\dfrac{\mathtt{x}:[\sigma],\mathtt{y}:[\alpha_0,\alpha_1]\vdash\mathtt{xy}:[\,]\to\tau}{\mathtt{x}:[\sigma],\mathtt{y}:[\alpha_0,\alpha_1]\vdash\mathtt{xy}(\mathtt{ID_{up}}):\tau}\,(\to_{\mathtt{E}})\quad\dfrac{}{\emptyset\vdash\mathtt{ID_{up}}:[\,]}\,(\mathtt{m})}\,(\to_{\mathtt{E}})$$

Then,

- $\mathtt{meas}(\Pi) = 5$.
- $\mathtt{to}(\Pi) = \{\Box, \Box\mathtt{y}(\mathtt{ID_{up}}), \mathtt{x}\Box(\mathtt{ID_{up}}), \Box(\mathtt{ID_{up}})\}$.
- $\mathtt{xy}(\mathtt{ID_{up}})$ is in Π-normal form (since its only redex $\mathtt{ID_{up}}$ is untyped).

Theorem 3.6.

(1) *(Subject reduction and expansion)* $\Gamma \vdash \mathtt{t} : \sigma$ *and* $\mathtt{t} =_\beta \mathtt{u}$ *imply* $\Gamma \vdash \mathtt{u} : \sigma$.
(2) *(Characterization)* $\mathtt{t}$ *is* $\mathcal{H}$*-typable if and only if* $\mathtt{t}$ *has* $\mathtt{hnf}$.

Proof. See [9, 7]. In particular, the proof of subject reduction is based on the weighted subject reduction property that $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$ and $\mathtt{t} \to_\beta \mathtt{u}$ imply $\Pi' \triangleright \Gamma \vdash \mathtt{u} : \sigma$, where $\mathtt{meas}(\Pi') \leq \mathtt{meas}(\Pi)$. Moreover, if the reduced redex is typed in Π, then $\mathtt{meas}(\Pi') < \mathtt{meas}(\Pi)$. $\Box$

As a matter of fact, the two properties stated in the theorem above may be proved using a semantic shortcut: in [28] the class of *essential* type systems is introduced, and it is shown that such systems supply a logical description of the linear relational models of the λ-calculus, in the sense of [5]. Since the type system $\mathcal{H}$ is an instance of this class, both statements of Theorem 3.6 are particular cases of the results proved in [28].

3.1. **The key role of approximants.** System $\mathcal{H}$ assigns types to terms without giving types to all their subterms (*cf.* the rule $(\to_{\mathtt{E}})$ in case $\mathtt{A} = [\,]$). So in order to reconstruct all the possible subjects of derivations we need a notation for these untyped subterms, which is supplied by the notion of approximate normal forms introduced in Section 2. As a consequence, some notions previously defined for terms, naturally apply for approximants too. Thus for example, given $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$ or $\Pi \triangleright \Gamma \vdash \mathtt{a} : \mathtt{A}$, $\mathtt{a}$ is said to be in Π-normal form according to Definition 3.4. But remark that *every* typing derivation of an approximant is necessarily in normal form. More precisely, if $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$ or $\Pi \triangleright \Gamma \vdash \mathtt{a} : \mathtt{A}$, then $\mathtt{a}$ is in Π normal form. Quite surprisingly, $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$, $\mathtt{a} \leq \mathtt{t}$ and $\Pi' \triangleright \Gamma \vdash \mathtt{t} : \sigma$ do not imply that $\mathtt{t}$ is in Π' normal form, as the following example shows.

Example 3.7. Let $\mathtt{a} = \mathtt{x}\Omega(\mathtt{xyy}) \leq \mathtt{x}(\mathtt{Ix})(\mathtt{xyy}) = \mathtt{t}$. Let $\Gamma = \mathtt{x} : [\sigma_1, \sigma_2], \mathtt{y} : [\alpha, \alpha]$, where $\sigma_1 = [\,] \to [\alpha] \to \alpha$ and $\sigma_2 = [\alpha] \to [\alpha] \to \alpha$. There are type derivations $\Pi \triangleright \Gamma \vdash \mathtt{a} : \alpha$ (given on the top) and $\Pi' \triangleright \Gamma \vdash \mathtt{t} : \alpha$ (given in the bottom) such that $\mathtt{a}$ is in Π-normal form but $\mathtt{t}$ is not in Π'-normal form.

$$\dfrac{\dfrac{\dfrac{}{\mathtt{x}:[\sigma_1]\vdash \mathtt{x}:\sigma_1}\qquad \dfrac{}{\emptyset\vdash\Omega:[\,]}}{\mathtt{x}:[\sigma_1]\vdash \mathtt{x}\Omega:[\alpha]\to\alpha}\qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\mathtt{x}:[\sigma_2]\vdash \mathtt{x}:\sigma_2}\qquad \Pi_{\mathtt{y}}}{\mathtt{x}:[\sigma_2],\mathtt{y}:[\alpha]\vdash \mathtt{xy}:[\alpha]\to\alpha}\qquad \Pi_{\mathtt{y}}}{\mathtt{x}:[\sigma_2],\mathtt{y}:[\alpha,\alpha]\vdash \mathtt{xyy}:\alpha}}{\mathtt{x}:[\sigma_2],\mathtt{y}:[\alpha,\alpha]\vdash \mathtt{xyy}:[\alpha]}}{\mathtt{x}:[\sigma_1,\sigma_2],\mathtt{y}:[\alpha,\alpha]\vdash \mathtt{x}\Omega(\mathtt{xyy}):\alpha}$$

$$\dfrac{\dfrac{\dfrac{}{\mathtt{x}:[\sigma_2]\vdash \mathtt{x}:\sigma_2}\qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\mathtt{z}:[\alpha]\vdash \mathtt{z}:\alpha}}{\emptyset\vdash \mathtt{I}:[\alpha]\to\alpha}\qquad \Pi_{\mathtt{y}}}{\mathtt{y}:[\alpha]\vdash \mathtt{Iy}:\alpha}}{\mathtt{y}:[\alpha]\vdash \mathtt{Iy}:[\alpha]}}{\mathtt{x}:[\sigma_2],\mathtt{y}:[\alpha]\vdash \mathtt{x}(\mathtt{Iy}):[\alpha]\to\alpha}\qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\mathtt{x}:[\sigma_1]\vdash \mathtt{x}:\sigma_1}\qquad \dfrac{}{\emptyset\vdash \mathtt{y}:[\,]}}{\mathtt{x}:[\sigma_1]\vdash \mathtt{xy}:[\alpha]\to\alpha}\qquad \Pi_{\mathtt{y}}}{\mathtt{x}:[\sigma_1],\mathtt{y}:[\alpha]\vdash \mathtt{xyy}:\alpha}}{\mathtt{x}:[\sigma_1],\mathtt{y}:[\alpha]\vdash \mathtt{xyy}:[\alpha]}}{\mathtt{x}:[\sigma_1,\sigma_2],\mathtt{y}:[\alpha,\alpha]\vdash \mathtt{x}(\mathtt{Iy})(\mathtt{xyy}):\alpha}$$

where $\Pi_{\mathtt{y}}$ denotes the following subderivation

$$\dfrac{\dfrac{}{\mathtt{y}:[\alpha]\vdash \mathtt{y}:\alpha}}{\mathtt{y}:[\alpha]\vdash \mathtt{y}:[\alpha]}$$

Given $\Pi\triangleright\Gamma\vdash\mathtt{t}:\tau$, where $\mathtt{t}$ is in Π-normal form, we denote by $\mathcal{A}(\Pi)$ the least approximate normal form $\mathtt{b}$ of $\mathtt{t}$ such that $\Pi\triangleright\Gamma\vdash\mathtt{b}:\tau$. Formally:

Definition 3.8. Given $\Pi\triangleright_{\mathtt{nf}}\Gamma\vdash\mathtt{t}:\sigma$, the approximate normal form $\mathcal{A}(\Pi)\in\mathcal{A}(\mathtt{t})$, called the *approximant of* Π, is defined by induction on the structure of Π as follows:

- If the last rule of Π is $(\mathtt{var})$, then $\Pi\triangleright\Gamma\vdash\mathtt{x}:\sigma$, and $\mathcal{A}(\Pi):=\mathtt{x}$.
- If the last rule of Π is $(\to_{\mathtt{I}})$, then $\sigma=\mathtt{A}\to\rho$ and $\Pi\triangleright\Gamma\vdash\lambda\mathtt{x}.\mathtt{t}:\mathtt{A}\to\rho$ 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 the last rule of Π is $(\to_{\mathtt{E}})$, $\Pi\triangleright\Gamma+\Delta\vdash\mathtt{vu}:\sigma$ with premises $\Pi'\triangleright\Gamma\vdash\mathtt{v}:\mathtt{A}\to\sigma$ and $\Pi''\triangleright\Delta\vdash\mathtt{u}:\mathtt{A}$, so by induction $\mathcal{A}(\Pi')\in\mathcal{A}(\mathtt{v})$ and
$Ap(\Pi'')\in\mathcal{A}(\mathtt{u})$. Moreover the hypothesis that $\mathtt{vu}$ is in Π-normal form implies that $\mathtt{v}$ is of the form $\mathtt{x}\mathtt{v}_1\ldots\mathtt{v}_n$, so that $\mathcal{A}(\Pi):=\mathcal{A}(\Pi')\mathcal{A}(\Pi'')\in\mathcal{A}(\mathtt{vu})$.
- If the last rule of Π is $(\mathtt{m})$, then $\Pi\triangleright\Gamma\vdash\mathtt{u}:\mathtt{A}$ follows from $(\Pi_i\triangleright\Gamma_i\vdash\mathtt{u}:\sigma_i)_{i\in I}$, then $\mathcal{A}(\Pi)=\bigvee_{i\in I}\mathcal{A}(\Pi_i)$. In fact $\uparrow_{i\in I}\mathcal{A}(\Pi_i)$, since $\mathcal{A}(\Pi_i)\in\mathcal{A}(\mathtt{u})$, for all $i\in I$.

Remark that in the last case the approximate normal form corresponding to the case $I=\emptyset$ is Ω. Coming back to the derivation Π in Example 3.5, we have $\mathcal{A}(\Pi)=\mathtt{xy}\Omega$. More generally, given $\Pi\triangleright_{\mathtt{nf}}\Gamma\vdash\mathtt{t}:\sigma$, the approximant $\mathcal{A}(\Pi)$ can be obtained from $\mathtt{t}$ by replacing all its maximal subterms untyped in Π by Ω.

Simple inductions on Π allow to show the following properties:

Proposition 3.9.

(1) *Let* $\Pi\triangleright\Gamma\vdash\mathtt{a}:\sigma$. *If* $\mathtt{a}\leq\mathtt{b}$ *(resp.* $\mathtt{a}\leq\mathtt{t}$*) then there exists* Π' *such that* $\Pi'\triangleright\Gamma\vdash\mathtt{b}:\sigma$ *(resp.* $\Pi'\triangleright_{\mathtt{nf}}\Gamma\vdash\mathtt{t}:\sigma$*) and* $\mathcal{A}(\Pi')=\mathcal{A}(\Pi)$.
(2) *If* $\Pi\triangleright\Gamma\vdash\mathtt{a}:\sigma$ *or* $\Pi\triangleright_{\mathtt{nf}}\Gamma\vdash\mathtt{t}:\sigma$, *then there exists* $\Pi'\triangleright\Gamma\vdash\mathcal{A}(\Pi):\sigma$ *and* $\mathcal{A}(\Pi')=\mathcal{A}(\Pi)$.

Proposition 3.9.2 ensures the completeness of approximants, in the sense that all the typings having subject $\mathtt{t}$ may also type some approximant of $\mathtt{t}$.

On the other hand, it is easy to see that if $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$ and $\mathtt{t}$ is in Π-normal form, then $\mathcal{A}(\Pi) \leq \mathtt{t}$. Summing up, we get a new proof of the approximation theorem for the relational λ-model associated to system $\mathcal{H}$, in the sense of [28]:

Theorem 3.10. *The relational λ-model induced by system $\mathcal{H}$ satisfies the approximation theorem, i.e. the interpretation of a term is the union of the interpretations of its approximants.*

We are ready to prove that the set of approximate normal forms solving any given instance of the inhabitation problem is finite. This property allows us to design a complete inhabitation algorithm, *i.e.* one which supplies all solutions. First, we will prove that a derivation in normal form enjoys a sort of subformula property. Let us define the *subtype* relation as the transitive closure of the following one:

- σ is a subtype of σ.
- $\mathtt{A}$ is a subtype of $\mathtt{A}$.
- $\mathtt{A}$ and τ are subtypes of $\mathtt{A} \to \tau$.
- σ_i is a subtype of $[\sigma_i]_{i\in I}$ for all $i \in I$.

Finally, $\mathtt{A}$ is a subtype of Γ if $\mathtt{x} : \mathtt{A} \in \Gamma$.

Lemma 3.11. *Let $\mathtt{a}$ be an approximate normal form. Let $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$. For every subderivation of Π of the shape $\Pi' \triangleright \Gamma' \vdash \mathtt{b} : \tau$, τ is either a subtype of σ or a subtype of Γ. Moreover, if $\mathtt{a}$ is an $\mathcal{L}$ approximate normal form, then σ is a subtype of Γ.*

Proof. The proof is by induction on Π. We do not consider the case of rule $(\mathtt{m})$ since the derivation ends in a type σ.

If Π ends with $(\mathtt{var})$, then $\mathtt{a} = \mathtt{x}$ and $\Pi \triangleright \mathtt{x} : [\sigma] \vdash \mathtt{x} : \sigma$, and the property is obviously true, being σ a subtype of itself. Moreover, $\mathtt{x}$ is an $\mathcal{L}$ approximate normal form, and σ is a subtype of $[\sigma]$, thus a subtype of $\mathtt{x} : [\sigma]$.

If Π ends with $(\to_{\mathtt{I}})$, then $\mathtt{a} = \lambda\mathtt{x}.\mathtt{a}'$, $\sigma = \mathtt{A} \to \tau$, and Π has the following shape:

$$\frac{\Gamma, \mathtt{x} : \mathtt{A} \vdash \mathtt{a}' : \tau}{\Gamma \vdash \lambda\mathtt{x}.\mathtt{a}' : \mathtt{A} \to \tau}\ (\to_{\mathtt{I}})$$

By the *i.h.* the property holds for the derivation with subject $\mathtt{a}'$. Since both $\mathtt{A}$ and τ are subtypes of $\mathtt{A} \to \tau$, we conclude by transitivity of the subtype relation. The moreover statement does not apply to abstractions.

If Π ends with $(\to_{\mathtt{E}})$, then $\mathtt{a} = \mathtt{a}_0\mathtt{a}_1$, where $\mathtt{a}$ and $\mathtt{a}_0$ are $\mathcal{L}$ approximate normal forms, and $\mathtt{a}_1$ is an approximate normal form. Then Π has the following shape:

$$\frac{\Gamma \vdash \mathtt{a}_0 : \mathtt{A} \to \tau \qquad \dfrac{(\Delta_i \vdash \mathtt{a}_1 : \rho_i)_{i\in I}}{\Delta \vdash \mathtt{a}_1 : \mathtt{A}}\ (\mathtt{m})}{\Gamma + \Delta \vdash \mathtt{a}_0\mathtt{a}_1 : \tau}\ (\to_{\mathtt{E}})$$

where $\Delta = +_{i\in I}\Delta_i$ and $\mathtt{A} = [\rho_i]_{i\in I}$. We have that ρ_i is a subtype of $\mathtt{A}$, which in turn is a subtype of $\mathtt{A} \to \tau$, and by the *i.h.* $\mathtt{A} \to \tau$ is a subtype of Γ, thus a subtype of $\Gamma + \Delta$. For any other subderivation we conclude by the *i.h.* and the fact that each $\mathtt{x} : \mathtt{B}_i \in \Delta_i$ implies $\mathtt{x} : \mathtt{B}_i \in \Gamma + \Delta$. $\square$

This lemma gives to system $\mathcal{H}$ a quantitative flavour. In fact, define the *degree* of (Γ, σ), written $d(\Gamma, \sigma)$, to be the sum of the cardinalities of all the multisets in Γ and σ. The following property holds.

Property 3.12. Let $\mathtt{a}$ be an approximate normal form. Let $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$. Then the number of typed positions of $\mathtt{a}$ in Π which correspond to both bound and free variables (*cf.* Def. 3.4) is bounded by $d(\Gamma, \sigma)$.

Proof. Consider a (free or bound) variable $\mathtt{x}$ of $\mathtt{t}$, and let $(\mathtt{x} : [\sigma_i] \vdash \mathtt{x} : \sigma_i)_{i \in I}$ be all the axioms with subject $\mathtt{x}$ in Π. By the hygiene convention, $\mathtt{x}$ is either free or bound in $\mathtt{t}$. If $\mathtt{x}$ is free in $\mathtt{t}$, then $\mathtt{x} : [\sigma_i]_{i \in I} \in \Gamma$, and since every axiom corresponds to a typed position of $\mathtt{x}$ in Π, the proof is given. If $\mathtt{x}$ is bound in $\mathtt{t}$, then there is a subderivation of Π ending with rule $(\rightarrow_{\mathtt{I}})$, with premise $\Gamma', \mathtt{x} : [\sigma_i]_{i \in I} \vdash \mathtt{a}' : \tau$ and conclusion $\Gamma' \vdash \lambda \mathtt{x}.\mathtt{a}' : [\sigma_i]_{i \in I} \rightarrow \tau$. By the subformula property (Lemma 3.11), $[\sigma_i]_{i \in I} \rightarrow \tau$ is a subformula of either Γ or σ, so the cardinality of I, which corresponds to the number of bound typed positions of $\mathtt{x}$ in Π, is smaller or equal to $d(\Gamma, \sigma)$. $\square$

This property has a very important corollary.

Corollary 3.13. *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. By Property 3.12, if $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$, then the number of typed positions of variables of $\mathtt{a}$ in Π is bounded by $d(\Gamma, \sigma)$. By definition of $\mathcal{A}(\Pi)$, Ω is the only untyped subterm of $\mathtt{a}$, then $d(\Gamma, \sigma)$ gives an upper bound for the number of *all* positions of variables of $\mathtt{a}$. But the number of approximate normal forms with a bounded number of variable positions is finite, so the number of such approximate normal forms $\mathtt{a}$ is finite too. $\square$

3.2. **The inhabitation algorithm.** An instance of the inhabitation problem is a pair (Γ, σ), and solving such an instance consists in deciding whether there exists a term $\mathtt{t}$ such that $\Gamma \vdash \mathtt{t} : \sigma$ is derivable.

We find convenient to present inhabitation algorithms as deductive systems, proving judgements of the form $\mathtt{a} \Vdash \mathtt{T}(\Gamma, \sigma)$, whose intended meaning is that there exists a derivation $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$ such that $\mathtt{a} = \mathcal{A}(\Pi)$. In this spirit, a *run* of the algorithm on the input (Γ, σ) is a proof of the judgement $\mathtt{a} \Vdash \mathtt{T}(\Gamma, \sigma)$, for some approximate normal form $\mathtt{a}$, which is called the *output* of that run. Hence, running the algorithm on (Γ, σ) corresponds to searching a proof of $\mathtt{a} \Vdash \mathtt{T}(\Gamma, \sigma)$, for some (unknown) $\mathtt{a}$. As usual, when writing $\mathtt{a} \Vdash \mathtt{T}(\Gamma, \sigma)$ we mean that there exists a proof of that judgement.

When facing a problem (Γ, σ), in case $\sigma = \mathtt{B} \rightarrow \tau$, there exist two possibilities. The first one consists in guessing that the last rule used in a derivation solving the problem is $(\rightarrow_{\mathtt{I}})$, and trying to solve the problem $(\Gamma + \mathtt{x} : \mathtt{B}, \tau)$ for a fresh variable $\mathtt{x}$. The second one consists in guessing that the last rule is $(\rightarrow_{\mathtt{E}})$, choosing a head variable among those in the domain of Γ having a type of the form $\mathtt{A}_1 \rightarrow \ldots \rightarrow \mathtt{A}_n \rightarrow \sigma$ (if any), and trying to construct the arguments $\mathtt{a}_1, ..., \mathtt{a}_n$ using the resources still avaliable in Γ.

This gives the algorithm in Figure 4. The two alternatives described above are implemented by rule ($\mathtt{Abs}$) and rule ($\mathtt{Head}$) respectively. Once the head variable has been chosen, by ($\mathtt{Head}$), rule ($\mathtt{Head}_{>0}$) allows to construct its argument one by one, using rule ($\mathtt{Union}$). All the branches of any given run of the algorithm stop on a ($\mathtt{Head}_0$) rule, or on a ($\mathtt{Union}$) rule with $I = \emptyset$, or fail (no rule applies).

More precisely, the algorithm uses three forms of judgements:

- $\mathtt{a} \Vdash \mathtt{TI}(\Gamma, \mathtt{A})$ meaning that the approximate normal form $\mathtt{a}$ solves the inhabitation problem $(\Gamma, \mathtt{A})$.
- $\mathtt{a} \Vdash \mathtt{T}(\Gamma, \sigma)$, meaning that the $\mathcal{N}$ approximate normal form $\mathtt{a}$ solves the inhabitation problem (Γ, σ).
- $\mathtt{a} \Vdash \mathtt{H}^{\mathtt{x}:[\tau]}(\Gamma, \sigma)$ meaning that the $\mathcal{L}$ approximate normal form $\mathtt{a}$, headed by a variable $\mathtt{x}$ of type τ, solves the inhabitation problem (Γ, σ).

We prove termination, soundness and completeness of the algorithm presented in Figure 4. Completeness, in particular, is obtained thanks to a non-deterministic behaviour: given an environment Γ and a type σ, different runs can be chosen, each one constructing a judgement of the form $\mathtt{a} \Vdash \mathtt{T}(\Gamma, \sigma)$. By collecting all such possible runs we recover *all* the approximate normal forms $\mathtt{a}$ such that there exists a derivation $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$, with $\mathtt{a} = \mathcal{A}(\Pi)$.

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

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

$$\frac{\Gamma = \Gamma_1 + \Gamma_2 \quad \mathtt{a} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \mathtt{A}_n \to \mathtt{B} \to \tau]}(\Gamma_1, \mathtt{B} \to \tau) \quad \mathtt{b} \Vdash \mathtt{TI}(\Gamma_2, \mathtt{B}) \quad n \geq 0}{\mathtt{ab} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \mathtt{A}_n \to \mathtt{B} \to \tau]}(\Gamma, \tau)}\ (\mathtt{Head}_{>0})$$

$$\frac{}{\mathtt{x} \Vdash \mathtt{H}^{\mathtt{x}:[\tau]}(\emptyset, \tau)}\ (\mathtt{Head}_0)$$

$$\frac{\mathtt{a} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \mathtt{A}_n \to \tau]}(\Gamma, \tau)}{\mathtt{a} \Vdash \mathtt{T}(\Gamma + \mathtt{x} : [\mathtt{A}_1 \to \ldots \mathtt{A}_n \to \tau], \tau)}\ (\mathtt{Head})$$

Figure 4: The inhabitation algorithm for system $\mathcal{H}$

Some comments on the rules of the algorithm are in order. Rule $(\mathtt{Abs})$ looks for an $\mathcal{N}$ approximate normal form, when the input type is an arrow. Rule $(\mathtt{Union})$ applies the approximation theorem 3.10; notice that in the particular case $I = \emptyset$ it gives $\Omega \Vdash \mathtt{TI}(\emptyset, [\,])$, where $\emptyset$ denotes the environment having empty domain. Rule $(\mathtt{Head}_0)$ is self-explaining, rule $(\mathtt{Head}_{>0})$ is based on the property that, if an approximate normal form is an application $\mathtt{ab}$, then $\mathtt{a}$ must be an $\mathcal{L}$ approximate normal form, and its type is a subtype of a type assigned to its head variable. Rule $(\mathtt{Head})$ selects a head variable and launches the search for a suitable $\mathcal{L}$ approximate normal form.

Example 3.14.

(1) Let $\Gamma = \emptyset$ and $\sigma = [[\alpha] \to \alpha] \to [\alpha] \to \alpha$. Given input (Γ, σ), the algorithm succeeds with two different runs, generating respectively the following deduction trees:

(1.1)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathtt{x} \Vdash \mathtt{H}^{\mathtt{x}:[[\alpha]\to\alpha]}(\emptyset, [\alpha] \to \alpha)}\,(\mathtt{Head}_0) \qquad \dfrac{\dfrac{\dfrac{}{\mathtt{y} \Vdash \mathtt{H}^{\mathtt{y}:[\alpha]}(\emptyset, \alpha)}\,(\mathtt{Head}_0)}{\mathtt{y} \Vdash \mathtt{T}(\mathtt{y} : [\alpha], \alpha)}\,(\mathtt{Head})}{\mathtt{y} \Vdash \mathtt{TI}(\mathtt{y} : [\alpha], [\alpha])}\,(\mathtt{Union})}{\mathtt{xy} \Vdash \mathtt{H}^{\mathtt{x}:[[\alpha]\to\alpha]}(\mathtt{y} : [\alpha], \alpha)}\,(\mathtt{Head}_{>0})}{\mathtt{xy} \Vdash \mathtt{T}(\mathtt{x} : [[\alpha] \to \alpha], \mathtt{y} : [\alpha], \alpha)}\,(\mathtt{Head})}{\lambda\mathtt{y}.\mathtt{xy} \Vdash \mathtt{T}(\mathtt{x} : [[\alpha] \to \alpha], [\alpha] \to \alpha)}\,(\mathtt{Abs})}{\lambda\mathtt{xy}.\mathtt{xy} \Vdash \mathtt{T}(\emptyset, \sigma)}\,(\mathtt{Abs})$$

(1.2)

$$\dfrac{\dfrac{\dfrac{}{\mathtt{x} \Vdash \mathtt{H}^{\mathtt{x}:[[\alpha]\to\alpha]}(\emptyset, [\alpha] \to \alpha)}\,(\mathtt{Head}_0)}{\mathtt{x} \Vdash \mathtt{T}(\mathtt{x} : [[\alpha] \to \alpha], [\alpha] \to \alpha)}\,(\mathtt{Head})}{\lambda\mathtt{x}.\mathtt{x} \Vdash \mathtt{T}(\emptyset, \sigma)}\,(\mathtt{Abs})$$

Remark that the type σ does not correspond to the simple type $(\alpha \to \alpha) \to \alpha \to \alpha$, which represent the data types of Church numerals and so has an infinite number of inhabitants. In system $\mathcal{H}$ there is no common type for all the Church numerals, since the numeral $\underline{n}$ has type $[\underbrace{[\alpha] \to \alpha, ..., [\alpha] \to \alpha}_{n}] \to [\alpha] \to \alpha$ (among others, all of degree n).

(2) Let $\Gamma = \emptyset$ and $\sigma = [[\,] \to \alpha] \to \alpha$. Given input (Γ, σ), a succesful run of the algorithm is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathtt{x} \Vdash \mathtt{H}^{\mathtt{x}:[[\,]\to\alpha]}(\emptyset, [\,] \to \alpha)}\,(\mathtt{Head}_0)) \qquad \dfrac{}{\Omega \Vdash \mathtt{TI}(\emptyset, [\,])}\,(\mathtt{Union})}{\mathtt{x}\Omega \Vdash \mathtt{H}^{\mathtt{x}:[[\,]\to\alpha]}(\emptyset, \alpha)}\,(\mathtt{Head}_{>0})}{\mathtt{x}\Omega \Vdash \mathtt{T}(\mathtt{x} : [[\,] \to \alpha], \alpha}\,(\mathtt{Head})}{\lambda\mathtt{x}.\mathtt{x}\Omega \Vdash \mathtt{T}(\emptyset, \sigma)}\,(\mathtt{Abs})$$

(3) Given input $(\emptyset, [\alpha_1] \to \alpha_2)$, where each α_i is a base type, then the unique possible run consists in starting with rule ($\mathtt{Abs}$), then rule ($\mathtt{Head}$), then the algorithm stops since no other rule can be applied. Then $\mathtt{T}(\emptyset, [\alpha_1] \to \alpha_2)$ is empty.

It follows from Example 3.14.1 that the algorithm is not an obvious extension of the classical inhabitation algorithm for simple types [3, 16], and cannot be conservative with respect to it, since the two algorithms take input types belonging to different grammars.

Definition 3.15. In order to show that the inhabitation algorithm terminates, we define a *measure on types and environments*, as follows:

$$\begin{array}{lcllcl} \#(\alpha) & = & 1 & \#([\sigma_i]_{i\in I}) & = & \sum_{i\in I} \#(\sigma_i) + 1 \\ \#(\mathtt{A} \to \rho) & = & \#(\mathtt{A}) + \#(\rho) + 1 & \#(\Gamma) & = & \sum_{\mathtt{x}\in\mathtt{dom}(\Gamma)} \#(\Gamma(\mathtt{x})) \end{array}$$

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

$\#(\mathtt{T}(\Gamma, \rho)) = \#(\Gamma) + \#(\rho)$

$\#(\mathtt{TI}(\Gamma, \mathtt{A})) = \#(\Gamma) + \#(A)$

$\#(\mathtt{H}^{\mathtt{x}:[\mathtt{A}_1\to\ldots\mathtt{A}_n\to\tau]}(\Gamma, \tau)) = \#(\Gamma) +_{i=1\ldots n} \#(\mathtt{A}_i) + n$

Lemma 3.16 (Termination)**.** *The inhabitation algorithm for system $\mathcal{H}$ terminates.*

Proof. Given input (Γ, σ), every run of the algorithm is a deduction tree labelled with the rules of the algorithm (as in Example 3.14), where a node n' is a child of n iff there exists an instance of a rule having n as conclusion and n' as one of their premises. We associate to every rule application the measure of its conclusion, and we will prove that at every recursive call the measure decreases. The proof is by induction on the deduction tree.

If the last applied rule is $(\mathtt{Head}_0)$, then the proof is trivial. If the last rule is $(\mathtt{Abs})$ or $(\mathtt{Union})$, then the proof is straightforward by application of the *i.h.* and the definition of the measure. So let us assume that the last rule is $(\mathtt{Head})$, with premise $\mathtt{a} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \mathtt{A}_n \to \tau]}(\Gamma, \tau)$ and conclusion $\mathtt{a} \Vdash \mathtt{T}(\Gamma + \mathtt{x} : [\mathtt{A}_1 \to \ldots \mathtt{A}_n \to \tau], \tau)$. Then the measure of the premise is $\#(\Gamma) +_{i=1\ldots n} \#(\mathtt{A}_i) + n$, while that of the conclusion is $\#(\Gamma) +_{i=1\ldots n} \#(\mathtt{A}_i) + n + 2 \times \#(\tau)$, which is strictly bigger. Let us assume that the last rule be $(\mathtt{Head}_{>0})$, with premises $\mathtt{a} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \mathtt{A}_n \to \mathtt{B} \to \tau]}(\Gamma_1, \mathtt{B} \to \tau)$ and $\mathtt{b} \Vdash \mathtt{TI}(\Gamma_2, \mathtt{B})$, and conclusion $\mathtt{ab} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \mathtt{A}_n \to \mathtt{B} \to \tau]}(\Gamma, \tau)$. The measure of the conclusion is $\#(\Gamma) +_{i=1\ldots n} \#(\mathtt{A}_i) + \#(\mathtt{B}) + n + 1$, while the measures of the premises are respectively $\#(\Gamma_1) + +_{i=1\ldots n} \#(\mathtt{A}_i) + n$ and $\#(\Gamma_2) + \#(\mathtt{B})$. Also taking into account that either Γ_1 or Γ_2 can be empty, both the premises have a measure strictly smaller than then measure of the conclusion.

Hence any particular run of the algorithm terminates.

Since any instance $\mathtt{T}(\Gamma, \sigma)$ (resp. $\mathtt{TI}(\Gamma, \mathtt{A})$ or $\mathtt{H}^{\mathtt{x}:\sigma}(\Gamma, \tau)$) gives rise to a finite number of possible runs, as in Example 3.14, we conclude. □

The definition below gives the intended meaning of the three components of the inhabitation algorithm:

Definition 3.17.

- $[\![\mathtt{T}(\Gamma, \sigma)]\!] = \{\mathtt{a} \mid \exists \Pi.\ \Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma \text{ and } \mathtt{a} = \mathcal{A}(\Pi)\}$,
- $[\![\mathtt{TI}(\Gamma, \mathtt{A})]\!] = \{\mathtt{a} \mid \exists \Pi.\ \Pi \triangleright \Gamma \vdash \mathtt{a} : \mathtt{A} \text{ and } \mathtt{a} = \mathcal{A}(\Pi)\}$,
- $[\![\mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to A_n \to \tau]}(\Gamma, \tau)]\!] = \{\mathtt{xa}_1 \ldots \mathtt{a}_n \mid \exists \Pi.\ \Pi \triangleright \Gamma + \mathtt{x} : [\mathtt{A}_1 \to \ldots \to \mathtt{A}_n \to \tau] \vdash \mathtt{xa}_1 ... \mathtt{a}_n : \tau,\ \mathtt{xa}_1 ... \mathtt{a}_n = \mathcal{A}(\Pi),\ \Gamma = +_{i=1,\ldots,n} \Gamma_i \text{ and } \Gamma_i \vdash \mathtt{a}_i : \mathtt{A}_i\}$.

Remark that $[\![\mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to \mathtt{A}_n \to \tau]}(\Gamma, \tau)]\!] \subseteq [\![\mathtt{T}(\Gamma + \mathtt{x} : [\mathtt{A}_1 \to \ldots \to \mathtt{A}_n \to \tau], \tau)]\!]$. Moreover, if $\mathtt{a} \in [\![\mathtt{T}(\Gamma, \sigma)]\!]$ and $\mathtt{a} = \mathtt{xa}_1 ... \mathtt{a}_n$, then $\mathtt{a} \in [\![\mathtt{H}^{\mathtt{x}:[T_\tau]}(\Gamma', \tau)]\!]$ for some type $T_\tau = \mathtt{A}_1 \to \ldots \to \mathtt{A}_n \to \tau$ such that $\Gamma = \Gamma' + \mathtt{x} : [T_\tau]$.

Soundness and completeness of the inhabitation algorithm follow from the following Lemma, relating typing derivations in system $\mathcal{H}$ and runs of the algorithm:

Lemma 3.18. *Let* $\mathtt{a}$ *be an approximate normal form,* Γ *a typing environment and* σ *a type. Then* $\mathtt{a} \Vdash \mathtt{T}(\Gamma, \sigma) \Leftrightarrow \mathtt{a} \in [\![\mathtt{T}(\Gamma, \sigma)]\!]$.

Proof. As for $(\Rightarrow)$, we prove the following:

a) $\mathtt{a} \Vdash \mathtt{T}(\Gamma, \sigma) \Rightarrow \mathtt{a} \in [\![\mathtt{T}(\Gamma, \sigma)]\!]$.
b) $\mathtt{a} \Vdash \mathtt{TI}(\Gamma, \mathtt{A}) \Rightarrow \mathtt{a} \in [\![\mathtt{TI}(\Gamma, \mathtt{A})]\!]$.
c) $\mathtt{a} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to A_n \to \tau]}(\Gamma, \tau) \Rightarrow \mathtt{a} \in [\![\mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to \mathtt{A}_n \to \tau]}(\Gamma, \tau)]\!]$.

These three properties are proved by mutual induction on the definitions of the jugdments $\mathtt{a} \Vdash \mathtt{T}(\Gamma, \sigma)$, $\mathtt{a} \Vdash \mathtt{TI}(\Gamma, \mathtt{A})$ and $\mathtt{a} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to A_n \to \tau]}(\Gamma, \tau)$.

a) There are two cases:
   - If $\lambda \mathtt{x}.\mathtt{a} \Vdash \mathtt{T}(\Gamma, \mathtt{A} \to \tau)$ follows from $\mathtt{a} \Vdash \mathtt{T}(\Gamma + (\mathtt{x} : \mathtt{A}), \tau)$ by $(\mathtt{Abs})$, then we conclude by the *i.h.* (a) and by an application of rule $(\to_{\mathtt{I}})$.

- If $\mathtt{a} \Vdash \mathtt{T}(\Gamma + (\mathtt{x} : [\mathtt{A}_1 \to \ldots \to \mathtt{A}_n \to \tau]), \tau)$ follows from $\mathtt{a} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to A_n \to \tau]}(\Gamma, \tau)$, then we conclude immediately by the *i.h.* (c). Note that in this case $\mathtt{a}$ is of the form $\mathtt{x}\mathtt{a}_1 \ldots \mathtt{a}_n$.

b) If $\bigvee_{i \in I} \mathtt{a}_i \Vdash \mathtt{TI}(+_{i \in I}\Gamma_i, [\sigma_i]_{i \in I})$ follows from $(\mathtt{a}_i \Vdash \mathtt{T}(\Gamma_i, \sigma_i))_{i \in I}$ and $\uparrow_{i \in I} \mathtt{a}_i$ by $(\mathtt{Union})$, then by the *i.h.* (a), for all $i \in I$ there exists $(\Pi_i \triangleright \Gamma_i \vdash \mathtt{a}_i : \sigma_i)_{i \in I}$ such that $\mathtt{a}_i = \mathcal{A}(\Pi_i)$. By Proposition 3.9.1, for all $i \in I$ there exists $\Pi'_i \triangleright \Gamma_i \vdash \bigvee_{i \in I} \mathtt{a}_i : \sigma_i$, with $\mathcal{A}(\Pi'_i) = \mathcal{A}(\Pi_i)$. By rule $(\mathtt{m})$ we obtain $\Pi_{\mathtt{m}} \triangleright +_{i \in I}\Gamma_i \vdash \bigvee_{i \in I} \mathtt{a}_i : [\sigma_i]_{i \in I}$. We conclude by observing that $\bigvee_{i \in I} \mathtt{a}_i = \bigvee_{i \in I} \mathcal{A}(\Pi_i) = \bigvee_{i \in I} \mathcal{A}(\Pi'_i) = \mathcal{A}(\Pi_{\mathtt{m}})$.

c) There are two cases:
- If $\mathtt{x} \Vdash \mathtt{H}^{\mathtt{x}:[\tau]}(\Gamma, \tau)$ is an axiom, then $\mathtt{x} = \mathcal{A}(\Pi)$ where Π is the unique proof of $\mathtt{x} : [\tau] \vdash \mathtt{x} : \tau$.
- If $\mathtt{ab} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to A_n \to \tau]}(\Gamma + \Delta, \tau)$ follows from $\mathtt{a} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to \mathtt{A}_{n-1} \to \mathtt{A}_n \to \tau]}(\Gamma, \mathtt{A}_n \to \tau)$ and $\mathtt{b} \Vdash \mathtt{TI}(\Delta, \mathtt{A}_n)$, then:
  - by the *i.h.*(c), $\mathtt{a} = \mathtt{x}\mathtt{a}_1 \ldots \mathtt{a}_{n-1}$, there exists $\Pi_1 \triangleright \Gamma + \mathtt{x} : [\mathtt{A}_1 \to \ldots \to A_n \to \tau] \vdash \mathtt{a} : \mathtt{A}_n \to \tau$ such that $\mathtt{a} = \mathcal{A}(\Pi_1)$.
  - by the *i.h.*(b) there exists $\Pi_2 \triangleright \Delta \vdash \mathtt{b} : \mathtt{A}_n$ such that $\mathtt{b} = \mathcal{A}(\Pi_2)$.

  By using Rule $(\to_{\mathtt{E}})$, we obtain a proof $\Pi \triangleright \Gamma + \Delta + \mathtt{x} : [\mathtt{A}_1 \to \ldots \to A_n \to \tau] \vdash \mathtt{x}\mathtt{a}_1 \ldots \mathtt{a}_{n-1}\mathtt{b} : \tau$, with $\mathcal{A}(\Pi) = \mathtt{x}\mathtt{a}_1 \ldots \mathtt{a}_{n-1}\mathtt{b} = \mathtt{ab}$, and we are done.

Concerning $(\Leftarrow)$, we prove the following:

a) $\mathtt{a} \Vdash \mathtt{T}(\Gamma, \sigma) \Leftarrow \mathtt{a} \in [\![\mathtt{T}(\Gamma, \sigma)]\!]$.
b) $\mathtt{a} \Vdash \mathtt{TI}(\Gamma, \mathtt{A}) \Leftarrow \mathtt{a} \in [\![\mathtt{TI}(\Gamma, \mathtt{A})]\!]$.
c) $\mathtt{a} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to A_n \to \tau]}(\Gamma, \tau) \Leftarrow \mathtt{a} \in [\![\mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to A_n \to \tau]}(\Gamma, \tau)]\!]$.

We proceed by induction on typing derivations $\Pi \triangleright \Gamma \vdash \mathtt{a} : \sigma$ or $\Pi \triangleright \Gamma \vdash \mathtt{a} : \mathtt{A}$ such that $\mathtt{a} = \mathcal{A}(\Pi)$. For such derivations, we show that $\mathtt{a}$ is generated by a suitable run of the inhabitation algorithm.

- Case $(\mathtt{var})$. Then there is a type derivation of the following form:

$$\frac{}{\Pi \triangleright \mathtt{x} : [\rho] \vdash \mathtt{x} : \rho}$$

In this case $\mathcal{A}(\Pi) = \mathtt{x}$ belongs to $[\![\mathtt{T}(\mathtt{x} : [\rho], \rho)]\!]$ and to $[\![\mathtt{H}^{\mathtt{x}:[\rho]}(\emptyset, \rho)]\!]$, so we need to show a) and c) respectively.

By rule $(\mathtt{Head}_0)$ we have $\mathtt{x} \Vdash \mathtt{H}^{\mathtt{x}:[\rho]}(\emptyset, \rho)$, then by rule $(\mathtt{Head})$ we also conclude $\mathtt{x} \Vdash \mathtt{T}(\mathtt{x} : [\rho], \rho)$.

- Case $(\to_{\mathtt{I}})$. Then there is a type derivation of the following form:

$$\frac{\Pi' \triangleright \Gamma \vdash \mathtt{b} : \tau}{\Pi \triangleright \Gamma \setminus \mathtt{x} \vdash \lambda\mathtt{x}.\mathtt{b} : \Gamma(\mathtt{x}) \to \tau}\ (\to_{\mathtt{I}})$$

with $\lambda\mathtt{x}.\mathtt{b} = \mathcal{A}(\Pi)$, and hence $\mathtt{b} = \mathcal{A}(\Pi')$. In this case, $\lambda\mathtt{x}.\mathtt{b} \in [\![\mathtt{T}(\Gamma \setminus \mathtt{x}, \Gamma(\mathtt{x}) \to \tau)]\!]$ (and thus $\mathtt{b} \in [\![\mathtt{T}(\Gamma, \tau)]\!]$), so that we need to show a). By the *i.h.*(a) we have that $\mathtt{b} \Vdash \mathtt{T}(\Gamma, \tau)$, and we conclude that $\lambda\mathtt{x}.\mathtt{b} \Vdash \mathtt{T}(\Gamma \setminus \mathtt{x}, \Gamma(\mathtt{x}) \to \tau))$ by rule $(\mathtt{Abs})$.

- Case $(\to_{\mathtt{E}})$. Then $\mathtt{a} = \mathtt{x}\mathtt{b}_1 \ldots \mathtt{b}_n$ for some $n > 0$ and the typing derivation Π has the following form:

$$\frac{\Pi^1 \triangleright \Gamma_1 \vdash \mathtt{x}\mathtt{b}_1 \ldots \mathtt{b}_{n-1} : \mathtt{A}_n \to \tau \qquad \Pi^2 \triangleright \Gamma_2 \vdash \mathtt{b}_n : \mathtt{A}_n}{\Gamma \vdash \mathtt{x}\mathtt{b}_1 \ldots \mathtt{b}_n : \tau}\ (\to_{\mathtt{E}})$$

where $\Gamma = \Gamma_1 + \Gamma_2$. We analyse two cases:

- $\mathtt{a} \in [\![\mathtt{H}^{\mathtt{x}:[T_\tau]}(\Gamma', \tau)]\!]$, where $\Gamma = \Gamma' + \mathtt{x} : [T_\tau]$. Then $\Gamma_1 = \Gamma'_1 + \mathtt{x} : [T_\tau]$, and $\Gamma' = \Gamma'_1 + \Gamma_2$. Moreover, $\mathtt{xb}_1...\mathtt{b}_{n-1} = \mathcal{A}(\Pi^1)$ and $\mathtt{b}_n = \mathcal{A}(\Pi^2)$. By the *i.h.*(c), $\mathtt{xb}_1...\mathtt{b}_{n-1} \Vdash \mathtt{H}^{\mathtt{x}:[T_\tau]}(\Gamma'_1, \mathtt{A}_n \to \tau)$ and, by the *i.h.*(b) $\mathtt{b}_n \Vdash \mathtt{TI}(\Gamma_2, \mathtt{A}_n)$, so we get $\mathtt{a} \Vdash \mathtt{H}^{\mathtt{x}:[T_\tau]}(\Gamma', \tau)$ by rule $(\mathtt{Head}_{>0})$.
- $\mathtt{a} \in [\![\mathtt{T}(\Gamma, \tau)]\!]$. Then $\mathtt{a} \in [\![\mathtt{H}^{\mathtt{x}:[T_\tau]}(\Gamma', \tau)]\!]$ for some type T_τ such that $\Gamma = \Gamma' + \mathtt{x} : [T_\tau]$ as remarked after Definition 3.17. Then we conclude $\mathtt{a} \Vdash \mathtt{H}^{\mathtt{x}:[T_\tau]}(\Gamma', \tau)$ by the previous point and $\mathtt{a} \Vdash \mathtt{T}(\Gamma, \tau)$ by rule $(\mathtt{Head})$.

- Case $(\mathtt{m})$: If $\mathtt{a} \in [\![\mathtt{TI}(\Gamma, \mathtt{A})]\!]$, then $\Pi \triangleright \Gamma \vdash \mathtt{a} : \mathtt{A}$, so we have a derivation of the following form:
$$\frac{(\Pi_i \triangleright \Delta_i \vdash \mathtt{a} : \sigma_i)_{i \in I}}{\Pi \triangleright +_{i \in I} \Delta_i \vdash \mathtt{a} : [\sigma_i]_{i \in I}} (\mathtt{m})$$
with $\Gamma = +_{i \in I} \Delta_i$, $\mathtt{A} = [\sigma_i]_{i \in I}$ and $\mathtt{a} = \mathcal{A}(\Pi) = \bigvee_{i \in I} \mathcal{A}(\Pi_i)$. In this case we need to prove $\mathtt{a} \Vdash \mathtt{TI}(+_{i \in I} \Delta_i, [\sigma_i]_{i \in I})$. By Proposition 3.9.2 we obtain, for all $i \in I$, a proof $\Pi'_i \triangleright \Gamma_i \vdash \mathcal{A}(\Pi_i) : \sigma_i$ such that $\mathcal{A}(\Pi'_i) = \mathcal{A}(\Pi_i)$. By the *i.h.*(a) we have that for all $i \in I$, $\mathcal{A}(\Pi_i) \Vdash \mathtt{T}(\Delta_i, \sigma_i)$, hence we conclude $\mathtt{a} = \bigvee_{i \in I} \mathcal{A}(\Pi_i) \Vdash \mathtt{TI}(+_{i \in I} \Delta_i, [\sigma_i]_{i \in I})$ by rule $(\mathtt{Union})$.

□

Theorem 3.19 (Soundness and Completeness of the inhabitation algorithm for $\mathcal{H}$)**.**
(1) *If* $\mathtt{a} \Vdash \mathtt{T}(\Gamma, \sigma)$ *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 a* β*-reduct* t' *of* t *and a derivation* $\Pi' \triangleright_{\mathtt{nf}} \Gamma \vdash \mathtt{t}' : \sigma$ *such that* $\mathcal{A}(\Pi') \Vdash \mathtt{T}(\Gamma, \sigma)$.

Proof. Soundness: if $\mathtt{a} \Vdash \mathtt{T}(\Gamma, \sigma)$ then by Lemma 3.18 ($\Rightarrow$) we have that $\Gamma \vdash \mathtt{a} : \sigma$. Then, by Proposition 3.9.2 we get $\Gamma \vdash \mathtt{t} : \sigma$ for all $\mathtt{a} \leq \mathtt{t}$.

Completeness: if $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$ then by Theorem 3.6.1, there exists $\Pi' \triangleright_{\mathtt{nf}} \Gamma \vdash \mathtt{t}' : \sigma$, with $\mathtt{t} \to_\beta^* \mathtt{t}'$. By Proposition 3.9.2, there exists $\Pi'' \triangleright \Gamma \vdash \mathcal{A}(\Pi') : \sigma$, and $\mathcal{A}(\Pi'') = \mathcal{A}(\Pi')$. We conclude that $\mathcal{A}(\Pi') \Vdash \mathtt{T}(\Gamma, \sigma)$ by Lemma 3.18 ($\Leftarrow$). □

The main difference between our inhabitation algorithm for system $\mathcal{H}$ and the classical inhabitation algorithm for simple types [3, 16] lies in the use of approximants. Besides that, our algorithm is peculiar in the fact that the head variables' arguments are constructed one by one, in such a way that the runs of the algorithm mirror exactly the typing derivations of system $\mathcal{H}$. As a matter of fact, a less fine-grained version of the algorithm is also possible, where all the arguments of the head variable are constructed at once (see Figure 5). Our step-by-step version is a simplified version of the algorithm presented in [6]. It is worth noticing that the simplification has a price: both the systems in Fig.4 and 5 are not linear, because of the rules $(\mathtt{Head}_\mathtt{b})$ and $(\mathtt{Head}_{>0})$: the type τ occurs in the conclusion of those rules both as the target type and as a subtype of the head variable's type.

3.3. **Breaking relevance.** Among the non-idempotent intersection type systems, $\mathcal{H}$ has certainly a particular status, since it induces a denotational model of the λ-calculus, based on relational semantics [28]. Thus our investigation of the inhabitation problem for non-idempotent intersection types systems started with $\mathcal{H}$. The system $\mathcal{H}$ is relevant, and it is reasonable to suppose that relevance plays a role in the decidability of the inhabitation problem, as it allows for a fine control of resource management. Hence, breaking relevance is our next step in the investigation pointed out above. Adding weakening to the type system is semantically unsound, in this framework, in the sense that the extended system does not

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

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

$$\frac{\Gamma = +_{i=1 \ldots n} \Gamma_i \qquad (\mathtt{b}_i \Vdash \mathtt{TI}(\Gamma_i, \mathtt{A}_i))_{i=1 \ldots n}}{\mathtt{x}\mathtt{b}_1 \ldots \mathtt{b}_n \Vdash \mathtt{T}(\Gamma + (\mathtt{x} : [\mathtt{A}_1 \to \ldots \mathtt{A}_n \to \tau]), \tau)} \; (\mathtt{Head}_{\mathtt{b}})$$

Figure 5: The basic inhabitation algorithm for system $\mathcal{H}$

induce a λ-model. Nevertheless it enjoys several interesting properties, and in particular the approximation theorem continues to hold.

It turns out that adding weakening to the system $\mathcal{H}$ does not break the decidability of the type inhabitation problem. The system $\mathcal{H}_w$, presented in Figure 6, is obtained by relaxing the axiom ($\mathtt{var}$) of system $\mathcal{H}$ to the more general axiom ($\mathtt{var}_w$), where the context is possibly over-defined, meaning that it may assign several different types both to the subject $\mathtt{x}$ and to other variables.

$$\frac{1 \leq i \leq n}{\Gamma, \mathtt{x} : [\rho_1, ..., \rho_n] \vdash \mathtt{x} : \rho_i} \; (\mathtt{var}_w) \qquad \frac{\Gamma \vdash \mathtt{t} : \tau}{\Gamma \setminus \mathtt{x} \vdash \lambda \mathtt{x}.\mathtt{t} : \Gamma(\mathtt{x}) \to \tau} \; (\to_{\mathtt{I}})$$

$$\frac{\Gamma \vdash \mathtt{t} : \mathtt{A} \to \tau \qquad \Delta \vdash \mathtt{u} : \mathtt{A}}{\Gamma + \Delta \vdash \mathtt{tu} : \tau} \; (\to_{\mathtt{E}}) \qquad \frac{(\Delta_i \vdash \mathtt{t} : \sigma_i)_{i \in I}}{+_{i \in I} \Delta_i \vdash \mathtt{t} : [\sigma_i]_{i \in I}} (\mathtt{m})$$

Figure 6: The type assignment system $\mathcal{H}_w$

It is easy to see that the usual weakening rule:

$$\frac{\Gamma \vdash \mathtt{t} : \sigma}{\Gamma + \Delta \vdash \mathtt{t} : \sigma} \; (\mathtt{W})$$

is admissible in system $\mathcal{H}_w$. Moreover, $\mathcal{H}_w$ is still syntax directed, and thus in particular $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$ and $\Pi' \triangleright \Gamma + \Delta \vdash \mathtt{t} : \sigma$ imply $\mathtt{meas}(\Pi) = \mathtt{meas}(\Pi')$. Note that, while in the idempotent case the presence of weakening makes the multiplicative and the additive versions equivalent, this is no longer true in the non-idempotent case. In fact, in order to preserve decidability of inhabitation, we need to preserve the quantitative flavour of the system. Indeed, for $\mathcal{H}_w$, Property 3.12 still holds. Clearly, in case of an additive definition of rule ($\to_{\mathtt{E}}$), like in Figure 1, that property does not hold. It is quite easy to check that system $\mathcal{H}_w$ enjoys subject reduction, subject expansion and that it characterizes head-normal forms. The algorithm proving the decidability of the type inhabitation problem for $\mathcal{H}_w$ can be easily obtained from the one in Figure 4 by changing the rule ($\mathtt{Head}_0$) as follows:

$$\frac{}{\mathtt{x} \Vdash \mathtt{H}^{\mathtt{x}:[\tau]}(\Gamma, \tau)} \; (\mathtt{Head}_0)$$

This new rule takes into account the fact that, in presence of weakening, not all the resources in the type context need to be consumed during a derivation. So, the algorithm may stop and produce a variable even if the context has not been fully consumed, assuming that the unused resources are created by weakening, and can be discarded. The same soundness and completness arguments of Theorem 3.19 apply, with the obvious changes. We can conclude that the undecidability of the inhabitation problem for systems like those in Figures 1 and 2 is due to the idempotency of the intersection, and not to the lack of relevance.

4. Systems characterizing strong normalization

4.1. **Disallowing untyped formal parameters.** Apparently the easiest way to modify the system $\mathcal{H}$ in order to restrict the class of typable terms to the set of strongly normalizing terms is to forbid untyped subterms inside typed terms.

The technical way to obtain this behaviour is to mimic what happens in the simply type assignment system, where, if we want to abstract a term with respect to a variable $\mathtt{x}$ which is not in the domain of the context, we just guess a type for it. Consequently, the empty multiset is no longer a multi-type, and the grammar of types becomes:

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

The resulting type system $\mathcal{H}_e$ is presented in Figure 7.

$$\frac{}{\mathtt{x} : [\rho] \vdash \mathtt{x} : \rho}\ (\mathtt{var})$$

$$\frac{\Gamma \vdash \mathtt{t} : \tau \qquad \mathtt{x} \in \mathtt{dom}(\Gamma)}{\Gamma \setminus \mathtt{x} \vdash \lambda\mathtt{x}.\mathtt{t} : \Gamma(\mathtt{x}) \to \tau}\ (\to_{\mathtt{I}\not[]}) \qquad \frac{\Gamma \vdash \mathtt{t} : \tau \qquad \mathtt{x} \notin \mathtt{dom}(\Gamma)}{\Gamma \vdash \lambda\mathtt{x}.\mathtt{t} : [\sigma] \to \tau}\ (\to_{\mathtt{I}[]})$$

$$\frac{\Gamma \vdash \mathtt{t} : \mathtt{A} \to \tau \qquad \Delta \vdash \mathtt{u} : \mathtt{A}}{\Gamma + \Delta \vdash \mathtt{tu} : \tau}\ (\to_{\mathtt{E}}) \qquad \frac{(\Delta_i \vdash \mathtt{t} : \sigma_i)_{i\in I} \quad I \neq \emptyset}{+_{i\in I}\Delta_i \vdash \mathtt{t} : [\sigma_i]_{i\in I}}\ (\mathtt{m})$$

Figure 7: The type assignment system $\mathcal{H}_e$

The type σ in rule $(\to_{\mathtt{I}[]})$ is arbitrary, so that it is chosen non-deterministically. Remark also that untyped subterms are no longer allowed inside typed terms, since the $(\to_{\mathtt{E}})$ rule always impose a functional type with non-empty multiset type on its left hand side. System $\mathcal{H}_e$ does not enjoys subject reduction nor subject expansion. In fact $\mathtt{x} : [\sigma] \vdash (\lambda\mathtt{y}.\mathtt{I})\mathtt{x} : [\tau] \to \tau$, while $\mathtt{x} : \sigma \not\vdash \mathtt{I} : [\tau] \to \tau$. In contrast, in the simple type case, the subject reduction is verified since the system allows weakening.

So, in order to restore subject reduction, we add weakening to system $\mathcal{H}_e$. The resulting system $\mathcal{H}_{e,w}$ is presented in Figure 8.

The weakening is only introduced in the axioms, as for system $\mathcal{H}_w$, making the rule:

$$\frac{\Gamma \vdash \mathtt{t} : \sigma}{\Gamma + \Delta \vdash \mathtt{t} : \sigma}\ (\mathtt{w})$$

$$\frac{1 \leq i \leq n}{\Gamma, \mathtt{x} : [\rho_1, ..., \rho_n] \vdash \mathtt{x} : \rho_i} \ (\mathtt{var}_w) \qquad \frac{\Gamma \vdash \mathtt{t} : \tau \quad \mathtt{x} \in \mathtt{dom}(\Gamma)}{\Gamma \setminus \mathtt{x} \vdash \lambda \mathtt{x}.\mathtt{t} : \Gamma(\mathtt{x}) \to \tau} \ (\to_{\mathtt{I}})$$

$$\frac{\Gamma \vdash \mathtt{t} : \mathtt{A} \to \tau \quad \Delta \vdash \mathtt{u} : \mathtt{A}}{\Gamma + \Delta \vdash \mathtt{tu} : \tau} \ (\to_{\mathtt{E}}) \qquad \frac{(\Delta_i \vdash \mathtt{t} : \sigma_i)_{i \in I}}{+_{i \in I} \Delta_i \vdash \mathtt{t} : [\sigma_i]_{i \in I}} (\mathtt{m})$$

Figure 8: The type assignment system $\mathcal{H}_{e,w}$

admissible. The resulting system is still syntax directed, so that $\Pi \rhd \Gamma \vdash \mathtt{t} : \sigma$ and $\Pi' \rhd \Gamma + \Delta \vdash \mathtt{t} : \sigma$ imply $\mathtt{meas}(\Pi) = \mathtt{meas}(\Pi')$. Note that, as for system $\mathcal{H}_w$, the rule $(\to_{\mathtt{E}})$ is multiplicative, thus preserving the fact that $d(\Gamma, \sigma)$ is an upper bound to the number of variable positions in any subject of a derivation with context Γ and type σ. Let us remark that, in contrast to system $\mathcal{H}_e$, just one rule introducing the arrow is needed, since the presence of the weakening ensures that, if $\Pi \rhd \Gamma \vdash \mathtt{t} : \sigma$, then there is always $\Pi' \rhd \Gamma' \vdash \mathtt{t} : \sigma$, where Γ' extends Γ and $\mathtt{x} \in \mathtt{dom}(\Gamma')$.

It is important to stress the fact that, in presence of weakening, the types assigned to a term only depend on the axioms giving types to its free variables. In fact, if $\mathtt{x}$ does not occur in the subject of a derivation, no axiom with subject $\mathtt{x}$ is used in the derivation itself. Formally:

Property 4.1.

(1) $\Pi \rhd \Gamma, \mathtt{x} : \mathtt{A} \vdash_{\mathcal{H}_{e,w}} \mathtt{t} : \sigma$ and $\mathtt{x} \notin \mathtt{fv}(\mathtt{t})$ imply that there are no axioms in Π with subject $\mathtt{x}$, so $\mathtt{x} : \mathtt{A}$ has necessarily been introduced by weakening some other axiom.
(2) If $\Pi \rhd \Gamma, \mathtt{x} : \mathtt{A} \vdash_{\mathcal{H}_{e,w}} \mathtt{t} : \sigma$ and $\mathtt{x} \notin \mathtt{fv}(\mathtt{t})$, then $\Pi' \rhd \Gamma \vdash_{\mathcal{H}_{e,w}} \mathtt{t} : \sigma$.

The system enjoys the following good properties.

Theorem 4.2.

(1) *(Subject reduction)* $\Gamma \vdash_{\mathcal{H}_{e,w}} \mathtt{t} : \sigma$ *and* $\mathtt{t} \to_\beta \mathtt{u}$ *imply* $\Gamma \vdash_{\mathcal{H}_{e,w}} \mathtt{u} : \sigma$.
(2) *(Typed subject expansion) Let* $\mathtt{C}$ *be a context. Then* $\Gamma \vdash_{\mathcal{H}_{e,w}} \mathtt{C}[\mathtt{v}\{\mathtt{w}/\mathtt{x}\}] : \sigma$ *and* $\Delta \vdash \mathtt{w} : \mathtt{A}$ *imply* $\Gamma' \vdash_{\mathcal{H}_{e,w}} \mathtt{C}[(\lambda \mathtt{x}.\mathtt{v})\mathtt{w}] : \sigma$*, for some* Γ'.
(3) *(Strong normalization)* $\Gamma \vdash_{\mathcal{H}_{e,w}} \mathtt{t} : \sigma$ *iff* $\mathtt{t}$ *is strongly normalizing.*

Proof.

(1) We will prove something more, namely that $\Pi \rhd \Gamma \vdash \mathtt{t} : \sigma$ and $\mathtt{t} \to_\beta \mathtt{u}$ imply $\Pi' \rhd \Gamma \vdash \mathtt{u} : \sigma$, where $\mathtt{meas}(\Pi') < \mathtt{meas}(\Pi)$. Indeed, $\mathtt{t} \to_\beta \mathtt{u}$ means $\mathtt{t} = \mathtt{C}[(\lambda \mathtt{x}.\mathtt{v})\mathtt{w}]$ and $\mathtt{u} = \mathtt{C}[\mathtt{v}\{\mathtt{w}/\mathtt{x}\}]$ for some context $\mathtt{C}$. The proof is by induction on $\mathtt{C}$. For the base case $\mathtt{C} = \Box$, it is sufficient to consider an erasure reduction, *i.e.* the situation where $\mathtt{x} \notin \mathtt{fv}(\mathtt{v})$, being the not erasing case already proved *e.g.* in [7]. The derivation Π for $\Gamma \vdash (\lambda \mathtt{x}.\mathtt{v})\mathtt{w} : \sigma$ is of the shape

$$\frac{\dfrac{\Pi' \rhd \Gamma \vdash \mathtt{v} : \sigma}{\Gamma \setminus \mathtt{x} \vdash \lambda \mathtt{x}.\mathtt{v} : \Gamma(\mathtt{x}) \to \sigma} \ (\to_{\mathtt{I}}) \qquad \Delta \vdash \mathtt{w} : \mathtt{A}}{(\Gamma \setminus \mathtt{x}) + \Delta \vdash (\lambda \mathtt{x}.\mathtt{v})\mathtt{w} : \sigma} \ (\to_{\mathtt{E}})$$

where $\mathtt{x} \notin \mathtt{fv}(\mathtt{v})$ implies by Proposition 4.1 that there is no axiom rule with subject $\mathtt{x}$. So from $\Pi' \rhd \Gamma \vdash \mathtt{v} : \sigma$, and the admissibility of the weakening rule, there is a derivation $\Pi'' \vdash \Gamma + \Delta \vdash \mathtt{v} : \sigma$. Note that $\mathtt{meas}(\Pi'') = \mathtt{meas}(\Pi')$. Since $\mathtt{v}\{\mathtt{w}/\mathtt{x}\} = \mathtt{v}$, then Π'' is the desired derivation, and the case is proved, being $\mathtt{meas}(\Pi'') < \mathtt{meas}(\Pi)$. The other cases come easily by induction.

(2) By induction on $\mathtt{C}$. Let us consider the base case, *i.e.* $\mathtt{C} = \square$. Also in this point, we will consider only the case when $\mathtt{x} \notin \mathtt{fv}(\mathtt{v})$, so that $\mathtt{v}\{\mathtt{w}/\mathtt{x}\} = \mathtt{v}$, being the not erasing case treated in [7]. Then, Property 4.1 gives $\Gamma \setminus \mathtt{x} \vdash \mathtt{v} : \sigma$. By hypothesis $\Delta \vdash \mathtt{w} : \tau$, for some Δ and τ, so by the admissibility of weakening we can build a derivation $\Gamma \setminus \mathtt{x} + \mathtt{x} : [\tau] \vdash \mathtt{v} : \sigma$, and then, by applying rule $(\rightarrow_{\mathtt{I}})$, we get $\Gamma \setminus \mathtt{x} \vdash \lambda \mathtt{x}.\mathtt{t} : [\tau] \rightarrow \sigma$. By rule $(\rightarrow_{\mathtt{E}})$ we obtain $\Gamma \setminus \mathtt{x} + \Delta \vdash (\lambda \mathtt{x}.\mathtt{v})\mathtt{w} : \sigma$ as desired. The inductive case is straightforward.

(3) Full details can be found in [7]. $\square$

Note that the proof of the third point of the previous theorem guarantees that all normal forms are typed in the system, since the set of normal forms is included in the set of strongly-normalizing terms.

The inhabitation problem for system $\mathcal{H}_{e,w}$ is decidable, the corresponding algorithm is given in the Figure 9.

The reader may remark that there are two rules which have been changed with respect to the algorithm for system $\mathcal{H}$ in Figure 4: rule ($\mathtt{Head}$) and rule ($\mathtt{Union}$). Rule ($\mathtt{Head}$) is changed in the same way we did for system $\mathcal{H}_w$, and so the same considerations hold. Rule ($\mathtt{Union}$) in Figure 4 was building the set of all the approximate normal forms $\mathtt{a} = \bigvee_{i \in I} \mathtt{a}_i$ such that $\Gamma = +_{i \in I} \Gamma_i$, $\mathtt{a}_i \Vdash \mathtt{T}(\Gamma_i, \sigma_i)$ for all $i \in I$, and $\uparrow_{i \in I} \mathtt{a}_i$. In system $\mathcal{H}_{e,w}$ the situation is easier, since normal forms are pairwise unbounded (*i.e.* there is no common upper bound). So the set $\mathtt{TI}(+_{i \in I} \Gamma_i, [\sigma_i]_{i \in I})$ now contains the unique normal form $\mathtt{t}$ such that $\Gamma_i \vdash \mathtt{t} : \sigma_i$, for all $i \in I$. The algorithm is sound and complete, as the next theorem shows.

Theorem 4.3 (Soundness and Completeness for $\mathcal{H}_{e,w}$)**.**

(1) *If* $\mathtt{t} \Vdash \mathtt{T}(\Gamma, \sigma)$ *then* $\Gamma \vdash_{\mathcal{H}_{e,w}} \mathtt{t} : \sigma$.

(2) *If* $\Gamma \vdash_{\mathcal{H}_{e,w}} \mathtt{t} : \sigma$ *then* $\mathtt{t}' \Vdash \mathtt{T}(\Gamma, \sigma)$, *where* $\mathtt{t} \rightarrow_\beta^* \mathtt{t}'$ *and* $\mathtt{t}'$ *is in normal form.*

The proof is similar to the corresponding proof for the system $\mathcal{H}$, but simpler, because of the observations made before.

$$\frac{\mathtt{t} \Vdash \mathtt{T}(\Gamma + \mathtt{x} : \mathtt{A}, \tau) \qquad \mathtt{x} \notin \mathtt{dom}(\Gamma)}{\lambda \mathtt{x}.\mathtt{t} \Vdash \mathtt{T}(\Gamma, \mathtt{A} \rightarrow \tau)}\ (\mathtt{Abs})$$

$$\frac{(\mathtt{t} \Vdash \mathtt{T}(\Gamma_i, \sigma_i))_{i \in I}}{\mathtt{t} \Vdash \mathtt{TI}(+_{i \in I} \Gamma_i, [\sigma_i]_{i \in I})}\ (\mathtt{Union})$$

$$\frac{\Gamma = \Gamma_1 + \Gamma_2 \quad \mathtt{t} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \rightarrow \ldots \mathtt{A}_n \rightarrow \mathtt{B} \rightarrow \tau]}(\Gamma_1, \mathtt{B} \rightarrow \tau) \quad \mathtt{u} \Vdash \mathtt{TI}(\Gamma_2, \mathtt{B}) \quad n \geq 0}{\mathtt{tu} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \rightarrow \ldots \mathtt{A}_n \rightarrow \mathtt{B} \rightarrow \tau]}(\Gamma, \tau)}\ (\mathtt{Head}_{>0})$$

$$\frac{}{\mathtt{x} \Vdash \mathtt{H}^{\mathtt{x}:[\tau]}(\Gamma, \tau)}\ (\mathtt{Head}_0)$$

$$\frac{\mathtt{t} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \rightarrow \ldots \mathtt{A}_n \rightarrow \tau]}(\Gamma, \tau)}{\mathtt{t} \Vdash \mathtt{T}(\Gamma + \mathtt{x} : [\mathtt{A}_1 \rightarrow \ldots \mathtt{A}_n \rightarrow \tau], \tau)}\ (\mathtt{Head})$$

Figure 9: The inhabitation algorithm for system $\mathcal{H}_{e,w}$

We conclude this section by showing that the inhabitation problem for $\mathcal{H}_e$ reduces to the one for $\mathcal{H}_{e,w}$.

Lemma 4.4. *Let* (Γ, τ) *be an instance of the inhabitation problem having a solution in system* $\mathcal{H}_e$. *Then* $(\Gamma + \mathtt{x} : [\sigma], \tau)$ *has also a solution in* $\mathcal{H}_e$, *for each variable* $\mathtt{x}$ *and type* σ.

Proof. A suitable type derivation is the following:

$$\dfrac{\dfrac{\dfrac{\vdots}{\Gamma \vdash \mathtt{t}{:}\tau \quad \mathtt{z} \notin \mathtt{fv}(\mathtt{t})}}{\Gamma \vdash \lambda \mathtt{z}.\mathtt{t}{:}[\sigma] \to \tau}\,(\to_{\mathtt{I}[]}) \qquad \dfrac{}{\mathtt{x}{:}[\sigma] \vdash \mathtt{x}{:}\sigma}}{\Gamma + \mathtt{x}{:}[\sigma] \vdash (\lambda \mathtt{z}.\mathtt{t})\mathtt{x}{:}\tau}$$

□

Lemma 4.5. (Γ, σ) *is inhabited in* $\mathcal{H}_e$ *if and only if* (Γ, σ) *is inhabited in* $\mathcal{H}_{e,w}$.

Proof. The *only if* part is trivial since any type derivation in $\mathcal{H}_e$ is also a derivation in $\mathcal{H}_{e,w}$. Conversely, it is easy to show that any derivation Π in $\mathcal{H}_{e,w}$ may be mimicked in $\mathcal{S}$, by induction on the size of Π: if Π is an axiom, then Lemma 4.4 allows to conclude, otherwise the conclusion is by induction on the subderivations of the premises of Π's last rule. □

Systems $\mathcal{H}_e$ and $\mathcal{H}_{e,w}$ inhabit exactly the same types, but $\mathcal{H}_{e,w}$ enjoys subject reduction (Theorem 4.2), and hence allow for searching inhabitants in normal form.

4.2. **Disallowing untyped actual parameters.** In Section 4.1 we have seen how strong normalization may be characterized by disallowing the empty (multi-)type. This choice leads to the system $\mathcal{H}_{e,w}$, in which a type is guessed arbitrarily whenever a term is λ-abstracted with respect to a fresh variable (*i.e.* one which is not in the domain of the context). This corresponds to choosing an arbitrary type for the *formal* parameter of an erasing function. Dually, it is possible to characterize strong normalization without disallowing the empty type, by just guessing an arbitrary type for the *actual* parameters of all the erasing functions. The resulting system, called $\mathcal{S}$, is presented in Figure 10, where types are defined as in Definition 3.1.

Along with system $\mathcal{H}_e$, system $\mathcal{S}$ does not enjoy subject reduction, and this is certainly a major weakness. Still, it is a very simple system characterising strong normalisation, which has been considered for example in [7, 19].

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

$$\dfrac{\Gamma \vdash \mathtt{t} : \mathtt{A} \to \tau \quad \Delta \vdash \mathtt{u} : \mathtt{A} \quad \mathtt{A} \neq [\,]}{\Gamma + \Delta \vdash \mathtt{tu} : \tau}\,(\to_{\mathtt{E}\not[]}) \qquad \dfrac{\Gamma \vdash \mathtt{t} : [\,] \to \tau \quad \Delta \vdash \mathtt{u} : [\sigma]}{\Gamma + \Delta \vdash \mathtt{tu} : \tau}\,(\to_{\mathtt{E}[]})$$

$$\dfrac{(\Delta_i \vdash \mathtt{t} : \sigma_i)_{i \in I}}{+_{i \in I} \Delta_i \vdash \mathtt{t} : [\sigma_i]_{i \in I}}\,(\mathtt{m})$$

Figure 10: The type assignment system $\mathcal{S}$

Concerning the failure of subject reduction in system $\mathcal{S}$, the same counterexample given for system $\mathcal{H}_e$ applies: $\mathtt{x} : [\sigma] \vdash_{\mathcal{S}} (\lambda\mathtt{y}.\mathtt{I})\mathtt{x} : [\alpha] \rightarrow \alpha$, while $\mathtt{x} : [\sigma] \not\vdash_{\mathcal{S}} \mathtt{I} : [\alpha] \rightarrow \alpha$.

Notice that, if in the conclusion of rule $(\rightarrow_{\mathtt{E}[\,]})$ we forget the context Δ, then the system does not characterize strong normalization anymore. In fact, starting from $\mathtt{x} : [[\,] \rightarrow \sigma] \vdash_{\mathcal{S}} \mathtt{x} : [\,] \rightarrow \sigma$, we would obtain $\mathtt{x} : [[\,] \rightarrow \sigma] \vdash_{\mathcal{S}} \mathtt{x}(\mathtt{y}\mathtt{D}_{\mathtt{up}}) : \sigma$ by rule $(\rightarrow \mathtt{E}_{[\,]})$, then $\mathtt{x} : [[\,] \rightarrow \sigma] \vdash_{\mathcal{S}} \lambda\mathtt{y}.\mathtt{x}(\mathtt{y}\mathtt{D}_{\mathtt{up}}) : [\,] \rightarrow \sigma$ by $(\rightarrow \mathtt{I})$, and finally $\mathtt{x} : [[\,] \rightarrow \sigma] \vdash_{\mathcal{S}} (\lambda\mathtt{y}.\mathtt{x}(\mathtt{y}\mathtt{D}_{\mathtt{up}}))\mathtt{D}_{\mathtt{up}} : \sigma$, by $(\rightarrow \mathtt{E}_{[\,]})$. The subject of the last judgement reduces to $\mathtt{x}(\mathtt{D}_{\mathtt{up}}\mathtt{D}_{\mathtt{up}})$, which diverges.

In [7] it is shown that the $\mathcal{S}$-typable terms are exactly the strongly normalizing ones, and the decidability of the inhabitation problem for $\mathcal{S}$ is left open. We are going to show that the inhabitation problem for $\mathcal{S}$ is decidable, by reducing it to the inhabitation problem for the system $\mathcal{S}_w$ presented in Figure 11. System $\mathcal{S}_w$ is obtained from $\mathcal{S}$ by adding weakening to the axiom so that, again, the usual weakening rule becomes admissible.

$$\frac{1 \leq i \leq n}{\Gamma, \mathtt{x} : [\rho_1, ..., \rho_n] \vdash \mathtt{x} : \rho_i}\,(\mathtt{var}_w) \qquad \frac{\Gamma \vdash \mathtt{t} : \tau}{\Gamma \setminus \mathtt{x} \vdash \lambda\mathtt{x}.\mathtt{t} : \Gamma(\mathtt{x}) \rightarrow \tau}\,(\rightarrow_{\mathtt{I}})$$

$$\frac{\Gamma \vdash \mathtt{t} : \mathtt{A} \rightarrow \tau \quad \Delta \vdash \mathtt{u} : \mathtt{A} \quad \mathtt{A} \neq [\,]}{\Gamma + \Delta \vdash \mathtt{tu} : \tau}\,(\rightarrow_{\mathtt{E}\not[\,]}) \qquad \frac{\Gamma \vdash \mathtt{t} : [\,] \rightarrow \tau \quad \Delta \vdash \mathtt{u} : [\sigma]}{\Gamma + \Delta \vdash \mathtt{tu} : \tau}\,(\rightarrow_{\mathtt{E}[\,]})$$

$$\frac{(\Delta_i \vdash \mathtt{t} : \sigma_i)_{i \in I}}{+_{i \in I}\Delta_i \vdash \mathtt{t} : [\sigma_i]_{i \in I}}\,(\mathtt{m})$$

Figure 11: The type assignment system $\mathcal{S}_w$

Before introducing the inhabitation algorithm for $\mathcal{S}_w$, let us present the reduction of the inhabitation problem for system $\mathcal{S}$ to the one for system $\mathcal{S}_w$. The argument is the same as in the case $\mathcal{H}_e$ versus $\mathcal{H}_{e,w}$, treated in Section 4.1.

Lemma 4.6. *Let* (Γ, τ) *be an instance of the inhabitation problem having a solution in system* $\mathcal{S}$*. Then* $(\Gamma + \mathtt{x} : [\sigma], \tau)$ *has also a solution in* $\mathcal{S}$*, for each variable* $\mathtt{x}$ *and type* σ*.*

Proof. A suitable type derivation is the following:

$$\frac{\dfrac{\dfrac{\vdots}{\Gamma \vdash \mathtt{t}{:}\tau}}{\Gamma \vdash \lambda\mathtt{z}.\mathtt{t}{:}[\,] \rightarrow \tau} \qquad \overline{\mathtt{x}{:}[\sigma] \vdash \mathtt{x}{:}\sigma}}{\Gamma + \mathtt{x}{:}[\sigma] \vdash (\lambda\mathtt{z}.\mathtt{t})\mathtt{x}{:}\tau}$$

$\mathtt{z}$ being a fresh variable. □

Lemma 4.7. (Γ, σ) *is inhabited in* $\mathcal{S}$ *if and only if* (Γ, σ) *is inhabited in* $\mathcal{S}_w$*.*

Proof. The *only if* part is trivial since any type derivation in $\mathcal{S}$ is also a derivation in $\mathcal{S}_w$. Conversely, it is easy to show that any derivation Π in $\mathcal{S}_w$ may be mimicked in $\mathcal{S}$, by induction on the size of Π: if Π is an axiom, then Lemma 4.6 allows to conclude, otherwise the conclusion is by induction on the subderivations of the premises of Π's last rule. □

Systems $\mathcal{S}$ and $\mathcal{S}_w$ inhabit exactly the same types, but $\mathcal{S}_w$ enjoys subject reduction, and hence allow for searching inhabitants in normal form.

Proposition 4.8. *The type system $\mathcal{S}_w$ enjoys subject reduction.*

Proof. In order to show subject reduction, we first need a substitution lemma that can be stated as follows. Let $\Gamma, \mathtt{x} : \mathtt{A} \vdash \mathtt{t} : \sigma$ and $\Delta \vdash \mathtt{u} : \mathtt{A}$, then $\Gamma + \Delta \vdash \mathtt{t}\{\mathtt{u}/\mathtt{x}\} : \sigma$. The proof of this lemma is by straightforward induction on Π. It must be noticed that this proof uses the admissible rule $(\mathtt{w})$. In fact, in case $\mathtt{x} \notin \mathtt{fv}(\mathtt{t})$, $\mathtt{t}\{\mathtt{u}/\mathtt{x}\} = \mathtt{t}$, and in the resulting derivation $\Gamma + \Delta \vdash \mathtt{t} : \sigma$ all the premises in Δ need to be introduced by weakening.

Now, in order to show subject reduction, we need to prove that $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$ and $\mathtt{t} \rightarrow_\beta \mathtt{t}'$ implies there exists a derivation $\Pi' \triangleright \Gamma \vdash \mathtt{t}' : \sigma$. The proof is by induction on the context $\mathtt{C}$ such that $\mathtt{t} = \mathtt{C}[(\lambda \mathtt{x}.\mathtt{u})\mathtt{v}]$ and $\mathtt{t}' = \mathtt{C}[\mathtt{u}\{\mathtt{v}/\mathtt{x}\}]$. The case $\mathtt{C} = \Box$ comes directly from the substitution lemma, the induction cases are straightforward. $\Box$

Moreover, the type system $\mathcal{S}_w$ characterizes strong-normalization.

Proposition 4.9. *Let* $\mathtt{t}$ *be a* λ*-term. Then* $\Gamma \vdash \mathtt{t} : \sigma$ *iff* $\mathtt{t}$ *is strongly-normalizing.*

Proof. Exactly the same reasoning used in Theorem 4.2. $\Box$

4.2.1. *Inhabitation for $\mathcal{S}_w$.* In system S_w, types have in general infinitely many different inhabitants in normal form. For instance, the problem $(\mathtt{x} : [[\,] \rightarrow \sigma], \sigma)$ admits all the solutions of the form $\mathtt{xt}$, where $\mathtt{t}$ is closed and normal. In order to be complete in the sense of our previous Theorem 3.19, the inhabitation algorithm for S_w should produce an arbitrary normal form anytime the argument of an erasing function has to be constructed. Instead, we decide to treat all such case uniformly, by always constructing the same fake argument, namely the identity.

Definition 4.10. A type derivation Π in $\mathcal{S}_w$ is *standard* if all instances of $(\rightarrow_{\mathtt{E}[\,]})$ in Π have the shape

$$\frac{\Gamma \vdash \mathtt{t} : [\,] \rightarrow \rho \qquad \Delta \vdash \mathtt{I} : [\alpha] \rightarrow \alpha}{\Gamma + \Delta \vdash \mathtt{tI} : \rho}$$

We write Π_{st} is Π is a standard derivation.

Lemma 4.11. *If* (Γ, σ) *is inhabited in* $\mathcal{S}_w$*, then there exists a standard derivation* $\Pi \triangleright \Gamma \vdash_{\mathcal{S}_w} \mathtt{t} : \sigma$

Proof. Let $\Pi' \triangleright \Gamma \vdash_{\mathcal{S}_w} \mathtt{t}' : \sigma$ be any derivation. Replacing the non-standard instances of $(\rightarrow_{\mathtt{E}[\,]})$ in Π' of the form

$$\frac{\Gamma \vdash \mathtt{t} : [\,] \rightarrow \rho \qquad \Delta \vdash \mathtt{u} : \sigma}{\Gamma + \Delta \vdash_w \mathtt{tu} : \rho}\ (\rightarrow_{\mathtt{E}[\,]})$$

by

$$\frac{\Gamma \vdash \mathtt{t} : [\,] \rightarrow \rho \qquad \dfrac{\overline{\Delta + \mathtt{x} : [\alpha] \vdash \mathtt{x} : \alpha}\ (\mathtt{var})}{\Delta \vdash \mathtt{I} : [\alpha] \rightarrow \alpha}\ (\rightarrow_{\mathtt{I}})}{\Gamma + \Delta \vdash \mathtt{tI} : \rho}\ (\rightarrow_{\mathtt{E}[\,]})$$

where $\mathtt{x}$ is fresh, we obtain a standard derivation for an inhabitant of (Γ, σ). $\Box$

The previous Lemma shows that, in order to decide whether a type is inhabited in $\mathcal{S}_w$, it is sufficient to look for standard derivations. This is what the inhabitation algorithm given in Figure 12 does. The rule $(\mathtt{Head}_{>0})$ splits in two: the usual one, used now for the case $\mathtt{A} \neq [\,]$, and the new rule $(\mathtt{Head}^{[\,]}_{>0})$ introducing subterms of the form $\mathtt{I}$ to be taken as actual parameters of erasing functions. The soundness of the inhabitation algorithm holds in general, as for the other systems, with the aside that the obtained derivation is standard. On the other hand, completeness holds in a relativized form: only the normal forms of the subjects of standard derivations are reconstructed by the algorithm. In order to establish completeness we will make use of the following stability property:

Lemma 4.12 (Stability of Standard Derivations)**.** *If* $\Pi \triangleright \Gamma \vdash \mathtt{t} : \sigma$ *is a standard derivation and* $\mathtt{t} \to_\beta \mathtt{t}'$, *then not only there exists* $\Pi' \triangleright \Gamma \vdash \mathtt{t}' : \sigma$, *but* Π' *is also a standard derivation.*

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

$$\frac{(\mathtt{t} \Vdash \mathtt{T}(\Gamma_i, \sigma_i))_{i \in I}}{\mathtt{t} \Vdash \mathtt{TI}(+_{i \in I}\Gamma_i, [\sigma_i]_{i \in I})}\ (\mathtt{Union})$$

$$\frac{\Gamma = \Gamma_1 + \Gamma_2 \quad \mathtt{t} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \mathtt{A}_n \to \mathtt{B} \to \tau]}(\Gamma_1, \mathtt{B} \to \tau) \quad \mathtt{u} \Vdash \mathtt{TI}(\Gamma_2, \mathtt{B}) \quad \mathtt{B} \neq [\,] \quad n \geq 0}{\mathtt{tu} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \mathtt{A}_n \to \mathtt{B} \to \tau]}(\Gamma, \tau)}\ (\mathtt{Head}_{>0})$$

$$\frac{\mathtt{t} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \mathtt{A}_n \to [\,] \to \tau]}(\Gamma_1, [\,] \to \tau) \quad n \geq 0}{\mathtt{tI} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \mathtt{A}_n \to [\,] \to \tau]}(\Gamma, \tau)}\ (\mathtt{Head}^{[\,]}_{>0})$$

$$\frac{}{\mathtt{x} \Vdash \mathtt{H}^{\mathtt{x}:[\tau]}(\Gamma, \tau)}\ (\mathtt{Head}_0)$$

$$\frac{\mathtt{t} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \mathtt{A}_n \to \tau]}(\Gamma, \tau)}{\mathtt{t} \Vdash \mathtt{T}(\Gamma + \mathtt{x} : [\mathtt{A}_1 \to \ldots \mathtt{A}_n \to \tau], \tau)}\ (\mathtt{Head})$$

Figure 12: The inhabitation algorithm for $\mathcal{S}_w$

As expected, the inhabitation algorithm terminates, a property that can be shown exactly as in Lemma 3.16.

Lemma 4.13 (Termination)**.** *The inhabitation algorithm for system* $\mathcal{S}_w$ *terminates.*

We now change the interpretation of the sets involved in the algorithm to:
$[\![\mathtt{T}(\Gamma, \sigma)]\!] = \{\mathtt{t} \mid \exists \Pi_{st}.\ \Pi_{st} \triangleright \Gamma \vdash \mathtt{t} : \sigma$ and $\mathtt{t}$ in normal form$\}$.
$[\![\mathtt{TI}(\Gamma, \mathtt{A})]\!] = \{\mathtt{t} \mid \exists \Pi_{st}.\ \Pi_{st} \triangleright \Gamma \vdash \mathtt{t} :$ and $\mathtt{t}$ in normal form$\}$.
$[\![\mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to A_n \to \tau]}(\Gamma, \tau)]\!] = \{\mathtt{xt}_1 \ldots \mathtt{t}_n \mid \exists \Pi_{st}.\ \Pi_{st} \triangleright \Gamma + \mathtt{x} : [\mathtt{A}_1 \to \ldots \to \mathtt{A}_n \to \tau] \vdash \mathtt{xt}_1 \ldots \mathtt{t}_n : \tau$, $\Gamma = +_{i=1,\ldots,n}\Gamma_i$, $\Gamma_i \vdash \mathtt{t}_i : \mathtt{A}_i$ and $\mathtt{t}_i$ in normal form$\}$

Theorem 4.14 (Soundness and Completeness for $\mathcal{S}_w$)**.**
(1) *If* $\mathtt{t} \Vdash \mathtt{T}(\Gamma, \sigma)$ *then* $\mathtt{t} \in [\![\mathtt{T}(\Gamma, \sigma)]\!]$.
(2) *If* $\mathtt{t} \in [\![\mathtt{T}(\Gamma, \sigma)]\!]$, *then* $\mathtt{t} \Vdash \mathtt{T}(\Gamma, \sigma)$.

Proof. (1) We proceed analogously to the proof of Lemma 3.18, proving the following statements by induction on the rules in Figure 12, from which soundness follows directly.
a) $\mathtt{t} \Vdash \mathtt{T}(\Gamma, \sigma) \Rightarrow \mathtt{t} \in [\![\mathtt{T}(\Gamma, \sigma)]\!]$.
b) $\mathtt{t} \Vdash \mathtt{TI}(\Gamma, \mathtt{A}) \Rightarrow \mathtt{t} \in [\![\mathtt{T}(\Gamma, \mathtt{A})]\!]$.
c) $\mathtt{t} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to A_n \to \tau]}(\Gamma, \tau) \Rightarrow \mathtt{t} \in [\![\mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to A_n \to \tau]}(\Gamma, \tau)]\!]$.

The cases of rules ($\mathtt{Abs}$) and ($\mathtt{Union}$) are similar to the analogous cases in the proof of Lemma 3.18, modulo considering normal forms instead than approximants. There are three cases ending in the judgement $\mathtt{t} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to A_n \to \tau]}(\Gamma, \tau)$:

- If $\mathtt{x} \Vdash \mathtt{H}^{\mathtt{x}:[\tau]}(\Gamma, \tau)$ comes from ($\mathtt{Head}_0$), then $\Pi_{st} \triangleright \Gamma + \mathtt{x} : [\tau] \vdash \mathtt{x} : \tau$ is obtained by rule ($\mathtt{var}_w$).
- If $\mathtt{tu} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to A_n \to \tau]}(\Gamma + \Delta, \tau)$ comes from rule ($\mathtt{Head}_{>0}$), where $\mathtt{A}_n \neq [\,]$, $\mathtt{t} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to \mathtt{A}_{n-1} \to \mathtt{A}_n \to \tau]}(\Gamma, \mathtt{A}_n \to \tau)$ and $\mathtt{u} \Vdash \mathtt{TI}(\Delta, \mathtt{A}_n)$, then:
  - by the *i.h.*(c), $\mathtt{t} = \mathtt{x}\mathtt{t}_1 ... \mathtt{t}_{n-1}$ and there exists $\Pi^1_{st} \triangleright \Gamma + \mathtt{x} : [\mathtt{A}_1 \to \ldots \to A_n \to \tau] \vdash \mathtt{t} : \mathtt{A}_n \to \tau$.
  - by the *i.h.*(b) there exists $\Pi^2_{st} \triangleright \Delta \vdash \mathtt{u} : \mathtt{A}_n$.

  So, using Rule ($\to_{\mathtt{E}[\not\,]}$), we obtain a proof $\Pi_{st} \triangleright \Gamma + \Delta + \mathtt{x} : [\mathtt{A}_1 \to \ldots \to A_n \to \tau] \vdash \mathtt{x}\mathtt{t}_1 ... \mathtt{t}_{n-1}\mathtt{u} : \tau$, and we are done.
- If $\mathtt{tI} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to \mathtt{A}_n \to [\,] \to \tau]}(\Gamma, \tau)$ comes from rule ($\mathtt{Head}^{[\,]}_{>0}$), where the premise is $\mathtt{t} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to \mathtt{A}_n \to [\,] \to \tau]}(\Gamma, [\,] \to \tau)$, then $\mathtt{t} = \mathtt{x}\mathtt{t}_1 \ldots \mathtt{t}_{n-1}$ and by the *i.h.*(c) there exists Π_{st} such that $\Pi_{st} \triangleright \Gamma + \mathtt{x} : [\mathtt{A}_1 \to \ldots \to \mathtt{A}_n \to [\,] \to \tau] \vdash \mathtt{t} : [\,] \to \tau$. Then the proof follows by rule ($\to_{\mathtt{E}[\,]}$).

(2) Let $\Pi_{st} \triangleright \Gamma \vdash \mathtt{t} : \sigma$ ($\Pi_{st} \triangleright \Gamma \vdash \mathtt{t} : \mathtt{A}$), $\mathtt{t}$ being a normal form. We prove the following statements by induction on the typing derivations:
  a) $\mathtt{t} \in [\![\mathtt{T}(\Gamma, \sigma)]\!] \Rightarrow \mathtt{t} \Vdash \mathtt{T}(\Gamma, \sigma)$.
  b) $\mathtt{t} \in [\![\mathtt{T}(\Gamma, \mathtt{A})]\!] \Rightarrow \mathtt{t} \Vdash \mathtt{TI}(\Gamma, \mathtt{A})$.
  c) $\mathtt{t} \in [\![\mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to A_n \to \tau]}(\Gamma, \tau)]\!] \Rightarrow \mathtt{t} \Vdash \mathtt{H}^{\mathtt{x}:[\mathtt{A}_1 \to \ldots \to A_n \to \tau]}(\Gamma, \tau)$.

- Case ($\mathtt{var}$). Then there is a standard type derivation of the following form:

$$\frac{}{\Pi \triangleright \Delta + \mathtt{x} : [\rho_1, \ldots, \rho_n] \vdash \mathtt{x} : \rho_i} \ (\mathtt{var}_w)$$

  In this case $\mathcal{A}(\Pi) = \mathtt{x}$ belongs to $[\![\mathtt{T}(\Delta + \mathtt{x} : [\rho_1, \ldots, \rho_n], \rho_i)]\!]$ and to $[\![\mathtt{H}^{\mathtt{x}:[\rho_i]}(\Delta + \mathtt{x} : [\rho_1, \ldots, \rho_{i-1}, \rho_{i+1}, \ldots, \rho_n], \rho_i)]\!]$, so we need to show a) and c) respectively.
  By rule ($\mathtt{Head}_0$) we have $\mathtt{x} \Vdash \mathtt{H}^{\mathtt{x}:[\rho_i]}(\Delta + \mathtt{x} : [\rho_1, \ldots, \rho_{i-1}, \rho_{i+1}, \ldots, \rho_n], \rho_i)$, then by rule ($\mathtt{Head}$) we also conclude $\mathtt{x} \Vdash \mathtt{T}(\Delta + \mathtt{x} : [\rho_1, \ldots, \rho_n], \rho_i)$.
- Case ($\to_{\mathtt{I}}$). Then there is a standard type derivation of the following form:

$$\frac{\Pi'_{st} \triangleright \Gamma \vdash \mathtt{u} : \tau}{\Pi_{st} \triangleright \Gamma \setminus \mathtt{x} \vdash \lambda \mathtt{x}.\mathtt{u} : \Gamma(\mathtt{x}) \to \tau} \ (\to_{\mathtt{I}})$$

  where also $\mathtt{u}$ is in normal form. In this case, $\lambda\mathtt{x}.\mathtt{u} \in [\![\mathtt{T}(\Gamma \setminus \mathtt{x}, \Gamma(\mathtt{x}) \to \tau)]\!]$ (and thus $\mathtt{u} \in [\![\mathtt{T}(\Gamma, \tau)]\!]$), so that we need to show a). By the *i.h.*(a) we have that $\mathtt{u} \Vdash \mathtt{T}(\Gamma, \tau)$, and we conclude that $\lambda\mathtt{x}.\mathtt{u} \Vdash \mathtt{T}(\Gamma \setminus \mathtt{x}, \Gamma(\mathtt{x}) \to \tau))$ by rule ($\mathtt{Abs}$).
- Case ($\to_{\mathtt{E}}$). Then $\mathtt{t} = \mathtt{x}\mathtt{u}_1 ... \mathtt{u}_n$ for some $n > 0$, where the $\mathtt{u}_i$'s are in turn normal forms ($1 \leq i \leq n, n \geq 0$). There is a multi type $\mathtt{A}_n$ such that the standard typing derivation Π has the following form:

$$\frac{\Pi^1_{st} \triangleright \Gamma_1 \vdash \mathtt{x}\mathtt{u}_1 ... \mathtt{u}_{n-1} : \mathtt{A}_n \to \tau \qquad \Pi^2_{st} \triangleright \Gamma_2 \vdash \mathtt{u}_n : \mathtt{A}_n}{\Gamma \vdash \mathtt{x}\mathtt{u}_1 ... \mathtt{u}_n : \tau} \ (\to_{\mathtt{E}})$$

  where $\Gamma = \Gamma_1 + \Gamma_2$. We analyse two cases:
  - $\mathtt{t} \in [\![\mathtt{H}^{\mathtt{x}:[T_\tau]}(\Gamma', \tau)]\!]$, where $\Gamma = \Gamma' + \mathtt{x} : [T_\tau]$. Then $\Gamma_1 = \Gamma'_1 + \mathtt{x} : [T_\tau]$, and $\Gamma' = \Gamma'_1 + \Gamma_2$. By the *i.h.*(c), $\mathtt{x}\mathtt{u}_1 ... \mathtt{u}_{n-1} \Vdash \mathtt{H}^{\mathtt{x}:[T_\tau]}(\Gamma'_1, \mathtt{A}_n \to \tau)$.

If $\mathtt{A}_n = [\,]$, then Π standard implies $\mathtt{u}_n = \mathtt{I}$, and the result then follows by rule $(\mathtt{Head}^{[\,]}_{>0})$.
If $\mathtt{A}_n \neq [\,]$, then by the *i.h.*(b) $\mathtt{u}_n \Vdash \mathtt{TI}(\Gamma_2, \mathtt{A}_n)$, so we get $\mathtt{t} \Vdash \mathtt{H}^{\mathtt{x}:[T_\tau]}(\Gamma', \tau)$ by rule $(\mathtt{Head}_{>0})$.

– $\mathtt{t} \in [\![\mathtt{T}(\Gamma, \tau)]\!]$. Then $\mathtt{t} \in [\![\mathtt{H}^{\mathtt{x}:[T_\tau]}(\Gamma', \tau)]\!]$ for some type T_τ such that $\Gamma = \Gamma' + \mathtt{x} : [T_\tau]$ as remarked after Definition 3.17. Then we conclude $\mathtt{t} \Vdash \mathtt{H}^{\mathtt{x}:[T_\tau]}(\Gamma', \tau)$ by the previous point and $\mathtt{t} \Vdash \mathtt{T}(\Gamma, \tau)$ by rule $(\mathtt{Head})$.

- Case $(\mathtt{m})$: Then $\Pi \triangleright \Gamma \vdash \mathtt{t} : \mathtt{A}$ implies we have a standard derivation of the following form:
$$\frac{(\Pi^i_{st} i \triangleright \Delta_i \vdash \mathtt{t} : \sigma_i)_{i \in I}}{\Pi_{st} \triangleright +_{i \in I} \Delta_i \vdash \mathtt{t} : [\sigma_i]_{i \in I}} (\mathtt{m})$$
with $\Gamma = +_{i \in I} \Delta_i$ and $\mathtt{A} = [\sigma_i]_{i \in I}$. By the *i.h.*(a) we have that for all $i \in I$, $\mathtt{t} \Vdash \mathtt{T}(\Delta_i, \sigma_i)$, hence we conclude $\mathtt{t} \Vdash \mathtt{TI}(+_{i \in I} \Delta_i, [\sigma_i]_{i \in I})$ by rule $(\mathtt{Union})$. □

5. Conclusion

In this paper we have studied the inhabitation problem for some intersection type assignment systems for the λ-calculus, where intersection is considered modulo associativity and commutativity, but not idempotency. We proved that the problem is decidable in all the considered cases. There is a plethora of intersection type assignment systems that can be classified with respect to their semantic power, *i.e.* the class of terms they characterize and their logical aspects. Concerning the latter, we focus on the lack or the presence of weakening and of the empty multitype; for the former, we consider systems characterizing either the solvable or the strongly normalizing terms.

Figure 13 represents all the systems we took into consideration, lying on six vertices of the unitary cube, whose axes represent the following features (the feature being present if the corresponding coordinate has value 1):

x-axis: weakening,
y-axis: empty multitype,
z-axis: characterizing strong normalization.

The unoccupied vertex $\bullet$ (resp. $\circ$) corresponds to a system without (resp. with) weakening and without the empty type, that does not characterize strong normalization. Since such systems do not seem pertinent, the cube of the three features reduces to a prism. We have shown that the inhabitation problem is decidable for all the systems of the prism: for those enjoying the subject reduction property, *i.e.* for $\mathcal{H}$, $\mathcal{H}_w$, $\mathcal{S}_w$ and $\mathcal{H}_{e,w}$, this is done by showing the soundness and completeness of a suitable terminating algorithm. On the other hand, the inhabitation problem of $\mathcal{H}_e$ reduces to that of $\mathcal{H}_{e,w}$, and the one of $\mathcal{S}$ reduces to that of $\mathcal{S}_w$. Remark that the inhabitation problem of $\mathcal{H}$ does not reduces to that of $\mathcal{H}_w$, since the latter inhabits stricly more typings than the former, *e.g.* the typing $(\mathtt{x} : [\sigma, \tau], \sigma)$ is inhabited in $\mathcal{H}_w$ but not in $\mathcal{H}$.

The starting point is the type system $\mathcal{H}$, which characterizes the class of solvable terms, and whose inhabitation algorithm has been originally presented in [6]. That algorithm turns out to be remarkably stable with respect to addition or deletion of all the considered features. Showing its robustness is one of the point of this work. Another remarkable fact is that seemingly hard inhabitation problems, like those of $\mathcal{H}_e$ and $\mathcal{S}_w$, become easily tractable by simply adding weakening to the type system.

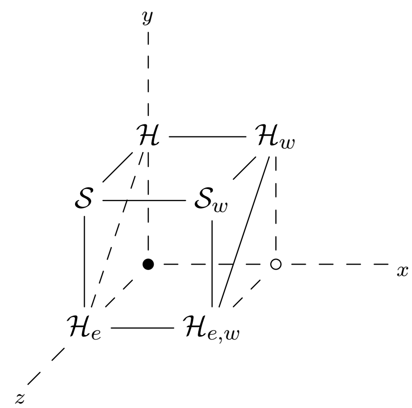

Figure 13: A prism of non-idempotent type assignment systems

Acknowledgements. We would like to thank the anonymous reviewers for their useful criticisms and suggestions.

References

[1] T. Balabonski, P. Barenbaum, E. Bonelli, and D. Kesner. Foundations of strong call by need. *Proceedings of the ACM on Programming Languages (ICFP)*:20:1–20:29, 2017.

[2] H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, Amsterdam, revised edition, 1984.

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

[4] A. Bernadet and S. Lengrand. Non-idempotent intersection types and strong normalisation. *Logical Methods in Computer Science*, 9(4), 2013.

[5] A. Bucciarelli, T. Ehrhard, and G. Manzonetto. Not enough points is enough. In *CSL 2007*, volume 4646 of *LNCS*, pages 298–312. Springer, 2007.

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

[7] A. Bucciarelli, D. Kesner, and D. Ventura. Non-idempotent intersection types for the lambda-calculus. *Logic Journal of the IGPL*, 25(4):431–464, 2017.

[8] M. W. Bunder. The inhabitation problem for intersection types. In *CATS*, volume 77 of *CRPIT*, pages 7–14. Australian Computer Society, 2008.

[9] D. D. Carvalho. *Sèmantique de la logique linéaire et temps de calcul*. PhD thesis, Université de la Mediterranée Aix-Marseille 2, 2007.

[10] M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame Journal Formal Logic*, 21(4):685–693, 1980.

[11] E. De Benedetti and S. Ronchi Della Rocca. Bounding normalization time through intersection types. In *ITRS 2012*, pages 48–57. EPTCS, 2013.

[12] P. Di Gianantonio, F. Honsell, and M. Lenisa. A type assignment system for game semantics. *Theoretical Computer Science*, 398:150–169, 2008.

[13] A. Dudenhefner and J. Rehof. Intersection type calculi of bounded dimension. In *POPL 2017*, pages 653–665. ACM, 2017.

[14] T. Ehrhard. The Scott model of linear logic is the extensional collapse of its relational model. *Theoretical Computer Science*, 424:20–45, 2012.

[15] P. Gardner. Discovering needed reductions using type theory. In *TACS 1994*, volume 789 of *LNCS*, pages 555–574. Springer, 1994.

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

[17] D. Kesner. Reasoning about call-by-need by means of types. In *FOSSACS 2016*, volume 9634 of *LNCS*, pages 424–441. Springer, 2016.

[18] D. Kesner and D. Ventura. Quantitative types for the linear substitution calculus. In *TCS 2014*, Volume 8705 of *LNCS*, pages 296-310. Springer, 2014.

[19] D. Kesner and P. Vial. Types as resources for classical natural deduction. In *FSCD 2017*, volume 84 of *LIPIcs*, pages 1–17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[20] A. J. Kfoury. A linearization of the lambda-calculus and consequences. *Journal of Logic and Computation*, 10(3):411–436, 2000.

[21] 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.

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

[23] T. Kurata and M. Takahashi. Decidable properties of intersection type systems. In *TLCA 1995*, volume 902 of *LNCS*, pages 297–311, 1995.

[24] D. Kuśmierek. The inhabitation problem for rank two intersection types. In *TLCA 2007*, volume 4583 of *LNCS*, pages 240–254. Springer, 2007.

[25] H. Mairson and P. M. Neergaard. Types, potency, and idempotency: why nonlinearity and amnesia make a type system work. In *ICFP 2004*, pages 138–149. ACM, 2004.

[26] M. Pagani and S. Ronchi Della Rocca. Linearity, non-determinism and solvability. *Fundamenta Informaticae*, 103:358–373, 2010.

[27] M. Pagani and S. Ronchi Della Rocca. Solvability in resource lambda-calculus. In *FOSSACS 2010*, volume 6014 of *LNCS*, pages 358–373. Springer, 2010.

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

[29] J. Bessai, A. Dudenhefner, B. Düdder, T-C. Chen, U. de'Liguoro, J. Rehof. Mixin Composition Synthesis Based on Intersection Types. In *TLCA 2015*, volume 38 of *LIPIcs*, pages 76–91. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.

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

[31] P. Urzyczyn. The Logic of Persistent Intersection. *Fundamanta Informaticae* 103:303–322, 2010

[32] P. Urzyczyn. Personal communication, 2014.

[33] S. van Bakel. Complete restrictions of the intersection type discipline. *Theoretical Computer Science*, 102:135–163, 1992.

[34] P. Vial. Infinitary intersection types as sequences: A new answer to Klop's problem. In *LICS 2017*, pages 1–12. IEEE Computer Society, 2017.

[35] C.P. Wadsworth. The Relation Between Computational and Denotational Properties for Scott's D_∞-Models of the Lambda-Calculus", *SIAM Journal on Computing*, vol. 5, 3,pages 488–521,1976.

This work is licensed under the Creative Commons Attribution License. To view a copy of this license, visit `https://creativecommons.org/licenses/by/4.0/` or send a letter to Creative Commons, 171 Second St, Suite 300, San Francisco, CA 94105, USA, or Eisenacher Strasse 2, 10777 Berlin, Germany