Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 344 (2019) 25–45

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Pre-grammars and Inhabitation for a Subset of Rank 2 Intersection Types

Sandra Alves[1,3] Sabine Broda[2,4]

CMUP & DCC/FCUP
University of Porto, Portugal

Abstract

In this paper, we identify a subset of types in the rank 2 intersection type system, where types do not contain positive occurrences of intersections. We extend the notion of pre-grammar of a type and address the type-inhabitation problem for types in this subset, as well as their intersections.

Keywords: Intersection types, Pre-grammars, Inhabitation

1 Introduction

The inhabitation of types in the lambda calculus has been extensively studied over the years and is of manifest importance to different fields, which range from more foundational aspects to applications in program language theory, where types figure as a main tool for specifying the behaviour of programs. In proof theory algorithms for deciding type-inhabitation can be used for indirectly decide provability. For simple types the problem has been addressed, throughout the years, using a variety of techniques [4,8,21,20,19,6]. Systems with intersection types [9], extend simple types with an intersection operator $\cap$, which increases their expressive power substantially. Intersection types have been used in abstract interpretation, model check-

[1] Partially funded by Project "TEC4Growth - Pervasive Intelligence, Enhancers and Proofs of Concept with Industrial Impact/NORTE-01-0145-FEDER-000020" is financed by the North Portugal Regional Operational Programme (NORTE 2020A), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF).

[2] Partially funded by CMUP (UID/MAT/00144/2013), which is funded by FCT (Portugal) with national (MEC) and european structural funds through the programs FEDER, under the partnership agreement PT2020. Partially funded by FCT within project Elven POCI-01-0145-FEDER-016844, Project 9471 - Reforar a Investigao, o Desenvolvimento Tecnolgico e a Inovao (Project 9471-RIDTI), by project PTDC/EEI-CTP/3506/2014, and by Fundo Comunitrio Europeu FEDER.

[3] Email: sandra@dcc.fc.up.pt

[4] Email: sbb@dcc.fc.up.pt

https://doi.org/10.1016/j.entcs.2019.07.003

ing, process calculi, dependent types, to mention a few. Furthermore intersection types are an increasingly important feature in modern programming languages, such as TypeScript, Flow and Scala. For an overview on intersection types and some of their applications we refer to [25,10,5,14,1]. Type inhabitation for finite rank intersection types was proved undecidable for ranks over 2 [22,23], and decidable for rank 2 [17], although exponentially hard. In recent years, the problem of type-inhabitation has been further explored for bounded ranks of non-idempotent intersection types [7,12,11].

More recently [3], type inhabitation problems for simple types were addressed using an alternative representation for types, called the pre-grammar of the type. Pre-grammars capture the underlying structure of types, providing a common framework for addressing a number of type-inhabitation related problems such as type-checking, counting and generation of inhabitants. Furthermore, a scheme for a decision algorithm was given, which was instantiated to decide emptiness, counting and principal inhabitation. Closure properties were studied and the notion of pre-grammar and its corresponding methods were extended to sums of intersections of simple types. In this paper, we further explore this formalism for types with intersections, by extending the notion of pre-grammar and addressing the emptiness problem for a subset of rank 2 intersection types, that we denote by $\mathcal{T}_2^-$. Types in $\mathcal{T}_2^-$ do not contain positive occurrences of intersections, which intuitively means that the result type of a term used at a function position will not be an intersection. We point out that this is not such a strong restriction on rank 2 types. Actually, systems addressing type assignment for rank 2 [26,15,16] consider even stronger restrictions of the rank 2 system, when compared to the original notion by Leivant [18]. In fact, the same type of restriction is imposed on the strict type assignment system [24], and the essential type assignment system [25], both of which eliminate superfluous steps, leading to derivations that are syntax directed yet still semantically complete. Note that, when restricted to rank 2, the strict type system is a proper subset of $\mathcal{T}_2^-$, since it forbids the occurrences of intersections to the right of any arrow, and any positive intersection to the left of an arrow is only possible at a rank greater or equal than 3. For example, $(a \cap (a \to (a \cap a)) \to a$ is a type in $\mathcal{T}_2^-$ (both occurrences of intersections are negative), but is not a strict rank 2 type (the inner occurrence of an intersection is to the right of an arrow).

From the notion of pre-grammar, the methods in [3] are extended smoothly for $\mathcal{T}_2^-$-types. In fact, when applied to types containing no intersection, the extended definition of pre-grammar will coincide with the one for simple types given in [3]. This nice property suggests that we can address other problems related to type inhabitation, for more expressive types, using the formalism of pre-grammars. We will restrict our methods and definitions to terms in normal form, since each inhabited type in the rank 2 type system, and therefore in any of its restrictions, has a normal inhabitant (intersection types characterize exactly the set of strongly normalising λ-terms). Furthermore, we consider types modulo commutativity, idempotence and associativity, always identifying maximal subtypes containing intersections, that is, in any subtype of the form $\tau_1 \cap \cdots \cap \tau_n$, none of the τ_i is itself an intersection.

The rest of the paper is structured as follows. In the next section we introduce some preliminary notions. In Section 3, we present the set of $\mathcal{T}_2^-$-types and prove some properties. In Section 4, we present the notion of pre-grammar for $\mathcal{T}_2^-$-types. Using the pre-grammar representation, in Section 5 we define a decision procedure for $\mathcal{T}_2^-$-types and in Section 6, we show how to extend the decision procedure to intersections of $\mathcal{T}_2^-$-types. Finally, in Section 7, we draw some conclusions and highlight some future work.

2 Preliminaries

We assume familiarity with the simply typed λ-calculus à la Curry [13]. We denote type variables (atoms) by $a, b, c, \ldots$ and arbitrary types (simple or of higher rank) by lower-case Greek letters.

Definition 2.1 The set of intersection types $\mathcal{T}_\cap$ is inductively defined as follows.

- Every type variable is a(n) (intersection) type;
- if α and β are types, then $\alpha \to \beta$ is a type;
- if $\alpha_1, \ldots, \alpha_n$ are types and $n \geq 2$, then $\alpha_1 \cap \cdots \cap \alpha_n$ is a type.

Note that the set of simple types, which we denote by $\mathcal{T}$, is exactly the set of types in $\mathcal{T}_\cap$, that contain no occurrence of the intersection operator. We assume that $\cap$ is associative, commutative, idempotent, and that it binds stronger than $\to$. As usual, the operator $\to$ associates to the right. For instance, we write $\alpha \cap \beta \cap \delta \to \gamma \to \delta$ instead of $((\alpha \cap \beta) \cap \gamma) \to (\gamma \to \delta)$. In a finite rank intersection type system, intersections on types are allowed only up to a certain depth. The notion of rank was defined for intersection types in [18].

Definition 2.2 The *rank* of a type in $\mathcal{T}_\cap$ is defined by the following rules.

- $rank(\alpha) = 0$, if $\alpha \in \mathcal{T}$;
- $rank(\alpha \to \beta) = max(1 + rank(\alpha), rank(\beta))$, when $rank(\alpha) > 0$ or $rank(\beta) > 0$;
- $rank(\alpha_1 \cap \cdots \cap \alpha_n) = max(1, rank(\alpha_1), \ldots, rank(\alpha_n))$ for $n \geq 2$.

Given $r \in \mathbb{N}$, we denote by $\mathcal{T}_r$ the set of intersection types of rank $\leq r$.

We denote λ-terms by $M, N, \ldots$, which are built from an infinite countable set of term variables $\mathcal{V}$. Unless stated otherwise, we identify terms modulo α-equivalence. For type assignment we consider a system equivalent to the one presented in [17]. A *context* is a finite set Γ of declarations $x : \sigma$, where $x \in \mathcal{V}$ and $\sigma \in \mathcal{T}_\cap$, such that all term variables occurring in Γ are distinct from each other. The set of term variables occurring in Γ is denoted by $\mathsf{Subj}(\Gamma)$. The union of contexts is *consistent*, if it does not contain different type declarations for the same term variable.

Definition 2.3 We say that type $\theta \in \mathcal{T}_\cap$ can be assigned to term M in context Γ, and write $\Gamma \vdash M : \theta$, if and only if this formula can be obtained by applying the

rules below a finite number of times.

$$\frac{}{\Gamma \vdash x : \tau}\ \mathsf{var}\ \ (\text{if } x:\tau \in \Gamma) \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}\ \mathrm{E}{\to} \qquad \frac{\Gamma \cup \{x:\sigma\} \vdash N : \tau}{\Gamma \vdash \lambda x.N : \sigma \to \tau}\ \mathrm{I}{\to}$$

$$\frac{\Gamma \vdash M : \tau_1 \cap \cdots \cap \tau_n}{\Gamma \vdash M : \tau_i \ \ (1 \le i \le n)}\ \mathrm{E}{\cap} \qquad \frac{\Gamma \vdash M : \tau_i \ \ (1 \le i \le n, n > 1)}{\Gamma \vdash M : \tau_1 \cap \cdots \cap \tau_n}\ \mathrm{I}{\cap}$$

If $\Gamma = \emptyset$, then we also write $\vdash M : \theta$ instead of $\Gamma \vdash M : \theta$ and say that M is an *inhabitant* of type θ. The set of all β-normal inhabitants of θ is denoted by $\mathsf{Nhabs}(\theta)$.

It is easy to verify that $\Gamma \vdash M : \theta$ implies that the set of term variables in Γ contains the set of free variables in M, i.e. $\mathsf{Subj}(\Gamma) \supseteq \mathsf{FV}(M)$. A derivation of a formula $\Gamma \vdash M : \theta$ can be represented as a derivation tree Π, in which all nodes are labelled by formulas, such that $\Gamma \vdash M : \theta$ is the root of Π, every internal node is obtained from its children by one of the rules (E→), (I→), (E∩) or (I∩). Every leaf is labelled with an instance of (var).

Example 2.4 Consider type

$$\tau = (((o \to o) \to o) \cap (o \to ((o \to ((o \to o) \cap o)) \cap o))) \to o,$$

let $\tau_{17} = ((o \to o) \to o) \cap (o \to \tau_{15})$ and $\tau_{15} = (o \to ((o \to o) \cap o)) \cap o$. Note that $\tau \in \mathcal{T}_2$, while $\tau_{15}, \tau_{17} \in \mathcal{T}_1$. The following is a derivation tree of $\vdash \lambda x.x(\lambda y.xyy) : \tau$.

$$\dfrac{\dfrac{\dfrac{}{x:\tau_{17} \vdash x:\tau_{17}}\,\mathsf{var}}{x:\tau_{17} \vdash x:(o\to o)\to o}\,\mathrm{E}{\cap} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x:\tau_{17},y:o \vdash x:\tau_{17}}\,\mathsf{var}}{x:\tau_{17},y:o \vdash x:o\to\tau_{15}}\,\mathrm{E}{\cap} \qquad \dfrac{}{x:\tau_{17},y:o \vdash y:o}\,\mathsf{var}}{x:\tau_{17},y:o \vdash xy:\tau_{15}}\,\mathrm{E}{\to}}{x:\tau_{17},y:o \vdash xy:o\to((o\to o)\cap o)}\,\mathrm{E}{\cap} \qquad \dfrac{}{x:\tau_{17},y:o \vdash y:o}\,\mathsf{var}}{x:\tau_{17},y:o \vdash xyy:(o\to o)\cap o}\,\mathrm{E}{\to}}{x:\tau_{17},y:o \vdash xyy:o}\,\mathrm{E}{\cap}}{x:\tau_{17} \vdash \lambda y.xyy:o\to o}\,\mathrm{I}{\to}}{x:\tau_{17} \vdash x(\lambda y.xyy):o}\,\mathrm{E}{\to}}{\vdash \lambda x.x(\lambda y.xyy):\tau}\,\mathrm{I}{\to}$$

Definition 2.5 The polarity of occurrences of subtypes in a type is defined as follows.

- θ is a positive occurrence of a subtype in θ;
- if $\sigma \to \tau$ is a positive (resp. negative) occurrence of a subtype in θ, then that occurrence of σ is a negative (resp. positive) and that occurrence of τ is a positive (resp. negative) subtype in θ;
- if $\tau = \tau_1 \cap \cdots \cap \tau_n$ is a positive (resp. negative) subtype in θ, where $n \ge 2$, then $\tau_1, \ldots, \tau_n$ are positive (resp. negative) subtypes in θ. We also call τ an intersection in θ of arity n. Subtypes $\tau_1, \ldots, \tau_n$ are called the components of intersection τ.

Following the notation in [13], we will on occasions write $\underline{o}$ when referring to a particular occurrence of an object o. Every type τ is either an intersection, or can be uniquely written as $\tau = \tau_1 \to \cdots \to \tau_n \to \theta$ $(n \ge 0)$, where θ is a type variable or an intersection $\theta_1 \cap \cdots \cap \theta_m$ $(m \ge 2)$. Subtype θ is called the *tail* of τ and denoted by $\mathsf{tail}(\tau)$. If $n \ge 1$, then $\tau_1, \ldots, \tau_n$ are called the *arguments* of τ.

An occurrence $\underline{\sigma}$ in τ is called a *negative subpremise* of τ iff it is the argument of a positive occurrence of a subtype in τ.

Example 2.6 Formula $\theta = a \cap (b \to c \cap d) \cap (e \cap f \to g) \to h \cap i$ contains four subtypes that are intersections. $a \cap (b \to c \cap d) \cap (e \cap f \to g)$ is of arity three and is a negative subpremise. $c \cap d$ is negative, but no subpremise. Both, $e \cap f$ and $h \cap i$ are positive occurrences. Note that we do not regard at $a \cap (b \to c \cap d)$ as an intersection in θ, since we always consider the maximal intersection possible.

3 $\mathcal{T}_2^-$-Types

From now on, we will focus on inhabitation of rank 2 types that have no positive occurrences of intersections. We denote the set of types with this property by $\mathcal{T}_2^-$. Note that this is no major restriction on the set $\mathcal{T}_2$ of rank 2 types. In fact, it follows from Definition 2.2 that each type $\theta \in \mathcal{T}_2$ is either an intersection $\theta_1 \cap \cdots \cap \theta_n$ with $\theta_1, \ldots, \theta_n \in \mathcal{T}_2$ and $n \geq 2$, or of the form $\tau_1 \to \cdots \to \tau_m \to \theta'$ $(m \geq 0)$, where $\tau_i \in \mathcal{T}_1$ for $1 \leq i \leq m$ and θ' is either an intersection of rank 2 types (recall that we consider intersections to have at least two components) or a type variable. On the other hand, $\tau_i \in \mathcal{T}_1$ implies that there is no negative occurrence of an intersection in τ_i, and consequently τ_i introduces no positive occurrence of an intersection in θ. So the set $\mathcal{T}_2^-$ includes all types of this last form, as long as θ' is a type variable. Furthermore, in Section 6 we will show how our method can be extended to intersections of types in $\mathcal{T}_2^-$.

In Definition 3.3 we give a set of inference rules for terms in β-normal form, which for types in $\mathcal{T}_2^-$ is equivalent to the set of rules in Definition 2.3, in the sense of Proposition 3.4. This suffices for the purpose of type inhabitation, and the system has the property that, for every formula $\Gamma \vdash_2 N : \tau$ that can be derived, one has $\mathsf{Subj}(\Gamma) = \mathsf{FV}(N)$. As stated before, considering only normal terms is sufficient, since every inhabited type has a normal inhabitant. The following, straightforward characterisation of types in $\mathcal{T}_2^-$ and in $\mathcal{T}_1$ (rank 1) will be used to prove the adequateness of our set of inference rules.

Lemma 3.1 *One has $\theta \in \mathcal{T}_2^-$ if and only if $\theta = \tau_1 \to \cdots \to \tau_n \to a$, where $\tau_i \in \mathcal{T}_1$, for $1 \leq i \leq n$ and $n \geq 0$, and a is a type variable. In particular, any type in $\mathcal{T}_2^-$ is either a type variable or of the form $\tau \to \theta'$, where $\tau \in \mathcal{T}_1$ and $\theta' \in \mathcal{T}_2^-$. Furthermore, $\tau \in \mathcal{T}_1$ if and only if τ is either (i) a type variable or of the form $\sigma \to \tau'$, where $\sigma \in \mathcal{T}$ (simple type) and $\tau' \in \mathcal{T}_1$, or (ii) it is an intersection of types of the form described in (i).*

Every β-normal λ-term is of the form $\lambda x.N$ or $xN_1 \cdots N_s$, where $N, N_1, \ldots, N_s$ are in normal form and $s \geq 0$. Similar to the case of the simple type system, the general structure of a derivation tree Π of a formula $\Gamma \vdash M : \theta$, where M is in β-normal form and $\theta \in \mathcal{T}_\cap$, is determined by the structure of M.

Definition 3.2 Consider a derivation tree Π for some formula $\Gamma \vdash M : \theta$. Let $\mathtt{n}$ be a node in Π with label $\Gamma' \vdash N : \tau$, such that τ is not an intersection. Consider

the maximal subtree $\Pi_{\mathtt{n}}$ of Π, rooted in $\mathtt{n}$ such that each node results from another node in $\Pi_{\mathtt{n}}$ by rule (E$\cap$), or results from two other nodes in $\Pi_{\mathtt{n}}$ by rule (I$\cap$).

Since τ is not an intersection, and in $\Pi_{\mathtt{n}}$ only rules (I$\cap$) and (E$\cap$) are used, there must be a leaf $\mathtt{n}'$ of $\Pi_{\mathtt{n}}$ labelled with a formula of the form $\Gamma' \vdash N : \tau_1 \cap \cdots \cap \tau_n \cap \tau$, with $n \geq 0$. Consider the[5] tree $\Pi'_{\mathtt{n}}$ with (exactly one) leaf $\mathtt{n}'$, labelled with $\Gamma' \vdash N : \tau_1 \cap \cdots \cap \tau_n \cap \tau$, from which $\Gamma' \vdash N : \tau$ is derived by (E$\cap$) in case $n > 0$. Otherwise, $\Pi'_{\mathtt{n}}$ consists just of $\mathtt{n}'$. This means that $\Pi'_{\mathtt{n}}$ consists always of either of one or two nodes. We denote by $\Pi[\mathtt{n}]$ the tree obtained from Π by replacing the complete subtree rooted in node $\mathtt{n}$ by $\Pi'_{\mathtt{n}}$, attaching to the (unique) leaf node in $\Pi'_{\mathtt{n}}$ the subtree, that in Π was rooted in $\mathtt{n}'$.

It is easy to see that, if Π is a derivation tree for some formula $\Gamma \vdash M : \theta$ and $\mathtt{n}$ a node with label $\Gamma' \vdash N : \tau$, such that τ is not an intersection, then $\Pi[\mathtt{n}]$ is still a derivation tree for $\Gamma \vdash M : \theta$.

Definition 3.3 We say that a type θ can be assigned to term M in context Γ, and write $\Gamma \vdash_2 M : \theta$, if and only if this formula can be obtained by applying the rules below a finite number of times. For the parameters in these rules we suppose that $s \geq 0$, $n \geq 1$ and $i \in \{1, \ldots, n\}$.

$$\frac{}{\{x : \tau_1 \cap \cdots \cap \tau_n\} \vdash_2 x : \tau_i}\ \mathsf{var}_2 \qquad \frac{\Gamma \vdash_2 N : \tau}{\Gamma \setminus \{x : \sigma\} \vdash_2 \lambda x.N : \sigma \to \tau}\ \mathrm{I}{\to}_2 \ \text{ if } \Gamma \cup \{x : \sigma\} \text{ is consistent}$$

$$\frac{\Gamma_1 \vdash_2 xN_1 \cdots N_s : \sigma \to \tau_1 \cap \cdots \cap \tau_n \quad \Gamma_2 \vdash_2 N_{s+1} : \sigma}{\Gamma_1 \cup \Gamma_2 \vdash_2 xN_1 \cdots N_s N_{s+1} : \tau_i}\ \mathrm{E}{\to}_2 \ \text{ if } \Gamma_1 \cup \Gamma_2 \text{ is consistent}$$

Note 1 *Consider a β-normal form M and a derivation tree Π of a formula $\Gamma \vdash_2 M : \theta$. It is important to realise that one might have $\theta \notin \mathcal{T}_2^-$. Furthermore, even if $\theta \in \mathcal{T}_2^-$, then it is possible that there are formulas $\Gamma' \vdash_2 N : \tau$ in Π with $\tau \notin \mathcal{T}_2^-$.*

Proposition 3.4 *Consider a β-normal form M and $\theta \in \mathcal{T}_2^-$. Then, $\vdash M : \theta$ if and only if $\vdash_2 M : \theta$.*

Proof. We prove the two directions separately.

($\Leftarrow$) We will prove, by structural induction on M, the following, more general result: if M is in β-nf and $\Gamma \vdash_2 M : \theta$, where $\theta \in \mathcal{T}_\cap$, then $\Gamma \vdash M : \theta$. First note that, in $\vdash$, the following property holds and can easily be proved by induction: if $\Gamma \vdash M : \theta$, and $\Gamma \cup \{x : \tau\}$ is consistent, then $\Gamma \cup \{x : \tau\} \vdash M : \theta$.

- If $M = x$, then $\{x : \tau_1 \cap \cdots \cap \tau_n\} \vdash_2 x : \tau_i$. Therefore, by ($\mathsf{var}$) we have $\{x : \tau_1 \cap \cdots \cap \tau_n\} \vdash x : \tau_1 \cap \cdots \cap \tau_n$, from which we obtain $\{x : \tau_1 \cap \cdots \cap \tau_n\} \vdash x : \tau_i$, by (E$\cap$).
- If $M = \lambda x.N$, then $\Gamma \setminus \{x : \sigma\} \vdash_2 \lambda x.N : \sigma \to \tau$ is a consequence of $\Gamma \vdash_2 N : \tau$ and $\Gamma \cup \{x : \sigma\}$ being consistent. By the induction hypothesis $\Gamma \vdash N : \tau$, and by the property above $\Gamma \cup \{x : \sigma\} \vdash N : \tau$. But $\Gamma \cup \{x : \sigma\} = (\Gamma \setminus \{x : \sigma\}) \cup \{x : \sigma\}$. Thus, $\Gamma \setminus \{x : \sigma\} \vdash \lambda x.N : \sigma \to \tau$ follows by (I$\to$).

[5] Given some strategy for choosing that leaf, e.g. left-most, this tree is uniquely defined.

- If $M = xN_1 \cdots N_s N_{s+1}$, then $\Gamma_1 \cup \Gamma_2 \vdash_2 xN_1 \cdots N_s N_{s+1} : \tau_i$ is a consequence of $\Gamma_1 \vdash_2 xN_1 \cdots N_s : \sigma \to \tau_1 \cap \cdots \cap \tau_n$, $\Gamma_2 \vdash_2 N_{s+1} : \sigma$ and $\Gamma_1 \cup \Gamma_2$ is consistent. By the induction hypothesis $\Gamma_1 \vdash xN_1 \cdots N_s : \sigma \to \tau_1 \cap \cdots \cap \tau_n$ and $\Gamma_2 \vdash N_{s+1} : \sigma$. Since $\Gamma_1 \cup \Gamma_2$ is consistent then, by the property above, $\Gamma_1 \cup \Gamma_2 \vdash xN_1 \cdots N_s : \sigma \to \tau_1 \cap \cdots \cap \tau_n$ and $\Gamma_1 \cup \Gamma_2 \vdash N_{s+1} : \sigma$. By (E→), $\Gamma_1 \cup \Gamma_2 \vdash xN_1 \cdots N_s N_{s+1} : \tau_1 \cap \cdots \cap \tau_n$, from which follows $\Gamma_1 \cup \Gamma_2 \vdash xN_1 \cdots N_s N_{s+1} : \tau_i$, by (E∩).

(⇒) Consider a derivation tree Π of $\vdash M : \theta$, where $\theta \in \mathcal{T}_2^-$. In the first part of the proof we will construct from Π a derivation tree Π' of $\vdash M : \theta$, for which the following hold.

(a) If a node in Π' is labelled with $\Gamma \vdash N : \tau$, then $x : \sigma \in \Gamma$ implies $\sigma \in \mathcal{T}_1$.

(b) Every node $\Gamma \vdash \lambda x.N : \tau$ in Π' is such that $\tau = \tau_1 \to \tau_2 \in \mathcal{T}_2^-$ and results from $\Gamma \cup \{x : \tau_1\} \vdash N : \tau_2$ by (I→), where $\tau_1 \in \mathcal{T}_1$ and $\tau_2 \in \mathcal{T}_2^-$.

(c) Every node $\Gamma \vdash x : \tau$ in Π', where $\tau \in \mathcal{T}_2^-$ or τ is of rank 1 but is not an intersection, results from $\Gamma \vdash x : \tau_1 \cap \cdots \cap \tau_n \cap \tau$ by rule (var), followed by an application of rule (E∩) in case $n > 0$.

(d) Every node $\Gamma \vdash xN_1 \cdots N_k : \tau$ in Π', where $\tau \in \mathcal{T}_2^-$ or τ is of rank 1 but is not an intersection ($k \geq 1$), results from two formulas $\Gamma \vdash xN_1 \cdots N_{k-1} : \sigma \to \tau_1 \cap \cdots \cap \tau_n \cap \tau$ and $\Gamma \vdash N_k : \sigma$, where $\sigma \in \mathcal{T} \subseteq \mathcal{T}_2^-$ and $\tau_1, \cdots, \tau_n \in \mathcal{T}_1$ (and consequently $\sigma \to \tau_1 \cap \cdots \cap \tau_n \cap \tau \in \mathcal{T}_1$), by rule (E→), followed by an application of rule (E∩) in case $n > 0$.

Note that, in particular, Π' contains no application of rule (I∩). We construct Π' bottom-up from Π, applying recursively the transformation method, described below, to nodes labelled with formulas $\Gamma \vdash N : \tau$, which satisfy the following premises:

(i) all types in Γ are of rank 1;

(ii) $\tau \in \mathcal{T}_2^-$, or N is of the form $xN_1 \cdots N_k$ and τ is of rank 1 but is not an intersection, for some $k \geq 0$.

For the root, (i) is true and also $\theta \in \mathcal{T}_2^-$ (thus satisfying (ii)). Since M is closed it has to be of the form $\lambda x.N$. However, the construction is performed bottom up and we want to consider any node $\mathtt{n}$ labelled with a formula of the form $\Gamma \vdash \lambda x.N : \tau$ satisfying the premises. In particular $\tau \in \mathcal{T}_2^-$. Consider the maximal subtree $\Pi_{\mathtt{n}}$, rooted in node $\mathtt{n}$, as defined in Definition 3.2, as well as the corresponding tree $\Pi'_{\mathtt{n}}$, whose leaf $\mathtt{n}'$ is labelled with a formula $\Gamma \vdash \lambda x.N : \tau_1 \cap \cdots \cap \tau_n \cap \tau$. Since $\Pi_{\mathtt{n}}$ is maximal this formula must have been derived by rule (I→), and we conclude that $n = 0$. Thus, $\Pi[\mathtt{n}]$ is obtained from Π by replacement of the entire subtree rooted in $\mathtt{n}$ by the subtree rooted in node $\mathtt{n}'$, which has also label $\Gamma \vdash \lambda x.N : \tau$. Since this formula was derived by rule (I→), τ has to be of the form $\tau_1 \to \tau_2$ and $\Gamma \vdash \lambda x.N : \tau_1 \to \tau_2$, was derived from formula $\Gamma \cup \{x : \tau_1\} \vdash N : \tau_2$. It follows from $\tau \in \mathcal{T}_2^-$ and from Lemma 3.1 that $\tau_1 \in \mathcal{T}_1$ and $\tau_2 \in \mathcal{T}_2^-$. Thus, (b) is true, and $\Gamma \cup \{x : \tau_1\} \vdash N : \tau_2$ satisfies the premises. We proceed from this node in the new tree.

For (c) consider a node $\mathtt{n}$ with label $\Gamma \vdash x : \tau$, where $\tau \in \mathcal{T}_2^-$ or τ is of rank 1 but is not an intersection, and all types in Γ are of rank 1. In $\Pi[\mathtt{n}]$ this node

results from $\Gamma \vdash x : \tau_1 \cap \cdots \cap \tau_n \cap \tau$ by rule (var), followed by an application of rule (E$\cap$) in case $n > 0$.

For the last case consider a node $\mathtt{n}$ labelled with $\Gamma \vdash xN_1 \cdots N_k : \tau$ ($k \geq 1$), satisfying the premises, i.e. all types in Γ are of rank 1 and $\tau \in \mathcal{T}_2^-$ or τ is of rank 1 but is not an intersection. Consider again $\Pi[\mathtt{n}]$ obtained from Π and $\Pi'_{\mathtt{n}}$, as described in Definition 3.2. The leaf node of $\Pi'_{\mathtt{n}}$ is labelled with a formula of the form $\Gamma \vdash xN_1 \cdots N_k : \tau_1 \cap \cdots \cap \tau_n \cap \tau$ and has to result by rule (E$\to$) from two nodes, respectively labelled with $\Gamma \vdash xN_1 \cdots N_{k-1} : \sigma \to \tau_1 \cap \cdots \cap \tau_n \cap \tau$ and $\Gamma \vdash N_k : \sigma$. Thus, $\Gamma \vdash xN_1 \cdots N_k : \tau$ is derived from these two formulas by rule (E$\to$), followed by an application of rule (E$\cap$). Since the type declared for x in Γ is of rank 1 and $\Gamma \vdash xN_1 \cdots N_{k-1} : \sigma \to \tau_1 \cap \cdots \cap \tau_n \cap \tau$ ($n \geq 0$), then $\sigma \in \mathcal{T}$ and $\tau_1 \cap \cdots \cap \tau_n \cap \tau \in \mathcal{T}_1$, thus $\sigma \to \tau_1 \cap \cdots \cap \tau_n \cap \tau \in \mathcal{T}_1$. Consequently, (d) is true. Also, $\Gamma \vdash xN_1 \cdots N_{k-1} : \sigma \to \tau_1 \cap \cdots \cap \tau_n \cap \tau$ and $\Gamma \vdash N_k : \sigma$ satisfy the premises. The construction, continues in the new tree from these two nodes.

Now, the construction of a derivation tree for $\vdash_2 M : \theta$ from Π' is straightforward. We proceed as follows, replacing one after another, one or two derivation steps of a formula $\Gamma \vdash N : \tau$ by a derivation step of formula $\Gamma_{|\mathsf{FV}(N)} \vdash N : \tau$. Here $\Gamma_{|\mathsf{FV}(N)}$ denotes the restriction of Γ to the free variables in N. The replacement is performed top-down, starting at the leafs of Π'.

- Any derivation of $\Gamma \vdash x : \tau$ in Π', resulting from $\Gamma \vdash x : \tau_1 \cap \cdots \cap \tau_n \cap \tau$ by rule (var), followed by an application of rule (E$\cap$) in case $n > 0$, is replaced by rule $\{x : \tau_1 \cap \cdots \cap \tau_n \cap \tau\} \vdash_2 x : \tau$, where $x : \tau_1 \cap \cdots \cap \tau_n \cap \tau$ is the type declaration for x in Γ.
- Any derivation of $\Gamma \vdash xN_1 \cdots N_k : \tau$ in Π', resulting from $\Gamma \vdash xN_1 \cdots N_{k-1} : \sigma \to \tau_1 \cap \cdots \cap \tau_n \cap \tau$ and $\Gamma \vdash N_k : \sigma$ by rule (E$\to$), followed by an application of rule (E$\cap$) in case $n > 0$, is replaced by one application of rule (E$\to_2$) to formulas $\Gamma_{|\mathsf{FV}(xN_1 \cdots N_{k-1})} \vdash_2 xN_1 \cdots N_{k-1} : \sigma \to \tau_1 \cap \cdots \cap \tau_n \cap \tau$ and $\Gamma_{|\mathsf{FV}(N_k)} \vdash_2 N_k : \sigma$, deriving $\Gamma_{|\mathsf{FV}(xN_1 \cdots N_k)} \vdash_2 xN_1 \cdots N_k : \tau$.
- Any derivation of $\Gamma \vdash \lambda x.N : \tau_1 \to \tau_2$ in Π', resulting from $\Gamma \cup \{x : \tau_1\} \vdash N : \tau_2$ by (I$\to$), is replaced by one application of (E$\to_2$) to $(\Gamma \cup \{x : \tau_1\})_{|\mathsf{FV}(N)} \vdash_2 N : \tau_2$, deriving $(\Gamma \cup \{x : \tau_1\})_{|\mathsf{FV}(N)} \setminus \{x : \tau_1\} \vdash_2 \lambda x.N : \tau_1 \to \tau_2$, i.e. $\Gamma_{|\mathsf{FV}(\lambda x.N)} \vdash_2 \lambda x.N : \tau_1 \to \tau_2$. □

□

Example 3.5 Consider again types τ, τ_{17} and τ_{15} and context Γ from Example 2.4 and the normal term $M = \lambda x.x(\lambda y.xyy)$. A deduction for $\vdash_2 M : \tau$ is depicted

below.

$$\dfrac{\dfrac{}{x : (o \to o) \to o}\,\mathsf{var}_2 \qquad \dfrac{\dfrac{\dfrac{}{x : \tau_{17} \vdash_2 x : o \to \tau_{15}}\,\mathsf{var}_2 \quad \dfrac{}{y : o \vdash_2 y : o}\,\mathsf{var}_2}{x : \tau_{17}, y : o \vdash_2 xy : o \to ((o \to o) \cap o)}\,\mathrm{E}{\to_2} \quad \dfrac{}{y : o \vdash_2 y : o}\,\mathsf{var}_2}{x : \tau_{17}, y : o \vdash_2 xyy : o}\,\mathrm{E}{\to_2}}{x : \tau_{17} \vdash_2 \lambda y.xyy : o \to o}\,\mathrm{I}{\to_2}}{x : \tau_{17} \vdash_2 x(\lambda y.xyy) : o}\,\mathrm{E}{\to_2}}{\vdash_2 \lambda x.x(\lambda y.xyy) : \tau}\,\mathrm{I}{\to_2}$$

Note that this deduction tree for $\vdash_2 M : \tau$ was obtained from the deduction tree for $\vdash M : \tau$ displayed in Example 2.4, applying the transformation described in the proof of Proposition 3.4. Since in that particular tree there was no application of rule (I$\cap$), in this case the transformation consists essentially of successive substitutions of subtrees of the form

$$\dfrac{\dfrac{\dfrac{\cdots}{\Gamma \vdash_2 M : \tau_1 \cap \cdots \cap \tau_n}\,\mathtt{rule}}{\Gamma \vdash_2 M : \tau_i}\,\mathrm{E}\cap}$$

by

$$\dfrac{\cdots}{\Gamma \vdash_2 M : \tau_i}\,\mathtt{rule}_2$$

where $\mathtt{rule} \in \{\mathtt{var}, \mathtt{I}{\to}, \mathtt{E}{\to}\}$, $n \geq 2$ and $1 \leq i \leq n$.

The following example shows that one may have more than one derivation for a statement $\vdash_2 M : \tau$.

Example 3.6 Consider $\alpha = (a \to b) \cap (c \to b) \to a \cap c \to b$. There are two derivations for $\vdash_2 \lambda xy.xy : \alpha$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : (a \to b) \cap (c \to b) \vdash_2 x : a \to b}\,\mathsf{var}_2 \quad \dfrac{}{y : a \cap c \vdash_2 y : a}\,\mathsf{var}_2}{x : (a \to b) \cap (c \to b), y : a \cap c \vdash_2 xy : b}\,\mathrm{E}{\to_2}}{x : (a \to b) \cap (c \to b) \vdash_2 \lambda y.xy : a \cap c \to b}\,\mathrm{I}{\to_2}}{\vdash_2 \lambda xy.xy : \tau}\,\mathrm{I}{\to_2}$$

The other derivation projects on the second component in intersections.

Consider a term M in β-normal form, a type $\theta \in \mathcal{T}_2^-$, and a $\vdash_2$-derivation tree Π of $\vdash_2 M : \theta$. In the following, we assign occurrences of subtypes of θ, to all variables in $\mathsf{Subj}(\Gamma)$ and to all subterms of N, for every formula $\Gamma \vdash_2 N : \sigma$ appearing in Π. This assignment is crucial to prove correctness of the pre-grammars defined in the next section. Every $x \in \mathsf{Subj}(\Gamma)$ is assigned a negative subpremise $\mathsf{nsp}(x)$ of θ. Every subterm N' of N, which is not in function position, i.e. has no arguments, is assigned a positive occurrence $\mathsf{pst}(N')$ of a subtype of θ. Additionally, every subterm of the form $xN_1 \cdots N_s$, with $s \geq 0$, is assigned a negative occurrence of a subtype of θ, denoted by $\mathsf{nst}(xN_1 \cdots N_s)$. Note, that subterms of this last form, that have no further arguments, are assigned both a positive, as well as a negative occurrence of a subtype of θ.

Definition 3.7 Consider a term M, a type $\theta \in \mathcal{T}_2^-$, and a $\vdash_2$-derivation tree Π of $\vdash_2 M : \theta$. The assignment of nsp, nst and pst to occurrences of variables and subtypes in the formulas, that appear in Π, is bottom-up, starting with $\vdash_2 M : \theta$.

- For $\vdash_2 M : \theta$, let $\mathsf{pst}(M) = \theta$ (note that M is necessarily of the form $\lambda x.N$, thus $\mathsf{nst}(M)$ is not defined).
- Now consider $\Gamma \setminus \{x : \sigma\} \vdash_2 \lambda x.N : \sigma \to \tau$, because $\Gamma \vdash_2 N : \tau$ and $\Gamma \cup \{x : \sigma\}$ is consistent. Consider $\mathsf{pst}(\lambda x.N) = \underline{\sigma \to \tau}$ for $\Gamma \setminus \{x : \sigma\} \vdash_2 \lambda x.N : \sigma \to \tau$. Then, for $\Gamma \vdash_2 N : \tau$ let $\mathsf{pst}(N)$ be the occurrence of τ in $\mathsf{pst}(\lambda x.N)$. If $x \in \mathsf{Subj}(\Gamma)$, then $\mathsf{nsp}(x)$ is the occurrence of σ in $\underline{\sigma \to \tau}$. All other variables in $\mathsf{Subj}(\Gamma)$ are assigned the same occurrences as for the formula $\Gamma \setminus \{x : \sigma\} \vdash_2 \lambda x.N : \sigma \to \tau$.
- Finally, consider a formula of the form $\Gamma \vdash_2 xN_1 \cdots N_k : \tau$ with $k \geq 0$. Note that all elements in $\mathsf{Subj}(\Gamma)$ are already assigned negative subpremises in θ. Also, this occurrence of $xN_1 \cdots N_k$ is already assigned a positive subtype $\mathsf{pst}(xN_1 \cdots N_k) = \underline{\tau}$.

(i) If $k = 0$, then the formula is of the form $\{x : \tau_1 \cap \cdots \cap \tau_n\} \vdash_2 x : \tau_i$, for $n \geq 1$ and $i \in \{1, \ldots, n\}$. At this point, the occurrence of x in the context is already assigned a negative subpremise $\mathsf{nsp}(x) = \underline{\tau_1 \cap \cdots \cap \tau_n}$ and the occurrence on the right side of $\vdash_2$ is assigned a positive occurrence of a subtype $\mathsf{pst}(x) = \underline{\tau_i}$. We further assign $\mathsf{nst}(x)$ to x on the right side of $\vdash_2$, choosing the occurrence of τ_i in $\mathsf{nsp}(x)$, which is a negative subtype in θ.

(ii) If $k > 0$, then there is a subtree Π' of Π that derives formula $\Gamma \vdash_2 xN_1 \cdots N_k : \tau$. In this subtree a declaration of the form $\{x : \tau_1 \cap \cdots \cap \tau_n\} \vdash_2 x : \tau_i$ is first combined with a derived formula $\Gamma_1 \vdash_2 N_1 : \sigma_1$ (by some subtree Π_1 of Π'). The resulting formula $\{x : \tau_1 \cap \cdots \cap \tau_n\} \cup \Gamma_1 \vdash_2 xN_1 : \tau_i^1$ is then combined with a derived formula $\Gamma_2 \vdash_2 N_2 : \sigma_2$ (by some subtree Π_2 of Π'), etc. In the final formula we have $\Gamma = \{x : \tau_1 \cap \cdots \cap \tau_n\} \cup \Gamma_1 \cup \cdots \cup \Gamma_k$. At this point, every occurrence of a variable y in Γ is already assigned a negative subpremise $\mathsf{nsp}(y)$. Thus, the same is true for $\Gamma_1, \ldots, \Gamma_k$. We now successively assign, operating top down, negative subtypes to terms $x, xN_1, \ldots, xN_1 \cdots N_k$, as well as positive subtypes to terms $N_1, \ldots, N_k$, in the formulas in this part of the tree. For x on the right side of $\{x : \tau_1 \cap \cdots \cap \tau_n\} \vdash_2 x : \tau_i$ let $\mathsf{nst}(x)$ be the occurrence of τ_i in $\mathsf{nsp}(x) = \underline{\tau_1 \cap \cdots \cap \tau_n}$. Now, suppose that formula $\{x : \tau_1 \cap \cdots \cap \tau_n\} \cup \Gamma_1 \cup \cdots \cup \Gamma_j \vdash_2 xN_1 \cdots N_j : \sigma \to \tau_1^j \cap \cdots \cap \tau_{n_j}^j$ is combined with $\Gamma_{j+1} \vdash_2 N_{j+1} : \sigma$, deriving $\{x : \tau_1 \cap \cdots \cap \tau_n\} \cup \Gamma_1 \cup \cdots \cup \Gamma_{j+1} \vdash_2 xN_1 \cdots N_j N_{j+1} : \tau_{i_j}^j$, where $i_j \in \{1, \ldots, n_j\}$. Consider $\mathsf{nst}(xN_1 \cdots N_j) = \underline{\sigma \to \tau_1^j \cap \cdots \cap \tau_{n_j}^j}$ (which is already assigned). Then, $\mathsf{pst}(N_{j+1})$ is the occurrence of σ in $\mathsf{nst}(xN_1 \cdots N_j)$ and $\mathsf{nst}(xN_1 \cdots N_j N_{j+1})$ is the occurrence of $\tau_{i_j}^j$ in $\mathsf{nst}(xN_1 \cdots N_j)$.

4 Pre-grammars for types in $\mathcal{T}_2^-$

In [2,3] it was described how to obtain for a simple type $\alpha \in \mathcal{T}$ a set of rewriting rules, denoted by $\mathsf{pre}(\alpha)$, and called the *pre-grammar* of α. It was shown how to use pre-grammars to address problems such as type-checking, the emptiness problem and principal inhabitation. Furthermore, closure properties were studied. In particular, it was shown how to obtain, given pre-grammars of two types α and β, a new grammar corresponding exactly to the normal inhabitants of both α and

β, i.e. to the normal inhabitants of $\alpha \cap \beta$.

In order to extend the methods described in [3], in this section we define the notion of pre-grammar for types in $\mathcal{T}_2^-$. The definition of pre-grammars given here will coincide with the one in [3], when applied to types containing no intersections, i.e. simple types. In Section 6 we will combine pre-grammars, following the method described in [3], to obtain grammars corresponding to intersections of $\mathcal{T}_2^-$-types.

We start by associating to each type τ a set $\mathsf{occT}(\tau)$ that contains for every occurrence of a subtype $\underline{\tau'}$ a tuple (τ', n, l), where $n \in \mathbb{N}$, and $\mathsf{l} \in \{\mathsf{var}\} \cup \{\, n \to m \mid n, m \in \mathbb{N} \,\} \cup \{\, n_1 \cap \cdots \cap n_m \mid n_1, \ldots, n_m \in \mathbb{N} \,\}$. Distinct occurrences of subtypes are assigned distinct tuples. This set is uniquely defined, up to isomorphism between integers used in the tuples.

Definition 4.1 Given a type $\tau \in \mathcal{T}_2$, let $\mathsf{occT}(\tau)$ be the smallest set satisfying the following.

- For each occurrence of a type variable a in τ there is a tuple $(a, n, \mathsf{var}) \in \mathsf{occT}(\tau)$;
- if $\sigma_1 \to \tau_1$ is an occurrence of a subtype of τ, and $(\sigma_1, n, \mathsf{l}_{\sigma_1}), (\tau_1, m, \mathsf{l}_{\tau_1}) \in \mathsf{occT}(\tau)$ are the tuples corresponding to σ_1 and τ_1 in this occurrence, then there is a tuple $(\sigma_1 \to \tau_1, k, n \to m) \in \mathsf{occT}(\tau)$;
- if $\alpha_1 \cap \cdots \cap \alpha_m$, with $m \geq 2$, is an occurrence of a subtype of τ, and $(\alpha_1, n_1, \mathsf{l}_{\alpha_1}), \ldots, (\alpha_m, n_m, \mathsf{l}_{\alpha_m}) \in \mathsf{occT}(\tau)$ are the tuples corresponding to $\alpha_1, \ldots, \alpha_m$ in this occurrence, then $(\alpha_1 \cap \cdots \cap \alpha_m, k, n_1 \cap \cdots \cap n_m) \in \mathsf{occT}(\tau)$;
- for each $n \in \mathbb{N}$ there is at most one tuple $(\sigma, n, \mathsf{l}) \in \mathsf{occT}(\tau)$.

Furthermore, given a particular occurrence $\underline{\tau'}$ of a subtype of τ we denote by $\mathsf{n}(\underline{\tau'})$ the unique integer n such that $(\tau', n, \mathsf{l}) \in \mathsf{occT}(\tau)$. We frequently will refer to $\mathsf{n}(\tau')$ as the *identifier* of $\underline{\tau'}$ w.r.t. $\mathsf{occT}(\tau)$. The type of identifier n is $\mathsf{t}(n) = \tau'$, its label is $\mathsf{lab}(n) = \mathsf{l}$, and $\mathsf{N}(\tau) = \{\, n \mid (\tau', n, \mathsf{l}) \in \mathsf{occT}(\tau) \,\}$. Finally, if $(\alpha_1 \cap \cdots \cap \alpha_m, k, n_1 \cap \cdots \cap n_m) \in \mathsf{occT}(\tau)$ $(m \geq 2)$, then each of $n_1, \ldots, n_m$ is called a component of identifier k.

In order to deal correctly with the correspondence between occurrences of subtypes and occurrences of subterms, polarities have to be taken into account. With this purpose, and whenever convenient, we might superscript an integer n with '+' if n corresponds to a positive occurrence of a subtype, i.e. an occurrence that can be the type of a subterm of an inhabitant, and with '−' if it corresponds to a negative subpremise, i.e. if it corresponds to an occurrence that can be the type of a variable in an abstraction sequence. Integers that correspond to a negative occurrence, which is no subpremise, will not be superscripted.

Definition 4.2 We say that two integers $n, m \in \mathsf{N}(\tau)$ are equivalent w.r.t. $\mathsf{occT}(\tau)$, and write $n \equiv_{\mathsf{occT}} m$, if and only if $\mathsf{t}(n) = \mathsf{t}(m)$. The binary relation $T(\tau) \subseteq \mathsf{N}(\tau) \times \mathsf{N}(\tau)$ is defined by $(p_2, p_3) \in T(\tau)$ iff $(\tau_3, p_3, p_1 \to p_2) \in \mathsf{occT}(\tau)$, i.e. $\tau_3 = \tau_1 \to \tau_2$, $\mathsf{n}(\tau_1) = p_1$, $\mathsf{n}(\tau_2) = p_2$, and $\mathsf{n}(\tau_3) = p_3$. Furthermore, for $(p_2, p_3) \in T(\tau)$ let $\mathsf{q}(p_2, p_3) = p_1$.

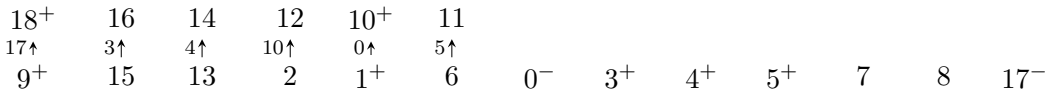
Example 4.3 For τ from Example 2.4 the set $\mathsf{occT}(\tau)$ contains nineteen tuples

(τ', n, l), where τ', n and l are given below (tuples for occurrences of identical subtypes, i.e. equivalent modulo $\equiv_{\mathsf{occT}}$, are displayed in the same line)

τ'	n	l
o	$0, \ldots, 9$	var
$o \to o$	$10, 11$	$0 \to 1, 5 \to 6$
$(o \to o) \to o$	12	$10 \to 2$
$(o \to o) \cap o$	13	$11 \cap 7$
$o \to (o \to o) \cap o$	14	$4 \to 13$

τ'	n	l
$(o \to (o \to o) \cap o) \cap o$	15	$14 \cap 8$
$o \to (o \to (o \to o) \cap o) \cap o$	16	$3 \to 15$
$\tau_1 \cap \tau_2$	17	$12 \cap 16$
τ	18	$17 \to 9$

where $\tau_1 = (o \to o) \to o$ and $\tau_2 = o \to (o \to (o \to o) \cap o) \cap o$. The equivalence relation $\equiv_{\mathsf{occT}}$ partitions $\mathsf{N}(\tau)$ into nine equivalence classes, which are $\{0^-, 1^+, 2, 3^+, 4^+, 5^+, 6, 7, 8, 9^+\}$,$\{10^+, 11\}$, $\{12\}$, $\{13\}$, $\{14\}$, $\{15\}$, $\{16\}$,$\{17^-\}$, and $\{18^+\}$. The associated graph $T(\tau)$ is depicted below.



Now, $\mathsf{pre}(\tau)$ is constructed from $\mathsf{occT}(\tau)$ and $T(\tau)$.

Definition 4.4 Given $m, n \in \mathsf{N}(\tau)$, we write $m \preceq_\cap n$ if and only if either $m = n$ and n is not the identifier of an intersection, or n is the identifier of an intersection of arity ≥ 2 and m is a component of n.

Example 4.5 For $\mathsf{occT}(\tau)$ from Example 4.3, we have $7, 11 \preceq_\cap 13$, $8, 14 \preceq_\cap 15$, and $12, 16 \preceq_\cap 17^-$.

Definition 4.6 Given a type τ and a set of tuples $\mathsf{occT}(\tau)$, we denote by $\mathsf{pre}(\tau)$ the smallest set of rules satisfying the following conditions.

- If $m^+, k^-, n^+ \in \mathsf{N}(\tau)$, $(\beta, m, k \to n) \in \mathsf{occT}(\tau)$, then $m := \lambda k.n \in \mathsf{pre}(\tau)$;
- if $m^+, p_0^- \in \mathsf{N}(\tau)$ and $(p_s, c_{s-1}), \ldots, (p_2, c_1), (p_1, c_0) \in T(\tau)$, for some $s \geq 0$, $m^+ \equiv_{\mathsf{occT}} c_s$, $c_j \preceq_\cap p_j$ $(1 \leq j \leq s)$, $\mathsf{q}(p_i, c_{i-1}) = n_i$ $(1 \leq i \leq s)$, then $m := p_0\ n_1 \cdots n_s \in \mathsf{pre}(\tau)$.

Example 4.7 From $\mathsf{occT}(\tau)$ and $T(\tau)$ in Example 4.3 we obtain the following set

$\mathsf{pre}(\tau)$ containing twenty eight rewriting rules.

$$18 := \lambda 17.9$$

$$1, 3, 4, 5, 9 := 17\ 3\ 4\ 5 \mid 17\ 3\ 4 \mid 17\ 3 \mid 17\ 10 \mid 0$$

$$10 := \lambda 0.1 \mid 17\ 3\ 4$$

For instance, we have $9 := 17\ 10 \in \mathsf{pre}(\tau)$, because $9 \equiv_{\mathsf{occT}} 2$, $2 \preceq_\cap 2$, $(2, 12) \in T(\tau)$, $12 \preceq_\cap 17^-$, and $\mathsf{q}(2, 12) = 10$. Also, $1 := 17\ 3\ 4 \in \mathsf{pre}(\tau)$, because $1 \equiv_{\mathsf{occT}} 7$, $7 \preceq_\cap 13$, $(13, 14) \in T(\tau)$, $14 \preceq_\cap 15$, $(15, 16) \in T(\tau)$, $16 \preceq_\cap 17^-$, $\mathsf{q}(15, 16) = 3$, and $\mathsf{q}(13, 14) = 4$.

Intuitively, each rule $m^+ := \ldots$ in this pre-grammar describes the different possibilities of constructing a subterm (of an inhabitant of τ) of type $\mathsf{t}(m)$ (where $\mathsf{t}(m)$ is a positive occurrence in τ), based on the occurrences of subtypes of τ. For instance, rule $18 := \lambda 17.9$ means that a subterm of type $\mathsf{t}(18)$ can be of the form $\lambda x.N$, where x is a variable whose type corresponds to the negative occurrence of $\mathsf{t}(17)$ in τ, and N is a subterm whose type corresponds to the positive occurrence of $\mathsf{t}(9)$ in τ. On the other hand, rule $9 := 17\ 10$ means that a subterm of type $\mathsf{t}(9)$ can be of the form xN, where x is a variable whose type corresponds to the negative occurrence of $\mathsf{t}(17)$ in τ, and N is a subterm whose type corresponds to the positive occurrence of $\mathsf{t}(10)$ in τ. This is due to the fact that $\mathsf{t}(17) = \mathsf{t}(12) \cap \mathsf{t}(16)$, $\mathsf{t}(12) = \mathsf{t}(10) \to \mathsf{t}(2)$, and $\mathsf{t}(2) = \mathsf{t}(9)$.

5 Deciding Inhabitation for Types in $\mathcal{T}_2^-$

In this section we define a rewrite-based non-deterministic decision algorithm, that checks if a given type in $\mathcal{T}_2^-$ has a normal inhabitant, thus solving the emptiness problem for the set of $\mathcal{T}_2^-$ types.

Definition 5.1 Given a type $\tau \in \mathcal{T}_2^-$, an identifier $m \in \mathsf{N}(\tau)$ and a set $V \subseteq \mathsf{N}(\tau)$, we write
$(m, V) \rightsquigarrow (n_1, V'), \ldots, (n_s, V')$, if one of the following applies.

- If $m := \lambda k.n \in \mathsf{pre}(\tau)$, then $(m, V) \rightsquigarrow (n, V \cup \{k\})$;
- if $m := k\ n_1 \cdots n_s \in \mathsf{pre}(\tau)$ and $k \in V$, then $(m, V) \rightsquigarrow (n_1, V), \ldots, (n_s, V)$.

The definition of $\rightsquigarrow$ extends, in the usual way, to rewriting of sequences of pairs. Then, $\rightsquigarrow^*$ denotes the reflexive, transitive closure of $\rightsquigarrow$.

Definition 5.2 For a particular rewriting sequence of $(\mathsf{n}(\tau), \emptyset) \rightsquigarrow^* \epsilon$, we define a function pair that computes for each (m, V) in that rewriting sequence a tuple $(M, \Gamma) = \mathsf{pair}(m, V)$. For convenience, we will use identifiers as indexes of term variables in such a way that the type assigned to a variable with name x_n, for $n \in \mathsf{N}(\tau)$, is always $\mathsf{t}(n)$. The function pair is recursively defined as follows.

- If $(m, V) \rightsquigarrow (n, V \cup \{k\})$ because $m := \lambda k.n \in \mathsf{pre}(\tau)$, then $\mathsf{pair}(m, V) = (\lambda x_k.N, \Gamma \setminus \{x_k : \mathsf{t}(k)\})$, where $(N, \Gamma) = \mathsf{pair}(n, V \cup \{k\})$;
- if $(m, V) \rightsquigarrow (n_1, V), \ldots, (n_s, V)$ because $m := k\ n_1 \cdots n_s \in \mathsf{pre}(\tau)$ and $k \in V$,

then $\mathsf{pair}(m, V) = (x_k\ N_1 \cdots N_s, \{x_k : \mathsf{t}(k)\} \cup \Gamma_1 \cup \cdots \cup \Gamma_s)$, where $(N_i, \Gamma_i) = \mathsf{pair}(n_i, V)$, for $1 \leq i \leq s$ $(s \geq 0)$.

Note that function pair actually does not depend on set V, but on the identifier m and on the rule in $\mathsf{pre}(\tau)$, which is used in each step of the rewriting sequence. Actually, it is the applicability of a rule, that depends on set V. Nevertheless, it is easy to verify that, $x_k : \mathsf{t}(k) \in \Gamma$ implies $k \in V$. Consequently, if $V = \emptyset$, then we have that $\Gamma = \emptyset$. To guarantee that pair is well-defined, we suppose that in each rewriting step, the corresponding rewriting rule is given, either implicitly or explicitly. The correctness of function pair is stated in the following lemma.

Lemma 5.3 *If $(m, V) \rightsquigarrow^* \epsilon$ and $(M, \Gamma) = \mathsf{pair}(m, V)$ for some rewriting sequence, then $\Gamma \vdash_2 M : \mathsf{t}(m)$.*

Proof. By structural induction on M. We first consider the case where $\mathsf{pair}(m, V) = (\lambda x_k.N, \Gamma \backslash \{x_k : \mathsf{t}(k)\})$, which follows from $(m, V) \rightsquigarrow (n, V \cup \{k\}) \rightsquigarrow^* \epsilon$ because $m := \lambda k.n \in \mathsf{pre}(\tau)$ and $(N, \Gamma) = \mathsf{pair}(n, V \cup \{k\})$. By the induction hypothesis, $\Gamma \vdash N : \mathsf{t}(n)$ and by definition $\Gamma \cup \{x_k : \mathsf{t}(k)\}$ is always consistent. Therefore, $\Gamma \vdash \lambda x_k.N : \mathsf{t}(k) \to \mathsf{t}(n) = \mathsf{t}(m)$.

Now consider $\mathsf{pair}(m, V) = (x_k N_1 \cdots N_s, \{x_k : \mathsf{t}(k)\} \cup \Gamma_1 \cup \cdots \cup \Gamma_s)$, which follows from $(m, V) \rightsquigarrow (n_1, V), \ldots, (n_s, V) \rightsquigarrow^* \epsilon$ because $m := k\ n_1 \cdots n_s \in \mathsf{pre}(\tau)$ and $k \in V$, $(N_i, \Gamma_i) = \mathsf{pair}(n_i, V)$, for $1 \leq i \leq s$ $(s \geq 0)$. By the induction hypothesis $\Gamma_i \vdash N_i : \mathsf{t}(n_i)$ and by definition $\{x_k : \mathsf{t}(k)\} \cup \Gamma_1 \cup \cdots \cup \Gamma_s$ is consistent. It follows from $m := k\ n_1 \cdots n_s \in \mathsf{pre}(\tau)$ that

$$
\begin{aligned}
\mathsf{t}(k) &= \cdots \cap \underbrace{(\mathsf{t}(n_1) \to \mathsf{t}(p_1))}_{\mathsf{t}(c_0)} \cap \cdots \\
\mathsf{t}(p_1) &= \cdots \cap \underbrace{(\mathsf{t}(n_2) \to \mathsf{t}(p_2))}_{\mathsf{t}(c_1)} \cap \cdots \\
&\vdots \\
\mathsf{t}(p_{s-1}) &= \cdots \cap \underbrace{(\mathsf{t}(n_s) \to \mathsf{t}(p_s))}_{\mathsf{t}(c_{s-1})} \cap \cdots \\
\mathsf{t}(p_s) &= \cdots \cap \mathsf{t}(c_s) \cap \cdots,
\end{aligned}
$$

where $\mathsf{t}(c_s) = \mathsf{t}(m)$. By rule (var_2), we have $\{x_k : \mathsf{t}(k)\} \vdash_2 x_k : \mathsf{t}(c_0)$. Since $\Gamma_1 \vdash N_1 : \mathsf{t}(n_1)$, we obtain $\{x_k : \mathsf{t}(k)\} \cup \Gamma_1 \vdash x_k N_1 : \mathsf{t}(c_1)$, by rule $(\mathrm{E}{\to_2})$. Repeating this process the necessary number of times, we finally get $\{x_k : \mathsf{t}(k)\} \cup \Gamma_1 \cup \cdots \cup \Gamma_s \vdash x_k N_1 \cdots N_s : \mathsf{t}(c_s)$, where $\mathsf{t}(c_s) = \mathsf{t}(m)$. □

Example 5.4 Consider τ and $\mathsf{pre}(\tau)$ from Examples 2.4 and 4.3. Then,

$$
\begin{aligned}
(18, \emptyset) \rightsquigarrow (9, \{17\}) \rightsquigarrow (10, \{17\}) \rightsquigarrow (1, \{0, 17\}) \rightsquigarrow (3, \{0, 17\}), (4, \{0, 17\}) \\
\rightsquigarrow (4, \{0, 17\}) \rightsquigarrow \epsilon.
\end{aligned}
$$

For the first two pairs in this rewriting sequence we have respectively $\mathsf{pair}(17, \emptyset) = (\lambda x_{17}.x_{17}(\lambda x_0.x_{17}x_0x_0), \emptyset)$ and $\mathsf{pair}(9, \{17\}) = (x_{17}(\lambda x_0.x_{17}x_0x_0), \{x_{17} : \mathsf{t}(17)\})$, where $\mathsf{t}(17) = ((o \to o) \to o) \cap (o \to ((o \to ((o \to o) \cap o)) \cap o))$. Also,

$\vdash_2 \lambda x_{17}.x_{17}(\lambda x_0.x_{17}x_0x_0)) : \mathsf{t}(18)$ and $\{x_{17} : \mathsf{t}(17)\} \vdash_2 x_{17}(\lambda x_0.x_{17}x_0x_0) : \mathsf{t}(9)$, where $\mathsf{t}(18) = \tau$ and $\mathsf{t}(9) = o$.

Theorem 5.5 *Let $\theta \in \mathcal{T}_2^-$. Then, $\mathsf{Nhabs}(\theta) \neq \emptyset$ if and only if $(\mathsf{n}(\theta), \emptyset) \rightsquigarrow^* \epsilon$.*

Proof. The 'if' part follows from Lemma 5.3. For the 'only if' part, we show that for any term M, context Γ and type σ, if a formula $\Gamma \vdash_2 M : \sigma$ appears in a derivation of $\vdash_2 M' : \theta$, then $(\mathsf{n}(\underline{\sigma}), V_\Gamma) \rightsquigarrow^* \epsilon$, where $\underline{\sigma}$ is the positive occurrence of σ in θ assigned to M for that formula, i.e. $\underline{\sigma} = \mathsf{pst}(M)$, and $V_\Gamma = \{\ \mathsf{n}(\underline{\rho}) \mid x : \rho \in \Gamma \wedge \underline{\rho} = \mathsf{nsp}(x)\ \}$.

First, consider $M = \lambda x.N$ and suppose that we have $\Gamma \backslash \{x : \sigma\} \vdash_2 \lambda x.N : \sigma \to \tau$, because $\Gamma \vdash_2 N : \tau$ and $\Gamma \cup \{x : \sigma\}$ is consistent. Consider the corresponding positive occurrence $\mathsf{pst}(\lambda x.N) = \underline{\sigma \to \tau}$. Then, there is a tuple $(\sigma \to \tau, m, k \to n) \in \mathsf{occ\Gamma}(\theta)$, where $\mathsf{n}(\sigma \to \tau) = m$, $\mathsf{n}(\sigma) = k$, and $\mathsf{n}(\tau) = n$, corresponding to that positive occurrence. The occurrences of σ in $\underline{\sigma \to \tau}$ is, by definition, a negative subpremise and the occurrence of τ is positive. We conclude that $m^+, k^-, n^+ \in \mathsf{N}(\theta)$ and consequently, $m := \lambda k.n \in \mathsf{pre}(\theta)$. Thus, $(m, V_{\Gamma\backslash\{x:\sigma\}}) \rightsquigarrow (n, V_{\Gamma\backslash\{x:\sigma\}} \cup \{k\})$. But, $V_{\Gamma\backslash\{x:\sigma\}} \cup \{k\} \supseteq V_\Gamma$ and $(n, V_\Gamma) \rightsquigarrow^* \epsilon$ follows from the induction hypothesis.

Now, consider a formula of the form $\Gamma \vdash_2 xN_1 \cdots N_s : \tau$ with $s \geq 0$ and suppose that this formula was derived, step by step, starting with $\{x : \tau_1^0 \cap \cdots \cap \tau_{n_0}^0\} \vdash_2 x : \tau_{i_0}^0$, where $i_0 \in \{0, , \ldots, n_0\}$. This formula, combined with $\Gamma_1 \vdash_2 N_1 : \sigma_1$ lead to $\{x : \tau_1^0 \cap \cdots \cap \tau_{n_0}^0\} \cup \Gamma_1 \vdash_2 xN_1 : \tau_{i_1}^1$, etc., until reaching $\Gamma \vdash_2 xN_1 \cdots N_s : \tau_{i_s}^s$, where $\tau_{i_s}^s = \tau$. Consider the identifiers $p_0^- = \mathsf{n}(\mathsf{nsp}(x))$ and $c_0 = \mathsf{n}(\mathsf{nst}(x))$ of the negative occurrences of subtypes of θ, respectively assigned to x in the context and on the right side of $\vdash_2$ in the starting formula. Then, $c_0 \preceq_\cap p_0^-$. Let $n_1 = \mathsf{n}(\mathsf{pst}(N_1))$, where $\mathsf{pst}(N_1)$ is the positive occurrence of σ_1 assigned to N_1 for $\Gamma_1 \vdash_2 N_1 : \sigma_1$. Necessarily, $\tau_{i_0}^0 = \sigma_1 \to \tau_1^1 \cap \cdots \cap \tau_{n_1}^1$, and we have $\mathsf{lab}(c_0) = n_1 \to p_1$, where $\mathsf{t}(p_1) = \tau_1^1 \cap \cdots \cap \tau_{n_1}^1$. We conclude that $(p_1, c_0) \in T(\theta)$, with $\mathsf{q}(p_1, c_0) = n_1$. Next, let $c_1 = \mathsf{n}(\mathsf{nst}(xN_1))$, assigned for formula $\{x : \tau_1^0 \cap \cdots \cap \tau_{n_0}^0\} \cup \Gamma_1 \vdash_2 xN_1 : \tau_{i_1}^1$. Again, we conclude that $c_1 \preceq_\cap p_1$. This, argument can be repeated until finally reaching formula $\Gamma \vdash_2 xN_1 \cdots N_s : \tau_{i_s}^s$ where $\tau_{i_s}^s = \tau$. For this formula, term $xN_1 \cdots N_s$ is assigned both a negative, as well as a positive, occurrence of subtype τ. We have $c_s = \mathsf{n}(\mathsf{nst}(xN_1 \cdots N_s))$. Furthermore, for $m^+ = \mathsf{n}(\mathsf{pst}(xN_1 \cdots N_s))$ we have $m^+ \equiv_{\mathsf{occ\Gamma}} c_s$, since $\tau_{i_s}^s = \tau$. From all this, we conclude that there is a rule $m := p_0\ n_1 \cdots n_s \in \mathsf{pre}(\tau)$. On the other hand, $p_0 \in V$, thus $(m, V_\Gamma) \rightsquigarrow (n_1, V_\Gamma), \ldots, (n_s, V_\Gamma)$. By the induction hypothesis, we have $(n_i, V_{\Gamma_i}) \rightsquigarrow^* \epsilon$, for $1 \leq i \leq s$. We conclude that $(\mathsf{n}(\sigma_1), V_\Gamma), \ldots, (\mathsf{n}(\sigma_s), V_\Gamma) \rightsquigarrow^* \epsilon$, since $V_{\Gamma_i} \subseteq V_\Gamma$, for $1 \leq i \leq s$. □

Definition 5.6 Given a type τ, let $|\tau| = |\tau|_v + |\tau|_\to + |\tau|_\cap$, where $|\tau|_v$ represents the number of occurrences of type variables in τ, $|\tau|_\to$ the number of occurrences of $\to$ in τ, and $|\tau|_\cap$ the number of occurrences of $\cap$ in τ. Furthermore, let $|\tau|^+$ and $|\tau|^-$ denote the number of positive occurrences of subformulas and the number of negative subpremises in τ, respectively.

Example 5.7 For τ from Examples 2.4 and 4.3, we have $|\tau| = 19$, $|\tau|^+ = 7$, $|\tau|^- = 2$.

Now, we present a non-deterministic algorithm to address the emptiness problem of $\mathcal{T}_2^-$-types. In every step of the algorithm there are only a finite number of choices that apply. On the other hand, there is an upper bound on the number of steps that can be performed in each recursive call. Consequently, the algorithm can be used as the foundation for a decision procedure.

Theorem 5.8 *Consider $\theta \in \mathcal{T}_2^-$ and let $\mathsf{D} = |\theta|^+ \cdot (|\theta|^- + 1)$. Then, $\mathsf{Nhabs}(\theta) \neq \emptyset$ if and only if the algorithm below terminates with success.*

The algorithm operates as follows, starting with the initial call $(m, V, i) = (\mathsf{n}(\theta), \emptyset, 0)$:

- *if $i > \mathsf{D}$ the loop aborts with failure;*
- *otherwise the algorithm:*
  - *non-deterministically chooses a rule in $\mathsf{r} \in \mathsf{pre}(\theta)$ such that $(m, V) \rightsquigarrow (n_1, V'), \ldots, (n_s, V')$;*
  - *universally applies to calls $(n_1, V', i+1), \ldots, (n_s, V', i+1)$.*

Other than by failure, a loop finishes with success if the rule chosen from $\mathsf{pre}(\theta)$ is such that $s = 0$, or if the recursive calls $(n_1, V', i+1), \ldots, (n_s, V', i+1)$ $(s \geq 1)$ all finish sucessfully. The algorithm finishes with success, if the initial loop starting with call $(\mathsf{n}(\theta), \emptyset, 0)$ finishes successfully.

Proof. By Theorem 5.5 $\mathsf{Nhabs}(\theta) \neq \emptyset$ is equivalent to the existence of an rewriting sequence for $(\mathsf{n}(\theta), \emptyset) \rightsquigarrow^* \epsilon$. Each rewriting sequence of $(\mathsf{n}(\theta), \emptyset) \rightsquigarrow^* \epsilon$ can be represented in the usual way by a unique tree $\mathtt{t}$, whose internal nodes are labelled with tuples (m, V, i) and such that all leafs are labelled with ϵ. The root of $\mathtt{t}$ is $(\mathsf{N}(\theta), \emptyset, 0)$, and whenever a rule $\mathsf{r} \in \mathsf{pre}(\theta)$ is applied to a pair (m, V, i), such that $(m, V) \rightsquigarrow (n_1, V'), \ldots, (n_s, V')$, then the corresponding node in $\mathtt{t}$, labelled with (m, V), has s children labelled with $(n_1, V', i+1), \ldots, (n_s, V', i+1)$ if $s > 0$, and it has one child labelled with ϵ if $s = 0$. Consider a node $\mathtt{n}$ in $\mathtt{t}$ with label (m, V, i). The last coordinate i is the depth of $\mathtt{n}$ in $\mathtt{t}$, i.e. the number of edges from node $\mathtt{n}$ to the tree's root node (labelled with $(\mathsf{N}(\theta), \emptyset, 0)$). The subtree rooted in $\mathtt{n}$ corresponds to a rewriting sequence $(m, V) \rightsquigarrow^* \epsilon$. If there are two nodes $\mathtt{n}_i$ and $\mathtt{n}_j$ in one branch of $\mathtt{t}$ respectively labelled with (m, V, i) and (m, V, j) such that $i < j$, we call this a repetition. This repetition can be eliminated, if one replaces the subtree of $\mathtt{t}$ rooted in $\mathtt{n}_i$ by the subtree rooted in $\mathtt{n}_j$. The resulting tree corresponds still to a (shorter) rewriting sequence of $(\mathsf{n}(\theta), \emptyset) \rightsquigarrow^* \epsilon$. Repeating this process, as long as the tree contains repetitions, one eventually obtains a repetition-free tree. The height of a repetition-free tree is thus limited by the number os possible combinations (m, V) in one branch. On the other hand, if a node $\mathtt{n}_j$, with label (m_j, V_j, j), is a descendant of a node $\mathtt{n}_i$, with label (m_i, V_i, i), where $i < j$, then $V_i \subseteq V_j$. Since the coordinates m and V in a tuple (m, V, i) are respectively an identifier of a positive occurrence of a subtype in θ, and a set of identifiers of negative subpremises in θ, there are at most $|\theta|^+$ different values for m and at most $|\theta|^- + 1$ different sets V in a branch of a repetition-free tree. Thus, all branches in a repetition-free tree are of height $\leq \mathsf{D} = |\theta|^+ \cdot (|\theta|^- + 1)$. Finally, note that for each call (m, V, i) there is only a

finite number of rules $\mathsf{r} \in \mathsf{pre}(\theta)$ that apply. This guarantees the termination of the algorithm. We conclude that, $\mathsf{Nhabs}(\theta) \neq \emptyset$ if and only if the algorithm terminates with success. □

6 Intersections of $\mathcal{T}_2^-$-Types

In this section we recall the method, from [3], of combining pre-grammars of $m \geq 2$ types $\tau_1, \ldots, \tau_m$, in order to obtain a pre-grammar for $\mathsf{Nhabs}(\tau_1) \cap \cdots \cap \mathsf{Nhabs}(\tau_m) = \mathsf{Nhabs}(\tau_1 \cap \cdots \cap \tau_m)$. For illustrative purposes we then instantiate with a type of the family $\{T(n)\}_{n \geq 1}$ considered in [17], where for each $n \geq 1$ type $T(n)$ has a unique inhabitant, which is of size $O(2^n)$.

Definition 6.1 Given types $\tau_1, \ldots, \tau_n \in \mathcal{T}_2^-$ $(n \geq 2)$, we define $\mathsf{N}(\tau_1 \cap \cdots \cap \tau_n) = \mathsf{N}(\tau_1) \times \cdots \times \mathsf{N}(\tau_n)$. Furthermore, let $\mathsf{pre}(\tau_1 \cap \cdots \cap \tau_n)$ denote the smallest set of rules satisfying the following.

- If $m_i := \lambda k_i.p_i \in \mathsf{pre}(\tau_i)$ $(i = 1, \ldots, n)$, then $(m_1, \ldots, m_n) := \lambda(k_1, \ldots, k_n).(p_1, \ldots, p_n) \in \mathsf{pre}(\tau_1 \cap \cdots \cap \tau_n)$;
- if $m_i := k_i\ p_1^i \cdots p_s^i \in \mathsf{pre}(\tau_i)$ for $i = 1, \ldots, n$ and $s \geq 0$, then $(m_1, \ldots, m_n) := (k_1, \ldots, k_n)\ (p_1^1, \ldots, p_1^n) \cdots (p_s^1, \ldots, p_s^n) \in \mathsf{pre}(\tau_1 \cap \cdots \cap \tau_n)$.

Theorem 6.2 *Consider types $\tau_1, \ldots, \tau_n \in \mathcal{T}_2^-$, where $n \geq 2$. Then, one has $\mathsf{Nhabs}(\tau_1 \cap \cdots \cap \tau_n) \neq \emptyset$ if and only if $((\mathsf{n}(\tau_1), \ldots, \mathsf{n}(\tau_n)), \emptyset) \rightsquigarrow^* \epsilon$, with pre-grammar $\mathsf{pre}(\tau_1 \cap \cdots \cap \tau_n)$.*

Proof. For the 'only if' part, suppose that $\mathsf{Nhabs}(\tau_1 \cap \cdots \cap \tau_n) \neq \emptyset$ and consider $M \in \mathsf{Nhabs}(\tau_1 \cap \cdots \cap \tau_n)$. Then, $\vdash_2 M : \tau_i$, for $i = 1, \ldots, n$. On the other hand, every tuple of identifiers $(m_1, \ldots, m_n) \in \mathsf{N}(\tau_1 \cap \cdots \cap \tau_n) = \mathsf{N}(\tau_1) \times \cdots \times \mathsf{N}(\tau_n)$, corresponds directly to a tuple $(\sigma_1, \ldots, \sigma_n)$ of subtypes $\sigma_1, \ldots, \sigma_n$, respectively of $\tau_1, \ldots, \tau_n$, such that the identifier of σ_i in $\mathsf{occT}(\tau_i)$ is m_i, i.e. $\mathsf{n}(\sigma_i) = m_i$ and $\mathsf{t}(m_i) = \sigma_i$. Now, it remains to adapt the argument in the proof of Theorem 5.5, combining it with the fact that:

- $m_i := \lambda k_i.p_i \in \mathsf{pre}(\tau_i)$ for $i = 1, \ldots, n$, implies that there is a rule $(m_1, \ldots, m_n) := \lambda(k_1, \ldots, k_n).(p_1, \ldots, p_n) \in \mathsf{pre}(\tau_1 \cap \cdots \cap \tau_n)$;
- $m_i := k_i\ p_1^i \cdots p_s^i \in \mathsf{pre}(\tau_i)$ for $i = 1, \ldots, n$ and $s \geq 0$, implies that there is a rule $(m_1, \ldots, m_n) := (k_1, \ldots, k_n)\ (p_1^1, \ldots, p_1^n) \cdots (p_s^1, \ldots, p_s^n) \in \mathsf{pre}(\tau_1 \cap \cdots \cap \tau_n)$.

For the 'if' part, following Definition 5.2, we modify the definition of function pair, including a third argument $i \in \{1, \ldots, n\}$, in the following way. We use the abbreviations $\vec{m} = (m_1, \ldots, m_n)$, $\vec{p} = (p_1, \ldots, p_n)$ and $\vec{k} = (k_1, \ldots, k_n)$. Furthermore, $\mathsf{t}_i(k_i)$ will denote the type occurrence with identifier k_i in $\mathsf{occT}(\tau_i)$.

- If $(\vec{m}, V) \rightsquigarrow (\vec{p}, V \cup \{\vec{k}\})$ because $\vec{m} := \lambda\vec{k}.\vec{p} \in \mathsf{pre}(\tau_1 \cap \cdots \cap \tau_n)$, then $\mathsf{pair}(m, V, i) = (\lambda x_{\vec{k}}.N, \Gamma \setminus \{x_{\vec{k}} : \mathsf{t}_i(k_i)\})$, where $(N, \Gamma) = \mathsf{pair}(p, V \cup \{\vec{k}\}, i)$;
- if $(\vec{m}, V) \rightsquigarrow (p_1, V), \ldots, (p_s, V)$ because $\vec{m} := \vec{k}\ \vec{p_1} \cdots \vec{p_s} \in \mathsf{pre}(\tau_1 \cap \cdots \cap \tau_n)$ and $\vec{k} \in V$, then $\mathsf{pair}(m, V) = (x_{\vec{k}}\ N_1 \cdots N_s, \{x_{\vec{k}} : \mathsf{t}_i(k_i)\} \cup \Gamma_1 \cup \cdots \cup \Gamma_s)$, where $(N_i, \Gamma_i) = \mathsf{pair}(n_i, V, i)$, for $1 \leq i \leq s$ $(s \geq 0)$.

Then, following the proof of Lemma 5.3 one shows that for each $i \in \{1, \ldots, n\}$ we have $\vdash_2 M_i : \tau_i$, if $(M_i, \emptyset) = \mathsf{pair}((\mathsf{n}(\tau_1), \ldots, \mathsf{n}(\tau_n)), \emptyset, i)$, for a particular rewriting sequence of $((\mathsf{n}(\tau_1), \ldots, \mathsf{n}(\tau_n)), \emptyset) \rightsquigarrow^* \epsilon$. Now, it remains to observe that the term M_i constructed by function pair is the same for all $i \in \{1, \ldots, n\}$ (as long as the same rewriting sequence is considered). □

Example 6.3 Consider the family $\{T(n)\}_{n\geq 1}$ from [17], where $T(n) = \tau_1 \cap \cdots \cap \tau_n$ and for $1 \leq i \leq n$, $\tau_i = a \to \underbrace{\Psi \to \cdots \to \Psi}_{i-1} \to (a \to b) \to \underbrace{(b \to a) \to \cdots \to (b \to a)}_{n-i} \to b$ and $\Psi = (a \to a) \cap (b \to b)$.

The unique inhabitant of

$$\begin{aligned} T(3) &= \tau_1 \cap \tau_2 \cap \tau_3 \\ &= (a \to (a \to b) \to (b \to a) \to (b \to a) \to b) \quad \cap \\ &\quad (a \to ((a \to a) \cap (b \to b)) \to (a \to b) \to (b \to a) \to b) \quad \cap \\ &\quad (a \to ((a \to a) \cap (b \to b)) \to ((a \to a) \cap (b \to b)) \to (a \to b) \to b) \end{aligned}$$

is

$$M_3 = \lambda x_1 x_2 x_3 x_4 . x_2 (x_3 (x_2 (x_4 (x_2 (x_3 (x_2 x_1)))))).$$

Pre-grammars $\mathsf{pre}(\tau_1)$, $\mathsf{pre}(\tau_2)$ and $\mathsf{pre}(\tau_3)$ are as follows.

$14 := \lambda 0.13$	$18 := \lambda 0.17$	$22 := \lambda 0.21$
$13 := \lambda 8.12$	$17 := \lambda 14.16$	$21 := \lambda 17.20$
$12 := \lambda 9.11$	$16 := \lambda 12.15$	$20 := \lambda 18.19$
$11 := \lambda 10.7$	$15 := \lambda 13.9$	$19 := \lambda 16.11$
$1 := 9\ 3 \mid 10\ 5 \mid 0$	$1,5 := 14\ 1 \mid 13\ 7 \mid 0$	$1,5,9 := 17\ 1 \mid 18\ 5 \mid 0$
$3,5,7 := 8\ 1$	$3,7,9 := 14\ 3 \mid 12\ 5$	$3,7,11 := 17\ 3 \mid 18\ 7 \mid 16\ 9$

From this we obtain a pre-grammar below for $T(3)$, applying the construction in Definition 6.1. Note that we only include rules of the form $(m_1, m_2, m_3) := (n_1, n_2, n_3)\ \ (\_, \_, \_) \cdots (\_, \_, \_)$, if there is some other rule $(m'_1, m'_2, m'_3) := \lambda(n_1, n_2, n_3).(\_, \_, \_)$ in the pre-grammar. Otherwise, this rule would be useless, since there will never be a tuple (n_1, n_2, n_3) in the set of available variables V, during any rewriting sequence starting with $(14, 18, 22)$. So this rule can never be

applied.

$$
\begin{aligned}
(14,18,22) &:= \lambda(0,0,0).(13,17,21) & (3,5,7) &:= (8,14,17)\ (1,1,3)\\
(13,17,21) &:= \lambda(8,14,17).(12,16,20) & (1,1,3) &:= (10,13,16)\ (5,7,9)\\
(12,16,20) &:= \lambda(9,12,18).(11,15,19) & (5,7,9) &:= (8,14,17)\ (1,3,1)\\
(11,15,19) &:= \lambda(10,13,16).(7,9,11) & (1,3,1) &:= (9,12,18)\ (3,5,5)\\
(7,9,11) &:= (8,14,17)\ (1,3,3) & (3,5,5) &:= (8,14,17)\ (1,1,1)\\
(1,3,3) &:= (9,12,18)\ (3,5,7) & (1,1,1) &:= (0,0,0)
\end{aligned}
$$

There is exactly one rewriting sequence $((14,18,22),\emptyset) \rightsquigarrow^* \epsilon$, depicted below.

$$
\begin{aligned}
((14,18,22),\emptyset) &\rightsquigarrow^* ((13,17,21),\{(0,0,0)\})\\
&\rightsquigarrow^* ((12,16,20),\{(0,0,0),(8,14,17)\})\\
&\rightsquigarrow^* ((11,15,19),\{(0,0,0),(8,14,17),(9,12,18)\})\\
&\rightsquigarrow^* ((7,9,11),\{(0,0,0),(8,14,17),(9,12,18),(10,13,16)\})\\
&\rightsquigarrow^* ((1,3,3),\{(0,0,0),(8,14,17),(9,12,18),(10,13,16)\})\\
&\rightsquigarrow^* ((3,5,7),\{(0,0,0),(8,14,17),(9,12,18),(10,13,16)\})\\
&\rightsquigarrow^* ((1,1,3),\{(0,0,0),(8,14,17),(9,12,18),(10,13,16)\})\\
&\rightsquigarrow^* ((5,7,9),\{(0,0,0),(8,14,17),(9,12,18),(10,13,16)\})\\
&\rightsquigarrow^* ((1,3,1),\{(0,0,0),(8,14,17),(9,12,18),(10,13,16)\})\\
&\rightsquigarrow^* ((3,5,5),\{(0,0,0),(8,14,17),(9,12,18),(10,13,16)\})\\
&\rightsquigarrow^* ((1,1,1),\{(0,0,0),(8,14,17),(9,12,18),(10,13,16)\})\\
&\rightsquigarrow^* \epsilon
\end{aligned}
$$

For this sequence, $\mathsf{pair}((14,18,22),\emptyset) \equiv_\alpha M_3$.

7 Conclusions

We studied the notion of pre-grammar in the realm of rank 2 intersection types and addressed the emptiness problem for a rank 2 subsystem. We consider types in $\mathcal{T}_2$ without positive occurrences of intersections, for which the previous methods defined for simple types extend in an elegant way. As future work we aim at extending our methods to the full rank 2 type system as well as exploring other, more expressive, related type systems.

References

[1] João Alpuim, Bruno C. d. S. Oliveira, and Zhiyuan Shi. Disjoint polymorphism. In Hongseok Yang, editor, *Programming Languages and Systems - 26th European Symposium on Programming, ESOP 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, volume 10201 of *Lecture Notes in Computer Science*, pages 1–28. Springer, 2017.

[2] S. Alves and S. Broda. Inhabitation machines: determinism and principality. In *Ninth Workshop on Non-Classical Models of Automata and Applications, NCMA 2017*, pages 57–70, 2017.

[3] S. Alves and S. Broda. A unifying framework for type inhabitation. In *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018)*, Leibniz International Proceedings in Informatics (LIPIcs), 2018.

[4] Ch. Ben-Yelles. *Type Assignment in the Lambda-Calculus: Syntax and Semantics*. PhD thesis, University College of Swansea, September 1979.

[5] Flavien Breuvart and Ugo Dal Lago. On intersection types and probabilistic lambda calculi. In *Proceedings of the 20th International Symposium on Principles and Practice of Declarative Programming*, PPDP '18, pages 8:1–8:13, New York, NY, USA, 2018. ACM.

[6] S. Broda and L. Damas. On long normal inhabitants of a type. *Journal of Logic and Computation*, 15:353–390, June 2005.

[7] A. Bucciarelli, D. Kesner, and S. Ronchi Della Rocca. The inhabitation problem for non-idempotent intersection types. In *Theoretical Computer Science - 8th IFIP TC 1/WG 2.2 International Conference, TCS*, volume 8705 of *Lecture Notes in Computer Science*, pages 341–354. Springer, 2014.

[8] M.W. Bunder. Proof finding algorithms for implicational logics. *Theoretical Computer Science*, 232(12):165 – 186, 2000.

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

[10] M. Dezani-Ciancaglini, F. Honsell, and Y. Motohama. Compositional characterisations of *lambda*-terms using intersection types. *Theor. Comput. Sci.*, 340(3):459–495, 2005.

[11] A. Dudenhefner and J.Rehof. Typability in bounded dimension. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12. IEEE Computer Society, 2017.

[12] A. Dudenhefner and J. Rehof. Intersection type calculi of bounded dimension. In G. Castagna and A. D. Gordon, editors, *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, pages 653–665. ACM, 2017.

[13] J.R. Hindley. *Basic Simple Type Theory*, volume 42 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1997.

[14] H. Hüttel, I. Lanese, V. T. Vasconcelos, L. Caires, M. Carbone, P.-M. Deniélou, D. Mostrous, L. Padovani, A. Ravara, E. Tuosto, H. Torres Vieira, and G.Zavattaro. Foundations of session types and behavioural contracts. *ACM Comput. Surv.*, 49(1):3:1–3:36, 2016.

[15] T. Jim. Rank 2 type systems and recursive definitions, 1995.

[16] T. Jim. What are principal typings and what are they good for? In *Proceedings of the 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '96, pages 42–53, New York, NY, USA, 1996. ACM.

[17] D. Kusmierek. The inhabitation problem for rank two intersection types. In S. Ronchi Della Rocca, editor, *Typed Lambda Calculi and Applications, 8th International Conference, TLCA 2007, Paris, France, June 26-28, 2007, Proceedings*, volume 4583 of *Lecture Notes in Computer Science*, pages 240–254. Springer, 2007.

[18] D. Leivant. Polymorphic type inference. In *Proceedings of Principles of Programming Languages (POPL'83)*, pages 88–98, New York, NY, USA, 1983. ACM Press.

[19] A. Schubert, W. Dekkers, and H.P. Barendregt. Automata theoretic account of proof search. In *CSL 2015*, pages 128–143, 2015.

[20] R. Statman. Intuitionistic propositional logic is polynomial-space complete. *Theoretical Computer Science*, 9:67–72, 1979.

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

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

[23] P. Urzyczyn. Inhabitation of low-rank intersection types. In *TLCA'09*, volume 5608 of *LNCS*, pages 356–370. Springer, 2009.

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

[25] S. van Bakel. *Intersection Type Disciplines in Lambda Calculus and Applicative Term Rewriting Systems*. PhD thesis, Department of Computer Science, University of Nijmegen, 1993.

[26] S. van Bakel. Rank 2 intersection type assignment in term rewriting systems. *Fundamenta Informaticae*, 26(2):141–166, April 1996.