The Complexity of Principal Inhabitation

Andrej Dudenhefner[1] and Jakob Rehof[2]

1 **Department of Computer Science, Technical University of Dortmund, Dortmund, Germany**
andrej.dudenhefner@cs.tu-dortmund.de

2 **Department of Computer Science, Technical University of Dortmund, Dortmund, Germany**
jakob.rehof@cs.tu-dortmund.de

Abstract

It is shown that in the simply typed λ-calculus the following decision problem of *principal inhabitation* is PSPACE-complete: Given a simple type τ, is there a λ-term N in β-normal form such that τ is the principal type of N?

While a Ben-Yelles style algorithm was presented by Broda and Damas in 1999 to count normal principal inhabitants (thereby answering a question posed by Hindley), it does not induce a polynomial space upper bound for principal inhabitation. Further, the standard construction of the polynomial space lower bound for simple type inhabitation does not carry over immediately.

We present a polynomial space bounded decision procedure based on a characterization of principal inhabitation using path derivation systems over subformulae of the input type, which does not require candidate inhabitants to be constructed explicitly. The lower bound is shown by reducing a restriction of simple type inhabitation to principal inhabitation.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Lambda Calculus, Type Theory, Simple Types, Inhabitation, Principal Type, Complexity

Digital Object Identifier 10.4230/LIPIcs.FSCD.2017.15

1 Introduction and Related Work

The inhabitation problem for simply typed λ-calculus [1] (given a type, is there a λ-term having the type?) is known to be PSPACE-complete by a well-known result of Statman [11]. Due to the subject reduction and normalization theorems for simple types, it is sufficient to decide the existence of inhabitants in β-normal form. A natural related problem is the problem of *principal inhabitation*: Given a type τ, is there a *normal principal inhabitant* of τ? A normal principal inhabitant of τ is a λ-term in β-normal form having τ as its principal type [8, Definition 8A11]. The principal inhabitation problem is different from the inhabitation problem. For, whereas every inhabited type τ is also the principal type of *some* λ-term [8, Lemma 7A2 (i)], this is not the case when we restrict attention to inhabitants in β-normal form: Some inhabited types are not principally inhabited, because they are not the principal types of any β-normal form. For example, $\tau = a \to a \to a$ is inhabited by $K \equiv \lambda x.\lambda y.x$, but τ is not the principal type of K (its principal type is $a \to b \to a$). In fact, there is no β-normal form having τ as its principal type (cf. [8, Remark 8A13 (iii)]), therefore τ is not principally inhabited. Since normal principal inhabitants can be seen as natural implementations of a given type specification in the context of type-based program synthesis [9, 6, 3], principal inhabitation is not only of systematic but also of practical importance.

2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017).
Editor: Dale Miller; Article No. 15; pp. 15:1–15:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In this paper we are concerned with the complexity of the principal inhabitation problem. The only known results directly related to the upper bound for principal inhabitation are the counting procedure by Broda and Damas [4] and its generalization by principal proof trees in [5]. Broda and Damas present in [4] a Ben-Yelles style counting algorithm [2, 8] for normal principal inhabitants, thereby solving a problem mentioned by Hindley in [8, Problem 8D10 (i)]. In [5], a more general technique (formula-tree proof method) is used to construct so-called principal proof trees deciding principal inhabitation. However, these results do not immediately imply a polynomial space upper bound for principal inhabitation. In particular, the counting procedure of [4] operates by explicitly enumerating inhabitants and checking for principality in each case. Although a depth-bound is provided on inhabitant terms, which is polynomial in the size of the input type τ, inhabitants may be of *exponential size*. Therefore, the upper bound for principal inhabitation induced by the procedure is exponential time. Because principality is a *global property* of a derivation and is therefore sensitive to the exact structure of inhabitants, it does not appear to be obvious how to obviate an exponential time construction. Similarly to [4], principal proof trees in [5] can be of exponential size, inducing a similar complexity as the previous approach. Further remarks comparing details of our decision procedure with the approaches in [4, 5] can be found within the technical development of the paper.

In comparison with the inhabitation problem for simple types, basic challenges for a polynomial space upper bound for principal inhabitation include the following two complications. For one, it is not possible to bound the size of the type environment during inhabitant search by simply coalescing type variables having the same type. In the standard approach [12], when searching for a long normal inhabitant[1] of a function type $\sigma \to \tau$, an assumption $(x : \sigma)$ is added to the environment only if it does not already contain some variable of type σ. This leads to a linear upper bound on environment size, because (as a consequence of the subformula property for normal forms) only subformulae of the original input need to be considered during inhabitant search. However, this approach leads to an incomplete procedure for principal inhabitation. Consider as an example principally inhabiting $(a \to a \to a) \to a \to a \to a$. The procedure would (implicitly) discover a λK-term $\lambda f.\lambda x.\lambda y.f(fxx)(fxx)$ as inhabitant in which the fifth (second from right) occurrence of a is implicitly associated with a variable y which is not used, because it is coalesced with x (also of type a) in the body of the term. But this term is not a principal inhabitant, whereas $\lambda f.\lambda x.\lambda y.f(fxy)(fyx)$ is. In essence, the solution to this problem for principal inhabitation lies in only coalescing type assumptions associated with the same subformula *occurrence* in the goal type. This approach is realized in our solution by keeping track of such occurrences and relations between them using a calculus of paths (subformula calculus) which distinguishes subformula occurrences.

The second complication in comparison with the standard procedure has to do with certain kinds of cyclic situations. The alternating search procedure of [12] does not need to inhabit a goal which has already appeared under the same assumptions on the current branch of the search tree, but such a strategy would be incomplete for principal inhabitation. To illustrate, consider the type $\tau \equiv (a \to a) \to a \to a$. It is inhabited by every Church numeral, but not principally so. Whereas the standard procedure would determine the inhabitant $\mathbf{c}_0 = \lambda f.\lambda x.x$, only the Church numerals $\mathbf{c}_n$ for $n \geq 2$ are normal principal inhabitants of τ. For example, $\mathbf{c}_2 = \lambda f.\lambda x.f(fx)$ is a normal principal inhabitant of τ, because the cyclic proof structure – proving inhabitation of a by (fx), although there is already an

[1] By a long normal inhabitant is meant a λ-term in η-long β-normal form, see [8, Definition 8A7].

inhabitant, x, of a in a subexpression (premise) – forces identification of the domain and range types of f. This phenomenon can apparently get more complicated. For example, the type $(a \to a) \to (a \to a) \to a \to a$ is principally inhabited by the term $\lambda f.\lambda g.\lambda x.f(g(g(fx)))$, but neither by $\lambda f.\lambda g.\lambda x.g(g(g(gx)))$ nor by $\lambda f.\lambda g.\lambda x.f(g(f(gx)))$. The complications arising from this phenomenon are handled by path deduction systems characterizing exactly the necessary and sufficient identifications among subformula occurrences of types without explicit reference to inhabitant terms. An important instrument to this end is an adaptation of the *subformula filtration* technique, which was introduced in [7] for the intersection type system.

To provide an upper bound, we present a polynomial space bounded decision procedure based on a characterization of principal inhabitation using a calculus over subformulae of the input type, which does not require candidate inhabitants to be constructed explicitly.

With regard to the lower bound, one cannot directly transfer the polynomial space lower bound for the inhabitation problem [11, 12], because it turns out (as will be shown) that the standard reduction (cf. [12]) from truth of quantified Boolean formulae uses types which are not necessarily principally inhabited. However, we observe that the standard reduction induces a PSPACE-hard restriction of simple type inhabitation. Therefore, for the polynomial space lower bound, we reduce this particular restriction to principal inhabitation.

The paper is organized as follows. After preliminary definitions (Section 2) we introduce (Section 3) subformula filtration to obtain a necessary condition (Lemma 14) on the form of type derivations for principal inhabitants, which will be of pervasive importance in the paper. We then (Section 4) define the subformula calculus, which allows us to talk about subformula occurrences and relations between them in type derivations to characterize principal inhabitants (Theorem 32). In Section 5 we present the algorithm (INH) to decide principal inhabitation and prove the polynomial space upper bound. The proof of the PSPACE lower bound is given in Section 6. We conclude the paper in Section 7 which also contains remarks about future work.

2 Simply-Typed Lambda Calculus

In this section we briefly assemble the necessary prerequisites in order to discuss principal inhabitation in the simply typed λ-calculus. We denote λ-terms (cf. Definition 1) by L, M, N and simple types (cf. Definition 2) are denoted by σ, τ, ρ, where type atoms are denoted by a, b, c and drawn from the denumerable set $\mathbb{A}$. The rules (Ax), ($\to$I) and ($\to$E) of the simple type system are given in Definition 3.

▶ **Definition 1** (λ-Terms). $L, M, N ::= x \mid (\lambda x.M) \mid (M\,N)$.

▶ **Definition 2** (Simple Types). $\sigma, \tau, \rho ::= a \mid \sigma \to \tau$ where $a \in \mathbb{A}$.

▶ **Definition 3** (Simple Type System).

$$\frac{}{\Gamma, x : \sigma \vdash x : \sigma}\,(\text{Ax}) \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}\,(\to\text{I}) \qquad \frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\ N : \tau}\,(\to\text{E})$$

We write $\mathcal{D} \triangleright \Gamma \vdash M : \tau$, if the derivation $\mathcal{D}$ derives the judgement $\Gamma \vdash M : \tau$, i.e. $\mathcal{D}$ is a finite tree of judgements with root $\Gamma \vdash M : \tau$ that respects the corresponding typing rules.

Type substitutions (cf. [8, Definition 3A1]) are denoted by S and are lifted from type atoms to types. A principal type (cf. Definition 4) of a term is the most general type that can be assigned to that term and is unique up to atom renaming. A normal principal inhabitant (cf. Definition 5) is a closed β-normal form for which the given type is principal.

▶ **Definition 4** (Principal Type)**.** We say that τ is a *principal type* of M, if $\vdash M : \tau$ and for all types σ such that $\vdash M : \sigma$ there exists a substitution S such that $S(\tau) = \sigma$.

▶ **Definition 5** (Normal Principal Inhabitant)**.** We say that a λ-term M in β-normal form is a *normal principal inhabitant* of τ, if τ is the principal type of M.

As usual, term application is left-associative. In accordance with [8], we define Long(τ) as the set of all long normal inhabitants of τ.

▶ **Definition 6** (Long(τ))**.** The set Long(τ) consists of all λ-terms M such that $\vdash M : \tau$ is derivable using only the rule ($\to$I) and the following rule ($\to_L$E)

$$\frac{\Gamma, x : \sigma_1 \to \ldots \to \sigma_n \to a \vdash M_i : \sigma_i \text{ for } i = 1 \ldots n}{\Gamma, x : \sigma_1 \to \ldots \to \sigma_n \to a \vdash x\ M_1 \ldots M_n : a}\ (\to_L\mathrm{E})$$

Clearly, longness is not violated by generalization (cf. Lemma 7) and η-expansion does not violate principality (cf. Lemma 8).

▶ **Lemma 7.** *If $M \in$ Long($S(\tau)$) and $\vdash M : \tau$, then $M \in$ Long(τ).*

▶ **Lemma 8** ([8, 8A11.2])**.** *If a β-normal form M has the principal type τ, then its unique η-expansion $M^+ \in$ Long(τ) has the principal type τ.*

Our main result is that the following principal inhabitation problem (cf. Problem 1) is PSPACE-complete.

▶ **Problem 1** (Principal Inhabitation)**.** *Given a simple type τ, is there a λ-term M in β-normal form such that τ is the principal type of M?*

▶ **Theorem 9.** *The principal inhabitation problem (cf. Problem 1) is* PSPACE*-complete.*

Proof. The upper bound is shown in Section 5 Lemma 38 and the lower bound is shown in Section 6 Lemma 42. ◀

3 Subformula Filtration

The *subformula filtration* technique, which was developed for the intersection type system in [7], eliminates unnecessary structure in type derivations. It can be used to show that if M is typable, then there exists a type derivation $\mathcal{D}$ for M such that any subformula of any type occurring in $\mathcal{D}$ also appears on the right-hand side of some judgement in $\mathcal{D}$. This can be seen as a generalization of the standard subformula property that only requires right-hand sides of judgements in $\mathcal{D}$ to be subformulae of types appearing in the root judgement of $\mathcal{D}$. Transferring the technique to the simply typed λ-calculus we obtain a necessary condition for principal inhabitation (cf. Lemma 14).

First, we adapt definitions from [7] to the simply typed λ-calculus, including that of the set $T(\mathcal{D})$ of types occurring on the right-hand sides of judgements in a given type derivation $\mathcal{D}$, and that of the notion of type filtration.

▶ **Definition 10** ($T(\mathcal{D})$)**.** Given a type derivation $\mathcal{D}$ we define the set $T(\mathcal{D}) = \{\tau \mid \Gamma \vdash M : \tau \text{ is a judgement in } \mathcal{D}\}$.

▶ **Definition 11** (Filtration Function $\mathcal{F}^a_X$)**.** Given a set X of types and a type atom a we define the *filtration function* $\mathcal{F}^a_X$ as follows

$$\mathcal{F}^a_X(b) = a \qquad \mathcal{F}^a_X(\sigma \to \tau) = \begin{cases} \mathcal{F}^a_X(\sigma) \to \mathcal{F}^a_X(\tau) & \text{if } \sigma \to \tau \in X \text{ and } \tau \in X \\ a & \text{otherwise} \end{cases}$$

Intuitively, a filtration function $\mathcal{F}^a_X$ collapses all type atoms and unnecessary subformulae wrt. X into a single type atom a. Let us tacitly lift filtration functions pointwise to type environments by $\mathcal{F}^a_X(\Gamma) = \{x : \mathcal{F}^a_X(\sigma) \mid (x : \sigma) \in \Gamma\}$.

Next, we formulate the corresponding filtration lemma for the simply typed λ-calculus.

▶ **Lemma 12.** *If $\mathcal{D} \triangleright \Gamma \vdash M : \tau$ and $T(\mathcal{D}) \subseteq X$, then $\mathcal{F}^a_X(\Gamma) \vdash M : \mathcal{F}^a_X(\tau)$, where a is fresh.*

Proof. Routine induction on the derivation $\mathcal{D}$.

Case (Ax): Clearly, $\mathcal{F}^a_X(\Gamma), x : \mathcal{F}^a_X(\sigma) \vdash x : \mathcal{F}^a_X(\sigma)$.

Case ($\to$I): The last rule is $\dfrac{\Gamma, x : \sigma' \vdash N : \tau'}{\Gamma \vdash \lambda x.M : \sigma' \to \tau'}\,(\to\mathrm{I})$.

We have $\tau' \in T(\mathcal{D}) \subseteq X$ and $\sigma' \to \tau' \in T(\mathcal{D}) \subseteq X$, therefore $\mathcal{F}^a_X(\sigma' \to \tau') = \mathcal{F}^a_X(\sigma') \to \mathcal{F}^a_X(\tau')$. By the induction hypothesis we have $\mathcal{F}^a_X(\Gamma), x : \mathcal{F}^a_X(\sigma') \vdash N : \mathcal{F}^a_X(\tau')$, which using ($\to$I) shows the claim.

Case ($\to$E): The last rule is $\dfrac{\Gamma \vdash N : \sigma' \to \tau' \qquad \Gamma \vdash L : \sigma'}{\Gamma \vdash N\ L : \tau'}\,(\to\mathrm{E})$.

We have $\tau' \in T(\mathcal{D}) \subseteq X$ and $\sigma' \to \tau' \in T(\mathcal{D}) \subseteq X$. Similarly to the previous case, the claim follows using the definition of $\mathcal{F}^a_X$, the induction hypothesis and the rule ($\to$E). ◀

The above Lemma 12 is useful to eliminate unnecessary subformulae in derivations as illustrated by the following Example 13.

▶ **Example 13.** Let $\sigma = b \to b$ and consider the derivation $\mathcal{D} = \dfrac{\dfrac{}{x : \sigma \vdash x : \sigma}\,(\mathrm{Ax})}{\vdash \lambda x.x : \sigma \to \sigma}\,(\to\mathrm{I})$.

We have $b \notin T(\mathcal{D}) = \{\sigma, \sigma \to \sigma\}$. Therefore, $\mathcal{D}$ contains unnecessary structure in order to type $\lambda x.x$. Applying $\mathcal{F}^a_{T(\mathcal{D})}$ we obtain $\dfrac{\dfrac{}{x : a \vdash x : a}\,(\mathrm{Ax})}{\vdash \lambda x.x : a \to a}\,(\to\mathrm{I})$, noting that $\mathcal{F}^a_{T(\mathcal{D})}(b \to b) = a$ because $b \notin T(\mathcal{D})$.

Finally, we conclude this section with a necessary condition for type derivations of principal types, which is connected to Property ($\star$) in Section 4 and, specifically, Lemma 30.

▶ **Lemma 14.** *If $\mathcal{D} \triangleright \emptyset \vdash M : \tau$ and τ contains a subformula $\sigma' \to \tau'$ such that $\tau' \notin T(\mathcal{D})$, then τ is not the principal type of M.*

Proof. By Lemma 12 we have $\emptyset \vdash M : \mathcal{F}^a_{T(\mathcal{D})}(\tau)$, where a is fresh. Since $\tau' \notin T(\mathcal{D})$, in $\mathcal{F}^a_{T(\mathcal{D})}(\tau)$ the corresponding subformula at the position of $\sigma' \to \tau'$ in τ is either undefined or a. Therefore, there is no substitution S such that $S(\tau) = \mathcal{F}^a_{T(\mathcal{D})}(\tau)$. ◀

4 Subformula Calculus

To distinguish distinct subformula occurrences in a given type τ, we use *paths* π in the syntax tree of τ, which are defined as follows

$$\pi \in \{1, 2\}^* .$$

Since paths are character sequences, we use abbreviations such as $\pi 2^n$ for the path π followed by n twos. We access a subformula at path π in a given type τ by $\tau(\pi)$, defined as

$$\tau(\varepsilon) = \tau\,, \qquad (\sigma \to \tau)(1\pi) = \sigma(\pi)\,, \qquad (\sigma \to \tau)(2\pi) = \tau(\pi)\,.$$

The above definition implies that we use types as functions from the set of their paths to their subformulae. In particular, $\mathrm{dom}(\tau)$ is the set of paths in τ and $\mathrm{ran}(\tau)$ is the set of subformulae in τ.

Similarly to the simply typed system, we define *path environments* $\Delta = \{x_1 : \pi_1, \dots, x_n : \pi_n\}$, where $\mathrm{dom}(\Delta) = \{x_1, \dots, x_n\}$. For a relation R on paths, the calculus $\vdash_R$ is given by rules ($\to_R$I) and ($\to_R$E) in the following Definition 15.

▶ **Definition 15** (Calculus $\vdash_R$).

$$\frac{\Delta, x : \pi 1 \vdash_R M : \pi 2}{\Delta \vdash_R \lambda x.M : \pi}\,(\to_R\mathrm{I}) \qquad \frac{\pi 2^n \mathrel{R} \pi' \qquad \Delta, x : \pi \vdash_R M_i : \pi 2^{i-1}1 \text{ for } i = 1 \dots n}{\Delta, x : \pi \vdash_R x\, M_1 \dots M_n : \pi'}\,(\to_R\mathrm{E})$$

We call conditions of the form $\pi \mathrel{R} \pi'$ *side conditions.* The above calculus $\vdash_R$, similarly to the calculus $\mathrm{TA}_{\mathrm{pln}}$ in [4], captures as side conditions identities imposed by the typed term. In contrast to $\mathrm{TA}_{\mathrm{pln}}$ it does not contain or require actual type information. Additionally, for any closed λ-term M in β-normal form there exists a relation R such that $\vdash_R M : \varepsilon$. Clearly, $\vdash_R$ is monotonous in the sense of the following Lemma 16.

▶ **Lemma 16.** *If $\vdash_R M : \varepsilon$ and $R \subseteq R'$, then $\vdash_{R'} M : \varepsilon$.*

As in the simply typed system, paths on the left-hand side (resp. right-hand side) of $\vdash_R$ are of negative (resp. positive) variance, which is formalized in the following Lemma 17.

▶ **Lemma 17.** *If $\mathcal{D} \rhd \emptyset \vdash_R M : \varepsilon$, then each judgement $\Delta \vdash_R N : \pi$ in $\mathcal{D}$ satisfies*
(i) *The number of 1s in π is even.*
(ii) *For each $(x : \pi') \in \Delta$ the number of 1s in π' is odd.*

Proof. Induction on depth of derivation for the more general claim: if $\Delta \vdash_R M : \pi$ is derived by $\mathcal{D}$ and satisfies (i) and (ii), then each judgement in $\mathcal{D}$ satisfies (i) and (ii). Clearly, if the concluding judgement satisfies (i) and (ii), then all premise judgements satisfy (i) and (ii) in both ($\to_R$I) and ($\to_R$E). ◀

The above observation restricts paths in side conditions as follows.

▶ **Corollary 18.** *If $\mathcal{D} \rhd \emptyset \vdash_R M : \varepsilon$, then $\mathcal{D}$ contains no side condition of the form $\pi \mathrel{R} \pi$.*

Intuitively, a derivation in $\vdash_R$ contains (as side conditions) necessary equality constraints on atomic subformulae that are required for typing a given term M. Therefore, we are interested in the minimal relation R such that $\vdash_R M : \varepsilon$.

Given a relation R let us denote the *reflexive, symmetric, transitive closure* of R by $R^{\equiv}$. Clearly, if $\vdash_R M : \varepsilon$, then $\vdash_{R^{\equiv}} M : \varepsilon$.

▶ **Definition 19** (R_M). Given a λ-term M in β-normal form, let R_M be the minimal (wrt. inclusion) equivalence relation such that $\vdash_{R_M} M : \varepsilon$.

Derivations in $\vdash_R$ are uniquely defined by the concluding judgement, therefore, the minimal relation R of necessary side conditions is uniquely defined as well. By monotonicity (cf. Lemma 16) we can take $R_M = R^{\equiv}$.

▶ **Example 20.** We have $R_{\lambda x.\lambda y.x} = \{(1, 22)\}^{\equiv}$ and $R_{\lambda x.\lambda y.y} = \{(21, 22)\}^{\equiv}$. Note that the domain of $R_{\lambda x.\lambda y.x}$ (resp. $R_{\lambda x.\lambda y.y}$) does not contain the path 21 (resp. 1) which would correspond to the type of y (resp. x).

Similar to the simply typed system, we can identify term variables in the path environment that are bound to same paths.

▶ **Lemma 21.** $\Delta, x : \pi, y : \pi \vdash_R M : \pi'$ *iff* $\Delta, x : \pi \vdash_R M[y := x] : \pi'$.

Proof. Induction on the derivation. The structure of both derivations is identical. ◀

The above Lemma 21 has a subtle implication regarding interchangeability of abstracted variables in a given term M referring to same paths without changing the corresponding relation R_M. This property will be crucial in the upper bound construction in Section 5 and is illustrated in the following Example 22.

▶ **Example 22.** Consider $M = \lambda f.f\ (\lambda x.f\ (\lambda y.y))$ and $M' = \lambda f.f\ (\lambda x.f\ (\lambda y.x))$. Both M and M' are normal principal inhabitants of $((a \to a) \to a) \to a$. Let $\Delta = \{f : 1, x : 111, y : 111\}$. The only difference between the derivation of $\vdash_{R_M} M : \varepsilon$ and a derivation of $\vdash_{R_{M'}} M' : \varepsilon$ is the leaf judgement. For the former it is $\Delta \vdash_{R_M} y : 112$ and for the latter $\Delta \vdash_{R_{M'}} x : 112$. By Lemma 21 we have $\Delta \vdash_{R_{M'}} y : 112$ and $\Delta \vdash_{R_M} x : 112$. Since the rest of the derivations is identical, we have $R_M = R_{M'}$.

The equivalence relation R_M intuitively captures equality constraints on atomic subformulae imposed by a given term M. Complementarily, given a type τ, we are interested in equality constraints on atomic subformulae satisfied by τ. To capture such constraints we define the equivalence relation R_τ in the following Definition 23.

▶ **Definition 23** (R_τ). Given a type τ we define the equivalence relation R_τ on paths in $\mathrm{dom}(\tau)$ as $R_\tau = \{(\pi, \pi') \mid \pi \neq \pi' \wedge \tau(\pi) = \tau(\pi') \in \mathbb{A}\}^{\equiv}$.

Observe that the condition $\pi \neq \pi'$ in the definition of R_τ excludes singular occurrences of type atoms in τ from the domain of R_τ while the subsequent equivalence closure ensures reflexivity. This is illustrated in the following Example 24.

▶ **Example 24.** We have $R_{a\to b\to a} = \{(1, 22)\}^{\equiv}$ and $R_{a\to b\to b} = \{(21, 22)\}^{\equiv}$. Similarly to Example 20 the domain of $R_{a\to b\to a}$ (resp. $R_{a\to b\to b}$) does not contain the path 21 (resp. 1).

Due to structural similarity between the rules ($\to_L$E) and ($\to_R$E) we obtain a simple characterization of long normal inhabitants of a given type in the following Lemma 25.

▶ **Lemma 25.** *Given a λ-term M in β-normal form, the following conditions are equivalent*

(i) $M \in \mathrm{Long}(\tau)$,
(ii) $\vdash_{R_\tau} M : \varepsilon$,
(iii) $R_M \subseteq R_\tau$.

Proof.

(i) $\implies$ (ii): Assume $\mathcal{D} \rhd \emptyset \vdash M : \tau$ using only the rules ($\to$I) and ($\to_L$E) (cf. Definition 6). By routine induction on $\mathcal{D}' \rhd \emptyset \vdash_{R_M} M : \varepsilon$ we have that for each judgement $\Delta \vdash_{R_M} N : \pi$ in $\mathcal{D}'$ there is a judgment $\{x : \tau(\pi') \mid (x : \pi') \in \Delta\} \vdash N : \tau(\pi)$ in $\mathcal{D}$. Therefore, if $\Delta, x : \pi \vdash_{R_M} x\ M_1 \dots M_n : \pi'$ is a judgement in $\mathcal{D}'$, then $\tau(\pi 2^n) = \tau(\pi') \in \mathbb{A}$. By Corollary 18 we additionally have $\pi 2^n \neq \pi'$, and ultimately $(\pi 2^n, \pi') \in R_\tau$. Therefore, $\vdash_{R_\tau} M : \varepsilon$.

(ii) $\implies$ (iii): If $\vdash_{R_\tau} M : \varepsilon$ but $R_M \not\subseteq R_\tau$, then R_M is not minimal, which contradicts the definition of R_M.

(iii) $\implies$ (i): Assume $R_M \subseteq R_\tau$. We directly translate the derivation of $\vdash_{R_M} M : \varepsilon$ to a derivation using rules ($\to$I) and ($\to_L$E) (cf. Definition 6). The side condition $\pi 2^n\ R_M\ \pi'$ in ($\to_{R_M}$E) implies $\tau(\pi 2^n) = \tau(\pi') \in \mathbb{A}$. Additionally, in case of ($\to_{R_M}$I) we have that $\pi 1 \in \operatorname{dom}(\tau)$ iff $\pi 2 \in \operatorname{dom}(\tau)$. ◀

Since R_M contains atomic equality constraints imposed by M, we are free to rename some atomic subformulae in a given type τ without violating type derivations wrt. M.

▶ **Lemma 26.** *Given a λ-term $M \in \operatorname{Long}(\tau)$ and $\pi \in \operatorname{dom}(\tau)$ such that $\tau(\pi) = a$. Let b be a fresh atom. Define τ' by τ replacing for each $\pi' \in \operatorname{dom}(\tau)$ such that $\pi = \pi'$ or $\pi\ R_M\ \pi'$ the subformula a at π' by b. Then $M \in \operatorname{Long}(\tau')$.*

Proof. $M \in \operatorname{Long}(\tau)$ by Lemma 25 implies $R_M \subseteq R_\tau$. Renaming subformulae a in τ at path π and at all paths π' with $\pi\ R_M\ \pi'$ to b preserves $R_M \subseteq R_{\tau'}$. By Lemma 25 we obtain $M \in \operatorname{Long}(\tau')$. ◀

Next, we formulate a necessary condition (cf. Lemma 27) for principal inhabitation.

▶ **Lemma 27.** *Given a type τ let $M \in \operatorname{Long}(\tau)$. If τ is the principal type of M, then $R_\tau = R_M$.*

Proof. Since $M \in \operatorname{Long}(\tau)$, by Lemma 25 we have $R_M \subseteq R_\tau$. Assume there exists $(\pi, \pi') \in R_\tau$ such that $(\pi, \pi') \notin R_M$. Let a be a fresh atom. Define τ' by renaming each subformula of τ in $\{\pi'' \mid \pi'' = \pi \text{ or } \pi''\ R_M\ \pi\}$ to a. Since $\tau(\pi) \in \mathbb{A}$, by Lemma 26 we have $M \in \operatorname{Long}(\tau')$. However, τ' is strictly more general than τ. ◀

Unfortunately, the converse of the above Lemma 27 is not true as illustrated in the following Example 28.

▶ **Example 28.** Consider $M = \lambda x.\lambda y.x$ and $\tau = a \to (b \to c) \to a$. We have $R_M = \{(1, 22)\}^{\equiv} = R_\tau$. However, τ has no normal principal inhabitant.

One could follow the approach of [4] of marking necessary arrows in derivations (requiring further interplay between terms, derivations and types) to close the gap exposed in the above Example 28. At first sight, taking arrow subformulae in derivations into account appears inevitable. Surprisingly, this is not the case. As stated by Lemma 14 in Section 3, certain types (such as $a \to (b \to c) \to a$) have no normal principal inhabitants. Strikingly, formulated as a necessary (and easy to verify) condition ($\star$) in the following Definition 29 we are able to close the mentioned gap without additional constraints on terms or derivations.

▶ **Definition 29** (($\star$)). We say τ satisfies ($\star$), if $\forall \pi \in \operatorname{dom}(\tau).(\tau(\pi 2) \in \mathbb{A} \Rightarrow (\pi 2, \pi 2) \in R_\tau)$.

Intuitively, a given type τ satisfies ($\star$), if τ has no subformula $\sigma \to a$, where a occurs exactly once as a subformula of τ. This coincides with the first property in [5, Proposition 4.3] and is a necessary condition for principal inhabitation, as shown by the following Lemma 30.

▶ **Lemma 30.** *If τ does not satisfy ($\star$), then τ has no normal principal inhabitant.*

Proof. If τ does not satisfy ($\star$), then there exists a path $\pi \in \operatorname{dom}(\tau)$ such that $\tau(\pi 2) \in \mathbb{A}$ and $(\pi 2, \pi 2) \notin R_\tau$. Assume τ has a normal principal inhabitant $M \in \operatorname{Long}(\tau)$ (cf. Lemma 8). By Lemma 25 there exists a derivation $\mathcal{D} \rhd \emptyset \vdash_{R_\tau} M : \varepsilon$. Since $(\pi 2, \pi 2) \notin R_\tau$ the derivation $\mathcal{D}$ contains no judgement of the shape $\Delta \vdash_{R_\tau} L : \pi 2$ for some path environment Δ and term L. Therefore, replacing paths by corresponding subformulae in τ, there exists a derivation $\mathcal{D}' \rhd \emptyset \vdash M : \tau$ such that $a \notin T(\mathcal{D}')$, where $\tau(\pi) = \sigma \to a$ for some type σ. By Lemma 14 the type τ is not the principal type of M, which is a contradiction. ◀

Finally, we formulate a sufficient condition (cf. Lemma 31) for principal inhabitation.

▶ **Lemma 31.** *Given a type τ satisfying $(\star)$ let $M \in \mathrm{Long}(\tau)$. If $R_\tau = R_M$, then τ is the principal type of M.*

Proof. Assume M has a strictly more general principal type τ'. Fix the substitution S such that $S(\tau') = \tau$. By Lemma 7 we have $M \in \mathrm{Long}(\tau')$. Therefore, by Lemma 27 we have $R_M = R_{\tau'}$. We show that $R_\tau \neq R_{\tau'}$.

Case $S : \mathbb{A} \to \mathbb{A}$: There exist π, π' such that $\tau(\pi) = \tau(\pi') \in \mathbb{A}$ and $\tau'(\pi) \neq \tau'(\pi')$. Therefore, $(\pi, \pi') \in R_\tau$ but $(\pi, \pi') \notin R_{\tau'} = R_M$.

Case $S(a) = \sigma_1 \to \ldots \to \sigma_n \to b$ for some $n > 0$ and $a \in \mathrm{ran}(\tau') \cap \mathbb{A}$: Fix any path $\pi \in \mathrm{dom}(\tau')$ such that $\tau'(\pi) = a$. Since $\tau(\pi 2^n) = b$ and $n > 0$, due to $(\star)$ we have $(\pi 2^n, \pi 2^n) \in R_\tau$. However, $\tau'(\pi 2^n)$ is undefined, therefore $(\pi 2^n, \pi 2^n) \notin R_{\tau'} = R_M$. ◀

In sum, the equality $R_M = R_\tau$ characterizes principality in the sense of the following Theorem 32.

▶ **Theorem 32.** *Given a type τ satisfying $(\star)$ and a λ-term $M \in \mathrm{Long}(\tau)$ we have that τ is the principal type of M iff $R_M = R_\tau$.*

Proof. '$\Longrightarrow$' by Lemma 27. '$\Longleftarrow$' by Lemma 31. ◀

Bearing resemblance to the characterization in [4, Proposition 17], the above characterization in Theorem 32 has two benefits. First, it does not require marking of arrows in derivations. Second, it is factored into R_M (uniquely defined by M) and R_τ (uniquely defined by τ). Since by Lemma 25 any long normal inhabitant M of τ satisfies $R_M \subseteq R_\tau$ and the size of R_τ is polynomial in the size of τ, we will only require polynomial space for principal inhabitation in the following Section 5.

5 PSPACE Upper Bound

In this section we develop a polynomial space algorithm to decide principal inhabitation. As mentioned in the introduction, there are three hurdles to overcome to get a polynomial space upper bound.

First, if $\Gamma \vdash M : \tau$ is derivable in the simple type system, then there is a derivation of that judgement which does not contain any judgement $\Gamma \vdash M' : \tau$ such that $M \neq M'$. For principal inhabitation this does not hold as shown in the following Example 33. This issue is solved by taking into account the impact on R_M by corresponding judgements.

▶ **Example 33.** Let $\tau = (a \to a) \to a \to a$. The normal principal inhabitants of τ are exactly the Church numerals greater equal to two, i.e. $\lambda f.\lambda x.f\ (f\ x)$, $\lambda f.\lambda x.f\ (f\ (f\ x))$, ... The corresponding type derivations necessarily assign the type a to the terms $x, f\ x$ and $f\ (f\ x)$ in identical type environments.

Second, term variables with identical types are interchangeable in the simple type system. However, this may violate principality as shown in the following Example 34. This issue is solved using Lemma 21, due to which an identification of x and y is allowed, if x and y are both bound to the same subformula occurrence, i.e. the same path.

▶ **Example 34.** Let $\tau = (a \to a \to a) \to a \to a \to a$, $M = \lambda f.\lambda x.\lambda y.f\ (f\ x\ y)\ (f\ y\ x)$, $M_x = \lambda f.\lambda x.\lambda y.f\ (f\ x\ x)\ (f\ x\ x)$ and $M_y = \lambda f.\lambda x.\lambda y.f\ (f\ y\ y)\ (f\ y\ y)$. Each M, M_x and M_y is an inhabitant of τ. However, only M of the three is a normal principal inhabitant of τ.

Algorithm 1 Algorithm INH deciding existence of normal principal inhabitants

```
 1: Input: simple type $\tau$
 2: Output: accept iff there exists a normal principal inhabitant of $\tau$
 3: if $\neg\Big(\forall\pi \in \mathrm{dom}(\tau).(\tau(\pi 2) \in \mathbb{A} \Rightarrow (\pi 2, \pi 2) \in R_\tau)\Big)$ then
 4:     fail
 5: end if
 6: $R := \mathsf{AUX}(\tau, \emptyset, \varepsilon, \emptyset)$
 7: if $R = R_\tau$ then
 8:     accept
 9: else
10:     fail
11: end if
```

Algorithm 2 Non-deterministic Algorithm AUX

```
 1: Input: simple type $\tau$, set of paths $P$, path $\pi$, relation on paths $R$
 2: Output: updated relation on paths $R$
 3: if $\tau(\pi) = \sigma \to \tau$ then
 4:     return $\mathsf{AUX}(\tau, P \cup \{\pi 1\}, \pi 2, R)$
 5: else if $\tau(\pi) = a$ for some $a$ then
 6:     choose $\pi' \in P$ such that $\tau(\pi' 2^n) = a$ for some $n \geq 0$
 7:     $R := (R \cup \{(\pi' 2^n, \pi)\})^{\equiv}$
 8:     for $i = 1$ to $n$ do
 9:         $R := \mathsf{AUX}(\tau, P, \pi' 2^{i-1} 1, R)$
10:     end for
11: end if
12: return $R$
```

Third, a normal principal inhabitant M of a given type τ may be of exponential size. Therefore, we cannot in polynomial space construct an inhabitant explicitly and then check for principality as in [4, 5]. This issue is solved using the characterization in Theorem 32. Particularly, instead of M it suffices to construct R_M of size at most the size of R_τ, which is polynomial in the size of τ. This key observation allows us to stay in polynomial space.

Given a type τ, the idea behind the following Algorithm 1 to decide principal inhabitation (cf. Problem 1) is as follows. Start by verifying that τ satisfies ($\star$). Continue with the auxiliary Algorithm AUX to construct a relation R corresponding to R_M for some long normal inhabitant M (which is not constructed explicitly). Last, verify $R_M = R_\tau$.

▶ **Example 35.** Let $\tau = ((a \to a) \to a) \to a$ and consider $\mathsf{INH}(\tau)$. Since τ satisfies ($\star$), the condition in line 3 does not trigger a failure.

- Proceed with $\mathsf{AUX}(\tau, \emptyset, \varepsilon, \emptyset)$, which corresponds to inhabitant search of $\tau(\varepsilon) = \tau$.
- Since $\tau(\varepsilon)$ is an arrow type, take the first branch (line 4). This induces a potential inhabitant to be of the shape $\lambda f.N$ for a fresh f and some λ-term N. Proceed with $\mathsf{AUX}(\tau, \{1\}, 2, \emptyset)$, which corresponds to the search for N of type $\tau(2) = a$ in the type environment $\{f : \tau(1) = (a \to a) \to a\}$.
- Since $\tau(2) = a = \tau(12)$, take the second branch (lines 6–10) choosing the path $1 \in P$. This induces $N = f\ L$ for some λ-term L. Proceed with $\mathsf{AUX}(\tau, \{1\}, 11, \{(12, 2)\}^{\equiv})$, searching for L of type $\tau(11) = a \to a$ in the type environment $\{f : \tau(1) = (a \to a) \to a\}$.
- Since $\tau(11) = a \to a$ is an arrow type, take the first branch, i.e. $L = \lambda x.L'$ for a fresh x

and some λ-term L'. Proceed with $\mathsf{AUX}(\tau, \{1, 111\}, 112, \{(12, 2)\}^{\equiv})$, searching for L' of type $\tau(112) = a$ in the type environment $\{f : \tau(1) = (a \to a) \to a, x : \tau(111) = a\}$.

- Since $\tau(112) = a$, take the second branch. There are two options. The first option is to choose the path 111, since $\tau(112) = \tau(111)$. In this case, AUX would return control to INH with the result $R = \{(12, 2), (111, 112)\}^{\equiv}$ and INH would fail. The corresponding run of INH would induce the inhabitant $\lambda f.f\ (\lambda x.x)$, which is not a normal principal inhabitant of τ. The second option is to choose the path 1 since $\tau(112) = \tau(12)$ and proceed with $\mathsf{AUX}(\tau, \{1, 111\}, 11, \{(12, 2), (12, 112)\}^{\equiv})$. Choose the second option.
- Again, $\tau(11) = a \to a$ is an arrow type, take the first branch and proceed with $\mathsf{AUX}(\tau, \{1, 111\}, 112, \{(12, 2), (12, 112)\}^{\equiv})$.
- Again, $\tau(112) = a$, take the second branch, choosing the path 111. After AUX returns $R = \{(12, 2), (12, 112), (111, 112)\}^{\equiv}$ to INH, INH accepts. The corresponding run of INH induces the normal principal inhabitant $\lambda f.f\ (\lambda x.f\ (\lambda y.x))$ (cf. Example 22) of τ.

▶ **Lemma 36** (Soundness of INH). *Given a type τ, if Algorithm 1 accepts, then there exists a normal principal inhabitant of τ.*

Proof. A successful run of Algorithm 1 induces a type derivation $\mathcal{D} \triangleright \emptyset \vdash_{R_M} M : \varepsilon$ for some M. In particular, line 4 in Algorithm AUX induces a λ-abstraction and lines 6–10 in Algorithm AUX induce an application with head variable of type $\tau(\pi')$ and n arguments. By Lemma 21 it suffices to take the variable that is bound to π' and in M is abstracted outermost. Line 3 in in Algorithm INH ensures that τ satisfies $(\star)$ and line 7 ensures that $R_M = R_\tau$. By Theorem 32 the term M is a normal principal inhabitant of τ. ◀

▶ **Lemma 37** (Completeness of INH). *Given a type τ, if there exists a normal principal inhabitant of τ, then there exists an accepting run of Algorithm 1 requiring at most polynomial space in the size of τ.*

Proof. Assume that τ has a normal principal inhabitant M. By Theorem 32 we have that τ satisfies Property $(\star)$ and there exists a normal principal inhabitant $M' \in \mathrm{Long}(\tau)$ such that $\mathcal{D} \triangleright \emptyset \vdash_{R_{M'}} M' : \varepsilon$ and $R_{M'} = R_\tau$. By induction on $\mathcal{D}$ there exists an accepting run $\mathcal{R}$ of Algorithm INH such that for each judgement $\Delta \vdash_{R_{M'}} L : \pi$ in $\mathcal{D}$ the run $\mathcal{R}$ invokes $\mathsf{AUX}(\tau, \mathrm{ran}(\Delta), \pi, R)$ where $R \subseteq R_{M'}$. Therefore, for each side condition $\pi'\ R_{M'}\ \pi''$ in $\mathcal{D}$ the corresponding invocation of AUX in line 7 ensures $\pi'\ R\ \pi''$. Overall, by Theorem 32 we have $R_\tau = R_{M'} = R$ and INH accepts.

Space requirements: The parameters τ, $P \subseteq \mathrm{dom}(\tau)$, $\pi \in \mathrm{dom}(\tau)$ and $R \subseteq \mathrm{dom}(\tau)^2$ are polynomial in the size of τ. Since the above run $\mathcal{R}$ is accepting and there are no side-effects, there exists an accepting run $\mathcal{R}'$ that has no invocations with identical parameters along the recursion branches of AUX. Since P and R are non-decreasing along the recursion branches of AUX, the invocation stack of AUX in $\mathcal{R}'$ is of polynomial depth in size of τ. ◀

▶ **Lemma 38.** *Problem 1 is in* PSPACE.

Proof. By Lemma 36, Lemma 37 and the identity PSPACE=NPSPACE. ◀

6 PSPACE Lower Bound

In this section we establish a PSPACE lower bound for principal inhabitation. Unfortunately, the standard reduction (cf. [12]) from quantified Boolean formulae to inhabitation in the simply typed λ-calculus does not carry over immediately as illustrated by the following Example 39.

▶ **Example 39.** Consider the formula $\varphi = \exists p.\psi$, where $\psi = p \vee \neg p$. By the construction in [12] φ is true iff the type $\sigma = ((a_p \to a_\psi) \to a_\varphi) \to ((a_{\neg p} \to a_\psi) \to a_\varphi) \to (a_p \to a_\psi) \to (a_{\neg p} \to a_\psi) \to a_\varphi$ is inhabited in the simply typed λ-calculus. The only long normal inhabitants of σ are $\lambda x_1.\lambda x_2.\lambda y_1.\lambda y_2.x_1\ (\lambda z.y_1\ z)$ and $\lambda x_1.\lambda x_2.\lambda y_1.\lambda y_2.x_2\ (\lambda z.y_2\ z)$ for both of which σ is not principal. Although φ is true, there is no normal principal inhabitant of σ.

The inherent issue with the standard approach is that existential quantifiers and disjunctions may introduce unnecessary (or even unusable) subformulae. We solve this issue by introducing additional subformulae not affecting inhabitability to secure principal inhabitability.

The construction in [12] shows that the following Problem 2, which is a restriction of inhabitation in the simply typed λ-calculus, is PSPACE-hard.

▶ **Problem 2.** *Given a type $\tau = \sigma_1 \to \ldots \to \sigma_n \to a$ such that $\sigma_i = (b_i^1 \to c_i^1) \to (b_i^2 \to c_i^2) \to d_i$ for some $b_i^1, c_i^1, b_i^2, c_i^2, d_i \in \mathbb{A}$ for $i = 1 \ldots n$, is there a λ-term M such that $\vdash M : \tau$?*

Note that the exact construction in [12] also uses types of the shape $a \to b$ which can be represented by $(c \to a) \to (c \to a) \to b$ where c is fresh.

In the remainder of this section we fix a simple type τ according to Problem 2 with corresponding subformulae $\sigma_1, \ldots, \sigma_n$ and a. Our goal is to construct a type τ^* such that τ is inhabited iff τ^* is principally inhabited. Let $\{a_1, \ldots, a_l\}$ be the set of type atoms in τ and fix k such that $a = a_k$. We construct τ^* (of size polynomial in the size of τ) as follows

$$\begin{aligned}\tau^* =&((a_1 \to \ldots \to a_l \to a) \to a \to a) \to (a_1 \to a_1 \to a_1) \to \ldots \to (a_1 \to a_l \to a_l)\\ &\to (a_2 \to a_1 \to a_1) \to \ldots \to (a_2 \to a_l \to a_l)\\ &\to \ldots \to (a_l \to a_1 \to a_1) \to \ldots \to (a_l \to a_l \to a_l)\\ &\to (a \to a) \to \sigma_1 \to \ldots \to \sigma_n \to a\end{aligned}$$

Since all additional arguments in τ^* are intuitionistically valid formulae, an inhabitant of τ^* induces an inhabitant of τ.

▶ **Lemma 40.** *If $\vdash M : \tau^*$, then $\vdash M\ \underbrace{K^* \ldots K^*}_{1+l^2\ times}\ I : \tau$, where $I = \lambda x.x$ and $K^* = \lambda x.\lambda y.y$.*

It remains to show that if τ is inhabited, then τ^* is principally inhabited.

▶ **Lemma 41.** *If τ has an inhabitant, then τ^* has a normal principal inhabitant.*

Proof. Assume that τ has an inhabitant, then there exists a λ-term N such that $\{w_1 : \sigma_1, \ldots, w_n : \sigma_n\} \vdash N : a$ and N is in long β-normal form. Define the λ-term M^* as follows

$$\begin{aligned}&M^* = \lambda z.\lambda x_1^1 \ldots \lambda x_1^l.\lambda x_2^1 \ldots \lambda x_2^l \ldots \lambda x_l^1 \ldots \lambda x_l^l.\lambda x.\lambda w_1 \ldots \lambda w_n.x\ (z\ F\ (x\ N))\\ &F = \lambda y_1 \ldots \lambda y_l.x_1^k\ G_1^1\ (x\ (x\ y_k))\\ &G_i^j = x_{j+1}^j\ G_i^{j+1}\ (x_i^j\ y_i\ (x_i^j\ y_i\ y_j)) \text{ for } i = 1 \ldots l, j = 1 \ldots l-1\\ &G_i^l = x_1^l\ G_{i+1}^1\ (x_i^l\ y_i\ (x_i^l\ y_i\ y_l)) \text{ for } i = 1 \ldots l-1\\ &G_l^l = x_1^l\ H_1\ (x_l^l\ y_i\ (x_l^l\ y_l\ y_l)) \text{ for } i = 1 \ldots l-1\\ &H_i = x_j^i\ (w_i\ L_{i_1}^{i_2}\ L_{i_3}^{i_4})\ (x_{i+1}^i\ H_{i+1}\ y_i) \text{ for } i = 1 \ldots l-1\\ &\ \text{where } \sigma_i = (a_{i_1} \to a_{i_2}) \to (a_{i_3} \to a_{i_4}) \to a_j\\ &H_l = x_j^l\ (w_l\ L_{i_1}^{i_2}\ L_{i_3}^{i_4})\ y_l \text{ where } \sigma_l = (a_{i_1} \to a_{i_2}) \to (a_{i_3} \to a_{i_4}) \to a_j\\ &L_i^j = \lambda t.x_i^j\ t\ y_j \text{ for } i = 1 \ldots l, j = 1 \ldots l\end{aligned}$$

We have $M^* \in \mathrm{Long}(\tau^*)$. Types of key subterms are outlined in the following overview.

$x : a_k \to a_k$	$x_i^j : a_i \to a_j \to a_j$ for $i = 1 \dots l, j = 1 \dots l$
$y_i : a_i$ for $i = 1 \dots l$	$z : (a_1 \to \dots \to a_l \to a_k) \to a_k \to a_k$
$w_i : \sigma_i$ for $i = 1 \dots n$	
$F : a_1 \to \dots \to a_l \to a$	$G_i^j : a_j$ for $i = 1 \dots l, j = 1 \dots l$
$H_i : a_i$ for $i = 1 \dots l$	$L_i^j : a_i \to a_j$

We use Theorem 32 to show that τ^* is the principal type of M^* by showing $R_{M^*} = R_{\tau^*}$. Since $M^* \in \mathrm{Long}(\tau^*)$, by Lemma 25 we have $R_{M^*} \subseteq R_{\tau^*}$. To obtain $R_{M^*} \supseteq R_{\tau^*}$, we show that for each path $\pi \in \mathrm{dom}(\tau^*)$ if $\tau^*(\pi) = a_i$, then $(\pi, 112^{i-1}1) \in R_{M^*}$. Therefore, if $(\pi, \pi') \in R_{\tau^*}$, then $\pi \; R_{M^*} \; 112^{i-1}1 \; R_{M^*} \; \pi'$ and $(\pi, \pi') \in R_{M^*}$ by transitivity. Let $\mathcal{D}$ be the derivation of $\vdash_{R_{M^*}} R_{M^*} : \varepsilon$.

Let $\pi_i^j = 22^{l \cdot (i-1)+(j-1)}1$ for $i, j = 1 \dots l$. We have $\tau^*(\pi_i^j) = a_i \to a_j \to a_j$. Let $\bar{\pi}_i = 112^{i-1}1$ for $i = 1 \dots l$. We have $\tau^*(\bar{\pi}_i) = a_i$. Let $\hat{\pi}_i = 2^{1+l^2+i}1$ for $i = 1 \dots n$. We have $\tau^*(\hat{\pi}_i) = \sigma_i$. In $\mathcal{D}$ the paths π_i^j are assigned to x_i^j, the paths $\bar{\pi}_i$ are assigned to y_i and the paths $\hat{\pi}_i$ are assigned to w_i.

For each $i, j = 1 \dots l$ the term M^* contains the subterm G_i^j. Leaving out some details by $[\dots]$, $\mathcal{D}$ contains the judgement $[\dots], x_i^j : \pi_i^j, y_i : \bar{\pi}_i, y_j : \bar{\pi}_j \vdash_{R_{M^*}} x_i^j \; y_i \; (x_i^j \; y_i \; y_j) : [\dots]$. The corresponding subderivation therefore entails $(\pi_i^j 1, \bar{\pi}_i) \in R_{M^*}$, $(\pi_i^j 21, \pi_i^j 22) \in R_{M^*}$ and $(\pi_i^j 21, \bar{\pi}_j) \in R_{M^*}$.

We proceed similarly with the remaining subformulae of τ^*. The most crucial subformulae are at paths $\hat{\pi}_i$ that correspond to σ_i for $i = 1 \dots n$. For each $i = 1 \dots n$ the term M^* contains the subterm H_i. Consider $\sigma_i = (a_{i_1} \to a_{i_2}) \to (a_{i_3} \to a_{i_4}) \to a_j$. Due to the subterm $x_j^i \; (w_i \; L_{i_1}^{i_2} \; L_{i_3}^{i_4}) \; [\dots]$ the corresponding subderivation entails $(\pi_j^i 1, \hat{\pi}_i 22) \in R_{M^*}$, therefore $(\bar{\pi}_j, \hat{\pi}_i 22) \in R_{M^*}$. Due to the subterm $w_i \; L_{i_1}^{i_2} \; L_{i_3}^{i_4}$ the corresponding subderivation entails $(\hat{\pi}_i 12, \pi_{i_1}^{i_2} 22) \in R_{M^*}$, $(\hat{\pi}_i 11, \pi_{i_1}^{i_2} 1) \in R_{M^*}$, $(\hat{\pi}_i 212, \pi_{i_3}^{i_4} 22) \in R_{M^*}$ and $(\hat{\pi}_i 211, \pi_{i_3}^{i_4} 1) \in R_{M^*}$. ◀

▶ **Lemma 42.** *Problem 1 is* PSPACE*-hard.*

Proof. By reduction from Problem 2 using Lemma 41 and Lemma 40. ◀

Finally, we conjecture that the construction in the proof of Lemma 41 can be generalized to arbitrary simple types (not restricted to the shape in Problem 2). The main idea is, instead of using the subformula $(a_1 \to \dots \to a_l \to a) \to a \to a$, to use the subformula $(\rho_1 \to \dots \rho_m \to a) \to a \to a$, where $\{\rho_1, \dots, \rho_m\}$ is the set of subformulae in the given type. Although the conjectured generalization is not necessary for the lower bound proof, it may be of systematic interest as a 'principal closure' of simple types.

7 Conclusion and Future Work

We have studied the problem of principal inhabitation in the simply typed λ-calculus, showing that the problem is PSPACE-complete. We believe that the techniques employed here (including filtration and path relations) condense the algorithmic essence of the problem. The presented polynomial space bounded algorithm should be a good starting point for further algorithm engineering for efficiency, relying on the subformula calculus and the logic of path relations it gives rise to.

In future work we intend to apply the algorithm in the context of type-based and combinatory logic synthesis [9, 6, 3]. In this context, we plan to add a facility for synthesizing normal principal inhabitants as combinators of general applicability in component repositories. Further, when types and corresponding terms are inductively defined, the provided characterization of normal principal inhabitants may prove useful in mechanized certification of principality by proof assistants. Finally, the presented approach could be useful to inspect principal inhabitation in the simply typed λI-calculus for which inhabitation is 2-EXPTIME-complete [10].

References

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

2 C.-B. Ben-Yelles. *Type-assignment in the lambda-calculus; syntax and semantics*. PhD thesis, Mathematics Dept., University of Wales Swansea, UK, 1979.

3 Jan Bessai, Andrej Dudenhefner, Boris Düdder, Moritz Martens, and Jakob Rehof. Combinatory Logic Synthesizer. In *Leveraging Applications of Formal Methods, Verification and Validation, 6th International Symposium ISoLA 2014, Corfu, Greece, October 8-11, 2014*, pages 26–40, 2014. doi:10.1007/978-3-662-45234-9_3.

4 Sabine Broda and Luís Damas. Counting a Type's Principal Inhabitants. In *TLCA'99*, pages 69–82, 1999. doi:10.1007/3-540-48959-2_7.

5 Sabine Broda and Luís Damas. On long normal inhabitants of a type. *J. Log. Comput.*, 15(3):353–390, 2005. doi:10.1093/logcom/exi016.

6 Boris Düdder, Moritz Martens, and Jakob Rehof. Staged Composition Synthesis. In *ESOP 2014, Proceedings*, pages 67–86, 2014. doi:10.1007/978-3-642-54833-8_5.

7 Andrej Dudenhefner and Jakob Rehof. Typability in Bounded Dimension. In *LICS 2017, Proceedings of the 32nd ACM/IEEE Symposium on Logic in Computer Science, Reykjavik, Iceland, June*, 2017.

8 J. Roger Hindley. *Basic Simple Type Theory*. Cambridge Tracts in Theoretical Computer Science, vol. 42, Cambridge University Press, 2008.

9 Jakob Rehof. Towards Combinatory Logic Synthesis. In *BEAT 2013, 1st International Workshop on Behavioural Types*. ACM, 2013.

10 Sylvain Schmitz. Implicational relevance logic is 2-exptime-complete. *J. Symb. Log.*, 81(2):641–661, 2016. doi:10.1017/jsl.2015.7.

11 Richard Statman. Intuitionistic Propositional Logic is Polynomial-space Complete. *Theoretical Computer Science*, 9:67–72, 1979. doi:10.1016/0304-3975(79)90006-9.

12 P. Urzyczyn. Inhabitation in Typed Lambda-Calculi (A Syntactic Approach). In *TLCA'97, Typed Lambda Calculi and Applications, Proceedings*, volume 1210 of *LNCS*, pages 373–389. Springer, 1997. doi:10.1007/3-540-62688-3-47.