On Average-Case Hardness of Higher-Order Model Checking

Yoshiki Nakamura
Tokyo Institute of Technology, Japan

Kazuyuki Asada
Tohoku University, Japan

Naoki Kobayashi
The University of Tokyo, Japan

Ryoma Sin'ya
Akita University, Japan

Takeshi Tsukada
The University of Tokyo, Japan

Abstract

We study a mixture between the *average* case and worst case complexities of higher-order model checking, the problem of deciding whether the tree generated by a given λY-term (or equivalently, a higher-order recursion scheme) satisfies the property expressed by a given tree automaton. Higher-order model checking has recently been studied extensively in the context of higher-order program verification. Although the worst-case complexity of the problem is k-EXPTIME complete for order-k terms, various higher-order model checkers have been developed that run efficiently for typical inputs, and program verification tools have been constructed on top of them. One may, therefore, hope that higher-order model checking can be solved efficiently in the *average* case, despite the worst-case complexity. We provide a negative result, by showing that, under certain assumptions, for almost every term, the higher-order model checking problem specialized for the term is k-EXPTIME hard with respect to the size of automata. The proof is based on a novel intersection type system that characterizes terms that do not contain any useless subterms.

2012 ACM Subject Classification Theory of computation → Program verification

Keywords and phrases Higher-order model checking, Average-case complexity, Intersection type system, Useless analysis

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Related Version A full version of the paper is available at [20].

1 Introduction

Higher-order model checking [12, 23, 25] asks whether the (possibly infinite) tree generated by a given λY-term (or equivalently, a higher-order recursion scheme) is accepted by a given tree automaton. The problem was shown to be decidable by Ong in 2006 [23], and has been applied to higher-order program verification [15, 16, 22, 19]. Although the worst-case complexity of higher-order model checking is k-EXPTIME complete (where k is the type-theoretic order of the given λY-term), practical higher-order model checkers have been developed that run fast for many typical inputs. They lead to the development of various automated verification tools for higher-order functional programs.

In view of the situation above, we are interested in the following question: why do higher-order model checkers run efficiently, despite the extremely high worst case complexity? There are a couple of known reasons. First, the worst-case time complexity of higher-order model checking is actually polynomial in the size of a given term, provided that the other

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:36
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

parameters (the largest order and arity of functions, and the size of an automaton) are fixed [18]. Second, linear functions do not blow up the complexity [5]. These reasons alone, however, do not fully explain why higher-order model checking works in practice. For example, for the first point above, the constant factor determined by the other parameters is huge.

In the present paper, we consider another possibility: higher-order model checking may actually be easy in the *average* case; in other words, it may be the case that hard instances that cost k-EXPTIME are sparse and many of the instances of higher-order model checking can be solved more efficiently. We give a somewhat negative result on that possibility. For each term t of the λY-calculus, we consider the following higher-order model checking problem specialized to t:

HOMC$(t,\cdot)$: Given a tree automaton $\mathcal{A}$, decide whether the tree generated by t is accepted by $\mathcal{A}$.

Our main result is that for *almost every* term t of order-k that is sufficiently large, HOMC$(t,\cdot)$ is k-EXPTIME hard. A little more precisely, we prove that, for the set $\mathtt{Terms}_{n,k}$ of terms of size n and order k (modulo certain additional conditions that we explain later), the ratio of "hard" terms:

$$\frac{\#\{t \in \mathtt{Terms}_{n,k} \mid \mathrm{HOMC}(t,\cdot) \text{ is } k\text{-EXPTIME hard}\}}{\#\mathtt{Terms}_{n,k}}$$

tends to 1 if $n \to \infty$ (where $\#S$ denotes the cardinality of a set S). In other words, if we pick up a term randomly according to the uniform distribution over $\mathtt{Terms}_{n,k}$, it is likely that there exists a bad automaton $\mathcal{A}$ such that HOMC$(t,\mathcal{A})$ is very hard. Note that this is a mixture between the average case and worst-case analysis: the result above says that in the *average case* on the choice of a term t, the complexity of HOMC$(t,\cdot)$ is k-EXPTIME hard in the *worst-case* on the choice of an automaton.

In order to make the above analysis meaningful, we have to carefully define the set $\mathtt{Terms}_{n,k}$ of terms. To see why, consider a term of the form $(\lambda x.\mathtt{c})t$, where $\mathtt{c}$ is a nullary tree constructor. The term generates the singleton tree $\mathtt{c}$; so, no matter how large t is, the problem HOMC$((\lambda x.\mathtt{c})t,\cdot)$ is easy. Thus, if we include such terms in $\mathtt{Terms}_{n,k}$, the ratio of hard instances above would not be 1 for the trivial reason. In the context of applications of higher-order model checking to program verification, however, such instances are unlikely to appear: a λY-term corresponds to a program, and it is unlikely that one writes a program that contains such a huge useless term t. (It might be the case for machine-generated programs, but even in that case, one can apply simple preprocessing to remove such useless terms before invoking a costly higher-order model checking algorithm.) We, therefore, exclude out, from $\mathtt{Terms}_{n,k}$, terms that contain any useless subterms. Here, a subterm t_1 of t is useless if replacing t_1 with another term never changes the tree generated by t. (We will impose further conditions such as the number of variables, which will be explained in Section 2.)

Once the set $\mathtt{Terms}_{n,k}$ is properly chosen as explained above, our main result can be proved as follows. First, according to Kobayashi and Ong's work on the complexity of higher-order model checking [17], there exists an order-k "hard" term $t_{\mathrm{HARD},k}$ such that HOMC$(t_{\mathrm{HARD},k},\cdot)$ is k-EXPTIME complete. Second, according to Asada et al.'s work on quantitative analysis on λ-terms [1], any sufficiently large term t can be decomposed to the form $E[C_1,\dots,C_m]$ for sufficiently many contexts $C_1,\dots,C_m$, where each C_i is large enough to be replaced by a context, say C'_i, that contains the hard term $t_{\mathrm{HARD},k}$, without changing the term size. Thus, by using their argument (which originates from the so called "infinite monkey theorem"), we can deduce that almost every sufficiently large term contains the hard term $t_{\mathrm{HARD},k}$, *if we ignore the condition that useless terms should be excluded.* Finally (and most importantly), we can choose the context C'_i that contains the hard term, so that if $E[C_1, \ldots, C_i, \ldots, C_m]$ belongs to $\mathtt{Terms}_{n,k}$ (and therefore does not contain any useless subterms), then so does $E[C_1, \ldots, C'_i, \ldots, C_m]$.

To obtain the last part of the result, we develop a novel intersection type system that completely characterizes the set of terms that do not contain useless terms, in the sense that a closed term t is typable if and only if t does not contain any useless term. This type system is one of the main contributions of the present paper, and may be of independent interest. Type systems for useless code elimination have been studied before [6, 7, 13] (in particular, Damiani [7] used intersection types), but the complete characterization was not known, to our knowledge.

The rest of this paper is structured as follows. Section 2 provides formal definitions of λY-terms and the higher-oder model checking. Section 3 states our main result and gives an proof outline. Sections 4–6 prove the theorem. Section 7 discusses related work, and Section 8 concludes this article.

2 Preliminaries

For a map f, we write $\mathrm{dom}(f)$ for the domain of f and $\mathrm{rng}(f)$ for the range of f. We denote by $\mathbb{N}$ the set of non-negative integers and by $\mathbb{N}_+$ the set of positive integers. For $m, n \in \mathbb{N}$, we write $[m, n]$ for the set $\{i \in \mathbb{N} \mid m \leq i \leq n\}$, and $[n]$ for $[1, n]$; note that $[0] = \emptyset$. The cardinality of a set A is denoted by $\#(A)$. We use $A \uplus B$ instead of $A \cup B$ if sets A and B are disjoint. For a set A, we write A^* for the set of finite sequences consisting of elements of A. An L-*labeled tree* is a partial map T from $\mathbb{N}_+^*$ to L such that, for every $\langle \alpha, i\rangle \in \mathbb{N}_+^* \times \mathbb{N}_+$, if $\alpha \cdot i \in \mathrm{dom}(T)$, then $\{\alpha, \alpha \cdot 1, \ldots, \alpha \cdot (i-1)\} \subseteq \mathrm{dom}(T)$. An L-labeled tree T is called *finite* if $\mathrm{dom}(T)$ is finite. We write $\mathtt{r}_T(\alpha)$ for the number of children of a node α in T, i.e., $\mathtt{r}_T(\alpha) = \#\{i \in \mathbb{N}_+ \mid \alpha \cdot i \in \mathrm{dom}(T)\}$. A *ranked alphabet* Σ is a map from a finite set of symbols to $\mathbb{N}$. We call $\Sigma(a)$ the *rank* of a. A $\mathrm{dom}(\Sigma)$-labeled tree T is called a Σ-*ranked tree* (Σ-*tree*, for short) if, for every $\alpha \in \mathrm{dom}(T)$, $\mathtt{r}_T(\alpha) = \Sigma(T(\alpha))$.

2.1 λY-Terms as Tree Generators

In this subsection, we introduce (simply-typed) λY-terms [28] as generators of (possibly infinite) Σ-trees. In the context of higher-order model checking, higher-order recursion schemes have originally been used as generators of trees [12, 23], but the λY-terms (with constants of order up to 1 as tree constructors), which is equi-expressive as tree generators (see, e.g., [26]), have also been used in later studies on higher-order model checking [25]. For the purpose of the present paper, we find it more convenient to use λY-terms.

Let Σ be a ranked alphabet. Each $a \in \mathrm{dom}(\Sigma)$ is called a *tree constructor*. We use meta-variables a, b, c for tree constructors (and $\mathtt{a}, \mathtt{b}, \mathtt{c}, \ldots$ for concrete symbols). The set of *simple types* is defined by: $\kappa ::= \mathtt{o} \mid \kappa_1 \to \kappa_2$. The *ground type* $\mathtt{o}$ is the type of trees. The *order* and *arity* of a simple type κ, written $\mathtt{ord}(\kappa)$ and $\mathtt{ar}(\kappa)$ respectively, are defined by: $\mathtt{ord}(\kappa_1 \to \cdots \to \kappa_n \to \mathtt{o}) \triangleq \max(\{0\} \cup \{\mathtt{ord}(\kappa_i) + 1 \mid 1 \leq i \leq n\})$ and $\mathtt{ar}(\kappa_1 \to \cdots \to \kappa_n \to \mathtt{o}) \triangleq n$, where $n \geq 0$. Let $\mathcal{V}$ be a countably infinite set, which is ranged over by x, y, z.

▶ **Definition 1** (λY-terms)**.** *The set of* (λY-)terms *(over Σ) is defined by:*

$$t ::= x^\kappa \mid \lambda x^\kappa.t \mid \lambda\_^\kappa.t \mid t_1\, t_2 \mid \mathbf{Y}^\kappa t \mid a(t_1, \ldots, t_{\Sigma(a)}) \mid \bot^\kappa.$$

We call elements of $\mathcal{V} \uplus \{\_\}$ *variables* and use meta-variables $\bar{x}, \bar{y}, \bar{z}$ for them. As in the standard λY-calculus, the constructor $\mathbf{Y}^\kappa$ may be considered a fixpoint operator of type

$(\kappa \to \kappa) \to \kappa$. The special variable '_' denotes an unused variable (hence can occur only in a binder, not in the body of a function). For each type κ, we have a special term $\perp^\kappa$, which intuitively represents an unused term and will play an important role in the definition of minimal terms. We often omit type annotations (for example, $\lambda x^\kappa.x^\kappa$ is just written $\lambda x.x$). For a term t, we write $\mathbf{FV}(t)$ for the set of all the free variables of t.

A *simple type environment* Γ is a finite partial map from $\mathcal{V}$ (recall that the special variable _ does not belong to $\mathcal{V}$) to the set of simple types. We simply write $\Gamma, x : \kappa$ for $\Gamma \uplus \{x \mapsto \kappa\}$. The type judgment relation $\Gamma \vdash_{\mathrm{ST}} t : \kappa$ is inductively defined by the following rules:

$$\frac{}{x : \kappa \vdash_{\mathrm{ST}} x^\kappa : \kappa}(\mathrm{Var}) \quad \frac{\Gamma, x : \kappa \vdash_{\mathrm{ST}} t : \kappa'}{\Gamma \vdash_{\mathrm{ST}} \lambda x^\kappa.t : \kappa \to \kappa'}(\mathrm{Abs1}) \quad \frac{\Gamma \vdash_{\mathrm{ST}} t : \kappa'}{\Gamma \vdash_{\mathrm{ST}} \lambda \bar{x}^\kappa.t : \kappa \to \kappa'}(\mathrm{Abs2}) \quad \frac{}{\emptyset \vdash_{\mathrm{ST}} \perp^\kappa : \kappa}(\perp)$$

$$\frac{\Gamma_1 \vdash_{\mathrm{ST}} t : \kappa \to \kappa' \quad \Gamma_2 \vdash_{\mathrm{ST}} s : \kappa}{\Gamma_1 \cup \Gamma_2 \vdash_{\mathrm{ST}} t\, s : \kappa'}(\mathrm{App}) \quad \frac{\Gamma_1 \vdash_{\mathrm{ST}} t_1 : \mathtt{o} \ \ldots \ \Gamma_n \vdash_{\mathrm{ST}} t_n : \mathtt{o}}{\bigcup_{i\in[n]} \Gamma_i \vdash_{\mathrm{ST}} a(t_1, \ldots, t_n) : \mathtt{o}}(a) \quad \frac{\Gamma \vdash_{\mathrm{ST}} t : \kappa \to \kappa}{\Gamma \vdash_{\mathrm{ST}} \mathbf{Y}^\kappa\, t : \kappa}(\mathbf{Y})$$

Henceforth, we only consider *well-typed* terms (i.e., terms t such that $\Gamma \vdash_{\mathrm{ST}} t : \kappa$ for some $\langle \Gamma, \kappa \rangle$). Note that for every well-typed term t, there is a unique pair $\langle \Gamma, \kappa \rangle$ such that $\Gamma \vdash_{\mathrm{ST}} t : \kappa$; and moreover, its derivation tree is also uniquely determined. We sometimes annotate a term with its type, like t^κ, when t has type κ (under a certain type environment). We say that t is *closed* if $\Gamma = \emptyset$; and that t is *ground-typed* if $\kappa = \mathtt{o}$.

► **Definition 2.** *The* (call-by-name) reduction relation $\longrightarrow$ *is defined as the least binary relation on well-typed terms (up to α-sequivalence) closed under the following rules, where we write $t\{s/x\}$ for the term obtained from t by substituting s for all the free occurrences of x in a capture-avoiding manner:*

(β) $(\lambda \bar{x}.t)\, s \longrightarrow t\{s/\bar{x}\}$*;* $(\mathbf{Y})$ $\mathbf{Y}t \longrightarrow t\,(\mathbf{Y}t)$*;* $(\perp)$ $\perp^{\kappa_1 \to \kappa_2} t \longrightarrow \perp^{\kappa_2}$*;*
(App) $tu \longrightarrow t'u$ *if* $t \longrightarrow t'$*;* (a) $a(t_1, \ldots, t_n) \longrightarrow a(t_1, \ldots, t_{i-1}, t'_i, t_{i+1}, \ldots, t_n)$ *if* $t_i \longrightarrow t'_i$*.*

We write $\longrightarrow^*$ for the reflexive transitive closure of $\longrightarrow$.

The *tree generated by a closed and ground λY-term t* is the one obtained from t by (possibly) infinite rewriting with respect to the above reduction relation. The precise definition is given below.

We write $\Sigma^\perp$ for the ranked alphabet $\Sigma \uplus \{\perp \mapsto 0\}$. We define *the binary relation $\sqsubseteq$ on $\Sigma^\perp$-trees* by: $T_1 \sqsubseteq T_2$ if and only if (i) $\mathrm{dom}(T_1) \subseteq \mathrm{dom}(T_2)$ and (ii) for every $\alpha \in \mathrm{dom}(T_1)$, $T_1(\alpha) = \perp$ or $T_1(\alpha) = T_2(\alpha)$. We write $T_1 \sqsubset T_2$ if $T_1 \sqsubseteq T_2$ and $T_1 \neq T_2$. We denote the join of $\{T_i\}_{i\in I}$ on $\sqsubseteq$ by $\bigsqcup_{i\in I} T_i$ if defined.

A term consisting of only tree constructors and $\perp^{\mathtt{o}}$ can naturally be regarded as a $\Sigma^\perp$-tree. For example, $\mathtt{b}(\mathtt{c}, \mathtt{a}(\perp^{\mathtt{o}}))$ can be regarded as the $\Sigma^\perp$-tree: $\{\epsilon \mapsto \mathtt{b}, 1 \mapsto \mathtt{c}, 2 \mapsto \mathtt{a}, 2 \cdot 1 \mapsto \perp\}$; hence we identify finite trees and terms consisting of tree constructors and $\perp^{\mathtt{o}}$ below. For each closed and ground-typed term t, the $\Sigma^\perp$-tree $t^\perp$ is defined by: $t^\perp \triangleq a(t_1^\perp, \ldots, t_{\Sigma(a)}^\perp)$ if $t = a(t_1, \ldots, t_{\Sigma(a)})$; and $t^\perp \triangleq \perp$ otherwise. The *value tree* of a closed and ground-typed term t, written $T(t)$, is defined by: $T(t) \triangleq \bigsqcup\{s^\perp \mid t \longrightarrow^* s\}$. For example, consider the value tree of $(\mathbf{Y}t_1)\mathtt{c}$ where $t_1 = \lambda f^{\mathtt{o}\to\mathtt{o}}.\lambda x^{\mathtt{o}}.\mathtt{b}(x, f(\mathtt{a}(x)))$. By applying the reduction rules $(\mathbf{Y})$ and (β), we can obtain the following reduction sequence

$$(\mathbf{Y}t_1)\mathtt{c} \longrightarrow t_1(\mathbf{Y}t_1)\mathtt{c} \longrightarrow^* \mathtt{b}(\mathtt{c}, (\mathbf{Y}t_1)(\mathtt{a}(\mathtt{c}))) \longrightarrow^* \mathtt{b}(\mathtt{c}, \mathtt{b}(\mathtt{a}(\mathtt{c}), (\mathbf{Y}t_1)(\mathtt{a}(\mathtt{a}(\mathtt{c})))))$$

and observe that $T(t)$ is the infinite tree of the form $\mathtt{b}(\mathtt{c}, \mathtt{b}(\mathtt{a}(\mathtt{c}), \mathtt{b}(\mathtt{a}(\mathtt{a}(\mathtt{c})), \mathtt{b}(\cdots))))$.

We also define the size and order of a term, which will be used in the complexity analysis.

▶ **Definition 3** (size, order). *The* size *of a term t is defined by:* $|x| = |\bot| \triangleq 1$, $|\lambda \bar{x}.t| = |\mathbf{Y}t| \triangleq 1 + |t|$, $|t_1\, t_2| \triangleq 1 + |t_1| + |t_2|$*, and* $|a(t_1, \ldots, t_{\Sigma(a)})| \triangleq 1 + \sum_{i\in[\Sigma(a)]} |t_i|$*. The* order *of a term t, written* $\mathtt{ord}\,(t)$*, is defined by:*

$$\mathtt{ord}\,(t) \triangleq \max(\{0\} \cup \{\mathtt{ord}\,(\kappa) \mid \lambda x^{\kappa}.s \textit{ or } \mathbf{Y}^{\kappa} s \textit{ is a subterm of } t\}).$$

Note that the size of a variable is a constant; this is appropriate in our context, as we fix the number of variables in the main theorem (Theorem 6).

2.2 Higher-Order Model Checking

We assume the notion of *alternating parity tree automaton* (*APT* for short): see, e.g., [10]. A formal definition of APT can be found in Appendix A; but the precise definition of APT is unnecessary for understanding our technical development in later sections, once you admit the results in this subsection. We recall the definition of higher-order model checking.

▶ **Definition 4** (higher-order model checking problem). *The* higher-order model checking problem*, written* $\mathrm{HOMC}\,(\cdot,\cdot)$*, is the problem of, given a closed and ground-typed λY-term t over Σ and an APT $\mathcal{A}$ over Σ as input, deciding whether $\mathcal{A}$ accepts $T(t)$. We write* $\mathrm{HOMC}_k(\cdot,\cdot)$ *when the first input is restricted to a term of order-k. We denote by* $\mathrm{HOMC}\,(t,\cdot)$ *the problem obtained by fixing the first input to t, i.e., the problem of, given an APT $\mathcal{A}$ as input, deciding whether $\mathcal{A}$ accepts $T(t)$.*

Ong [21] has shown that the $\mathrm{HOMC}_k(\cdot,\cdot)$ is k-EXPTIME complete (combined complexity) for each $k \geq 0$. The following theorem states the complexity of $\mathrm{HOMC}\,(t,\cdot)$, which serves as a basis of the present work.

▶ **Theorem 5** ([17, Theorem 3.8] for (2)). *For each $k \geq 1$,*

(1) *for every order-k λY-term t,* $\mathrm{HOMC}\,(t,\cdot)$ *is decidable in k-*EXPTIME*; and*
(2) *for some order-k λY-term $t_{\mathrm{HARD},k}$,* $\mathrm{HOMC}\,(t_{\mathrm{HARD},k},\cdot)$ *is k-*EXPTIME *hard.*

3 Main Theorem

This section formally states the main result of the paper: for almost every order-k λY-term, the higher-order model checking problem $\mathrm{HOMC}(t,\cdot)$ is k-EXPTIME hard, under a certain assumption, and sketches an overall structure of the proof. We first prepare some auxiliary notations. We denote by $[t]_\alpha$ the α-equivalence class of t. In our quantitative analysis, we count α-equivalent terms at most once (e.g., we do not distinguish $(\lambda x.\lambda y.x)z$ and $(\lambda z.\lambda\_.z)z$). We define $\#\mathtt{vars}\,(t) \triangleq \min\{\#(\mathbf{V}(t')) \mid t' \in [t]_\alpha\}$, where $\mathbf{V}(t)$ denotes the set of all the variables (except _) occurring in t. Namely, $\#\mathtt{vars}\,(t)$ means the *minimum number of variables* occurring in term t, up to α-equivalence. For example, $\#\mathtt{vars}\,((\lambda x.\lambda y.x)z) = 1$ since the term is α-equivalent to $(\lambda z.\lambda\_.z)z$. Also the *internal arity* of a term t, written $\mathtt{iar}\,(t)$, is defined by: $\mathtt{iar}\,(t) \triangleq \max(\{\mathtt{ar}\,(\kappa) \mid s^{\kappa} \text{ is a subterm of } t\})$.

Let $\hat{\Lambda}_n(k,\iota,\xi)$ be the set of all (α-equivalence classes of) closed and ground-typed λY-terms such that[1] (i) the size is n (i.e., $|t| = n$); (ii) the order is up to k (i.e., $\mathtt{ord}\,(t) \leq k$); (iii) the internal arity is up to ι (i.e., $\mathtt{iar}\,(t) \leq \iota$); (iv) the number of variable names is up to ξ (i.e., $\#\mathtt{vars}\,(t) \leq \xi$); and (v) the terms are *minimal* (see Section 3.1 below for the definition).

The main theorem is stated as follows.

[1] The set $\hat{\Lambda}_n(k,\iota,\xi)$ implicitly depends on the choice of ranked alphabet Σ. The main theorem holds independently of the choice of Σ unless Σ is unreasonably small.

▶ **Theorem 6** (main theorem). *For each $k \geq 1$, let ι and ξ be sufficiently large natural numbers. Then,*

$$\lim_{n\to\infty} \frac{\#\Big(\{t \in \hat{\Lambda}_n(k,\iota,\xi) \mid \mathrm{HOMC}(t,\cdot) \text{ is } k\text{-}\mathsf{EXPTIME} \text{ hard}\}\Big)}{\#\Big(\hat{\Lambda}_n(k,\iota,\xi)\Big)} = 1.$$

Below we first define the minimality in Section 3.1 and give a proof outline in Section 3.2.

3.1 Minimal Terms

Intuitively, a term is *minimal* if it has no useless subterm. The formal definition of *minimal term* is given as follows. We define the relation $\sqsubseteq$ on *terms*, which is analogous to the corresponding relation ($\sqsubseteq$) on trees.

▶ **Definition 7.** *The* approximate relation $\sqsubseteq$ *is the least binary relation on (well-typed) terms closed under the following rules: $\bot^\kappa \sqsubseteq t^\kappa$; $x^\kappa \sqsubseteq x^\kappa$; if $t_1 \sqsubseteq s_1$ and $t_2 \sqsubseteq s_2$, then $t_1\, t_2 \sqsubseteq s_1\, s_2$; if $t \sqsubseteq s$, then $\lambda \bar{x}^\kappa.t \sqsubseteq \lambda \bar{x}^\kappa.s$; if $t \sqsubseteq s$, then $\mathbf{Y}^\kappa t \sqsubseteq \mathbf{Y}^\kappa s$; and if $t_i \sqsubseteq s_i$ for every $i \in [\Sigma(a)]$, then $a(t_1,\dots,t_{\Sigma(a)}) \sqsubseteq a(s_1,\dots,s_{\Sigma(a)})$.*

In other words, $s \sqsubseteq t$ means that s is obtained from t by replacing subterms $t_1^{\kappa_1},\dots,t_n^{\kappa_n}$ with $\bot^{\kappa_1},\dots,\bot^{\kappa_n}$. We write $s \sqsubset t$ if $s \sqsubseteq t$ and $s \neq t$. We denote the join of $\{t_i\}_{i\in I}$ on $\sqsubseteq$ by $\bigsqcup_{i\in I} t_i$ if defined, and we sometimes write $t_1 \sqcup \ldots \sqcup t_n$ for $\bigsqcup_{i\in[n]} t_i$. With respect to $\Sigma^\bot$-tree terms, the relation $\sqsubseteq$ on terms is equivalent to the relation $\sqsubseteq$ on $\Sigma^\bot$-trees.

▶ **Definition 8.** *A closed and ground-typed term t is* minimal *if for every $s \sqsubset t$, $T(s) \neq T(t)$.*[2]

In other words, a term t is *not minimal* if there exists s obtained by replacing a non-$\bot$ subterm u of t with $\bot$ such that $T(s) = T(t)$.

▶ **Example 9.** Let $t = (\lambda x.\lambda y.x)\ \mathtt{a}\ u$. Then the value tree is the finite tree expressed by the term $\mathtt{a}$ (since $(\lambda x.\lambda y.x)\ \mathtt{a}\ u \longrightarrow (\lambda y.\mathtt{a})\ u \longrightarrow \mathtt{a}$). Note that, for generating the value tree of the above term, the subterm u is "unused". In fact, if $u \neq \bot$, then t is not minimal as expected. This is because $s = (\lambda x.\lambda y.x)\ \mathtt{a}\ \bot \sqsubset t$ but $T(s) = T(t)$. The term s is minimal.

The following proposition gives an important property of minimal term. We write $t' \preceq t$ when t' is a subterm of a term t.

▶ **Proposition 10.** *Let t be a closed and ground-typed term. If t is minimal, then for every non-$\bot$, closed and ground-typed subterm $s \preceq t$, its value tree $T(s)$ is a subtree of $T(t)$.*

This property is intuitively obvious. Since t is minimal, the subterm s assumed to be non-$\bot$ must be used in the computation of the value tree $T(t)$. As s is closed and ground-typed, the only way to use s is to place its value tree $T(s)$ somewhere in $T(t)$; hence the proposition. For a formal proof, see Appendix G in the full version [20].

3.2 Proof Outline

For each k, let $t_{\mathrm{HARD},k}$ be an order-k closed and ground-typed term such that the problem $\mathrm{HOMC}(t,\cdot)$ is k-$\mathsf{EXPTIME}$ hard. Such $t_{\mathrm{HARD},k}$ always exists by Theorem 5 (2). We can

[2] Here $T(s) \neq T(t)$ is equivalent to $T(s) \sqsubset T(t)$. It is because $s \sqsubseteq t$ implies $T(s) \sqsubseteq T(t)$ for every s and t.

assume without loss of generality that $t_{\mathrm{HARD},k}$ is minimal; otherwise take a minimal element $t'_{\mathrm{HARD},k}$ of $\{s \mid T(s) = T(t_{\mathrm{HARD},k})\}$. The proof idea of Theorem 6 is fairly simple, and can be divided into two parts. We will show that (a) for each order k, every order-k minimal term containing the "hard" term $t_{\mathrm{HARD},k}$ as a subterm yields k-EXPTIME-hardness for the higher-order model checking problem, and (b) almost every minimal term of order-k contains the "hard" term $t_{\mathrm{HARD},k}$ as a subterm. The ideas (a) and (b) are formalized as the following Lemma 11 and Lemma 12, respectively.

▶ **Lemma 11.** *Let $k \geq 1$. For every* minimal *λY-term $t \succeq t_{\mathrm{HARD},k}$,* $\mathrm{HOMC}\,(t,\cdot)$ *is k-*EXPTIME *hard.*

▶ **Lemma 12.** *For each $k \geq 1$, let ι and ξ be sufficiently large natural numbers. Then,*

$$\lim_{n\to\infty} \frac{\#\Big(\{t \in \hat{\Lambda}_n(k,\iota,\xi) \mid t \succeq t_{\mathrm{HARD},k}\}\Big)}{\#\Big(\hat{\Lambda}_n(k,\iota,\xi)\Big)} = 1.$$

Theorem 6 follows immediately from the two lemmas above. Lemma 11 is relatively easily proved as follows.

Proof (of Lemma 11). Assume that $t \succeq t_{\mathrm{HARD},k}$. Then $T(t) \succeq T(t_{\mathrm{HARD},k})$ by Proposition 10, i.e. $T(t_{\mathrm{HARD},k}) = (T(t)\restriction_\alpha)$ for some $\alpha \in \mathrm{dom}(T(t))$ where $(T\restriction_\alpha)$ denotes the subtree of T induced by the node α. Let c be the length of α. For any APT $\mathcal{A}$, we can construct an automaton $\mathcal{A}\restriction_\alpha$ by adding c states to $\mathcal{A}$ and replacing the initial state so that $\mathcal{A}\restriction_\alpha$ accepts T if and only if $\mathcal{A}$ accepts $T\restriction_\alpha$ (intuitively, $\mathcal{A}\restriction_\alpha$ first moves to the node α then behaves like $\mathcal{A}$). Then the polynomial-time function $\mathcal{A} \mapsto (\mathcal{A}\restriction_\alpha)$ gives a polynomial-time reduction from $\mathrm{HOMC}\,(t_{\mathrm{HARD},k},\cdot)$ to $\mathrm{HOMC}\,(t,\cdot)$ The lemma follows from k-EXPTIME-hardness of $\mathrm{HOMC}\,(t_{\mathrm{HARD},k},\cdot)$. ◀

The remaining part is to show Lemma 12. To prove it, we introduce the following lemma (where the precise definition of *second-order context* will be given in Section 4).

▶ **Lemma 13.** *Let $k \geq 1$. For each k, let ι and ξ be sufficiently large natural numbers. There is m such that the following holds: Let $n \geq m$, E be any second-order linear context, and C be any affine context of $|C| \geq m$ such that $E[C] \in \hat{\Lambda}_n(k,\iota,\xi)$. Then there is an affine context $D \succeq t_{\mathrm{HARD},k}$ such that $E[D] \in \hat{\Lambda}_n(k,\iota,\xi)$.*

We show how Lemma 12 follows from Lemma 13 in Section 4. We then introduce a new intersection type system that characterizes the minimality in Section 5, and use it to prove Lemma 13 in Section 6.

4 Infinite Monkey Theorem for Minimal Terms

Our proof of Lemma 12 is alangous to that of the following classical so-called *infinite monkey theorem* (a.k.a. "Borges's theorem" [9, p.61, Note I.35]) for words:

▶ **Theorem 14.** *Let Σ be a finite alphabet. For any word $x \in \Sigma^*$, almost all words contain x as a subword, i.e.*

$$\lim_{n\to\infty} \frac{\#(\{w \in \Sigma^n \mid w = uxv \text{ for some } u,v \in \Sigma^*\})}{\#(\Sigma^n)} = 1.$$

The theorem above follows from the following reasoning: Any word w can be decomposed to the form $w_1 w_2 \cdots w_p w'$ where $|w_i| = |x|$ and $|w'| < |x|$. If we pick w randomly, the probability that w_i coincides with x is $(\frac{1}{|\Sigma|})^{|x|}$; hence the probability that w contains x is at least $1 - (1 - (\frac{1}{|\Sigma|})^{|x|})^p$, which tends to 1 when n tends to infinity. For the purpose of proving Lemma 12, we analogously decompose each term t to the form $E[C_1, \ldots, C_p]$ (where E and C_i respectively correspond to w' and w_i above), by using the tree decomposition in [1]. Below, we first recall the tree decomposition of [1] (adapted to our setting) in Section 4.1. We then prove Lemma 12, modulo Lemma 13.

4.1 Decomposition of Terms

In this subsection, we recall the decomposition function $\Phi_m(\cdot)$ given in [1] and its properties. Hereafter we regard the set of λY-terms $\hat{\Lambda}(k, \iota, \xi)$ over Σ as $\Sigma_{\Lambda(k,\iota,\xi)}$-trees where $\Sigma_{\Lambda(k,\iota,\xi)}$ is an extension of Σ defined by:

$$\begin{aligned}
\Sigma_{\Lambda(k,\iota,\xi)} \triangleq \Sigma &\uplus \{x \mapsto 0 \mid x \in \mathcal{V}_\xi\} \\
&\uplus \{\lambda \bar{x}^\kappa \mapsto 1 \mid \bar{x} \in \mathcal{V}_\xi \uplus \{_\}, \mathtt{ord}(\kappa) \le k, \mathtt{iar}(\kappa) \le \iota\} \\
&\uplus \{@ \mapsto 2\} \uplus \{\mathbf{Y}^\kappa \mapsto 1, \perp^\kappa \mapsto 0 \mid \mathtt{ord}(\kappa) \le k, \mathtt{iar}(\kappa) \le \iota\}
\end{aligned}$$

where $\mathcal{V}_\xi = \{x_1, \cdots, x_\xi\}$ is a finite subset of $\mathcal{V}$ and the symbol @ represents the application operation. One can easily observe that $\Sigma_{\Lambda(k,\iota,\xi)}$ is finite. Since λY-terms are $\Sigma_{\Lambda(k,\iota,\xi)}$-trees, we can apply the decomposition method for trees to our λY-terms.

The decomposition function $\Phi_m(\cdot)$ (where m is a parameter) that decomposes a λY-term t into (i) a (sufficiently long) sequence $\overrightarrow{C} = C_1 \cdots C_k$ consisting of "affine" subcontexts of size no less than m, and (ii) a "second-order" context E (defined later), which is the remainder of extracting $\overrightarrow{C}$ from T. For example, the term on the left hand side of Figure 1 can be decomposed to the second-order context and affine contexts shown on the right hand side. Here, the symbol $[\![\,]\!]$ in the second-order context on the right-hand side represents the original position of each subcontext. By filling the i-th occurrence (counted in the depth-first, left-to-right pre-order) of $[\![\,]\!]$ with the i-th affine context, we can recover the original tree on the left hand side. Before introducing the decomposition function $\Phi_m(\cdot)$, we give formal definitions of contexts and second-order contexts.

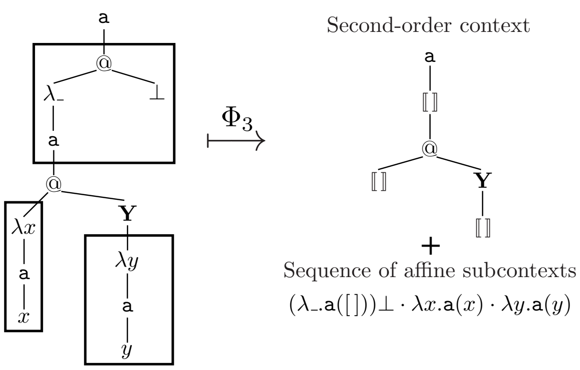

Figure 1 An example of term decomposition. The parts surrounded by rectangles on the left hand side show the extracted affine subcontexts, and the remaining part of the tree is the second-order tree context.

The set of *contexts* over a ranked alphabet Σ, also called Σ-contexts and ranged over C, is a set of $\Sigma \uplus \{[\,] \mapsto 0\}$-trees where $[\,]$ is a special nullary symbol called *hole*:

$$C ::= [\,] \mid a(C_1, \ldots, C_{\Sigma(a)}).$$

The *size* of a context C, denoted by $|C|$, is inductively defined as follows: $|[\,]| \triangleq 0$ and $|a(C_1, \ldots, C_{\Sigma(a)})| \triangleq 1 + |C_1| + \cdots + |C_{\Sigma(a)}|$. Note that $[\,]$ and each rank-0 constructor $a \in \mathrm{dom}(\Sigma)$ have different sizes: $|[\,]| = 0$ but $|a| = 1$. For a context C, we denote the number of occurrences of $[\,]$ in C by $\mathtt{hn}\,(C)$: $\mathtt{hn}\,([\,]) \triangleq 1$ and $\mathtt{hn}\,\left(a(C_1, \ldots, C_{\Sigma(a)})\right) \triangleq \mathtt{hn}\,(C_1) + \cdots + \mathtt{hn}\,\left(C_{\Sigma(a)}\right)$. $\mathtt{hn}\,(C) = 0$ means that C is a tree. We call C *linear* if $\mathtt{hn}\,(C) = 1$, and call *affine* if it is either a tree or linear. In general, we call C a *k-context* if $\mathtt{hn}\,(C) = k$. For contexts $C, \overrightarrow{C} = C_1 \cdots C_{\mathtt{hn}(C)}$, we write $C[\overrightarrow{C}]$ or $C[C_1, \ldots, C_{\mathtt{hn}(C)}]$ for the context which can be obtained by replacing each occurrence of $[\,]$ in C with C_i in the left-to-right non-capture-avoiding manner: $[\,][C] \triangleq C$ and $(a(C_1, \ldots, C_{\Sigma(a)}))[\overrightarrow{C}_1 \cdots \overrightarrow{C}_{\Sigma(a)}] \triangleq a(C_1[\overrightarrow{C_1}], \ldots, C_{\Sigma(a)}[\overrightarrow{C}_{\Sigma(a)}]])$, where $\#\left(\overrightarrow{C_i}\right) = \mathtt{hn}\,(C_i)$ for each $i \in [\Sigma(a)]$. For a 0-context C, $|C|$ coincides with the size of C as a Σ-tree. For contexts C, C', we call C' a *subcontext* of C, written $C' \preceq C$, if there exists contexts $C_0, C_1, \ldots, C_{\mathtt{hn}(C)}$ such that $C' = C_0[C[C_1, \ldots, C_{\mathtt{hn}(C)}]]$. In particular, if C, C' are trees then we say that C' is a *subtree* of C.

▶ **Definition 15** (second-order contexts). *The set of* second-order contexts *over Σ, ranged over by E, is defined by:*

$$E ::= [\![\,]\!]^n_k[E_1, \ldots, E_k] \;\mid\; a(E_1, \ldots, E_{\Sigma(a)}) \quad (a \in \mathrm{dom}(\Sigma)).$$

Intuitively, the second-order context is an expression having holes of the form $[\![\,]\!]^n_k$ (called *second-order holes*), which should be filled with a k-context of size n. By filling all the second-order holes, we obtain a Σ-tree. Note that k may be 0. In the technical development below, we only consider second-order holes $[\![\,]\!]^n_k$ such that k is 0 or 1. We write $\mathtt{shn}\,(E)$ for the number of the second-order holes in E. Note that Σ-trees can be regarded as second-order contexts E such that $\mathtt{shn}\,(E) = 0$, and vice versa. For $i \le \mathtt{shn}\,(E)$, we write $E.i$ for the i-th second-order hole (counted in the depth-first, left-to-right pre-order). We define the *size* $|E|$ by: $|[\![\,]\!]^n_k[E_1, \ldots, E_k]| \triangleq n + |E_1| + \cdots + |E_k|$ and $|a(E_1, \ldots, E_{\Sigma(a)})| \triangleq |E_1| + \cdots + |E_{\Sigma(a)}| + 1$. Note that $|E|$ includes the size of contexts to fill the second-order holes in E.

▶ **Definition 16** (substitution for second-order contexts). *For a context C and a second-order hole $[\![\,]\!]^n_k$, we write $C : [\![\,]\!]^n_k$ if C is a k-context of size n. For a second-order context E and a sequence of contexts $\overrightarrow{C} = C_1 \cdots C_{\mathtt{shn}(E)}$ such that $C_i : E.i$ for each $i \in [\mathtt{shn}\,(E)]$, we write $E[\overrightarrow{C}]$ or $E[C_1, \ldots, C_{\mathtt{shn}(E)}]$ for the tree which can be obtained by replacing each occurrence of $[\![\,]\!]$ in E with C_i in the left-to-right manner (and by interpreting the syntactical bracket $[-]$ as the substitution operation for usual contexts), where $\#\left(\overrightarrow{C_i}\right) = \mathtt{shn}\,(E_i)$ for each i:*

$$\left([\![\,]\!]^n_k[E_1, \ldots, E_k]\right)[C \cdot \overrightarrow{C_1} \cdots \overrightarrow{C_k}] \triangleq C[E_1[\overrightarrow{C_1}], \ldots, E_k[\overrightarrow{C_k}]]$$
$$(a(E_1, \ldots, E_{\Sigma(a)}))[\overrightarrow{C_1} \cdots \overrightarrow{C}_{\Sigma(a)}] \triangleq a(E_1[\overrightarrow{C_1}], \ldots, E_{\Sigma(a)}[\overrightarrow{C}_{\Sigma(a)}]).$$

We say that an affine context C is *good for m* (or *m-good*) if $|C| \ge m$ and C is of the form $a(C_1, \ldots, C_{\Sigma(a)})$ where $|C_i| < m$ for each $i \in [\Sigma(a)]$. In other words, C is good if C is of an appropriate size: it is large enough (i.e. $|C| \ge m$) but not too large (i.e. the size of any proper subcontext is less than m). For example, $\mathtt{a}(\mathtt{b}([\,]), \mathtt{b}(\mathtt{c}))$ is good for 3, but neither $C_1 = \mathtt{b}(\mathtt{b}([\,]))$ (since $|C_1| < 3$) nor $C_2 = \mathtt{a}(\mathtt{c}, \mathtt{b}(\mathtt{b}(\mathtt{b}([\,]))))$ (since $C_2' = \mathtt{b}(\mathtt{b}(\mathtt{b}([\,]))) \prec C_2$ has size 3) is.

▶ **Theorem 17** (decomposition function [1]). *For any $m \geq 2$, there exists a function $\Phi_m(\cdot)$ which takes a Σ-tree T and returns a pair $(E, \overrightarrow{C})$ of a second-order context and a sequence of good affine contexts such that:*

(1) $E[\overrightarrow{C}] = T$;

(2) $\mathtt{shn}(E) = \#\left(\overrightarrow{C}\right) \geq \frac{|T|}{2rm}$ *if* $m \leq |T|$ *where* $r = \max \mathrm{rng}(\Sigma)$; *and*

(3) *for any* $i \in [\mathtt{shn}(E)]$ *and any m-good affine context* $C : E.i$, $\Phi_m\left(E[\overrightarrow{C'}]\right) = (E, \overrightarrow{C'})$ *holds where* $\overrightarrow{C'}$ *is a sequence of contexts obtained by replacing the i-th component* $\overrightarrow{C}(i)$ *of* $\overrightarrow{C}$ *with* C.

4.2 Proof of Lemma 12

We are now ready to prove Lemma 12, under the assumption that Lemma 13 is correct (the proof of Lemma 13 is given in Section 6). For readability, in this subsection we fix the parameters k, ι, ξ and write $\hat{\Lambda}$ and $\hat{\Lambda}_n$ for $\hat{\Lambda}(k, \iota, \xi)$ and $\hat{\Lambda}_n(k, \iota, \xi)$, respectively. Let $r = \max \mathrm{rng}\big(\Sigma_{\Lambda(k,\iota,\xi)}\big)$.

We firstly introduce some auxiliary notation. For a term $t \in \hat{\Lambda}$ and $m \in \mathbb{N}$, we simply write E^t_m and $\overrightarrow{C}^t_m$ for the second-order context and sequence of contexts obtained by $\Phi_m(t)$, *i.e.*, $\Phi_m(t) = (E^t_m, \overrightarrow{C}^t_m)$. For $n, m \geq 2$ and a term $t \in \hat{\Lambda}$, we define

$$\mathcal{E}^n_m \triangleq \left\{E^t_m \mid t \in \hat{\Lambda}_n\right\} \qquad \Phi^{-1}_m(E) \triangleq \left\{t \in \hat{\Lambda}_{|E|} \mid E^t_m = E\right\}$$

$$\mathcal{C}_m(t,i) \triangleq \overrightarrow{C}^t_m(i) \qquad \mathcal{C}_m(E,i) \triangleq \left\{\mathcal{C}_m(t,i) \mid t \in \hat{\Lambda}_{|E|} \text{ and } E^t_m = E\right\}$$

For a second-order context E, we define a family of sets $S^E_0 \supseteq S^E_1 \supseteq \cdots \supseteq S^E_{\mathtt{shn}(E)}$ of minimal terms of size n as follows:

$$S^E_i \triangleq \left\{t \in \Phi^{-1}_m(E) \mid t_{\text{HARD},k} \not\preceq \mathcal{C}_m(t,j) \text{ for each } j \in [i]\right\}.$$

Note that $S^E_0 = \Phi^{-1}_m(E)$ and thus the fraction $\frac{\#\left(S^E_{\mathtt{shn}(E)}\right)}{\#\left(S^E_0\right)}$ means the probability that a randomly chosen term t from $\Phi^{-1}_m(E)$ does not contain $t_{\text{HARD},k}$ in any its decomposed subcontexts.

By using Lemma 13, we can easily prove that, for any term $t \in \hat{\Lambda}$ and $i \in [\mathtt{shn}(E^t_m)]$, there exists a good affine context C such that $t_{\text{HARD},k} \preceq C$, $C : E^t_m.i$ and $E^t_m[\overrightarrow{C}] \in \hat{\Lambda}$. This means that a term t can contain $t_{\text{HARD},k}$ as a subterm in arbitrary decomposed part *independently with other decomposed parts*. Hence, if S^E_{i-1} is non-empty, $S^E_{i-1} \setminus S^E_i$ is also non-empty (*i.e.*, $S^E_{i-1} \supsetneq S^E_i$) for each $i \in [2, \mathtt{shn}(E)]$. Moreover, since we can bound the number of possible decomposed contexts as $\#(\mathcal{C}_m(E,i)) \leq \gamma^{rm}$ for some constant γ (intuitively, γ is an upper-bound of the *growth rate* of the number of contexts of size at most rm), the fraction $\#\left(S^E_i\right)/\#\left(S^E_{i-1}\right)$ is bounded above by $(\gamma^{rm} - 1)/\gamma^{rm} = 1 - \gamma^{-rm}$. Summing up above discussion, by using Lemma 13 and some analysis, we can bound the probability that no decomposed part contains $t_{\text{HARD},k}$ as follows (see Appendix B for details).

▶ **Lemma 18.** *For some real number $\gamma > 0$,* $\dfrac{\sum_{E \in \mathcal{E}^n_m} \#\left(S^E_{\mathtt{shn}(E)}\right)}{\sum_{E \in \mathcal{E}^n_m} \#\left(S^E_0\right)} \leq (1 - \gamma^{-rm})^{\mathtt{shn}(E)}$.

Thus we have our Lemma 12 as:

$$\frac{\#\Big(\{t \in \hat{\Lambda}_n(k,\iota,\xi) \mid t_{\textsc{Hard},k} \not\preceq t\}\Big)}{\#\Big(\hat{\Lambda}_n(k,\iota,\xi)\Big)} \leq \frac{\sum_{E \in \mathcal{E}_m^n} \#\Big(S^E_{\mathtt{shn}(E)}\Big)}{\sum_{E \in \mathcal{E}_m^n} \#\big(S^E_0\big)}$$

$$(\because \text{Lemma 18}) \qquad \leq (1-\gamma^{-rm})^{\mathtt{shn}(E)}$$

$$(\because \text{Item 2 of Theorem 17}) \qquad \leq (1-\gamma^{-rm})^{\frac{n}{2rm}} \longrightarrow 0 \quad (\text{as } n \longrightarrow \infty).$$

5 Intersection Types for Minimal Terms

In this section, we introduce an intersection type system for characterizing minimal terms. This type system will be a key tool to show Lemma 13. We define the sets of *prime intersection types* and *intersection types* as follows, where $n \geq 0$:

$$\tau, \sigma ::= \mathsf{o} \mid \theta \to \tau \qquad \theta, \delta ::= \bigwedge^{\kappa}\{\tau_1, \dots, \tau_n\}.$$

We often abbreviate $\bigwedge^{\kappa}\{\tau_1, \dots, \tau_n\}$ by $\bigwedge\{\tau_1, \dots, \tau_n\}$. We also often write $\bigwedge^{\kappa}_{i\in[n]} \tau_i$ (or $\tau_1 \wedge \cdots \wedge \tau_n$) for $\bigwedge^{\kappa}\{\tau_1, \dots, \tau_n\}$, and $\top^{\kappa}$ (or $\top$) for $\bigwedge^{\kappa}\emptyset$. For each intersection types $\theta = \bigwedge^{\kappa} S$ and $\delta = \bigwedge^{\kappa} T$, We denote by $\theta \wedge \delta$ the intersection type $\bigwedge^{\kappa}(S \cup T)$. We use $\bar{\theta}$, $\bar{\delta}$ to denote a prime intersection type or an intersection type. An *intersection type environment*, written as Θ or Δ, is a finite partial mapping from $\mathcal{V}$ to the set of intersection types. For each Θ, $x \in \mathcal{V} \setminus \mathrm{dom}(\Theta)$, and θ, we write $(\Theta, x : \theta)$ for $\Theta \cup \{x \mapsto \theta\}$. The *refinement relation* $\bar{\theta} :: \kappa$ (resp. $\Theta :: \Gamma$) is the least relation closed under the following rules, where $n \geq 0$:

$$\frac{}{\mathsf{o} :: \mathsf{o}} \qquad \frac{\tau_1 :: \kappa \quad \dots \quad \tau_n :: \kappa}{\bigwedge^{\kappa}_{i\in[n]} \tau_i :: \kappa} \qquad \frac{\theta :: \kappa \quad \tau :: \kappa'}{\theta \to \tau :: \kappa \to \kappa'} \qquad \frac{}{\emptyset :: \emptyset} \qquad \frac{\Theta :: \Gamma \quad \theta :: \kappa}{(\Theta, x : \theta) :: (\Gamma, x : \kappa)}.$$

Henceforth we only consider intersection types occurring in this refinement relation (so, we always make the assumption that for each $\bar{\theta}$, $\bar{\theta} :: \kappa$ holds for some κ). Thanks to the κ in $\bigwedge^{\kappa}$, for each $\bar{\theta}$ (and similarly for Θ), the type κ such that $\bar{\theta} :: \kappa$ is unique.

We write $\Theta \wedge \Delta$ for the intersection type environment $\{x \mapsto \Theta(x) \wedge \Delta(x) \mid x \in \mathrm{dom}(\Theta) \cup \mathrm{dom}(\Delta)\}$, where $\Theta(x) = \top^{\kappa}$ (similarly for $\Delta(x)$) if $x \notin \mathrm{dom}(\Theta)$ (where κ is determined by $\Delta(x)$). The *intersection type judgement relation* $\Theta \vdash t : \bar{\theta}$ is inductively defined by the rules in Figure 2, where we force that $\Theta \vdash t : \bar{\theta}$ holds only when $\Gamma \vdash_{\mathrm{ST}} t : \kappa$, $\Theta :: \Gamma$, and $\bar{\theta} :: \kappa$ hold.

$$\frac{}{x : \wedge\{\tau\} \vdash x^{\kappa} : \tau}(\mathrm{Var}) \qquad \frac{\Theta, x : \theta \vdash t : \tau}{\Theta \vdash \lambda x.t : \theta \to \tau}(\mathrm{Abs1}) \qquad \frac{\Theta \vdash t : \tau}{\Theta \vdash \lambda \bar{x}.t : \top \to \tau}(\mathrm{Abs2})$$

$$\frac{\Theta \vdash t : \theta \to \tau \quad \Delta \vdash s : \theta}{\Theta \wedge \Delta \vdash t\, s : \tau}(\mathrm{App}) \qquad \frac{\Theta \vdash t_1\,(\mathbf{Y} t_2) : \tau}{\Theta \vdash \mathbf{Y}(t_1 \sqcup t_2) : \tau}(\mathbf{Y}1) \qquad \frac{\Theta \vdash t \perp : \tau}{\Theta \vdash \mathbf{Y} t : \tau}(\mathbf{Y}2)$$

$$\frac{\Theta_1 \vdash t_1 : \theta_1 \quad \dots \quad \Theta_n \vdash t_n : \theta_n}{\bigwedge_{i\in[n]} \Theta_i \vdash a(t_1, \dots, t_n) : \mathsf{o}}(a) \qquad \frac{\Theta_1 \vdash t_1 : \tau_1 \quad \dots \quad \Theta_n \vdash t_n : \tau_n}{\bigwedge_{i\in[n]} \Theta_i \vdash \bigsqcup_{i\in[n]} t_i : \bigwedge_{i\in[n]} \tau_i}(\wedge) \qquad \frac{\Theta \vdash t : \bar{\theta}}{\Theta, x : \top \vdash t : \bar{\theta}}(\top)$$

Figure 2 The intersection type system for the minimality.

Intuitively, we write $\emptyset \vdash t : \bigwedge\{\tau_1, \dots, \tau_n\}$ if t is typed by each of $\tau_1, \dots, \tau_n$, in a standard (idempotent) intersection type system, but in this intersection type system, we write the one if there is a partition $\{t_i\}_{i\in[n]}$ of t (i.e., $t = \bigsqcup_{i\in[n]} t_i$) such that each t_i is typed by τ_i. This difference is useful for characterizing the minimality introduced in Section 3 in cases of that terms are "used" in multiple ways; see Example 21. The following theorem states that the minimality can be characterized by this intersection type system.

▶ **Theorem 19** (soundness and completeness). *For every closed and ground-typed term t, t is minimal if and only if $\emptyset \vdash t : \bar{\theta}$ for some $\bar{\theta}$.*

Proof Sketch. Both the soundness and the completeness can be proved by showing a subject-reduction lemma and a subject-expansion lemma for this intersection type system, respectively. The proof is proceeded in a standard way (using an alternative definition of the minimality), but not so concise. For the details of the proof, see Appendix H in the full version [20]. ◀

The following are examples of proving the minimality by using the intersection type system.

▶ **Example 20.** Let $t = (\lambda x^{\mathsf{o}}.\lambda y^{\mathsf{o}}.x^{\mathsf{o}})\ \mathsf{a}\ \bot^{\mathsf{o}}$ be the term appeared in Section 3. Then we can show that t is minimal by giving the derivation tree of $\emptyset \vdash t : \mathsf{o}$ as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : \wedge\{\mathsf{o}\} \vdash x^{\mathsf{o}} : \mathsf{o}}\,(\text{Var})}{x : \wedge\{\mathsf{o}\} \vdash \lambda y^{\mathsf{o}}.x^{\mathsf{o}} : \top \to \mathsf{o}}\,(\text{Abs2})}{\emptyset \vdash \lambda x^{\mathsf{o}}.\lambda y^{\mathsf{o}}.x^{\mathsf{o}} : \wedge\{\mathsf{o}\} \to \top \to \mathsf{o}}\,(\text{Abs1}) \quad \dfrac{\dfrac{}{\emptyset \vdash \mathsf{a} : \mathsf{o}}\,(\mathsf{a})}{\emptyset \vdash \mathsf{a} : \wedge\{\mathsf{o}\}}\,(\wedge)}{\emptyset \vdash (\lambda x^{\mathsf{o}}.\lambda y^{\mathsf{o}}.x^{\mathsf{o}})\ \mathsf{a} : \top \to \mathsf{o}}\,(\text{App}) \quad \dfrac{}{\emptyset \vdash \bot^{\mathsf{o}} : \top}\,(\wedge)}{\emptyset \vdash (\lambda x^{\mathsf{o}}.\lambda y^{\mathsf{o}}.x^{\mathsf{o}})\ \mathsf{a}\ \bot^{\mathsf{o}} : \mathsf{o}}\,(\text{App})$$

Note that in contrast, $\emptyset \nvdash (\lambda x^{\mathsf{o}}.\lambda y^{\mathsf{o}}.x^{\mathsf{o}})\ \mathsf{a}\ \mathsf{a} : \mathsf{o}$ by $x : \wedge\{\mathsf{o}\}, y : \wedge\{\mathsf{o}\} \nvdash x^{\mathsf{o}} : \mathsf{o}$; see (Var).

The following case is a bit more complicated, but the *intersection types* are essentially used.

▶ **Example 21.** Let $s = (\lambda f^{(\mathsf{o}\to\mathsf{o}\to\mathsf{o})\to\mathsf{o}}.\mathsf{a}(f\,\mathsf{fst}, f\,\mathsf{snd}))$, $u = (\lambda g^{\mathsf{o}\to\mathsf{o}\to\mathsf{o}}.g\,\mathsf{b}\,\mathsf{c})$, and $t = s\,u$, where $\mathsf{fst} = \lambda x^{\mathsf{o}}.\lambda y^{\mathsf{o}}.x^{\mathsf{o}}$ and $\mathsf{snd} = \lambda x^{\mathsf{o}}.\lambda y^{\mathsf{o}}.y^{\mathsf{o}}$. Then $\emptyset \vdash t : \mathsf{o}$ is derived from the following two by applying (App), where $\tau_1 = \wedge\{\mathsf{o}\} \to \top \to \mathsf{o}$ and $\tau_2 = \top \to \wedge\{\mathsf{o}\} \to \mathsf{o}$. Hence this t is minimal. Note that the term u is "used" in two ways when it is applied to the term s (the $f\,\mathsf{fst}$ uses the b and the $f\,\mathsf{snd}$ uses the c, respectively).

$$\dfrac{\dfrac{\dfrac{\dfrac{}{f : \wedge\{\wedge\{\tau_1\} \to \mathsf{o}\} \vdash f : \wedge\{\tau_1\} \to \mathsf{o}}\,(\text{Var}) \quad \dfrac{\dfrac{}{\emptyset \vdash \mathsf{fst} : \tau_1}}{\emptyset \vdash \mathsf{fst} : \wedge\{\tau_1\}}\,(\wedge)}{f : \wedge\{\wedge\{\tau_1\} \to \mathsf{o}\} \vdash f\,\mathsf{fst} : \mathsf{o}}\,(\text{App})}{f : \wedge\{\wedge\{\tau_1\} \to \mathsf{o}\} \vdash f\,\mathsf{fst} : \{\mathsf{o}\}}\,(\wedge) \quad \dfrac{\text{(similarly to the left)}}{f : \wedge\{\wedge\{\tau_2\} \to \mathsf{o}\} \vdash f\,\mathsf{snd} : \{\mathsf{o}\}}\,(\wedge)}{\dfrac{f : \wedge\{\wedge\{\tau_1\} \to \mathsf{o}, \wedge\{\tau_2\} \to \mathsf{o}\} \vdash \mathsf{a}(f\,\mathsf{fst}, f\,\mathsf{snd}) : \mathsf{o}}{\emptyset \vdash \lambda f.\mathsf{a}(f\,\mathsf{fst}, f\,\mathsf{snd}) : \bigwedge_{l\in[2]}\{\wedge\{\tau_l\} \to \mathsf{o}\} \to \mathsf{o}}\,(\text{Abs1})}\,(\mathsf{a})$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{g : \wedge\{\tau_1\} \vdash g : \wedge\{\mathsf{o}\} \to \top \to \mathsf{o}}\,(\text{Var}) \quad \dfrac{\dfrac{}{\emptyset \vdash \mathsf{b} : \mathsf{o}}\,(\mathsf{b})}{\emptyset \vdash \mathsf{b} : \wedge\{\mathsf{o}\}}\,(\wedge) \quad \dfrac{}{\emptyset \vdash \bot : \top}\,(\wedge)}{g : \wedge\{\tau_1\} \vdash g\,\mathsf{b}\,\bot : \mathsf{o}}\,(\text{App})}{\emptyset \vdash \lambda g.g\,\mathsf{b}\,\bot : \wedge\{\tau_1\} \to \mathsf{o}}\,(\text{Abs1}) \quad \dfrac{\dfrac{\text{(similarly to the left)}}{g : \wedge\{\tau_2\} \vdash g\,\bot\,\mathsf{c} : \mathsf{o}}\,(\text{App})}{\emptyset \vdash \lambda g.g\,\bot\,\mathsf{c} : \wedge\{\tau_2\} \to \mathsf{o}}\,(\text{Abs1})}{\emptyset \vdash \lambda g.g\,\mathsf{b}\,\mathsf{c} : \bigwedge_{l\in[2]}\{\wedge\{\tau_l\} \to \mathsf{o}\}}\,(\wedge)$$

6 Proof of the Main Lemma (Lemma 13)

In this section, we prove Lemma 13 by using the intersection type system in the previous section. Recall that we need to prove that if $E[C] \in \hat{\Lambda}_n(k, \iota, \xi)$, then there is a context

$D \succeq t_{\text{HARD},k}$ such that $E[D] \in \hat{\Lambda}_n(k,\iota,\xi)$. Thanks to the result of the previous section, $E[C] \in \hat{\Lambda}_n(k,\iota,\xi)$ implies that $E[C]$ is typable in the intersection type system. Thus, it suffices to construct D of the same size such that (i) C has "the same typing properties" as D, and (ii) D contains $t_{\text{HARD},k}$. To this end, we first extend the notion of types to those of contexts (called *context-types*) in Section 6.1. We then show in Section 6.2 that we can indeed construct a context D that has the same context types as C, and prove Lemma 13.

6.1 Context-Types

For each affine-context C, we write $C \lhd_{\text{ST}} \{\langle\Gamma'_1,\kappa'_1\rangle,\dots,\langle\Gamma'_n,\kappa'_n\rangle\} \Rrightarrow \langle\Gamma,\kappa\rangle$ if there is a derivation tree of $\Gamma \vdash_{\text{ST}} C[\mathtt{x}] : \kappa$ with the assumptions $\{\Gamma'_1 \vdash_{\text{ST}} \mathtt{x} : \kappa'_1,\dots,\Gamma'_n \vdash_{\text{ST}} \mathtt{x} : \kappa'_n\}$, where $\mathtt{x}$ is a variable not occurring in C (informally speaking, it means that there is a derivation tree of $\Gamma' \vdash_{\text{ST}} C : \kappa'$ with the assumptions $\{\Gamma'_1 \vdash_{\text{ST}} [\,] : \kappa'_1,\dots,\Gamma'_n \vdash_{\text{ST}} [\,] : \kappa'_n\}$). For example, let $t = (\lambda x^{\mathtt{o}}.[\,]\,x)\,\mathtt{a}$; then $t \lhd_{\text{ST}} \{\langle(\Gamma,x:\mathtt{o}),\mathtt{o}\to\kappa\rangle\} \Rrightarrow \langle\Gamma,\kappa\rangle$, where Γ is any environment and κ is any simple-type. We often write $t \lhd_{\text{ST}} \tilde{\theta}$ for $t \lhd_{\text{ST}} \emptyset \Rrightarrow \tilde{\theta}$. We use $\tilde{\kappa}$ to denote a pair $\langle\Gamma,\kappa\rangle$ and use $\tilde{\nu}$ to denote a $\{\langle\Gamma'_1,\kappa'_1\rangle,\dots,\langle\Gamma'_n,\kappa'_n\rangle\} \Rrightarrow \langle\Gamma,\kappa\rangle$. Note that C is a term (resp. a linear-context) if $C \lhd_{\text{ST}} \{\langle\Gamma'_1,\kappa'_1\rangle,\dots,\langle\Gamma'_n,\kappa'_n\rangle\} \Rrightarrow \langle\Gamma,\kappa\rangle$ holds for $n=0$ (resp. $n=1$). In the following, we extend the notion of $\lhd_{\text{ST}}$ to the intersection type system. The set of *(affine-)context-types*, ranged over by $\tilde{\mu}$, is defined as follows, where $n \geq 0$ and we may write $\tilde{\theta}^+$ for $\tilde{\theta}$ if $\tilde{\theta} \neq \emptyset$:

$$\tilde{\tau} ::= \langle\Theta,\tau\rangle \qquad \tilde{\theta} ::= \{\tilde{\tau}_1,\dots,\tilde{\tau}_n\} \qquad \tilde{\pi} ::= \tilde{\tau} \mid \tilde{\theta}^+ \qquad \tilde{\mu} ::= \tilde{\theta} \Rrightarrow \tilde{\pi}.$$

The *refinement relation* is the least relation closed under the following rules, where $n \geq 0$:

$$\frac{\Theta :: \Gamma \qquad \tau :: \kappa}{\langle\Theta,\tau\rangle :: \langle\Gamma,\kappa\rangle} \qquad \frac{\tilde{\tau}_1 :: \langle\Gamma,\kappa\rangle \quad \dots \quad \tilde{\tau}_n :: \langle\Gamma,\kappa\rangle}{\{\tilde{\tau}_1,\dots,\tilde{\tau}_n\} :: \langle\Gamma,\kappa\rangle} \qquad \frac{\tilde{\theta}' :: \langle\Gamma',\kappa'\rangle \qquad \tilde{\pi} :: \langle\Gamma,\kappa\rangle}{\tilde{\theta}' \Rrightarrow \tilde{\pi} :: \langle\Gamma',\kappa'\rangle \Rrightarrow \langle\Gamma,\kappa\rangle}.$$

Henceforth we only consider context-types occurring in this refinement relation (so, we always make the assumptions that for each $\tilde{\theta}' \Rrightarrow \tilde{\theta}$, for some $\langle\Gamma,\kappa\rangle$ and $\langle\Gamma',\kappa'\rangle$, $\tilde{\theta} :: \langle\Gamma,\kappa\rangle$ and $\tilde{\theta}' :: \langle\Gamma',\kappa'\rangle$). For each affine-context C, we write $C \lhd \{\langle\Theta'_1,\tau'_1\rangle,\dots,\langle\Theta'_n,\tau'_n\rangle\} \Rrightarrow \langle\Theta,\tau\rangle$ if there is a derivation tree of $\Theta \vdash C[\mathtt{x}] : \tau$ with the assumptions $\{\Theta'_1 \vdash \mathtt{x} : \tau'_1,\dots,\Theta'_n \vdash \mathtt{x} : \tau'_n\}$. For $n \geq 1$, we write $(\bigsqcup_{i\in[n]} C_i) \lhd (\bigcup_{i\in[n]} \tilde{\theta}'_i) \Rrightarrow \{\tilde{\tau}_1,\dots,\tilde{\tau}_n\}$ if $C_i \lhd \tilde{\theta}'_i \Rrightarrow \tilde{\tau}_i$ for each $i \in [n]$. We often write $t \lhd \tilde{\theta}$ for $t \lhd \emptyset \Rrightarrow \tilde{\theta}$. We list a few properties (see Appendix D for the proofs).

▶ **Proposition 22** (substitution). *Suppose that C is a linear-context. If $C \lhd \tilde{\theta}' \Rrightarrow \tilde{\theta}$ and $C' \lhd \tilde{\theta}'' \Rrightarrow \tilde{\theta}'$, then $C[C'] \lhd \tilde{\theta}'' \Rrightarrow \tilde{\theta}$.*

▶ **Proposition 23** (inverse substitution). *Suppose that C is a linear-context. If $C[C'] \lhd \tilde{\theta}'' \Rrightarrow \tilde{\theta}$, then $C \lhd \tilde{\theta}' \Rrightarrow \tilde{\theta}$ and $C' \lhd \tilde{\theta}'' \Rrightarrow \tilde{\theta}'$ for some $\tilde{\theta}'$.*

These properties enable us to replace contexts preserving the minimality. For example, given $\emptyset \vdash C[D[t]] : \mathtt{o}$ (i.e., $C[D[t]]$ is minimal); then by Proposition 23, $C \lhd \tilde{\theta} \Rrightarrow \{\langle\emptyset,\mathtt{o}\rangle\}$, $D \lhd \tilde{\theta}' \Rrightarrow \tilde{\theta}$, and $t \lhd \tilde{\theta}'$ for some $\tilde{\theta}$ and $\tilde{\theta}'$; then by Proposition 22, $C[D'[t]] \lhd \{\langle\emptyset,\mathtt{o}\rangle\}$ (hence, $C[D'[t]]$ is also minimal) for each linear context $D' \lhd \tilde{\theta}' \Rrightarrow \tilde{\theta}$.

6.2 Proof of Lemma 13

Here, we fix parameters k, ι, and ξ. W.l.o.g., in the following, we only consider terms, contexts, and environments having only variables in a fixed set $\mathcal{V}_\xi \triangleq \{z_1,\dots,z_\xi\}$ (of size ξ). We say that $\langle\Gamma,\kappa\rangle$ is *($\langle k,\iota,\xi\rangle$-)bounded* if $\max\{\mathtt{ord}\,(\kappa') \mid \kappa' \in \{\kappa\} \cup \mathrm{rng}(\Gamma)\} \leq k$ and

$\max\{\mathtt{iar}(\kappa') \mid \kappa' \in \{\kappa\} \cup \mathrm{rng}(\Gamma)\} \le \iota$; and that $\langle \Gamma', \kappa' \rangle \Rrightarrow \langle \Gamma, \kappa \rangle$ is *bounded* if both $\langle \Gamma', \kappa' \rangle$ and $\langle \Gamma, \kappa \rangle$ are; and that a context-type $\tilde{\mu}$ is *bounded* if the $\tilde{\nu}$ such that $\tilde{\mu} :: \tilde{\nu}$ is. We also say that t is *bounded* if $\mathtt{ord}(t) \le k$ and $\mathtt{iar}(t) \le \iota$; and that a linear-context C is *bounded* if $C[\bot]$ is. Also, we use $\mathtt{a}$ (resp. $\mathtt{b}$, $\mathtt{c}$) to denote a tree constructor of arity 0 (resp. 2, 1).

The following technical lemma allows conversion between a ground-typed term and a term of a required typing property: see Appendix C for a proof.

▶ **Lemma 24.** (1) *Suppose that* $\tilde{\theta}^+ :: \langle \Gamma, \kappa \rangle$ *is bounded. If* $\#(\mathrm{dom}(\Gamma)) < \xi$ *or* $\mathtt{ar}(\kappa) < \iota$, *then* $C_{\tilde{\theta}^+} \lhd \{\langle \emptyset, \mathtt{o} \rangle\} \Rrightarrow \tilde{\theta}^+$ *for some bounded linear-context* $C_{\tilde{\theta}^+}$.
(2) *Suppose that* $\tilde{\theta}$ *is bounded. Then,* $D_{\tilde{\theta}} \lhd \tilde{\theta} \Rrightarrow \{\langle \emptyset, \mathtt{o} \rangle\}$ *for a bounded affine-context* $D_{\tilde{\theta}}$.

The following is the key lemma, which shows that for any bounded context-type, one can construct a context D that has the context-type and contains the hard term $t_{\mathrm{HARD},k}$.

▶ **Lemma 25.** *Suppose that* $C \lhd \tilde{\theta}' \Rrightarrow \tilde{\theta}^+$ *for some bounded affine-context* C. *Then for some* m_0, *for every* $m \ge m_0$, *there is a bounded affine-context* D *of size* m *such that* $D \lhd \tilde{\theta}' \Rrightarrow \tilde{\theta}^+$ *and* $D \succeq t_{\mathrm{HARD},k}$.

Proof. Let $\langle \Gamma, \kappa \rangle$ be such that $\tilde{\theta}^+ \lhd \langle \Gamma, \kappa \rangle$. Note that $\tilde{\theta}'$ and $\tilde{\theta}^+$ are also bounded.

(a) $\#(\mathrm{dom}(\Gamma)) < \xi$ or $\mathtt{ar}(\kappa) < \iota$: For each $l \ge 0$, let D_l be as follows, where $\mathtt{c}^l(\mathtt{a})$ is the term $\mathtt{c}(\dots\mathtt{c}(\mathtt{a})\dots)$ that $\mathtt{c}$ occurs l times and $D_{\tilde{\theta}'}$ and $C_{\tilde{\theta}^+}$ are the ones in Lemma 24:

$$D_l \triangleq C_{\tilde{\theta}^+}[\mathtt{b}(t_{\mathrm{HARD},k}, \mathtt{b}(\mathtt{c}^l(\mathtt{a}), []))][D_{\tilde{\theta}'}].$$

Then $D_l \succeq t_{\mathrm{HARD},k}$ is obvious, and $D_l \lhd \tilde{\theta}' \Rrightarrow \tilde{\theta}^+$ by Proposition 22 (since $\mathtt{b}(t_{\mathrm{HARD},k}, \mathtt{b}(\mathtt{c}^l, [])) \lhd \{\langle \emptyset, \mathtt{o} \rangle\} \Rrightarrow \{\langle \emptyset, \mathtt{o} \rangle\}$). Therefore, the claim has been proved by using these $D_1, D_2, \cdots$.

(b) Otherwise: Then, $\Gamma \vdash_{\mathrm{ST}} C[\bot] : \kappa$, $C[\bot]$ is bounded, and $\#(\mathrm{dom}(\Gamma)) = \xi$ and $\mathtt{ar}(\kappa) = \iota$, so C should be of the form $\lambda\_.C_0$ (see Lemma 40 in Appendix C). By Proposition 23, $C_0 \lhd \tilde{\theta}' \Rrightarrow \tilde{\theta}_0$ and $\lambda\_.[] \lhd \tilde{\theta}_0 \Rrightarrow \tilde{\theta}$ for some $\tilde{\theta}_0$. Then $\mathtt{ar}(C_0) < \mathtt{ar}(C) \le \iota$ and $\tilde{\theta}_0 \ne \emptyset$ by $C_0 \ne \bot$ (since $\xi > 0$). Therefore by (a), for some m_0', there is $\{D_l'\}_{l \ge m_0'}$ such that $D_l' \lhd \tilde{\theta}' \Rrightarrow \tilde{\theta}_0$, $D_l' \succeq t_{\mathrm{HARD},k}$, and $|D_l'| = l$ for each $l \ge m_0'$. Let $D_l = \lambda\_.D_l'$. Then $D_l \succeq t_{\mathrm{HARD},k}$ is obvious, and $D_l \lhd \tilde{\theta}' \Rrightarrow \tilde{\theta}^+$ by Proposition 22. Therefore, the claim has been proved by using these $D_{m_0'}, D_{m_0'+1}, \cdots$. ◀

We are now ready to prove the main lemma.

Proof (of Lemma 13). Let $m \triangleq \max\{m_{\tilde{\theta}' \Rrightarrow \tilde{\theta}^+} \mid C \lhd \tilde{\theta}' \Rrightarrow \tilde{\theta}^+ \text{ for some bounded } C\}$, where each $m_{\tilde{\theta}' \Rrightarrow \tilde{\theta}^+}$ is the m_0 in Lemma 25. Indeed such m exists, since the number of bounded context-types is finite. Recall $E[C] \in \hat{\Lambda}_n(k, \iota, \xi)$. Let $\tilde{E}$ be an affine-context such that $E[C] = \tilde{E}[C[t]]$ for some t or $E[C] = \tilde{E}[C]$. For the sake of brevity, we only write the case of that $\tilde{E}$ is linear-context (i.e., $E[C] = \tilde{E}[C[t]]$). Since $\tilde{E}[C[t]]$ is minimal, $\emptyset \vdash \tilde{E}[C[t]] : \bar{\theta}$ for some $\bar{\theta} :: \mathtt{o}$ (Theorem 19). Then $\tilde{E}[C[t]] \lhd \emptyset \Rrightarrow \{\langle \emptyset, \mathtt{o} \rangle\}$ (by $\tilde{E}[C[t]] \ne \bot$). By Proposition 23, there are $\tilde{\theta}$ and $\tilde{\theta}'$ such that $\tilde{E} \lhd \tilde{\theta} \Rrightarrow \{\langle \emptyset, \mathtt{o} \rangle\}$, $C \lhd \tilde{\theta}' \Rrightarrow \tilde{\theta}$, and $t \lhd \emptyset \Rrightarrow \tilde{\theta}'$. By Lemma 25 (and $C \ne \bot$), there is a bounded linear-context $D \lhd \tilde{\theta}' \Rrightarrow \tilde{\theta}$ such that $D \succeq t_{\mathrm{HARD},k}$ and $|D| = |C|$. Therefore $\tilde{E}[D[t]] \lhd \emptyset \Rrightarrow \{\langle \emptyset, \mathtt{o} \rangle\}$ (hence, $\emptyset \vdash \tilde{E}[D[t]] : \wedge\{\mathtt{o}\}$) by Proposition 22, and thus $E[D]$ is minimal (Theorem 19). Hence, $E[D] \in \hat{\Lambda}_n(k, \iota, \xi)$. ◀

7 Related Work

Ong [21] proved the k-EXPTIME completeness of higher-order model checking. There have also been results on parameterized complexity [15, 18, 17] and the complexity of subclasses of the problem [17, 5]. To our knowledge, however, they are all about the worst-case complexity. Despite the extremely high worst-case complexity, practical model checkers have been developed that run quite fast for typical inputs [14, 4, 24, 29], which has led to the motivating question for our work: is higher-order model checking really hard in the average case?

Technically, closest to ours is the work of Asada et al. [27, 1] on a quantitative analysis of the length of β-reduction sequences of simply-typed λ-terms. In fact, our use of the tree-version of infinite monkey theorem (to show that almost every term contains a "hard" term), as well as the tree decomposition (Theorem 17) has been inspired by their work and other studies on quantitative analysis of the λ-calculus and combinatory logics [8, 2]. The main new difficulty was that, unlike in the case of the length of β-reduction sequences, even if t is a "hard" term to model-check, a term $C[t]$ that contains t as a subterm may not be hard to model-check, because t may not actually be used in $C[t]$ or may be irrelevant for the property to be checked. This has led us to restrict terms to "minimal ones" that do not contain unnecessary subterms. The restriction turned out to be natural also for our goal: we wish to model the *average* case that arises in the actual applications to program verification, and the restriction to minimal terms helps us exclude out unlikely inputs.

We have used an intersection type system to characterize minimal terms. Related type systems have been studied in the context of useless code elimination [6, 7, 13]. In particular, Daminani [7] also used an intersection type system. To our knowledge, however, previous studies do not provide a *complete* characterization of minimal terms (especially in the presence of recursion).

There has been much interest in the average-case complexity in the field of computational complexity: see [3] for a good survey. In their terminology, our ultimate goal is to answer whether $(\mathrm{HOMC}_k(\cdot,\cdot),\mathcal{U})$ belongs to $\mathrm{Avg}_\delta\mathrm{DTIME}(f(n))$ (the class of distributional problems that can be solved in time $f(n)$ for at least $(1-\delta(n))$-fraction of the inputs of size n),[3] where $\mathrm{HOMC}_k(\cdot,\cdot)$ is the higher-order model checking problem of order k, $\mathcal{U}$ is a uniform distribution on inputs of each size n, δ is a function that is asymptotically smaller than $\lambda n.1$, and $f(n)$ is a function asymptotically much smaller than $exp_k(cn)$ (a k-fold exponential function). The result obtained in the present paper (Theorem 6) is not yet of this form, and is rather a mixture of average-case and worst-case analysis, which may be of independent interest from the perspective of complexity theory.

8 Conclusion

We have studied a mixture of average-case and worst-case complexity of higher-order model checking, and shown that for almost every order-k λY-term t, the higher-order model checking problem specialized for t is k-EXPTIME hard with respect to the size of a tree automaton. To our knowledge, this is the first result on the average-case hardness of higher-order model checking. To obtain the result, we have given a complete type-based characterization of "minimal" terms that contain no useless subterms, which may be of independent interest. Pure average-case analysis of the hardness of higher-order model checking is left for future work.

[3] A similar notion has also been studied under the name "generic-case complexity" [11].

References

1 Kazuyuki Asada, Naoki Kobayashi, Ryoma Sin'ya, and Takeshi Tsukada. Almost Every Simply Typed Lambda-Term Has a Long Beta-Reduction Sequence. *Logical Methods in Computer Science*, Volume 15, Issue 1, February 2019. URL: `https://lmcs.episciences.org/5203`, `doi:10.23638/LMCS-15(1:16)2019`.

2 Maciej Bendkowski, Katarzyna Grygiel, and Marek Zaionc. On the likelihood of normalization in combinatory logic. *J. Log. Comput.*, 27(7):2251–2269, 2017. URL: `https://doi.org/10.1093/logcom/exx005`, `doi:10.1093/logcom/exx005`.

3 Andrej Bogdanov and Luca Trevisan. Average-case complexity. *CoRR*, abs/cs/0606037, 2006. URL: `http://arxiv.org/abs/cs/0606037`, `arXiv:cs/0606037`.

4 Christopher H. Broadbent and Naoki Kobayashi. Saturation-based model checking of higher-order recursion schemes. In *Proceedings of CSL 2013*, volume 23 of *LIPIcs*, pages 129–148, 2013.

5 Pierre Clairambault, Charles Grellois, and Andrzej S. Murawski. Linearity in higher-order recursion schemes. *PACMPL*, 2(POPL):39:1–39:29, 2018. URL: `https://doi.org/10.1145/3158127`, `doi:10.1145/3158127`.

6 Mario Coppo, Ferruccio Damiani, and Paola Giannini. Refinement types for program analysis. In *Proceedings of the Third International Symposium on Static Analysis*, SAS '96, pages 143–158, Berlin, Heidelberg, 1996. Springer-Verlag. URL: `http://dl.acm.org/citation.cfm?id=647165.717834`.

7 Ferruccio Damiani. A conjunctive type system for useless-code elimination. *Mathematical Structures in Computer Science*, 13(1):157–197, 2003. URL: `https://doi.org/10.1017/S0960129502003869`, `doi:10.1017/S0960129502003869`.

8 René David, Katarzyna Grygiel, Jakub Kozik, Christophe Raffalli, Guillaume Theyssier, and Marek Zaionc. Asymptotically almost all λ-terms are strongly normalizing. *Logical Methods in Computer Science*, 9(1), 2013.

9 Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, New York, NY, USA, 1 edition, 2009.

10 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.

11 Ilya Kapovich, Alexei G. Myasnikov, Paul Schupp, and Vladimir Shpilrain. Generic-case complexity, decision problems in group theory and random walks. *CoRR*, math.GR/0203239, 2002. URL: `http://arxiv.org/abs/math.GR/0203239`.

12 Teodor Knapik, Damian Niwinski, and Pawel Urzyczyn. Higher-order pushdown trees are easy. In *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002*, volume 2303 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2002.

13 Naoki Kobayashi. Type-based useless-variable elimination. *Higher-Order and Symbolic Computation*, 14(2-3):221–260, 2001.

14 Naoki Kobayashi. Model-checking higher-order functions. In *Proceedings of PPDP 2009*, pages 25–36. ACM Press, 2009.

15 Naoki Kobayashi. Types and higher-order recursion schemes for verification of higher-order programs. In *Proceedings of ACM SIGPLAN/SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 416–428. ACM Press, 2009.

16 Naoki Kobayashi. Model checking higher-order programs. *Journal of the ACM*, 60(3), 2013.

17 Naoki Kobayashi and C.-H. Ong. Complexity of Model Checking Recursion Schemes for Fragments of the Modal Mu-Calculus. *Logical Methods in Computer Science*, 7(4), jan 2012. URL: `http://arxiv.org/abs/1109.5267http://dx.doi.org/10.2168/LMCS-7(4:9)2011https://lmcs.episciences.org/1211`, `arXiv:1109.5267`, `doi:10.2168/LMCS-7(4:9)2011`.

18 Naoki Kobayashi and C.-H. Luke Ong. A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In *Proceedings of LICS 2009*, pages 179–188. IEEE Computer Society Press, 2009.

19 Naoki Kobayashi, Ryosuke Sato, and Hiroshi Unno. Predicate abstraction and CEGAR for higher-order model checking. In *Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 222–233. ACM Press, 2011.

20 Yoshiki Nakamura, Asada Kazuyuki, Naoki Kobayashi, Ryoma Sin'ya, and Takeshi Tsukada. On average-case hardness of higher-order model checking. Full version. Available from `https://www.kb.is.s.u-tokyo.ac.jp/~koba/papers/OnAverageCaseHOMC.pdf`.

21 C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006)*, pages 81–90. IEEE Computer Society Press, 2006.

22 C.-H. Luke Ong and Steven Ramsay. Verifying higher-order programs with pattern-matching algebraic data types. In *Proceedings of ACM SIGPLAN/SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 587–598. ACM Press, 2011.

23 C.-H.L. Ong. On Model-Checking Trees Generated by Higher-Order Recursion Schemes. In *21st Annual IEEE Symposium on Logic in Computer Science (LICS'06)*, pages 81–90. IEEE, 2006. URL: `http://ieeexplore.ieee.org/document/1691219/`, `doi:10.1109/LICS.2006.38`.

24 Steven Ramsay, Robin Neatherway, and C.-H. Luke Ong. An abstraction refinement approach to higher-order model checking. In *Proceedings of ACM SIGPLAN/SIGACT Symposium on Principles of Programming Languages (POPL)*, 2014.

25 Sylvain Salvati and Igor Walukiewicz. Krivine machines and higher-order schemes. In *Proceedings of ICALP 2011*, volume 6756 of *Lecture Notes in Computer Science*, pages 162–173. Springer, 2011.

26 Sylvain Salvati and Igor Walukiewicz. Recursive schemes, krivine machines, and collapsible pushdown automata. In Alain Finkel, Jérôme Leroux, and Igor Potapov, editors, *Reachability Problems - 6th International Workshop, RP 2012, Bordeaux, France, September 17-19, 2012. Proceedings*, volume 7550 of *Lecture Notes in Computer Science*, pages 6–20. Springer, 2012. URL: `https://doi.org/10.1007/978-3-642-33512-9_2`, `doi:10.1007/978-3-642-33512-9\_2`.

27 Ryoma Sin'ya, Kazuyuki Asada, Naoki Kobayashi, and Takeshi Tsukada. Almost every simply typed λ-term has a long β-reduction sequence. In *Foundations of Software Science and Computation Structures - 20th International Conference, FOSSACS 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 10203 of *Lecture Notes in Computer Science*, pages 53–68, 2017.

28 Richard Statman. On the lambda Y calculus. *APAL*, 130(1-3):325–337, 2004. `doi:10.1016/j.apal.2004.04.004`.

29 Ryota Suzuki, Koichi Fujima, Naoki Kobayashi, and Takeshi Tsukada. Streett automata model checking of higher-order recursion schemes. In *FSCD*, volume 84 of *LIPIcs*, pages 32:1–32:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

A Definition of Alternating Parity Tree Automata

▶ **Definition 26** (alternating parity tree automata)**.** *Let Σ be a ranked alphabet. An* alternating parity tree automaton *over Σ is a quadruple $\mathcal{A} = \langle Q, q_0, \delta, \Omega \rangle$, where*

- *Q is a finite set of states,*
- *$q_0 \in Q$ is the initial state,*
- *$\delta \colon Q \times \Sigma \to \mathbb{B}^+([m] \times Q)$ is the transition function, where m is the largest rank of symbols in $\mathrm{dom}(\Sigma)$; and $\mathbb{B}^+(X)$ denotes the set of positive boolean formulae over X.*
- *$\Omega \colon Q \to [p]$ assigns a priority to each state.*

A run *of an APT $\mathcal{A}$ over a Σ-tree T is a $(\mathrm{dom}(T) \times Q)$-labeled tree R such that: (1) $R(\varepsilon) = \langle \varepsilon, q_0 \rangle$; and (2) for every $\beta \in \mathrm{dom}(R)$ with $R(\beta) = \langle \alpha, q \rangle$, the formula $\delta(q, T(\alpha))$ evaluates to true when each variable in the set $\{\langle i, q' \rangle \mid \langle \alpha \cdot i, q' \rangle \in \bigcup_{j \in [\mathtt{r}_R(\beta)]} \{R(\beta \cdot j)\}\}$ is set to true. A run R is* accepting *if every infinite path β in R satisfies the* parity condition*: let $\beta = j_1 j_2 \cdots$ and for each $l \geq 1$, let q_l be such that $R(j_1 j_2 \ldots j_l) = \langle \alpha, q_l \rangle$ (for some α); then the largest priority that occurs infinitely often in $\Omega(q_0)\Omega(q_1)\Omega(q_2)\cdots$ is even. $\mathcal{A}$* accepts *T if there is an accepting run of $\mathcal{A}$ over T.*

B Proof of Lemma 18

To prove Lemma 18, we firstly introduce three lemmas.

▶ **Lemma 27.** *Let Σ be a finite ranked alphabet with $\#(\mathrm{dom}(\Sigma)) = \gamma$. The number of all Σ-trees of size n is bounded by γ^n for each $n \in \mathbb{N}$.*

Proof. It is well-known that any ranked tree can be represented without using parenthesis (*cf.* Polish notation). For example, a $\{\mathtt{a} \mapsto 0, \mathtt{b} \mapsto 2, \mathtt{c} \mapsto 1\}$-tree $t = \mathtt{c}(\mathtt{b}(\mathtt{a}, \mathtt{c}(\mathtt{a})))$ can be represented just as a word over $\mathrm{dom}(\Sigma)$: `cbaca`, which is the depth-first left-to-right traversal of t. Hence one can easily observe that there is an injection from the set of all Σ-trees of size n into the set $\mathrm{dom}(\Sigma)^n$ of all words over $\mathrm{dom}(\Sigma)$ of length n. The latter satisfies $\mathrm{dom}(\Sigma)^n = \gamma^n$. ◀

Since every linear contexts of size n over Σ can be regarded as a tree over $\Sigma \uplus \{[\,]\}$ of size $n+1$, the following is deduced.

▶ **Corollary 28.** *For any ranked alphabet Σ, there exists some real number γ such that the number of all affine contexts over Σ of size at most n is bounded by γ^n for each $n \in \mathbb{N}$.*

▶ **Lemma 29.** *Let A be a finite sequence of non-negative real numbers and B be a sequence of positive real numbers of the same length $\#(A) = \#(B) = n$. $\frac{\sum_{i \in [n]} A(i)}{\sum_{i \in [n]} B(i)}$ is bounded by $c = \max\left\{\frac{A(1)}{B(1)}, \cdots, \frac{A(n)}{B(n)}\right\}$.*

Proof.

$$\frac{\sum_{i \in [n]} A(i)}{\sum_{i \in [n]} B(i)} = \frac{\sum_{i \in [n]} \frac{A(i)}{B(i)} \cdot B(i)}{\sum_{i \in [n]} B(i)} \leq \frac{\sum_{i \in [n]} c \cdot B(i)}{\sum_{i \in [n]} B(i)} = c.$$

◀

The last lemma is similar to Lemma 13, but is modified for good affine contexts.

▶ **Lemma 30.** *Let $k \geq 1$. For each k, let ι and ξ be sufficiently large natural numbers. There is m such that the following holds: Let E be any second-order linear context and C be any affine context good for m such that $E[C] \in \hat{\Lambda}_n(k,\iota,\xi)$. Then there is an affine context $D \succeq t_{\mathrm{HARD},k}$ good for m such that $E[D] \in \hat{\Lambda}_n(k,\iota,\xi)$.*

Proof. Let m_0 be the natural number obtained by Lemma 13 and let $m = 1 + m_0 \times \max(\{2\} \cup \mathrm{rng}(\Sigma))$. We only write the case $C = \mathtt{a}(C_1,\ldots,C_{\Sigma(\mathtt{a})})$. (Other cases are proved in the same way.) Let C_i be such that $|C_i| = \max\{|C_1|,\ldots,|C_{\Sigma(\mathtt{a})}|\}$. Let E' be $E[\mathtt{a}(C_1,\ldots,C_{i-1},[\![]\!],C_{i+1},\ldots,C_{\Sigma(\mathtt{a})})]$. Then $|C_i| \geq m_0$ and $E[C] = E'[C_i]$, so by Lemma 13, there is $C'_i \succeq t_{\mathrm{HARD},k}$ such that $E'[C'_i] \in \hat{\Lambda}_n(k,\iota,\xi)$. Then $D = \mathtt{a}(C_1,\ldots,C_{i-1},C'_i,C_{i+1},\ldots,C_{\Sigma(\mathtt{a})})$ is an affine context good for m and $E[D] \in \hat{\Lambda}_n(k,\iota,\xi)$ (since $E[D] = E'[C'_i]$). ◀

The following is an immediate consequence of the last lemma.

▶ **Corollary 31.** *For any $t \in \hat{\Lambda}$ and $i \in [\mathtt{shn}\,(E^t_m)]$, there exists a good affine context C such that (1) $t_{\mathrm{HARD},k} \preceq C$; (2) $C : E^t_m{\bullet}i$; and (3) $E^t_m[\overrightarrow{C}] \in \hat{\Lambda}$, where $\overrightarrow{C}$ is a sequence of contexts obtained by replacing the i-th component of $\overrightarrow{C}^t_m$ by C.*

Then, we will prove that Lemma 18 is true if we take γ as a constant stated in Corollary 28 for $\Sigma_{\Lambda(k,\iota,\xi)}$. By Lemma 29,

$$\frac{\sum_{E\in\mathcal{E}^n_m} \#\left(S^E_{\mathtt{shn}(E)}\right)}{\sum_{E\in\mathcal{E}^n_m} \#(S^E_0)} \leq \frac{\#\left(S^E_{\mathtt{shn}(E)}\right)}{\#(S^E_0)}$$

holds for some $E \in \mathcal{E}^n_m$, thus it is suffice to show the following inequation for such E:

$$\frac{\#\left(S^E_{\mathtt{shn}(E)}\right)}{\#(S^E_0)} \leq 1 - \gamma^{-rm}. \tag{1}$$

If $S^E_{\mathtt{shn}(E)} = \emptyset$ the inequality (1) holds obviously, thus we assume $S^E_{\mathtt{shn}(E)}$ is non-empty. Since

$$\frac{\#\left(S^E_{\mathtt{shn}(E)}\right)}{\#(S^E_0)} = \frac{\#(S^E_1)}{\#(S^E_0)} \times \frac{\#(S^E_2)}{\#(S^E_1)} \times \cdots \times \frac{\#\left(S^E_{\mathtt{shn}(E)}\right)}{\#\left(S^E_{\mathtt{shn}(E)-1}\right)},$$

it is suffice to show that

$$\frac{\#(S^E_i)}{\#(S^E_{i-1})} \leq 1 - \gamma^{-rm} \tag{2}$$

holds for each $i \in [\mathtt{shn}\,(E)]$.

For $i \in [\mathtt{shn}\,(E)]$, we define:

$$\mathcal{D}_m(E,i) \triangleq \{C \in \mathcal{C}_m(E,i) \mid t_{\mathrm{HARD},k} \not\preceq C\}$$

$$\overrightarrow{\mathcal{D}}_m(E,i) \triangleq \left\{ (C_j)_{j\neq i} \in \prod_{j=1}^{i-1} \mathcal{D}_m(E,j) \times \prod_{j=i+1}^{\mathtt{shn}(E)} \mathcal{C}_m(E,i) \;\middle|\; \mathcal{C}_m(t,j) = C_j \ (j \neq i) \text{ for some } t \in \Phi^{-1}_m(E) \right\}.$$

Intuitively, $\mathcal{D}_m(E,i)$ consists of "non-hard" contexts appeared in i-th decomposed part of some minimal term in $\Phi^{-1}_m(E)$. For $(C_j)_{j\neq i} \in \mathcal{D}_m(E,j)$, we further define the number of "possible" contexts $N_m^{\mathcal{C}}((C_j)_{j\neq i})$ and the number of non-hard contexts $N_m^{\mathcal{D}}((C_j)_{j\neq i})$ that consistent with $(C_j)_{j\neq i}$ in minimal terms as follows:

$$N_m^{\mathcal{C}}((C_j)_{j\neq i}) \triangleq \#\left(\left\{C_i \in \mathcal{C}_m(E,i) \mid \overrightarrow{C}_m^t = C_1 \cdots C_{n-1} C_i C_{i+1} \cdots C_j \text{ for some } t \in \Phi_m^{-1}(E)\right\}\right)$$
$$N_m^{\mathcal{D}}((C_j)_{j\neq i}) \triangleq \#\left(\left\{C_i \in \mathcal{D}_m(E,i) \mid \overrightarrow{C}_m^t = C_1 \cdots C_{n-1} C_i C_{i+1} \cdots C_j \text{ for some } t \in \Phi_m^{-1}(E)\right\}\right)$$

Since S_{i-1}^E is non-empty, $\overrightarrow{\mathcal{D}}_m(E,i)$ is also non-empty. Further, by the definition of $\overrightarrow{\mathcal{D}}_m(E,i)$, $N_m^{\mathcal{C}}((C_j)_{j\neq i})$ is always positive. By regarding each $t \in \Phi_m^{-1}(E)$ as a sequence of extracted contexts (it is one-to-one if we fix E), we have

$$\#(S_i^E) = \sum_{(C_j)_{j\neq i} \in \overrightarrow{\mathcal{D}}_m(E,i)} N_m^{\mathcal{D}}((C_j)_{j\neq i})$$
$$\#(S_{i-1}^E) = \sum_{(C_j)_{j\neq i} \in \overrightarrow{\mathcal{D}}_m(E,i)} N_m^{\mathcal{C}}((C_j)_{j\neq i})$$

For each $\overrightarrow{\mathcal{D}}_m(E,i)$, by Corollary 31, there exists some $C \in \mathcal{C}_m(E,i) \setminus \mathcal{D}_m(E,i)$ such that $\overrightarrow{C}_m^t = C_1 \cdots C_{i-1} C C_{i+1} \cdots C_{\mathtt{shn}(E)}$ for some $t \in \Phi_m^{-1}(E)$. Thus we have

$$N_m^{\mathcal{D}}\left(\overrightarrow{\mathcal{D}}_m(E,i)\right) \leq N_m^{\mathcal{C}}\left(\overrightarrow{\mathcal{D}}_m(E,i)\right) - 1.$$

Moreover, because of the goodness for m, each element $C \in \mathcal{C}_m(E,i)$ satisfies $|C| \leq r(m-1)+1 \leq rm$ hence

$$\#(\mathcal{C}_m(E,i)) \leq \gamma^{rm}$$

by Corollary 28. Combining these two facts, the following holds

$$\frac{N_m^{\mathcal{D}}((C_j)_{j\neq i})}{N_m^{\mathcal{C}}((C_j)_{j\neq i})} \leq 1 - \frac{1}{N_m^{\mathcal{C}}((C_j)_{j\neq i})} \leq 1 - \frac{1}{\#(\mathcal{C}_m(E,i))} \leq 1 - \gamma^{-rm}.$$

Therefore, by Lemma 29, we obtain the inequality (2) as follows:

$$\frac{\#(S_i^E)}{\#(S_{i-1}^E)} = \frac{\sum_{(C_j)_{j\neq i} \in \mathcal{D}_m(E,i)} N_m^{\mathcal{D}}((C_j)_{j\neq i})}{\sum_{(C_j)_{j\neq i} \in \mathcal{D}_m(E,i)} N_m^{\mathcal{C}}((C_j)_{j\neq i})} \leq 1 - \gamma^{-rm}$$

for each $i \in [\mathtt{shn}(E)]$.

C Proof of Lemma 24

The *size* of a simple type κ and a simple type environment Γ, written $|\kappa|$ and $|\Gamma|$ respectively, is defined by: $|\kappa| \triangleq 1$ if $\kappa = \mathtt{o}$, $|\kappa| \triangleq 1+|\kappa_1|+|\kappa_2|$ if $\kappa = \kappa_1 \to \kappa_2$, and $|\Gamma| \triangleq 1+\sum_{x\in\mathrm{dom}(\Gamma)} |\Gamma(x)|$.

▶ **Definition 32.** *The term $t_{\Gamma,\kappa}$ is inductively defined as follows, where in the second case, $l = \min\{i \in [\xi] \mid z_i \in \mathrm{dom}(\Gamma)\}$; and in the third case, $l = \min\{i \in [\xi] \mid z_i \notin \mathrm{dom}(\Gamma)\}$:*

$$t_{\Gamma,\kappa} \triangleq \begin{cases} \mathtt{a} & (\kappa = \mathtt{o} \textit{ and } \Gamma = \emptyset) \\ \mathtt{b}(z_l t_{\emptyset,\kappa^1} \ldots t_{\emptyset,\kappa^m}, t_{\Gamma',\mathtt{o}}) & (\kappa = \mathtt{o} \textit{ and } \Gamma = (\Gamma', z_l : \kappa^1 \to \ldots \to \kappa^m \to \mathtt{o})) \\ \lambda z_l.t_{(\Gamma,z_l:\kappa'),\kappa''} & (\kappa = \kappa' \to \kappa'' \textit{ and } \#(\mathrm{dom}(\Gamma)) < \xi) \\ (\lambda z_1.t_{(z_1:\mathtt{o}),\kappa})\ t_{\Gamma,\mathtt{o}} & (\kappa = \kappa' \to \kappa'' \textit{ and } \mathtt{ar}(\kappa) < \iota) \\ \textit{undefined} & (\textit{otherwise}) \end{cases}.$$

▶ **Proposition 33.** *Suppose that $\langle \Gamma, \kappa \rangle$ is ($\langle k, \iota, \xi \rangle$-)bounded. If $\#(\mathrm{dom}(\Gamma)) < \xi$ or $\mathtt{ar}\,(\kappa) < \iota$, then (1) $t_{\Gamma,\kappa}$ is defined, (2) $\Gamma \vdash_{\mathrm{ST}} t_{\Gamma,\kappa} : \kappa$, and (3) $t_{\Gamma,\kappa}$ is bounded.*

Proof. By a straightforward induction on the parameter $\langle |\kappa|, |\Gamma| \rangle$. ◀

We now extend the above for intersection types.

▶ **Definition 34.** *The term $t_{\Theta,\bar{\theta}}$ is inductively defined as follows, where in the second case, $l = \min\{i \in [\xi] \mid z_i \in \mathrm{dom}(\Theta)\}$; and in the third case, $l = \min\{i \in [\xi] \mid z_i \notin \mathrm{dom}(\Theta)\}$:*

$$t_{\Theta,\bar{\theta}} \triangleq \begin{cases} \mathtt{a} & (\bar{\theta} = \mathtt{o} \text{ and } \Theta = \emptyset) \\ \mathtt{b}(\bigsqcup_{i \in [n]} z_l t_{\emptyset,\theta_i^1} \dots t_{\emptyset,\theta_i^m}, t_{\Theta',\mathtt{o}}) & (\bar{\theta} = \mathtt{o} \text{ and } \Theta = (\Theta', z_l : \bigwedge_{i \in [n]} \theta_i^1 \to \dots \to \theta_i^m \to \mathtt{o})) \\ \lambda z_l . t_{(\Theta, z_l : \theta'), \tau''} & (\bar{\theta} = \theta' \to \tau'' \text{ and } \#(\mathrm{dom}(\Theta)) < \xi) \\ (\lambda z_1 . t_{(z_1 : \wedge\{\mathtt{o}\}), \bar{\theta}})\ t_{\Theta,\mathtt{o}} & (\bar{\theta} = \theta' \to \tau'' \text{ and } \mathtt{ar}\,(\kappa) < \iota) \\ \bigsqcup_{i \in [n]} t_{\Theta,\tau_i} & (\bar{\theta} = \bigwedge_{i \in [n]} \tau_i \text{ and } n \geq 1) \\ \bot^{\kappa} & (\bar{\theta} = \top^{\kappa} \text{ and } \Theta = \emptyset) \\ \textit{undefined} & (\textit{otherwise}) \end{cases}.$$

▶ **Proposition 35.** *Suppose that $\langle \Theta, \bar{\theta} \rangle :: \langle \Gamma, \kappa \rangle$ for some bounded $\langle \Gamma, \kappa \rangle$. If $\#(\mathrm{dom}(\Gamma)) < \xi$, $\mathtt{ar}\,(\kappa) < \iota$, or $\langle \Theta, \bar{\theta} \rangle = \langle \emptyset, \top \rangle$, then (1) $t_{\Theta,\bar{\theta}}$ is defined, (2) $t_{\Theta,\bar{\theta}} \sqsubseteq t_{\Gamma,\kappa}$, (3) $\Theta \vdash t_{\Theta,\bar{\theta}} : \bar{\theta}$, and (4) $t_{\Theta,\bar{\theta}}$ is bounded.*

Proof. By a straightforward induction on the parameter $\langle |\kappa|, |\Gamma| \rangle$. The existence of the join in each case can be ensured by the assumption (2). ◀

We now extend the above for context-types (i.e., for Lemma 24).

▶ **Definition 36.** *The linear-context $C_{\tilde{\tau}}$ is inductively defined as follows, where in the second case, $l = \min\{i \in [\xi] \mid z_i \notin \mathrm{dom}(\Theta)\}$:*

$$C_{\langle \Theta, \tau \rangle} \triangleq \begin{cases} \mathtt{b}(t_{\Theta,\mathtt{o}}, [\,]) & (\tau = \mathtt{o}) \\ \lambda z_l . C_{\langle (\Theta, z_l : \theta'), \tau'' \rangle} & (\tau = \theta' \to \tau'' \text{ and } \#(\mathrm{dom}(\Theta)) < \xi) \\ (\lambda z_1 . t_{(z_1 : \wedge\{\mathtt{o}\}), \tau})\ C_{\langle \Theta, \mathtt{o} \rangle} & (\tau = \theta' \to \tau'' \text{ and } \mathtt{ar}\,(\tau) < \iota) \\ \textit{undefined} & (\textit{otherwise}) \end{cases}.$$

For each $\tilde{\theta}^+ = \{\tilde{\tau}_1, \dots, \tilde{\tau}_n\}$, let $C_{\tilde{\theta}^+} \triangleq \bigsqcup_{i \in [n]} C_{\tilde{\tau}_i}$. This is well-defined by using Proposition 35(2).

▶ **Proposition 37.** *Suppose that $\tilde{\theta}^+ :: \langle \Gamma, \kappa \rangle$ for some bounded $\langle \Gamma, \kappa \rangle$. If $\#(\mathrm{dom}(\Gamma)) < \xi$ or $\mathtt{ar}\,(\kappa) < \iota$, then (1) $C_{\tilde{\theta}^+}$ is defined, (2) $C_{\tilde{\theta}^+} \lhd \{\langle \emptyset, \mathtt{o} \rangle\} \Rrightarrow \tilde{\theta}$, and (3) $C_{\tilde{\theta}^+}$ is bounded.*

Proof. By a straightforward induction on the parameter $\langle |\kappa|, |\Gamma| \rangle$. ◀

▶ **Definition 38.** *The linear-context $D_{\tilde{\tau}}$ is defined as follows, where in the first case, $l = \min\{i \in [\xi] \mid z_i \in \mathrm{dom}(\Theta)\}$; and in the second case, $\tau = \theta^1 \to \dots \to \theta^m \to \mathtt{o}$:*

$$D_{\langle \Theta, \tau \rangle} \triangleq \begin{cases} (\lambda z_l . D_{\langle \Theta', \tau \rangle})\, t_{\emptyset,\theta_l} & (\Theta = (\Theta', z_l : \theta_l)) \\ \mathtt{c}([\,]\, t_{\emptyset,\theta^1} \ \dots\ t_{\emptyset,\theta^m}) & (\Theta = \emptyset) \end{cases}.$$

Let $D_{\tilde{\theta}^+} \triangleq \bigsqcup_{i \in [n]} D_{\tilde{\tau}_i}$ for each $\tilde{\theta}^+ = \{\tilde{\tau}_1, \dots, \tilde{\tau}_n\}$. This is well-defined by using Proposition 35(2). Also, specially, let $D_{\emptyset} \triangleq \mathtt{a}$.

▶ **Proposition 39.** *Suppose that $\tilde{\theta} :: \langle \Gamma, \kappa \rangle$ for some bounded $\langle \Gamma, \kappa \rangle$. Then, (1) $D_{\tilde{\theta}}$ is defined, (2) $D_{\tilde{\theta}} \lhd \tilde{\theta} \Rrightarrow \{\langle \emptyset, \mathtt{o} \rangle\}$, and (3) $D_{\tilde{\theta}}$ is bounded.*

Proof. By a straightforward induction on the parameter $\langle |\kappa|, |\Gamma| \rangle$. ◀

As a consequence of Proposition 37 and 39, Lemma 24 has been proved.

C.1 On the Boundary Case for Lemma 24(1)

Here, we consider the boundary case for Lemma 24(1), i.e., $\Gamma \vdash_{\mathrm{ST}} t : \kappa$, t is $\langle k, \iota, \xi \rangle$-bounded, $\#(\mathrm{dom}(\Gamma)) = \xi$, and $\mathtt{ar}\,(\kappa) = \iota$. Actually in this case, t should be of a special form.

▶ **Lemma 40.** *Suppose that* (1) $\Gamma \vdash_{\mathrm{ST}} t : \kappa$, (2) t *is* $\langle k, \iota, \xi \rangle$*-bounded,* (3) $\#(\mathrm{dom}(\Gamma)) = \xi$, *and* (4) $\mathtt{ar}\,(\kappa) = \iota$. *Then, t is α-equivalent to a term of the form* $\lambda\_.t_1$.

Proof. By $\xi > 1$, $t \neq x$ and $t \neq \bot$. By $\iota > 0$, $t \neq a(t_1, \dots, t_{\Sigma(a)})$. By $\mathtt{ar}\,(\kappa) = \iota$, $t \neq t_1 t_2$ and $t \neq \mathbf{Y} t_1$. Therefore t is of the form $\lambda \bar{x}.t_1$. By that t is bounded and $\#(\mathrm{dom}(\Gamma)) = \xi$, the last rule of $\Gamma \vdash_{\mathrm{ST}} \lambda \bar{x}.t_1 : \kappa$ should be (Abs2), so $\Gamma \vdash_{\mathrm{ST}} t_1 : \kappa''$, where $\kappa = \kappa' \to \kappa''$. Then $\bar{x}$ does not occur in t_1 as a free variable. Therefore t is α-equivalent to the term $\lambda\_.t_1$. ◀

D Proof of Proposition 22 and 23

▶ **Lemma 41.** *Suppose that C is a linear-context. If $C \lhd \tilde{\theta}' \Rrightarrow \tilde{\tau}$ and $C' \lhd \tilde{\theta}'' \Rrightarrow \tilde{\theta}'$, then $C[C'] \lhd \tilde{\theta}'' \Rrightarrow \{\tilde{\tau}\}$.*

Proof. Let $\tilde{\theta}' = \{\tilde{\tau}'_1, \dots, \tilde{\tau}'_n\}$. By $C' \lhd \tilde{\theta}'' \Rrightarrow \tilde{\theta}'$, there exists $\{\langle \tilde{\theta}''_{i,j}, C'_{i,j} \rangle\}_{i \in [n], j \in [k_i]}$ such that $C' = \bigsqcup_{i \in [n], j \in [k_i]} C'_{i,j}$, $\tilde{\theta}'' = \bigcup_{i \in [n], j \in [k_i]} \tilde{\theta}''_{i,j}$, and $C'_{i,j} \lhd \tilde{\theta}''_{i,j} \Rrightarrow \tilde{\tau}'_i$. Here, we can assume that $k_1 = \cdots = k_n$ (so, we denote them by k). Then from the derivation tree of $C \lhd \tilde{\theta}' \Rrightarrow \tilde{\tau}$ (see the left-hand side below), we can construct a derivation tree of $C[C'] \lhd \tilde{\theta}'' \Rrightarrow \tilde{\tau}$ (see the right-hand side below) as follows, where $\tilde{\tau} = \langle \Theta, \tau \rangle$ and $f \colon [m] \to [n']$ is a surjective map:

$$\frac{\mathtt{x} \lhd \tilde{\tau}'_{f(1)} \quad \dots \quad \mathtt{x} \lhd \tilde{\tau}'_{f(m)}}{\Theta \vdash C[\mathtt{x}] : \tau}\,\mathcal{T} \quad \rightsquigarrow \quad \frac{\dfrac{C'_{f(1),1} \lhd \tilde{\theta}''_{f(1),1} \Rrightarrow \tilde{\tau}'_{f(1)} \quad \dots \quad C'_{f(m),1} \lhd \tilde{\theta}''_{f(m),1} \Rrightarrow \tilde{\tau}'_{f(m)}}{\Theta \vdash C[\bigsqcup_{i \in [n]} C'_{i,1}] : \tau}\,\mathcal{T} \quad \dots \quad \dfrac{\dots}{\Theta \vdash C[\bigsqcup_{i \in [n]} C'_{i,k}] : \tau}\,\mathcal{T}}{\Theta \vdash C[C'] : \tau}\,(\wedge) .$$ ◀

Proof of Proposition 22. Let $\tilde{\theta}' = \{\tilde{\tau}'_1, \dots, \tilde{\tau}'_{n'}\}$ and $\tilde{\theta} = \{\tilde{\tau}_1, \dots, \tilde{\tau}_n\}$. By $C \lhd \tilde{\theta}' \Rrightarrow \tilde{\theta}$, there exists $\{\langle \tilde{\theta}'_i, C_i \rangle\}_{i \in [m]}$ such that $C = \bigsqcup_{i \in [m]} C_i$, $\tilde{\theta}' = \bigcup_{i \in [m]} \tilde{\theta}'_i$, and $C_i \lhd \tilde{\theta}'_i \Rrightarrow \tilde{\tau}_{f(i)}$. By $C' \lhd \tilde{\theta}'' \Rrightarrow \tilde{\theta}'$, there exists $\{\langle \tilde{\theta}''_j, C''_j \rangle\}_{j \in [n']}$ such that $C' = \bigsqcup_{j \in [n']} C''_j$, $\tilde{\theta}'' = \bigcup_{j \in [n']} \tilde{\theta}''_j$, and $C''_j \lhd \tilde{\theta}''_j \Rrightarrow \{\tilde{\tau}_j\}$. Let $C'_i = \bigsqcup_{j \in [n']; \tilde{\tau}'_j \in \tilde{\theta}'_i} C''_j$ and let $\tilde{\theta}''_i = \bigcup_{j \in [n']; \tilde{\tau}'_j \in \tilde{\theta}'_i} \tilde{\theta}''_j$. Then $C'_i \lhd \tilde{\theta}''_i \Rrightarrow \tilde{\theta}'_i$. By Lemma 41, $C_i[C'_i] \lhd \tilde{\theta}''_i \Rrightarrow \tilde{\tau}_{f(i)}$. Therefore, $C[C'] \lhd \tilde{\theta}'' \Rrightarrow \tilde{\theta}$. ◀

▶ **Lemma 42.** *Suppose that C is a linear-context. If $C[C'] \lhd \tilde{\theta}'' \Rrightarrow \tilde{\tau}$, then $C \lhd \tilde{\theta}' \Rrightarrow \tilde{\tau}$ and $C' \lhd \tilde{\theta}'' \Rrightarrow \tilde{\theta}'$ for some $\tilde{\theta}'$.*

Proof. Then (the derivation tree of) $C[C'] \lhd \tilde{\theta}'' \Rrightarrow \tilde{\tau}$ should be of the form in the right-hand side below, where $\tilde{\tau} = \langle \Theta, \tau \rangle$, $C' = \bigsqcup_{i \in [m]} C'_i$, and $\tilde{\theta}'' = \bigcup_{i \in [m]} \tilde{\theta}''_i$. We let $\tilde{\theta}' = \{\tilde{\tau}'_1, \dots, \tilde{\tau}'_m\}$. Then, $C' \lhd \tilde{\theta}'' \Rrightarrow \tilde{\theta}'$ is immediate and $C \lhd \tilde{\theta}' \Rrightarrow \tilde{\tau}$ is shown by replacing each subterm arise from t to $\mathtt{x}$ (see the left-hand side below).

$$\frac{\mathtt{x} \lhd \tilde{\tau}'_1 \quad \dots \quad \mathtt{x} \lhd \tilde{\tau}'_m}{\Theta \vdash C[\mathtt{x}] : \tau} \quad \leftsquigarrow \quad \frac{C'_1 \lhd \tilde{\theta}''_1 \Rrightarrow \tilde{\tau}'_1 \quad \dots \quad C'_m \lhd \tilde{\theta}''_m \Rrightarrow \tilde{\tau}'_m}{\Theta \vdash C[C'] : \tau}.$$ ◀

Proof of Proposition 23. Let $\tilde{\theta}'' = \{\tilde{\tau}''_1, \dots, \tilde{\tau}''_{n''}\}$ and $\tilde{\theta} = \{\tilde{\tau}_1, \dots, \tilde{\tau}_n\}$. By $C[C'] \lhd \tilde{\theta}'' \Rrightarrow \tilde{\theta}$, there are a surjective map $f \colon [m] \to [n]$ and a sequence $\{\langle C_i, C'_i, \tilde{\theta}''_i \rangle\}_{i \in [m]}$ such that $C_i[C'_i] \lhd \tilde{\theta}''_i \Rrightarrow \tilde{\tau}_{f(i)}$, $C = \bigsqcup_{i \in [m]} C_i$, $C' = \bigsqcup_{i \in [m]} C'_i$, and $\tilde{\theta}'' = \bigcup_{i \in [m]} \tilde{\theta}''_i$ (see also Proposition 45 in the full version [20]). By Lemma 42, $C_i \lhd \tilde{\theta}'_i \Rrightarrow \tilde{\tau}_{f(i)}$ and $C'_i \lhd \tilde{\theta}''_i \Rrightarrow \tilde{\theta}'_i$ for some $\tilde{\theta}'_i$. We now let $\tilde{\theta}' = \bigcup_{j \in [m]} \tilde{\theta}'_i$. Then, both $C' \lhd \tilde{\theta}'' \Rrightarrow \tilde{\theta}'$ and $C \lhd \tilde{\theta}' \Rrightarrow \tilde{\theta}$ are immediate. ◀

E Properties of the Approximate Relation

In this section we list some properties of the approximate relation $\sqsupseteq$.

▶ **Proposition 43.**

(1) *If $s \sqsupseteq s'$, then $t\{s/x\} \sqsupseteq t\{s'/x\}$.*
(2) *If $s \sqsupset s'$ and $x \in \mathbf{FV}(t)$, then $t\{s/x\} \sqsupset t\{s'/x\}$.*
(3) *If $t' \sqsupseteq t$, then $t'\{s/x\} \sqsupseteq t\{s/x\}$.*
(4) *If $t' \sqsupset t$ and $s \neq \bot$, then $t'\{s/x\} \sqsupset t\{s/x\}$.*

Proof. By simple induction on the structure of t. ◀

▶ **Proposition 44.** *If $s, u \sqsubseteq t$ for some t, then the join $s \sqcup u$ is defined.*

Proof. By induction on the structure of t. If $s = \bot$ or $u = \bot$, then the existence of $s \sqcup u$ is obvious. Otherwise we do case analysis on the structure of t. We only write the case of $t = t_1\, t_2$ (Other cases are shown in the same way). By $t \sqsupseteq s \neq \bot$, s is of the form $s_1\, s_2$. As well for u, u is of the form $u_1\, u_2$. For each $l \in [2]$, by $s_l, u_l \sqsubseteq t_l$ and I.H., $s_l \sqcup u_l$ is defined. Then $t' = (s_1 \sqcup u_1)\,(s_2 \sqcup u_2)$ is the join of s and u, i.e, for every t'' such that $s, u \sqsubseteq t''$, $t' \sqsubseteq t''$. We now show it. By $s, u \sqsubseteq t''$, t'' is of the form $t''_1\, t''_2$ and also $s_1, u_1 \sqsubseteq t''_1$ and $s_2, u_2 \sqsubseteq t''_2$ hold. Therefore by $s_1 \sqcup u_1 \sqsubseteq t''_1$ and $s_2 \sqcup u_2 \sqsubseteq t''_2$, $t' \sqsubseteq t''$ has been proved. ◀

We write $\mathbf{FVC}_{\mathtt{x}}(t)$ for the number of occurrences of $\mathtt{x}$ in t as a free variable. We say that a substitution $t\{s/\mathtt{x}\}$ (more formally, a tuple $\langle t, s, \mathtt{x}\rangle$) is *conservative* if the following holds: (1) $\mathbf{FVC}_{\mathtt{x}}(t) \leq 1$; and (2) if $\mathtt{x} \notin \mathbf{FV}(t)$, then $s = \bot$. By this restriction, the following useful proposition holds.

▶ **Proposition 45.**

(1) *If $t\{s/\mathtt{x}\} = \bigsqcup_{l\in[n]} u_l$ and $t\{s/\mathtt{x}\}$ is a conservative substitution, then there is $\{\langle t_l, s_l\rangle\}_{l\in[n]}$ such that (a) for each $l \in [n]$, $t_l\{s_l/\mathtt{x}\} = u_l$ and $t_l\{s_l/\mathtt{x}\}$ is a conservative substitution; (b) $t = \bigsqcup_{l\in[n]} t_l$; and (c) $s = \bigsqcup_{l\in[n]} s_l$.*
(2) *If (a) for each $l \in [n]$, $t_l\{s_l/\mathtt{x}\} = u_l$ and $t_l\{s_l/\mathtt{x}\}$ is a conservative substitution; (b) $t = \bigsqcup_{l\in[n]} t_l$; and (c) $s = \bigsqcup_{l\in[n]} s_l$, then $t\{s/\mathtt{x}\} = \bigsqcup_{l\in[n]} u_l$ (and $t\{s/\mathtt{x}\}$ is a conservative substitution).*

Proof. (1): By induction on the structure of t. Without loss of generality, we can assume that, for each $l \in [n]$, $u_l \neq \bot$ (if $u_l = \bot$, let $\langle t_l, s_l\rangle = \langle \bot, \bot\rangle$).

Case $t = \mathtt{x}$: For each l, let $\langle t_l, s_l\rangle = \langle \mathtt{x}, u_l\rangle$. Then (a)(b)(c) hold.

Case $t = x$ (for $x \neq \mathtt{x}$) or $t = \bot$: By $\mathtt{x} \notin \mathbf{FV}(t)$, $s = \bot$. For each l, let $\langle t_l, s_l\rangle = \langle u_l, \bot\rangle$. Then (a)(b)(c) hold.

Case $t = t^1\, t^2$: Then (i) $t\{s/\mathtt{x}\} = (t^1\{s/\mathtt{x}\})\, t^2$; or (ii) $t\{s/\mathtt{x}\} = t^1\, (t^2\{s/\mathtt{x}\})$ holds, because $\mathbf{FVC}_{\mathtt{x}}(t) \leq 1$. We only write case (i) (in the same way for (ii)). For each l, by $(t^1\{s/\mathtt{x}\})\, t^2 \sqsupseteq u_l \neq \bot$, u_l is of the form $u^1_l\, u^2_l$. Then $t^1\{s/\mathtt{x}\} = \bigsqcup_{l\in[n]} u^1_l$ and $t^2 = \bigsqcup_{l\in[n]} u^2_l$. By I.H., there is $\{\langle t^1_l, s'_l\rangle\}_{l\in[n]}$ such that (a') for each $l \in [n]$, $t^1_l\{s'_l/\mathtt{x}\} = u^1_l$ and $t^1_l\{s'_l/\mathtt{x}\}$ is a conservative substitution; (b') $t^1 = \bigsqcup_{l\in[n]} t^1_l$; and (c') $s = \bigsqcup_{l\in[n]} s'_l$. For each l, let $\langle t_l, s_l\rangle = \langle t^1_l\, u^2_l, s'_l\rangle$. Then (a)(b)(c) hold by using the above (a')(b')(c').

Case $t = \lambda x.t_1$, $t = \mathbf{Y} t_1$, or $t = a(t_1, \ldots, t_{\Sigma(a)})$: In the same way as Case $t = t^1\, t^2$.

(2): It suffices to show the case when $n = 2$. $t\{s/\mathtt{x}\} \sqsupseteq \bigsqcup_{l\in[2]} u_l$ is shown by Proposition 43. We now show $t\{s/\mathtt{x}\} \sqsubseteq \bigsqcup_{l\in[2]} u_l$ by induction on the structure of t. If $\mathtt{x} \notin \mathbf{FV}(t_1)$, then by this and $s_1 = \bot$, $(t_l \sqcup t_2)\{s_1 \sqcup s_2/\mathtt{x}\} = t_1 \sqcup t_2\{s_2/\mathtt{x}\} = u_1 \sqcup u_2$. Similar for $\mathtt{x} \notin \mathbf{FV}(t_2)$. Otherwise we can assume that, $\mathtt{x} \in \mathbf{FV}(t)$. We now do case analysis on the structure of t.

Case $t = x$ (for $x \neq \mathtt{x}$) or $t = \bot$: This case does not occur by $\mathtt{x} \notin \mathbf{FV}(t)$.

Case $t = \mathtt{x}$: Then $t_1 = t_2 = \mathtt{x}$, so $t\{s/\mathtt{x}\} = s = t_1\{s_1/\mathtt{x}\} \sqcup t_2\{s_2/\mathtt{x}\}$.

Case $t = t^1\,t^2$: Then (i) $t\{s/\mathtt{x}\} = (t^1\{s/\mathtt{x}\})\,t^2$; or (ii) $t\{s/\mathtt{x}\} = t^1\,(t^2\{s/\mathtt{x}\})$ holds, because $\mathbf{FVC}_{\mathtt{x}}(t) \leq 1$. We only write case (i) (in the same way for (ii)). For each l, by $t^1\,t^2 \sqsupseteq t_l \neq \bot$, t_l is of the form $t^1_l\,t^2_l$. By I.H., $t^1\{s/\mathtt{x}\} = t^1_1\{s_1/\mathtt{x}\} \sqcup t^1_2\{s_2/\mathtt{x}\}$. Therefore $t\{s/\mathtt{x}\} = (t^1\{s/\mathtt{x}\})\,t^2 = (t^1_1\{s_1/\mathtt{x}\} \sqcup t^1_2\{s_2/\mathtt{x}\})\,t^2 = (t^1_1\{s_1/\mathtt{x}\}\,t^2_1) \sqcup (t^1_2\{s_2/\mathtt{x}\}\,t^2_2) = t_1\{s_1/\mathtt{x}\} \sqcup t_2\{s_2/\mathtt{x}\}$.

Case $t = \lambda x.t_1$, $t = \mathbf{Y}t_1$, or $t = a(t_1, \ldots, t_{\Sigma(a)})$: In the same way as Case $t = t^1\,t^2$. ◀

The following is immediate from Proposition 45(1).

▶ **Proposition 46** (Cor. of Prop. 45(1))**.** *Assume that $u \sqsubseteq t\{s/\mathtt{x}\}$ and $t\{s/\mathtt{x}\}$ is a conservative substitution. Then there is $\langle t', s'\rangle$ such that (a) $u = t'\{s'/\mathtt{x}\}$ and $t'\{s'/\mathtt{x}\}$ is a conservative substitution, (b) $t' \sqsubseteq t$, and (c) $s' \sqsubseteq s$.*

In fact Proposition 45 holds even for the substitution in non-capture avoiding manner (the proof is proceeded in the same manner). We write $t[s/x]$ for the term obtained from t by substituting s for all the free occurrences of x in *non*-capture-avoiding manner. The following proposition is used for the substitution in linear contexts (see Proposition 23).

▶ **Proposition 47.**

(1) *If $t[s/\mathtt{x}] = \bigsqcup_{l\in[n]} u_l$ and $t[s/\mathtt{x}]$ is a conservative substitution, then there is $\{\langle t_l, s_l\rangle\}_{l\in[n]}$ such that (a) for each $l \in [n]$, $t_l[s_l/\mathtt{x}] = u_l$ and $t_l[s_l/\mathtt{x}]$ is a conservative substitution; (b) $t = \bigsqcup_{l\in[n]} t_l$; and (c) $s = \bigsqcup_{l\in[n]} s_l$.*

(2) *If (a) for each $l \in [n]$, $t_l[s_l/\mathtt{x}] = u_l$ and $t_l[s_l/\mathtt{x}]$ is a conservative substitution; (b) $t = \bigsqcup_{l\in[n]} t_l$; and (c) $s = \bigsqcup_{l\in[n]} s_l$, then $t[s/\mathtt{x}] = \bigsqcup_{l\in[n]} u_l$ (and $t[s/\mathtt{x}]$ is a conservative substitution).*

The following is a proposition between $\sqsupseteq$ and $\longrightarrow$. We write $\longrightarrow^{\leq 1}$ for the relation $(\longrightarrow) \cup (=)$.

▶ **Proposition 48.**

(1) *If $s \sqsupseteq t$ and $t \longrightarrow t'$, then $s \longrightarrow^{\leq 1} s'$ and $s' \sqsupseteq t'$ for some s', i.e., $(\sqsupseteq\longrightarrow) \subseteq (\longrightarrow^{\leq 1}\sqsupseteq)$ holds.*

(2) *If $t \sqsupseteq s$ and $t \longrightarrow t'$, then $s \longrightarrow^{\leq 1} s'$ and $t' \sqsupseteq s'$ for some s'.*

Proof. By simple induction on the derivation tree of $t \longrightarrow t'$. ◀

▶ **Proposition 49.** *If $t \sqsupseteq s$, then $T(t) \sqsupseteq T(s)$.*

Proof. It suffices to show that, for every $\Sigma^{\bot}$-tree V, if $s \longrightarrow^{*}\sqsupseteq V$, then $t \longrightarrow^{*}\sqsupseteq V$. It is shown by $t \sqsupseteq s \longrightarrow^{*}\sqsupseteq V$ and Proposition 48. ◀

F An Alternative Definition of the Minimality

In this section, we introduce an alternative definition of the minimality using label and we show that the minimality is equivalent to the minimality of Definition 8. This definition will be used to prove Theorem 19 (Appendix H) and Proposition 10 (Appendix G).

To define it, we introduce the special tree constructor ℓ (disjoint with Σ) of arity 1, called *label*. Let $\Sigma^{\ell} \triangleq \Sigma \uplus \{\ell\}$. We say that a term is *labelled* if ℓ occurs in the term. For each term t, we define the term t^{ℓ} as follows, where $\Gamma \vdash_{\mathrm{ST}} t : \kappa_1 \to \ldots \to \kappa_k \to \mathtt{o}$:

$$t^{\ell} \quad ::= \quad \lambda z_1^{\kappa_1}.\ldots.\lambda z_k^{\kappa_k}.\ell(t z_1 \ldots z_k).$$

We define the following operation $\natural$. Intuitively $\natural(t)$ denotes the term obtained from t by replacing each occurrence of the form $\ell(u)$ to u, repeatedly.

▶ **Definition 50.** *The term $\natural(t)$ is inductively defined as follows:*

- $\natural(x) = x$
- $\natural(\lambda\bar{x}.t) = \lambda\bar{x}.\natural(t)$
- $\natural(t_1\,t_2) = \natural(t_1)\,\natural(t_2)$
- $\natural(\mathbf{Y}t_1) = \mathbf{Y}\natural(t_1)$
- $\natural(\perp) = \perp$
- $\natural(a(t_1,\ldots,t_{\Sigma(a)})) = a(\natural(t_1),\ldots,\natural(t_{\Sigma(a)}))$ $\qquad$ *$(a \in \Sigma)$*
- $\natural(\ell(t_1)) = \natural(t_1)$

The following proposition can be shown by a straightforward induction.

▶ **Proposition 51.**

(1) *If $t \longrightarrow^*\sqsupseteq t'$, then $\natural(t) \longrightarrow^*\sqsupseteq \natural(t')$.*
(2) *If $\natural(t) = s$ and $s \longrightarrow^*\sqsupseteq s'$, then $t \longrightarrow^*\sqsupseteq t'$ and $s' = \natural(t')$ for some t'.*

We say that a term t is *tracked* (by ℓ) if there is $\langle C, u\rangle$ such that $t = C[\ell(u)]$ and $\natural u \neq \perp$. Then, the goal of this section is to show the following.

▶ **Theorem 52** (Alternative definition of the minimality)**.** *Let t be a closed and ground-typed term over Σ. Then, t is minimal if and only if for every $\langle C, s\rangle$ that $t = C[s]$ and $s \neq \perp$, there is a tracked finite tree V such that $C[s^\ell] \longrightarrow^*\sqsupseteq V$.*

F.1 Proof of Theorem 52

In this subsection, we prove Theorem 52. First, the following holds for the minimality.

▶ **Proposition 53.** *Let t be a closed and ground-typed term over Σ. Then, t is minimal if and only if for every $\langle C, s\rangle$ that $t = C[s]$ and $s \neq \perp$, $T(C[\perp]) \sqsubset T(C[s])$.*

Proof. ($\Longrightarrow$): By $C[\perp] \sqsubset C[s]$. ($\Longleftarrow$): It suffices to show the following: If $t = C[t_1,\ldots,t_n]$ and $t_i \neq \perp$ holds for some i, then $T(C[\perp,\ldots,\perp]) \sqsubset T(C[t_1,\ldots,t_n])$. It is shown by using the assumption as follows: $T(C[\perp,\ldots,\perp]) \sqsubseteq T(C[t_1,\ldots,t_{i-1},\perp,t_{i+1},\ldots,t_n]) \sqsubset T(C[t_1,\ldots,t_n])$. ◀

From this, to prove Theorem 52, it suffices to show the following (1) $\Leftrightarrow$ (3).

▶ **Lemma 54.** *For each closed and ground-typed term $C[s]$ over Σ, the following are equivalent:*

(1) *$T(C[\perp]) \sqsubset T(C[s])$;*
(2) *$T(\natural C[\perp^\ell]) \sqsubset T(\natural C[s^\ell])$; and*
(3) *there is a tracked finite tree V such that $C[s^\ell] \longrightarrow^*\sqsupseteq V$.*

To prove Lemma 54, we introduce the following operation $\flat$. Intuitively, $\flat(t)$ denotes the term obtained from t by replacing each occurrence of the form $\ell(u)$ to $\ell(\perp)$.

▶ **Definition 55.** *The term $\flat(t)$ is inductively defined as follows:*

- $\flat(x) = x$
- $\flat(\lambda\bar{x}.t) = \lambda\bar{x}.\flat(t)$

- $\flat(t_1\, t_2) = \flat(t_1)\,\flat(t_2)$
- $\flat(\mathbf{Y}t_1) = \mathbf{Y}\flat(t_1)$
- $\flat(\bot) = \bot$
- $\flat(a(t_1, \ldots, t_{\Sigma(a)})) = a(\flat(t_1), \ldots, \flat(t_{\Sigma(a)})) \qquad (a \in \Sigma)$
- $\flat(\ell(t_1)) = \ell(\bot)$

The following proposition can be shown by a straightforward induction.

▶ **Proposition 56.**

(1) *If $t \longrightarrow^* \sqsupseteq t'$, then $\flat(t) \longrightarrow^* \sqsupseteq \flat(t')$.*
(2) *If $\flat(t) = s$ and $s \longrightarrow^* \sqsupseteq s'$, then $t \longrightarrow^* \sqsupseteq t'$ and $s' = \flat(t')$ for some t'.*
Also the following holds between $\natural$ and $\flat$.

▶ **Proposition 57.** *If t is not tracked, then $\natural(\flat(t)) = \natural(t)$.*

Proof. By induction on t. We only write the case $t = \ell(t_1)$. Then note that $\natural(t_1) = \bot$ holds, because t is not tracked. From this, $\natural(\flat(t)) = \natural(\ell(\bot)) = \bot = \natural(t_1) = \natural(t)$. ◀

We now prove Lemma 54.

Proof of Lemma 54. (1) $\Longleftrightarrow$ (2): By η-conversion (note that $T(C[u]) = T(\natural C[u^\ell])$ holds, for every Σ-term $C[u]$). (3) $\Longrightarrow$ (2): Without loss of generality, we can take a tracked finite tree V such that $V = D[\ell(u)]$ and ℓ does not occur in D. By $C[s^\ell] \longrightarrow^* \sqsupseteq D[\ell(u)]$ (and Proposition 51(1)), $\natural(C[s^\ell]) \longrightarrow^* \sqsupseteq \natural(D[\ell(u)])$, so $\natural(C[s^\ell]) \longrightarrow^* \sqsupseteq D[\natural u]$. Assume (towards contradiction) that $T(\natural(C[\bot^\ell])) \sqsupseteq T(\natural(C[s^\ell]))$. By $T(C[\bot^\ell]) \sqsupseteq T(\natural(C[\bot^\ell]))$ and this assumption, and $\natural(C[s^\ell]) \longrightarrow^* \sqsupseteq D[\natural u]$, $C[\bot^\ell] \longrightarrow^* \sqsupseteq D[\natural u] \ldots (\star 1)$. Also by $C[s^\ell] \longrightarrow^* \sqsupseteq D[\ell(u)]$ (and Proposition 56(1)), $\flat(C[s^\ell]) \longrightarrow^* \sqsupseteq \flat(D[\ell(u)])$, so $C[\bot^\ell] \longrightarrow^* \sqsupseteq D[\ell(\bot)] \ldots (\star 2)$. By $(\star 1)$ and $(\star 2)$, $D[\natural u] \sqcup D[\ell(\bot)]$ is defined, but it is contradiction because $\natural u \neq \bot$ (since V is tracked). Therefore $T(C[\natural(C[\bot^\ell])]) \not\sqsupseteq T(\natural(C[s^\ell]))$, and thus $T(C[\natural(C[\bot^\ell])]) \neq T(\natural(C[s^\ell])$. Hence $T(C[\natural(C[\bot^\ell])]) \sqsubset T(\natural(C[s^\ell])$ has been proved (since $T(C[\natural(C[\bot^\ell])]) \sqsubseteq T(\natural(C[s^\ell])))$. (2) $\Longrightarrow$ (3): We show the contraposition. It suffices to show that $T(\natural(C[\bot^\ell])) \sqsupseteq T(\natural(C[s^\ell]))$ (since $T(\natural(C[\bot^\ell])) \sqsubseteq T(\natural(C[s^\ell]))$ holds). Namely, we show that, for every finite tree V, if $\natural(C[s^\ell]) \longrightarrow^* \sqsupseteq V$, then $\natural(C[\bot^\ell]) \longrightarrow^* \sqsupseteq V$. Assume that $\natural(C[s^\ell]) \longrightarrow^* \sqsupseteq V$. By Proposition 51(2), there is V' such that $C[s^\ell] \longrightarrow^* \sqsupseteq V'$ and $\natural(V') = V$. Note that V' is not tracked by the assumption. Therefore,

$$\begin{aligned}
\natural C[\bot^\ell] = \natural(\flat(C[s^\ell])) &\longrightarrow^* \sqsupseteq \natural(\flat(V')) && \text{(Prop. 51(1) and 56(1))}\\
&= \natural(V') && \text{(Prop. 57)}\\
&= V
\end{aligned}$$
◀

G Proof of Proposition 10

▶ **Proposition** (restatement of Prop. 10)**.** *Let t be a closed and ground-typed term. If t is minimal, then for every non-$\bot$, closed and ground-typed subterm $s \preceq t$, its value tree $T(s)$ is a subtree of $T(t)$.*

Proof. Let C be a linear-context such that $t = C[s]$. Since t is minimal, there is a tracked finite tree V such that $C[\ell(s)] \longrightarrow^* \sqsupseteq V$ (Theorem 52). Let $C[\ell(s)] = t_1 \longrightarrow t_2 \longrightarrow \ldots \longrightarrow t_n \sqsupseteq V$. Then let i be the maximum number such that, for every subterm of t_i of the form $\ell(u)$, $u = s$ holds; and let D be a linear-context such that $t_i \sqsupseteq D[\ell(s)]$ and $D[\bot]$ is a $\Sigma^\bot$-tree term (such i and D always exist by the existence of V). If we assume $s \longrightarrow^* \sqsupseteq W$, then $D[s] \longrightarrow^* \sqsupseteq D[W]$ (by that $D[\perp]$ is a $\Sigma^\perp$-tree), so $C[s] \longrightarrow^* \sqsupseteq D[W]$ holds (by Proposition 48). Therefore $T(s) \preceq T(t)$. ◀

H Proof of Theorem 19

In this section, we prove the soundness and the completeness of the intersection type section system (Section 5) via the alternative definition of the minimality (Appendix F). Let us recall that ℓ is a special tree constructor (disjoint with Σ) of arity 1, called *label*, and $\Sigma^\ell \triangleq \Sigma \uplus \{\ell\}$.

In the following proof, we introduce an alternative intersection type system as follows, where we redefine $(\Theta, x : \theta)$ as $\Theta \uplus \{x \mapsto \theta\}$ if $\theta \neq \top$, and Θ if $\theta = \top$. In a nutshell, the system is "the intersection type system (in Section 5)" + (ℓ) − $(\top)$. Also, (Abs1) and (Abs2) are put together as the rule (Abs) thanks to the redefinition of "$(\Theta, x : \theta)$".

$$\frac{}{x : \wedge\{\tau\} \vdash x^\kappa : \tau}\,(\mathrm{Var}) \qquad \frac{\Theta, \bar{x} : \theta \vdash t : \tau}{\Theta \vdash \lambda \bar{x}.t : \theta \to \tau}\,(\mathrm{Abs})$$

$$\frac{\Theta \vdash t : \theta \to \tau \quad \Delta \vdash s : \theta}{\Theta \wedge \Delta \vdash t\, s : \tau}\,(\mathrm{App}) \qquad \frac{\Theta \vdash t_1\,(\mathbf{Y} t_2) : \tau}{\Theta \vdash \mathbf{Y}(t_1 \sqcup t_2) : \tau}\,(\mathbf{Y}1) \qquad \frac{\Theta \vdash t\,\perp : \tau}{\Theta \vdash \mathbf{Y} t : \tau}\,(\mathbf{Y}2)$$

$$\frac{\Theta_1 \vdash t_1 : \theta_1 \quad \dots \quad \Theta_n \vdash t_n : \theta_n}{\bigwedge_{i \in [n]} \Theta_i \vdash a(t_1, \dots, t_n) : \mathtt{o}}\,(a) \qquad \frac{\Theta_1 \vdash t_1 : \tau_1 \quad \dots \quad \Theta_n \vdash t_n : \tau_n}{\bigwedge_{i \in [n]} \Theta_i \vdash \bigsqcup_{i \in [n]} t_i : \bigwedge_{i \in [n]} \tau_i}\,(\wedge) \qquad \frac{\Theta \vdash t : \mathtt{o}}{\Theta \vdash \ell(t) : \mathtt{o}}\,(\ell)$$

Figure 3 An alternative intersection type system.

▶ **Proposition 58.** *Suppose that t is a term over Σ. Then, $\Theta \vdash t : \bar{\theta}$ (in the intersection type system of Fig. 2) if and only if $\Theta^{\setminus\top} \vdash t : \bar{\theta}$ (in the intersection type system of Fig. 3), where $\Theta^{\setminus\top} \triangleq \{x \mapsto \Theta(x) \mid x \in \mathrm{dom}(\Theta), \Theta(x) \neq \top\}$. In particular, $\emptyset \vdash t : \bar{\theta}$ (in the intersection type system of Fig. 2) if and only if $\emptyset \vdash t : \bar{\theta}$ (in the intersection type system of Fig. 3).*

Proof. $(\Leftarrow)$: This part is trivial since ℓ does not occur in t. $(\Rightarrow)$: This part is also easy, because from a given derivation tree, we can construct a derivation tree such that $\Theta = \Theta^{\setminus\top}$ for each environment Θ. ◀

For simplicity, we will use this alternative intersection type system to prove Theorem 19.

H.1 Properties of the Intersection Type System

In this subsection we list some properties of the intersection type system (and some propositions to show them).

▶ **Proposition 59.** *If $\Theta \vdash t : \bar{\theta}$, then $\mathbf{FV}(t) = \mathrm{dom}(\Theta)$.*

Proof. By a straight-forward induction on the derivation tree. ◀

▶ **Proposition 60.** *The following rule $(\wedge')$ is admissible:* $\dfrac{\Theta_1 \vdash t_1 : \bar{\theta}_1 \qquad \dots \qquad \Theta_n \vdash t_n : \bar{\theta}_n}{\bigwedge_{i \in [n]} \Theta_i \vdash \bigsqcup_{i \in [n]} t_i : \bigwedge_{i \in [n]} \bar{\theta}_i}\,(\wedge')$.

Proof. Assume that $\Theta_i \vdash t_i : \bar{\theta}_i$ for each $i \in [n]$. If $\bar{\theta}_i$ is not prime, then the derivation tree of $\Theta_i \vdash t_i : \theta_i$ is of the following form (on the left-hand side). If $\bar{\theta}_i$ is prime, then let $m_i = 1$, $\Theta_i^1 = \Theta_i$, $t_i^1 = t_i$, and $\bar{\theta}_i^1 = \bar{\theta}_i$. Then $\bigwedge_{i \in [n]} \Theta_i \vdash \bigsqcup_{i \in [n]} t_i : \bigwedge_{i \in [n]} \theta_i$ is shown by the following derivation tree (on the right-hand side).

$$\dfrac{\dfrac{\dfrac{\Theta_i^1 \vdash t_i^1 : \tau_i^1 \quad \dots \quad \Theta_i^{m_i} \vdash t_i^{m_i} : \tau_i^{m_i}}{\bigwedge_{j\in[m_i]} \Theta_i^j \vdash \bigsqcup_{j\in[m_i]} t_i^j : \bigwedge_{j\in[m_i]} \theta_i^j}\ (\wedge)}{\Theta_i \vdash t_i : \theta_i}}{}$$

$$\dfrac{\dfrac{\Theta_1^1 \vdash t_1^1 : \tau_1^1 \quad \Theta_1^2 \vdash t_1^2 : \tau_1^2 \quad \dots \quad \Theta_n^{m_n} \vdash t_n^{m_n} : \tau_n^{m_n}}{\bigwedge_{i\in[n]} \bigwedge_{j\in[m_i]} \Theta_i^j \vdash \bigsqcup_{i\in[n]} \bigsqcup_{j\in[m_i]} t_i^j : \bigwedge_{i\in[n]} \bigwedge_{j\in[m_i]} \theta_i^j}\ (\wedge)}{\bigwedge_{i\in[n]} \Theta_i \vdash \bigsqcup_{i\in[n]} t_i : \bigwedge_{i\in[n]} \theta_i}$$

◀

▶ **Proposition 61.**

(1) *If* $\Theta \vdash t : \top$, *then* $t = \bot$ *and* $\Theta = \emptyset$.
(2) *If* $\Theta \vdash \bot : \bar{\theta}$, *then* $\bar{\theta} = \top$ *and* $\Theta = \emptyset$.

Proof. In these case, the last derivation step should be $(\wedge)$ and also $n = 0$ should. ◀

▶ **Proposition 62** (substitution). *Assume that* $\Theta, \mathtt{x} : \bar{\delta} \vdash t : \bar{\theta}$, $\Delta \vdash s : \bar{\delta}$, *and* $\mathbf{FVC}_{\mathtt{x}}(t) \leq 1$. *Then* $\Theta \wedge \Delta \vdash t\{s/\mathtt{x}\} : \bar{\theta}$.

Proof. By induction on the derivation tree of $\Theta, \mathtt{x} : \bar{\delta} \vdash t : \bar{\theta}$. If $\bar{\delta} = \top$, then $\mathtt{x} \notin \mathbf{FV}(t)$ by $x \notin \mathrm{dom}\big((\Theta, \mathtt{x} : \bar{\delta})\big)$ (Proposition 59); and $\Delta = \emptyset$ by Proposition 61. Therefore $\Theta \wedge \Delta \vdash t\{s/\mathtt{x}\} : \bar{\theta}$ is immediate from $\Theta, \mathtt{x} : \bar{\delta} \vdash t : \bar{\theta}$. We let $\langle n, \{\Delta_i\}_{i\in[n]}, \{s_i\}_{i\in[n]}, \{\sigma_i\}_{i\in[n]}\rangle$ be such that, if $\bar{\delta}$ is not prime (note that the last derivation step of $\Delta \vdash s : \bar{\delta}$ is $(\wedge)$), $\Delta = \bigwedge_{i\in[n]} \Delta_i$, $\bar{\delta} = \bigwedge_{i\in[n]} \sigma_i$, $s = \bigsqcup_{i\in[n]} s_i$, and for every $i \in [n]$, $\Delta_i \vdash s_i : \sigma_i$; and if $\bar{\delta}$ is prime, $\langle 1, \{\Delta\}, \{s\}, \{\bar{\delta}\}\rangle$. Then we do case analysis on the last derivation step.

Case (Var): By $\mathtt{x} \in \mathrm{dom}\big((\Theta, \mathtt{x} : \bar{\delta})\big)$ (since $\bar{\delta} \neq \top$), t should be $\mathtt{x}$. Also $\Theta = \emptyset$ and $\bar{\theta} = \bar{\delta}$ should hold. Therefore $\Theta \wedge \Delta \vdash t\{s/\mathtt{x}\} : \bar{\theta}$ is immediate from $\Delta \vdash s : \bar{\delta}$.

Case (Abs): Then t is of the form $\lambda x.t_1$. Without loss of generality, we can assume that $x \notin \mathbf{FV}(s)$ by using α-equivalence. The derivation tree is of the following form:

$$\dfrac{\dfrac{\Theta, \mathtt{x} : \bar{\delta}, x : \delta \vdash t_1 : \tau}{\Theta, \mathtt{x} : \bar{\delta} \vdash \lambda x.t_1 : \delta \to \tau}\ (\mathrm{Abs})}{\Theta, \mathtt{x} : \bar{\delta} \vdash \lambda x.t_1 : \bar{\theta}}$$

By I.H., $\Theta \wedge \Delta, \mathtt{x} : \theta, x : \delta \vdash t_1\{s/\mathtt{x}\} : \tau$. Therefore,

$$\dfrac{\dfrac{\Theta \wedge \Delta, \mathtt{x} : \bar{\delta}, x : \delta \vdash t_1\{s/\mathtt{x}\} : \tau}{\Theta \wedge \Delta, \mathtt{x} : \bar{\delta} \vdash \lambda x.t_1\{s/\mathtt{x}\} : \delta \to \tau}\ (\mathrm{Abs})}{\Theta \wedge \Delta, \mathtt{x} : \bar{\delta} \vdash (\lambda x.t_1)\{s/\mathtt{x}\} : \bar{\theta}}$$

Case ($\mathbf{Y}$1): Then t is of the form $\mathbf{Y}t_0$. The derivation tree is of the following form.

$$\dfrac{\dfrac{\dfrac{\Theta_1, \mathtt{x} : \bar{\delta}_1 \vdash t_1 : \delta \to \bar{\theta} \qquad \Theta_2, \mathtt{x} : \bar{\delta}_2 \vdash \mathbf{Y}t_2 : \delta}{\Theta_1 \wedge \Theta_2, \mathtt{x} : \bar{\delta}_1 \wedge \bar{\delta}_2 \vdash t_1(\mathbf{Y}t_2) : \bar{\theta}}\ (\mathrm{App})}{\Theta_1 \wedge \Theta_2, \mathtt{x} : \bar{\delta}_1 \wedge \bar{\delta}_2 \vdash \mathbf{Y}(t_1 \sqcup t_2) : \bar{\theta}}\ (\mathbf{Y}1)}{\Theta, \mathtt{x} : \bar{\delta} \vdash \mathbf{Y}t_0 : \bar{\theta}}$$

For each $l \in [2]$, let S_l be a subset of $[n]$ such that $\bar{\delta}_l = \bigwedge_{i\in S_l} \sigma_i$ if $\bar{\delta}$ is not prime; and $\bar{\delta}_l = \bar{\delta}$ otherwise. Then $\bigwedge_{i\in S_l} \Delta_i \vdash \bigsqcup_{i\in S_l} s_i : \bar{\delta}_l$ (by using $(\wedge)$ if $\bar{\delta}$ is not prime). By I.H.,

$\Theta_1 \wedge \bigwedge_{i\in S_1} \Delta_i \vdash t_1\{\bigsqcup_{i\in S_1} s_i\} : \delta \to \bar{\theta}$. Also by I.H., $\Theta_2 \wedge \bigwedge_{i\in S_2} \Delta_i \vdash (\mathbf{Y}t_2)\{\bigsqcup_{i\in S_2} s_i\} : \delta$. (Note that $\mathbf{FVC}_{\mathtt{x}}(t_1) \leq 1$ and $\mathbf{FVC}_{\mathtt{x}}(\mathbf{Y}t_2) \leq 1$.) Therefore,

$$\dfrac{\dfrac{\Theta_1 \wedge \bigwedge_{i\in S_1} \Delta_i \vdash t_1\{\bigsqcup_{i\in S_1} s_i/\mathtt{x}\} : \delta \to \bar{\theta} \qquad \Theta_2 \wedge \bigwedge_{i\in S_2} \Delta_i \vdash (\mathbf{Y}t_2)\{\bigsqcup_{i\in S_2} s_i/\mathtt{x}\} : \delta}{\Theta_1 \wedge \Theta_2 \wedge \bigwedge_{i\in S_1\cup S_2} \Delta_i \vdash t_1\{\bigsqcup_{i\in S_1} s_i/\mathtt{x}\}\,(\mathbf{Y}t_2)\{\bigsqcup_{i\in S_2} s_i/\mathtt{x}\} : \bar{\theta}}\,(\mathrm{App})}{\dfrac{\Theta \wedge \Delta \vdash \mathbf{Y}(t_1\{\bigsqcup_{i\in S_1} s_i/\mathtt{x}\} \sqcup t_2\{\bigsqcup_{i\in S_2} s_i/\mathtt{x}\}) : \bar{\theta}}{\dfrac{\Theta \wedge \Delta \vdash \mathbf{Y}((t_1 \sqcup t_2)\{(\bigsqcup_{i\in S_1} s_i) \sqcup (\bigsqcup_{i\in S_2} s_i)/\mathtt{x}\}) : \bar{\theta}}{\Theta \wedge \Delta \vdash (\mathbf{Y}t_0)\{s/\mathtt{x}\} : \bar{\theta}}}\,\text{Prop. 45}}\,(\mathbf{Y}1)$$

Case (App)($\mathbf{Y}2$)(a)(ℓ)($\wedge$): In the same way as ($\mathbf{Y}1$). ◀

▶ **Proposition 63** (inverse substitution). *Assume that $\Theta^0 \vdash t\{s/\mathtt{x}\} : \bar{\theta}$ and $\mathbf{FVC}_{\mathtt{x}}(t) \leq 1$. Also assume that if $\mathbf{FVC}_{\mathtt{x}}(t) = 0$, then $s = \bot$. Then there is $\langle \Theta, \Delta, \bar{\delta}\rangle$ such that (a) $\Theta^0 = \Theta \wedge \Delta$, (b) $\Theta, \mathtt{x} : \bar{\delta} \vdash t : \theta$, and (c) $\Delta \vdash s : \bar{\delta}$.*

Proof. By induction on the derivation tree of $\Theta^0 \vdash t\{s/\mathtt{x}\} : \bar{\theta}$. If $\bar{\theta}$ is not prime, then the derivation tree is of the following form (using Proposition 45), where $t = \bigsqcup_{i\in[n]} t_i$, $s = \bigsqcup_{i\in[n]} s_i$, and for each $i \in [n]$, if $\mathtt{x} \notin \mathbf{FV}(t_i)$, then $s_i = \bot$.

$$\dfrac{\dfrac{\Theta^0_1 \vdash t_1\{s_1/\mathtt{x}\} : \tau_1 \qquad \ldots \qquad \Theta^0_n \vdash t_n\{s_n/\mathtt{x}\} : \tau_n}{\bigwedge_{i=1}^n \Theta^0_i \vdash \bigsqcup_{i\in[n]} t_i\{s_i/\mathtt{x}\} : \bigwedge_{i=1}^n \tau_i}\,(\wedge)}{\Theta^0 \vdash t\{s/\mathtt{x}\} : \bar{\theta}}$$

For each $i \in [n]$, let $\langle \Theta_i, \Delta_i, \bar{\delta}_i\rangle$ be a tuple obtained by I.H. for $\Theta^0_n \vdash t_n\{s_n/\mathtt{x}\} : \tau_n$. Then $\langle \bigwedge_{i\in[n]} \Theta_i, \bigwedge_{i\in[n]} \Delta_i, \bigwedge_{i\in[n]} \delta_i\rangle$ satisfies (a)(b)(c). (b) and (c) are shown by using the admissible rule ($\wedge'$). Otherwise we do case analysis on the structure of t.

Case $t = \mathtt{x}$: Then $\langle \Theta, \Delta, \bar{\delta}\rangle = \langle \emptyset, \Theta^0, \bar{\theta}\rangle$ satisfies (a)(b)(c). (a) is trivial. (b) is directly derived by the rule (Var). (c) is shown by $t\{s/\mathtt{x}\} = s$.

Case $t = \bot$ or $t = x$ (where $x \neq \mathtt{x}$): Then $\langle \Theta, \Delta, \bar{\delta}\rangle = \langle \Theta^0, \emptyset, \top\rangle$ satisfies (a)(b)(c) (note that $s = \bot$ by $\mathbf{FVC}_x(t) = 0$).

Case $t = \lambda x.t_1$: Without loss of generality, we can assume that $x \notin \mathbf{FV}(s)$ by using α-equivalence. Then the derivation tree is of the following form:

$$\dfrac{\dfrac{\Theta^0, x : \delta \vdash t_1\{s/\mathtt{x}\} : \tau}{\Theta^0 \vdash \lambda x.t_1\{s/\mathtt{x}\} : \delta \to \tau}\,(\mathrm{Abs})}{\Theta^0 \vdash t\{s/\mathtt{x}\} : \bar{\theta}}$$

Let $\langle \Theta_1, \Delta_1, \bar{\delta}_1\rangle$ be a tuple obtained by I.H.. Then by $x \notin \mathbf{FV}(s)$ and Proposition 59, $x \notin \mathrm{dom}(\Delta_1)$, and thus $\Theta_1(x) = \delta$. Let Θ'_1 be such that $\Theta_1 = \Theta'_1, x : \delta$. Then $\langle \Theta, \Delta, \bar{\delta}\rangle = \langle \Theta'_1, \Delta_1, \bar{\delta}_1\rangle$ satisfies (a)(b)(c). (a) and (c) are trivial. (b) is derived from $\Theta'_1, x : \delta, \mathtt{x} : \bar{\delta}_1 \vdash t_1 : \tau$ by applying (Abs).

Case $t = \mathbf{Y}t_0$ and the last derivation step is ($\mathbf{Y}1$): Then the derivation tree is of the following form (using Proposition 45), where $t_0 = t_1 \sqcup t_2$, $s = s_1 \sqcup s_2$, and for each $l \in [2]$, if

$\mathtt{x} \notin \mathbf{FV}(t_l)$, then $s_l = \bot$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Theta_1^0 \vdash t_1\{s_1/\mathtt{x}\} : \delta \to \bar{\theta} \qquad \Theta_2^0 \vdash \mathbf{Y} t_2\{s_2/\mathtt{x}\} : \delta}{\Theta_1^0 \wedge \Theta_2^0 \vdash t_1\{s_1/\mathtt{x}\}(\mathbf{Y} t_2\{s_2/\mathtt{x}\}) : \bar{\theta}}\,(\mathrm{App})}{\Theta_1^0 \wedge \Theta_2^0 \vdash \mathbf{Y}(t_1\{s_1/\mathtt{x}\} \sqcup t_2\{s_2/\mathtt{x}\}) : \bar{\theta}}\,(\mathbf{Y}1)}{\Theta_1^0 \wedge \Theta_2^0 \vdash \mathbf{Y} t_0\{s/\mathtt{x}\} : \bar{\theta}}}{\Theta^0 \vdash t\{s/\mathtt{x}\} : \bar{\theta}}$$

Let $\langle \Theta_1, \Delta_1, \bar{\delta}_1 \rangle$ be a tuple obtained by I.H. for $\Theta_1^0 \vdash t_1\{s_1/\mathtt{x}\} : \delta \to \bar{\theta}$. Also let $\langle \Theta_2, \Delta_2, \bar{\delta}_2 \rangle$ be a tuple obtained by I.H. for $\Theta_2^0 \vdash \mathbf{Y} t_2\{s_2/\mathtt{x}\} : \delta$. Then $\langle \Theta, \Delta, \delta \rangle = \langle \Theta_1 \wedge \Theta_2, \Delta_1 \wedge \Delta_2, \bar{\delta}_1 \wedge \bar{\delta}_2 \rangle$ satisfies (a)(b)(c). (a) is trivial. (b) is derived from $\Theta_1, \mathtt{x} : \bar{\delta}_1 \vdash t_1 : \delta \to \tau$ and $\Theta_2, \mathtt{x} : \bar{\delta}_2 \vdash \mathbf{Y} t_2 : \delta$ by applying (App) and then applying ($\mathbf{Y}1$). (c) is shown by using the admissible rule ($\wedge'$).

Case $t = t_1\, t_2$, $t = a(t_1, \ldots, t_{\Sigma(a)})$, $t = \ell(t_1)$, or ($t = \mathbf{Y} t_0$ and the last derivation step is ($\mathbf{Y}2$)): In the same way as the above case. ◀

▶ **Proposition 64** (subject reduction). *Assume that* $\Theta \vdash t : \bar{\theta}$.

(1) *If* $t \longrightarrow t'$, *then there is* $s' \sqsubseteq t'$ *such that (a)* $\Theta \vdash s' : \bar{\theta}$ *and (b) if* t *is labelled, then so is* s'.

(2) *If* $t \longrightarrow^* t'$, *then there is* $s' \sqsubseteq t'$ *such that (a)* $\Theta \vdash s' : \bar{\theta}$ *and (b) if* t *is labelled, then so is* s'.

Proof. (1): By induction on $\langle |t|, |\bar{\theta}| \rangle$. If $\bar{\theta}$ is not prime (note that the last derivation step of $\Theta \vdash t : \bar{\theta}$ is ($\wedge$)), then let $\langle \{\Theta_i\}_{i \in [n]}, \{t_i\}_{i \in [n]}, \{\tau_i\}_{i \in [n]} \rangle$ be such that, for each $i \in [n]$, $\Theta_i \vdash t_i : \tau_i$, $\Theta = \bigwedge_{i \in [n]} \Theta_i$, $t = \bigsqcup_{i \in [n]} t_i$, and $\bar{\theta} = \bigwedge_{i \in [n]} \tau_i$. By $t_i \sqsubseteq t$ and $t \longrightarrow t'$ (Proposition 48), there is $t'_i \sqsubseteq t'$ such that $t_i \longrightarrow t'_i$. Then by I.H., there is $s'_i \sqsubseteq t'_i$ such that $\Theta_i \vdash s'_i : \tau_i$. $\Theta \vdash s' : \bar{\theta}$ has been proved by letting $s' = \bigsqcup_{i \in [n]} s'_i$. Otherwise we do case analysis on the last derivation step of $t \longrightarrow t'$.

Case (β): Then $t \longrightarrow t'$ is of the form $(\lambda x.t_0\{x/\mathtt{x}_1\} \ldots \{x/\mathtt{x}_m\})u \longrightarrow t_0\{u/\mathtt{x}_1\} \ldots \{u/\mathtt{x}_m\}$, where $\mathtt{x}_1, \ldots, \mathtt{x}_m$ are all distinct, $\mathtt{x}_1, \ldots, \mathtt{x}_m \notin \mathbf{FV}(x) \cup \mathbf{FV}(t) \cup \mathbf{FV}(u)$, and each of $\mathtt{x}_1, \ldots, \mathtt{x}_m$ occurs in t just once. Also the derivation tree of $\Theta \vdash t : \bar{\theta}$ is of the following form.

$$\dfrac{\dfrac{\dfrac{\Theta_1, x : \bigwedge_{i \in [n]} \sigma_i \vdash t_0\{x/\mathtt{x}_1\} \ldots \{x/\mathtt{x}_m\} : \tau}{\Theta_1 \vdash \lambda x.t_0\{x/\mathtt{x}_1\} \ldots \{x/\mathtt{x}_m\} : \bigwedge_{i=1}^n \sigma_i \to \tau}\,(\mathrm{Abs}) \qquad \dfrac{\Delta_1 \vdash u_1 : \sigma_1 \quad \ldots \quad \Delta_n \vdash u_n : \sigma_n}{\bigwedge_{i=1}^n \Delta_i \vdash \bigsqcup_{i=1}^n u_i : \bigwedge_{i=1}^n \sigma_i}\,(\wedge)}{\Theta_1 \wedge \bigwedge_{i=1}^n \Delta_i \vdash (\lambda x.t_0\{x/\mathtt{x}_1\} \ldots \{x/\mathtt{x}_m\})\, u : \tau}\,(\mathrm{App})}{\Theta \vdash t : \tau}$$

By applying inverse substitution lemma (Proposition 63) to $\Theta_1, x : \bigwedge_{i \in [n]} \sigma_i \vdash t_0\{x/\mathtt{x}_1\} \ldots \{x/\mathtt{x}_m\} : \tau$ iteratively, there is $\langle \bar{\delta}_1, \ldots \bar{\delta}_m \rangle$ such that $\bigwedge_{i \in [n]} \sigma_i = \bigwedge_{j \in [m]} \bar{\delta}_j$ and $\Theta_1, \mathtt{x}_1 : \bar{\delta}_1, \ldots, \mathtt{x}_m : \bar{\delta}_m \vdash t_0 : \tau$. Also for each $j \in [m]$, there is a subset S_j of $[n]$ such that $\bar{\delta}_j = \bigwedge_{i \in S_j} \sigma_i$. By using ($\wedge$), $\bigwedge_{i \in S_j} \Delta_i \vdash \bigsqcup_{i \in S_j} u_i : \sigma_j$. Then $s' = t_0\{\bigsqcup_{i \in S_1} u_i/\mathtt{x}_1\} \ldots \{\bigsqcup_{i \in S_m} u_i/\mathtt{x}_m\}$ satisfies the conditions: $s' \sqsubseteq t'$ is shown by Proposition 43 and $\Theta \vdash s' : \bar{\theta}$ is shown by applying substitution lemma (Proposition 62) to $\Theta_1, x : \bigwedge_{i \in [n]} \sigma_i \vdash t_0\{x/\mathtt{x}_1\} \ldots \{x/\mathtt{x}_m\} : \tau$ iteratively.

Case ($\mathbf{Y}$): Then $t \longrightarrow t'$ is of the form $\mathbf{Y} t_0 \longrightarrow t_0(\mathbf{Y} t_0)$. From this, the last derivation rule of $\Theta \vdash t : \bar{\theta}$ is ($\mathbf{Y}1$) or ($\mathbf{Y}2$).

Sub-Case $(\mathbf{Y}1)$: The derivation tree is of the following form:

$$\dfrac{\Theta \vdash u_1(\mathbf{Y}u_2) : \bar{\theta}}{\dfrac{\Theta \vdash \mathbf{Y}(u_1 \sqcup u_2) : \bar{\theta}}{\Theta \vdash \mathbf{Y}t_0 : \bar{\theta}}} \;(\mathbf{Y}\ 1)$$

Then $s' = u_1(\mathbf{Y}u_2)$ satisfies the conditions. $s' \sqsubseteq t'$ is derived from $u_1, u_2 \sqsubseteq t_0$ and $\Theta \vdash s' : \bar{\theta}$ is immediately shown by using the above derivation tree.

Sub-Case $(\mathbf{Y}2)$: The derivation tree is of the following form:

$$\dfrac{\Theta \vdash t_0\bot : \bar{\theta}}{\Theta \vdash \mathbf{Y}t_0 : \bar{\theta}} \;(\mathbf{Y}\ 2)$$

Then $s' = t_0\bot$ satisfies the conditions. $s' \sqsubseteq t'$ is derived from $\bot \sqsubseteq \mathbf{Y}t_0$ and $\Theta \vdash s' : \bar{\theta}$ is immediately shown by using the above derivation tree.

Case $(\bot)$: Then $t \longrightarrow t'$ is of the form $\bot t_2 \longrightarrow \bot$. Also the derivation tree of $\Theta \vdash t : \bar{\theta}$ is of the following form:

$$\dfrac{\Theta_1 \vdash \bot : \delta \to \bar{\theta} \qquad \Theta_2 \vdash t_2 : \delta}{\dfrac{\Theta_1 \wedge \Theta_2 \vdash \bot t_2 : \bar{\theta}}{\Theta \vdash t : \bar{\theta}}} \;(\mathrm{App})$$

However it is contradiction, because $\Theta_1 \not\vdash \bot : \delta \to \bar{\theta}$ by Proposition 61.

Case (App): Then $t \longrightarrow t'$ is of the form $t_1\, t_2 \longrightarrow t_1'\, t_2$ and is derived from $t_1 \longrightarrow t_1'$. The derivation tree of $\Theta \vdash t : \bar{\theta}$ is of the following form.

$$\dfrac{\Theta_1 \vdash t_1 : \delta \to \bar{\theta} \qquad \Theta_2 \vdash t_2 : \delta}{\dfrac{\Theta_1 \wedge \Theta_2 \vdash t_1\, t_2 : \bar{\theta}}{\Theta \vdash t : \bar{\theta}}} \;(\mathrm{App})$$

By I.H., there is $s_1' \sqsubseteq t_1'$ such that $\Theta_1 \vdash s_1' : \delta \to \bar{\theta}$. Then $s' = s_1' t_2$ satisfies the conditions.

Case (a) ($a \in \Sigma$ and $a = \ell$): In the same way as case (App).

(2): Let $t_1, \dots, t_n$ be such that $t = t_1 \longrightarrow \dots \longrightarrow t_n = t'$. We prove the following by induction on i $(\star)$: there is a term $s_i \sqsubseteq t_i$ such that $\Theta \vdash s_i : \bar{\theta}$. If $i = 1$, then $s_1 = t_1$ satisfies $(\star)$. Otherwise by I.H., we have $s_{i-1} \sqsubseteq t_{i-1}$ such that $\Theta \vdash s_{i-1} : \bar{\theta}$. By Proposition 48 (since $t_{i-1} \sqsupseteq s_{i-1}$ and $t_{i-1} \longrightarrow t_i$), there is s_i' such that $s_{i-1} \longrightarrow^{\leq 1} s_i'$ and $t_i \sqsupseteq s_i'$. If $s_{i-1} \longrightarrow^0 s_i'$, then $s_i = s_{i-1}$ satisfies the conditions. If $s_{i-1} \longrightarrow^1 s_i'$, then by (1), there is $s_i \sqsubseteq s_i'$ such that $\Theta \vdash s_i : \bar{\theta}$ (and also if s_{i-1} is labelled, then s_i is labelled). Indeed this s_i satisfies $(\star)$. Finally, this lemma has been proved by letting $s' = s_n$. ◀

▶ **Proposition 65** (subject expansion). *Assume that $s' \sqsubseteq t'$ and $\Theta \vdash s' : \bar{\theta}$.*

(1) *If $t \longrightarrow t'$, then there is $s \sqsubseteq t$ such that (a) $s \longrightarrow^{\leq 1} \sqsupseteq s'$ and (b) $\Theta \vdash s : \bar{\theta}$.*
(2) *If $t \longrightarrow^* t'$, then there is $s \sqsubseteq t$ such that (a) $s \longrightarrow^* \sqsupseteq s'$ and (b) $\Theta \vdash s : \bar{\theta}$.*

Proof. By induction on $|t|$. In the later we only consider the case of that $\bar{\theta}$ is not prime. (The case of that $\bar{\theta}$ is prime can be proved in the same way.) Then note that the last derivation step of $\Theta \vdash s' : \bar{\theta}$ is $(\wedge)$). We let $\langle n, \{\Theta_i\}_{i\in[n]}, \{s_i'\}_{i\in[n]}, \{\tau_i\}_{i\in[n]}\rangle$ be such that, $\Theta = \bigwedge_{i\in[n]} \Theta_i$, $\bar{\theta} = \bigwedge_{i\in[n]} \tau_i$, $s' = \bigsqcup_{i\in[n]} s_i'$, and for every $i \in [n]$, $\Theta_i \vdash s_i' : \tau_i$. If $s' = \bot$, then $s = \bot$ satisfies the conditions. Otherwise we do case analysis on the last derivation rule.

Case (β): Then $t \longrightarrow t'$ is of the form $(\lambda x.t^0\{x/\mathtt{x}^1\}\dots\{x/\mathtt{x}^m\})t^1 \longrightarrow t^0\{t^1/\mathtt{x}^1\}\dots\{t^1/\mathtt{x}^m\}$, where $\mathtt{x}^1, \dots, \mathtt{x}^m$ are all distinct, $\mathtt{x}^1, \dots, \mathtt{x}^m \notin \mathbf{FV}(x) \cup \mathbf{FV}(t^0) \cup \mathbf{FV}(t^1)$, and each of

$\mathtt{x}^1, \ldots, \mathtt{x}^m$ occurs in t just once. By $t^0\{t^1/\mathtt{x}^1\}\ldots\{t^1/\mathtt{x}^m\} \sqsupseteq s'$ and applying Proposition 46 iteratively, we have a tuple $\langle s^0, s^1, \ldots, s^m\rangle$ such that $s' = s^0\{s^1/\mathtt{x}^1\}\ldots\{s^m/\mathtt{x}^m\}$, $t^0 \sqsupseteq s^0$, $t^1 \sqsupseteq s^1$, $\ldots$, and $t^1 \sqsupseteq s^m$. By using Proposition 45 iteratively, we have a set $\{\langle s_i^0, s_i^1, \ldots, s_i^m\rangle\}_{i\in[n]}$ such that for each $i \in [n]$, $s'_i = s_i^0\{s_i^1/\mathtt{x}^1\}\ldots\{s_i^m/\mathtt{x}^m\}$; and for each $j \in [0,m]$, $s^j = \bigsqcup_{i\in[n]} s_i^j$. Then for each i, by $\Theta_i \vdash s_i^0\{s_i^1/\mathtt{x}^1\}\ldots\{s_i^m/\mathtt{x}^m\} : \tau_i$ and applying inverse substitution lemma (Proposition 63) iteratively, there is $\langle\{\Theta_i^j\}_{j\in[0,m]}, \{\bar{\delta}_i^j\}_{j\in[m]}\rangle$ such that (i) $\Theta_i = \bigwedge_{j\in[0,m]} \Theta_i^j$, (ii) $\Theta_i^0, \mathtt{x}^1 : \bar{\delta}_i^1, \ldots, \mathtt{x}^m : \bar{\delta}_i^m \vdash s_i^0 : \tau_i$, and (iii) for each $j \in [m]$, $\Theta_i^j \vdash s_i^j : \bar{\delta}_i^j$. Then let $s = (\lambda x.(\bigsqcup_{i\in[n]} s_i^0)\{x/\mathtt{x}^1\}\ldots\{x/\mathtt{x}^m\})\,(\bigsqcup_{i\in[n]}\bigsqcup_{j\in[m]} s_i^j)$. $s \sqsubseteq t$ is shown by $s_i^0 \sqsubseteq t^0$ and $s_i^j \sqsubseteq t^1$ ($j \geq 1$). Indeed this s satisfies (a)(b). (a) is shown by $s \longrightarrow (\bigsqcup_{i\in[n]} s_i^0)\{\bigsqcup_{i\in[n]}\bigsqcup_{j\in[m]} s_i^j/\mathtt{x}^1\}\ldots\{\bigsqcup_{i\in[n]}\bigsqcup_{j\in[m]} s_i^j/\mathtt{x}^m\} \sqsupseteq s^0\{s^1/\mathtt{x}_1\}\ldots\{s^m/\mathtt{x}^m\} = s'$. Also (b) is derived from $\bigwedge_{j\in[0,m]} \Theta_i^j \vdash (\lambda x.s^0\{x/\mathtt{x}^1\}\ldots\{x/\mathtt{x}^m\})\,(\bigsqcup_{j\in[m]} s_i^j) : \tau_i$ (for $i = 1, \ldots, n$) by applying ($\wedge$). Each of them is shown by the following derivation tree, where (ii') is shown by (ii) and applying substitution lemma (Proposition 62) iteratively.

$$\dfrac{\dfrac{\overset{\text{(ii')}}{\Theta_i^0, x : \bigwedge_{j\in[m]} \bar{\delta}_i^j \vdash s_i^0\{x/\mathtt{x}^1\}\ldots\{x/\mathtt{x}^m\} : \tau_i}}{\Theta_i^0 \vdash \lambda x.s_i^0\{x/\mathtt{x}^1\}\ldots\{x/\mathtt{x}^m\} : \bigwedge_{j\in[m]} \bar{\delta}_i^j \to \tau_i}\,(\mathrm{Abs}) \qquad \dfrac{\overset{\text{(iii)}}{\Theta_i^1 \vdash s_i^1 : \bar{\delta}_i^1} \quad \ldots \quad \overset{\text{(iii)}}{\Theta_i^m \vdash s_i^m : \bar{\delta}_i^m}}{\bigwedge_{j\in[m]} \Theta_i^j \vdash \bigsqcup_{j\in[m]} s_i^j : \bigwedge_{j\in[m]} \bar{\delta}_i^j}\,(\wedge')}{\bigwedge_{j\in[0,m]} \Theta_i^j \vdash (\lambda x.s_i^0\{x/\mathtt{x}^1\}\ldots\{x/\mathtt{x}^m\})\,(\bigsqcup_{j\in[m]} s_i^j) : \tau_i}\,(\mathrm{App})$$

Case ($\mathbf{Y}$): Then $t \longrightarrow t'$ is of the form $\mathbf{Y}t^0 \longrightarrow t^0(\mathbf{Y}t^0)$. For each $i \in [n]$, by $t' \sqsupseteq s'_i \neq \bot$, s'_i is of the form $s_i^1 s_i^0$. s_i^0 is one of the forms (i) $\bot$ or (ii) $\mathbf{Y}s_i^2$ (let $s_i^2 = \bot$ in (i) for convenience). Then let $s = \mathbf{Y}(\bigsqcup_{\langle i,l\rangle\in[n]\times[2]} s_i^l)$. $s \sqsubseteq t$ is shown by $s_i^l \sqsubseteq t_0$. (a) is shown by $s \longrightarrow (\bigsqcup_{\langle i,l\rangle\in[n]\times[2]} s_i^l)\,(\mathbf{Y}(\bigsqcup_{\langle i,l\rangle\in[n]\times[2]} s_i^l)) \sqsupseteq (\bigsqcup_{i\in[n]} s_i^1)\,(\mathbf{Y}(\bigsqcup_{i\in[n]} s_i^2)) = \bigsqcup_{i\in[n]} s'_i = s'$. Also for (b), it suffices to show that, for each $i \in [n]$, $\Theta_i \vdash \mathbf{Y}(s_i^1 \sqcup s_i^2) : \tau_i$. It is shown by the following derivation trees, where the left-hand side is for (i) ($s_i^0 = \bot$); and the right-hand side is for (ii) ($s_i^0 = \mathbf{Y}s_i^2$).

$$\dfrac{\dfrac{\Theta_i \vdash s'_i : \tau_i}{\Theta_i \vdash (s_i^1 \sqcup s_i^2)\,\bot : \tau_i}\; s'_i = s_i^1\bot,\ s_i^2 = \bot}{\Theta_i \vdash \mathbf{Y}(s_i^1 \sqcup s_i^2) : \tau_i}\,(\mathbf{Y}2) \qquad \dfrac{\dfrac{\Theta_i \vdash s'_i : \tau_i}{\Theta_i \vdash s_i^1\,(\mathbf{Y}s_i^2) : \tau_i}\; s'_i = s_i^1(\mathbf{Y}s_i^2)}{\Theta_i \vdash \mathbf{Y}(s_i^1 \sqcup s_i^2) : \tau_i}\,(\mathbf{Y}1)$$

Case ($\bot$): Then $t \longrightarrow t'$ is of the form $\bot t_2 \longrightarrow \bot$, but it is contradiction because $t' \sqsupseteq s' \neq \bot$.

Case (App): Then $t \longrightarrow t'$ is of the form $t^0t^2 \longrightarrow t^1t^2$ and is derived from $t^0 \longrightarrow t^1$. For each $i \in [n]$, by $t' \sqsupseteq s'_i \neq \bot$, s'_i is of the form $s_i^1 s_i^2$. Then the derivation tree of $\Theta_i \vdash s_i^1 s_i^2 : \tau_i$ is of the following form:

$$\dfrac{\dfrac{\Theta_i^1 \vdash s_i^1 : \delta_i \to \tau_i \qquad \Theta_i^2 \vdash s_i^2 : \delta_i}{\Theta_i^1 \wedge \Theta_i^2 \vdash s_i^1 s_i^2 : \tau_i}\,(\mathrm{App})}{\Theta_i \vdash s_i^1 s_i^2 : \tau_i}$$

Let $s^1 = \bigsqcup_{i\in[n]} s_i^1$, let $\Theta^1 = \bigwedge_{i\in[n]} \Theta_i^1$, and let $\bar{\theta}' = \bigwedge_{i\in[n]}(\delta_i \to \tau_i)$. Then $\Theta^1 \vdash s^1 : \bar{\theta}'$ is derived from $\Theta_i^1 \vdash s_i^1 : \delta_i \to \tau_i$ ($i = 1, \ldots, n$) by applying ($\wedge$). By I.H., there is $s^0 \sqsubseteq t^0$ such that $s^0 \longrightarrow^{\leq 1}\ \sqsupseteq s^1$ and $\Theta^1 \vdash s^0 : \bar{\theta}'$. Let m and $\{\langle {\Theta'}_i^1, s_i^0, \tau'_i\rangle\}_{i\in[m]}$ be such that $\Theta^1 = \bigwedge_{i\in[m]} {\Theta'}_i^1$, $\bar{\theta}' = \bigwedge_{i\in[m]} \tau'_i$, $s^0 = \bigsqcup_{i\in[m]} s_i^0$, and for every $i \in [m]$, ${\Theta'}_i^1 \vdash s_i^0 : \tau'_i$. Note that for every $i \in [m]$, there is $j \in [n]$ such that $\tau'_i = \delta_j \to \tau_j$, and vice versa. Then let $s = (\bigsqcup_{i\in[m]} s_i^0)\,(\bigsqcup_{j\in[n]} s_j^2)$. $s \sqsubseteq t$ is shown by $s_i^0 \sqsubseteq t^0$ and $s_j^2 \sqsubseteq t^2$. (a) is shown by using $s^0 \longrightarrow^{\leq 1}\ \sqsupseteq s^1$. (b) is derived from ${\Theta'}_i^1 \wedge \Theta_j^2 \vdash s_i^0\, s_j^2 : \tau_j$ by applying ($\wedge$), where $\langle i,j\rangle$ is all pairs such that $\tau'_i = \delta_j \to \tau_j$. Each ${\Theta'}^1_i \wedge \Theta^2_j \vdash s^0_i\, s^2_j : \tau_j$ is derived from ${\Theta'}^1_i \vdash \delta_j \to \tau_j$ and $\Theta^2_j \vdash s^2_j : \tau_j$ by applying (App).

Case (a) ($a \in \Sigma$ and $a = \ell$): In the same way as case (App).

(2): Let $t_1, \dots, t_n$ be s.t. $t = t_1 \longrightarrow \dots \longrightarrow t_n \sqsupseteq s'$ and let $s_n = s'$. By using (1) iteratively, there exist $s_{n-1}, \dots, s_1$ s.t. $t_i \sqsupseteq s_i$, $s_i \longrightarrow^{\leq 1}\ \sqsupseteq s_{i+1}$, and $\Theta \vdash s_i : \bar{\theta}$ for each $i \in [n-1]$. Then $s = s_1$ satisfies the conditions. $t \sqsupseteq s$ and $\vdash s : \bar{\theta}$ are obvious from the above. Also $s \longrightarrow^* \sqsupseteq s'$ is shown by $s\ (\longrightarrow^{\leq 1}\ \sqsupseteq)^*\ s'$ and $(\sqsupseteq\longrightarrow) \subseteq (\longrightarrow^{\leq 1}\ \sqsupseteq)$ (Proposition 48). ◀

H.2 Proof of the Completeness

▶ **Proposition 66.** *Let V be any finite $\Sigma^{\perp}$-tree. Then $\emptyset \vdash V : \bar{\theta}$ for some $\bar{\theta}$.*

Proof. By simple induction on the structure of V. ◀

▶ **Theorem 67** (completeness)**.** *Let t be any closed and ground-typed term over Σ. If t is minimal, then $\emptyset \vdash t : \bar{\theta}$ for some $\bar{\theta}$.*

Proof. Since t is minimal, by Theorem 52, for each $\langle C, s\rangle$ such that $t = C[s]$, s is a ground-typed term, and $s \neq \perp$, let $\langle D_C, u_C\rangle$ (note s is uniquely determined by C) be such that $D_C[\ell(u_C)]$ is a tracked finite tree and $C[s^{\ell}] \longrightarrow^* \sqsupseteq D_C[\ell(u_C)] \dots (\star 1)$. We can assume that ℓ does not occur in D_C. Also let $V = \bigsqcup_C D_C[\natural u_C]$ (where C ranges over linear contexts such that $t = C[s]$ holds for some $s \neq \perp$). (Note that V is defined by $T(t) = T(\natural C[s^{\ell}]) \sqsupseteq \natural D_C[\ell(u_C)]$.) By Proposition 66, $\emptyset \vdash V : \bar{\theta}$ for some $\bar{\theta}$. Then by subject expansion lemma (Proposition 65), there exists $t' \sqsubseteq t$ such that $t' \longrightarrow^* \sqsupseteq V$ and $\emptyset \vdash t' : \bar{\theta}$. From this. it suffices to show that $t' = t$. Assume $t' \sqsubset t$ for contradiction. By the assumption, there is $\langle C, s\rangle$ such that $t = C[s]$, $s \neq \perp$, and $t' \sqsubseteq C[\perp]$. Then $C[s^{\ell}] \sqsupseteq C[\perp] \sqsupseteq t' \longrightarrow^* \sqsupseteq V \sqsupseteq D_C[\natural u_C]$, and thus $C[s^{\ell}] \longrightarrow^* \sqsupseteq D_C[\natural u_C] \dots (\star 2)$. By $(\star 1)$ and $(\star 2)$, $D_C[\ell(u_C)] \sqcup D_C[\natural u_C]$ is defined, but it is contradiction because $\natural u_C \neq \perp$ (since $D_C[\ell(u_C)]$ is tracked). ◀

H.3 Label-Generation Lemma

In this subsection we give a key lemma (Lemma 68) to prove the soundness.

▶ **Lemma 68** (label-generation)**.** *Assume that t is a closed and ground-typed term and $\emptyset \vdash t : \bar{\theta}$. Then there is a finite tree V such that (a) $t \longrightarrow^* \sqsupseteq V$; (b) $\emptyset \vdash V : \bar{\theta}$; and (c) if t is labelled, then V so is.*

To prove it, we introduce a new reduction relation $\succeq_{\mathbf{Y}}$, for only unfolding $\mathbf{Y}$. Precisely, $\succeq_{\mathbf{Y}}$ is the binary relation on terms and $\mathbf{Y}$-free terms defined as the least relation closed under the following rules:

$$\frac{}{t^{\kappa} \succeq_{\mathbf{Y}} \perp^{\kappa}}(\succeq_{\mathbf{Y}} \perp) \quad \frac{t(\mathbf{Y}^{\kappa} t) \succeq_{\mathbf{Y}} s}{\mathbf{Y}^{\kappa} t \succeq_{\mathbf{Y}} s}(\succeq_{\mathbf{Y}} \mathbf{Y}) \quad \frac{}{x^{\kappa} \succeq_{\mathbf{Y}} x^{\kappa}}(\mathrm{Var}) \quad \frac{t_1 \succeq_{\mathbf{Y}} s_1 \quad t_2 \succeq_{\mathbf{Y}} s_2}{t_1 t_2 \succeq_{\mathbf{Y}} s_1 s_2}(\mathrm{App})$$

$$\frac{t \succeq_{\mathbf{Y}} s}{\lambda \bar{x}^{\kappa}.t \succeq_{\mathbf{Y}} \lambda \bar{x}^{\kappa}.s}(\mathrm{Abs}) \quad \frac{t_1 \succeq_{\mathbf{Y}} s_1 \quad \dots \quad t_{\Sigma(a)} \succeq_{\mathbf{Y}} s_{\Sigma(a)}}{a(t_1, \dots, t_{\Sigma(a)}) \succeq_{\mathbf{Y}} a(s_1, \dots, s_{\Sigma(a)})}(a) \quad \frac{t_1 \succeq_{\mathbf{Y}} s_1}{\ell(t_1) \succeq_{\mathbf{Y}} \ell(s_1)}(\ell)$$

We list some properties with respect to the reduction relation $\succeq_{\mathbf{Y}}$.

▶ **Proposition 69.**

(1) *If $t \sqsupseteq s \succeq_{\mathbf{Y}} u$, then $t \succeq_{\mathbf{Y}} u$.*

(2) *If* $t \succeq_{\mathbf{Y}} s \sqsupseteq u$, *then* $t \succeq_{\mathbf{Y}} u$.
(3) *If* $s \succeq_{\mathbf{Y}} s'$, *then* $t\{s/x\} \succeq_{\mathbf{Y}} t\{s'/x\}$.
(4) *If* $t \succeq_{\mathbf{Y}} t'$, *then* $t\{s/x\} \succeq_{\mathbf{Y}} t'\{s/x\}$.
(5) *If* $t \succeq_{\mathbf{Y}} s \longrightarrow u$, *then* $t \longrightarrow^* \succeq_{\mathbf{Y}} u$.
(6) *If* V *is a finite tree and* $t \succeq_{\mathbf{Y}} V$, *then* $t \longrightarrow^* \sqsupseteq V$.

Proof. (1): By simple induction on the derivation tree of $s \succeq_{\mathbf{Y}} u$. (2): By simple induction on the derivation tree of $t \succeq_{\mathbf{Y}} s$. (3)(4): By simple induction on the structure t.

(5): By induction on the derivation trees of $t \succeq_{\mathbf{Y}} s$. We do case analysis on the last derivation step of $t \succeq_{\mathbf{Y}} s$.

Case $(\succeq_{\mathbf{Y}} \bot)$(Var)(Abs): These cases does not occur because $s \longrightarrow u$.

Case $(\succeq_{\mathbf{Y}} \mathbf{Y})$: Then the derivation tree is of the following form.

$$\begin{array}{c} \dfrac{t_1(\mathbf{Y}t_1) \succeq_{\mathbf{Y}} s}{\mathbf{Y}t_1 \succeq_{\mathbf{Y}} s}\;(\succeq_{\mathbf{Y}} \mathbf{Y}) \\ \hdashline t \succeq_{\mathbf{Y}} s \end{array}$$

By I.H. $t_1(\mathbf{Y}t_1) \longrightarrow^* \succeq_{\mathbf{Y}} s$, and thus $t = \mathbf{Y}t_1 \longrightarrow t_1(\mathbf{Y}t_1) \longrightarrow^* \succeq_{\mathbf{Y}} s$.

Case (a): Then s is of the form $a(s_1, \dots, s_n)$, u is of the form $a(s_1, \dots, s_{i-1}, s'_i, s_{i+1}, \dots, s_n)$, and t is of the form $a(t_1, \dots, t_n)$. By $t_i \succeq_{\mathbf{Y}} s_i \longrightarrow s'_i$ and I.H., $t_i \longrightarrow^* \succeq_{\mathbf{Y}} s'_i$. Let u_i be $t_i \longrightarrow^* u_i \succeq_{\mathbf{Y}} s'_i$. Then $t \longrightarrow^* a(t_1, \dots, t_{i-1}, u_i, t_{i+1}, \dots, t_n) \succeq_{\mathbf{Y}} s'$. Hence $t \longrightarrow^* \succeq_{\mathbf{Y}} s'$.

Case (App): We do case analysis on the last rule of the derivation tree of $s \longrightarrow s'$.

Sub-Case $(\bot)$: Then $u = \bot$, so $t \longrightarrow^0 t \succeq_{\mathbf{Y}} u$ by $(\succeq_{\mathbf{Y}} \bot)$.

Sub-Case (β): Then s is of the form $(\lambda x.s_1)s_2$, u is of the form $s_1\{s_2/x\}$, and t is of the form $(\lambda x.t_1)t_2$. Then $t \longrightarrow t_1\{t_2/x\} \succeq_{\mathbf{Y}} s_1\{s_2/x\} = u$ by $t_1 \succeq_{\mathbf{Y}} s_1$, $t_2 \succeq_{\mathbf{Y}} s_2$, (3) and (4).

Sub-Case (App): Then s is of the form $s_1 s_2$, u is of the form $s'_1 s_2$, and t is of the form $t = t_1 t_2$. Then by $t_1 \succeq_{\mathbf{Y}} s_1$, $s_1 \longrightarrow s'_1$, and I.H., $t_1 \longrightarrow^* \succeq_{\mathbf{Y}} s'_1$. Let u_1 be s.t. $t_1 \longrightarrow^* u_1 \succeq_{\mathbf{Y}} s'_1$. Then $t = t_1 t_2 \longrightarrow^* u_1 t_2 \succeq_{\mathbf{Y}} s'_1 s_2 = s$ by $t_2 \succeq_{\mathbf{Y}} s_2$. Hence $t \longrightarrow^* \succeq_{\mathbf{Y}} s'$.

(6): By induction on the derivation tree. Case $(\succeq_{\mathbf{Y}} \bot)$: Then $V = \bot$, and thus $t \longrightarrow^* \sqsupseteq V$.

Case $(\succeq_{\mathbf{Y}} \mathbf{Y})$: Then the derivation tree is of the following form.

$$\begin{array}{c} \dfrac{t_1(\mathbf{Y}t_1) \succeq_{\mathbf{Y}} V}{\mathbf{Y}t_1 \succeq_{\mathbf{Y}} V}\;(\succeq_{\mathbf{Y}} \mathbf{Y}) \\ \hdashline t \succeq_{\mathbf{Y}} V \end{array}$$

By I.H., $t_1(\mathbf{Y}t_1) \longrightarrow^* \sqsupseteq V$. Therefore $t \longrightarrow^* \sqsupseteq V$ is shown by $t = \mathbf{Y}t_1 \longrightarrow t_1(\mathbf{Y}t_1) \longrightarrow^* \sqsupseteq V$.

Case (a): Then the derivation tree is of the following form.

$$\begin{array}{c} \dfrac{t_1 \succeq_{\mathbf{Y}} V_1 \quad \dots \quad t_{\Sigma(a)} \succeq_{\mathbf{Y}} V_{\Sigma(a)}}{a(t_1, \dots, t_{\Sigma(a)}) \succeq_{\mathbf{Y}} a(V_1, \dots, V_{\Sigma(a)})}\;(a) \\ \hdashline t \succeq_{\mathbf{Y}} V \end{array}$$

For each $i \in [\Sigma(a)]$, by I.H., $t_i \longrightarrow^* \sqsupseteq V_i$. Let s_i be such that $t_i \longrightarrow^* s_i \sqsupseteq V_i$. Then $t = a(t_1, \dots, t_{\Sigma(a)}) \longrightarrow^* a(s_1, \dots, s_{\Sigma(a)}) \sqsupseteq a(V_1, \dots, V_{\Sigma(a)}) = V$.

Case (ℓ): In the same manner as Case (a).

Other cases do not occur because V is a finite tree. ◀

▶ **Lemma 70.** (1) *Assume that* $t \lhd \tilde{\theta}$. *Then there is a* $\mathbf{Y}$*-free term* s *such that (a)* $t \succeq_{\mathbf{Y}} s$*; (b)* $s \lhd \tilde{\theta}$*; and (c) if* t *is labelled, then* s *so is.*
(2) *Assume that* $\Theta \vdash t : \bar{\theta}$. *Then there is a* $\mathbf{Y}$*-free term* s *such that (a)* $t \succeq_{\mathbf{Y}} s$*; (b)* $\Theta \vdash s : \bar{\theta}$*; and (c) if* t *is labelled, then* s *so is.*

Proof. (1): By induction on the minimum sum of the size of derivation trees of $t_1 \lhd \{\langle \Theta_1, \tau_1 \rangle\}$, $\dots$, $t_n \lhd \{\langle \Theta_n, \tau_n \rangle\}$ such that $t = \bigsqcup_{i \in [n]} t_i$ and $\tilde{\theta} = \bigcup_{i \in [n]} \{\langle \Theta_i, \tau_i \rangle\}$. We do case analysis on the structure of t.

Case $t = x$: Then $s = x$ satisfies (a)(b)(c).

Case $t = t^1\, t^2$: Then each t_i is of the form $t_i^1\, t_i^2$ and the derivation tree of $\Theta_i \vdash t_i : \tau_i$ is of the following:

$$\cfrac{\Theta_i^1 \vdash t_i^1 : \bigwedge_{j \in [m_i]} \sigma_{i,j} \to \tau_i \qquad \cfrac{\Theta_{i,1}^2 \vdash t_{i,1}^2 : \sigma_{i,1} \quad \dots \quad \Theta_{i,m_i}^2 \vdash t_{i,m_i}^2 : \sigma_{i,m_i}}{\bigwedge_{j \in [m_i]} \Theta_{i,j}^2 \vdash \bigsqcup_{j \in [m_i]} t_{i,j}^2 : \bigwedge_{j \in [m_i]} \sigma_{i,j}}\,(\wedge)}{\dfrac{\Theta_i^1 \wedge \bigwedge_{j \in [m_i]} \Theta_{i,j}^2 \vdash t_i^1\, (\bigsqcup_{j \in [m_i]} t_{i,j}^2) : \tau_i}{\Theta_i \vdash t_i : \tau_i}}\,(\mathrm{App})$$

Let s^1 be the $\mathbf{Y}$-free term obtained from I.H. for $t^1 \lhd \bigcup_{i \in [n]} \{\langle \Theta_i^1, \bigwedge_{j \in [m_i]} \sigma_{i,j} \to \tau_i \rangle\}$. Also let s^2 be the $\mathbf{Y}$-free term obtained from I.H. for $t^2 \lhd \bigcup_{i \in [n]} \bigcup_{j \in [m_i]} \{\langle \Theta_{i,j}^2, \sigma_{i,j} \rangle\}$. Then $s = s^1\, s^2$ satisfies (a)(b)(c).

Case $t = \lambda \tilde{x}.t_1$, $t = a(t_1, \dots, t_{\Sigma(a)})$, or $t = \ell(t_1)$: In the same way as Case $t = t^1\, t^2$.

Case $t = \mathbf{Y}t^0$: Then each t_i is of the form $\mathbf{Y}t_i^0$ and the derivation tree of $\Theta_i \vdash t_i : \tau_i$ is one of the following two forms:

$$\cfrac{\Theta_i \vdash t_i^0\, (\mathbf{Y}t_i^0) : \tau_i}{\Theta_i \vdash \mathbf{Y}t_i : \tau_i}\,(\mathbf{Y}\ 1) \qquad\qquad \cfrac{\Theta_i \vdash t_i^0\, \bot : \tau_i}{\Theta_i \vdash \mathbf{Y}t_i^0 : \tau_i}\,(\mathbf{Y}\ 2)$$

Then let s be the $\mathbf{Y}$-free term obtained from I.H. for $(\bigsqcup_{i \in [n]} t_i^0)\, (\bigsqcup_{i \in [n]} t_i^1) \lhd \bigcup_{i \in [n]} \{\langle \Theta_i, \tau_i \rangle\}$, where, for each i, let $t_i^1 = \mathbf{Y}t_i^0$ if the last derivation step is $(\mathbf{Y}\ 1)$ and $t_i^1 = \bot$ if the last derivation step is $(\mathbf{Y}\ 2)$. This s satisfies (a)(b)(c). In particular (a) is shown as follows:

$$\cfrac{\cfrac{t^0\, (\mathbf{Y}t^0) \sqsupseteq (\bigsqcup_{i \in [n]} t_i^0)\, (\bigsqcup_{i \in [n]} t_i^1) \qquad (\bigsqcup_{i \in [n]} t_i^0)\, (\bigsqcup_{i \in [n]} t_i^1) \succeq_{\mathbf{Y}} s}{t^0 (\mathbf{Y}t^0) \succeq_{\mathbf{Y}} s}\,\text{Prop. 69(1)}}{\mathbf{Y}t^0 \succeq_{\mathbf{Y}} s}\,(\succeq_{\mathbf{Y}})$$

(2): Immediate from (1). ◀

▶ **Lemma 71.** *Assume that t is a closed and ground-typed term, t is $\mathbf{Y}$-free, and $\emptyset \vdash t : \bar{\theta}$. Then there is a finite tree V such that (a) $t \longrightarrow^* \sqsupseteq V$; (b) $\emptyset \vdash V : \bar{\theta}$; and (c) if t is labelled, then V so is.*

Proof. Let V' be the finite tree such that $t \longrightarrow^* V'$ (note that such V' always exists since t is $\mathbf{Y}$-free). By subject reduction lemma (Proposition 64), we have a finite tree V such that $V \sqsubseteq V'$, $\emptyset \vdash V : \mathsf{o}$, and if t is labelled, then so V is. Hence this V satisfies (a)(b)(c). ◀

Proof of Lemma 68. Assume that t is a closed and ground-typed term and $\emptyset \vdash t : \bar{\theta}$. By Lemma 70(2), there is a $\mathbf{Y}$-free term s such that (a) $t \succeq_{\mathbf{Y}} s$; (b) $\emptyset \vdash s : \bar{\theta}$; and (c) if t is labelled, then s so is. By Lemma 71, there is a finite tree V such that (a) $s \longrightarrow^* \sqsupseteq V$; (b) $\emptyset \vdash V : \bar{\theta}$; and (c) if s is labelled, then V so is. This V satisfies (a)(b)(c). In particular (a) is shown as follows: By the above two, $t \succeq_{\mathbf{Y}} \longrightarrow^* \sqsupseteq V$. Then by Proposition 69(4)(5), $t \longrightarrow^* \succeq_{\mathbf{Y}} V$. Therefore by Proposition 69(6), $t \longrightarrow^* \sqsupseteq V$. ◀

H.4 Proof of the Soundness

▶ **Proposition 72.** *If $\Theta \vdash C[s] : \bar{\theta}$ and $s \neq \bot$, then $\Theta \vdash C[s^\ell] : \bar{\theta}$.*

Proof. (Recall context-types introduced in Section 6.1.) Let $\tilde{\theta} = \bigcup_{i\in[n]}\{\langle \Theta_i, \tau_i\rangle\}$ be such that $\Theta = \bigwedge_{i\in[n]} \Theta_i$, $\bigwedge \bar{\theta} = \bigwedge_{i\in[n]} \tau_i$, and $C[s] \lhd \tilde{\theta}$. By $C[s] \lhd \tilde{\theta}$ and inverse substitution lemma (Proposition 23), there is $\tilde{\theta}'$ such that $C \lhd \tilde{\theta}' \Rrightarrow \tilde{\theta}$ and $s \lhd \tilde{\theta}'$. If $s^\ell \lhd \tilde{\theta}'$ holds, by substitution lemma (Proposition 22), $C[s^\ell] \lhd \tilde{\theta}$, and hence $\Theta \vdash C[s^\ell] : \bar{\theta}$. We now show $s^\ell \lhd \tilde{\theta}'$. Let $\{\langle \Theta_i', s_i, \tau_i'\rangle\}_{i\in[n]}$ be such that $\tilde{\theta}' = \bigcup_{i\in[n]}\{\langle \Theta_i', \tau_i'\rangle\}$, $s = \bigsqcup_{i\in[n]} s_i$, and for each $i \in [n]$, $\Theta_i' \vdash s_i : \tau_i'$. Note that $n > 0$ by $s \neq \bot$. For each i, $\Theta_i' \vdash {s_i}^\ell : \tau_i'$ is shown as follows (where let ${s_i}^\ell = \lambda z_1.\ldots.\lambda z_k.\ell(s_i z_1 \ldots z_k)$ and let $\tau_i' = \delta_1 \to \ldots \to \delta_k \to \mathsf{o}$):

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Theta \vdash s_i : \tau_i'}{\Theta \vdash s_i : \delta_1 \to \ldots \to \delta_k \to \mathsf{o}} \qquad \dfrac{}{x_1 : \delta_1 \vdash x_1 : \delta_1}\,(\mathrm{Var})(\wedge) \quad \ldots \quad \dfrac{}{x_n : \delta_k \vdash x_k : \delta_k}\,(\mathrm{Var})(\wedge)}{\Theta, x_1 : \delta_1, \ldots, x_k : \delta_k \vdash s_i x_1 \ldots x_n : \mathsf{o}}\,(\mathrm{App})}{\Theta, x_1 : \delta_1, \ldots, x_k : \delta_k \vdash \ell(s_i x_1 \ldots x_n) : \mathsf{o}}\,(\ell)}{\Theta \vdash \lambda z_1.\ldots.\lambda z_k.\ell(s_i z_1 \ldots z_k) : \delta_1 \to \ldots \to \delta_k \to \mathsf{o}}\,(\mathrm{Abs})}{\Theta \vdash {s_i}^\ell : \tau_i'}$$

Therefore $s^\ell \lhd \tilde{\theta}'$ has been proved, because $s^\ell = \bigsqcup_{i\in[n]} s_i^\ell$ (note $n > 0$). ◀

▶ **Proposition 73.**

(1) *If $\emptyset \vdash \ell(u) : \bar{\theta}$ for some $\bar{\theta}$, then $\natural u \neq \bot$ (i.e., $\ell(u)$ is tracked).*
(2) *If V is a labelled finite tree and $\emptyset \vdash V : \bar{\theta}$ for some $\bar{\theta}$, then V is tracked.*

Proof. (1): We show the contraposition. By $\natural u = \bot$, u is of the form $\ell(\ldots\ell(\bot)\ldots)$. Assume that $\emptyset \vdash u : \bar{\theta}$ for some $\bar{\theta}$ (towards contradiction). We only write the case of that $\bar{\theta}$ is not prime (the case of that $\bar{\theta}$ is prime is shown in the same way). Then the derivation tree is of the following.

$$\dfrac{\dfrac{\dfrac{\dfrac{\emptyset \vdash \bot : \mathsf{o}}{\emptyset \vdash \ell(\bot) : \mathsf{o}}\,(\ell)}{\vdots}}{\emptyset \vdash \ell(\ldots\ell(\bot)\ldots) : \mathsf{o}}\,(\ell)}{\emptyset \vdash \ell(\ldots\ell(\bot)\ldots) : \bar{\theta}}\,(\wedge)$$

However it is contradiction because $\emptyset \vdash \bot : \mathsf{o}$ can not be derived.

(2): By a straight forward induction on the derivation tree of $\emptyset \vdash V : \bar{\theta}$ using (1). ◀

▶ **Theorem 74** (soundness)**.** *Let t be any closed and ground-typed term over Σ. If $\emptyset \vdash t : \bar{\theta}$ for some $\bar{\theta}$, then t is minimal.*

Proof. If $\bar{\theta} = \top$, then $t = \bot^{\mathsf{o}}$ by Proposition 61, and thus t is minimal. Otherwise, by Theorem 52, it suffices to show that, for every $\langle C, s\rangle$ such that $t = C[s]$ and $s \neq \bot$, there is a tracked finite tree V such that $C[s^\ell] \longrightarrow^* \sqsupseteq V$. Then by $\emptyset \vdash C[s] : \bar{\theta}$ (Proposition 72), $\emptyset \vdash C[s^\ell] : \bar{\theta}$. By label-generation lemma (Lemma 68), there is a labelled finite tree V such that $C[s^\ell] \longrightarrow^* \sqsupseteq V$ and $\emptyset \vdash V : \bar{\theta}$. By $\emptyset \vdash V : \bar{\theta}$ (Proposition 73), V is tracked. Hence it has been proved. ◀