Inhabitation in simply typed lambda-calculus through a lambda-calculus for proof search

José Espírito Santo, Ralph Matthes, Luís Pinto

Inhabitation in simply typed lambda-calculus through a lambda-calculus for proof search

JOSÉ ESPÍRITO SANTO[†], RALPH MATTHES[‡] and LUÍS PINTO[†]

[†]*Centro de Matemática, Universidade do Minho, Braga, Portugal*
Email: jes@math.uminho.pt, luis@math.uminho.pt

[‡]*Institut de Recherche en Informatique de Toulouse (IRIT), CNRS and University of Toulouse, Toulouse, France*
Email: Ralph.Matthes@irit.fr

A new approach to inhabitation problems in simply typed lambda-calculus is shown, dealing with both decision and counting problems. This approach works by exploiting a representation of the search space generated by a given inhabitation problem, which is in terms of a lambda-calculus for proof search that the authors developed recently. The representation may be seen as extending the Curry–Howard representation of proofs by lambda terms. Our methodology reveals inductive descriptions of the decision problems, driven by the syntax of the proof-search expressions, and produces simple, recursive decision procedures and counting functions. These allow to predict the number of inhabitants by testing the given type for syntactic criteria. This new approach is comprehensive and robust: based on the same syntactic representation, we also derive the state-of-the-art coherence theorems ensuring uniqueness of inhabitants.

1. Introduction

In this paper, we study inhabitation problems in the simply typed λ-calculus, by which we mean both decision problems, like 'does type A have an inhabitant?,' and related questions like counting the inhabitants of a type or predicting finiteness or their number under syntactic criteria (Hindley 1997). We propose a new approach based on a λ-calculus for proof search that the authors developed recently (Espírito Santo et al. 2013, 2016). This is a λ-calculus with fixed points and formal sums, here named $\lambda^{\mathsf{gfp}}_{\Sigma}$, able to represent as a single term the entire space generated by the search for inhabitants for a given type.

Our previous work showed the correctness of this representation: the representation F_A of the search space for a given type A has a meaning $[\![F_A]\!]$ that matches the actual search space S_A. Now this S_A is another λ-term, also representing the search space for A, but pertaining to another calculus λ^{co}_{Σ}, which is our semantic domain. The calculus λ^{co}_{Σ} is the coinductive simply typed λ-calculus, extended with formal sums. We regard λ^{co}_{Σ} as a canonical system, obtained by extending the Curry–Howard paradigm of representation: in the same way as a term of the simply typed λ-calculus represents a proof, a term of λ^{co}_{Σ} is a possibly infinite object representing all runs of the search process. From λ^{co}_{Σ}, we derived in our previous work the calculus $\lambda^{\mathsf{gfp}}_{\Sigma}$, which is a finitary, effective counterpart with an inductive syntax. The existence of the representation F_A entails the regularity of

the search space generated by a type A. In this paper, we show that we can also base on F_A a comprehensive study of proof search and type inhabitation.

We consider two decision problems: the problem exemplified above, studied by several authors, described with more references in books on λ-calculus (Barendregt et al. 2013; Hindley 1997), and also the problem 'does type A have finitely many inhabitants?,' studied in particular in Ben-Yelles (1979) and Takahashi et al. (1996). In each case, we are able to pin down a predicate on $\lambda_\Sigma^{\mathsf{gfp}}$ that characterizes existence – or existence of a finite number – of inhabitants. The correctness of the predicates is always established relative to the semantic domain λ_Σ^{co}, which is our verification tool. In addition, such predicates are *syntax-directed*. So the decision procedure is the composition of the map $A \mapsto F_A$ with the predicate. This not only re-establishes decidability, but sheds an entirely new light on the result: our proof is neither a combinatorial argument nor a reduction to another problem, but rather a structural analysis.

We also consider the problem of counting inhabitants, addressed first in Ben-Yelles (1979). Again, one can define a syntax-directed (recursive) function on the set of $\lambda_\Sigma^{\mathsf{gfp}}$-terms that is proved to return the number of inhabitants of A, when the argument is F_A and the predicate characterizing existence of a finite number of inhabitants holds of F_A. The proof is by establishing that such a *counting function* returns $\#(S_A)$, where S_A is the 'real' search space and $\#$ is a function defined on λ_Σ^{co}-terms that counts inhabitants.

Next, we revisit the question of unique inhabitation, and theorems that predict at most one inhabitant if the type fulfils a certain syntactic criterion, studied in works like Mints (1979), Aoto and Ono (1994), Broda and Damas (2005) and Bourreau and Salvati (2011). Still for simple types with implication as the only connective, we prove the most general form of the so-called coherence theorem, and the counting function is a crucial tool for that. Again, our proof gives a structural analysis: the syntactic criterion the type A fulfils guarantees a certain property of F_A; and we see, by applying the counting function, that such a property of F_A entails uniqueness of inhabitants.

The paper is organized as follows. Section 2 gives a somewhat lengthy review of previous material of ours (Espírito Santo et al. 2013, 2016), providing a reasonably self-contained introduction to our methodology. Section 3 studies the decision problems, while Section 4 is devoted to counting and the coherence theorem. Section 5 concludes.

2. Background

This section has three subsections. First, we fix our presentation of the simply typed λ-calculus, next we recall our two representations of proof search, developed before in Espírito Santo et al. (2013, 2016), and recast here as search for inhabitants of a given type.

2.1. *Simply typed λ-calculus, reduced to normal forms*

It is well-known that η-long β-normal terms are complete for simply typed λ-calculus in the sense that any typable term normalizes to a β-normal form, which in turn can be expanded to an η-long β-normal form (see, e. g., 2D5 and 8A8 of the book Hindley

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x^A.t : A \supset B}\ \mathit{RIntro} \qquad \frac{(x : \vec{B} \supset p) \in \Gamma \quad \forall i,\, \Gamma \vdash t_i : B_i}{\Gamma \vdash x\langle t_i\rangle_i : p}\ \mathit{LVecIntro}$$

Fig. 1. Typing rules of λ.

(1997)). We lay out a presentation of the η-long β-normal fragment of simply typed λ-calculus, a system we often refer to by λ.

Simple types (or simply, types) are given by the grammar:

$$(\mathit{types}) \quad A, B, C \ := \ p \mid A \supset B,$$

where p, q, r range over *atoms*. We thus do not distinguish types from propositional implicational formulas. We will write $A_1 \supset A_2 \supset \cdots \supset A_k \supset p$, with $k \geqslant 0$, in vectorial notation as $\vec{A} \supset p$. For example, if the vector $\vec{A}$ is empty, the notation means simply p.

Normal (i.e., β-normal) λ-terms are given by

$$(\text{terms}) \quad t, u \ ::= \ \lambda x^A.t \mid x\,\langle t_1, \ldots, t_k\rangle,$$

where a countably infinite set of variables, ranged over by letters x, y, w, z, is assumed. Note that in λ-abstractions we adopt a *domain-full* presentation (a. k. a. Church-style syntax), annotating the bound variable with a type. As is common-place with lambda-calculi, we will throughout identify terms up to α-equivalence. The term constructor $x\,\langle t_1, \ldots, t_k\rangle$ is called *application* (traditionally, this would be expressed as a multiple application $xt_1 \ldots t_k$ of λ-calculus). When $k = 0$, we often simply write the variable x.

We view contexts Γ as finite sets of declarations $x : A$, where no variable x occurs twice. The letters Γ, Δ, Θ are used to range over contexts, and the notation $\mathrm{dom}(\Gamma)$ stands for the set of variables declared in Γ. The context $\Gamma, x : A$ is obtained from Γ by adding the declaration $x : A$, and is only written if x is not declared in Γ. Context union is written as concatenation Γ, Δ for contexts Γ and Δ if $\mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Delta) = \emptyset$. We will write $\Gamma(x)$ for the type associated with x for $x \in \mathrm{dom}(\Gamma)$, hence viewing Γ as a function on $\mathrm{dom}(\Gamma)$. Context inclusion $\Gamma \subseteq \Delta$ is just set inclusion.

The typing rules are in Figure 1 and derive sequent $\Gamma \vdash t : A$. *LVecIntro* presupposes that the indices for the t_i range over $1, \ldots, k$ and that $\vec{B} = B_1, \ldots, B_k$, for some $k \geqslant 0$. Such obvious constraints for finite vectors will not be spelt out in the rest of the paper. In the particular case of $k = 0$, in which $(x : p) \in \Gamma$ is the only hypothesis of *LVecIntro*, we type variables (with atoms). Note that the conclusion of the *LVecIntro* rule is an atomic sequent, in other words, a typable term will always correspond to a λ-term in η-long form. This restriction to atoms is typically adopted in systems tailored for proof search, as for example systems of focused proofs. In fact, our presentation of λ corresponds to a focused backward chaining system where all atoms are *asynchronous* (Liang and Miller 2009).

Example 1. The running simple examples of this paper are with the following types:

— INFTY $:= (p \supset p) \supset p$, which is obviously uninhabited in λ-calculus (as is the type p alone)

— CHURCH $:= (p \supset p) \supset p \supset p$, the type of Church numerals $\lambda f^{p \supset p}.\lambda x^p.f^n\langle x\rangle$, $n \geqslant 0$.

More complicated examples are found in our previous work (Espírito Santo et al. 2013, 2016).

2.2. *Search for inhabitants, coinductively*

We are concerned with a specific kind of search problems: given Γ and A, to find t such that $\Gamma \vdash t : A$, that is, to find an *inhabitant* of type A in context Γ. Under the Curry–Howard correspondence, a pair (Γ, A) may be seen as a *logical sequent* $\Gamma \Rightarrow A$, and searching for an inhabitant of A in context Γ is the same as searching for a proof of that sequent.

Following Espírito Santo et al. (2013, 2016), we model this search process through the coinductive λ-calculus, denoted λ^{co}. The terms of λ^{co}, also called *coterms*, are given by

$$M, N ::=_{co} \lambda x^A.N \mid x\langle N_1, \ldots, N_k\rangle \ .$$

This is exactly the previous grammar for λ-terms, but read coinductively, as indicated by the index *co* (still with finite tuples $\langle N_i\rangle_i$). The natural notion of equality between coterms is bisimilarity modulo α-equivalence, which we still write as plain equality. The coterms can be seen as total Böhm-like trees. However, a coterm need not arise as Böhm tree of a λ-term. We therefore use the more neutral name 'coterm' that reminds of being element of a coinductive data structure. This concept has no relation to the dual of terms in the Curry–Howard interpretation of sequent calculus.

In λ^{co}, also the typing rules of Figure 1 have to be interpreted coinductively – but the types stay inductive and the contexts finite. Following common practice in the presentation of coinductive syntax, we will symbolize the coinductive reading of an inference (rule) by the double horizontal line, but we refrain from displaying Figure 1 again with double lines – a figure where the two inference rules would be called $RIntro_{co}$ and $LVecIntro_{co}$. Such a system defines when $\Gamma \vdash N : A$ holds for a *finite* context Γ, a coterm N and a type A.

Suppose $\Gamma \vdash N : A$ holds. Then this sequent has a derivation which is a (possibly infinite) tree of sequents, generated by applying the inference rules bottom-up; and N is a (possibly infinite) coterm, which we call a *solution* of σ, with $\sigma = (\Gamma \Rightarrow A)$. Therefore, such derivations are the structures generated by the search process which does not fail, even if it runs forever, and so they subsume proofs; likewise solutions subsume typable terms, so we may refer to the latter as *finite* solutions. The next step is to extend even further the paradigm, representing also the choice points of the search process. To this end, we extend λ^{co} to λ^{co}_{Σ}, whose syntax is this

$$\begin{array}{lrl} \text{(terms)} & M, N & ::=_{co} \ \lambda x^A.N \mid E_1 + \cdots + E_n \\ \text{(elimination alternatives)} & E & ::=_{co} \ x\langle N_1, \ldots, N_k\rangle \end{array}$$

where both $n, k \geqslant 0$ are arbitrary (thus including the empty sum of elimination alternatives). T ranges over both terms and elimination alternatives. Note that summands cannot be λ-abstractions. We will often use $\sum_i E_i$ instead of $E_1 + \cdots + E_n$ – in generic situations or if the dependency of E_i on i is clear, as well as the number of elements.

The most natural notion of equality of terms in λ^{co}_{Σ} is again bisimilarity modulo α-equivalence, but the notation $\sum_i E_i$ already hints that we consider $+$ to be associative.

$$\frac{\mathsf{mem}(M,N)}{\mathsf{mem}(\lambda x^A.M, \lambda x^A.N)} \qquad \frac{\mathsf{mem}(M,E_j)}{\mathsf{mem}(M,\sum_i E_i)} \qquad \frac{\forall i,\ \mathsf{mem}(M_i,N_i)}{\mathsf{mem}(x\langle M_i\rangle_i, x\langle N_i\rangle_i)}$$

Fig. 2. Membership relations.

$$\frac{\forall i,\ \Gamma \vdash E_i : p}{\Gamma \vdash \sum_i E_i : p}\ \mathit{Alts}$$

Fig. 3. Extra typing rule of λ^{co}_{Σ} w. r. t. λ^{co}.

We even want to neglect the precise order of the summands and their (finite) multiplicity. We thus consider the sums of elimination alternatives as if they were sets of alternatives, i. e., we further assume that $+$ is symmetric and idempotent. As for λ^{co}, we just use mathematical equality for this notion of bisimilarity on expressions of λ^{co}_{Σ}, and so the sums of elimination alternatives can plainly be treated as if they were finite sets of elimination alternatives (given by finitely many elimination alternatives of which several might be identified through bisimilarity). See Section 4.2 for a more restricted reading in order to make counting possible.

We call *forests* the expressions of λ^{co}_{Σ} – and a coterm M is a member of a forest N when the relation $\mathsf{mem}(M,N)$, defined coinductively in Figure 2 holds. (In Espírito Santo et al. (2013), we called the forests 'Böhm forests' to hint to their coinductive nature.)

In the typing system for λ^{co}_{Σ}, one derives sequents $\Gamma \vdash N : A$ and $\Gamma \vdash E : p$. The coinductive typing rules are the ones of λ^{co}, together with the rule given in Figure 3 (the empty sum of elimination alternatives receives any atom as type).

A typing derivation of λ^{co}_{Σ} is a possibly infinite tree of sequents, generated by the bottom-up application of the inference rules, with 'multiplicative' branching (logically: 'and' branching) caused by the list of arguments in elimination alternatives, and 'additive' branching (logically: 'or' branching) caused by sums – the latter being able to express the alternatives found in the search process when an atom p can be proved by picking different head variables with their appropriate arguments. So, it is no surprise that, with this infrastructure, we can express, as a (single) forest, the entire *solution space* generated by the search process when applied to given Γ and A. That forest can be defined as a function $\mathcal{S}$ of $\Gamma \Rightarrow A$ defined by corecursion as follows:

Definition 2 (Solution spaces).

$$\mathcal{S}(\Gamma \Rightarrow \vec{A} \supset p) := \lambda \vec{x} : \vec{A}.\ \sum_{(y:\vec{B}\supset p)\in\Delta} y\langle \mathcal{S}(\Delta \Rightarrow B_j)\rangle_j \quad \text{with } \Delta := \Gamma, \vec{x} : \vec{A}$$

Notice that, in the recursive calls to $\mathcal{S}$, the contexts are (weakly) monotonically increasing on each branch. While we are here only concerned with propositional logic, this is reminiscent of the dynamic nature of the set of facts one has in higher order logic programming (Miller and Nadathur 2012).

The following properties witness the robustness of the definition (Espírito Santo et al. 2013, 2016).

Proposition 1 (Properties of solution spaces). The following properties hold:

1. Given Γ and A, the typing $\Gamma \vdash \mathcal{S}(\Gamma \Rightarrow A) : A$ holds in λ^{co}_{Σ}.
2. For $N \in \lambda^{co}$, $\mathsf{mem}(N, \mathcal{S}(\Gamma \Rightarrow A))$ iff $\Gamma \vdash N : A$ in λ^{co}.
3. For $t \in \lambda$, $\mathsf{mem}(t, \mathcal{S}(\Gamma \Rightarrow A))$ iff $\Gamma \vdash t : A$ in λ.

In particular, the last two properties say that solutions subsume finite solutions conservatively, i.e., given a λ-term t, $\Gamma \vdash t : A$ in λ iff $\Gamma \vdash t : A$ in λ^{co}.

Example 3. The following forests are obtained by unfolding the definition for the two running examples:

— Consider $it^{\infty} := \lambda f^{p \supset p}.N$ with $N = f\langle N \rangle$ (this term N exists as an infinitely repeated application of f). Using ν as means to communicate solutions of fixed-point equations on the *meta-level*, we may write $it^{\infty} := \lambda f^{p \supset p}.\nu N.f\langle N \rangle$. Then, $\mathcal{S}(\Rightarrow \mathsf{INFTY}) = it^{\infty}$ – both sides under the initial lambda-abstraction solve the same equation.

— $\mathcal{S}(\Rightarrow \mathsf{CHURCH}) = \lambda f^{p \supset p}.\lambda x^{p}.\mathcal{S}(f : p \supset p, x : p \Rightarrow p)$ with $\mathcal{S}(f : p \supset p, x : p \Rightarrow p)$ the solution for N of the equation $N = f\langle N \rangle + x$. In other words, $\mathcal{S}(\Rightarrow \mathsf{CHURCH})$ is the forest $\lambda f^{p \supset p}.\lambda x^{p}.\nu N.f\langle N \rangle + x$.

2.3. *Search for inhabitants, inductively*

Finitary algorithms cannot in general process forests as input, so the next step is to find an alternative, equivalent, effective representation that works at least for solution spaces. To this end, an extension $\lambda^{\mathsf{gfp}}_{\Sigma}$ of λ is introduced, whose syntax is given by the following grammar (read inductively):

$$\begin{array}{lll} \text{(terms)} & N & ::= \lambda x^{A}.N \mid \mathsf{gfp}\ X^{\sigma}.E_1 + \cdots + E_n \mid X^{\sigma} \\ \text{(elimination alternatives)} & E & ::= x\langle N_1, \ldots, N_k \rangle \end{array}$$

where X is assumed to range over a countably infinite set of *fixpoint variables* (also letters Y, Z will range over them), and where, as for λ^{co}_{Σ}, both $n, k \geqslant 0$ are arbitrary. We extend our practice established for λ^{co}_{Σ} of writing the sums $E_1 + \cdots + E_n$ in the form $\sum_i E_i$ for $n \geqslant 0$. As for λ^{co}_{Σ}, we will identify expressions modulo associativity, symmetry and idempotence of $+$, thus treating sums of elimination alternatives as if they were the set of those elimination alternatives. Again, we will write T for any expression of $\lambda^{\mathsf{gfp}}_{\Sigma}$, comprising N and E.

In the term formation rules, σ in X^{σ} is required to be *atomic*, i.e., of the form $\Gamma \Rightarrow p$. Let $\mathrm{FPV}(T)$ denote the set of free occurrences of typed fixed-point variables in T, i.e.,

$$\begin{array}{rcl} \mathrm{FPV}(X^{\sigma}) & = & \{X^{\sigma}\} \\ \mathrm{FPV}(\lambda x^{A}.N) & = & \mathrm{FPV}(N) \\ \mathrm{FPV}(x\langle N_1, \ldots, N_k \rangle) & = & \bigcup_i \mathrm{FPV}(N_i) \\ \mathrm{FPV}(\mathsf{gfp}\ X^{\sigma}.E_1 + \cdots + E_n) & = & (\bigcup_i \mathrm{FPV}(E_i)) \setminus \{X^{\sigma'} \mid \sigma' \text{ atomic sequent}\} \end{array}$$

Perhaps unexpectedly, in $\mathsf{gfp}\ X^{\sigma}.\sum_i E_i$, the fixed-point construction gfp binds *all* free occurrences of $X^{\sigma'}$ in the elimination alternatives E_i, not just X^{σ}. But we only want this to happen when $\sigma \leqslant \sigma'$ – which means: the context of σ' has more declarations than that of σ, but not with new types. Formally:

$$
\begin{aligned}
[\![X^{\sigma'}]\!]_\xi &= [\sigma'/\sigma]\xi(X^\sigma) \quad \text{for the unique } \sigma \leqslant \sigma' \text{ with } X^\sigma \in \mathrm{dom}(\xi) \\
[\![\mathsf{gfp}\ X^\sigma.\textstyle\sum_i E_i]\!]_\xi &= \nu N.\textstyle\sum_i [\![E_i]\!]_{\xi \cup [X^\sigma \mapsto N]} \\
[\![\lambda x^A.N]\!]_\xi &= \lambda x^A.[\![N]\!]_\xi \\
[\![x\langle N_i\rangle_i]\!]_\xi &= x\langle [\![N_i]\!]_\xi\rangle_i
\end{aligned}
$$

Fig. 4. Interpretation of finitary forests.

Definition 4 (Inessential extension of contexts and sequents).

1. $\Gamma \leqslant \Gamma'$ iff $\Gamma \subseteq \Gamma'$ and $|\Gamma| = |\Gamma'|$, with the set $|\Delta| := \{A \mid \exists x, (x : A) \in \Delta\}$ of assumed types of Δ, for an arbitrary context Δ.
2. $\sigma \leqslant \sigma'$ iff for some $\Gamma \leqslant \Gamma'$ and for some atom p, $\sigma = (\Gamma \Rightarrow p)$ and $\sigma' = (\Gamma' \Rightarrow p)$.

In the sequel, when we refer to *finitary forests*, we have in mind the expressions of $\lambda_\Sigma^{\mathsf{gfp}}$. The fixed-point operator is called gfp (greatest fixed point) to indicate that its semantics is defined in terms of the *infinitary* syntax λ_Σ^{co}, but there, fixed points are unique. Hence, the reader may just read this as 'the fixed point.'

Although we will use a simplified semantics for the results in the later sections, we now recall the interpretation of expressions of $\lambda_\Sigma^{\mathsf{gfp}}$ in terms of the coinductive syntax of λ_Σ^{co} (Espírito Santo et al. 2016), using the ν operation on the meta-level to designate unique fixed points. It is done with the help of *environments* ξ, which are partial functions from typed fixed-point variables X^σ to (co)terms of λ_Σ^{co}, with domain $\mathrm{dom}(\xi)$ a finite set of typed fixpoint variables *without duplicates*, which means: $X^{\sigma_1}, X^{\sigma_2} \in \mathrm{dom}(\xi) \Rightarrow \sigma_1 = \sigma_2$.

Some technicalities are needed before giving the interpretation. We say an environment ξ is *admissible* for an expression T of $\lambda_\Sigma^{\mathsf{gfp}}$ if, for every $X^{\sigma'} \in \mathrm{FPV}(T)$, there is an $X^\sigma \in \mathrm{dom}(\xi)$ such that $\sigma \leqslant \sigma'$. It is easy to see that T admits an environment iff it is *regular* in the following sense: if X occurs free in T, there is a sequent σ that is the minimum of all σ' such that $X^{\sigma'} \in \mathrm{FPV}(T)$.

The interpretation is only given for well-bound expressions, where $T \in \lambda_\Sigma^{\mathsf{gfp}}$ is *well-bound* if, for any of its subterms $\mathsf{gfp}\ X^\sigma.\sum_i E_i$ and any (free) occurrence of $X^{\sigma'}$ in the E_i's, $\sigma \leqslant \sigma'$.

Definition 5 (Interpretation of finitary forests as forests). For a well-bound expression T of $\lambda_\Sigma^{\mathsf{gfp}}$, the interpretation $[\![T]\!]_\xi$ for an environment ξ that is admissible for T is given by structural recursion on T in Figure 4, where the notation $[\_/\_]\_$ does not stand for ordinary substitution, but rather refers to the *decontraction* operation explained next.†

The clause for fixpoint variables in this definition has to cope with $\sigma \leqslant \sigma'$. This is done by adjusting the value $N = \xi(X^\sigma)$ looked up in the environment with the following decontraction operation on forests which adds elimination alternatives to the sums in N, in order to match the new declarations in σ'. If $\sigma = (\Gamma \Rightarrow p)$ and $\sigma' = (\Gamma' \Rightarrow p)$, then $[\sigma'/\sigma]N$ is defined to be $[\Gamma'/\Gamma]N$, with the latter given as follows:

† In Espírito Santo et al. (2013, 2016), the decontraction operation was named *co-contraction*. The new name reflects better the inverse nature of this operation w. r. t. *contraction*, and avoids possible associations to a dual concept or any relation with coinductive structures.

Definition 6 (Decontraction). Let $\Gamma \leqslant \Gamma'$. For T an expression of λ^{co}_{Σ}, we define the expression $[\Gamma'/\Gamma]T$ of λ^{co}_{Σ} by corecursion as follows:

$$\begin{array}{lcll}
[\Gamma'/\Gamma](\lambda x^A.N) & = & \lambda x^A.[\Gamma'/\Gamma]N & \\
[\Gamma'/\Gamma]\sum_i E_i & = & \sum_i [\Gamma'/\Gamma]E_i & \\
[\Gamma'/\Gamma]\big(z\langle N_i\rangle_i\big) & = & z\langle[\Gamma'/\Gamma]N_i\rangle_i & \text{if } z \notin \mathrm{dom}(\Gamma) \\
[\Gamma'/\Gamma]\big(z\langle N_i\rangle_i\big) & = & \sum_{(w:A)\in\Delta_z} w\langle[\Gamma'/\Gamma]N_i\rangle_i & \text{if } z \in \mathrm{dom}(\Gamma)
\end{array}$$

where, in the last clause, $A := \Gamma(z)$ and $\Delta_z := \{(z : A)\} \cup (\Gamma' \setminus \Gamma)$.‡

Decontraction precisely captures the extension of the solution space when going from σ to some σ' with $\sigma \leqslant \sigma'$:

Lemma 7 (Solution spaces and decontraction). Let $\sigma \leqslant \sigma'$. Then, $\mathcal{S}(\sigma') = [\sigma'/\sigma]\mathcal{S}(\sigma)$.

If T is closed, i. e., $\mathrm{FPV}(T) = \emptyset$, then the empty function is an admissible environment for T. For a well-bound expression T that is also closed, we write $[\![T]\!]$ for the interpretation of T w. r. t. the empty function, and this is a forest.

With the finitary forests and their semantics in place, we can provide an alternative representation $\mathcal{F}(\sigma)$ of the search space generated by a sequent σ.

Definition 8 (Finitary solution space). Let $\Xi := \overrightarrow{X : \Theta \Rightarrow q}$ be a vector of $m \geqslant 0$ declarations $(X_i : \Theta_i \Rightarrow q_i)$ where no fixpoint variable name and no sequent occurs twice. The specification of $\mathcal{F}(\sigma;\Xi)$ is as follows, with $\sigma = (\Gamma \Rightarrow \vec{A} \supset p)$:

If, for some $1 \leqslant i \leqslant m$, $p = q_i$ and $\Theta_i \subseteq \Gamma$ and $|\Theta_i| = |\Gamma| \cup \{A_1, \ldots, A_n\}$, then

$$\mathcal{F}(\sigma;\Xi) = \lambda z_1^{A_1} \cdots z_n^{A_n}.X_i^{\sigma'} \ ,$$

where i is taken to be the biggest such index. Otherwise,

$$\mathcal{F}(\sigma;\Xi) = \lambda z_1^{A_1} \cdots z_n^{A_n}.\mathsf{gfp}\ Y^{\sigma'}.\sum_{(y:\vec{B}\supset p)\in\Delta} y\langle\mathcal{F}(\Delta \Rightarrow B_j;\Xi, Y\ :\sigma')\rangle_j,$$

where in both cases, $\Delta := \Gamma, z_1 : A_1, \ldots, z_n : A_n$ and $\sigma' := \Delta \Rightarrow p$.

Notice that, in the first case, X_i occurs with sequent σ' in the resulting finitary forest instead of with $\Theta_i \Rightarrow p$ in Ξ, but $\Theta_i \leqslant \Delta$, hence $(\Theta_i \Rightarrow p) \leqslant \sigma'$ makes it plausible that this process generates well-bound terms.

$\mathcal{F}(\sigma)$ denotes $\mathcal{F}(\sigma;\Xi)$ with empty Ξ. It can be proved that (i) $\mathcal{F}(\sigma)$ is well-defined (the above recursive definition terminates); (ii) $\mathcal{F}(\sigma)$ is a closed well-bound term.

Example 9. The finitary representations (i.e., by way of finitary forests) for our running examples are

‡ Note some abuse of notation in this definition: while decontraction associates terms with terms, it associates with elimination alternatives either a sum of elimination alternatives or a single one (in the case of the third clause). Hence, in the second clause, some flattening is required before forming the sum, but this is left implicit. However, recall that sums are treated as if they were sets of elimination alternatives, so this flattening operation need not be detailed.

— $\mathcal{F}(\Rightarrow \mathsf{INFTY}) = \lambda f^{p\supset p}.\mathsf{gfp}\ X^{f:p\supset p\Rightarrow p}.f\langle X^{f:p\supset p\Rightarrow p}\rangle$,
— $\mathcal{F}(\Rightarrow \mathsf{CHURCH}) = \lambda f^{p\supset p}.\lambda x^{p}.\mathsf{gfp}\ X^{\sigma}.(f\langle X^{\sigma}\rangle + x)$ with $\sigma := (f : p\supset p, x : p\Rightarrow p)$.

So the meta-level fixpoints of the coinductive representations turn into formal fixpoints of the finitary calculus. This is not surprising. Theorem 10 ensures that we always get expressions with formal fixpoints. If no decontraction phenomena appear, the solution spaces can then already be presented with meta-level fixpoints as in these simple examples, but this already presupposes a form of cycle analysis.

The semantics into λ^{co}_{Σ} of the finitary representation coincides with $\mathcal{S}(\sigma)$ (Espírito Santo et al. 2013, 2016).

Theorem 10 (Equivalence). For any sequent σ, $[\![\mathcal{F}(\sigma)]\!] = \mathcal{S}(\sigma)$.

This theorem can be read as saying that the solution spaces $\mathcal{S}(\sigma)$ are regular, in that they are the semantics of finitary forests. This is a weak form of regularity in view of the decontraction operation that is applied in the semantics. The theorem particularly depends on the well-definedness of $\mathcal{F}(\sigma)$, proved by a not so obvious argument given in full in our previous work (Espírito Santo et al. 2013, 2016); that proof, in turn, crucially exploits the subformula property of β-normal lambda terms.

Example 11. We illustrate this fundamental theorem with an example of an expansion on Church numerals that has been studied in two of our important references (Takahashi et al. 1996, Examples 2.11 and 3.6) (Broda and Damas 2005, Example 4.6). Pick different atoms p, q, set $A_0 := p\supset q$, $A_1 := A_0\supset A_0$ and $A_2 := A_1\supset A_1$. Thus, A_2 is obtained from CHURCH by replacing p by A_0. We calculate

$$\begin{aligned}\mathcal{F}(\Rightarrow A_2) \;=\; \lambda x^{A_1}\lambda y^{A_0}\lambda z^{p}.\mathsf{gfp}\,X^{\sigma_1}.\;\; & x\langle \mathcal{F}(\Delta_1\Rightarrow A_0; X^{\sigma_1}), \mathcal{F}(\Delta_1\Rightarrow p; X^{\sigma_1})\rangle \;+\\ & y\langle \mathcal{F}(\Delta_1\Rightarrow p; X^{\sigma_1})\rangle\end{aligned}$$

with $\Delta_1 := x : A_1, y : A_0, z : p$ and $\sigma_1 := \Delta_1\Rightarrow q$.

$$\begin{array}{lll}\mathcal{F}(\Delta_1\Rightarrow A_0; X^{\sigma_1}) \;=\; \lambda w^{p}.\mathcal{F}(\Delta_2\Rightarrow q; X^{\sigma_1}) & \text{with } \Delta_2 := \Delta_1, w : p. \text{ Notice } \Delta_1 \leqslant \Delta_2.\\ \mathcal{F}(\Delta_2\Rightarrow q; X^{\sigma_1}) \;=\; X^{\sigma_2} & \text{with } \sigma_2 := \Delta_2\Rightarrow q\\ \mathcal{F}(\Delta_1\Rightarrow p; X^{\sigma_1}) \;=\; \mathsf{gfp}\,Y^{\sigma_1'}.z & \text{with } \sigma_1' := \Delta_1\Rightarrow p\end{array}$$

We tacitly simplify $\mathsf{gfp}\,Y^{\sigma_1'}.z$ to z. Hence,

$$\mathcal{F}(\Rightarrow A_2) \;=\; \lambda x^{A_1}\lambda y^{A_0}\lambda z^{p}.\mathsf{gfp}\,X^{\sigma_1}.x\langle \lambda w^{p}.X^{\sigma_2}, z\rangle + y\langle z\rangle.$$

By the equivalence theorem, $\mathcal{S}(\Rightarrow A_2) = [\![\mathcal{F}(\Rightarrow A_2)]\!] = \lambda x^{A_1}\lambda y^{A_0}\lambda z^{p}.N$ with N the solution of $N = x\langle \lambda w^{p}.[\sigma_2/\sigma_1]N, z\rangle + y\langle z\rangle$. This should be compared with the production rules for the *standard* inhabitants in Broda and Damas (2005, p. 377), which are followed by an expansion to all η-long β-normal inhabitants. The result of Takahashi et al. (1996, Example 2.11) is an infinite grammar whose description relies on ellipsis, and mathematical intuition is needed to obtain a compact description. To the contrary, the result of Takahashi et al. (1996, Example 3.6) is finite but is based on the *total discharge convention* which restricts simply typed λ-calculus to a single variable per type (see also the discussion in the final section).

3. The inhabitation problems

We will study two decision problems in simply typed λ-calculus: the inhabitation problem (=type emptiness problem) and the type finiteness problem. First, we lay down the common approach we will adopt to solve both problems. Next, we introduce a simplification of the semantics for $\lambda_\Sigma^{\mathsf{gfp}}$ that we will employ in our proofs. After these two preliminary subsections, we study the inhabitation and the finiteness problems in this order, since the solution of the former is used in the solution of the latter. In each case, an analysis of the obtained algorithm is offered, through a comparison with classical algorithms for emptiness and finiteness for context-free languages (recall the inhabitation and the type finiteness problems have been reduced to these problems in language theory (Takahashi et al. 1996)).

3.1. *The common approach*

Given a sequent $\sigma = (\Gamma \Rightarrow A)$, we will write $\mathcal{I}(\sigma)$ for the set of inhabitants of A relative to context Γ in λ, i.e., for the set $\{t \in \lambda \mid \Gamma \vdash t : A \text{ in } \lambda\}$. Recall that this describes the set of η-long β-normal terms of ordinary simply typed λ-calculus receiving type A in context Γ.

For $T \in \lambda_\Sigma^{co}$, we call *finite extension* of T, which we denote by $\mathcal{E}_{\text{fin}}(T)$, the set of the finite members of T, i.e., $\mathcal{E}_{\text{fin}}(T) = \{t \in \lambda \mid \mathsf{mem}(t, T)\}$.

Lemma 12 (Finite extension of forests).

— $\mathcal{E}_{\text{fin}}(\lambda x^A.N) = \{\lambda x^A.t \mid t \in \mathcal{E}_{\text{fin}}(N)\}$,
— $\mathcal{E}_{\text{fin}}(\sum_i E_i) = \bigcup_i \mathcal{E}_{\text{fin}}(E_i)$,
— $\mathcal{E}_{\text{fin}}(x\langle N_i\rangle_i) = \{x\langle t_i\rangle_i \mid \forall i, t_i \in \mathcal{E}_{\text{fin}}(N_i)\}$.

Proof. Obvious. Notice that, because of the coinductive nature of forests, these equations do not constitute a structurally recursive definition of the finite extension. □

We will be mainly interested in the following predicates on forests concerning the finite extension:

— $\mathsf{exfinext}(T) :\Leftrightarrow \mathcal{E}_{\text{fin}}(T)$ is non-empty. $\mathsf{nofinext}(T) :\Leftrightarrow \mathcal{E}_{\text{fin}}(T)$ is empty.
— $\mathsf{finfinext}(T) :\Leftrightarrow \mathcal{E}_{\text{fin}}(T)$ is finite. $\mathsf{inffinext}(T) :\Leftrightarrow \mathcal{E}_{\text{fin}}(T)$ is infinite.

The predicates exfinext and finfinext will be characterized inductively in Sections 3.3 and 3.4, respectively, together with coinductive characterizations of nofinext and inffinext by the generic De Morgan's law relating least and greatest fixed points.

Observe that, due to Proposition 1.3 and Theorem 10,

$$\mathcal{I}(\sigma) = \mathcal{E}_{\text{fin}}(\mathcal{S}(\sigma)) = \mathcal{E}_{\text{fin}}([\![\mathcal{F}(\sigma)]\!]). \tag{1}$$

The *inhabitation problem* in simply typed λ-calculus is the problem 'given sequent σ, is the set $\mathcal{I}(\sigma)$ nonempty?,' called INHAB in this paper. Its negation is called the 'emptiness problem' (as is well-known, the answer to this question does not depend on whether all λ-terms are considered or only the β-normal ones or even the η-long β-normal terms). Decidability of the inhabitation problem in simply typed λ-calculus is a well-known result. The first instance of interest of the Equation (1) is worth a lemma.

Lemma 13 (Characterization of existence of inhabitants in λ). The set of inhabitants $\mathcal{I}(\sigma)$ is non-empty iff $\mathsf{exfinext}([\![\mathcal{F}(\sigma)]\!])$.

As seen above, the function $\mathcal{F}$ is effectively computable, and it yields closed well-bound terms among the finitary forests. The missing link to deciding INHAB is thus the decision of the problem 'given a closed well-bound term T, does $\mathsf{exfinext}([\![T]\!])$ hold?.' Of course, one cannot deal with closed finitary forests T in isolation and needs to address fixpoint variables properly. Neither the interpretation function $[\![\cdot]\!]$ nor the predicate exfinext are effective, but we will define in Section 3.3 a syntax-directed predicate EF (more precisely, it will be a predicate EF_P parameterized over a decidable predicate P) on finitary forests that is equivalent to the composition $\mathsf{exfinext} \circ [\![\cdot]\!]$, for at least those closed well-bound terms that arise as $\mathcal{F}(\sigma)$ for some sequent σ (technically, the restriction will be to proper terms, as defined in Section 3.2). Syntax-directedness immediately entails that the predicate is decidable.

The appeal of our approach is that, once the finitary representation of the corresponding sequent has been built as $\mathcal{F}(\sigma)$, the decision of inhabitation is achieved through a simple recursive function over the structure of $\lambda_\Sigma^{\mathsf{gfp}}$-terms, corresponding to an inductive predicate adequately characterizing non-emptiness of types.

Using the same methodology, we can also reprove a more difficult and not so well-known result of inhabitation for simply typed λ-calculus, namely, that the problem 'given sequent σ, is the set $\mathcal{I}(\sigma)$ finite?' is decidable. This problem – henceforth called FINHAB – depends on studying only β-normal terms; to recall, the inhabitants of our system λ are η-long β-normal simply typed λ-terms, for which the problem is studied in the literature Hindley (1997) (there, in particular, the algorithm by Ben-Yelles Ben-Yelles (1979)). The second instance of Equation (1) is again worth a lemma.

Lemma 14 (Characterization of type finiteness in λ). The set of inhabitants $\mathcal{I}(\sigma)$ is finite iff $\mathsf{finfinext}([\![\mathcal{F}(\sigma)]\!])$.

Analogously to the emptiness problem, our method for establishing decidability of FINHAB is to define a recursive predicate on finitary forests that is equivalent to the composition $\mathsf{finfinext} \circ [\![\cdot]\!]$, for at least those closed well-bound terms that arise as $\mathcal{F}(\sigma)$ for some sequent σ (with the same technical condition as for the emptiness problem). This will be the predicate FF (again, rather a parameterized predicate FF_P), studied in Section 3.4.

3.2. *A simplified semantics*

We introduce a simplified interpretation of expressions of $\lambda_\Sigma^{\mathsf{gfp}}$ in terms of the coinductive syntax of λ_Σ^{co}. We adopt a simple and even possibly 'wrong' interpretation, which, however, for $\lambda_\Sigma^{\mathsf{gfp}}$-terms representing solution spaces will be seen to be equivalent. Notably, the simplified semantics dispenses with environments:

Definition 15 (Simplified interpretation of finitary forests as forests). For an expression T of $\lambda_\Sigma^{\mathsf{gfp}}$, the simplified interpretation $[\![T]\!]^s$ is a forest given by structural recursion on T:

$$[\![X^\sigma]\!]^s = \mathcal{S}(\sigma) \qquad [\![\lambda x^A.N]\!]^s = \lambda x^A.[\![N]\!]^s$$
$$[\![\mathsf{gfp}\ X^\sigma.\sum_i E_i]\!]^s = \sum_i [\![E_i]\!]^s \qquad [\![x\langle N_i\rangle_i]\!]^s = x\langle [\![N_i]\!]^s\rangle_i$$

Note that the base case profits from the sequent annotation at fixpoint variables, and the interpretation of the gfp-constructor has nothing to do with a greatest fixed point. Of course, this may be 'wrong' according to our understanding of a greatest fixed point.

Below, we will be specially interested in the finitary forests which guarantee that a $\mathsf{gfp}\,X^\sigma$ construction represents the solution space of σ.

Definition 16 (Proper expressions). An expression $T \in \lambda_\Sigma^{\mathsf{gfp}}$ is *proper* if for any of its subterms T' of the form $\mathsf{gfp}\ X^\sigma.\sum_i E_i$, it holds that $[\![T']\!]^s = \mathcal{S}(\sigma)$.

This means that an expression T is considered proper if, despite having used the simplified definition of semantics for the embedded fixed points, those subterms have the 'proper' semantics, and this is only expressed with respect to our main question of representing solution spaces, hence where for the fixed-point variables, the reference semantics of solution spaces is assumed.

For proper expressions, the simplified semantics agrees with the semantics we studied before. Of course, this can only make sense for expressions which have that previous semantics, in other words for well-bound and regular expressions.

Lemma 17. Let T be well-bound and ξ be an admissible environment for T such that for all $X^\sigma \in \mathrm{dom}(\xi)$: $\xi(X^\sigma) = \mathcal{S}(\sigma)$. If T is proper, then $[\![T]\!]_\xi = [\![T]\!]^s$.

We remark that for any regular T, there is exactly one such *minimal* environment ξ, based on comparing environments with respect to definedness.

Proof. By induction on expressions T. The variable case needs Lemma 7, lambda-abstraction and tuples are fine by the induction hypothesis. For the gfp case, it has to be shown that $[\![T]\!]^s$ fulfils the fixed-point equation defining $[\![T]\!]_\xi$, which suffices by uniqueness of the solution. The induction hypothesis can be applied to the elimination alternatives since the extended environment in which they have to be interpreted is of the required form, just by T being proper. □

Corollary 18. For well-bound, closed and proper T, $[\![T]\!] = [\![T]\!]^s$.

The corollary is sufficient for our purposes since $\mathcal{F}(\sigma)$ is not only well-bound and closed, but also proper, as will be stated now.

Theorem 19 (Equivalence for simplified semantics). Let σ be a sequent and Ξ as in Definition 8 so that $\mathcal{F}(\sigma;\Xi)$ exists (in particular, this holds for empty Ξ).

1. $\mathcal{F}(\sigma;\Xi)$ is proper.
2. $[\![\mathcal{F}(\sigma;\Xi)]\!]^s = \mathcal{S}(\sigma)$.

$$\frac{\mathsf{exfin}(N)}{\mathsf{exfin}(\lambda x^A.N)} \qquad \frac{\mathsf{exfin}(E_j)}{\mathsf{exfin}(\sum_i E_i)} \qquad \frac{\forall i,\ \mathsf{exfin}(N_i)}{\mathsf{exfin}(x\langle N_i\rangle_i)}$$

$$\frac{\mathsf{nofin}(N)}{\mathsf{nofin}(\lambda x^A.N)} \qquad \frac{\forall i,\ \mathsf{nofin}(E_i)}{\mathsf{nofin}(\sum_i E_i)} \qquad \frac{\mathsf{nofin}(N_j)}{\mathsf{nofin}(x\langle N_i\rangle_i)}$$

Fig. 5. exfin predicate and nofin predicate.

Proof. Both items together by structural induction on the term $\mathcal{F}(\sigma;\Xi)$. This all goes by unfolding the definitions and use of the induction hypothesis (the main case in the proof of 1 needs 2 for the subterms, so 1 cannot be proven separately before 2, and the main case of 2 immediately follows from the main case of 1, so it is better to prove both together, although 2 could be proven separately before 1). □

We remark that the proof is a simplification of the proof for Theorem 10 given previously (Espírito Santo et al. 2016). We also remark that the equivalence theorem for simplified semantics does not provide a compact description of solution spaces: applying it to the term for $\mathcal{F}(\Rightarrow A_2)$ found in Example 11 yields the much less telling

$$\mathcal{S}(\Rightarrow A_2) = \lambda x^{A_1}\lambda y^{A_0}\lambda z^p.x\langle\lambda w^p.\mathcal{S}(\sigma_2), z\rangle + y\langle z\rangle$$

that references $\mathcal{S}(\sigma_2)$ in the right-hand side.

3.3. *Deciding type emptiness*

We introduce predicate $\mathsf{nofin}(T)$, for T an expression of λ^{co}_{Σ} (a forest), which holds iff $\mathsf{nofinext}(T)$, i.e., if the finite extension of T is empty, but it is defined coinductively in Figure 5, together (but independently) with the inductive definition of the predicate $\mathsf{exfin}(T)$ that is supposed to mean the negation of $\mathsf{nofin}(T)$, but which is expressed positively as existence of a finite member (i.e., that the finite extension is non-empty – that $\mathsf{exfinext}(T)$ holds).

Lemma 20. Given a forest T, $\mathsf{exfin}(T)$ iff $\mathsf{nofin}(T)$ does not hold.

Proof. This is plainly an instance of the generic result in the style of De Morgan's laws that presents inductive predicates as complements of coinductive predicates, by a dualization operation on the underlying clauses. The principle is recalled with details now.

Assume a set U (the 'universe') and a function $F : \mathcal{P}(U) \to \mathcal{P}(U)$ that is monotone, i.e., for $\mathcal{M} \subseteq \mathcal{N} \subseteq U$, one has $F(\mathcal{M}) \subseteq F(\mathcal{N})$. Then, by Tarski's fixed-point theorem, there exist the least fixed point μF and the greatest fixed point νF of F, with respect to set inclusion. Moreover, μF is the intersection of all pre-fixed points $\mathcal{M} \subseteq U$ of F, i.e., with $F(\mathcal{M}) \subseteq \mathcal{M}$, and νF is the union of all post-fixed points $\mathcal{M} \subseteq U$ of F, i.e., with $\mathcal{M} \subseteq F(\mathcal{M})$. This lattice-theoretic duality allows to relate both concepts through complements, with $\mathcal{M}^{\complement} := U \setminus \mathcal{M}$. Given F as before, define a monotone function $F^{\dagger} : \mathcal{P}(U) \to \mathcal{P}(U)$ by setting $F^{\dagger}(\mathcal{M}) := (F(\mathcal{M}^{\complement}))^{\complement}$. Then,

$$\mu F = (\nu(F^{\dagger}))^{\complement}.$$

This formula (written in logical terms with negation in place of set complement) is often used to *define* μF, e. g., in μ-calculus. For a proof, it suffices to consider the inclusion from left to right (the other direction is obtained by duality, using $(F^\dagger)^\dagger = F$). Since the left-hand side is included in every pre-fixed point of F, it suffices to show that the right-hand side is such a pre-fixed point, i.e., $F((\nu(F^\dagger))^\complement) \subseteq (\nu(F^\dagger))^\complement$. We show the contrapositive $\nu(F^\dagger) \subseteq F^\dagger(\nu(F^\dagger))$ (using $F^\dagger$ as abbreviation): but $\nu(F^\dagger)$ is a post-fixed point itself (it is even a fixed point). □

The following lemma shows that the predicate nofin corresponds to the intended meaning in terms of the finite extension. Additionally, the lemma shows that the negation of nofin holds exactly for the forests which have finite members.

Lemma 21 (Coinductive characterization). Given a forest T. Then, $\mathsf{nofin}(T)$ iff $\mathcal{E}_{\mathrm{fin}}(T)$ is empty, i.e., $\mathsf{nofin} = \mathsf{nofinext}$ as sets of forests.

Proof. We give a coinductive argument in both directions. First, for a coterm M, let $\mathsf{inf}(M)$ be defined coinductively, as belonging to the greatest predicate inf satisfying

$$\mathsf{inf}(\lambda x^A.M) \Leftrightarrow \mathsf{inf}(M) \quad \text{and} \quad \mathsf{inf}(x\langle M_i\rangle_i) \Leftrightarrow \exists j,\, \mathsf{inf}(M_j).$$

This is a characterization of infinity: for a coterm M, M is a λ-term iff $\mathsf{inf}(M)$ does not hold. The direction from left to right is by induction on λ-terms, and for the other direction, the contrapositive is proven by coinduction on inf. Now, the statement of the lemma is equivalent to: $\mathsf{nofin}(T)$ iff $\mathsf{inf}(M)$ for all M s. t. $\mathsf{mem}(M, T)$. The 'only if' is equivalent to: if $\mathsf{nofin}(T)$ and $\mathsf{mem}(M, T)$, then $\mathsf{inf}(M)$. This is provable by coinduction on inf, using the obvious $\mathsf{mem}(M, M)$ for M in λ^{co}. The 'if' implication is suitable for coinduction on nofin, and this works smoothly. □

Thus, we are authorized to work with nofin and exfin in place of their 'extensional variants' $\mathsf{nofinext}$ and $\mathsf{exfinext}$.

Next, we turn to finitary representation of solution spaces and consider the predicate $\mathsf{EF}(T)$, for T an expression in $\lambda_\Sigma^{\mathsf{gfp}}$, which should hold when there is a finite solution. It is not obvious from the outset if free fixpoint variables should be considered as contributing to these finite solutions. If one already knows that $\mathsf{exfin}(\mathcal{S}(\sigma))$ holds, then it would be reasonable to put X^σ into the predicate EF. However, since our aim is to decide $\mathsf{exfin} \circ \mathcal{S}$ through decidability of EF, we cannot base rules for EF on a decision concerning $\mathsf{exfin} \circ \mathcal{S}$. Still, once we have a decision procedure for $\mathsf{exfin} \circ \mathcal{S}$, we can profit from a definition of EF that is sharp in the sense of containing variables X^σ by definition if and only if $\mathsf{exfin}(\mathcal{S}(\sigma))$. And this we will do in Section 3.4, building more complex predicates from EF.

We therefore consider a parameterized notion EF_P with P a predicate on sequents and instantiate it twice

— with $P := \emptyset$, the empty predicate which is trivially decidable, and,
— once $\mathsf{exfin} \circ \mathcal{S}$ is given a decision procedure, with $P := \mathsf{exfin} \circ \mathcal{S}$.

$$\frac{P(\sigma)}{\mathsf{EF}_P(X^\sigma)} \qquad \frac{\mathsf{EF}_P(N)}{\mathsf{EF}_P(\lambda x^A.N)} \qquad \frac{\mathsf{EF}_P(E_j)}{\mathsf{EF}_P(\mathsf{gfp}\, X^\sigma.\sum_i E_i)} \qquad \frac{\forall i,\ \mathsf{EF}_P(N_i)}{\mathsf{EF}_P(x\langle N_i\rangle_i)}$$

$$\frac{\neg P(\sigma)}{\mathsf{NEF}_P(X^\sigma)} \qquad \frac{\mathsf{NEF}_P(N)}{\mathsf{NEF}_P(\lambda x^A.N)} \qquad \frac{\forall i,\ \mathsf{NEF}_P(E_i)}{\mathsf{NEF}_P(\mathsf{gfp}\, X^\sigma.\sum_i E_i)} \qquad \frac{\mathsf{NEF}_P(N_j)}{\mathsf{NEF}_P(x\langle N_i\rangle_i)}$$

Fig. 6. EF_P and NEF_P predicates, for P satisfying the *proviso*: $P \subseteq \mathsf{exfin} \circ \mathcal{S}$ and P decidable.

The general proviso on P is decidability of P and that, for all sequents σ, $P(\sigma)$ implies $\mathsf{exfin}(\mathcal{S}(\sigma))$, i.e., $P \subseteq \mathsf{exfin} \circ \mathcal{S}$, when the predicates are seen as sets of sequents. This proviso is trivially satisfied in both instantiations.[§]

The definition of this (parameterized) predicate EF_P is inductive and presented in the first line of Figure 6, although it is clear that it could equivalently be given by a definition by recursion over the term structure. Therefore, the predicate EF_P is decidable, and we tacitly identify the inductive definition and the recursive procedure.

The negation of predicate EF_P is denoted by NEF_P. Its inductive characterization is easy, as all the rules of EF_P are 'invertible,' and is given in the second line of Figure 6.

Below, when EF_P or NEF_P is written, it is implicitly assumed that P satisfies the proviso of Figure 6.

Lemma 22. For all $T \in \lambda_\Sigma^{\mathsf{gfp}}$, $\mathsf{NEF}_P(T)$ iff $\mathsf{EF}_P(T)$ does not hold.

Proof. Routine induction on T. In terms of the equivalent recursive definitions of the predicates, this would have been just an application of De Morgan's laws. □

Lemma 23. Let $P \in \{\mathsf{exfin}, \mathsf{nofin}\}$ and $\sigma \leqslant \sigma'$. For all forests T, $P(T)$ iff $P([\sigma'/\sigma]T)$.

Proposition 2 (Finitary characterization).

1. If $\mathsf{EF}_P(T)$, then $\mathsf{exfin}([\![T]\!]^s)$.
2. Let $T \in \lambda_\Sigma^{\mathsf{gfp}}$ be well-bound and proper. If $\mathsf{NEF}_P(T)$ and for all $X^\sigma \in \mathrm{FPV}(T)$, $\mathsf{exfin}(\mathcal{S}(\sigma))$ implies $P(\sigma)$, then $\mathsf{nofin}([\![T]\!]^s)$.

Proof. Item 1: Proof by induction on the predicate EF_P (or, equivalently, on T). The base case for fixed-point variables needs the proviso on P, and all other cases are immediate by the induction hypothesis. Item 2 is proved by induction on the predicate NEF_P (or, equivalently, on T).

Case $T = X^\sigma$. Then, $\neg P(\sigma)$, hence, since $X^\sigma \in \mathrm{FPV}(T)$, by contraposition and Lemma 20, we get $\mathsf{nofin}(\mathcal{S}(\sigma))$.

Case $T = \mathsf{gfp}\, X^\sigma.\sum_i E_i$. Let $N := [\![T]\!]^s = \sum_i [\![E_i]\!]^s$. As T is proper, $N = \mathcal{S}(\sigma)$. We hence have to show $\mathsf{nofin}(\mathcal{S}(\sigma))$, which we do by an embedded coinduction for the coinductively defined predicate nofin. We have $\mathsf{NEF}_P(E_i)$ for all i and want to use the

[§] In a previous version of this paper, P was accidentally set to the always true predicate, in order to solve a problem of extensionality of a predicate that was used to deal with FINHAB. That was an error and led to incorrect proofs. We found this out by ourselves, but we also received a counterexample from Michał Ziobro in January 2017 which we gratefully acknowledge.

induction hypothesis, which would give us $\mathsf{nofin}([\![E_i]\!]^s)$ and thus $\mathsf{nofin}(\sum_i [\![E_i]\!]^s)$, which was our goal. Fix an i. Of course, E_i is also well-bound and proper. We have to consider all $Y^{\sigma'} \in \mathrm{FPV}(E_i)$. Either $Y^{\sigma'} \in \mathrm{FPV}(T)$, and we are fine by hypothesis, or $Y = X$ and, since T is well-bound, $\sigma \leqslant \sigma'$. We just show that $\mathsf{exfin}(\mathcal{S}(\sigma'))$ does not hold: from our coinductive hypothesis $\mathsf{nofin}(\mathcal{S}(\sigma))$, we get through Lemmas 7 and 23 even $\mathsf{nofin}(\mathcal{S}(\sigma'))$, and this is the negation of $\mathsf{exfin}(\mathcal{S}(\sigma'))$. This is a proper application of the coinductive hypothesis since it enters a lemma on nofin that does not change needed observation depths and then goes into an elimination alternative, where the occurrences of free fixpoint variables are at least 'guarded' by an ordinary variable of a tuple.

The other cases are simple applications of the induction hypothesis. □

Theorem 24 (Deciding the existence of inhabitants in λ).

1. For any $T \in \lambda_\Sigma^{\mathsf{gfp}}$ well-bound, proper and closed, $\mathsf{EF}_P(T)$ iff $\mathsf{exfin}([\![T]\!]^s)$.
2. $\mathsf{EF}_\emptyset(\mathcal{F}(\sigma))$ iff $\mathsf{exfin}(\mathcal{S}(\sigma))$ iff $\mathcal{I}(\sigma)$ is non-empty.
3. $\mathsf{exfin}(\mathcal{S}(\sigma))$ is decided by deciding $\mathsf{EF}_\emptyset(\mathcal{F}(\sigma))$; in other words, INHAB is decided by the computable predicate $\mathsf{EF}_\emptyset \circ \mathcal{F}$.

Proof. Item 1 follows from both the parts of Proposition 2, Lemmas 20 and 22, and the fact that, trivially, the extra condition in Proposition 2.2 is satisfied for closed terms. Item 2: The first equivalence follows from item 1 since by construction $\mathcal{F}(\sigma)$ is closed and well-bound, and additionally, by Theorem 19, $\mathcal{F}(\sigma)$ is proper and $[\![\mathcal{F}(\sigma)]\!]^s = \mathcal{S}(\sigma)$. For the equivalence between $\mathsf{exfin}(\mathcal{S}(\sigma))$ and non-emptiness of $\mathcal{I}(\sigma)$, first observe that non-emptiness of $\mathcal{I}(\sigma)$ is equivalent to $\mathsf{exfin}([\![\mathcal{F}(\sigma)]\!])$ by Lemma 13, the definition of exfinext, and Lemmas 20 and 21; then, observe that $\mathsf{exfin}(\mathcal{S}(\sigma))$ is the same as $\mathsf{exfin}([\![\mathcal{F}(\sigma)]\!])$, by the already shown $[\![\mathcal{F}(\sigma)]\!]^s = \mathcal{S}(\sigma)$ and Corollary 18.

Item 3: Use item 2, computability of $\mathcal{F}$ and the equivalence of the inductively defined $\mathsf{EF}_\emptyset$ with a recursive procedure over the term structure of its argument. □

Example 25. As expected, for our running examples,

— $\mathsf{NEF}_\emptyset(\mathcal{F}(\Rightarrow \mathsf{INFTY}))$ holds, because $\mathsf{NEF}_\emptyset(X^{f:p\supset p\Rightarrow p})$; hence, INFTY is empty;
— $\mathsf{EF}_\emptyset(\mathcal{F}(\Rightarrow \mathsf{CHURCH}))$ holds, because $\mathsf{EF}_\emptyset(\mathsf{gfp}\ X^\sigma.f\langle X^\sigma\rangle + x)$, due to the second alternative x; hence, CHURCH is non-empty.

If we disregard for a moment the ability of the present development to deal with the decontraction phenomenon, we can see a close resemblance with the emptiness checking for context-free grammars suggested by Hopcroft and Ullman (1979, Theorem 6.6). In fact, they refer to an algorithm in the proof of their Lemma 4.1 (p. 89) that we will recall in our own words. The set R of relevant non-terminals is computed. First, one puts all the non-terminals in R that have a production with only terminals on the right-hand side. Then, iteratively, all the non-terminals that have a production whose non-terminals in the right-hand side are all in R are added to R. After the process comes to a halt, check if the start symbol is in R.

Now to the algorithm provided by Theorem 24, in particular, the effect of deciding $\mathsf{EF}_\emptyset$, given that the solution space is already presented in form of $\mathcal{F}(\sigma)$: $\mathsf{EF}_\emptyset$ does not have a case for fixed-point variables, it goes under the terminal symbol '$\lambda x^A.$,' it branches

disjunctively over the elimination alternatives (corresponding to multiple productions), and it branches conjunctively over the list of arguments after having gone under the head variable (considered as terminal symbol). This is all what obviously has to be done, while going under a gfp-operator does not make a fixpoint variable more productive for the emptiness question. The latter observation corresponds to the Hopcroft and Ullman process of generating R that obviously does not get any help from recursive production rules: only if all the non-terminals in the right-hand side were in R, the left-hand side could be added to R.

The present proof is longer than the argument by Hopcroft and Ullman. This is because the specification and the algorithm are separated very carefully, invariants are embodied in separate definitions and also their negations are made explicit. And, more importantly, the chosen methodology is preparing the ground for the more challenging decision of finiteness. Finally, the decontraction phenomenon is being captured and does not dominate the course of the arguments.

After having decided inhabitation, we can instantiate the predicate P anew, serving as preparation for the decision of the question if there are finitely many inhabitants.

Definition 26. Let the decidable predicates $\mathsf{EF}_\star$ and $\mathsf{NEF}_\star$ on $\lambda_\Sigma^{\mathsf{gfp}}$ be defined by $\mathsf{EF}_\star := \mathsf{EF}_P$ and $\mathsf{NEF}_\star := \mathsf{NEF}_P$ for $P := \mathsf{EF}_\emptyset \circ \mathcal{F}$, which satisfies the proviso by Theorem 24.2.

Proposition 2.2 and Theorem 24.2 give that $\mathsf{NEF}_\star(T)$ implies $\mathsf{nofin}([\![T]\!]^s)$ for all well-bound and proper expressions T. However, an inspection of the proof of that proposition even shows that, for this particular case of NEF_P, the latter two properties of T are not needed:

Lemma 27 (Sharp finitary characterization). For all $T \in \lambda_\Sigma^{\mathsf{gfp}}$, $\mathsf{EF}_\star(T)$ iff $\mathsf{exfin}([\![T]\!]^s)$.

Proof. By the previous proposition, we only need to consider the direction from right to left, and we prove its contraposition $\mathsf{NEF}_\star(T)$ implies $\mathsf{nofin}([\![T]\!]^s)$ by induction on the predicate $\mathsf{NEF}_\star$.

Case $T = X^\sigma$. Then, $\neg\mathsf{exfin}(\mathcal{S}(\sigma))$ by hypothesis of this case and Theorem 24.2, and this is $\mathsf{nofin}([\![X^\sigma]\!]^s)$.

Case $T = \mathsf{gfp}\, X^\sigma.\sum_i E_i$. Then, $[\![T]\!]^s = \sum_i [\![E_i]\!]^s$. We have $\mathsf{NEF}(E_i)$ for all i and we use the induction hypothesis, which gives us $\mathsf{nofin}([\![E_i]\!]^s)$ for all i and thus $\mathsf{nofin}(\sum_i [\![E_i]\!]^s)$, which was our goal. Notice that this reasoning does not need further properties of T.

The other cases are likewise simple applications of the induction hypothesis. □

In particular, $\mathsf{exfin}([\![T]\!]^s)$ is decidable, by deciding $\mathsf{EF}_\star(T)$.

3.4. *Deciding type finiteness*

The development of this section will mostly mirror that of the preceding one. However, it builds on its results already in the definitions.

We characterize the predicate $\mathsf{finfinext}$ by an inductively defined predicate finfin. Generically, we obtain a characterization of its negation $\mathsf{inffinext}$ by the coinductively defined dual inffin of finfin. The inductive definition of finfin is given in the first line

$$\frac{\mathsf{nofin}(N)}{\mathsf{finfin}(\lambda x^A.N)} \qquad \frac{\mathsf{finfin}(N)}{\mathsf{finfin}(\lambda x^A.N)} \qquad \frac{\forall i,\ \mathsf{finfin}(E_i)}{\mathsf{finfin}(\sum_i E_i)} \qquad \frac{\mathsf{nofin}(N_j)}{\mathsf{finfin}(x\langle N_i\rangle_i)} \qquad \frac{\forall i,\ \mathsf{finfin}(N_i)}{\mathsf{finfin}(x\langle N_i\rangle_i)}$$

$$\frac{\mathsf{exfin}(N) \quad \mathsf{inffin}(N)}{\mathsf{inffin}(\lambda x^A.N)} \qquad \frac{\mathsf{inffin}(E_j)}{\mathsf{inffin}(\sum_i E_i)} \qquad \frac{\forall i,\ \mathsf{exfin}(N_i) \quad \mathsf{inffin}(N_j)}{\mathsf{inffin}(x\langle N_i\rangle_i)}$$

Fig. 7. finfin predicate and inffin predicate.

of Figure 7. Notice that, while finfin is inductively defined and has only finitely many premises in each clause, there is absolutely no claim on decidability since the coinductively defined predicate nofin enters the premises.

By inversion (decomposing the summands into tuples) on nofin, one can show that nofin $\subseteq$ finfin (which corresponds semantically to the trivial nofinext $\subseteq$ finfinext). Thus, in particular, no clause pertaining to nofin is necessary for the definition of $\mathsf{finfin}(\sum_i E_i)$. The predicate finfin is sound and complete in terms of membership:

Lemma 28 (Coinductive characterization). Given a forest T. Then, $\mathsf{finfin}(T)$ iff $\mathcal{E}_{\mathrm{fin}}(T)$ is finite, i.e., finfin = finfinext as sets of forests.

Proof. The direction from left to right (soundness) is immediate by induction on finfin, using Lemma 21. From right to left, we do induction on the sum of the term heights of all finite members, which is a finite measure. The first and fourth rule of finfin are necessary to capture the cases when one passes from λ-abstractions to their bodies resp. from tuples to their components – thus when the individual heights decrease – but when there is just no element whose height decreases. The case of sums of elimination alternatives needs a further decomposition into tuples, in order to be able to apply the inductive hypothesis. □

Combined with Lemma 21, this gives an alternative proof of nofin $\subseteq$ finfin.

The announced coinductive definition inffin that is meant to characterize inffinext is found in the second line of Figure 7.

Lemma 29. Given a forest T, $\mathsf{finfin}(T)$ iff $\mathsf{inffin}(T)$ does not hold.

Proof. inffin is defined from finfin by the De Morgan's law (as recalled in the proof of Lemma 20). In the first clause for inffin, the proviso $\mathsf{exfin}(N)$ is necessary for soundness, and as well the proviso $\mathsf{exfin}(N_j)$ (with $i = j$) in the last clause. Only through these guards, we can ensure that inffin $\subseteq$ exfin, which is a minimum requirement given what they say in terms of finite membership. Otherwise, the first clause would allow to derive $\mathsf{inffin}(N)$ for the infinite λ-abstraction, satisfying the equation $N = \lambda x^A.N$ for any choice of A and without any relevance of the variable x. Similarly, for the third clause with the coterm N satisfying $N = x\langle N\rangle$. □

As a corollary, we obtain inffin = inffinext as sets of forests.

Now we introduce two predicates on expressions of $\lambda_\Sigma^{\mathsf{gfp}}$ which will allow to characterize type finiteness, with the following intuitive meanings:

$$\frac{P(\sigma)}{\mathsf{FF}_P(X^\sigma)} \qquad \frac{\mathsf{FF}_P(N)}{\mathsf{FF}_P(\lambda x^A.N)} \qquad \frac{\forall i,\ \mathsf{FF}_P(E_i)}{\mathsf{FF}_P(\mathsf{gfp}\, X^\sigma.\sum_i E_i)} \qquad \frac{\forall i,\ \mathsf{FF}_P(N_i)}{\mathsf{FF}_P(x\langle N_i\rangle_i)} \qquad \frac{\mathsf{NEF}_\star(N_j)}{\mathsf{FF}_P(x\langle N_i\rangle_i)}$$

$$\frac{\neg P(\sigma)}{\mathsf{NFF}_P(X^\sigma)} \qquad \frac{\mathsf{NFF}_P(N)}{\mathsf{NFF}_P(\lambda x^A.N)} \qquad \frac{\mathsf{NFF}_P(E_j)}{\mathsf{NFF}_P(\mathsf{gfp}\, X^\sigma.\sum_i E_i)} \qquad \frac{\mathsf{NFF}_P(N_j) \quad \forall i,\ \mathsf{EF}_\star(N_i)}{\mathsf{NFF}_P(x\langle N_i\rangle_i)}$$

Fig. 8. FF_P and NFF_P predicates, for P satisfying the *proviso*: $P \subseteq \mathsf{finfin} \circ \mathcal{S}$ and P decidable.

1. $\mathsf{FF}_P(T)$: there are only finitely many finite members of T (the case of no finite members is included in this formulation).
2. $\mathsf{NFF}_P(T)$: there are infinitely many finite members of T.

Here, the predicate P on sequents controls the case of fixpoint variables, as before for EF_P and NEF_P. The general proviso on P is that it is decidable and that for all sequents σ, $P(\sigma)$ implies $\mathsf{finfin}(\mathcal{S}(\sigma))$, i. e., $P \subseteq \mathsf{finfin} \circ \mathcal{S}$. For our main result, it will be sufficient to take $P := \emptyset$. Another possibility – used in Section 4.3 – is choosing $P := \mathsf{NEF}_\emptyset \circ \mathcal{F}$, i. e., with the negation of the predicate underlying the definition of $\mathsf{EF}_\star$ and $\mathsf{NEF}_\star$. The definitions of these predicates are inductive, and they are presented in Figure 8. Analogously to the predicates EF_P and NEF_P, they could equivalently be defined recursively over the term structure, thus ensuring their decidability, thanks to the decision procedure for $\mathsf{EF}_\star$. Again, we do not formally distinguish between the inductive definition and the equivalent recursive procedure.

Lemma 30. Let $P := \mathsf{NEF}_\emptyset \circ \mathcal{F}$. Then, P qualifies as parameter for the just introduced predicates, and $\mathsf{NFF}_P \subseteq \mathsf{EF}_\star$ or, equivalently, $\mathsf{NEF}_\star \subseteq \mathsf{FF}_P$. This would allow to remove the condition $\mathsf{EF}_\star(N_j)$ from the tuple rule for NFF_P.

Proof. We use that $\mathsf{nofin} \subseteq \mathsf{finfin}$ and Theorem 24.2. The inclusion is verified by induction on the definition of the predicate that is proven to be included. □

Below, when FF_P or NFF_P is written, it is implicitly assumed that P satisfies the proviso of Figure 8.

Lemma 31. For all $T \in \lambda_\Sigma^{\mathsf{gfp}}$, $\mathsf{NFF}_P(T)$ iff $\mathsf{FF}_P(T)$ does not hold.

Proof. Routine induction on T, using Lemma 22. □

Lemma 32. Let $P \in \{\mathsf{finfin}, \mathsf{inffin}\}$ and $\Gamma \leqslant \Gamma'$. For all forests T, $P(T)$ iff $P([\Gamma'/\Gamma]T)$.

Proposition 3 (Finitary characterization).

1. If $\mathsf{FF}_P(T)$, then $\mathsf{finfin}([\![T]\!]^s)$.
2. Let $T \in \lambda_\Sigma^{\mathsf{gfp}}$ be well-bound and proper. If $\mathsf{NFF}_P(T)$ and for all $X^\sigma \in \mathrm{FPV}(T)$, $\mathsf{finfin}(\mathcal{S}(\sigma))$ implies $P(\sigma)$, then $\mathsf{inffin}([\![T]\!]^s)$.

Proof. Structurally similar to the proof of Proposition 2.

1. By induction on FF_P (or equivalently by structural induction on T). We only show the tuple cases with $T = x\langle N_i\rangle_i$. The other cases are equally simple.

Case for some j, $\mathsf{NEF}_\star(N_j)$. By Lemma 27, $\mathsf{nofin}([\![N_j]\!]^s)$, hence $\mathsf{finfin}(x\langle [\![N_i]\!]^s\rangle_i)$, which is $\mathsf{finfin}([\![T]\!]^s)$.

Case for all i, $\mathsf{FF}_P(N_i)$. By induction hypothesis, $\mathsf{finfin}([\![N_i]\!]^s)$ for all i, hence $\mathsf{finfin}([\![x\langle N_i\rangle_i]\!]^s)$.

2. By induction on FF_P (or equivalently by structural induction on T).

Case $T = X^\sigma$. Then, $\neg P(\sigma)$, hence, since $X^\sigma \in \mathrm{FPV}(T)$, by contraposition and Lemma 29, we get $\mathsf{inffin}(\mathcal{S}(\sigma))$.

Case $T = x\langle N_i\rangle_i$. For some j, $\mathsf{NFF}_P(N_j)$ and, for all i, $\mathsf{EF}_\star(N_i)$. The induction hypothesis is applicable for N_j since $\mathrm{FPV}(N_j) \subseteq \mathrm{FPV}(T)$. Therefore, we have $\mathsf{inffin}([\![N_j]\!]^s)$. By Lemma 27, $\mathsf{exfin}([\![N_i]\!]^s)$, for all i, hence, we are done by definition of inffin.

Case $T = \mathsf{gfp}\, X^\sigma.\sum_i E_i$. For some j, $\mathsf{NFF}_P(E_j)$. Let $N := [\![T]\!]^s = \sum_i [\![E_i]\!]^s$. As T is proper, $N = \mathcal{S}(\sigma)$. We hence have to show $\mathsf{inffin}(\mathcal{S}(\sigma))$, which we do by an embedded coinduction for the coinductively defined predicate inffin. We want to use the induction hypothesis for E_j, which would give us $\mathsf{inffin}([\![E_j]\!]^s)$ and thus $\mathsf{inffin}(\sum_i [\![E_i]\!]^s)$, which was our goal. Of course, E_j is also well-bound and proper. We have to consider all $Y^{\sigma'} \in \mathrm{FPV}(E_j)$. Either $Y^{\sigma'} \in \mathrm{FPV}(T)$, and we are fine by hypothesis, or $Y = X$ and, since T is well-bound, $\sigma \leqslant \sigma'$. We just show that $\mathsf{finfin}(\mathcal{S}(\sigma'))$ does not hold: from our coinductive hypothesis $\mathsf{inffin}(\mathcal{S}(\sigma))$, we get through Lemmas 7 and 32 even $\mathsf{inffin}(\mathcal{S}(\sigma'))$, and this is the negation of $\mathsf{finfin}(\mathcal{S}(\sigma'))$. This is a proper application of the coinductive hypothesis since it enters a lemma on inffin that does not change needed observation depths and then goes into an elimination alternative, where the occurrences of free fixpoint variables are at least 'guarded' by an ordinary variable of a tuple.

The case of λ-abstractions is a simple application of the induction hypothesis. □

We remark that the proposition and its proof are rather analogous to Proposition 2 than dual to it, although the logical structure of the predicates is rather dual: to enter a fixed point into FF_P, all of the elimination alternatives have to be there already, while for EF_P, only one of the elimination alternatives is required. However, this duality is broken for the tuples: while for EF_P, all arguments are required to be in the same predicate, FF_P has a rule that asks only about one argument, but for a different predicate, and there is even a second possibility. Anyway, the proof structure needs to be analogous since exfin and finfin are both inductively defined and therefore do not admit reasoning by coinduction.

Now the problem FINHAB can be solved in the same way as INHAB.

Theorem 33 (Deciding type finiteness in λ).

1. For any $T \in \lambda_\Sigma^{\mathsf{gfp}}$ well-bound, proper and closed, $\mathsf{FF}_P(T)$ iff $\mathsf{finfin}([\![T]\!]^s)$.
2. $\mathsf{FF}_\emptyset(\mathcal{F}(\sigma))$ iff $\mathsf{finfin}(\mathcal{S}(\sigma))$ iff $\mathcal{I}(\sigma)$ is finite.
3. $\mathsf{finfin}(\mathcal{S}(\sigma))$ is decided by deciding $\mathsf{FF}_\emptyset(\mathcal{F}(\sigma))$; in other words, FINHAB is decided by the computable predicate $\mathsf{FF}_\emptyset \circ \mathcal{F}$.

Proof. Follows the structure of the proof of Theorem 24.

1. Follows from both parts of Proposition 3, Lemmas 29 and 31, and the fact that, trivially, the extra condition in Proposition 3.2 is satisfied for closed terms.

2. For the first equivalence, apply 1. with both parts of Theorem 19. For the equivalence between $\mathsf{FF}_\emptyset(\mathcal{F}(\sigma))$ and finiteness of $\mathcal{I}(\sigma)$, apply 1. with Corollary 18 and Lemmas 28 and 14.

3. Use 2, computability of $\mathcal{F}$ and the equivalence of the inductively defined $\mathsf{FF}_\emptyset$ with a recursive procedure over the term structure of its argument, where, corresponding to the last rule of Figure 8, the decisions for predicate $\mathsf{NEF}_\star$ are invoked. □

Example 34. For our running examples, we have

— $\mathsf{FF}_\emptyset(\mathcal{F}(\mathsf{INFTY}))$ holds, because $\mathsf{FF}_\emptyset(f\langle X^{f:p\supset p\Rightarrow p}\rangle)$, due to $\mathsf{NEF}_\star(X^{f:p\supset p\Rightarrow p})$ (note that the fact $\mathsf{NEF}_\emptyset(\mathcal{F}(f : p \supset p \Rightarrow p))$ can be seen from the steps behind Example 25); hence, INFTY is finite;

— $\mathsf{NFF}_\emptyset(\mathcal{F}(\mathsf{CHURCH}))$ holds, because $\mathsf{NFF}_\emptyset(f\langle X^\sigma\rangle)$ holds, for which we need both $\mathsf{NFF}_\emptyset(X^\sigma)$ and $\mathsf{EF}_\star(X^\sigma)$ (again $\mathsf{EF}_\emptyset(\mathcal{F}(\sigma))$ can be seen from the steps behind Example 25; recall $\sigma := f : p \supset p, x : p \Rightarrow p$); hence, CHURCH is infinite.

If, as for emptiness checking, we disregard the ability of the described method to deal with decontraction, we can see similarities with finiteness checking for context-free grammars (Hopcroft and Ullman 1979, Theorem 6.6). Additionally, Hopcroft and Ullman mention that decisions based on the pumping lemma are 'highly inefficient,' but those were alluded to in the approach by Takahashi et al. to inhabitation based on grammars (Takahashi et al. 1996, Corollary 3.8). The algorithm described by Hopcroft and Ullman does not work on a given context-free grammar but on an equivalent one that is in Chomsky normal form and has no *useless* symbols (Hopcroft and Ullman 1979, p.88); since there is no empty word in λ-calculus, we slightly simplified the presentation. In the Chomsky normal form, without useless symbols, effective dependency of a non-terminal is identified by looking at the non-terminals that appear in the right-hand side of its productions. Those dependencies constitute a directed graph, and finiteness of the grammar is *equivalent* to the absence of cycles in the dependency graph, thanks to the absence of useless symbols.

The algorithm of Theorem 33 executes $\mathsf{FF}_\emptyset$ on $\mathcal{F}(\sigma)$. Basically, the algorithm for confirming finiteness follows the term structure and has to avoid 'hitting' a fixed-point variable (since $P = \emptyset$). So even the sums of elimination alternatives are treated conjunctively, as are the arguments in a tuple in the fourth rule of Figure 8. This is all naturally mapped to the Hopcroft and Ullman description. However, the 'grammar' has not been shrunk to one without useless symbols, since the algorithm works on the original data structure obtained in form of the finitary representation of the solution space. Hitting on a fixed-point variable does not mean the existence of infinitely many inhabitants if it corresponds to a useless symbol, and this is taken care of by the fifth rule of Figure 8 that allows any N_i, $i \neq j$ if only N_j cannot contribute any inhabitant. The latter is the high-level description; the decision algorithm runs a modification of the algorithm for deciding emptiness, with a non-trivial choice of the parameter P of NEF_P, so that fixed-point variables of the whole finitary forest are handled according to the needs of this subproblem (which, once again, is ruled out in Hopcroft and Ullman's algorithm by the assumption on useless symbols).

As for the treatment of INHAB, we claim separation of specification and implementation, explicit invariants, and no undue overhead for dealing with the fact that we are not faced with context-free grammars but work with the original structure of λ-terms with possible decontraction.

The analysis we did with $\mathsf{EF}_\star$ and $\mathsf{NEF}_\star$ can be replayed. In this paper, we do not exploit that possibility. It is shown here for completeness.

Definition 35. Let the predicates $\mathsf{FF}_\star$ and $\mathsf{NFF}_\star$ on $\lambda_\Sigma^{\mathsf{gfp}}$ be defined by $\mathsf{FF}_\star := \mathsf{FF}_P$ and $\mathsf{NFF}_\star := \mathsf{NFF}_P$ for $P := \mathsf{finfin} \circ \mathcal{S}$, which satisfies the proviso by Theorem 33.3. In particular, $\mathsf{FF}_\star$ and $\mathsf{NFF}_\star$ are decidable.

Proposition 3.2 gives that $\mathsf{NFF}_\star(T)$ implies $\mathsf{inffin}([\![T]\!]^s)$ for all well-bound and proper expressions T. Again (as for Lemma 27), an inspection of the proof of that proposition even shows that the latter two properties of T are not needed:

Lemma 36 (Sharp finitary characterization). For all $T \in \lambda_\Sigma^{\mathsf{gfp}}$, $\mathsf{FF}_\star(T)$ iff $\mathsf{finfin}([\![T]\!]^s)$.

In particular, $\mathsf{finfin}([\![T]\!]^s)$ is decidable, by deciding $\mathsf{FF}_\star(T)$.

4. On the number of inhabitants

The method of the preceding section is not confined to the mere decision problems. In particular, instead of only deciding FINHAB, the finitely many inhabitants can be effectively obtained. We will illustrate this for the somehow more basic question of determining their number.

4.1. *Head-variable controlled expressions*

We have considered finitary forests throughout the paper modulo idempotence of the summation operation (among other identifications). This is not an appropriate data structure if finite members of its semantics are to be counted. A way out is to consider the finite sums as functions from a finite set of (head) variables x into finite tuples $\langle N_{x,i}\rangle_i$ of terms (among the finitary forests) headed by x. Then, the summand for x is $x\langle N_{x,i}\rangle_i$, and the whole sum is over the finitely many chosen head variables. This disallows different summands with the same head variable, but we remark that $\mathcal{F}(\sigma)$ obeys to this restriction that we call being *head-variable controlled*. Let those finitary expressions form the set $\mathsf{H}\lambda_\Sigma^{\mathsf{gfp}}$ and call them head-variable controlled finitary forests. Analogously, form the set $\mathsf{H}\lambda_\Sigma^{co}$ of head-variable controlled forests and remark that $\mathcal{S}(\sigma)$ falls into that restricted class.

Most readers might want to skip the following more detailed explanations and immediately proceed to the next section (4.2).

In the following, we try to explain in more detail how to understand the head-variable controlled elements: We have also considered forests throughout the paper modulo idempotence of the summation operation (among other identifications). This does not hinder us from counting the number of finite members in case it is finite. The finite members themselves are 'concrete,' and the only identification that is not expressed in the

grammar of λ is α-equivalence. However, we would prefer counting summandwise and thus need to be sure that finite members do not belong to more than one summand in a sum, and this by taking into account that occurrences are identified up to bisimulation. Technically, this desideratum is achieved by considering a subset of forests that we call *head-variable controlled*. The set $\mathsf{H}\lambda_\Sigma^{co}$ of head-variable controlled forests is obtained by the same grammar of terms and elimination alternatives as λ_Σ^{co}, but with the restriction for the formation of $\sum_i E_i$ with $E_i = x_i\langle N_j^i\rangle_j$ that the x_i are pairwise different, i.e., no variable is head of two summands in one sum, and this recursively throughout the forest. If we consider this restriction in our view of sums as sets of elimination alternatives, this only means that a given head variable cannot appear with two distinct tuples of arguments but still can appear multiply. So, in order to profit from the extra property of forests in $\mathsf{H}\lambda_\Sigma^{co}$, we regard sums as functions from a finite set of (head) variables x into finite tuples of forest headed by x and use the associated notion of bisimilarity (modulo α-equivalence). This means, when we speak about head-variable controlled forests, we not only consider forests satisfying this extra property, but also their presentation in this form that takes profit from it. This change of view does not change the notion of bisimilarity.

Analogously, we introduce the set $\mathsf{H}\lambda_\Sigma^{\mathsf{gfp}}$ of head-variable controlled elements of $\lambda_\Sigma^{\mathsf{gfp}}$. Again, this is not only a subset but comes with a different presentation of sums as functions from a finite set of (head) variables x into finite tuples of terms (among the finitary forests) headed by x.

Notice that $\mathcal{S}(\sigma)$ and $\mathcal{F}(\sigma)$ always yield head-variable controlled terms, in the respective term systems.

4.2. *Counting inhabitants*

We define the counting function # for head-variable controlled forests in finfin only, by recursion on finfin. Notice that the well-definedness of the following function is addressed in detail in Lemma 39 below.

Definition 37 (Infinitary counting function $\# : \mathsf{H}\lambda_\Sigma^{co} \cap \mathsf{finfin} \to \mathbb{N}$).

$$\begin{aligned}
\#(\lambda x^A.N) &:= \begin{cases} 0 & \text{if } \mathsf{nofin}(N) \\ \#(N) & \text{else} \end{cases} \\
\#(\textstyle\sum_i E_i) &:= \textstyle\sum_i \#(E_i) \\
\#(x\langle N_i\rangle_i) &:= \begin{cases} 0 & \text{if } \exists j,\ \mathsf{nofin}(N_j) \\ \prod_i \#(N_i) & \text{else} \end{cases}
\end{aligned}$$

Note that, as usual, summing in the integers over no elements yields 0 and the corresponding product is 1, hence $\#(x\langle\rangle) = 1$.

Lemma 38. Let $T \in \mathsf{H}\lambda_\Sigma^{co}$. If $\mathsf{nofin}(T)$ (in particular, $\mathsf{finfin}(T)$), then $\#(T) = 0$.

Proof. Neither induction on T nor on nofin are available. The proof is by case analysis, where one has to use that elimination alternatives are tuples. □

The following lemma can be considered a refinement of the soundness part of Lemma 28.

Lemma 39. Let T be a head-variable controlled forest such that $\mathsf{finfin}(T)$. Then, $\#(T)$ is a well-defined natural number, and it is the cardinality of $\mathcal{E}_{\mathrm{fin}}(T)$.

Proof. Since sums are identified throughout the paper modulo idempotence, the rule for sums would not make sense if T was not head-variable controlled. With this restriction, function # becomes compatible with our identifications.

Intuitively, the recursive calls to # occur only with forests that enter finfin 'earlier.' We are thus heading for a recursive definition of # that is over the inductive structure of the supposed proofs of $\mathsf{finfin}(T)$ for the allowed arguments T. Strictly speaking, predicate finfin does not look suitable for such a recursive definition since its inductive definition is not deterministic: there are two rules that allow to infer that lambda-abstractions enter finfin, and their hypotheses can be simultaneously satisfied. The same holds of tuples. The inductive definition can be turned into a deterministic one rather trivially: just restrict the second and fifth rule in Figure 7 to the cases where the first and fourth rule do not apply, i.e., thanks to Lemma 20, they become

$$\frac{\mathsf{exfin}(N) \quad \mathsf{finfin}(N)}{\mathsf{finfin}(\lambda x^A.N)} \qquad \frac{\forall i,\, \mathsf{exfin}(N_i) \quad \forall i,\, \mathsf{finfin}(N_i)}{\mathsf{finfin}(x\langle N_i\rangle_i)}$$

It is with respect to this restriction of allowed derivations that # becomes a plainly structurally recursive function. The 'official' definition of finfin is not given this way to avoid clutter. Moreover, for the sake of well-definedness of #, it is not even necessary to apply this modification: the 'else' branches explicitly disallow that the first and fourth rule of Figure 7 have been applied in the last derivation step, so the recursive calls are indeed with forests that come from the properly inductive clauses.

Being the correct number in the sense that $\#(T)$ is the cardinality of $\mathcal{E}_{\mathrm{fin}}(T)$ is seen by induction on $\mathsf{finfin}(T)$, using Lemma 12 throughout. It also makes use of Lemma 21 for the 'if' cases of the definition, and the clause for sums of elimination alternatives is subject to the presentation we convened for elements of $\mathsf{H}\lambda^{co}_{\Sigma}$ which ensures that the elements of a disjoint union are counted. □

Remark: The proof of Lemma 38 only exploits the given defining equations, hence does not depend on the question of well-definedness of the previous lemma. It shows in particular that $\#(\lambda x^A.N) = \#(N)$ whenever N is in the domain of #. This does not entitle us to remove the case distinction in the λ-abstraction case of the definition of #: the infinite λ-abstraction used in the proof of Lemma 29 would not be associated with a unique value then (thus, # would no longer be well-defined). For tuples, even claiming that, generally, $\#(x\langle N_i\rangle_i) = \prod_i \#(N_i)$ would require a very non-strict reading of the product that would have to be defined and be of value 0 as soon as one of the factors is 0.

While the infinitary counting function will rather serve as a tool for verification, we now come to the definition of the counting function of interest for our applications.

Definition 40 (Finitary counting function $\# : \mathsf{H}\lambda_\Sigma^{\mathsf{gfp}} \to \mathbb{N}$). Define by recursion over the term structure

$$\begin{aligned} \#(X^\sigma) &:= 0 \\ \#(\lambda x^A.N) &:= \#(N) \\ \#(\mathsf{gfp}\, X^\sigma.\textstyle\sum_i E_i) &:= \textstyle\sum_i \#(E_i) \\ \#(x\langle N_i\rangle_i) &:= \textstyle\prod_i \#(N_i) \end{aligned}$$

Lemma 41. Let $T \in \mathsf{H}\lambda_\Sigma^{\mathsf{gfp}} \cap \mathsf{NEF}_\star$. Then, $\#(T) = 0$.

Proof. Induction over $\mathsf{NEF}_\star$ (or, equivalently, over T).

Case $T = X^\sigma$. Trivial.

Case $T = \lambda x^A.N$. Trivial by induction hypothesis.

Case $T = x\langle N_i\rangle_i$. By induction hypothesis, one of the factors is 0.

Case $T = \mathsf{gfp}\, X^\sigma.\sum_i E_i$. By induction hypothesis, all summands are 0. □

Proposition 4. Let $P \subseteq \mathsf{nofin} \circ \mathcal{S}$ and $T \in \mathsf{H}\lambda_\Sigma^{\mathsf{gfp}} \cap \mathsf{FF}_P$. Then, $\#(T) = \#([\![T]\!]^s)$.

Proof. We will write L and R for left-hand side and right-hand side of the equation to prove. Since $\mathsf{nofin} \subseteq \mathsf{finfin}$, P qualifies as parameter in FF_P. By Proposition 3.1, $\mathsf{finfin}([\![T]\!]^s)$, hence R is well-defined. The proof of $L = R$ is by induction on T (or, equivalently, by induction on FF_P).

Case $T = X^\sigma$. Then, $\mathsf{nofin}(\mathcal{S}(\sigma))$, hence $\#(\mathcal{S}(\sigma)) = 0$ by Lemma 38. Hence, $R = 0 = L$.

Case $T = \lambda x^A.N$. Then, $\mathsf{FF}_P(N)$. $L = \#(N)$. $R = \#(\lambda x^A.[\![N]\!]^s)$. According to the definition of R, we have to distinguish if $\mathsf{nofin}([\![N]\!]^s)$ or not. In the first case, by Lemma 38, we have $\#([\![N]\!]^s) = 0$. Thus, in both case, this gives $R = \#([\![N]\!]^s)$, while $L = \#(N)$. Done by induction hypothesis.

Case $T = x\langle N_i\rangle_i$. Subcase $\mathsf{NEF}_\star(N_j)$ for some j. By Lemma 27, $\mathsf{nofin}([\![N_j]\!]^s)$. Hence, $R = 0$. By Lemma 41, $\#(N_j) = 0$, hence also $L = 0$ (since one factor is 0).

Subcase $\mathsf{FF}_P(N_i)$ for all i. We may assume that we are not in the first subcase that has already been treated, hence $\mathsf{EF}_\star(N_i)$ for all i. By Lemma 27, $\neg\mathsf{nofin}([\![N_i]\!]^s)$ for all i. Therefore, $R = \prod_i \#([\![N_i]\!]^s)$, while $L = \prod_i \#(N_i)$. Done by induction hypothesis for all i.

Case $T = \mathsf{gfp}\, X^\sigma.\sum_i E_i$. Then, $\mathsf{FF}_P(E_i)$ for all i. Just apply the induction hypothesis to all the summands and sum up. (Notice how this case becomes the simplest one in our setting with simplified semantics.) □

Theorem 42 (Counting theorem). Let $P \subseteq \mathsf{nofin} \circ \mathcal{S}$ (e. g., $P = \emptyset$). If $\mathsf{FF}_P(\mathcal{F}(\sigma))$, then $\#(\mathcal{F}(\sigma))$ is the cardinality of $\mathcal{I}(\sigma)$.

Proof. $\mathcal{F}(\sigma) \in \mathsf{H}\lambda_\Sigma^{\mathsf{gfp}}$. By the preceding proposition, using the assumption that $\mathsf{FF}_P(\mathcal{F}(\sigma))$, we obtain $\#(\mathcal{F}(\sigma)) = \#([\![\mathcal{F}(\sigma)]\!]^s)$, which is $\#(\mathcal{S}(\sigma))$ by Theorem 19. Thanks to Proposition 3.1, $\mathsf{finfin}(\mathcal{S}(\sigma))$, hence, by Lemma 39, $\#(\mathcal{S}(\sigma))$ is the cardinality of $\mathcal{E}_{\mathrm{fin}}(\mathcal{S}(\sigma))$. □

Notice that when $\mathsf{FF}_P(\mathcal{F}(\sigma))$ does not hold, then $\#(\mathcal{F}(\sigma))$ is meaningless, but $\mathsf{NFF}_P(\mathcal{F}(\sigma))$ holds, and thus, $\mathsf{inffin}(\mathcal{S}(\sigma))$, which ensures that $\mathcal{I}(\sigma)$ is infinite.

Remark on Theorem 42: Without any extra effort, we can give an effective definition of the associated set of finite inhabitants through a function $\mathcal{C} : \mathsf{H}\lambda_{\Sigma}^{\mathsf{gfp}} \to \mathcal{P}_{\mathsf{fin}}(\lambda)$ by

$$\begin{aligned} \mathcal{C}(X^{\sigma}) &:= \emptyset \\ \mathcal{C}(\lambda x^{A}.N) &:= \{\lambda x^{A}.t \mid t \in \mathcal{C}(N)\} \\ \mathcal{C}(\mathsf{gfp}\, X^{\sigma}.\textstyle\sum_i E_i) &:= \cup_i\, \mathcal{C}(E_i) \\ \mathcal{C}(x\langle N_i\rangle_i) &:= \{x\langle t_i\rangle_i \mid \forall i,\, t_i \in \mathcal{C}(N_i)\} \end{aligned}$$

Then, for $T \in \mathsf{H}\lambda_{\Sigma}^{\mathsf{gfp}}$, $\#(T)$ is the cardinality of $\mathcal{C}(T)$ (notice that the set union in the gfp case is always a disjoint union), and if $\mathsf{FF}_{\emptyset}(\mathcal{F}(\sigma))$, then $\mathcal{I}(\sigma) = \mathcal{C}(\mathcal{F}(\sigma))$. If not, $\mathcal{I}(\sigma)$ is infinite.

The counting functions may look simple, but they count 'properly' and are fully integrated with our framework that establishes the decidability of the inhabitation problems we addressed. By counting 'properly,' we understand that they identify α-equivalent inhabitants (we do this implicitly in the whole paper), but do not apply the 'total discharge convention' that identifies variables of the same type. For example, in previous work on grammar presentations for β-normal λ-terms (Takahashi et al. 1996), this convention is applied to obtain a finite context-free grammar. If the question is just emptiness or finiteness, the difference is of no importance, but already the question if there is at most one inhabitant obviously depends on it. For example, the type $p \supset p \supset p$ has only one inhabitant with the total discharge convention while it has two in λ.

4.3. *Predicting the number of inhabitants*

The identification of syntatic criteria on types telling about their number of inhabitants has received a lot of attention for a long time, specially in the case of uniqueness of inhabitants, due to its intimate relation to coherence theorems in category theory (Mints 1979). For example, consider the type $A = (p^{+} \supset q^{+} \supset r^{-}) \supset (p^{+} \supset q^{-}) \supset p^{-} \supset r^{+}$ of the S-combinator, with atoms p, q and r, where we have marked positive and negative occurrences by the superscripts $+$ and $-$ (positive and negative occurrences in a formula are defined as usual, with change of *polarity* when moving to the left argument of $\supset$). Now, only by observing that no atom occurs negatively more than once in A, it is possible to tell immediately that A has at most one ($\beta\eta$-normal) inhabitant.

In this section, we show the tools we developed at work in this kind of questions, and we give a new and perspicuous proof of an old theorem, in a rather recent more general form.

We consider the coherence theorem for balanced types. Those are types where no atom occurs positively more than once or negatively more than once. Put differently, any atom in a balanced formula has at most two occurrences, and if there are two, they have opposite polarities. Using proof-theoretic methods, Mints (1992) proved that balanced types have at most one $\beta\eta$-normal inhabitant, extracting from this a coherence theorem for cartesian closed categories. By a close inspection of Mints' proof, Aoto and Ono (1994) generalized the result to the so-called *negatively non-duplicated types*, i.e., types where no atom occurs negatively more than once. (For example, uniqueness of inhabitation of

the type A of the S-combinator above does not follow from the coherence theorem for balanced formulas – since A has two positive occurrences of atom p, but follows from this generalization.) Bourreau and Salvati (2011) reproved this result through a game semantics approach. Using the proof-tree method, Broda and Damas (2005) were able to reprove the uniqueness result of Aoto and Ono, and extend it to the class of *deterministic types*, a class defined by means of the proof-tree generated by a type. In the same work, Broda and Damas also established that *positively non-duplicated types* have finitely many inhabitants.

We now show how to obtain the state-of-the-art results with our tools.

Definition 43. Let $T \in \lambda_{\Sigma}^{\mathsf{gfp}}$.

1. T is *strongly acyclic* if T has no occurrence, free or bound, of fixed-point variables (other than the binding occurrences after gfp).
2. T is *deterministic* if every sum in T has at most one summand.

Lemma 44. Let $T \in \lambda_{\Sigma}^{\mathsf{gfp}}$.

1. If T is strongly acyclic, then $\mathsf{FF}_{\emptyset}(T)$.
2. If T is deterministic, then $\#(T) \leqslant 1$ and, in addition, $\mathsf{EF}_{\emptyset}(T)$ implies $\mathsf{FF}_{\emptyset}(T)$.

Proof. In each case by a straightforward induction on T. □

Let $A = \vec{A} \supset p$. We say p is the *target* atom of A. The occurrence of p that makes p the target atom of A is called its *tail* occurrence in A. Let $\sigma = (\Gamma \Rightarrow A)$, with $\Gamma = \{x_1 : C_1, \cdots, x_n : C_n\}$. Put $A_\sigma := \vec{C} \supset \vec{A} \supset p$ (the order of the C_i's does not matter). In particular, if σ is $\Rightarrow A$, then $A_\sigma = A$.

Lemma 45.

1. If no atom occurs positively more than once in A, then $\mathcal{F}(\Rightarrow A)$ is strongly acyclic.
2. If no atom occurs negatively more than once in A and $\mathsf{EF}_{\emptyset}(\mathcal{F}(\Rightarrow A))$, then $\mathcal{F}(\Rightarrow A)$ is deterministic.

Proof. Recall Definition 8 of $\mathcal{F}(\sigma;\Xi)$. Some extra, auxiliary definitions are needed. We say an atom p is a *target* atom of Ξ if $p = q_i$, for some $(X_i : \Theta_i \Rightarrow q_i) \in \Xi$. Let $A\backslash\Xi$ be the formula resulting from $A = \vec{A} \supset p$ by erasing in $\vec{A}$ each A_i whose target atom is a target atom in Ξ.

We generalize item 1 and prove: if $\mathcal{F}(\sigma;\Xi)$ is defined and $P(\sigma;\Xi)$, then $\mathcal{F}(\sigma;\Xi)$ is strongly acyclic, with the predicate P (the invariant for the 'positive' case) defined as follows:

Predicate $P(\sigma;\Xi)$ holds if, for every atom p, p does not occur positively in $A_\sigma\backslash\Xi$ more than once; and if p does occur positively in $A_\sigma\backslash\Xi$, then p is not a target atom of Ξ.

Notice that, if $\sigma = (\Rightarrow A)$ and $\Xi = \cdot$, then $P(\sigma;\Xi)$ iff no atom occurs positively more than once in A. Thus, the implication we are proving is a proper generalization of item 1.

The proof is by induction on $\mathcal{F}(\sigma;\Xi)$. Suppose $P(\sigma;\Xi)$. Let $\sigma := \Gamma \Rightarrow \vec{A} \supset p$ and $\Delta := \Gamma \cup \{z_1 : A_1, \cdots, z_n : A_n\}$ and $\sigma' := \Delta \Rightarrow p$.

Since p occurs positively in $A_\sigma \backslash \Xi$, p is not a target atom of Ξ, hence the if-guard in the definition of $\mathcal{F}(\sigma;\Xi)$ does not hold. Then, $\mathcal{F}(\sigma;\Xi)$ is given by

$$\lambda z_1^{A_1} \cdots z_n^{A_n}.\mathsf{gfp}\ Y^{\sigma'}.\ \sum_{(y:\vec{B}\supset p)\in\Delta} y\langle \mathcal{F}(\Delta \Rightarrow B_j;\Xi, Y : \sigma')\rangle_j \qquad (*)$$

In order to show that $\mathcal{F}(\sigma;\Xi)$ is strongly acyclic, it suffices to show that, for each y in the sum and each j, $\mathcal{F}(\sigma_j;\Xi')$ is strongly acyclic, where we let $\sigma_j := \Delta \Rightarrow B_j$ and $\Xi' := \Xi, Y : \sigma'$. We need some work before we embark on the proof.

Observe that $A_\sigma = A_{\sigma'} = \vec{C} \supset \vec{A} \supset p$, for some $\vec{C}$; moreover, $A_{\sigma_j} = \vec{C} \supset \vec{A} \supset B_j$, with B_j a component of $\vec{B}$; and $\vec{B} \supset p$ is a component of $\vec{C}$ or $\vec{A}$ that is erased in $A_{\sigma_j}\backslash\Xi'$ by the extra target atom p of Ξ'.

Each atom occurrence in A_{σ_j} has a corresponding atom occurrence in A_σ. Due to the duplication of B_j in A_{σ_j}, there may be two occurrences in A_{σ_j} that correspond to the same occurrence in A_σ, and the tail occurrence of p in A_σ corresponds to no occurrence. Now we need two observations: (i) the new copy of B_j preserves the polarities of atom occurrences; (ii) the old copy of B_j is erased in $A_{\sigma_j}\backslash\Xi'$. Hence, *when starting from atom occurrences in* $A_{\sigma_j}\backslash\Xi'$ *only*, there is a corresponding atom occurrence in $A_\sigma\backslash\Xi$; moreover, this restricted correspondence is injective and polarity preserving. So, using that p is not a target atom of Ξ, we proved:

If an atom has two occurrences in $A_{\sigma_j}\backslash\Xi'$ with some polarity, so it does in $A_\sigma\backslash\Xi$. (**)

Fact (**), together with $P(\sigma;\Xi)$, yields that no atom occurs positively more than once in $A_{\sigma_j}\backslash\Xi'$.

Now we argue that, if an atom q has a positive occurrence in $A_{\sigma_j}\backslash\Xi'$, then it is not a target atom of Ξ'. Suppose atom q has a positive occurrence in $A_{\sigma_j}\backslash\Xi'$. We want to prove that q is neither p nor a target atom of Ξ. First, q cannot be p (otherwise p would have two positive occurrences in $A_\sigma\backslash\Xi$: the corresponding one and the tail occurrence). Second, given that q has a positive occurrence in $A_\sigma\backslash\Xi$, q is not a target atom of Ξ, due to $P(\sigma;\Xi)$.

So, we secured $P(\sigma_j;\Xi')$. By induction hypothesis, $\mathcal{F}(\sigma_j;\Xi')$ is strongly acyclic.

Item 2. More auxiliary definitions are needed.

Predicate $N(\sigma;\Xi)$ holds if, for every atom p, p does not occur negatively in $A_\sigma\backslash\Xi$ more than once.

Notice that, if $\sigma = (\Rightarrow A)$ and $\Xi = \cdot$, then $N(\sigma;\Xi)$ iff no atom occurs negatively more than once in A. Notice also that this predicate N is not the dual of predicate P, used for item 1.

Predicate $S(\sigma)$ holds if, given $A_\sigma = \vec{C} \supset p$, for every atom q (including p): (i) q has at most one negative occurrence (relatively to A_σ) in each C_i, and (ii) if $\vec{D} \supset q$ and $\vec{D}' \supset q$ are negative subformulas of A_σ with maximally extended argument vectors, then $\vec{D} = \vec{D}'$.

Predicate $R(\sigma;\Xi)$ holds if (i) $\Xi = (\Xi', X^{\Theta\Rightarrow p})$ implies that there is $(y : \vec{B} \supset p) \in \Theta$, and $B_j \in \vec{B}$ s.t. $\sigma = \Theta \Rightarrow B_j$; and (ii) $\Xi = \Xi', X^{\Theta\Rightarrow p}, Y^{\Theta'\Rightarrow q}, \Xi''$ implies that there is $(y : \vec{B} \supset p) \in \Theta$, and $B_j \in \vec{B}$ s.t. $B_j = \vec{D} \supset q$, and $\Theta' = \Theta \cup \{z_1 : D_1, \cdots, z_m : D_m\}$. ¶

The following two claims are easily proved by induction on $\mathcal{F}(\sigma;\Xi)$:

Claim 1. If $S(\sigma)$ holds and $\mathcal{F}(\sigma';\Xi')$ is a recursive call in the computation of $\mathcal{F}(\sigma;\Xi)$ (for any Ξ), then $S(\sigma')$ holds.

Claim 2. If $R(\sigma;\Xi)$ holds and $\mathcal{F}(\sigma';\Xi')$ is a recursive call in the computation of $\mathcal{F}(\sigma;\Xi)$, then $R(\sigma';\Xi')$ holds.

The following is also needed:

Claim 3. If $\sigma = \Gamma \Rightarrow \vec{A} \supset p$, and $\Xi = \Xi_1, X^{\Theta\Rightarrow p}, \Xi_2$, and $R(\sigma;\Xi)$ holds, then $\mathcal{F}(\sigma;\Xi)$ is not strongly acyclic.

This claim is proved by induction on $\mathcal{F}(\sigma;\Xi)$.

If the if-guard of $\mathcal{F}(\sigma;\Xi)$ holds, $\mathcal{F}(\sigma;\Xi)$ has form $\lambda z_1^{A_1} \cdots z_n^{A_n}.Y^{\sigma'}$, which is not strongly acyclic.

If not, let $\Delta := \Gamma \cup \{z_1 : A_1, \ldots, z_n : A_n\}$ and $\sigma' := \Delta \Rightarrow p$. Then, $\mathcal{F}(\sigma;\Xi)$ is given by (*) above.

As before, let $\Xi' := \Xi, Y : \sigma'$ and, for a given choice of $(y : \vec{B} \supset p) \in \Delta$ and index j, let $\sigma_j := \Delta \Rightarrow B_j$.

Subcase $\Xi_2 = \emptyset$. Because $R(\sigma;\Xi)$ (part (i)) holds, there is $(y : \vec{B} \supset p) \in \Gamma$ s.t. $\vec{A} \supset p = B_j$ for some j. So, $\mathcal{F}(\sigma_j;\Xi') = \lambda w_1^{A_1} \cdots w_n^{A_n}.Y^{\Delta'\Rightarrow p}$, for $\Delta' := \Delta \cup \{w_1 : A_1, \ldots, w_n : A_n\}$ (since $|\Delta| = |\Delta'|$), hence this recursive call is not strongly acyclic, and so is $\mathcal{F}(\sigma;\Xi)$.

Subcase $\Xi_2 = Z^{\Theta'\Rightarrow q}, \Xi_3$. Because $R(\sigma;\Xi)$ (part (ii)) holds, there is $(y : \vec{B} \supset p) \in \Theta$ s.t. $B_j = \vec{D} \supset q$ for some j. It is an easy observation that $\Theta \subseteq \Delta$ (even $\Theta \subseteq \Gamma$), hence there is a recursive call $\mathcal{F}(\Delta \Rightarrow \vec{D} \supset q;\Xi')$, and thanks to Claim 2 and to the fact that q is a target atom of Ξ', the induction hypothesis guarantees that this recursive call produces a not strongly acyclic term, and so $\mathcal{F}(\sigma;\Xi)$ is not strongly acyclic.

Another observation needed is:

Claim 4. If $S(\sigma)$ and $\mathsf{EF}_{\emptyset}(\mathcal{F}(\sigma;\Xi))$, then $\mathcal{F}(\sigma;\Xi)$ is strongly acyclic.

Again, this claim is proved by induction on $\mathcal{F}(\sigma;\Xi)$. The if-guard of $\mathcal{F}(\sigma;\Xi)$ cannot hold because of $\mathsf{EF}_{\emptyset}(\mathcal{F}(\sigma;\Xi))$. So $\mathcal{F}(\sigma;\Xi)$ is given by (*). Because $\mathsf{EF}_{\emptyset}(\mathcal{F}(\sigma;\Xi))$, for some $(y : \vec{B} \supset p) \in \Delta$, $\mathsf{EF}_{\emptyset}(y\langle\mathcal{F}(\sigma_j;\Xi')\rangle_j)$, hence for all j, we get $\mathsf{EF}_{\emptyset}(\mathcal{F}(\sigma_j;\Xi'))$, and because of $S(\sigma)$, and Claim 1, the induction hypothesis can be used to conclude $\mathcal{F}(\sigma_j;\Xi')$ is strongly acyclic. Because of $S(\sigma)$ all summands in (*) are of the form $y'\langle\mathcal{F}(\sigma_j;\Xi')\rangle_j$ (with $(y' : \vec{B} \supset p) \in \Delta$), hence all summands in (*) are strongly acyclic, and so is $\mathcal{F}(\sigma;\Xi)$.

Now we prove:

Claim 5. If $N(\sigma;\Xi)$, $S(\sigma)$, $R(\sigma;\Xi)$ and $\mathsf{EF}_{\emptyset}(\mathcal{F}(\sigma;\Xi))$, then $\mathcal{F}(\sigma;\Xi)$ is deterministic.

Again, this claim is proved by induction on $\mathcal{F}(\sigma;\Xi)$, and the if-guard of $\mathcal{F}(\sigma;\Xi)$ cannot hold because of $\mathsf{EF}_{\emptyset}(\mathcal{F}(\sigma;\Xi))$. So, $\mathcal{F}(\sigma;\Xi)$ is given by (*). From, $\mathsf{EF}_{\emptyset}(\mathcal{F}(\sigma;\Xi))$ there is

¶ *R* comes from *recursive*, and has to do with the observation that any recursive call resulting from $\mathcal{F}(\sigma;\emptyset)$ satisfies *R*. This is not intrinsically related to the control of negative occurrences of atoms.

at least one summand. From the assumptions and Claims 3 and 4, it must be the case that p is not a target atom of Ξ. For each recursive call $\mathcal{F}(\sigma_j;\Xi')$, using the assumptions, Claims 1 and 2 and reasoning analogous to that used in the previous claim, we have $S(\sigma_j)$, $R(\sigma_j;\Xi')$ and $\mathsf{EF}_\emptyset(\mathcal{F}(\sigma_j;\Xi'))$, and from assumption $N(\sigma;\Xi)$, we also get $N(\sigma_j;\Xi')$ from (**) above. Hence, by induction hypothesis, $\mathcal{F}(\sigma_j;\Xi')$ is deterministic. It remains to argue that there is exactly one summand in (*), but this is a consequence of $N(\sigma;\Xi)$ and $p \notin \Xi$, which guarantee that there is at most one $(y : \vec{B} \supset p) \in \Delta$.

Finally, the following claim suffices to conclude the proof of the lemma:

Claim 6. If no atom occurs negatively in σ more than once, and $\mathsf{EF}_\emptyset(\mathcal{F}(\sigma;\emptyset))$, then $\mathcal{F}(\sigma;\emptyset)$ is deterministic.

This follows by taking $\Xi = \emptyset$ in Claim 5, since $N(\sigma;\emptyset)$ and $S(\sigma)$ are both consequences of σ having no atom occurring negatively more than once, and since $R(\sigma;\emptyset)$ holds vacuously. □

Theorem 46 (Generalized coherence).

1. If no atom occurs positively more than once in A, then A has only finitely many inhabitants.
2. If no atom occurs negatively more than once in A, then A has at most one inhabitant.

Proof. Item 1: Suppose no atom occurs positively more than once in A. By Lemma 45, $\mathcal{F}(\Rightarrow A)$ is strongly acyclic. By Lemma 44, $\mathsf{FF}_\emptyset(\mathcal{F}(\Rightarrow A))$. By Theorem 33.2, $\mathcal{I}(\Rightarrow A)$ is finite. Item 2: Suppose no atom occurs negatively more than once in A. If $\mathsf{EF}_\emptyset(\mathcal{F}(\Rightarrow A))$, then by Lemma 45, $\mathcal{F}(\Rightarrow A)$ is deterministic. By Lemma 44, $\#(\mathcal{F}(\Rightarrow A)) \leqslant 1$, and also $\mathsf{FF}_\emptyset(\mathcal{F}(\Rightarrow A))$; hence, by Theorem 42, $\#(\mathcal{F}(\Rightarrow A))$ is the cardinality of $\mathcal{I}(\Rightarrow A)$ – and this is $\leqslant 1$. If $\mathsf{EF}_\emptyset(\mathcal{F}(\Rightarrow A))$ does not hold, then by Theorem 24.2, $\mathcal{I}(\Rightarrow A)$ is empty. □

We remark that the condition $\mathsf{EF}_\emptyset(\mathcal{F}(\Rightarrow A))$ in Lemma 45.2 cannot be omitted: consider the formula $A := ((((r^- \supset p^+) \supset q^-) \supset q^+) \supset p^-) \supset p^+$ with different atoms p, q, r, where the polarities are indicated as superscripts. It has only one negative occurrence of these atoms, but $\mathcal{F}(\Rightarrow A)$ is not deterministic. (Atom p has two positive occurrences in A, hence the classical coherence theorem does not apply, but the generalized one that we reproved here.)

Trivially, if the conditions on the occurrences in the theorem are not met, the conclusions can become wrong: $\mathsf{CHURCH} = (p^+ \supset p^-) \supset p^- \supset p^+$ has two positive occurrences of p and infinitely many inhabitants (by the second part of the theorem, the two negative occurrences are needed for that as well), while $p^- \supset p^- \supset p^+$ has two negative occurrences of p and two inhabitants.

5. Final remarks

This paper develops a methodology to address inhabitation problems in the simply typed λ-calculus based on the finitary representation of the search space offered by the calculus $\lambda^{\mathsf{gfp}}_{\Sigma}$. With it, we could cover a representative range of decision and counting problems and two recent generalizations of the coherence theorem, confirming the scope of the methodology. Moreover, we obtained new solutions and new proofs which shed new

light on the problems and theorems, as they also come with a structural analysis: there are *syntax-directed* definitions of certain crucial predicates and functions, and the syntax underneath is precisely $\lambda_{\Sigma}^{\mathsf{gfp}}$. In the background, there is the calculus λ_{Σ}^{co}, as the semantic domain against which we assess the development in $\lambda_{\Sigma}^{\mathsf{gfp}}$. λ_{Σ}^{co} provides an extension of the Curry–Howard paradigm of representation from proofs, to search runs, and to the search space. Because of this paradigm, binding and α-equivalence are unproblematic, and we did not have to concern with discharge conventions or resort to methods or binding representations in other areas than λ-calculus and types.

Many approaches exist in the study of inhabitation problems in the simply typed λ-calculus, some adapting tools from automata and language theory (Dowek and Jiang 2009; Schubert et al. 2015; Takahashi et al. 1996) or game theory (Bourreau and Salvati 2011), others creating new representations like in the formula-tree method (Alves and Broda 2015; Broda and Damas 2005) and others through a direct combinatorial analysis of a graph-theoretic representation of the search space (Wells and Yakobowski 2004). All these approaches have their merits, and making connections to other fields is one of them. But our structural approach also has noteworthy properties, like the elegance and novelty of the solutions for a wide range of problems, never covered before by a single approach, going from decision algorithms to coherence. In addition, given the paradigm and standpoint of our approach, not only does it occupy a special place, but also it enjoys certain advantages coming from the proximity to the λ-calculus, which are not to be dismissed: one is the possibility of a mechanical formalization, the other is the already referred absence of problems with binding, on which we want to insist.

In fact, we see in some of the alternative approaches mentioned above the struggle with α-equivalence and discharge conventions, which is the price to pay for having to account for binding with foreign tools. For instance, several times we see the need to resort to the total discharge convention in order to obtain finite structures, like grammars or graphs, and effective procedures (Dowek and Jiang 2009; Takahashi et al. 1996; Wells and Yakobowski 2004); and only in a second stage may one enumerate or count all the proofs, after some expansion procedure to capture the inhabitants that do not obey the convention. But we already see the indication that this second stage, in more complex logics, requires hard work (Dowek and Jiang 2009). In our approach, instead, there is no second stage, nor do we need to care about the discharge convention: $\mathcal{F}(\sigma)$ is immediately and simultaneously the full space and data for algorithms.

The present paper covers simple types only, as most papers in the field (notable exceptions are Dowek and Jiang (2009) and Wells and Yakobowski (2004)). Our methodology for attacking decision problems related to inhabitation rests on the availability of a correct and effective representation of the search space in a finitary calculus like $\lambda_{\Sigma}^{\mathsf{gfp}}$. For the logic studied here, such representation was obtained previously (Espírito Santo et al. 2013, 2016), and is a consequence of the subformula property, as a close inspection reveals. So any other logic with this property is amenable to our methodology in principle, and we are interested in investigating whether our methodology can obtain again 'structural' solutions for decision problems in richer logics. For example, it would be interesting to know if, in the presence of a connective like disjunction, our methodology produces a (simple) decision function for the INHAB problem; or if we can even move to the $\lambda\mu$-calculus, as

was done by David and Zaionc (2009), or to the rich setting of intersection types, whose general inhabitation problem is undecidable, but where recent developments (Bucciarelli et al. 2014; Dudenhefner and Rehof 2017) identify decidable fragments.

We would like to thank our anonymous referees for their detailed and thoughtful reviews. The first and the last author were partially financed by Fundação para a Ciência e a Tecnologia (FCT) through project UID/MAT/00013/2013. The second author was partially financed by the project *Climt*, ANR-11-BS02-016, of the French Agence Nationale de la Recherche. All authors got financial support by the COST action CA15123 EUTYPES.

References

Alves, S. and Broda, S. (2015). A short note on type-inhabitation: Formula-trees vs. game semantics. *Information Processing Letters* **115** (11) 908–911.

Aoto, T. and Ono, H. (1994). Uniqueness of normal forms in $\{\rightarrow, \wedge\}$-fragment of NJ. Technical report, Research Report IS-RR-94-0024F.

Barendregt, H., Dekkers, W. and Statman, R. (2013). *Lambda Calculus with Types*. Perspectives in Logic, Cambridge University Press.

Ben-Yelles, C.-B. (1979). *Type Assignment in the Lambda-Calculus: Syntax & Semantics*. PhD thesis, University College of Swansea.

Bourreau, P. and Salvati, S. (2011). Game semantics and uniqueness of type inhabitance in the simply-typed λ-calculus. In: Ong, L. (ed.), *Proceedings of TLCA 2011*, LNCS, vol. 6690, Springer, 61–75.

Broda, S. and Damas, L. (2005). On long normal inhabitants of a type. *Journal of Logic and Computation* **15** (3) 353–390.

Bucciarelli, A., Kesner, D., and Rocca, S. R. D. (2014). The inhabitation problem for non-idempotent intersection types. In: Díaz, J., Lanese I., and Sangiorgi, D. (eds.), *Proceedings of IFIP TCS 2014*, LNCS, vol. 8705, Springer, 341–354.

David, R. and Zaionc, M. (2009). Counting proofs in propositional logic. *Archive for Mathematical Logic* **48** (2) 185–199.

Dowek, G. and Jiang, Y. (2009). Enumerating proofs of positive formulae. *The Computer Journal* **52** (7) 799–807.

Dudenhefner, A. and Rehof, J. (2017). Intersection type calculi of bounded dimension. In: *Proceedings of POPL 2017*, ACM, 653–665.

Espírito Santo, J., Matthes, R., and Pinto, L. (2013). A coinductive approach to proof search. In: Baelde, D. and Carayol, A. (eds.), *Proceedings of FICS 2013*, vol. 126, EPTCS, 28–43.

Espírito Santo, J., Matthes, R., and Pinto, L. (2016). A coinductive approach to proof search through typed lambda-calculi. Available at `http://arxiv.org/abs/1602.04382v2`.

Hindley, J. R. (1997). *Basic Simple Type Theory*, vol. 42. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press.

Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley.

Liang, C. and Miller, D. (2009). Focusing and polarization in linear, intuitionistic, and classical logic. *Theoretical Computer Science* **410** 4747–4768.

Miller, D. and Nadathur, G. (2012). *Programming with Higher-Order Logic*, Cambridge University Press.

Mints, G. (1979). A coherence theorem for cartesian closed categories (abstract). *The Journal of Symbolic Logic* **44** 453–454.

Mints, G. (1992). A simple proof for the coherence theorem for cartesian closed categories. In: *Selected papers in proof theory*, vol. 3, 213–220 North-Holland Publishing Co., Amsterdam.

Schubert, A., Dekkers, W., and Barendregt, H. P. (2015). Automata theoretic account of proof search. In: Kreutzer, S. (ed.), *Proceedings CSL 2015*, LIPIcs, vol. 41, Schloss Dagstuhl, 128–143.

Takahashi, M., Akama, Y., and Hirokawa, S. (1996). Normal proofs and their grammar. *Information and Computation* **125** (2) 144–153.

Wells, J. B. and Yakobowski, B. (2004). Graph-based proof counting and enumeration with applications for program fragment synthesis. In: *Proceedings of LOPSTR 2004*, LNCS, vol. 3573, Springer, 262–277.