On Word and Frontier Languages of Unsafe Higher-Order Grammars

Kazuyuki Asada and Naoki Kobayashi[1]

1 **The University of Tokyo**

Abstract

Higher-order grammars are extensions of regular and context-free grammars, where non-terminals may take parameters. They have been extensively studied in 1980's, and restudied recently in the context of model checking and program verification. We show that the class of unsafe order-$(n+1)$ word languages coincides with the class of frontier languages of unsafe order-n tree languages. We use intersection types for transforming an order-$(n+1)$ word grammar to a corresponding order-n tree grammar. The result has been proved for safe languages by Damm in 1982, but it has been open for unsafe languages, to our knowledge. Various known results on higher-order grammars can be obtained as almost immediate corollaries of our result.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases intersection types, higher-order grammars

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

Higher-order grammars are extension of regular and context-free grammars, where non-terminals may take trees or (higher-order) functions on trees as parameters. They have been extensively studied in 1980's [6, 7, 8], and recently reinvestigated in the context of model checking [10, 17] and applied to program verification [11].

The present paper shows that the class of unsafe order-$(n+1)$ word languages coincides with the class of "frontier languages" of unsafe order-n tree languages. Here, the frontier of a tree is the sequence of symbols that occur in the leaves of the tree from left to right, and the frontier language of a tree language consists of the frontiers of elements of the tree language. The special case where $n = 0$ corresponds to the well-known fact that the frontier language of a regular tree language is a context-free language. The result has been proved by Damm [6] for grammars with the safety restriction (see [16] for a nice historical account of the safety restriction), but it has been open for unsafe grammars, to our knowledge.[1]

Damm's proof relied on the safety restriction (in particular, the fact that variable renaming is not required for safe grammars [2]) and does not apply (at least directly) to the case of unsafe grammars. We instead use intersection types to transform an order-$(n+1)$ word grammar $\mathcal{G}$ to an order-n tree grammar $\mathcal{G}'$ such that the frontier language of $\mathcal{G}'$ coincides with the language generated by $\mathcal{G}$. Intersection types have been used for recent other studies of higher-order grammars and model checking [11, 13, 12, 15, 19, 18, 14, 21]; our proof in the present paper provides yet another evidence that intersection types are a versatile tool for studies of higher-order grammars. Compared with the previous work on intersection

[1] Kobayashi et al. [13] mentioned the result, referring to the paper under preparation: "On Unsafe Tree and Leaf Languages," which is actually the present paper.

types for higher-order grammars, the technical novelties include: (i) our intersection types (used in Section 3) are mixtures of non-linear and linear intersection types and (ii) our type-based transformation involves global restructuring of terms. These points have made the correctness of the transformations non-trivial and delicate.

As stressed by Damm [6] at the beginning of his paper, the result will be useful for analyzing properties of higher-order languages by induction on the order of grammars. Our result allows properties on (unsafe) order-n languages to be reduced to those on order-$(n-1)$ tree languages, and then the latter may be studied by investigating those on the path languages of order-$(n-1)$ tree languages, which are order-$(n-1)$ word languages. As a demonstration of this, we sketch an alternative proof of the decidability of the diagonal problem for unsafe languages [4] in Section 5, along with other applications.

The rest of this paper is structured as follows. Section 2 reviews the definition of higher-order grammars, and states the main result. Sections 3 and 4 prove the result by providing the (two-step) transformations from order-$(n+1)$ word grammars to order-n tree grammars. Section 5 discusses applications of the result. Section 6 discusses related work and Section 7 concludes the paper.

2 Preliminaries

This section defines higher-order grammars and the languages generated by them, and then explains the main result. Most of the following definitions follow those in [13].

A higher-order grammar consists of non-deterministic rewriting rules of the form $A \to t$, where A is a non-terminal and t is a simply-typed λ-term that may contain non-terminals and terminals (tree constructors).

▶ **Definition 1** (types and terms). The set of *simple types*,[2] ranged over by κ, is given by: $\kappa ::= \mathsf{o} \mid \kappa_1 \to \kappa_2$. The order and arity of a simple type κ, written $\mathtt{order}(\kappa)$ and $\mathtt{ar}(\kappa)$, are defined respectively by:

$$\begin{array}{ll} \mathtt{order}(\mathsf{o}) = 0 & \mathtt{order}(\kappa_1 \to \kappa_2) = \max(\mathtt{order}(\kappa_1)+1, \mathtt{order}(\kappa_2)) \\ \mathtt{ar}(\mathsf{o}) = 0 & \mathtt{ar}(\kappa_1 \to \kappa_2) = 1 + \mathtt{ar}(\kappa_2) \end{array}$$

The type o describes trees, and $\kappa_1 \to \kappa_2$ describes functions from κ_1 to κ_2. The set of λ-*terms*, ranged over by t, is defined by: $t ::= x \mid A \mid a \mid t_1\, t_2 \mid \lambda x : \kappa.t$. Here, x ranges over variables, A over symbols called non-terminals, and a over symbols called terminals. We assume that each terminal a has a fixed arity; we write Σ for the map from terminals to their arities. A term t is called an *applicative term* (or simply a *term*) if it does not contain λ-abstractions. A (simple) type environment $\mathcal{K}$ is a map from variables to types, where non-terminals are also treated as variables. A λ-term t has type κ under $\mathcal{K}$ if $\mathcal{K} \vdash_{\mathsf{ST}} t : \kappa$ is derivable from the following typing rules.

$$\frac{}{\mathcal{K} \cup \{x : \kappa\} \vdash_{\mathsf{ST}} x : \kappa} \qquad \frac{}{\mathcal{K} \vdash_{\mathsf{ST}} a : \underbrace{\mathsf{o} \to \cdots \to \mathsf{o}}_{\Sigma(a)} \to \mathsf{o}}$$

$$\frac{\mathcal{K} \vdash_{\mathsf{ST}} t_1 : \kappa_2 \to \kappa \qquad \mathcal{K} \vdash_{\mathsf{ST}} t_2 : \kappa_2}{\mathcal{K} \vdash_{\mathsf{ST}} t_1\, t_2 : \kappa} \qquad \frac{\mathcal{K} \cup \{x : \kappa_1\} \vdash_{\mathsf{ST}} t : \kappa_2}{\mathcal{K} \vdash_{\mathsf{ST}} \lambda x : \kappa_1.t : \kappa_1 \to \kappa_2}$$

We call t a (finite, Σ-ranked) *tree* if t is an applicative term consisting of only terminals, and

[2] We sometimes call simple types *sorts* in this paper, to avoid confusion with intersection types introduced later for grammar transformations.

$\emptyset \vdash_{\mathtt{ST}} t : \mathtt{o}$ holds. We write $\mathbf{Tree}_\Sigma$ for the set of Σ-ranked trees, and use the meta-variable π for a tree.

We often omit type annotations and just write $\lambda x.t$ for $\lambda x:\kappa.t$. We consider below only well-typed λ-terms of the form $\lambda x_1.\cdots\lambda x_k.t$, where t is an applicative term. We are now ready to define higher-order grammars.

▶ **Definition 2** (higher-order grammar)**.** A *higher-order grammar* is a quadruple $(\Sigma, \mathcal{N}, \mathcal{R}, S)$, where (i) Σ is a ranked alphabet; (ii) $\mathcal{N}$ is a map from a finite set of non-terminals to their types; (iii) $\mathcal{R}$ is a finite set of *rewriting rules* of the form $A \to \lambda x_1.\cdots\lambda x_\ell.t$, where $\mathcal{N}(A) = \kappa_1 \to \cdots \to \kappa_\ell \to \mathtt{o}$, t is an applicative term, and $\mathcal{N}, x_1\!:\!\kappa_1, \ldots, x_\ell\!:\!\kappa_\ell \vdash_{\mathtt{ST}} t : \mathtt{o}$ holds for some $\kappa_1, \ldots, \kappa_\ell$. (iv) S is a non-terminal called *the start symbol*, and $\mathcal{N}(S) = \mathtt{o}$. The *order* of a grammar $\mathcal{G}$, written $\mathtt{order}(\mathcal{G})$, is the largest order of the types of non-terminals. We sometimes write $\Sigma_\mathcal{G}, \mathcal{N}_\mathcal{G}, \mathcal{R}_\mathcal{G}, S_\mathcal{G}$ for the four components of $\mathcal{G}$.

For a grammar $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}, S)$, the rewriting relation $\longrightarrow_\mathcal{G}$ is defined by:

$$\frac{(A \to \lambda x_1.\cdots\lambda x_k.t) \in \mathcal{R}}{A\,t_1\,\cdots\,t_k \longrightarrow_\mathcal{G} [t_1/x_1, \ldots, t_k/x_k]t} \qquad \frac{t_i \longrightarrow_\mathcal{G} t_i' \qquad i \in \{1, \ldots, k\} \qquad \Sigma(a) = k}{a\,t_1\,\cdots\,t_k \longrightarrow_\mathcal{G} a\,t_1\,\cdots\,t_{i-1}\,t_i'\,t_{i+1}\,\cdots\,t_k}$$

Here, $[t_1/x_1, \ldots, t_k/x_k]t$ is the term obtained by substituting t_i for the free occurrences of x_i in t. We write $\longrightarrow_\mathcal{G}^*$ for the reflexive transitive closure of $\longrightarrow_\mathcal{G}$.

The *tree language generated by* $\mathcal{G}$, written $\mathcal{L}(\mathcal{G})$, is the set $\{\pi \in \mathbf{Tree}_{\Sigma_\mathcal{G}} \mid S \longrightarrow_\mathcal{G}^* \pi\}$. We call a grammar $\mathcal{G}$ a *word grammar* if all the terminal symbols have arity 1 except the special terminal $\mathtt{e}$, whose arity is 0. The *word language* generated by a word grammar $\mathcal{G}$, written $\mathcal{L}_\mathtt{w}(\mathcal{G})$, is $\{a_1 \cdots a_n \mid a_1(\cdots(a_n\,\mathtt{e})\cdots) \in \mathcal{L}(\mathcal{G})\}$. The frontier word of a tree π, written $\mathbf{leaves}(\pi)$, is the sequence of symbols in the leaves of π. It is defined inductively by: $\mathbf{leaves}(a) = a$, and $\mathbf{leaves}(a\,\pi_1\,\cdots\,\pi_k) = \mathbf{leaves}(\pi_1)\cdots\mathbf{leaves}(\pi_k)$ when $\Sigma(a) = k > 0$. The *frontier language* generated by $\mathcal{G}$, written $\mathcal{L}_\mathtt{leaf}(\mathcal{G})$, is the set: $\{\mathbf{leaves}(\pi) \mid S \longrightarrow_\mathcal{G}^* \pi \in \mathbf{Tree}_{\Sigma_\mathcal{G}}\}$. In our main theorem, we assume that there is a special nullary symbol $\mathtt{e}$ and consider $\mathtt{e} \in \mathcal{L}_\mathtt{leaf}(\mathcal{G})$ as the empty word ε; i.e., we consider $\mathcal{L}_\mathtt{leaf}^\varepsilon(\mathcal{G})$ defined by:

$$\mathcal{L}_\mathtt{leaf}^\varepsilon(\mathcal{G}) := (\mathcal{L}_\mathtt{leaf}(\mathcal{G}) \setminus \{\mathtt{e}\}) \cup \{\varepsilon \mid \mathtt{e} \in \mathcal{L}_\mathtt{leaf}(\mathcal{G})\}.$$

We note that the classes of order-0 and order-1 word languages coincide with those of regular and context-free languages respectively. We often write $A\,x_1\,\cdots\,x_k \to t$ for the rule $A \to \lambda x_1.\cdots\lambda x_k.t$. When considering the frontier language of a tree grammar, we assume, without loss of generality, that the ranked alphabet Σ has a unique binary symbol $\mathtt{br}$, and that all the other terminals have arity 0.

▶ **Example 3.** Consider the order-2 (word) grammar $\mathcal{G}_1 = (\{\mathtt{a}:1, \mathtt{b}:1, \mathtt{e}:0\}, \{S:\mathtt{o}, F:(\mathtt{o} \to \mathtt{o}) \to \mathtt{o}, A:(\mathtt{o} \to \mathtt{o}) \to (\mathtt{o} \to \mathtt{o}), B:(\mathtt{o} \to \mathtt{o}) \to (\mathtt{o} \to \mathtt{o})\}, \mathcal{R}_1, S)$, where $\mathcal{R}_1$ consists of:

$$\begin{array}{llll} S \to F\,\mathtt{a} & S \to F\,\mathtt{b} & A\,f\,x \to \mathtt{a}(f\,x) & B\,f\,x \to \mathtt{b}(f\,x), \\ F\,f \to f(f\,\mathtt{e}) & F\,f \to F\,(A\,f) & F\,f \to F\,(B\,f). & \end{array}$$

S is reduced, for example, as follows.

$$S \longrightarrow F\,\mathtt{b} \longrightarrow F\,(A\,\mathtt{b}) \longrightarrow (A\,\mathtt{b})(A\,\mathtt{b}\,\mathtt{e}) \longrightarrow \mathtt{a}\,(\mathtt{b}\,(A\,\mathtt{b}\,\mathtt{e})) \longrightarrow \mathtt{a}(\mathtt{b}\,(\mathtt{a}(\mathtt{b}\,\mathtt{e}))).$$

The word language $\mathcal{L}_\mathtt{w}(\mathcal{G}_1)$ is $\{ww \mid w \in \{\mathtt{a}, \mathtt{b}\}^+\}$.

Consider the order-1 (tree) grammar $\mathcal{G}_2 = (\{\mathtt{br}:2, \mathtt{a}:0, \mathtt{b}:0, \mathtt{e}:0\}, \{S:\mathtt{o}, F:\mathtt{o} \to \mathtt{o}\}, \mathcal{R}_2, S)$, where $\mathcal{R}_2$ consists of:

$$S \to F\,\mathtt{a} \qquad S \to F\,\mathtt{b} \qquad F\,f \to \mathtt{br}\,f\,f \qquad F\,f \to F(\mathtt{br}\,\mathtt{a}\,f) \quad F\,f \to F(\mathtt{br}\,\mathtt{b}\,f).$$

The frontier language $\mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}_2)$ coincides with $\mathcal{L}_{\mathtt{w}}(\mathcal{G}_1)$ above.

The following is the main theorem we shall prove in this paper.

▶ **Theorem 4.** *For any order-$(n+1)$ word grammar $\mathcal{G}$ $(n \geq 0)$, there exists an order-n tree grammar $\mathcal{G}'$ such that $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$.*

The converse of the above theorem also holds:

▶ **Theorem 5.** *For any order-n tree grammar $\mathcal{G}'$ such that no word in $\mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$ contains $\mathtt{e}$, there exists a word grammar $\mathcal{G}$ of order at most $n+1$ such that $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$.*

Since the construction of $\mathcal{G}$ is easy, we sketch it here; see Appendix C for a proof. For $n \geq 1$, the grammar $\mathcal{G}$ is obtained by (i) changing the arity of each nullary terminal a $(\neq \mathtt{e})$ to one, i.e., $\Sigma_{\mathcal{G}}(a) := 1$, (ii) replacing the terminal $\mathtt{e}$ with a new non-terminal E of type $\mathtt{o} \to \mathtt{o}$, defined by $E\,x \to x$, and also the unique binary terminal $\mathtt{br}$ with a new non-terminal Br of type $(\mathtt{o} \to \mathtt{o}) \to (\mathtt{o} \to \mathtt{o}) \to (\mathtt{o} \to \mathtt{o})$, defined by $Br\,f\,g\,x \to f(g\,x)$, (iii) applying η-expansion to the right hand side of each (original) rule to add an order-0 argument, and (iv) adding new start symbol S' with rule $S' \to S\mathtt{e}$. For example, given the grammar $\mathcal{G}_2$ above, the following grammar is obtained:

$$\begin{array}{lll} S' \to S\,\mathtt{e} \quad & S\,x \to F\,\mathtt{a}\,x \quad & S\,x \to F\,\mathtt{b}\,x \\ F\,f\,x \to Br\,f\,f\,x \quad & F\,f\,x \to F(Br\,\mathtt{a}\,f)\,x \quad & F\,f\,x \to F(Br\,\mathtt{b}\,f)\,x \\ E\,x \to x \quad & Br\,f\,g\,x \to f(g\,x). & \end{array}$$

Theorem 4 is proved by two-step grammar transformations, both of which are based on intersection types. In the first step, we transform an order-$(n+1)$ word grammar $\mathcal{G}$ to an order-n tree grammar $\mathcal{G}''$ such that $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}'')\!\uparrow_{\mathtt{e}}$, where $\mathcal{L}\!\uparrow_{\mathtt{e}}$ is the word language obtained from $\mathcal{L}$ by removing all the occurrences of the special terminal $\mathtt{e}$; that is, the frontier language of $\mathcal{G}''$ is almost the same as $\mathcal{L}_{\mathtt{w}}(\mathcal{G})$, except that the former may contain multiple occurrences of the special, dummy symbol $\mathtt{e}$. In the second step, we clean up the grammar to eliminate $\mathtt{e}$ (except that a singleton tree $\mathtt{e}$ may be generated when $\epsilon \in \mathcal{L}_{\mathtt{w}}(\mathcal{G})$). The first and second steps shall be formalized in Sections 3 and 4 respectively.

For the target of the transformations, we use the following extended terms, in which a *set* of terms may occur in an argument position:

$$\begin{array}{rl} u \text{ (extended terms) } & ::= x \mid A \mid a \mid u_0 U \mid \lambda x.u \\ U & ::= \{u_1, \ldots, u_k\} \ (k \geq 1). \end{array}$$

Here, $u_0\,u_1$ is interpreted as just a shorthand for $u_0\{u_1\}$. Intuitively, $\{u_1, \ldots, u_k\}$ is considered a non-deterministic choice $u_1 + \cdots + u_k$, which (lazily) reduces to u_i non-deterministically. The typing rules are extended accordingly by:

$$\frac{\mathcal{K} \vdash_{\mathtt{ST}} u_0 : \kappa_1 \to \kappa \qquad \mathcal{K} \vdash_{\mathtt{ST}} U : \kappa_1}{\mathcal{K} \vdash_{\mathtt{ST}} u_0\,U : \kappa} \qquad\qquad \frac{\mathcal{K} \vdash_{\mathtt{ST}} u_i : \kappa \text{ for each } i \in \{1, \ldots, k\}}{\mathcal{K} \vdash_{\mathtt{ST}} \{u_1, \ldots, u_k\} : \kappa}$$

An *extended higher-order grammar* is the same as a higher-order grammar, except that each rewriting rule in $\mathcal{R}$ may be of the form $\lambda x_1 \cdots \lambda x_\ell.u$, where u may be an applicative extended term. The reduction rule for non-terminals is replaced by:

$$\frac{(A \to \lambda x_1 \, \cdots \, \lambda x_k.u) \in \mathcal{R} \qquad u' \in [U_1/x_1, \ldots, U_k/x_k]u}{A\,U_1 \, \cdots \, U_k \longrightarrow_{\mathcal{G}} u'}$$

where the substitution θu is defined by:

$$\theta a = \{a\} \qquad \theta x = \begin{cases} \theta(x) & (\text{if } x \in \mathit{dom}(\theta)) \\ \{x\} & (\text{otherwise}) \end{cases}$$
$$\theta(u_0 U) = \{v(\theta U) \mid v \in \theta u_0\} \qquad \theta\{u_1, \dots, u_k\} = \theta u_1 \cup \dots \cup \theta u_k\,.$$

Also, the other reduction rule is replaced by the following two rules:

$$\frac{u \longrightarrow_{\mathcal{G}} u' \qquad i \in \{1, \dots, k\} \qquad \Sigma(a) = k}{a\, U_1 \cdots U_{i-1}\, \{u\}\, U_{i+1} \cdots U_k \longrightarrow_{\mathcal{G}} a\, U_1 \cdots U_{i-1}\, \{u'\}\, U_{i+1} \cdots U_k}$$

$$\frac{u \in U_i \qquad U_i \text{ is not a singleton} \qquad i \in \{1, \dots, k\} \qquad \Sigma(a) = k}{a\, U_1 \cdots U_k \longrightarrow_{\mathcal{G}} a\, U_1 \cdots U_{i-1}\, \{u\}\, U_{i+1} \cdots U_k}$$

Note that unlike in the extended grammar introduced in [13], there is no requirement that each of u_i is used at least once. Thus, the extended syntax does not change the expressive power of grammars. A term set $\{u_1, \dots, u_k\}$ can be replaced by $A\, x_1 \cdots x_\ell$ with the rewriting rules $A\, x_1 \cdots x_\ell \to u_i$, where $\{x_1, \dots, x_\ell\}$ is the set of variables occurring in some of $u_1, \dots, u_k$. In other words, for any order-n extended grammar $\mathcal{G}$, there is an (ordinary) order-n grammar $\mathcal{G}'$ such that $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}')$.

3 Step 1: from order-$(n+1)$ grammar to order-n tree grammar

In this section, we show that for any order-$(n+1)$ grammar $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}, S)$ such that $\Sigma(\mathtt{e}) = 0$ and $\Sigma(a) = 1$ for every $a \in \mathit{dom}(\Sigma) \setminus \{\mathtt{e}\}$, there exists an order-$n$ grammar $\mathcal{G}'$ such that $\Sigma_{\mathcal{G}'} = \{\mathtt{br} \mapsto 2, \mathtt{e} \mapsto 0\} \cup \{a \mapsto 0 \mid \Sigma(a) = 1\}$ and $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}')\!\uparrow_{\mathtt{e}}$.

For technical convenience, we assume below that, for every type κ occurring in the word grammar $\mathcal{G}$, if κ is of the form $\mathtt{o} \to \kappa'$, then $\mathtt{order}(\kappa') \leq 1$. This does not lose generality, since any function $\lambda x : \mathtt{o}.t$ of type $\mathtt{o} \to \kappa'$ with $\mathtt{order}(\kappa') > 1$ can be replaced by the term $\lambda x' : \mathtt{o} \to \mathtt{o}.[x'\mathtt{e}/x]t$ of type $(\mathtt{o} \to \mathtt{o}) \to \kappa'$ (without changing the order of the term), and any term t of type $\mathtt{o}$ can be replaced by the term $A\, t$ of type $\mathtt{o} \to \mathtt{o}$, where A is a non-terminal of type $\mathtt{o} \to \mathtt{o} \to \mathtt{o}$, with rule $A\, x\, y \to x$.

The basic idea of the transformation is to remove all the order-0 arguments (i.e., arguments of tree type $\mathtt{o}$). This reduces the order of each term by 1; for example, terms of types $\mathtt{o} \to \mathtt{o}$ and $(\mathtt{o} \to \mathtt{o}) \to \mathtt{o}$ will respectively be transformed to those of types $\mathtt{o}$ and $\mathtt{o} \to \mathtt{o}$. Order-0 arguments can indeed be removed as follows. Suppose we have a term $t_1\, t_2$ where $t_1 : \mathtt{o} \to \mathtt{o}$. If t_1 does not use the order-0 argument t_2, then we can simply replace $t_1\, t_2$ with $t_1^{\#}$ (where $t_1^{\#}$ is the result of recursively applying the transformation to t_1). If t_1 uses the argument t_2, the word generated by $t_1\, t_2$ must be of the form $w_1 w_2$, where w_2 is generated by t_2; in other words, t_1 can only append a word to the word generated by t_2. Thus, $t_1\, t_2$ can be transformed to $\mathtt{br}\ t_1^{\#}\ t_2^{\#}$, which can generate a tree whose frontier coincides with $w_1 w_2$ (if $\mathtt{e}$ is ignored). As a special case, a constant word $\mathtt{a}\,\mathtt{e}$ can be transformed to $\mathtt{br}\ \mathtt{a}\ \mathtt{e}$. As a little more complex example, consider the term $A\,(\mathtt{b}\,\mathtt{e})$, where A is defined by $A\, x \to \mathtt{a}\, x$. Since A uses the argument, the term $A\,(\mathtt{b}\,\mathtt{e})$ is transformed to $\mathtt{br}\ A\ (\mathtt{br}\ \mathtt{b}\ \mathtt{e})$. Since A no longer takes an argument, we substitute $\mathtt{e}$ for x in the body of the rule for A (and apply the transformation recursively to $\mathtt{a}\,\mathtt{e}$). The resulting rule for A is: $A \to \mathtt{br}\ \mathtt{a}\ \mathtt{e}$. Thus, the term after the transformation generates the tree $\mathtt{br}\,(\mathtt{br}\ \mathtt{a}\,\mathtt{e})\,(\mathtt{br}\,\mathtt{b}\,\mathtt{e})$. Its frontier word is $\mathtt{aebe}$, which is equivalent to the word $\mathtt{ab}$ generated by the original term, up to removals of $\mathtt{e}$; recall that redundant occurrences of $\mathtt{e}$ will be removed by the second transformation. Note that the transformation sketched above depends on whether each order-0 argument is actually used or not. Thus, we introduce intersection types to express such information, and define the transformation as a type-directed one.

Simple types are refined to the following intersection types.

$$\delta ::= \mathtt{o} \mid \sigma \to \delta \qquad \sigma ::= \delta_1 \wedge \cdots \wedge \delta_k$$

We write $\top$ for $\delta_1 \wedge \cdots \wedge \delta_k$ when $k = 0$. We assume some total order $<$ on intersection types, and require that $\delta_1 < \cdots < \delta_k$ whenever $\delta_1 \wedge \cdots \wedge \delta_k$ occurs in an intersection type. Intuitively, $(\delta_1 \wedge \cdots \wedge \delta_k) \to \delta$ describes a function that uses an argument according to types $\delta_1, \ldots, \delta_k$, and the returns a value of type δ. As a special case, the type $\top \to \mathtt{o}$ describes a function that ignores an argument, and returns a tree. Thus, according to the idea of the transformation sketched above, if x has type $\top \to \mathtt{o}$, $x\,t$ would be transformed to x; if x has type $\mathtt{o} \to \mathtt{o}$, $x\,t$ would be transformed to $\mathtt{br}\ x\ t^{\#}$. In the last example above, the type $\mathtt{o} \to \mathtt{o}$ should be interpreted as a function that uses the argument *just once*; otherwise the transformation to $\mathtt{br}\ x\ t^{\#}$ would be incorrect. Thus, the type $\mathtt{o}$ should be treated as a linear type, for which weakening and dereliction are disallowed. In contrast, we need not enforce, for example, that a value of the intersection type $\mathtt{o} \to \mathtt{o}$ should be used just once. Therefore, we classify intersection types into two kinds; one called *balanced*, which may be treated as non-linear types, and the other called *unbalanced*, which must be treated as linear types. For that purpose, we introduce two refinement relations $\delta ::_{\mathrm{b}} \kappa$ and $\delta ::_{\mathrm{u}} \kappa$; the former means that δ is a balanced intersection type of sort κ, and the latter means that δ is an unbalanced intersection type of sort κ. The relations are defined as follows, by mutual induction; k may be 0.

$$\frac{\delta_j ::_{\mathrm{u}} \kappa \qquad j \in \{1, \ldots, k\} \qquad \delta_i ::_{\mathrm{b}} \kappa \text{ (for each } i \in \{1, \ldots, k\} \setminus \{j\})}{\delta_1 \wedge \cdots \wedge \delta_k ::_{\mathrm{u}} \kappa} \qquad \frac{\delta_i ::_{\mathrm{b}} \kappa \text{ (for each } i \in \{1, \ldots, k\})}{\delta_1 \wedge \cdots \wedge \delta_k ::_{\mathrm{b}} \kappa}$$

$$\frac{}{\mathtt{o} ::_{\mathrm{u}} \mathtt{o}} \qquad \frac{\sigma ::_{\mathrm{b}} \kappa \qquad \delta ::_{\mathrm{u}} \kappa'}{\sigma \to \delta ::_{\mathrm{u}} \kappa \to \kappa'} \qquad \frac{\sigma ::_{\mathrm{u}} \kappa \qquad \delta ::_{\mathrm{u}} \kappa'}{\sigma \to \delta ::_{\mathrm{b}} \kappa \to \kappa'} \qquad \frac{\sigma ::_{\mathrm{b}} \kappa \qquad \delta ::_{\mathrm{b}} \kappa'}{\sigma \to \delta ::_{\mathrm{b}} \kappa \to \kappa'}$$

A type δ is called *balanced* if $\delta ::_{\mathrm{b}} \kappa$ for some κ, and called *unbalanced* if $\delta ::_{\mathrm{u}} \kappa$ for some κ. Intuitively, unbalanced types describe trees or closures that contain the end of a word (i.e., symbol $\mathtt{e}$). Intersection types that are neither balanced nor unbalanced are considered ill-formed, and excluded out. For example, the type $\mathtt{o} \to \mathtt{o} \to \mathtt{o}$ (as an intersection type) is ill-formed; since $\mathtt{o}$ is unbalanced, $\mathtt{o} \to \mathtt{o}$ must also be unbalanced according to the rules for arrow types, but it is actually balanced. Note that, in fact, no term can have the intersection type $\mathtt{o} \to \mathtt{o} \to \mathtt{o}$ in a word grammar. We write $\delta :: \kappa$ if $\delta ::_{\mathrm{b}} \kappa$ or $\delta ::_{\mathrm{u}} \kappa$.

We introduce a type-directed transformation relation $\Gamma \vdash t : \delta \Rightarrow u$ for terms, where Γ is a set of type bindings of the form $x : \delta$, called a *type environment*, t is a source term, and u is the image of the transformation, which may be an extended term. We write $\Gamma_1 \cup \Gamma_2$ for the union of Γ_1 and Γ_2; it is defined only if, whenever $x : \delta \in \Gamma_1 \cap \Gamma_2$, δ is balanced. In other words, unbalanced types are treated as linear types, whereas balanced ones as non-linear (or idempotent) types. We write $\mathbf{bal}(\Gamma)$ if δ is balanced for every $x : \delta \in \Gamma$.

The relation $\Gamma \vdash t : \delta \Rightarrow u$ is defined inductively by the following rules.

$$\frac{\mathbf{bal}(\Gamma)}{\Gamma, x : \delta \vdash x : \delta \Rightarrow x_\delta} \quad (\textsc{Tr1-Var}) \qquad \frac{A :: \mathcal{N}(A) \qquad \mathbf{bal}(\Gamma)}{\Gamma \vdash A : \delta \Rightarrow A_\delta} \quad (\textsc{Tr1-NT})$$

$$\frac{\mathbf{bal}(\Gamma)}{\Gamma \vdash \mathtt{e} : \mathtt{o} \Rightarrow \mathtt{e}} \quad (\textsc{Tr1-Const0}) \qquad \frac{\Sigma(a) = 1 \qquad \mathbf{bal}(\Gamma)}{\Gamma \vdash a : \mathtt{o} \to \mathtt{o} \Rightarrow a} \quad (\textsc{Tr1-Const1})$$

$$\frac{\begin{array}{c}\Gamma_0 \vdash s : \delta_1 \wedge \cdots \wedge \delta_k \to \delta \Rightarrow v\\ \Gamma_i \vdash t : \delta_i \Rightarrow U_i \text{ and } \delta_i \neq \mathtt{o} \text{ (for each } i \in \{1,\dots,k\})\end{array}}{\Gamma_0 \cup \Gamma_1 \cup \cdots \cup \Gamma_k \vdash st : \delta \Rightarrow vU_1 \cdots U_k} \qquad (\textsc{Tr1-App1})$$

$$\frac{\Gamma_0 \vdash s : \mathtt{o} \to \delta \Rightarrow V \qquad \Gamma_1 \vdash t : \mathtt{o} \Rightarrow U}{\Gamma_0 \cup \Gamma_1 \vdash st : \delta \Rightarrow \mathtt{br}\, V\, U} \qquad (\textsc{Tr1-App2})$$

$$\frac{\Gamma \vdash t : \delta \Rightarrow u_i \text{ (for each } i \in \{1,\dots,k\}) \qquad k \geq 1}{\Gamma \vdash t : \delta \Rightarrow \{u_1,\dots,u_k\}} \qquad (\textsc{Tr1-Set})$$

$$\frac{\begin{array}{c}\Gamma, x : \delta_1, \dots, x : \delta_k \vdash t : \delta \Rightarrow u \qquad x \notin |\Gamma|\\ \delta_i \neq \mathtt{o} \text{ for each } i \in \{1,\dots,k\}\end{array}}{\Gamma \vdash \lambda x.t : \delta_1 \wedge \cdots \wedge \delta_k \to \delta \Rightarrow \lambda x_{\delta_1} \cdots \lambda x_{\delta_k}.u} \qquad (\textsc{Tr1-Abs1})$$

$$\frac{\Gamma, x : \mathtt{o} \vdash t : \delta \Rightarrow u}{\Gamma \vdash \lambda x.t : \mathtt{o} \to \delta \Rightarrow [\mathtt{e}/x_{\mathtt{o}}]u} \qquad (\textsc{Tr1-Abs2})$$

In rule (Tr1-Var), a variable is replicated for each type. This is because the image of the transformation of a term substituted for x is different depending on the type of the term; accordingly, in rule (Tr1-Abs1), bound variables are also replicated, and in rule (Tr1-App1), arguments are replicated. In rule (Tr1-NT), a non-terminal is also replicated for each type. In rules (Tr1-Const0) and (Tr1-Const1), constants are mapped to themselves; however, the arities of all the constants become 0. In these rules, Γ may contain only bindings on balanced types.

In rule (Tr1-App1), the first premise indicates that the function s uses the argument t according to types $\delta_1, \dots, \delta_k$. Since the image of the transformation of t depends on its type, we replicate the argument to $U_1, \dots, U_k$. For each type δ_i, the result of the transformation is not unique (but finite); thus, we represent the image of the transformation as a *set* U_i of terms. (Recall the remark at the end of Section 2 that a set of terms can be replaced by an ordinary term by introducing auxiliary non-terminals.) For example, consider a term $A(x\,y)$. It can be transformed to $A_{\delta_1 \to \delta}\{x_{\delta_0 \to \delta_1} y_{\delta_0}, x_{\delta'_0 \to \delta_1} y_{\delta'_0}\}$ under the type environment $\{x : \delta_0 \to \delta_1, x : \delta'_0 \to \delta_1, y : \delta_0, y : \delta'_0\}$. Note that k in rule (Tr1-App1) (and also (Tr1-Abs1)) may be 0, in which case the argument disappears in the image of the transformation.

In rule (Tr1-App2), as explained at the beginning of this section, the argument t of type $\mathtt{o}$ is removed from s and instead attached as a sibling node of the tree generated by (the transformation image of) s. Accordingly, in rule (Tr1-Abs2), the binder for x is removed and x in the body of the abstraction is replaced with the empty tree $\mathtt{e}$. In rule (Tr1-Set), type environments are shared. This is because $\{u_1, \dots, u_k\}$ represents the choice $u_1 + \cdots + u_k$; unbalanced (i.e. linear) values should be used in the same manner in $u_1, \dots, u_k$.

The transformation rules for rewriting rules and grammars are given by:

$$\frac{\emptyset \vdash \lambda x_1. \cdots \lambda x_k.t : \delta \Rightarrow \lambda x'_1. \cdots \lambda x'_\ell.u \qquad \delta :: \mathcal{N}(A)}{(A\, x_1\, \cdots\, x_k \to t) \Rightarrow (A_\delta\, x'_1\, \cdots\, x'_\ell \to u)} \qquad (\textsc{Tr1-Rule})$$

$$\frac{\begin{array}{c}\Sigma' = \{\mathtt{br} \mapsto 2, \mathtt{e} \mapsto 0\} \cup \{a \mapsto 0 \mid \Sigma(a) = 1\}\\ \mathcal{N}' = \{A_\delta : [\![\delta :: \kappa]\!] \mid \mathcal{N}(A) = \kappa \wedge \delta :: \kappa\} \qquad \mathcal{R}' = \{r' \mid \exists r \in \mathcal{R}. r \Rightarrow r'\}\end{array}}{(\Sigma, \mathcal{N}, \mathcal{R}, S) \Rightarrow (\Sigma', \mathcal{N}', \mathcal{R}', S_{\mathtt{o}})} \qquad (\textsc{Tr1-Gram})$$

Here, $[\![\delta :: \kappa]\!]$ is defined by:

$$
\begin{array}{l}
[\![\delta :: \kappa]\!] = \mathtt{o} \text{ if } \mathtt{order}(\kappa) \leq 1 \\
[\![(\delta_1 \wedge \cdots \wedge \delta_k \to \delta) :: (\kappa_0 \to \kappa)]\!] = [\![\delta_1 :: \kappa_0]\!] \to \ldots \to [\![\delta_k :: \kappa_0]\!] \to [\![\delta :: \kappa\,]\!] \ \text{ if } \mathtt{order}(\kappa_0 \to \kappa) > 1
\end{array}
$$

▶ **Example 6.** Recall the grammar $\mathcal{G}_1$ in Example 3. For the term $\lambda f.\lambda x.\mathtt{a}(f\,x)$ of the rule for A, we have the following derivation:

$$
\dfrac{\dfrac{\dfrac{}{\emptyset \vdash \mathtt{a} : \mathtt{o} \to \mathtt{o} \Rightarrow \mathtt{a}}\,\textsc{Const1} \quad \dfrac{\dfrac{}{f : \mathtt{o} \to \mathtt{o} \vdash f : \mathtt{o} \to \mathtt{o} \Rightarrow f_{\mathtt{o}\to\mathtt{o}}}\,\textsc{Var} \quad \dfrac{}{x : \mathtt{o} \vdash x : \mathtt{o} \Rightarrow x_{\mathtt{o}}}\,\textsc{Var}}{f : \mathtt{o} \to \mathtt{o}, x : \mathtt{o} \vdash f\,x : \mathtt{o} \Rightarrow \mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, x_{\mathtt{o}}}\,\textsc{App2}}{\dfrac{f : \mathtt{o} \to \mathtt{o}, x : \mathtt{o} \vdash \mathtt{a}(f\,x) : \mathtt{o} \Rightarrow \mathtt{br}\,\mathtt{a}\,(\mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, x_{\mathtt{o}})}{f : \mathtt{o} \to \mathtt{o} \vdash \lambda x.\mathtt{a}(f\,x) : \mathtt{o} \to \mathtt{o} \Rightarrow \mathtt{br}\,\mathtt{a}\,(\mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, \mathtt{e})}\,\textsc{Abs2}}\,\textsc{App2}}{\emptyset \vdash \lambda f.\lambda x.\mathtt{a}(f\,x) : (\mathtt{o} \to \mathtt{o}) \to \mathtt{o} \to \mathtt{o} \Rightarrow \lambda f_{\mathtt{o}\to\mathtt{o}}.\mathtt{br}\,\mathtt{a}\,(\mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, \mathtt{e})}\,\textsc{Abs1}
$$

Notice that the argument x has been removed, and the result of the transformation has type $\mathtt{o} \to \mathtt{o}$. The whole grammar is transformed to the grammar consisting of the following rules.

$$
\begin{array}{ll}
S_{\mathtt{o}} \to F_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}}\,\mathtt{a} \qquad S_{\mathtt{o}} \to F_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}}\,\mathtt{b} & \\
A_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}\to\mathtt{o}}\, f_{\mathtt{o}\to\mathtt{o}} \to \mathtt{br}\,\mathtt{a}\,(\mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\,\mathtt{e}) & B_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}\to\mathtt{o}}\, f_{\mathtt{o}\to\mathtt{o}} \to \mathtt{br}\,\mathtt{b}\,(\mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\,\mathtt{e}) \\
F_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}}\, f_{\mathtt{o}\to\mathtt{o}} \to \mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\,(\mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\,\mathtt{e}) & F_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}}\, f_{\mathtt{o}\to\mathtt{o}} \to F_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}}(A_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}\to\mathtt{o}}\, f_{\mathtt{o}\to\mathtt{o}}) \\
F_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}}\, f_{\mathtt{o}\to\mathtt{o}} \to F_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}}(B_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}\to\mathtt{o}}\, f_{\mathtt{o}\to\mathtt{o}}). &
\end{array}
$$

Here, we have omitted rules that are unreachable from $S_{\mathtt{o}}$. For example, the rule

$$
F_{(\top\to\mathtt{o})\wedge(\mathtt{o}\to\mathtt{o})\to\mathtt{o}}\, f_{\top\to\mathtt{o}}\, f_{\mathtt{o}\to\mathtt{o}} \to \mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, f_{\top\to\mathtt{o}}
$$

may be obtained from the following derivation, but it is unreachable from $S_{\mathtt{o}}$, since F is never called with an argument of type $(\top \to \mathtt{o}) \wedge (\mathtt{o} \to \mathtt{o})$.

$$
\dfrac{\dfrac{\dfrac{}{f : \mathtt{o} \to \mathtt{o} \vdash f \Rightarrow f_{\mathtt{o}\to\mathtt{o}}}\,\textsc{Var} \quad \dfrac{\dfrac{}{f : \top \to \mathtt{o} \vdash f : \top \to \mathtt{o} \Rightarrow f_{\top\to\mathtt{o}}}\,\textsc{Var}}{f : \top \to \mathtt{o} \vdash f\,\mathtt{e} : \mathtt{o} \Rightarrow f_{\top\to\mathtt{o}}}\,\textsc{App1}}{f : \top \to \mathtt{o}, f : \mathtt{o} \to \mathtt{o} \vdash f(f\,\mathtt{e}) : \mathtt{o} \Rightarrow \mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, f_{\top\to\mathtt{o}}}\,\textsc{App2}}{\emptyset \vdash \lambda f.f(f\,\mathtt{e}) : (\top \to \mathtt{o}) \wedge (\mathtt{o} \to \mathtt{o}) \to \mathtt{o} \Rightarrow \lambda f_{\top\to\mathtt{o}}.\lambda f_{\mathtt{o}\to\mathtt{o}}.\mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, f_{\top\to\mathtt{o}}}\,\textsc{Abs1}
$$

The following theorem states the correctness of the first transformation. A proof is given in Appendix A.

▶ **Theorem 7.** *Let $\mathcal{G}$ be an order-$(n+1)$ word grammar. If $\mathcal{G} \Rightarrow \mathcal{G}''$, then $\mathcal{G}''$ is an (extended) grammar of order at most n. Furthermore, $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}'')\!\uparrow_{\mathtt{e}}$.*

4 Step 2: removing dummy symbols

We now describe the second step for eliminating redundant symbols $\mathtt{e}$, which have been introduced by (Tr1-Abs2). By the remark at the end of Section 2, we assume that the result of the first transformation is an ordinary grammar, not containing extended terms. We also assume that $\mathtt{br}$ occurs only in the fully applied form. This does not lose generality, because otherwise we can replace $\mathtt{br}$ by a new non-terminal Br and add the rule $Br\,x\,y \to \mathtt{br}\,x\,y$.

The idea of the transformation is to use intersection types to distinguish between terms that generate trees consisting of only $\mathtt{br}$ and $\mathtt{e}$, and those that generate trees containing other arity-0 terminals. We assign the type $\mathtt{o}_\epsilon$ to the former terms, and $\mathtt{o}_+$ to the latter. A term $\mathtt{br}\,t_0\,t_1$ is transformed to (i) $\mathtt{br}\,t_0^{\#}\,t_1^{\#}$ if both t_0 and t_1 have type $\mathtt{o}_+$ (where $t_i^{\#}$ is the image of the transformation of t_i), (ii) $t_i^{\#}$ if t_i has type $\mathtt{o}_+$ and t_{1-i} has type $\mathtt{o}_\epsilon$, and (iii) $\mathtt{e}$ if both t_0 and t_1 have type $\mathtt{o}_\epsilon$. As in the transformation of the previous section, we replicate each non-terminal and variable for each intersection type. For example, the nonterminal $A : \mathtt{o} \to \mathtt{o}$ defined by $A\,x \to x$ would be replicated to $A_{\mathtt{o}_+ \to \mathtt{o}_+}$ and $A_{\mathtt{o}_\epsilon \to \mathtt{o}_\epsilon}$.

We first define the set of intersection types by:

$$\xi ::= \mathtt{o}_\epsilon \mid \mathtt{o}_+ \mid \xi_1 \wedge \cdots \wedge \xi_k \to \xi$$

We assume some total order $<$ on intersection types, and require that whenever we write $\xi_1 \wedge \cdots \wedge \xi_k$, $\xi_1 < \cdots < \xi_k$ holds. We define the refinement relation $\xi :: \kappa$ inductively by: (i) $\mathtt{o}_\epsilon :: \mathtt{o}$, (ii) $\mathtt{o}_+ :: \mathtt{o}$, and (iii) $(\xi_1 \wedge \cdots \wedge \xi_k \to \xi) :: (\kappa_1 \to \kappa_2)$ if $\xi :: \kappa_2$ and $\xi_i :: \kappa_1$ for every $i \in \{1, \ldots, k\}$. We consider only types ξ such that $\xi :: \kappa$ for some κ. For example, we forbid an ill-formed type like $\mathtt{o}_+ \wedge (\mathtt{o}_+ \to \mathtt{o}_+) \to \mathtt{o}_+$.

We introduce a type-based transformation relation $\Xi \vdash t : \xi \Rightarrow u$, where Ξ is a type environment (i.e., a set of bindings of the form $x : \xi$), t is a source term, ξ is the type of t, and u is the result of transformation. The relation is defined inductively by the rules below.

$$\frac{}{\Xi, x : \xi \vdash x : \xi \Rightarrow x_\xi}\ \text{(Tr2-Var)} \qquad \frac{}{\Xi \vdash \mathtt{e} : \mathtt{o}_\epsilon \Rightarrow \mathtt{e}}\ \text{(Tr2-Const0)} \qquad \frac{\Sigma(a) = 0 \qquad a \neq \mathtt{e}}{\Xi \vdash a : \mathtt{o}_+ \Rightarrow a}\ \text{(Tr2-Const1)}$$

$$\frac{\begin{array}{c}\Xi \vdash t_0 : \xi_0 \Rightarrow u_0 \qquad \Xi \vdash t_1 : \xi_1 \Rightarrow u_1 \\ (u, \xi) = \begin{cases} (\mathtt{br}\, u_0\, u_1, \mathtt{o}_+) & \text{if } \xi_0 = \xi_1 = \mathtt{o}_+ \\ (u_i, \mathtt{o}_+) & \text{if } \xi_i = \mathtt{o}_+ \text{ and } \xi_{1-i} = \mathtt{o}_\epsilon \\ (\mathtt{e}, \mathtt{o}_\epsilon) & \text{if } \xi_0 = \xi_1 = \mathtt{o}_\epsilon \end{cases}\end{array}}{\Xi \vdash \mathtt{br}\, t_0\, t_1 : \xi \Rightarrow u}\ \text{(Tr2-Const2)}$$

$$\frac{\xi :: \mathcal{N}(F) \qquad A\, x_1 \cdots x_k \to t \in \mathcal{R} \qquad \emptyset \vdash \lambda x_1. \cdots \lambda x_k.t : \xi \Rightarrow \lambda y_1. \cdots \lambda y_\ell.u}{\Xi \vdash A : \xi \Rightarrow A_\xi}\ \text{(Tr2-NT)}$$

$$\frac{\Xi \vdash s : \xi_1 \wedge \cdots \wedge \xi_k \to \xi \Rightarrow v \qquad \Xi \vdash t : \xi_i \Rightarrow U_i \text{ (for each } i \in \{1, \ldots, k\})}{\Xi \vdash st : \xi \Rightarrow vU_1 \cdots U_k}\ \text{(Tr2-App)}$$

$$\frac{\Xi \vdash t : \xi \Rightarrow u_i \text{ (for each } i \in \{1, \ldots, k\}) \qquad k \geq 1}{\Xi \vdash t : \xi \Rightarrow \{u_1, \ldots, u_k\}}\ \text{(Tr2-Set)}$$

$$\frac{\Xi, x : \xi_1, \ldots, x : \xi_k \vdash t : \xi \Rightarrow u}{\Xi \vdash \lambda x.t : \xi_1 \wedge \cdots \wedge \xi_k \to \xi \Rightarrow \lambda x_{\xi_1} \cdots \lambda x_{\xi_k}.u}\ \text{(Tr2-Abs)}$$

The transformation of rewriting rules and grammars is defined by:

$$\frac{\emptyset \vdash \lambda x_1. \cdots \lambda x_k.t : \xi \Rightarrow \lambda x'_1. \cdots \lambda x'_\ell.t' \qquad \xi :: \mathcal{N}(A)}{(A \to \lambda x_1. \cdots \lambda x_k.t) \Rightarrow (A_\xi \to \lambda x'_1. \cdots \lambda x'_\ell.t')}\ \text{(Tr2-Rule)}$$

$$\frac{\begin{array}{c}\mathcal{N}' = \{A_\xi : [\![\xi]\!] \mid \mathcal{N}(A) = \kappa \wedge \xi :: \kappa\} \\ \mathcal{R}' = \{r' \mid \exists r \in \mathcal{R}.r \Rightarrow r'\} \cup \{S' \to S_{\mathtt{o}_\epsilon}, S' \to S_{\mathtt{o}_+}\}\end{array}}{(\Sigma, \mathcal{N}, \mathcal{R}, S) \Rightarrow (\Sigma, \mathcal{N}', \mathcal{R}', S')}\ \text{(Tr2-Gram)}$$

Here, $[\![\xi]\!]$ is defined by:

$$[\![\mathtt{o}_\epsilon]\!] = [\![\mathtt{o}_+]\!] = \mathtt{o} \qquad [\![\, \xi_1 \wedge \cdots \wedge \xi_k \to \xi]\!] = [\![\xi_1]\!] \to \cdots \to [\![\xi_k]\!] \to [\![\xi]\!]$$

We explain some key rules. In (Tr2-Var) we replicate a variable for each type, as in the first transformation. The rules (Tr2-Const0) and (Tr2-Const1) are for nullary constants, which are mapped to themselves. We assign type $\mathtt{o}_\epsilon$ to $\mathtt{e}$ and $\mathtt{o}_+$ to the other constants. The rule (Tr2-Const2) is for the binary tree constructor $\mathtt{br}$. As explained above, we eliminate terms that generate empty trees (those consisting of only $\mathtt{br}$ and $\mathtt{e}$). For example, if $\xi_0 = \mathtt{o}_\epsilon$ and $\xi_1 = \mathtt{o}_+$, then t_0 may generate an empty tree; thus, the whole term is transformed to u_1.

The rule (Tr2-NT) replicates a terminal for each type, as in the case of variables. The middle and rightmost premises require that there is some body t of A that can indeed be transformed according to type ξ. Without this condition, for example, A defined by the rule $A \to A$ would be transformed to $A_{\mathtt{o}_\epsilon}$ by $\emptyset \vdash A : \mathtt{o}_\epsilon \Rightarrow A_{\mathtt{o}_\epsilon}$, but $A_{\mathtt{o}_\epsilon}$ diverges and does not produce an empty tree. That would make the rule (Tr2-Const2) unsound: when a source term is $\mathtt{br}\,A\,\mathtt{a}$, it would be transformed to $\mathtt{a}$, but while the original term does not generate a tree, the result of the transformation does. In short, the two premises are required to ensure that whenever $\emptyset \vdash t : \mathtt{o}_\epsilon \Rightarrow u$ holds, t can indeed generate an empty tree. In (Tr2-App), the argument is replicated for each type. Unlike in the transformation in the previous section, type environments can be shared among the premises, since linearity does not matter here. The other rules for terms are analogous to those in the first transformation.

In rule (Tr2-Gram) for grammars, we prepare a start symbol S' and add the rules $S' \to S_{\mathtt{o}_\epsilon}, S' \to S_{\mathtt{o}_+}$. We remark that the rewriting rule for $S_{\mathtt{o}_\epsilon}$ (resp. $S_{\mathtt{o}_+}$) is generated only if the original grammar generates an empty (resp. non-empty) tree. For example, in the extreme case where $\mathcal{R} = \{S \to S\}$, we have $\mathcal{R}' = \{S' \to S_{\mathtt{o}_\epsilon}, S' \to S_{\mathtt{o}_+}\}$, without any rules to rewrite $S_{\mathtt{o}_\epsilon}$ or $S_{\mathtt{o}_+}$.

▶ **Example 8.** Let us consider the grammar $\mathcal{G}_3 = (\Sigma, \mathcal{N}, \mathcal{R}, S)$ where $\mathcal{N} = \{S : \mathtt{o}, A : \mathtt{o} \to \mathtt{o}, B : \mathtt{o} \to \mathtt{o}, F : \mathtt{o} \to \mathtt{o}\}$, and $\mathcal{R}$ consists of:

$$\begin{array}{llll} S \to F\,\mathtt{a} \qquad S \to F\,\mathtt{b} & A\,f \to \mathtt{br}\,\mathtt{a}\,(\mathtt{br}\,f\,\mathtt{e}) & B\,f \to \mathtt{br}\,\mathtt{b}\,(\mathtt{br}\,f\,\mathtt{e}) \\ F\,f \to \mathtt{br}\,f\,(\mathtt{br}\,f\,\mathtt{e}) & F\,f \to F(A\,f) & F\,f \to F(B\,f) \end{array}$$

It is the same as the grammar obtained in Example 6, except that redundant subscripts on non-terminals and variables have been removed. The body of the rule for A is transformed as follows.

$$\dfrac{\dfrac{\dfrac{}{f:\mathtt{o}_+ \vdash \mathtt{a}:\mathtt{o}_+ \Rightarrow \mathtt{a}}\,\textsc{Const1} \qquad \dfrac{\dfrac{}{f:\mathtt{o}_+ \vdash f:\mathtt{o}_+ \Rightarrow f_{\mathtt{o}_+}}\,\textsc{Var} \qquad \dfrac{}{f:\mathtt{o}_+ \vdash \mathtt{e}:\mathtt{o}_\epsilon \Rightarrow \mathtt{e}}\,\textsc{Const0}}{f:\mathtt{o}_+ \vdash \mathtt{br}\,f\,\mathtt{e}:\mathtt{o}_+ \Rightarrow f_{\mathtt{o}_+}}\,\textsc{Const2}}{f:\mathtt{o}_+ \vdash \mathtt{br}\,\mathtt{a}\,(\mathtt{br}\,f\,\mathtt{e}):\mathtt{o}_+ \Rightarrow \mathtt{br}\,\mathtt{a}\,f_{\mathtt{o}_+}}\,\textsc{Const2}}{\emptyset \vdash \lambda f.\mathtt{br}\,\mathtt{a}\,(\mathtt{br}\,f\,\mathtt{e}):\mathtt{o}_+ \to \mathtt{o}_+ \Rightarrow \lambda f_{\mathtt{o}_+}.\mathtt{br}\,\mathtt{a}\,f_{\mathtt{o}_+}}\,\textsc{Abs}$$

The whole rules are transformed to:

$$\begin{array}{llll} S' \to S_{\mathtt{o}_+} & S' \to S_{\mathtt{o}_\epsilon} & S_{\mathtt{o}_+} \to F_{\mathtt{o}_+\to\mathtt{o}_+}\,\mathtt{a} & S_{\mathtt{o}_+} \to F_{\mathtt{o}_+\to\mathtt{o}_+}\,\mathtt{b} \\ A_{\mathtt{o}_+\to\mathtt{o}_+}\,f_{\mathtt{o}_+} \to \mathtt{br}\;\mathtt{a}\,f_{\mathtt{o}_+} & B_{\mathtt{o}_+\to\mathtt{o}_+}\,f_{\mathtt{o}_+} \to \mathtt{br}\;\mathtt{b}\,f_{\mathtt{o}_+} & F_{\mathtt{o}_+\to\mathtt{o}_+}\,f_{\mathtt{o}_+} \to \mathtt{br}\;f_{\mathtt{o}_+}\,f_{\mathtt{o}_+} & \\ F_{\mathtt{o}_+\to\mathtt{o}_+}\,f_{\mathtt{o}_+} \to F_{\mathtt{o}_+\to\mathtt{o}_+}(A_{\mathtt{o}_+\to\mathtt{o}_+}\,f_{\mathtt{o}_+}) & F_{\mathtt{o}_+\to\mathtt{o}_+}\,f_{\mathtt{o}_+} \to F_{\mathtt{o}_+\to\mathtt{o}_+}(B_{\mathtt{o}_+\to\mathtt{o}_+}\,f_{\mathtt{o}_+}) & & \end{array}$$

Here, we have omitted rules on non-terminals unreachable from S'.

If the rules for S in the source grammar were replaced by:

$$S \to F\,E \qquad E \to \mathtt{a} \qquad E \to \mathtt{b} \qquad E \to \mathtt{e},$$

then $F_{\mathtt{o}_\epsilon\to\mathtt{o}_\epsilon}$ and $F_{\mathtt{o}_\epsilon\wedge\mathtt{o}_+\to\mathtt{o}_+}$ would become reachable. Hence, the following rules generated from $F\,f \to \mathtt{br}\,f\,(\mathtt{br}\,f\,\mathtt{e})$ would also become reachable:

$$F_{\mathtt{o}_\epsilon\to\mathtt{o}_\epsilon}\,f_{\mathtt{o}_\epsilon} \to \mathtt{e} \qquad F_{\mathtt{o}_\epsilon\wedge\mathtt{o}_+\to\mathtt{o}_+}\,f_{\mathtt{o}_\epsilon}\,f_{\mathtt{o}_+} \to f_{\mathtt{o}_+}.$$

From $F\,f \to F\,(A\,f)$, many reachable rules would be generated. One of the rules is:

$$F_{\mathtt{o}_\epsilon \wedge \mathtt{o}_+ \to \mathtt{o}_+}\, f_{\mathtt{o}_\epsilon}\, f_{\mathtt{o}_+} \to F_{\mathtt{o}_+ \to \mathtt{o}_+}\{A_{\mathtt{o}_\epsilon \to \mathtt{o}_+}\, f_{\mathtt{o}_\epsilon}, A_{\mathtt{o}_+ \to \mathtt{o}_+}\, f_{\mathtt{o}_+}\},$$

which can be replaced by the following rules without extended terms:

$$F_{\mathtt{o}_\epsilon \wedge \mathtt{o}_+ \to \mathtt{o}_+}\, f_{\mathtt{o}_\epsilon}\, f_{\mathtt{o}_+} \to F_{\mathtt{o}_+ \to \mathtt{o}_+}(C\, f_{\mathtt{o}_\epsilon}\, f_{\mathtt{o}_+}) \qquad C\, f_1\, f_2 \to A_{\mathtt{o}_\epsilon \to \mathtt{o}_+}\, f_1 \qquad C\, f_1\, f_2 \to A_{\mathtt{o}_+ \to \mathtt{o}_+}\, f_2.$$

The following theorem claims the correctness of the transformation. The proof is given in Appendix B. The main theorem (Theorem 4) follows from Theorems 7, 9, and the fact that any order-m grammar with $m < n$ can be converted to an order-n grammar by adding a dummy non-terminal of order n.

▶ **Theorem 9.** *Let $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}, S)$ be an order-n tree grammar. If $\mathcal{G} \Rightarrow \mathcal{G}'$, then $\mathcal{G}'$ is a tree grammar of order at most n, and $\mathcal{L}_{\mathtt{leaf}}(\mathcal{G})\!\uparrow_{\mathtt{e}} = \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$.*

5 Applications

5.1 Unsafe order-2 word languages = safe order-2 word languages

As mentioned in Section 1, many of the earlier results on higher-order grammars [6, 10] were for the subclass called *safe* higher-order grammars. In safe grammars, the (simple) types of terms are restricted to *homogeneous types* [6] of the form $\kappa_1 \to \cdots \to \kappa_k \to \mathtt{o}$, where $\mathtt{order}(\kappa_1) \geq \cdots \geq \mathtt{order}(\kappa_k)$, and arguments of the same order must be supplied simultaneously. For example, if A has type $(\mathtt{o} \to \mathtt{o}) \to (\mathtt{o} \to \mathtt{o}) \to \mathtt{o}$, then the term $f\,(A\,f\,f)$ where $f : \mathtt{o} \to \mathtt{o}$ is valid, but $g\,(A\,f)$ where $g : ((\mathtt{o} \to \mathtt{o}) \to \mathtt{o}) \to \mathtt{o}, f : \mathtt{o} \to \mathtt{o}$ is not: the partial application $A\,f$ is disallowed, since A expects another order-1 argument. *Unsafe* grammars (which are just called higher-order grammars in the present paper) are higher-order grammars without the safety restriction.

For order-2 word languages, Aehlig et al. [1] have shown that the safety is not a genuine restriction. Our result in the present paper provides an alternative, short proof. Given an unsafe order-2 word grammar $\mathcal{G}$, we can obtain an equivalent order-1 grammar $\mathcal{G}'$ such that $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$. Note that $\mathcal{G}'$ is necessarily safe, since it is order-1 and hence there are no partial applications. Now, apply the backward transformation sketched in Section 2 to obtain an order-2 word grammar $\mathcal{G}''$ such that $\mathcal{L}_{\mathtt{w}}(\mathcal{G}'') = \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$. By the construction of the backward transformation, $\mathcal{G}''$ is clearly a safe grammar: Since the type of each term occurring in $\mathcal{G}'$ is $\mathtt{o} \to \cdots \to \mathtt{o} \to \mathtt{o}$, the type of the corresponding term of $\mathcal{G}''$ is $(\mathtt{o} \to \mathtt{o}) \to \cdots \to (\mathtt{o} \to \mathtt{o}) \to (\mathtt{o} \to \mathtt{o})$. Since all the arguments of type $\mathtt{o}$ are applied simultaneously in $\mathcal{G}'$, all the arguments of type $\mathtt{o} \to \mathtt{o}$ are also applied simultaneously in $\mathcal{G}''$. Thus, for any unsafe order-2 word grammar, there exists an equivalent safe order-2 word grammar.

5.2 Diagonal problem

The diagonal problem [5] asks, given a (word or tree) language L and a set S of symbols, whether for all n, there exists $w_n \in L$ such that $\forall a \in S. |w_n|_a \geq n$. Here, $|w|_a$ denotes the number of occurrences of a in w. A decision algorithm for the diagonal problem can be used for computing downward closures [22], which in turn have applications to program verification. Hague et al. [9] recently showed that the diagonal problem is decidable for safe higher-order languages, and Clemente et al. [4] extended the result for unsafe languages. Hague et al.'s proof roughly consists of the following two steps. First, one can use logical reflection [3] to reduce the diagonal problem for a safe order-n tree language $\mathcal{L}$ to that for the path language of a safe order-n tree language $\mathcal{L}'$ (which is an order-n word language). One can then reduce the latter to the diagonal problem for a safe order-$(n-1)$ tree language $\mathcal{L}''$. The first step immediately applies to the unsafe case, since the logical reflection technique is also available for unsafe languages. Our transformation can be used for the second step, yielding the decidability of the diagonal problem for unsafe languages. Actually, Clemente et al.'s recent proof follows the same line, except that a more specialized transformation is used for the second step; note that for the purpose of the diagonal problem, they only need to preserve the number of occurrences of symbols in each word, not the word itself (see Section 6 for more about this point).

5.3 Context-sensitivity of order-3 word languages

By using the result of this paper and the context-sensitivity of order-2 tree languages [13], we can prove that any order-3 word language is context-sensitive, i.e., the membership problem for an order-3 word language can be decided in non-deterministic linear space. Given an order-3 word grammar $\mathcal{G}$, we first construct a corresponding order-2 tree grammar $\mathcal{G}'$ in advance. Given a word w, we can construct a tree π whose frontier word is w one by one, and check whether $\pi \in \mathcal{L}(\mathcal{G}')$. Since the size of π is linearly bounded by the length $|w|$ of w, $\pi \overset{?}{\in} \mathcal{L}(\mathcal{G}')$ can be checked in space linear with respect to $|w|$. Thus, $w \in \mathcal{L}_{\mathtt{w}}(\mathcal{G})$ can be decided in non-deterministic linear space (with respect to the size of w).

6 Related Work

As already mentioned in Section 1, higher-order grammars have been extensively studied in 1980's [6, 7, 8], but most of those results have been for safe grammars. In particular, Damm [6] has shown an analogous result for safe grammars, but his proof does not extend to the unsafe case.

As also mentioned in Section 1, intersection types have been used in recent studies of (unsafe) higher-order grammars. In particular, type-based transformations of grammars and λ-terms have been studied in [14, 13, 4]. Clement et al. [4], independently from ours,[3] gave a transformation from an order-$(n+1)$ "narrow" tree language (which subsumes a word language as a special case) to an order-n tree language; this transformation preserves the number of occurrences of each symbol in each tree. When restricted to word languages, our result is stronger in that our transformation is guaranteed to preserve the order of symbols as well, and does not add any additional leaf symbols (though they are introduced in the intermediate step); consequently, our proofs are more involved. They use different intersection types, but the overall effect of their transformation seems similar to that of our first transformation. Thus, it may actually be the case that their transformation also preserves the order of symbols, although they have not proved so.

7 Conclusion

We have shown that for any unsafe order-$(n+1)$ word grammar $\mathcal{G}$, there exists an unsafe order-n tree grammar $\mathcal{G}'$ whose frontier language coincides with the word language $\mathcal{L}_{\mathtt{w}}(\mathcal{G})$.

[3] They cite our work as "On Unsafe Tree and Leaf Languages, in preparation".

The proof is constructive in that we provided (two-step) transformations that indeed construct $\mathcal{G}'$ from $\mathcal{G}$. The transformations are based on a combination of linear/non-linear intersection types, which may be interesting in its own right. As Damm [6] suggested, we expect the result to be useful for further studies of higher-order languages; in fact, we have discussed a few applications of the result.

Acknowledgments

We would like to thank Takeshi Tsukada for helpful discussions and thank Pawel Parys for information about the related work [4].

References

1 Klaus Aehlig, Jolie G. de Miranda, and C.-H. Luke Ong. Safety is not a restriction at level 2 for string languages. In *FoSSaCS*, volume 3441 of *LNCS*, pages 490–504. Springer, 2005.

2 William Blum and C.-H. Luke Ong. The safe lambda calculus. *Logical Methods in Computer Science*, 5(1), 2009.

3 Christopher H. Broadbent, Arnaud Carayol, C.-H. Luke Ong, and Olivier Serre. Recursion schemes and logical reflection. In *Proceedings ofo LICS 2010*, pages 120–129. IEEE Computer Society Press, 2010.

4 Lorenzo Clemente, Pawel Parys, Sylvain Salvati, and Igor Walukiewicz. The diagonal problem for higher-order recusion schemes is decidable. submitted to LICS, available from the second author's web page, 2016.

5 Wojciech Czerwinski and Wim Martens. A note on decidable separability by piecewise testable languages. *CoRR*, abs/1410.1042, 2014.

6 Werner Damm. The IO- and OI-hierarchies. *Theor. Comput. Sci.*, 20:95–207, 1982.

7 Joost Engelfriet. Iterated stack automata and complexity classes. *Info. Comput.*, 95(1):21–75, 1991.

8 Joost Engelfriet and Heiko Vogler. High level tree transducers and iterated pushdown tree transducers. *Acta Inf.*, 26(1/2):131–192, 1988.

9 Matthew Hague, Jonathan Kochems, and C.-H. Luke Ong. Unboundedness and downward closures of higher-order pushdown automata. In Rastislav Bodik and Rupak Majumdar, editors, *Proceedings of POPL 2016*, pages 151–163. ACM, 2016.

10 Teodor Knapik, Damian Niwinski, and Pawel Urzyczyn. Deciding monadic theories of hyperalgebraic trees. In *TLCA 2001*, volume 2044 of *LNCS*, pages 253–267. Springer, 2001.

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

12 Naoki Kobayashi. Pumping by typing. In *Proceedings of LICS 2013*, pages 398–407. IEEE Computer Society, 2013.

13 Naoki Kobayashi, Kazuhiro Inaba, and Takeshi Tsukada. Unsafe order-2 tree languages are context-sensitive. In *FoSSaCS*, volume 8412 of *Lecture Notes in Computer Science*, pages 149–163. Springer, 2014.

14 Naoki Kobayashi, Kazutaka Matsuda, Ayumi Shinohara, and Kazuya Yaguchi. Functional programs as compressed data. *Higher-Order and Symbolic Computation*, 2013.

15 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.

16 Gregory M. Kobele and Sylvain Salvati. The IO and OI hierarchies revisited. *Inf. Comput.*, 243:205–221, 2015.

17 C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *LICS 2006*, pages 81–90. IEEE Computer Society Press, 2006.

18 Pawel Parys. How many numbers can a lambda-term contain? In Michael Codish and Eijiro Sumii, editors, *Proceedings of FLOPS 2014*, volume 8475 of *Lecture Notes in Computer Science*, pages 302–318. Springer, 2014.

19 Sylvain Salvati and Igor Walukiewicz. Typing weak MSOL properties. In Andrew M. Pitts, editor, *Proceedings of FoSSaCS 2015*, volume 9034 of *Lecture Notes in Computer Science*, pages 343–357. Springer, 2015.

20 Thomas Streicher. *Domain-theoretic foundations of functional programming*. World Scientific, 2006.

21 Takeshi Tsukada and C.-H. Luke Ong. Compositional higher-order model checking via ω-regular games over böhm trees. In Thomas A. Henzinger and Dale Miller, editors, *Proceedings of CSL-LICS '14*, pages 78:1–78:10. ACM, 2014.

22 Georg Zetzsche. An approach to computing downward closures. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Proceedings of ICALP 2015*, volume 9135 of *Lecture Notes in Computer Science*, pages 440–451. Springer, 2015.

Appendix

A Proof of Theorem 7

We give a proof of Theorem 7 in Section A.1 after preparing some basic definitions. Lemmas for the proof are given after that. In Section A.1 we give basic lemmas. In Sections A.3 and A.4, we give main lemmas for forward and backward directions of the theorem, i.e., left-to-right and right-to-left simulations, respectively. The both lemmas need one key lemma, which is given in Section A.2.

Throughout this section, we often write $\mathtt{br}\, u_1\, u_2$ as $u_1 * u_2$. For s, $t_1, \dots, t_n$, we write an iterated application $(\cdots(s\, t_1)\, t_2 \cdots)\, t_n$ as $s\, \overrightarrow{t_i}^{i \le n}$. We also write $[t_1/x_1, \dots, t_k/x_k]$ as $[t_i/x_i]_{i \le k}$.

A.1 Proof of Theorem 7 and Basic Definitions and Lemmas

The extended terms can be embedded into the simply typed λY-calculus with non-determinism and the same constants as the terminal symbols (but without any non-terminals); we represent also the non-determinism in this λY-calculus by the set-representation $\{u_1, \dots, u_n\}$ $(n \ge 1)$. The embedding transformation is given in the standard way: the mutual recursion allowed in a grammar is handled by using Bekič property of Y-combinator. Also for this λY-calculus, we consider call-by-name reduction. We call terms in this calculus simply *λY-terms*, which are also ranged over by u and v; but if we use u and v without mentioning where they range, they are meant to be extended applicative terms for a given grammar. Through this transformation, we identify extended terms in a grammar with the embedded λY-terms.

We define $\mathtt{e}$-*observational preorder* $\lesssim$ and $\mathtt{e}$-*observational equivalence* $\sim$ as follows. First we define $\sim_{\mathrm{v}}$ for trees as the least congruence (w.r.t. the definition of trees) satisfying $\pi \sim_{\mathrm{v}} \mathtt{e} * \pi$ and $\pi_1 * (\pi_2 * \pi_3) \sim_{\mathrm{v}} (\pi_1 * \pi_2) * \pi_3$. Now, for two λY-terms

$$x_1 : \kappa_1, \dots, x_n : \kappa_n \vdash u, u' : \kappa$$

we define $u \lesssim u'$ if, for any λY-term $C : (\kappa_1 \to \cdots \to \kappa_n \to \kappa) \to \mathtt{o}$ and for any tree π such that $C(\lambda x_1.\cdots\lambda x_n.u) \longrightarrow^* \pi$, there exists π' such that $C(\lambda x_1.\cdots\lambda x_n.u') \longrightarrow^* \pi'$ and $\pi \sim_{\mathrm{v}} \pi'$. And we define $u \sim u'$ if $u \lesssim u'$ and $u \gtrsim u'$.

We define the set $\mathbf{FV}(u)$ of *free variables* of an extended term u as follows:

$$\begin{aligned}
\mathbf{FV}(x) &:= \{x\}\\
\mathbf{FV}(a) &:= \emptyset\\
\mathbf{FV}(A) &:= \emptyset\\
\mathbf{FV}(u\, U) &:= \mathbf{FV}(u) \cup \mathbf{FV}(U)\\
\mathbf{FV}(\{u_1, \dots, u_k\}) &:= \cup_{i \le k} \mathbf{FV}(u_i)
\end{aligned}$$

For a word $a_1 \cdots a_n$, we define term $(a_1 \cdots a_n)^\star$ inductively by: $\epsilon^\star = \mathtt{e}$ and $(as)^\star = \mathtt{br}\, a\, s^\star$.

We write $\Gamma \vdash_{\mathrm{s}} t : \delta \Rightarrow u$ if the judgement is derived by using the following restricted rule instead of (Tr1-Set).

$$\frac{\Gamma \vdash t : \delta \Rightarrow u_i \text{ (for each } i \in \{1, \dots, k\}) \qquad k \ge 1 \qquad k = 1 \text{ if } \delta \text{ is unbalanced}}{\Gamma \vdash t : \delta \Rightarrow \{u_1, \dots, u_k\}} \quad \text{(Tr1-SetS)}$$

Clearly, if $\Gamma \vdash_{\mathrm{s}} t : \delta \Rightarrow u$ then $\Gamma \vdash t : \delta \Rightarrow u$. We use this restriction in the proof of the forward direction of the theorem.

Now we prove Theorem 7, whose statement is: Let $\mathcal{G}$ be an order-$(n+1)$ word grammar. If $\mathcal{G} \Rightarrow \mathcal{G}''$, then $\mathcal{G}''$ is an (extended) grammar of order at most n. Furthermore, $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}'')\!\uparrow_{\mathtt{e}}$.

Proof of Theorem 7. The well-typedness of the right hand side term of every rewriting rule of $\mathcal{G}''$ can be proved straightforwardly (in a way similar to Lemma 22 in Section A.2). By induction on κ, we can show that $\mathtt{order}(\llbracket \delta :: \kappa \rrbracket) \leq \mathtt{order}(\kappa) - 1$ if $\mathtt{order}(\kappa) \geq 1$ and $\mathtt{order}(\llbracket \delta :: \kappa \rrbracket) = \mathtt{order}(\kappa) = 0$ otherwise.

Now we show $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}'')\!\uparrow_{\mathtt{e}}$. Suppose $a_1 \cdots a_n \in \mathcal{L}_{\mathtt{w}}(\mathcal{G})$, i.e., $S \longrightarrow^*_{\mathcal{G}} a_1(\cdots(a_n \mathtt{e})\cdots)$. By Lemma 10, we have $\vdash_{\mathrm{s}} a_1(\cdots(a_n \mathtt{e})\cdots) : \mathtt{o} \Rightarrow (a_1 \cdots a_n)^\star$. By Lemma 27, we have u such that $\vdash_{\mathrm{s}} S : \mathtt{o} \Rightarrow u$ with $u(\longrightarrow_{\mathcal{G}''}\gtrsim)^*(a_1 \cdots a_n)^\star$. By the transformation rule, u must be $S_{\mathtt{o}}$. Thus, we have $S_{\mathtt{o}}(\longrightarrow_{\mathcal{G}''}\gtrsim)^*(a_1 \cdots a_n)^\star$, which implies $a_1 \cdots a_n \in \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}'')\!\uparrow_{\mathtt{e}}$ as required.

Conversely, suppose $a_1 \cdots a_n \in \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}'')\!\uparrow_{\mathtt{e}}$, i.e., $S_{\mathtt{o}} \longrightarrow^*_{\mathcal{G}''} \pi$ with $\mathbf{leaves}(\pi)\!\uparrow_{\mathtt{e}} = a_1 \cdots a_n$ for some π. By repeating Lemma 29, we have $S \longrightarrow^*_{\mathcal{G}} s$ and $\vdash s : \mathtt{o} \Rightarrow \pi'$ with $\pi' \sim_{\mathrm{v}} \pi$. By Lemma 11, $s = a_1(\cdots(a_n \mathtt{e})\cdots)$. Thus, we have $a_1 \cdots a_n \in \mathcal{L}_{\mathtt{w}}(\mathcal{G})$ as required. ☐

▶ **Lemma 10.** $\vdash_{\mathrm{s}} a_1(\cdots(a_n\, \mathtt{e})\cdots) : \mathtt{o} \Rightarrow (a_1 \cdots a_n)^\star$.

Proof. This follows by straightforward induction on n. ☐

▶ **Lemma 11.** *Let t be an applicative term. If $\vdash t : \mathtt{o} \Rightarrow \pi$ then $t = a_1(\cdots(a_n\, \mathtt{e})\cdots)$ with $(a_1 \cdots a_n)^\star = \pi$.*

Proof. This follows by induction on the structure of π.

- Case $\pi = \mathtt{e}$: $\vdash t : \mathtt{o} \Rightarrow \pi$ must have been derived by using (Tr1-Const0). Therefore $t = \mathtt{e}$ as required.
- Case $\pi = \mathtt{br}\, \pi_1\, \pi_2$: $\vdash t : \mathtt{o} \Rightarrow \pi$ must have been derived by using (Tr1-App2). Thus, we have:

  $$t = t_1 t_2 \qquad \vdash t_1 : \mathtt{o} \to \mathtt{o} \Rightarrow \pi_1 \qquad \vdash t_2 : \mathtt{o} \Rightarrow \pi_2 \qquad \pi = \mathtt{br}\, \pi_1\, \pi_2$$

  By the condition $\vdash t_1 : \mathtt{o} \to \mathtt{o} \Rightarrow \pi_1$, the head symbol of t_1 must be a terminal. (Because the type environment is empty, the head cannot be a variable, and because the output of transformation does not contain a non-terminal, the head cannot be a non-terminal.) Thus, t_1 is actually a terminal a_1. By the induction hypothesis and $\vdash t_2 : \mathtt{o} \Rightarrow \pi_2$, we have $t_2 = a_2(\cdots(a_n\, \mathtt{e})\cdots)$ with $(a_2 \cdots a_n)^\star = \pi_2$. Thus, we have $t = a_1(a_2(\cdots(a_n\, \mathtt{e})))$, with $(a_1 a_2 \cdots a_n)^\star = \pi$ as required.

☐

▶ **Lemma 12** (Context Lemma). *Given two λY-terms $(x_1 : \kappa_1, \ldots, x_n : \kappa_n \vdash u, u' : \kappa)$ where $\kappa = \kappa_{n+1} \to \cdots \to \kappa_\ell \to \mathtt{o}$, we have $u \lesssim u'$ iff for any closed terms $U_1, \ldots, U_\ell$ of type $\kappa_1, \ldots, \kappa_\ell$, respectively, and for any π such that $(\lambda x_1 \ldots . \lambda x_n . u)\, \overrightarrow{U_i}^{i \leq \ell} \longrightarrow^* \pi$, there exists π' such that $(\lambda x_1 \ldots . \lambda x_n . u')\, \overrightarrow{U_i}^{i \leq \ell} \longrightarrow^* \pi'$ and $\pi \sim_{\mathrm{v}} \pi'$. (We write $u \sqsubseteq u'$ if the latter condition of this equivalence holds.)*

Proof. The proof is obtained by a trivial modification of the proof of the context lemma for PCF by a logical relation given in [20].

The logical relation is between a cpo model and the syntax. The cpo model is the standard (call-by-name) cpo model extended with Hoare powerdomain, which corresponds to may convergence. Specifically, the interpretation $[\![\mathtt{o}]\!]$ of the base type $\mathtt{o}$ is defined as $(P(\mathbb{V}), \subseteq)$ where $\mathbb{V}$ is the quotient set of the set of trees modulo $\sim_{\mathrm{v}}$, and $P(\mathbb{V})$ is the powerset of $\mathbb{V}$. (This is the Hoare powerdomain of the flat cpo $\mathbb{V}_{\perp}$.) The interpretation of function types is given by the usual continuous function spaces. The interpretation of the constants is given as follows:

$$[\![\mathtt{br}]\!](L_1, L_2) := \{[\mathtt{br}\,\pi_1\,\pi_2]_{\sim_{\mathrm{v}}} \mid [\pi_i]_{\sim_{\mathrm{v}}} \in L_i\} \qquad (L_1, L_2 \in P(\mathbb{V}))$$
$$[\![a]\!] := \{[a]_{\sim_{\mathrm{v}}}\} \qquad (\Sigma(a) = 0).$$

Now the logical relation $R = (R_\kappa)_\kappa$ is defined as below. Let Term_κ be the set of closed λY-terms of sort κ. Then $R_\kappa \subseteq [\![\kappa]\!] \times \mathrm{Term}_\kappa$ is defined inductively as follows:

$$L\,R_{\mathtt{o}}\,u \quad \text{if} \quad \text{for any } d \in L \text{ there exists } \pi \text{ such that } u \longrightarrow^* \pi \text{ and } d = [\pi]_{\sim_{\mathrm{v}}}$$
$$f\,R_{\kappa\to\kappa'}\,u \quad \text{if} \quad \text{for any } g \in [\![\kappa]\!] \text{ and } v \in \mathrm{Term}_\kappa,\ g\,R_\kappa\,v \text{ implies } f(g)\,R_{\kappa'}\,(u\,v).$$

For $u, u' \in \mathrm{Term}_\kappa$, we can show that

$$u \lesssim u' \quad\Longrightarrow\quad u \sqsubseteq u' \quad\Longrightarrow\quad [\![u]\!]\,R_\kappa\,u' \quad\Longrightarrow\quad u \lesssim u'$$

whose proof is obtained in the same way as that of [20, Theorem 5.1]. ☐

▶ **Lemma 13.** *Given $\Gamma, x : \delta_1, \ldots, x : \delta_k \vdash t : \delta \Rightarrow u$ where $x \notin dom(\Gamma)$, $\delta_1 \wedge \cdots \wedge \delta_k \to \delta$ is well-formed.*

Proof. By straightforward induction on t. ☐

▶ **Lemma 14.** *Given $\Gamma \vdash t : \delta \Rightarrow u$ and $y \in \mathbf{FV}(u)$ there exists $x : \delta' \in \Gamma$ such that $y = x_{\delta'}$.*

Proof. By straightforward induction on t. ☐

▶ **Lemma 15.** *1. For any u, u_1, u_2, and u_3,*

$$u \lesssim \mathtt{e} * u \qquad \textit{and} \qquad u_1 * (u_2 * u_3) \sim (u_1 * u_2) * u_3\,.$$

2. For any π_1 and π_2,

$$\pi_1 \sim \pi_2 \qquad \textit{iff} \qquad \pi_1 \sim_{\mathrm{v}} \pi_2\,.$$

Proof. The both items can be easily shown by using the context lemma. ☐

▶ **Lemma 16.** *If $u \lesssim u'$, then $\theta u \lesssim \theta u'$.*

Proof. The proof is trivial from the definition of the contextual preorder $\lesssim$. ☐

▶ **Lemma 17.**

If $dom(\theta) \cap dom(\theta') = \emptyset$, then $\theta(\theta' u) = (\theta \cup \theta')u$.

Proof. The proof is given by straightforward induction on u. ☐

▶ **Lemma 18.** *Given $\Gamma, x : \delta' \vdash_{\mathrm{s}} t : \delta \Rightarrow u$, if $x \notin \mathbf{FV}(t)$, then we also have $\Gamma \vdash_{\mathrm{s}} t : \delta \Rightarrow u$ and δ' is balanced.*

Proof. This follows by straightforward induction on $\Gamma, x : \delta' \vdash_{\mathrm{s}} t : \delta \Rightarrow u$. ☐

A.2 Key Lemma

▶ **Lemma 19.** *Given* $x : \mathtt{o} \vdash s : \delta \Rightarrow v$ *where* $\mathtt{order}(\delta) \le 1$ *and* $\vdash t : \mathtt{o} \Rightarrow U$*,*

$$([\mathsf{e}/x_{\mathsf{o}}]v) * U \sim [U/x_{\mathsf{o}}]v\,.$$

Moreover, for any $p \ge 0$*,* π*, and a reduction sequence*

$$([\mathsf{e}/x_{\mathsf{o}}]v) * U \longrightarrow^{p} \pi$$

there exists π' *such that*

$$[U/x_{\mathsf{o}}]v \longrightarrow^{p} \pi' \sim_{\mathrm{v}} \pi\,.$$

The above lemma is the key of the proof of Theorem 7, and says that the variable x_{o} occurs at the rightmost position in (the trees of) v. For the proof of this lemma, we introduce a type system for the transformed grammar $\mathcal{G}''$. The set of types is given by the following grammar.

$$\rho ::= \mathtt{o} \mid \mathtt{oR} \mid \rho \to \rho$$

Intuitively, $\mathtt{oR}$ is the type of trees that can occur only at the rightmost position of a tree while $\mathtt{o}$ is the type of trees without any such restriction; for example, if t has type $\mathtt{oR}$ and t' has type $\mathtt{o}$, then $t' * t$ is valid but $t * t'$ is not.

We define a notion of balance/unbalance, which is similar to that for the types δ:

$$\frac{}{\mathtt{o}\text{ is balanced}} \qquad \frac{}{\mathtt{oR}\text{ is unbalanced}} \qquad \frac{\rho\text{ is balanced}\quad \rho'\text{ is balanced}}{\rho \to \rho'\text{ is balanced}}$$

$$\frac{\rho\text{ is unbalanced}\quad \rho'\text{ is unbalanced}}{\rho \to \rho'\text{ is balanced}} \qquad \frac{\rho\text{ is balanced}\quad \rho'\text{ is unbalanced}}{\rho \to \rho'\text{ is unbalanced}}$$

A type ρ is *well-formed* if it is either balanced or unbalanced. We assume that all the types occurring below are well-formed.

A type environment Φ is a set of type bindings of the form $x : \rho$. We write $\mathbf{bal}(\Phi)$ and say Φ is *balanced* if ρ is balanced for every $x : \rho \in \Phi$. As before, we treat unbalanced types as linear types, i.e., the union $\Phi_1 \cup \Phi_2$ of Φ_1 and Φ_2 is defined only if $\mathbf{bal}(\Phi_1 \cup \Phi_2)$.

We define three type transformations $(-)^{\sharp}$, $(-)^{\flat}$, and $(-)^{\sharp}_{\mathtt{o}}$ as follows:

$$\begin{aligned}
(\delta)^{\sharp} &:= \mathtt{oR} && (\mathtt{order}(\delta) \le 1,\ \delta\text{ is unbalanced})\\
(\delta)^{\sharp} &:= \mathtt{o} && (\mathtt{order}(\delta) \le 1,\ \delta\text{ is balanced})\\
(\wedge_{i\le k}\delta_i \to \delta)^{\sharp} &:= (\delta_1)^{\sharp} \to \ldots \to (\delta_k)^{\sharp} \to (\delta)^{\sharp} && (\mathtt{order}(\wedge_{i\le k}\delta_i \to \delta) \ge 2)\\
(x_1 : \delta_1, \ldots, x_n : \delta_n)^{\sharp} &:= \big((x_1)_{\delta_1} : (\delta_1)^{\sharp}, \ldots, (x_n)_{\delta_n} : (\delta_n)^{\sharp}\big)\\
(\mathtt{o})^{\flat} &:= \mathtt{o}\\
(\mathtt{oR})^{\flat} &:= \mathtt{o}\\
(\rho \to \rho')^{\flat} &:= (\rho)^{\flat} \to (\rho')^{\flat}\\
(x_1 : \rho_1, \ldots, x_n : \rho_n)^{\flat} &:= \big(x_1 : (\rho_1)^{\flat}, \ldots, x_n : (\rho_n)^{\flat}\big)\\
(\delta)^{\sharp}_{\mathtt{o}} &:= ((\delta)^{\sharp})^{\flat}\\
(\Gamma)^{\sharp}_{\mathtt{o}} &:= ((\Gamma)^{\sharp})^{\flat}
\end{aligned}$$

It is obvious that, if δ is balanced (resp. unbalanced), then $(\delta)^\sharp$ is balanced (resp. unbalanced).

Then the typing rules are given as follows:

$$\frac{\mathbf{bal}(\Phi)}{\Phi, x : \rho \vdash x : \rho} \qquad \text{(RTy-Var)}$$

$$\frac{\mathbf{bal}(\Phi) \qquad \Sigma(a) = 1 \text{ in } \mathcal{G}}{\Phi \vdash a : \mathtt{o}} \qquad \text{(RTy-Alph)}$$

$$\frac{\mathbf{bal}(\Phi)}{\Phi \vdash \mathtt{br} : \mathtt{o} \to \mathtt{o} \to \mathtt{o}} \ \text{(RTy-BrAll)} \qquad \frac{\mathbf{bal}(\Phi)}{\Phi \vdash \mathtt{br} : \mathtt{o} \to \mathtt{oR} \to \mathtt{oR}} \ \text{(RTy-BrRight)}$$

$$\frac{\mathbf{bal}(\Phi)}{\Phi \vdash \mathtt{e} : \mathtt{o}} \ \text{(RTy-EpsAll)} \qquad \frac{\mathbf{bal}(\Phi)}{\Phi \vdash \mathtt{e} : \mathtt{oR}} \ \text{(RTy-EpsRight)}$$

$$\frac{\mathbf{bal}(\Phi)}{\Phi \vdash A_\delta : (\delta)^\sharp_{\mathtt{o}}} \ \text{(RTy-NtAll)} \qquad \frac{\mathbf{bal}(\Phi)}{\Phi \vdash A_\delta : (\delta)^\sharp} \ \text{(RTy-NtRight)}$$

$$\frac{\Phi_0 \vdash v : \rho_1 \to \rho \qquad \Phi_1 \vdash U : \rho_1}{\Phi_0 \cup \Phi_1 \vdash v\,U : \rho} \qquad \text{(RTy-App)}$$

$$\frac{\Phi \vdash u_i : \rho \quad (\text{for each } i \leq k)}{\Phi \vdash \{u_1, \dots, u_k\} : \rho} \qquad \text{(RTy-Set)}$$

$$\frac{\Phi, x : \rho' \vdash u : \rho}{\Phi \vdash \lambda x.u : \rho' \to \rho} \qquad \text{(RTy-Abs)}$$

We prepare some lemmas for proving Lemma 19.

▶ **Lemma 20.** *If $\Phi, x : \rho \vdash u : \rho'$, then $\rho \to \rho'$ is well-formed.*

Proof. This follows by straightforward induction on the derivation $\Phi, x : \rho \vdash u : \rho'$. ☐

▶ **Lemma 21** (substitution). *Given $\Phi, x' : \rho' \vdash v : \rho$ and $\Phi' \vdash U : \rho'$, we have $\Phi \cup \Phi' \vdash [U/x']v : \rho$.*

Proof. The proof is given by induction on v. The base case is clear. The remaining case is application: we have rule (RTy-App)

$$\frac{\Phi_0 \vdash v' : \rho_1 \to \rho \qquad \Phi_1 \vdash U' : \rho_1}{\Phi_0 \cup \Phi_1 \vdash v'\,U' : \rho}$$

where

$$\Phi, x' : \rho' = \Phi_0 \cup \Phi_1 \qquad v = v'\,U'\,.$$

Further we have (RTy-Set)

$$\frac{\Phi_1 \vdash u'_i : \rho_1 \quad (\text{for each } i \in \{1, \dots, k\})}{\Phi_1 \vdash \{u'_1, \dots, u'_k\} : \rho_1}$$

where $U' = \{u'_1, \dots, u'_k\}$.

Now we perform a case analysis on whether ρ' is balanced or unbalanced.

- Case where ρ' is balanced: In this case, Φ' is balanced. By the induction hypotheses, we have

$$(\Phi_0\setminus\{x':\rho'\})\cup\Phi' \vdash [U/x']v' : \rho_1 \to \rho \qquad (\Phi_1\setminus\{x':\rho'\})\cup\Phi' \vdash [U/x']u'_i : \rho_1 \quad (\text{for each } i \leq k)\,.$$

and by (RTy-Set),

$$\frac{(\Phi_0 \setminus \{x' : \rho'\}) \cup \Phi' \vdash v'_j : \rho_1 \to \rho \quad (\text{for each } j \in \{1, \dots, k_0\})}{(\Phi_0 \setminus \{x' : \rho'\}) \cup \Phi' \vdash \{v'_1, \dots, v'_{k_0}\} : \rho_1 \to \rho} \qquad \{v'_1, \dots, v'_{k_0}\} = [U/x']v'$$

$$\frac{(\Phi_1 \setminus \{x' : \rho'\}) \cup \Phi' \vdash u'^i_j : \rho_1 \quad (\text{for each } j \in \{1, \dots, k_i\})}{(\Phi_1 \setminus \{x' : \rho'\}) \cup \Phi' \vdash \{u'^i_1, \dots, u'^i_{k_i}\} : \rho_1} \qquad \{u'^i_1, \dots, u'^i_{k_i}\} = [U/x']u'_i$$

Then, by (RTy-Set)

$$(\Phi_1 \setminus \{x' : \rho'\}) \cup \Phi' \vdash [U/x']U' : \rho_1$$

and by (RTy-Set) and (RTy-App), we have

$$(\Phi_0 \setminus \{x' : \rho'\}) \cup \Phi' \cup (\Phi_1 \setminus \{x' : \rho'\}) \cup \Phi' \vdash ([U/x']v')([U/x']U') : \rho$$

where the linearity condition is obvious, since Φ' is balanced and

$$(\Phi_0 \setminus \{x' : \rho'\}) \cap (\Phi_1 \setminus \{x' : \rho'\}) \subseteq \Phi_0 \cap \Phi_1 \subseteq (\text{the set of balanced bindings})\,.$$

- Case where ρ' is unbalanced and $x' : \rho' \in \Phi_1$: By the induction hypotheses, we have

$$(\Phi_1 \setminus \{x' : \rho'\}) \cup \Phi' \vdash [U/x']u'_i : \rho_1 \quad (\text{for each } i \leq k)$$

and by (RTy-Set), similarly to the previous case, we have

$$(\Phi_1 \setminus \{x' : \rho'\}) \cup \Phi' \vdash [U/x']U' : \rho_1\,.$$

Then by (RTy-App), we have

$$\Phi_0 \cup (\Phi_1 \setminus \{x' : \rho'\}) \cup \Phi' \vdash v'([U/x']U') : \rho$$

as required; here the linearity condition holds as follows: Since ρ' is unbalanced, Φ is balanced. Now $x' : \rho' \in \Phi_1$, and therefore Φ_0 and $\Phi_1 \setminus \{x' : \rho'\}$ are balanced.
- Case where ρ' is unbalanced and $x' : \rho' \in \Phi_0$: By the induction hypothesis, we have

$$(\Phi_0 \setminus \{x' : \rho'\}) \cup \Phi' \vdash [U/x']v' : \rho_1 \to \rho\,.$$

Then by (RTy-App), we have

$$(\Phi_0 \setminus \{x' : \rho'\}) \cup \Phi' \cup \Phi_1 \vdash ([U/x']v')U' : \rho$$

as required; here the linearity condition holds since $\Phi_0 \setminus \{x' : \rho'\}$ and Φ_1 are balanced (similarly to the previous case).

□

▶ **Lemma 22.** *For any $\Gamma \vdash s : \delta \Rightarrow v$, we have $(\Gamma)^\sharp \vdash v : (\delta)^\sharp$.*

Proof. The proof proceeds by straightforward induction on the derivation $\Gamma \vdash s : \delta \Rightarrow v$. Note that, since if δ is balanced so is $(\delta)^\sharp$, $\mathbf{bal}(\Gamma)$ implies $\mathbf{bal}((\Gamma)^\sharp)$.

- Case of (Tr1-Var):

$$\frac{\mathbf{bal}(\Gamma)}{\Gamma, x : \delta \vdash x : \delta \Rightarrow x_\delta}$$

The goal:

$$(\Gamma)^\sharp, x_\delta : (\delta)^\sharp \vdash x_\delta : (\delta)^\sharp$$

is obtained by (RTy-Var).

- Case of (Tr1-Const0):

$$\frac{\mathbf{bal}(\Gamma)}{\Gamma \vdash \mathtt{e} : \mathtt{o} \Rightarrow \mathtt{e}}$$

The goal:

$$(\Gamma)^\sharp \vdash \mathtt{e} : \mathtt{oR}$$

is obtained by (RTy-EpsRight).

- Case of (Tr1-Const1):

$$\frac{\mathbf{bal}(\Gamma) \qquad \Sigma(a) = 1}{\Gamma \vdash a : \mathtt{o} \to \mathtt{o} \Rightarrow a}$$

The goal:

$$(\Gamma)^\sharp \vdash a : \mathtt{o}$$

is obtained by (RTy-Alph).

- Case of (Tr1-NT):

$$\frac{\mathbf{bal}(\Gamma)}{\Gamma \vdash A : \delta \Rightarrow A_\delta}$$

The goal:

$$(\Gamma)^\sharp \vdash A_\delta : (\delta)^\sharp$$

is obtained by (RTy-NtRight).

- Case of (Tr1-App1):

$$\frac{\begin{array}{c}\Gamma_0 \vdash s : \delta_1 \wedge \cdots \wedge \delta_k \to \delta \Rightarrow v \\ \Gamma_i \vdash t : \delta_i \Rightarrow U_i \text{ and } \delta_i \neq \mathtt{o} \text{ (for each } i \in \{1, \ldots, k\})\end{array}}{\Gamma_0 \cup \Gamma_1 \cup \cdots \cup \Gamma_k \vdash st : \delta \Rightarrow vU_1 \cdots U_k}$$

The induction hypotheses are

$$(\Gamma_0)^\sharp \vdash v : (\delta_1)^\sharp \to \cdots \to (\delta_k)^\sharp \to (\delta)^\sharp$$
$$(\Gamma_i)^\sharp \vdash U_i : (\delta_i)^\sharp \quad \text{(for each } i \in \{1, \ldots, k\})$$

where the latter are obtained through (Tr1-Set) and (RTy-Set). The goal:

$$(\Gamma_0)^\sharp \cup (\Gamma_1)^\sharp \cup \cdots \cup (\Gamma_k)^\sharp \vdash vU_1 \cdots U_k : (\delta)^\sharp$$

is obtained by (RTy-App).

- Case of (Tr1-App2):

$$\frac{\Gamma_0 \vdash s : \mathtt{o} \to \delta \Rightarrow V \qquad \Gamma_1 \vdash t : \mathtt{o} \Rightarrow U}{\Gamma_0 \cup \Gamma_1 \vdash st : \delta \Rightarrow \mathtt{br}\, V\, U}$$

  By the well-formedness, δ is unbalanced. Hence, the induction hypotheses are

$$(\Gamma_0)^\sharp \vdash V : \mathtt{o}$$
$$(\Gamma_1)^\sharp \vdash U : \mathtt{oR}\,.$$

  The goal:

$$(\Gamma_0)^\sharp \cup (\Gamma_1)^\sharp \vdash \mathtt{br}\, V\, U : \mathtt{oR}$$

  is obtained by (RTy-App) and (RTy-BrRight).
- Case of (Tr1-Set):

$$\frac{\Gamma \vdash t : \delta \Rightarrow u_i \text{ (for each } i \in \{1,\dots,k\}) \qquad k \geq 1}{\Gamma \vdash t : \delta \Rightarrow \{u_1,\dots,u_k\}}$$

  The induction hypotheses are

$$(\Gamma)^\sharp \vdash u_i : (\delta)^\sharp \qquad (i \in \{1,\dots,k\})\,.$$

  The goal:

$$(\Gamma)^\sharp \vdash \{u_1,\dots,u_k\} : (\delta)^\sharp$$

  is obtained by (RTy-Set).

□

▶ **Lemma 23.** ***1.*** *Given $\Phi \vdash u : \rho$, we have $(\Phi)^\flat \vdash u : (\rho)^\flat$.*
2. *Given $\Gamma \vdash t : \delta \Rightarrow u$, we have $(\Gamma)^\sharp_{\mathtt{o}} \vdash u : (\delta)^\sharp_{\mathtt{o}}$.*
3. *Given $\Gamma, x : \mathtt{o} \vdash t : \delta \Rightarrow u$, we have $(\Gamma)^\sharp_{\mathtt{o}} \vdash [\mathtt{e}/x_\mathtt{o}]u : (\delta)^\sharp_{\mathtt{o}}$.*

Proof. Case of item 1: The proof is given by straightforward induction on u; the base case is trivial, since for every terminal and non-terminal there is a typing rule for having a type of the form $(\delta)^\sharp_{\mathtt{o}}$. In the case of application, we have rule (RTy-App):

$$\frac{\Phi_0 \vdash v : \rho_1 \to \rho \qquad \Phi_1 \vdash U : \rho_1}{\Phi_0 \cup \Phi_1 \vdash v\, U : \rho}$$

This case is also clear by the induction hypotheses.

Case of item 2: By Lemma 22, $(\Gamma)^\sharp \vdash u : (\delta)^\sharp$. By item 1, we have $(\Gamma)^\sharp_{\mathtt{o}} \vdash u : (\delta)^\sharp_{\mathtt{o}}$.

Case of item 3: By item 2, we have $(\Gamma)^\sharp_{\mathtt{o}}, x_\mathtt{o} : \mathtt{o} \vdash u : (\delta)^\sharp_{\mathtt{o}}$. Since $\vdash \mathtt{e} : \mathtt{o}$, by Lemma 21, we have $(\Gamma)^\sharp_{\mathtt{o}} \vdash [\mathtt{e}/x_\mathtt{o}]u : (\delta)^\sharp_{\mathtt{o}}$. □

▶ **Lemma 24** (subject reduction)**.** *Given a reduction $u \longrightarrow u'$,*

1. *if $x_\mathtt{o} : \mathtt{oR} \vdash u : \mathtt{oR}$ then $x_\mathtt{o} : \mathtt{oR} \vdash u' : \mathtt{oR}$, and*
2. *if $\vdash u : \mathtt{o}$ then $\vdash u' : \mathtt{o}$.*

Proof. The proof is given by induction on u simultaneously for the both items. Since $u \longrightarrow u'$, the head of u is either $\mathtt{br}$ or a non-terminal.

Case where the head of u is $\mathtt{br}$: Let $u = \mathtt{br}\,U_1\,U_2$. When U_1 is reduced, the case that U_1 is not a singleton is clear, since in the rule (RTy-Set), the type parts and the environment parts of judgments are common. Suppose $U_1 = \{u_1\}$ and $u_1 \longrightarrow u_1'$ and $u' = \mathtt{br}\,u_1'\,U_2$. First we consider item 1. For $(x_{\mathtt{o}} : \mathtt{oR} \vdash \mathtt{br}\,u_1\,U_2 : \mathtt{oR})$, (RTy-App) and (RTy-BrRight) are used, i.e., $\vdash \mathtt{br} : \mathtt{o} \to \mathtt{oR} \to \mathtt{oR}$. In the derivation tree, $(x_{\mathtt{o}} : \mathtt{oR})$ becomes an environment of either u_1 or U_2. If $(x_{\mathtt{o}} : \mathtt{oR} \vdash u_1 : \mathtt{o})$, by Lemma 20, $\mathtt{oR} \to \mathtt{o}$ is well-formed, which is a contradiction; hence, we have

$$\vdash u_1 : \mathtt{o} \qquad x_{\mathtt{o}} : \mathtt{oR} \vdash U_2 : \mathtt{oR}\,.$$

By item 2 of the induction hypothesis for u_1, we have $\vdash u_1' : \mathtt{o}$ and hence $(x_{\mathtt{o}} : \mathtt{oR} \vdash \mathtt{br}\,u_1'\,U_2 : \mathtt{oR})$ as required. Item 2 is similar (and easier); and the case where U_2 is reduced is also similar.

Case where the head of u is a non-terminal: Let $u = A_\delta\,U_1 \cdots U_\ell$, $u' \in [U_i/x_i']_{i \le \ell} v$, and the rule used for $u \to u'$ be $A_\delta\,x_1' \cdots x_\ell' \to v$. Suppose

$$\begin{aligned}\delta &= \wedge_{i \le k_1} \delta_i^1 \to \cdots \to \wedge_{i \le k_m} \delta_i^m \to \delta^0 \\ \delta^0 &= \wedge_{i \le k_{m+1}} \delta_i^{m+1} \to \cdots \to \wedge_{i \le k_n} \delta_i^n \to \mathtt{o}\end{aligned}$$

where $\mathtt{order}(\delta_i^j) \ge 1$ for $j \le m$ and $i \le k_j$ and $\mathtt{order}(\delta^0) \le 1$. In the case where δ^0 is unbalanced, $k_j = 0$ for all $j \in \{m+1, \ldots, n\}$, and in the case where δ^0 is balanced, $k_{j_0} = 1$ for some (unique) $j_0 \in \{m+1, \ldots, n\}$.

Let

$$\begin{array}{lll}\Phi := (x_{\mathtt{o}} : \mathtt{oR}) & \rho := \mathtt{oR} & \text{(in the case of item 1)} \\ \Phi := \emptyset & \rho := \mathtt{o} & \text{(in the case of item 2).}\end{array}$$

For the hypothesis $(\Phi \vdash A_\delta\,U_1 \cdots U_\ell : \rho)$, (RTy-App) are used ℓ-times, and we have

$$\vdash A_\delta : \rho_1 \to \cdots \to \rho_\ell \to \rho \tag{1}$$
$$\Phi_i \vdash U_i : \rho_i \quad (i \le \ell) \tag{2}$$
$$\Phi = \Phi_1 \cup \cdots \cup \Phi_\ell\,. \tag{3}$$

The rule used for (1) is (RTy-NtRight) or (RTy-NtAll): in the former case, we have

$$\begin{aligned}&\rho_1 \to \cdots \to \rho_\ell \to \rho = (\delta)^\sharp \\ &= (\delta_1^1)^\sharp \to \cdots \to (\delta_{k_1}^1)^\sharp \to \cdots \to (\delta_1^m)^\sharp \to \cdots \to (\delta_{k_m}^m)^\sharp \to (\delta^0)^\sharp\end{aligned}$$

i.e.,

$$(\rho_1, \ldots, \rho_\ell) = \big((\delta_1^1)^\sharp, \ldots, (\delta_{k_1}^1)^\sharp, \ldots, (\delta_1^m)^\sharp, \ldots, (\delta_{k_m}^m)^\sharp\big) \tag{4}$$
$$\rho = (\delta^0)^\sharp\,. \tag{5}$$

In the latter case, similarly we have

$$(\rho_1, \ldots, \rho_\ell) = \big((\delta_1^1)^\sharp_{\mathtt{o}}, \ldots, (\delta_{k_1}^1)^\sharp_{\mathtt{o}}, \ldots, (\delta_1^m)^\sharp_{\mathtt{o}}, \ldots, (\delta_{k_m}^m)^\sharp_{\mathtt{o}}\big) \tag{6}$$
$$\rho = (\delta^0)^\sharp_{\mathtt{o}}\,. \tag{7}$$

Meanwhile, since the rule $A_\delta\, x'_1 \cdots x'_\ell \to v$ in $\mathcal{G}''$ is produced by (Tr1-Rule), there is a rule $A\, x_1 \cdots x_n \to s$ in $\mathcal{G}$ such that

$$\vdash \lambda x_1.\cdots\lambda x_n.s : \delta \Rightarrow \lambda x'_1.\cdots\lambda x'_\ell.v \qquad \delta :: \mathcal{N}(A).$$

Therefore, by (Tr1-Abs1) and/or (Tr1-Abs2), we have the following.

$$x_1 : \delta^1_1, \ldots, x_1 : \delta^1_{k_1}, \ldots, x_n : \delta^n_1, \ldots, x_n : \delta^n_{k_n} \vdash s : \mathtt{o} \Rightarrow v' \tag{8}$$
$$v = v' \qquad \text{(when } \delta^0 \text{ is unbalanced)} \tag{9}$$
$$v = [\mathtt{e}/(x_{j_0})_\mathtt{o}]v' \quad \text{(when } \delta^0 \text{ is balanced)} \tag{10}$$
$$\left(x'_1, \ldots, x'_\ell\right) = \left((x_1)_{\delta^1_1}, \ldots, (x_1)_{\delta^1_{k_1}}, \ldots, (x_m)_{\delta^m_1}, \ldots, (x_m)_{\delta^m_{k_m}}\right). \tag{11}$$

From now on, the proof goes separately for each item.

Case of item 1: Since $\rho = \mathtt{oR}$, we have (4) and (5). By (5), δ^0 is unbalanced, and so we have (9). By (8) and Lemma 22 with (4), (9), and (11), we have

$$x'_1 : \rho_1, \ldots, x'_\ell : \rho_\ell \vdash v : \mathtt{oR}\,.$$

Then, by (2), (3), and Lemma 21, we have

$$x_\mathtt{o} : \mathtt{oR} \vdash [U_i/x'_i]_{i\le\ell} v : \mathtt{oR}$$

and by (RTy-Set), we have

$$x_\mathtt{o} : \mathtt{oR} \vdash u' : \mathtt{oR}$$

as required.

Case of item 2: We have

$$(x_1)_{\delta^1_1} : (\delta^1_1)^\sharp_\mathtt{o}, \ldots, (x_1)_{\delta^1_{k_1}} : (\delta^1_{k_1})^\sharp_\mathtt{o}, \ldots, (x_n)_{\delta^n_1} : (\delta^n_1)^\sharp_\mathtt{o}, \ldots, (x_n)_{\delta^n_{k_n}} : (\delta^n_{k_n})^\sharp_\mathtt{o} \vdash v : \mathtt{o}$$

either by using (8), Lemma 23-2, and (9) when δ^0 is unbalanced, or by using (8), Lemma 23-3, and (10) when δ^0 is balanced. By (2) and Lemma 23-1, we have

$$\vdash U_i : (\rho_i)^\flat \quad (i \le \ell)\,.$$

By either (4) or (6), we have

$$\left((\rho_1)^\flat, \ldots, (\rho_\ell)^\flat\right) = \left((\delta^1_1)^\sharp_\mathtt{o}, \ldots, (\delta^1_{k_1})^\sharp_\mathtt{o}, \ldots, (\delta^m_1)^\sharp_\mathtt{o}, \ldots, (\delta^m_{k_m})^\sharp_\mathtt{o}\right).$$

Hence, by Lemma 21,

$$\vdash [U_i/x'_i]_{i\le\ell} v : \mathtt{o}$$

and by (RTy-Set), we have

$$\vdash u' : \mathtt{oR}$$

as required. $\square$

Below, we write $u \not\longrightarrow$ if $u \longrightarrow v$ does not hold for any v.

▶ **Lemma 25.** *For any v such that $v \not\longrightarrow$, $x_\mathtt{o} : \mathtt{oR} \vdash v : \mathtt{oR}$, and $[U/x_\mathtt{o}]v \longrightarrow^* \pi$ for some U and π, there exist $\pi_1, \ldots, \pi_n$ $(n \ge 0)$ such that $v = \pi_1 * (\pi_2 * \cdots (\pi_n * x_\mathtt{o}) \cdots)$.*

Proof. The proof proceeds by induction on v.

- Case where the head of v is a non-terminal A: A has a rewriting rule since $[U/x_\circ]v \longrightarrow^* \pi$, but it contradicts $v \not\longrightarrow$.
- Case where the head of v is a variable: Since $x_\circ : \mathtt{oR} \vdash v : \mathtt{oR}$, $v = x_\circ$; hence the result holds for $n = 0$.
- Case where the head of v is a terminal a: a must have non-zero arity since $x_\circ\!:\!\mathtt{oR} \vdash a : \mathtt{oR}$ cannot be derived; thus $v = \mathtt{br}\, V_0\, V_1$ for some V_0 and V_1. Since $v = \mathtt{br}\, V_0\, V_1 \not\longrightarrow$, V_0 and V_1 must be singletons $\{v_0\}$ and $\{v_1\}$, respectively. Now $\mathtt{br}$ must has type $\mathtt{o} \to \mathtt{oR} \to \mathtt{oR}$ and hence we have $\vdash v_0 : \mathtt{o}$ and $x_\circ\!:\!\mathtt{oR} \vdash v_1 : \mathtt{oR}$ since if we had $x_\circ\!:\!\mathtt{oR} \vdash v_0 : \mathtt{o}$ then $\mathtt{oR} \to \mathtt{o}$ would be well-formed by Lemma 20, which is a contradiction. Also, since $[U/x_\circ]v = \mathtt{br}\,([U/x_\circ]v_0)\,([U/x_\circ]v_1) \longrightarrow^* \pi$, there exist π_0 and π_1 such that $([U/x_\circ]v_i) \longrightarrow^* \pi_i$. Thus we can use the induction hypothesis for v_1. Now v_0 is closed and hence $v_0 \longrightarrow^* \pi_0$, but since $v_0 \not\longrightarrow$, we have $v_0 = \pi_0$.

◻

Proof of Lemma 19. First note that, for any v' such that $v \longrightarrow^* v'$, we have $x_\circ\!:\!\mathtt{oR} \vdash v' : \mathtt{oR}$. This is because, from the assumption $x : \mathtt{o} \vdash s : \delta \Rightarrow v$, δ is unbalanced by Lemma 13, and hence we have $x_\circ : \mathtt{oR} \vdash v' : \mathtt{oR}$ by Lemmas 22 and 24-1.

Now we prove the goal of the current lemma by using the context lemma (Lemma 12). Given $[U/x_\circ]v \longrightarrow^* \pi$, there exists v' such that

$$v \longrightarrow^* v' \not\longrightarrow \qquad [U/x_\circ]v' \longrightarrow^* \pi\,.$$

By Lemma 25, there exist $\pi_1, \dots, \pi_n$ such that

$$v' = \pi_1 * (\pi_2 * \cdots (\pi_n * x_\circ) \cdots).$$

Since

$$[U/x_\circ]v' = \pi_1 * (\pi_2 * \cdots (\pi_n * U) \cdots) \longrightarrow^* \pi$$

there exist $u \in U$ and π' such that

$$u \longrightarrow^* \pi' \qquad \pi_1 * (\pi_2 * \cdots (\pi_n * \pi') \cdots) = \pi.$$

Therefore

$$\begin{aligned}[]
[\mathsf{e}/x_\circ]v * U &= (\pi_1 * (\pi_2 * \cdots (\pi_n * \mathsf{e}) \cdots)) * U \\
&\longrightarrow^* (\pi_1 * (\pi_2 * \cdots (\pi_n * \mathsf{e}) \cdots)) * \pi' \\
&\sim_{\mathrm{v}} \pi_1 * (\pi_2 * \cdots (\pi_n * \pi') \cdots) = \pi.
\end{aligned}$$

On the other hand, given $([\mathsf{e}/x_\circ]v) * U \longrightarrow^p \pi$, there exist p_0, p_1, π'_0, and π'_1 such that

$$[\mathsf{e}/x_\circ]v \longrightarrow^{p_0} \pi'_0 \qquad \pi'_0 * U \longrightarrow^{p_1} \pi'_0 * \pi'_1 = \pi \qquad p_0 + p_1 = p\,.$$

For $[\mathsf{e}/x_\circ]v \longrightarrow^{p_0} \pi'_0$, we have v' such that

$$v \longrightarrow^{p_2} v' \not\longrightarrow \qquad [\mathsf{e}/x_\circ]v' \longrightarrow^{p_3} \pi'_0 \qquad p_2 + p_3 = p_0\,.$$

By Lemma 25, there exist $\pi_1, \dots, \pi_n$ such that

$$v' = \pi_1 * (\pi_2 * \cdots (\pi_n * x_\circ) \cdots).$$

Since

$$[\mathtt{e}/x_\circ]v' = \pi_1 * (\pi_2 * \cdots (\pi_n * \mathtt{e}) \cdots) \longrightarrow^{p_3} \pi_0',$$

we have

$$p_3 = 0 \qquad p_2 = p_0 \qquad \pi_1 * (\pi_2 * \cdots (\pi_n * \mathtt{e}) \cdots) = \pi_0' .$$

Hence,

$$[U/x_\circ]v \longrightarrow^{p_2} [U/x_\circ]v' = \pi_1 * (\pi_2 * \cdots (\pi_n * U) \cdots) \longrightarrow^{p_1} \pi_1 * (\pi_2 * \cdots (\pi_n * \pi_1') \cdots)$$

i.e.,

$$[U/x_\circ]v \longrightarrow^{p} \pi_1 * (\pi_2 * \cdots (\pi_n * \pi_1') \cdots) \sim_{\mathrm{v}} \pi_0' * \pi_1' = \pi.$$

□

A.3 Lemmas for Forward Direction

▶ **Lemma 26** (de-substitution). *Given $\Gamma \vdash_{\mathrm{s}} [t/x]s : \delta \Rightarrow v$ where t is closed and s and t are applicative terms, there exist $k \geq 0$, $(\delta_i)_{i\leq k}$, $(U_i)_{i\leq k}$, and $v^\bullet$ such that*

1. $\Gamma, x : \delta_1, \ldots, x : \delta_k \vdash_{\mathrm{s}} s : \delta \Rightarrow v^\bullet$
2. $\vdash_{\mathrm{s}} t : \delta_i \Rightarrow U_i \quad (i \leq k)$
3. *for each $i \leq k$, if δ_i is unbalanced, U_i is a singleton*
4. $v \lesssim [U_i/x_{\delta_i}]_{i\leq k} v^\bullet$.

Proof. The proof is by induction on s and analysis on the last rule used for deriving $\Gamma \vdash_{\mathrm{s}} [t/x]s : \delta \Rightarrow v$.

Case $s = x$: Since $\Gamma \vdash_{\mathrm{s}} ([t/x]x =)\, t : \delta \Rightarrow v$ and t is closed, by Lemma 18, we also have $\vdash_{\mathrm{s}} t : \delta \Rightarrow v$ and Γ is balanced. For item 1, we define $k := 1$, $\delta_1 := \delta$, and $v^\bullet := x_\delta$. We define $U_1 := \{v\}$ for items 2 and 3. Then, item 4 is clear.

Case $s = a$, A, or $y \neq x$: Since $x \notin \mathbf{FV}(s)$, $[t/x]s = s$. So we define $k := 0$, $v^\bullet := v$.

Case s is an application: the last rule used for $\Gamma \vdash_{\mathrm{s}} [t/x]s : \delta \Rightarrow v$ is (Tr1-App1) or (Tr1-App2):

$$\frac{\begin{array}{c}\Gamma' \vdash_{\mathrm{s}} [t/x]s' : \delta_1' \wedge \cdots \wedge \delta_{k'}' \to \delta \Rightarrow v' \\ \Gamma_j' \vdash_{\mathrm{s}} [t/x]t' : \delta_j' \Rightarrow U_j' \quad (\text{for each } j \in \{1, \ldots, k'\})\end{array}}{\Gamma' \cup \Gamma_1' \cup \cdots \cup \Gamma_{k'}' \vdash_{\mathrm{s}} ([t/x]s')\,([t/x]t') : \delta \Rightarrow \begin{cases} v'U_1' \cdots U_{k'}' & (\mathtt{order}(t') \geq 1 \vee k' = 0) \\ v' * U_1' & (\mathtt{order}(t') = 0 \wedge k' = 1)\end{cases}}$$

where

$$\Gamma = \Gamma' \cup \Gamma_1' \cup \cdots \cup \Gamma_{k'}' \tag{12}$$
$$s = s't'$$
$$v = \begin{cases} v'U_1' \cdots U_{k'}' & (\mathtt{order}(t') \geq 1 \vee k' = 0) \\ v' * U_1' & (\mathtt{order}(t') = 0 \wedge k' = 1)\,. \end{cases}$$

For each $j \leq k'$, the rule used last for $\Gamma_j' \vdash_{\mathrm{s}} [t/x]t' : \delta_j' \Rightarrow U_j'$ is (Tr1-SetS):

$$\frac{\begin{array}{c}\Gamma_j' \vdash_{\mathrm{s}} [t/x]t' : \delta_j' \Rightarrow u_{jh}' \quad (\text{for each } h \in \{1, \ldots, k_j\}) \\ k_j = 1 \text{ if } \delta_j' \text{ is unbalanced}\end{array}}{\Gamma_j' \vdash_{\mathrm{s}} [t/x]t' : \delta_j' \Rightarrow \{u_{j1}', \ldots, u_{jk_j}'\} (= U_j')} \tag{13}$$

Hence, by induction hypotheses for s' and for t', there exist $k^0 \geq 0$, $(\delta_i^0)_{i\leq k^0}$, $(U_i^0)_{i\leq k^0}$, and $v'^\bullet$ such that

$$\Gamma', x:\delta_1^0, \dots, x:\delta_{k^0}^0 \vdash_{\mathrm{s}} s' : \delta'_1 \wedge \dots \wedge \delta'_{k'} \to \delta \Rightarrow v'^\bullet \tag{14}$$

$$\vdash_{\mathrm{s}} t : \delta_i^0 \Rightarrow U_i^0 \quad (i \leq k^0) \tag{15}$$

$$\text{for each } i \leq k^0\text{, if } \delta_i^0 \text{ is unbalanced, } U_i^0 \text{ is a singleton} \tag{16}$$

$$v' \lesssim [U_i^0/x_{\delta_i^0}]_{i\leq k^0} v'^\bullet \tag{17}$$

and for each $j \leq k'$ and $h \leq k_j$ there exist $k^{jh} \geq 0$, $(\delta_i^{jh})_{i\leq k^{jh}}$, $(U_i^{jh})_{i\leq k^{jh}}$, and $u'^\bullet_{jh}$ such that

$$\Gamma'_j, x:\delta_1^{jh}, \dots, x:\delta_{k^{jh}}^{jh} \vdash_{\mathrm{s}} t' : \delta'_j \Rightarrow u'^\bullet_{jh} \tag{18}$$

$$\vdash_{\mathrm{s}} t : \delta_i^{jh} \Rightarrow U_i^{jh} \quad (i \leq k^{jh}) \tag{19}$$

$$\text{for each } i \leq k^{jh}\text{, if } \delta_i^{jh} \text{ is unbalanced, } U_i^{jh} \text{ is a singleton} \tag{20}$$

$$u'_{jh} \lesssim [U_i^{jh}/x_{\delta_i^{jh}}]_{i\leq k^{jh}} u'^\bullet_{jh}\,. \tag{21}$$

For each $j \leq k'$, by (Tr1-SetS) and a (derived) weakening rule, we have

$$\dfrac{\dfrac{\Gamma'_j, x:\delta_1^{jh}, \dots, x:\delta_{k^{jh}}^{jh} \vdash_{\mathrm{s}} t' : \delta'_j \Rightarrow u'^\bullet_{jh} \quad (h \leq k_j)}{\Gamma'_j \cup \{x:\delta_i^{jh} \mid h \leq k_j, i \leq k^{jh}\} \vdash_{\mathrm{s}} t' : \delta'_j \Rightarrow u'^\bullet_{jh} \quad (h \leq k_j)} \qquad k_j = 1 \text{ if } \delta'_j \text{ is unbalanced}}{\Gamma'_j \cup \{x:\delta_i^{jh} \mid h \leq k_j, i \leq k^{jh}\} \vdash_{\mathrm{s}} t' : \delta'_j \Rightarrow \{u'^\bullet_{jh} \mid h \leq k_j\}}$$

where, when δ'_j is unbalanced, since $k_j = 1$ we do not need the weakening rule; when δ'_j is balanced, by Lemma 13 applied to (18), δ_i^{jh} must be balanced for each h and i, and hence we can use the weakening rule. Now we define

$$U'^\bullet_j := \{u'^\bullet_{jh} \mid h \leq k_j\}\,.$$

Then, by (Tr1-App1) or (Tr1-App2) with (14) and (12), we have

$$\Gamma \cup \{x:\delta_i^0 \mid i \leq k^0\} \cup (\cup_{j\leq k'}\{x:\delta_i^{jh} \mid h \leq k_j, i \leq k^{jh}\}) \vdash_{\mathrm{s}} s'\,t' : \delta \Rightarrow v^\bullet$$

where

$$v^\bullet := \begin{cases} v'^\bullet\, \overrightarrow{U'^\bullet_j}^{j\leq k'} & (\mathtt{order}(t') \geq 1 \vee k' = 0) \\ v'^\bullet * U'^\bullet_1 & (\mathtt{order}(t') = 0 \wedge k' = 1). \end{cases}$$

We define k and $(\delta_i)_{i\leq k}$ as the following enumeration:

$$\{x:\delta_i \mid i \leq k\} := \{x:\delta_i^0 \mid i \leq k^0\} \cup \{x:\delta_i^{jh} \mid j \leq k', h \leq k_j, i \leq k^{jh}\}\,.$$

Thus we have obtained item 1.

For each $i \leq k$ we define

$$U_i := \cup(\{U_{i'}^0 \mid \delta_{i'}^0 = \delta_i\} \cup \{U_{i'}^{jh} \mid \delta_{i'}^{jh} = \delta_i\})\,.$$

By (15), (19), and (Tr1-SetS), for each $i \leq k$ we have

$$\vdash_{\mathrm{s}} t : \delta_i \Rightarrow U_i$$

where, when δ_i is unbalanced, we use (16) and (20) and we can show that if δ_i is unbalanced, then $\{i' \le k^0 \mid \delta^0_{i'} = \delta_i\} \cup \{(j,h,i') \mid \delta^{jh}_{i'} = \delta_i\}$ is a singleton as follows. The set is non-empty by the definition of δ_i. If $\{i' \le k^0 \mid \delta^0_{i'} = \delta_i\}$ is non-empty, it is a singleton, and we show that $\{(j,h,i') \mid \delta^{jh}_{i'} = \delta_i\}$ is empty. For every j, h, and i', by Lemma 13 applied to (14) and (18) and the fact that $\delta^0_{i'} (= \delta_i)$ is unbalanced for some i', δ'_j and $\delta^{jh}_{i'}$ must be balanced. Hence, $\delta^{jh}_{i'} \neq \delta_i$ as δ_i is unbalanced. If $\{(j,h,i') \mid \delta^{jh}_{i'} = \delta_i\}$ is non-empty, similarly, $\{i' \le k^0 \mid \delta^0_{i'} = \delta_i\}$ is empty. To show that $\{(j,h,i') \mid \delta^{jh}_{i'} = \delta_i\}$ is a singleton, suppose $\delta^{j_0 h_0}_{i_0} = \delta^{j_1 h_1}_{i_1} = \delta_i$. By Lemma 13 applied to (18), δ'_{j_0} and δ'_{j_1} are unbalanced. Hence $j_0 = j_1$ from the well-formedness of $\delta'_1 \wedge \cdots \wedge \delta'_{k'} \to \delta$, and we have also $k_{j_0} = 1$ from (13). Therefore $h_0 = h_1$ ($\le k_{j_0} = 1$), and then $i_0 = i_1$. Thus, we have obtained items 2 and 3.

Finally we show item 4, i.e., $v \lesssim [U_i/x_{\delta_i}]_{i\le k} v^\bullet$. In the case where $\mathtt{order}(t') \ge 1 \vee k' = 0$,

$$\begin{aligned}
v &= v'\, \overrightarrow{U'_j}^{\,j\le k'} \\
&\lesssim ([U^0_i/x_{\delta^0_i}]_{i\le k^0} v'^\bullet)\, \overrightarrow{\left\{[U^{jh}_i/x_{\delta^{jh}_i}]_{i\le k^{jh}} u'^\bullet_{jh} \,\middle|\, h \le k_j\right\}}^{\,j\le k'} && \text{(by (17) and (21))} \\
&\lesssim ([U_i/x_{\delta_i}]_{i\le k} v'^\bullet)\, \overrightarrow{\left\{[U_i/x_{\delta_i}]_{i\le k} u'^\bullet_{jh} \,\middle|\, h \le k_j\right\}}^{\,j\le k'} \\
&= [U_i/x_{\delta_i}]_{i\le k} v^\bullet\,.
\end{aligned}$$

In the case where $\mathtt{order}(t') = 0 \wedge k' = 1$,

$$\begin{aligned}
v &= v' * U'_1 \\
&\lesssim ([U^0_i/x_{\delta^0_i}]_{i\le k^0} v'^\bullet) * ([U^{11}_i/x_{\delta^{11}_i}]_{i\le k^{11}} u'^\bullet_{11}) && \text{(by (17) and (21), and now } k_1 = 1) \\
&\lesssim ([U_i/x_{\delta_i}]_{i\le k} v'^\bullet) * ([U_i/x_{\delta_i}]_{i\le k} u'^\bullet_{11}) \\
&= [U_i/x_{\delta_i}]_{i\le k} v^\bullet\,.
\end{aligned}$$

◻

The following lemma states that the transformation relation (up to $\lesssim$) is a left-to-right backward simulation relation.

▶ **Lemma 27** (subject expansion). *If $t \longrightarrow_{\mathcal{G}} t'$ and $\vdash_{\mathrm{s}} t' : \mathtt{o} \Rightarrow u'$, then there exists u such that $\vdash_{\mathrm{s}} t : \mathtt{o} \Rightarrow u$ with $u \longrightarrow_{\mathcal{G}''} \gtrsim u'$.*

Proof. The proof is given by the induction on t and by the case analysis of the reduction $t \longrightarrow_{\mathcal{G}} t'$.

Case where $t = \mathtt{e}$: Trivial.

Case where $t = a\, t_1$ and $\Sigma(a) = 1$: Let the last rule used for $t \longrightarrow_{\mathcal{G}} t'$ be

$$\frac{t_1 \longrightarrow_{\mathcal{G}} t'_1}{a\, t_1 \longrightarrow_{\mathcal{G}} a\, t'_1}$$

Since $\vdash_{\mathrm{s}} (t' =)\, a\, t'_1 : \mathtt{o} \Rightarrow u'$ is derived by (Tr1-App2) and (Tr1-SetS), we have $\vdash_{\mathrm{s}} t'_1 : \mathtt{o} \Rightarrow u'_1$ such that $u' = \mathtt{br}\, a\, u'_1$. Hence by the induction hypothesis for t'_1, there exists u_1 such that $\vdash_{\mathrm{s}} t_1 : \mathtt{o} \Rightarrow u_1$ and $u_1 \longrightarrow_{\mathcal{G}''} \gtrsim u'_1$. Therefore we have $u := \mathtt{br}\, a\, u_1$ with $\vdash_{\mathrm{s}} a\, t_1 : \mathtt{o} \Rightarrow \mathtt{br}\, a\, u_1$ and $\mathtt{br}\, a\, u_1 \longrightarrow_{\mathcal{G}''} \gtrsim \mathtt{br}\, a\, u'_1$.

Case where $t = A\, t_1 \, \ldots \, t_n$: Let the last rule used for $t \longrightarrow_{\mathcal{G}} t'$ be

$$\frac{A\, x_1 \cdots x_n \to s \in \mathcal{R}}{A\, t_1 \cdots t_n \longrightarrow_{\mathcal{G}} [t_1/x_1, \ldots, t_n/x_n] s}$$

and let $\mathcal{N}(A) = \kappa_1 \to \cdots \to \kappa_n \to \mathtt{o}$. By the assumption on sorts, there exists unique m such that $0 \le m \le n$, $\mathtt{order}(\kappa_j) \ge 1$ for all $j \le m$, and $\mathtt{order}(\kappa_j) = 0$ for all $j > m$. Let $v^{n+1} := u'$ and $\Gamma^{n+1} := \emptyset$; then

$$\Gamma^{n+1} \vdash_{\mathrm{s}} [t_1/x_1, \ldots, t_n/x_n]s : \mathtt{o} \Rightarrow v^{n+1}\,.$$

Hence by Lemma 26, for each $j = n, \ldots, 1$, there exist $\Gamma^j, k^j, (\delta^j_i)_{i \le k^j}, (U^j_i)_{i \le k^j}, v^j$ such that

$$\begin{aligned}
&\Gamma^j = (\Gamma^{j+1}, x_j : \delta^j_1, \ldots, x_j : \delta^j_{k^j}) \\
&\quad \vdash_{\mathrm{s}} t_j : \delta^j_i \Rightarrow U^j_i \qquad (i \le k^j) \\
&\Gamma^j \vdash_{\mathrm{s}} [t_{j'}/x_{j'}]_{j' \le j-1} s : \mathtt{o} \Rightarrow v^j \\
&\text{for each } i \le k^j\text{, if } \delta^j_i \text{ is unbalanced, } U^j_i \text{ is a singleton} \\
&v^{j+1} \lesssim [U^j_i/(x_j)_{\delta^j_i}]_{i \le k^j} v^j\,.
\end{aligned}$$

Note that for each $j > m$, $k^j \le 1$, and there is at most one $j > m$ such that $k^j = 1$ by Lemma 13.

By (Tr1-NT), we have $\vdash_{\mathrm{s}} A : \delta \Rightarrow A_\delta$. Since we have also $\vdash_{\mathrm{s}} t_j : \delta^j_i \Rightarrow U^j_i$ $(i \le k^j)$ for $j = 1, \ldots, m$, by using (Tr1-App1) iteratively, we have

$$\vdash_{\mathrm{s}} A\, t_1 \cdots t_m : \wedge_{i \le k^{m+1}} \delta^{m+1}_i \to \cdots \wedge_{i \le k^n} \delta^n_i \to \mathtt{o} \Rightarrow A_\delta\, \overrightarrow{U^1_i}^{i \le k^1} \cdots \overrightarrow{U^m_i}^{i \le k^m}\,.$$

Then, since we have $\vdash_{\mathrm{s}} t_j : \delta^j_i \Rightarrow U^j_i$ $(i \le k^j)$ for $j = m+1, \ldots, n$, by using (Tr1-App1) where $k = 0$ and/or (Tr1-App2) iteratively, we have

$$\begin{aligned}
&\vdash_{\mathrm{s}} A\, t_1 \cdots t_n : \mathtt{o} \Rightarrow u \\
&u := \begin{cases} \left(A_\delta\, \overrightarrow{U^1_i}^{i \le k^1} \cdots \overrightarrow{U^m_i}^{i \le k^m}\right) * U^j_1 & (k_j > 0 \text{ for some (unique) } j \in \{m+1, \ldots, n\}) \\ A_\delta\, \overrightarrow{U^1_i}^{i \le k^1} \cdots \overrightarrow{U^m_i}^{i \le k^m} & (\text{otherwise}). \end{cases}
\end{aligned}$$

Meanwhile, since we have $\Gamma^1 \vdash_{\mathrm{s}} [t_{j'}/x_{j'}]_{j' \le 0} s : \mathtt{o} \Rightarrow v^1$, so do $\Gamma^1 \vdash [t_{j'}/x_{j'}]_{j' \le 0} s : \mathtt{o} \Rightarrow v^1$, i.e.,

$$x_1 : \delta^1_1, \ldots, x_1 : \delta^1_{k^1}, \ldots, x_n : \delta^n_1, \ldots, x_n : \delta^n_{k^n} \vdash s : \mathtt{o} \Rightarrow v^1\,.$$

Now we define

$$v^0 := \begin{cases} [\mathtt{e}/(x_j)_{\mathtt{o}}]v^1 & (k_j > 0 \text{ for some (unique) } j \in \{m+1, \ldots, n\}) \\ v^1 & (\text{otherwise}). \end{cases} \tag{22}$$

By iterating (Tr1-Abs1) where $k = 0$ and/or (Tr1-Abs2), we have

$$\begin{aligned}
&x_1 : \delta^1_1, \ldots, x_1 : \delta^1_{k^1}, \ldots, x_m : \delta^m_1, \ldots, x_m : \delta^m_{k^m} \vdash \\
&\lambda x_{m+1}. \cdots \lambda x_n.s : \wedge_{i \le k^{m+1}} \delta^{m+1}_i \to \cdots \to \wedge_{i \le k^n} \delta^n_i \to \mathtt{o} \Rightarrow v^0
\end{aligned}$$

and by iterating (Tr1-Abs1), we have

$$\begin{aligned}
&\vdash \lambda x_1. \cdots \lambda x_n.s : \wedge_{i \le k^1} \delta^1_i \to \cdots \to \wedge_{i \le k^n} \delta^n_i \to \mathtt{o} \Rightarrow \\
&\quad \lambda(x_1)_{\delta^1_1}. \cdots \lambda(x_1)_{\delta^1_{k^1}}. \cdots \lambda(x_m)_{\delta^m_1}. \cdots \lambda(x_m)_{\delta^m_{k^m}}.v^0\,.
\end{aligned}$$

Hence, by (TR1-RULE), we have

$$\vdash (A\,x_1 \cdots x_n \to s) \Rightarrow \big(A_\delta\,(x_1)_{\delta^1_1} \cdots (x_1)_{\delta^1_{k^1}} \cdots (x_m)_{\delta^m_1} \cdots (x_m)_{\delta^m_{k^m}} \to v^0\big) \tag{23}$$

where $\delta := \wedge_{i\le k^1}\delta^1_i \to \cdots \to \wedge_{i\le k^n}\delta^n_i \to \mathsf{o}$.

By Lemmas 16 and 17 and since $v^2 \lesssim [U^1_i/(x_1)_{\delta^1_i}]_{i\le k^1} v^1$, we have $v^3 \lesssim [U^2_i/(x_2)_{\delta^2_i}]_{i\le k^2} v^2 \lesssim ([U^1_i/(x_1)_{\delta^1_i}]_{i\le k^1} \cup [U^2_i/(x_2)_{\delta^2_i}]_{i\le k^2})v^1$. Iterating this reasoning, we have

$$v^{m+1} \lesssim ([U^1_i/(x_1)_{\delta^1_i}]_{i\le k^1} \cup \cdots \cup [U^m_i/(x_m)_{\delta^m_i}]_{i\le k^m})v^1\,. \tag{24}$$

Further,

$$\begin{aligned} v^{n+1} &\lesssim ([U^{m+1}_i/(x_{m+1})_{\delta^{m+1}_i}]_{i\le k^{m+1}} \cup \cdots \cup [U^n_i/(x_n)_{\delta^n_i}]_{i\le k^n})v^{m+1} \\ &= \begin{cases} [U^j_1/(x_j)_{\mathsf{o}}]v^{m+1} & (k_j > 0 \text{ for some (unique) } j \in \{m+1,\dots,n\}) \\ v^{m+1} & (\text{otherwise}). \end{cases} \end{aligned} \tag{25}$$

In the case where $k_j > 0$ for some $j \in \{m+1,\dots,n\}$, we have

$$\begin{aligned} u &= \Big(A_\delta\,\overrightarrow{U^1_i}^{i\le k^1} \cdots \overrightarrow{U^m_i}^{i\le k^m}\Big) * U^j_1 \\ \longrightarrow_{\mathcal{G}''} &\big(([U^1_i/(x_1)_{\delta^1_i}]_{i\le k^1} \cup \cdots \cup [U^m_i/(x_m)_{\delta^m_i}]_{i\le k^m})v^0\big) * U^j_1 && \text{(by (23))} \\ &= \big([\mathsf{e}/(x_j)_{\mathsf{o}}]([U^1_i/(x_1)_{\delta^1_i}]_{i\le k^1} \cup \cdots \cup [U^m_i/(x_m)_{\delta^m_i}]_{i\le k^m})v^1\big) * U^j_1 && \text{(by (22))} \\ &\gtrsim \big([\mathsf{e}/(x_j)_{\mathsf{o}}]v^{m+1}\big) * U^j_1 && \text{(by (24))} \\ &\sim [U^j_1/(x_j)_{\mathsf{o}}]v^{m+1} && \text{(Lemma 19)} \\ &\gtrsim v^{n+1} = u' && \text{(by (25)).} \end{aligned}$$

In the other case, we have

$$\begin{aligned} u &= A_\delta\,\overrightarrow{U^1_i}^{i\le k^1} \cdots \overrightarrow{U^m_i}^{i\le k^m} \\ \longrightarrow_{\mathcal{G}''} &([U^1_i/(x_1)_{\delta^1_i}]_{i\le k^1} \cup \cdots \cup [U^m_i/(x_m)_{\delta^m_i}]_{i\le k^m})v^0 && \text{(by (23))} \\ &= ([U^1_i/(x_1)_{\delta^1_i}]_{i\le k^1} \cup \cdots \cup [U^m_i/(x_m)_{\delta^m_i}]_{i\le k^m})v^1 && \text{(by (22))} \\ &\gtrsim v^{m+1} && \text{(by (24))} \\ &\gtrsim v^{n+1} = u' && \text{(by (25)).} \end{aligned}$$

◻

A.4 Lemmas for Backward Direction

For a given Γ, we write $\Gamma \setminus x$ for Γ' such that $\Gamma = (\Gamma', x : \delta_1, \dots, x : \delta_n)$ for some $\delta_1, \dots, \delta_n$ and $x \notin dom(\Gamma')$.

▶ **Lemma 28** (substitution). *Given $\Gamma, x : \delta_1, \dots, x : \delta_k \vdash s : \delta \Rightarrow v$ where $x \notin dom(\Gamma)$ and $k \ge 0$, and given $\vdash t : \delta_i \Rightarrow U_i$ for each $i \le k$, we have*

$$\Gamma \vdash [t/x]s : \delta \Rightarrow [U_i/x_{\delta_i}]_{i\le k}v\,.$$

Proof. The proof is given by induction on $\Gamma, x : \delta_1, \dots, x : \delta_k \vdash s : \delta \Rightarrow v$. For any Γ, we define $\langle\Gamma\rangle := \{i \in \{1,\dots,k\} \mid x : \delta_i \in \Gamma\}$. The base cases are clear; in the case of variables, we use a derived rule of weakening for balanced environments.

Case of (Tr1-App1):

$$\frac{\Gamma'_0 \vdash s' : \delta'_1 \wedge \cdots \wedge \delta'_{k'} \to \delta \Rightarrow v' \qquad \Gamma'_j \vdash t' : \delta'_j \Rightarrow U'_j \text{ and } \delta'_i \neq \mathsf{o} \text{ (for each } i \in \{1, \ldots, k'\})}{\Gamma'_0 \cup \Gamma'_1 \cup \cdots \cup \Gamma'_{k'} \vdash s'\, t' : \delta \Rightarrow v'\, \overrightarrow{U'_j}^{j \le k'}}$$

We have

$$\begin{aligned}
&\Gamma, x : \delta_1, \ldots, x : \delta_k = \Gamma'_0 \cup \Gamma'_1 \cup \cdots \cup \Gamma'_{k'}\\
&s = s'\, t'\\
&v = v'\, \overrightarrow{U'_j}^{j \le k'} .
\end{aligned}$$

The rule used for $(\Gamma'_j \vdash t' : \delta'_j \Rightarrow U'_j)$ is (Tr1-Set):

$$\frac{\Gamma'_j \vdash t' : \delta'_j \Rightarrow u'_{jh} \quad (h \le k_j)}{\Gamma'_j \vdash t' : \delta'_j \Rightarrow \{u'_{j1}, \ldots, u'_{jk_j}\} \ (= U'_j)}$$

By the induction hypothesis for s' and t', we have

$$\Gamma'_0 \setminus x \vdash [t/x]s' : \delta'_1 \wedge \cdots \wedge \delta'_{k'} \to \delta \Rightarrow [U_i/x_{\delta_i}]_{i \in \langle \Gamma'_0 \rangle} v' \tag{26}$$

$$\Gamma'_j \setminus x \vdash [t/x]t' : \delta'_j \Rightarrow [U_i/x_{\delta_i}]_{i \in \langle \Gamma'_j \rangle} u'_{jh} \qquad (j \in \{1, \ldots, k'\}, h \le k_j) \tag{27}$$

and by using (Tr1-Set), from (27), we have

$$\Gamma'_j \setminus x \vdash [t/x]t' : \delta'_j \Rightarrow \cup_{h \le k_j} [U_i/x_{\delta_i}]_{i \in \langle \Gamma'_j \rangle} u'_{jh} \qquad (j \in \{1, \ldots, k'\}). \tag{28}$$

Now

$$\begin{aligned}
[U_i/x_{\delta_i}]_{i \le k} v &= ([U_i/x_{\delta_i}]_{i \le k} v')\, \overrightarrow{\cup_{h \le k_j} [U_i/x_{\delta_i}]_{i \le k} u'_{jh}}^{j \le k'}\\
&= ([U_i/x_{\delta_i}]_{i \in \langle \Gamma'_0 \rangle} v')\, \overrightarrow{\cup_{h \le k_j} [U_i/x_{\delta_i}]_{i \in \langle \Gamma'_j \rangle} u'_{jh}}^{j \le k'}\\
&= \left\{ v'^{\bullet}\, \overrightarrow{\cup_{h \le k_j} [U_i/x_{\delta_i}]_{i \in \langle \Gamma'_j \rangle} u'_{jh}}^{j \le k'} \;\middle|\; v'^{\bullet} \in [U_i/x_{\delta_i}]_{i \in \langle \Gamma'_0 \rangle} v' \right\}
\end{aligned}$$

where the second equation is shown by Lemma 14. For any $v'^{\bullet} \in [U_i/x_{\delta_i}]_{i \in \langle \Gamma'_0 \rangle} v'$, by (Tr1-Set) and (26), we have

$$\Gamma'_0 \setminus x \vdash [t/x]s' : \delta'_1 \wedge \cdots \wedge \delta'_{k'} \to \delta \Rightarrow v'^{\bullet}$$

and hence, by (Tr1-App1) with (28), we have

$$\cup_{j \in \{0, \ldots, k'\}} (\Gamma'_j \setminus x) \vdash ([t/x]s')([t/x]t') : \delta \Rightarrow v'^{\bullet}\, \overrightarrow{\cup_{h \le k_j} [U_i/x_{\delta_i}]_{i \in \langle \Gamma'_j \rangle} u'_{jh}}^{j \le k'}$$

where the linearity condition is satisfied, as

$$(\Gamma'_j \setminus x) \cap (\Gamma'_{j'} \setminus x) \subseteq \Gamma'_j \cap \Gamma'_{j'} \subseteq \text{ (the set of balanced terms)}$$

for $j \neq j'$. Therefore, again by (Tr1-Set),

$$\cup_{j \in \{0, \ldots, k'\}} (\Gamma'_j \setminus x) \vdash ([t/x]s')([t/x]t') : \delta \Rightarrow [U_i/x_{\delta_i}]_{i \le k} v$$

Since $\cup_{j\in\{0,\dots,k'\}}(\Gamma'_j \setminus x) = \Gamma$ and $([t/x]s')([t/x]t') = [t/x](s'\,t')$, we have shown the required condition.

Case of (Tr1-App2):

$$\frac{\Gamma'_0 \vdash s' : \mathtt{o} \to \delta \Rightarrow V' \qquad \Gamma'_1 \vdash t' : \mathtt{o} \Rightarrow U'}{\Gamma'_0 \cup \Gamma'_1 \vdash s'\,t' : \delta \Rightarrow \mathtt{br}\,V'\,U'}$$

We have

$$\begin{aligned}
&\Gamma, x : \delta_1, \dots, x : \delta_k = \Gamma'_0 \cup \Gamma'_1\\
&s = s'\,t'\\
&v = \mathtt{br}\,V'\,U'\,.
\end{aligned}$$

The rule used for $(\Gamma'_0 \vdash s' : \mathtt{o} \to \delta \Rightarrow V')$ and $(\Gamma'_1 \vdash t' : \mathtt{o} \Rightarrow U')$ is (Tr1-Set):

$$\frac{\Gamma'_0 \vdash s' : \mathtt{o} \to \delta \Rightarrow v'_h \quad (h \le k_0)}{\Gamma'_0 \vdash s' : \mathtt{o} \to \delta \Rightarrow \{v'_1, \dots, v'_{k_0}\}\ (= V')}$$

$$\frac{\Gamma'_1 \vdash t' : \mathtt{o} \Rightarrow u'_h \quad (h \le k_1)}{\Gamma'_1 \vdash t' : \mathtt{o} \Rightarrow \{u'_1, \dots, u'_{k_1}\}\ (= U')}$$

By the induction hypothesis for s' and t', we have

$$\Gamma'_0 \setminus x \vdash [t/x]s' : \mathtt{o} \to \delta \Rightarrow [U_i/x_{\delta_i}]_{i\in\langle\Gamma'_0\rangle} v'_h \qquad (h \le k_0) \tag{29}$$

$$\Gamma'_1 \setminus x \vdash [t/x]t' : \mathtt{o} \Rightarrow [U_i/x_{\delta_i}]_{i\in\langle\Gamma'_1\rangle} u'_h \qquad (h \le k_1) \tag{30}$$

and by using (Tr1-Set) (going and back), from (29) and (30), we have

$$\begin{aligned}
&\Gamma'_0 \setminus x \vdash [t/x]s' : \mathtt{o} \to \delta \Rightarrow [U_i/x_{\delta_i}]_{i\in\langle\Gamma'_0\rangle} V'\\
&\Gamma'_1 \setminus x \vdash [t/x]t' : \mathtt{o} \Rightarrow [U_i/x_{\delta_i}]_{i\in\langle\Gamma'_1\rangle} U'
\end{aligned}$$

Hence, by (Tr1-App2), we have

$$(\Gamma'_0 \setminus x) \cup (\Gamma'_1 \setminus x) \vdash ([t/x]s')([t/x]t') : \delta \Rightarrow \mathtt{br}\,([U_i/x_{\delta_i}]_{i\in\langle\Gamma'_0\rangle} V')\,([U_i/x_{\delta_i}]_{i\in\langle\Gamma'_1\rangle} U')$$

where the linearity condition is clear as shown in the previous case. Since

$$\begin{aligned}
[U_i/x_{\delta_i}]_{i\le k} v &= \mathtt{br}\,([U_i/x_{\delta_i}]_{i\le k} V')\,([U_i/x_{\delta_i}]_{i\le k} U')\\
&= \mathtt{br}\,([U_i/x_{\delta_i}]_{i\in\langle\Gamma'_0\rangle} V')\,([U_i/x_{\delta_i}]_{i\in\langle\Gamma'_1\rangle} U')
\end{aligned}$$

we have shown the required condition. ◻

The following lemma states that, roughly speaking, the transformation relation is a right-to-left forward simulation relation; also, this can be seen as a form of subject reduction.

▶ **Lemma 29.** *Given $u \longrightarrow^p_{\mathcal{G}''} \pi$ where $p > 0$ and $\vdash t : \mathtt{o} \Rightarrow u$, there exist t', u', π' and $q < p$ such that $t \longrightarrow^*_{\mathcal{G}} t'$, $\vdash t' : \mathtt{o} \Rightarrow u'$, and $u' \longrightarrow^q \pi' \sim_{\mathrm{v}} \pi$.*

Proof. The proof is given by the induction on t and by the case analysis of the head of u. Since $u \longrightarrow^+ \pi$, the head of u must be $\mathtt{br}$ or a non-terminal.

Case where $u = \mathtt{br}\,V'\,U'$: In the reduction $\mathtt{br}\,V'\,U' \longrightarrow^p \pi$ suppose that $v' \in V'$ and $u' \in U'$ are chosen. The last rule used for $\vdash t : \mathtt{o} \Rightarrow \mathtt{br}\,V'\,U'$ is either (Tr1-App1):

$$\frac{\vdash s' : \top \to \mathtt{o} \Rightarrow \mathtt{br}\,V'\,U'}{\vdash s't' : \mathtt{o} \Rightarrow \mathtt{br}\,V'\,U'}$$

or (Tr1-App2):

$$\frac{\vdash s' : \mathtt{o} \to \mathtt{o} \Rightarrow V' \qquad \vdash t' : \mathtt{o} \Rightarrow U'}{\vdash s'\,t' : \mathtt{o} \Rightarrow \mathtt{br}\,V'\,U'}$$

In the former case above, we can iterate this reasoning, and then there exist $n' \geq 1$, s', $t'_1, \ldots, t'_{n'}$ such that $t = s'\,t'_1 \,\cdots\, t'_{n'}$ and the following:

$$\dfrac{\dfrac{\vdots}{\dfrac{\dfrac{\dfrac{\dfrac{\vdash s' : \mathtt{o} \to \top \to \cdots \to \top \to \mathtt{o} \Rightarrow v' \qquad \cdots}{\vdash s' : \mathtt{o} \to \top \to \cdots \to \top \to \mathtt{o} \Rightarrow V'}\,\textsc{Tr1-Set} \qquad \dfrac{\vdash t'_1 : \mathtt{o} \Rightarrow u' \qquad \cdots}{\vdash t'_1 : \mathtt{o} \Rightarrow U'}\,\textsc{Tr1-Set}}{\vdash s'\,t'_1 : \top \to \cdots \to \top \to \mathtt{o} \Rightarrow \mathtt{br}\,V'\,U'}\,\textsc{Tr1-App2}}{}\,\textsc{Tr1-App1}}{\vdash s'\,t'_1 \,\cdots\, t'_{n'-1} : \top \to \mathtt{o} \Rightarrow \mathtt{br}\,V'\,U'}\,\textsc{Tr1-App1}}}{\vdash s'\,t'_1 \,\cdots\, t'_{n'} : \mathtt{o} \Rightarrow \mathtt{br}\,V'\,U'}\,\textsc{Tr1-App1}$$

The head of v' is not $\mathtt{br}$ since if it is $\mathtt{br}$, by the same reasoning as above we have some s'' and v'' with

$$\vdash s'' : \mathtt{o} \to \top \to \cdots \to \top \to \mathtt{o} \to \top \to \cdots \to \top \to \mathtt{o} \Rightarrow v''$$

which contradicts the well-formedness condition on types. Hence, the head of v' must be a nullary terminal or a non-terminal.

In the case where the head of v' is a nullary terminal a, by $\vdash s' : \mathtt{o} \to \top \to \cdots \to \top \to \mathtt{o} \Rightarrow v'$, s' is a unary non-terminal and hence $s' = v' = a$ and $n' = 1$. Since v' is a tree, reduction of u goes on u'-side. Thus there exist $p' \leq p$ and π' such that $u' \longrightarrow^{p'} \pi'$ and $\mathtt{br}\,a\,\pi' = \pi$.

If $p' > 0$, by the induction hypothesis for t'_1, there exist t''_1, u''_1, π''_1 and $q''_1 < p'$ such that

$$t'_1 \longrightarrow^*_{\mathcal{G}} t''_1 \qquad \vdash t''_1 : \mathtt{o} \Rightarrow u''_1 \qquad u''_1 \longrightarrow^{q''_1} \pi''_1 \sim_{\mathrm{v}} \pi' \,.$$

Then, we have

$$t = a\,t'_1 \longrightarrow^* a\,t''_1 \qquad \vdash a\,t''_1 : \mathtt{o} \Rightarrow \mathtt{br}\,a\,u''_1 \qquad \mathtt{br}\,a\,u''_1 \longrightarrow^{q''_1} \mathtt{br}\,a\,\pi''_1 \sim_{\mathrm{v}} \mathtt{br}\,a\,\pi' = \pi \,.$$

If $p' = 0$,

$$t = a\,t'_1 \longrightarrow^0 a\,t'_1 \qquad \vdash a\,t'_1 : \mathtt{o} \Rightarrow \mathtt{br}\,a\,u' \qquad \mathtt{br}\,a\,u' \longrightarrow^0 \mathtt{br}\,a\,\pi' = \pi \,.$$

In the case where the head of v' is a non-terminal, let $v' = A_\delta\,U_1 \,\cdots\, U_\ell$. The rule used for

$$\vdash s' : \mathtt{o} \to \top \to \cdots \to \top \to \mathtt{o} \Rightarrow v' (= A_\delta\,U_1 \,\cdots\, U_\ell)$$

is (Tr1-NT) or (Tr1-App1); in the latter case, we have:

$$\frac{\begin{array}{c}\vdash s'' : \delta_{\ell'+1} \wedge \cdots \wedge \delta_\ell \to \mathtt{o} \to \top \to \cdots \to \top \to \mathtt{o} \Rightarrow A_\delta\,U_1 \,\cdots\, U_{\ell'} \quad (\ell' \leq \ell) \\ \vdash t'' : \delta_i \Rightarrow U_i \text{ and } \delta_i \neq \mathtt{o} \text{ (for each } i \in \{\ell'+1, \ldots, \ell\}) \end{array}}{\vdash (s' =) s''\,t'' : \mathtt{o} \to \top \to \cdots \to \top \to \mathtt{o} \Rightarrow A_\delta\,U_1 \,\cdots\, U_\ell}$$

Here if $\mathtt{order}(\delta_i) = 0$ then $\ell' = \ell$. Repeating this reasoning to the function side (i.e., s'') terminates at the case of (Tr1-NT). Thus, there exist m, m', $t''_1, \ldots, t''_{m'}$, $\ell_0, \ldots, \ell_m$ such that

$$\begin{array}{l}
m \leq m' \qquad \ell_0 = 0 \qquad \ell_m = \ell \\
s' = A\, t''_1 \,\cdots\, t''_{m'} \\
\texttt{order}(t''_j) \geq 1 \quad (j \in 1, \dots, m) \qquad \texttt{order}(t''_j) = 0 \quad (j \in m+1, \dots, m') \\
\vdash t''_j : \delta_i \Rightarrow U_i \text{ and } \delta_i \neq \mathsf{o} \qquad (j \in \{1, \dots, m\}, i \in \{\ell_{j-1}+1, \dots, \ell_j\}) \\
\delta = \delta_1 \wedge \cdots \wedge \delta_{\ell_1} \to \cdots \to \delta_{\ell_{m-1}+1} \wedge \cdots \wedge \delta_{\ell_m} \to \top \to \cdots \to \top \to \mathsf{o} \to \top \to \cdots \to \top \to \mathsf{o}
\end{array}$$

For the reduction sequence $u = \texttt{br}\, V'\, U' \longrightarrow^p \pi$, we can assume that $V' = \{v'\}$ for simplicity and that the first reduction of the reduction sequence is on v'. This does not lose generality since we can choose an argument to be reduced arbitrarily. Suppose that v' is reduced by a rule $A_\delta\, x'_1 \,\cdots\, x'_\ell \to v$. Since this is produced by (Tr1-Rule), there is a rule $A\, x^1 \,\cdots\, x^n \to s$ in $\mathcal{G}$ such that

$$\vdash \lambda x^1. \cdots \lambda x^n.s : \delta \Rightarrow \lambda x'_1. \cdots \lambda x'_\ell.v\,.$$

Then we have the following derivation tree:

$$\begin{array}{cl}
x^1 : \delta_1, \dots, x^1 : \delta_{\ell_1}, \dots, x^m : \delta_{\ell_{m-1}}, \dots, x^m : \delta_{\ell_m}, x^{m'+1} : \mathsf{o} \vdash s : \mathsf{o} \Rightarrow v_0 & \\ \hline
 & \textsc{Tr1-Abs1} \\
\vdots & \\ \hline
 & \textsc{Tr1-Abs1} \\
x^1 : \delta_1, \dots, x^1 : \delta_{\ell_1}, \dots, x^m : \delta_{\ell_{m-1}}, \dots, x^m : \delta_{\ell_m}, x^{m'+1} : \mathsf{o} \vdash \lambda x^{m'+2}. \cdots \lambda x^n.s : \delta^{m'+1} \Rightarrow v_0 & \\ \hline
 & \textsc{Tr1-Abs2} \\
x^1 : \delta_1, \dots, x^1 : \delta_{\ell_1}, \dots, x^m : \delta_{\ell_{m-1}}, \dots, x^m : \delta_{\ell_m} \vdash \lambda x^{m'+1}. \cdots \lambda x^n.s : \delta^{m'} \Rightarrow v = [\mathsf{e}/x^{m'+1}_{\mathsf{o}}]v_0 & \\ \hline
 & \textsc{Tr1-Abs1} \\
\vdots & \\ \hline
 & \textsc{Tr1-Abs1} \\
x^1 : \delta_1, \dots, x^1 : \delta_{\ell_1}, \dots, x^m : \delta_{\ell_{m-1}}, \dots, x^m : \delta_{\ell_m} \vdash \lambda x^{m+1}. \cdots \lambda x^n.s : \delta^m \Rightarrow v & \\ \hline
 & \textsc{Tr1-Abs1} \\
\vdots & \\ \hline
 & \textsc{Tr1-Abs1} \\
x^1 : \delta_1, \dots, x^1 : \delta_{\ell_1} \vdash \lambda x^2. \cdots \lambda x^n.s : \delta^1 \Rightarrow \lambda x'_{\ell_1+1}. \cdots \lambda x'_\ell.v & \\ \hline
 & \textsc{Tr1-Abs1} \\
\vdash \lambda x^1. \cdots \lambda x^n.s : \delta \Rightarrow \lambda x'_1. \cdots \lambda x'_\ell.v &
\end{array}$$

where $\delta^0 := \delta$ and δ^j is the codomain type of δ^{j-1} for each $j \leq n$; especially, for each $j \leq m$,

$$\delta^j := \delta_{\ell_j+1} \wedge \cdots \wedge \delta_{\ell_{j+1}} \to \cdots \to \delta_{\ell_{m-1}+1} \wedge \cdots \wedge \delta_{\ell_m} \to \top \to \cdots \to \top \to \mathsf{o} \to \top \to \cdots \to \top \to \mathsf{o}$$

and

$$\delta^{m'} = \mathsf{o} \to \top \to \cdots \to \top \to \mathsf{o}\,.$$

Thus we find that

$$x^j_{\delta_i} = x'_i \qquad (j \leq m, i \in \{\ell_{j-1}+1, \dots, \ell_j\}).$$

Now

$$v' = A_\delta\, U_1 \,\cdots\, U_\ell \longrightarrow [U_i/x'_i]_{i \leq \ell} v = [\mathsf{e}/x^{m'+1}_{\mathsf{o}}][U_i/x'_i]_{i \leq \ell} v_0$$

and hence

$$u = \texttt{br}\, v'\, U' \longrightarrow \texttt{br}\,([\mathsf{e}/x^{m'+1}_{\mathsf{o}}][U_i/x'_i]_{i \leq \ell} v_0)\, U' \longrightarrow^{p-1} \pi \tag{31}$$

while

$$t = s'\, t'_1 \cdots t'_{n'} = A\, t''_1 \cdots t''_{m'}\, t'_1 \cdots t'_{n'} \longrightarrow [t'_j/x^{m'+j}]_{j\le n'}[t''_j/x^j]_{j\le m'}s\,.$$

Recall that $\vdash t'_1 : \mathsf{o} \Rightarrow U'$; hence by Lemma 28, we have

$$\vdash [t'_j/x^{m'+j}]_{j\le n'}[t''_j/x^j]_{j\le m'}s : \mathsf{o} \Rightarrow [U'/x_{\mathsf{o}}^{m'+1}][U_i/x'_i]_{i\le\ell}v_0\,.$$

Thus we define

$$t' := [t'_j/x^{m'+j}]_{j\le n'}[t''_j/x^j]_{j\le m'}s \qquad u' := [U'/x_{\mathsf{o}}^{m'+1}][U_i/x'_i]_{i\le\ell}v_0\,.$$

Now by Lemma 28

$$x^{m'+1} : \mathsf{o} \vdash [t''_j/x^j]_{j\le m}s : \mathsf{o} \Rightarrow [U_i/x'_i]_{i\le\ell}v_0$$

and hence by the key lemma (Lemma 19),

$$u' = [U'/x_{\mathsf{o}}^{m'+1}]([U_i/x'_i]_{i\le\ell}v_0) \sim ([\mathsf{e}/x_{\mathsf{o}}^{m'+1}]([U_i/x'_i]_{i\le\ell}v_0)) * U'$$

but from (31) furthermore we have

$$u' = [U'/x_{\mathsf{o}}^{m'+1}]([U_i/x'_i]_{i\le\ell}v_0) \longrightarrow^{p-1} \pi' \sim_{\mathrm{v}} \pi$$

for some π'.

Case where the head of u is non-terminal: This case is similar to the above case where the head of v' is a non-terminal (and easier in the sense that we do not need the key lemma): replace v', s', $\mathsf{o} \to \top \to \cdots \to \top \to \mathsf{o}$ with u, t, o, respectively. ☐

B Proof of Theorem 9

In this section, we sometimes abbreviate a sequence $t_{1,1} \cdots t_{1,m_1} \cdots t_{k,1} \cdots t_{k,m_k}$ to $\overrightarrow{t_{i,j}}^{i\in\{1,\dots,k\},j\in\{1,\dots,m_i\}}$. We also write $[t_i/x_i]_{i\in\{1,\dots,k\}}$ for $[t_1/x_1,\dots,t_k/x_k]$.

We first prepare some lemmas. First, we show that the transformation preserves typing. We extend $[\![\,\cdot\,]\!]$ to the operation on type environments by: $[\![\Xi]\!] = \{x_\xi : [\![\xi]\!] \mid x : \xi \in \Xi\}$.

▶ **Lemma 30.** *If $\Xi \vdash t : \xi \Rightarrow u$, then $[\![\Xi]\!] \vdash u : [\![\xi]\!]$ holds.*

Proof. This follows by straightforward induction on the derivation of $\Xi \vdash t : \xi \Rightarrow u$. ☐

▶ **Definition 31.** The relations $u \preceq u'$ and $U \preceq U'$ on (extended) terms and term sets are defined by:

$$\frac{h \text{ is a variable, a non-terminal, or a terminal}}{h \preceq h}$$

$$\frac{u \preceq u' \qquad U \preceq U'}{uU \preceq u'U'}$$

$$\frac{\forall v \in U.\exists v' \in U'.v \preceq v'}{U \preceq U'}$$

$$\frac{u \preceq u'}{\lambda x.u \preceq \lambda x.u'}$$

It is clear that the two relations $\preceq$ are pre-order relations.

▶ **Lemma 32** (de-substitution). *Let t be an applicative term. If $\Xi \vdash [s/x]t : \xi \Rightarrow u$, then*

$$\begin{array}{l}\Xi, x:\xi_1,\dots,x:\xi_k \vdash t:\xi \Rightarrow u' \\ u \preceq\in [U_1/x_{\xi_1},\dots,U_k/x_{\xi_k}]u' \\ \Xi \vdash s:\xi_i \Rightarrow U_i \text{ for each } i\in\{1,\dots,k\}\end{array}$$

for some $\xi_1,\dots,\xi_k,U_1,\dots,U_k,u'$. Similarly, if $\Xi \vdash [s/x]t : \xi \Rightarrow U$, then

$$\begin{array}{l}\Xi, x:\xi_1,\dots,x:\xi_k \vdash t:\xi \Rightarrow U' \\ U \preceq [U_1/x_{\xi_1},\dots,U_k/x_{\xi_k}]U' \\ \Xi \vdash s:\xi_i \Rightarrow U_i \text{ for each } i\in\{1,\dots,k\}\end{array}$$

for some $\xi_1,\dots,\xi_k,U_1,\dots,U_k,U'$.

Proof. The proof proceeds by simultaneous induction on the structure of t. We first show that the latter property follows from the former one for the same t. Suppose $\Xi \vdash [s/x]t : \xi \Rightarrow \{u_1,\dots,u_\ell\}$, i.e., $\Xi \vdash [s/x]t : \xi \Rightarrow u_j$ for each $j \in \{1,\dots,\ell\}$. By the former property, we have:

$$\begin{array}{l}\Xi, x:\xi_{j,1},\dots,x:\xi_{j,k_j} \vdash t:\xi \Rightarrow u'_j \\ u_j \preceq\in [U_{j,1}/x_{\xi_{j,1}},\dots,U_{j,k_j}/x_{\xi_{j,k_j}}]u'_j \\ \Xi \vdash s:\xi_{j,i} \Rightarrow U_{j,i} \text{ for each } i\in\{1,\dots,k_j\}\end{array}$$

for each $j \in \{1,\dots,\ell\}$. Let $\{\xi_1,\dots,\xi_k\} = \{\xi_{j,i} \mid j\in\{1,\dots,\ell\}, i\in\{1,\dots,k_j\}\}$, and:

$$U_i = \bigcup_{j,i'}\{U_{j,i'} \mid \xi_{j,i'} = \xi_i\}$$

for each $i \in \{1,\dots,k\}$. Then, the required result holds for $U' = \{u'_1,\dots,u'_\ell\}$.
Now, we show the former property (using the latter property for strict subterms of t).

- Case $t = x$: In this case, we have $\Xi \vdash s : \xi \Rightarrow u$. The result holds for $k = 1, \xi_1 = \xi, U_1 = \{u\}$, and $u' = x_\xi$.
- Case $t = y \neq x$: We have $u = y_\xi$. The result holds for $k = 0$ and $u' = y_\xi$.
- Case where t is a non-terminal or a constant: similar to the previous case.
- Case $t = \mathtt{br}\, t_0\, t_1$. In this case, we have:

$$\begin{array}{l}\Xi \vdash [s/x]t_j : \xi'_j \Rightarrow u_j \text{ for each } j\in\{0,1\} \\ (u,\xi) = \begin{cases}(\mathtt{br}\,u_0\,u_1, \mathtt{o}_+) & \text{if } \xi'_0 = \xi'_1 = \mathtt{o}_+ \\ (u_j, \mathtt{o}_+) & \text{if } \xi'_j = \mathtt{o}_+ \text{ and } \xi'_{1-j} = \mathtt{o}_\epsilon \\ (\mathtt{e}, \mathtt{o}_\epsilon) & \text{if } \xi'_0 = \xi'_1 = \mathtt{o}_\epsilon\end{cases}\end{array}$$

  By applying the induction hypothesis to $\Xi \vdash [s/x]t_j : \xi'_j \Rightarrow u_j$, we obtain:

$$\begin{array}{l}\Xi, x:\xi_{j,1},\dots,x:\xi_{j,k_j} \vdash t_j:\xi'_j \Rightarrow u'_j \\ u'_j \preceq\in [U_{j,1}/x_{\xi_{j,1}},\dots,U_{j,k_j}/x_{\xi_{j,k_j}}]u'_j \\ \Xi \vdash s:\xi_{j,i} \Rightarrow U_{j,i} \text{ for } j\in\{0,1\}, i\in\{1,\dots,k_j\}\end{array}$$

  Let $\{\xi_1,\dots,\xi_k\} = \{\xi_{j,i} \mid j\in\{0,1\}, i\in\{1,\dots,k_j\}\}$, and:

$$U_i = \bigcup_{j,i'}\{U_{j,i'} \mid \xi_{j,i'} = \xi_i\}$$

for each $i \in \{1,\dots,k\}$. Then, we have the required result for

$$u' = \begin{cases} (\mathtt{br}\, u'_0\, u'_1, \mathtt{o}_+) & \text{if } \xi'_0 = \xi'_1 = \mathtt{o}_+ \\ (u'_j, \mathtt{o}_+) & \text{if } \xi'_j = \mathtt{o}_+ \text{ and } \xi'_{1-j} = \mathtt{o}_\epsilon \\ (\mathtt{e}, \mathtt{o}_\epsilon) & \text{if } \xi'_0 = \xi'_1 = \mathtt{o}_\epsilon \end{cases}$$

- Case $t = t_0 t_1$, where the head symbol of t_0 is not $\mathtt{br}$. In this case, we have:

$$\begin{array}{l} u = u_0 V_1 \cdots V_\ell \\ \Xi \vdash [s/x]t_0 : \xi'_1 \wedge \cdots \wedge \xi'_\ell \to \xi \Rightarrow u_0 \\ \Xi \vdash [s/x]t_1 : \xi'_j \Rightarrow V_j \text{ for each } j \in \{1,\dots,\ell\} \end{array}$$

By applying the induction hypothesis, we obtain

$$\begin{array}{l} \Xi, x : \xi_{0,1}, \dots, x : \xi_{0,k_0} \vdash t_0 : \xi'_1 \wedge \cdots \wedge \xi'_\ell \to \xi \Rightarrow u'_0 \\ u_0 \preceq\in [U_{0,1}/x_{\xi_{0,1}}, \dots, U_{0,k_0}/x_{\xi_{0,k_0}}]u'_0 \\ \Xi \vdash s : \xi_{0,i} \Rightarrow U_{0,i} \text{ for each } i \in \{1,\dots,k_0\} \end{array}$$

and

$$\begin{array}{l} \Xi, x : \xi_{j,1}, \dots, x : \xi_{j,k_j} \vdash t_1 : \xi'_j \Rightarrow V'_j \\ V_j \preceq [U_{j,1}/x_{\xi_{j,1}}, \dots, U_{j,k_j}/x_{\xi_{j,k_j}}]V'_j \\ \Xi \vdash s : \xi_{j,i} \Rightarrow U_{j,i} \text{ for each } i \in \{1,\dots,k_j\} \end{array}$$

for each $j \in \{1,\dots,\ell\}$. Let $\{\xi_1,\dots,\xi_k\} = \{\xi_{j,i} \mid j \in \{0,\dots,\ell\}, i \in \{1,\dots,k_j\}\}$, and

$$U_i = \bigcup_{j,i'} \{U_{j,i'} \mid \xi_{j,i'} = \xi_i\}$$

for each $i \in \{1,\dots,k\}$. Then, we have the required result for $u' = u'_0 V'_1 \cdots V'_\ell$.

□

Now we prove that the transformation is a left-to-right backward simulation.

▶ **Lemma 33** (subject expansion). *Suppose $\mathcal{G} \Rightarrow \mathcal{G}'$. If $s \longrightarrow_{\mathcal{G}} s'$ with $\vdash s' : \xi \Rightarrow u'$ and $\xi :: \mathtt{o}$, then there exist u and u'' such that $u \longrightarrow^*_{\mathcal{G}'} u''$ with $\vdash s : \xi \Rightarrow u$ and $u' \preceq u''$.*

Proof. This follows by induction on the structure of s, with case analysis on the head symbol of s.

- Case where s is of the form $A\, s_1 \cdots s_k$: In this case, we have:

$$\begin{array}{l} A\, x_1 \cdots x_k \to t \in \mathcal{G} \\ s' = [s_1/x_1, \dots, s_k/x_k]t \end{array}$$

By Lemma 32 and $\vdash [s_1/x_1,\dots,s_k/x_k]t : \xi \Rightarrow u'$, we have:

$$\begin{array}{l} x_1 : \bigwedge_{j=1,\dots,m_1} \xi_{1,j}, \dots, x_k : \bigwedge_{j=1,\dots,m_k} \xi_{k,j} \vdash t : \xi \Rightarrow v \\ u' \preceq u'' \in [U_{i,j}/(x_i)_{\xi_{i,j}}]_{i\in\{1,\dots,k\}, j\in\{1,\dots,m_i\}} v \\ \vdash s_i : \xi_{i,j} \Rightarrow U_{i,j} \end{array}$$

By the first condition, we have:

$$\begin{array}{l} \vdash (A\, x_1 \cdots x_k \to t) \Rightarrow (A_{\xi'}\, \overrightarrow{(x_i)_{\xi_{i,j}}}^{i\in\{1,\dots,k\}, j\in\{1,\dots,m_i\}} \to v) \\ \vdash A : \xi' \Rightarrow A_{\xi'} \end{array}$$

where $\xi' = \bigwedge_{j=1,\dots,m_1} \xi_{1,j} \to \cdots \to \bigwedge_{j=1,\dots,m_k} \xi_{k,j} \to \xi$. Thus, the required result holds for $u = A_{\xi'}\, \overrightarrow{U_{i,j}}^{i\in\{1,\dots,k\}, j\in\{1,\dots,m_i\}}$ and the above u''.

- Case where s is of the form $\texttt{br}\,s_0\,s_1$: In this case, we have:

$$\begin{array}{l} s' = \texttt{br}\,s'_0\,s'_1 \\ s_i \longrightarrow_{\mathcal{G}} s'_i \qquad s_{1-i} = s'_{1-i} \text{ for some } i \in \{0,1\} \\ \vdash s'_j : \xi_j \Rightarrow u'_j \text{ for each } j \in \{0,1\} \end{array}$$

  By the induction hypothesis, we also have: $\vdash s_i : \xi_i \Rightarrow u_i$ with $u_i \longrightarrow u''_i$ and $u'_i \preceq u''_i$. Let $u_{1-i} = u'_{1-i} = u''_{1-i}$. The required condition holds for

$$(u, u'') = \begin{cases} (\texttt{br}\,u_0\,u_1, \texttt{br}\,u''_0\,u''_1) & \text{if } \xi_0 = \xi_1 = \texttt{o}_+ \\ (u_j, u''_j) & \text{if } \xi_j = \texttt{o}_+ \text{ and } \xi_{1-j} = \texttt{o}_\epsilon \\ (\texttt{e}, \texttt{e}) & \text{if } \xi_0 = \xi_1 = \texttt{o}_\epsilon \end{cases}$$

□

▶ **Lemma 34.** *For applicative terms u and v, if $u \preceq v$ and $u \longrightarrow_{\mathcal{G}'} u'$, then $v \longrightarrow^*_{\mathcal{G}'} v'$ and $u' \preceq v'$ for some v'.*

Proof. This follows by straightforward induction on term u and case analysis on the shape of u.

- Case $u = A\,U_1\,\cdots\,U_k$: In this case $v = A\,V_1\,\cdots\,V_k$ and $U_i \preceq V_i$ for each $i \in \{1,\ldots,k\}$. Since $u = A\,U_1\,\cdots\,U_k \longrightarrow_{\mathcal{G}'} u'$, we have $(A\,x_1\,\cdots\,x_k \longrightarrow u_0) \in \mathcal{G}'$ such that $u' = [U_i/x_i]_{i\in\{1,\ldots,k\}}u_0$. We can show that $U \preceq V$ and $u \preceq v$ implies $[U/x]u \preceq [V/x]v$, by induction on $u \preceq v$. Hence, as required,

$$\begin{array}{l} v = A\,V_1\,\cdots\,V_k \longrightarrow_{\mathcal{G}'} [V_i/x_i]_{i\in\{1,\ldots,k\}}u_0 \\ {[U_i/x_i]_{i\in\{1,\ldots,k\}}u_0} \preceq [V_i/x_i]_{i\in\{1,\ldots,k\}}u_0. \end{array}$$

- Case $u = \texttt{br}\,U_1\,U_2$: We consider only the case where $U_1 = \{u_1\}$ is reduced; the other cases are similar or clear. Thus, $u_1 \longrightarrow_{\mathcal{G}'} u'_1$ and $u' = \texttt{br}\,u'_1\,U_2$ for some u'_1. Since $u \preceq v$, $v = \texttt{br}\,V_1\,V_2$ for some V_1 and V_2 such that $U_i \preceq V_i$ for $i \in \{1,2\}$. As $U_1 \preceq V_1$, $u_1 \preceq v_1$ for some $v_1 \in V_1$. Hence by the induction hypothesis, there exists v'_1 such that $v_1 \longrightarrow_{\mathcal{G}'} v'_1$ and $u'_1 \preceq v'_1$. Then the required result holds for $v' = \texttt{br}\,v'_1\,V_2$.

□

▶ **Lemma 35.** *Suppose $\mathcal{G} \Rightarrow \mathcal{G}'$. For any π, there exist $\xi :: \texttt{o}$ and π' such that $\vdash \pi : \xi \Rightarrow \pi'$ and*

$$\begin{array}{rll} \textbf{\textit{leaves}}(\pi') = \textbf{\textit{leaves}}(\pi){\uparrow_{\texttt{e}}} & \xi = \texttt{o}_+ & (\textit{if } \textbf{\textit{leaves}}(\pi){\uparrow_{\texttt{e}}} \neq \varepsilon) \\ \pi' = \texttt{e} & \xi = \texttt{o}_\epsilon & (\textit{if } \textbf{\textit{leaves}}(\pi){\uparrow_{\texttt{e}}} = \varepsilon). \end{array}$$

Proof. This follows by induction on π:

- Case $\pi = a$: The result follows for $\pi' := a$ and $\xi := \texttt{o}_+$.
- Case $\pi = \texttt{e}$: The result follows for $\pi' := \texttt{e}$ and $\xi := \texttt{o}_\epsilon$.
- Case $\pi = \texttt{br}\,\pi_0\,\pi_1$: By induction hypothesis, we have $\vdash \pi_i : \xi_i \Rightarrow \pi'_i$ such that

$$\begin{array}{rll} \textbf{leaves}(\pi'_i) = \textbf{leaves}(\pi_i){\uparrow_{\texttt{e}}} & \xi_i = \texttt{o}_+ & (\text{if } \textbf{leaves}(\pi_i){\uparrow_{\texttt{e}}} \neq \varepsilon) \\ \pi'_i = \texttt{e} & \xi_i = \texttt{o}_\epsilon & (\text{if } \textbf{leaves}(\pi_i){\uparrow_{\texttt{e}}} = \varepsilon) \end{array}$$

  for each $i = 0, 1$. The result follows for

$$(\pi', \xi) := \begin{cases} (\texttt{br}\,\pi'_0\,\pi'_1, \texttt{o}_+) & \text{if } \xi_0 = \xi_1 = \texttt{o}_+ \\ (\pi'_i, \texttt{o}_+) & \text{if } \xi_i = \texttt{o}_+ \text{ and } \xi_{1-i} = \texttt{o}_\epsilon \\ (\texttt{e}, \texttt{o}_\epsilon) & \text{if } \xi_0 = \xi_1 = \texttt{o}_\epsilon. \end{cases}$$

□

The following lemma shows that the transformation is complete in the sense that for any tree generated by the source grammar, the tree obtained by removing $\mathtt{e}$ can be generated by the target grammar.

▶ **Lemma 36** (completeness of the transformation). *If $\mathcal{G} \Rightarrow \mathcal{G}'$, then $\mathcal{L}_{\mathtt{leaf}}(\mathcal{G})\!\uparrow_{\mathtt{e}} \subseteq \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$.*

Proof. Suppose $w \in \mathcal{L}_{\mathtt{leaf}}(\mathcal{G})\!\uparrow_{\mathtt{e}}$, i.e., $w = \mathbf{leaves}(\pi)\!\uparrow_{\mathtt{e}}$ for some tree π such that $S \longrightarrow^*_{\mathcal{G}} \pi$. By Lemma 35, there exist $\xi :: \mathtt{o}$ and π' such that $\vdash \pi : \xi \Rightarrow \pi'$ and

$$\begin{aligned}\mathbf{leaves}(\pi') &= \mathbf{leaves}(\pi)\!\uparrow_{\mathtt{e}} && (\text{if } \mathbf{leaves}(\pi)\!\uparrow_{\mathtt{e}} \neq \varepsilon)\\ \pi' &= \mathtt{e} && (\text{if } \mathbf{leaves}(\pi)\!\uparrow_{\mathtt{e}} = \varepsilon).\end{aligned}$$

By repeated applications of Lemmas 33 and 34, there exists u such that $S_\xi \longrightarrow^*_{\mathcal{G}'} u$ and $\pi' \preceq u$. We can easily show that for any π and u if $\pi \preceq u$ then $u \longrightarrow^*_{\mathcal{G}'} \pi$ by induction on $\pi \preceq u$; thus $S' \longrightarrow_{\mathcal{G}'} S_\xi \longrightarrow^*_{\mathcal{G}'} u \longrightarrow^*_{\mathcal{G}'} \pi'$. If $w = \mathbf{leaves}(\pi)\!\uparrow_{\mathtt{e}} \neq \varepsilon$, then $(\mathtt{e} \neq)\, w = \mathbf{leaves}(\pi') \in \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$ as required. If $w = \mathbf{leaves}(\pi)\!\uparrow_{\mathtt{e}} = \varepsilon$, then $\mathtt{e} = \mathbf{leaves}(\pi') \in \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}')$ and hence $w = \varepsilon \in \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$ as required. □

We now turn to prove the soundness of the transformation (Lemma 41 below). We again prepare several lemmas.

▶ **Lemma 37** (substitution). *Suppose $x \notin dom(\Xi)$. If $\Xi, x : \xi_1, \ldots, x : \xi_k \vdash t : \xi \Rightarrow u$ and $\Xi \vdash s : \xi_i \Rightarrow U_i$ for each $i \in \{1, \ldots, k\}$, then $\Xi \vdash [s/x]t : \xi \Rightarrow [U_1/x_{\xi_1}, \ldots, U_k/x_{\xi_k}]u$.*

Proof. This follows by induction on the derivation of $\Xi, x : \xi_1, \ldots, x : \xi_k \vdash t : \xi \Rightarrow u$, with case analysis on the last rule used. We discuss only the case for (Tr2-Var); the other cases follow immediately from the induction hypothesis.

- Case (Tr2-Var): In this case,

  $$\begin{aligned}&\Xi, x : \xi_1, \ldots, x : \xi_k = \Xi', y : \xi\\ &t = y \qquad u = y_\xi\end{aligned}$$

  If $x \neq y$, then $[U_1/x_{\xi_1}, \ldots, U_k/x_{\xi_k}]u = \{u\}$. $\Xi \vdash [s/x]y : \xi \Rightarrow u$ follows immediately from $\Xi, x : \xi_1, \ldots, x : \xi_k \vdash t : \xi \Rightarrow u$, as $[s/x]y = y = t$ by the (derived) strengthening rule.
  If $x = y$, then $\xi = \xi_i$ for some i. We have $[U_1/x_{\xi_1}, \ldots, U_k/x_{\xi_k}]u = U_i$. The required result $\Xi \vdash [s/x]t : \xi \Rightarrow U_i$ follows from $[s/x]t = s$ and $\Xi \vdash s : \xi_i \Rightarrow U_i$.
- Cases (Tr2-Const0), (Tr2-Const1), and (Tr2-NT): Similar to the previous case where $x \neq y$.
- Cases (Tr2-Const2) and (Tr2-App): These cases are straightforward from induction hypotheses; we consider only the case (Tr2-App). We have:

  $$\begin{aligned}&t = t_0\, t_1 \qquad u = u_0\, V_1 \cdots V_\ell\\ &\Xi, x : \xi_1, \ldots, x : \xi_k \vdash t_0 : \xi_1 \wedge \cdots \wedge \xi_\ell \to \xi \Rightarrow u_0\\ &\Xi, x : \xi_1, \ldots, x : \xi_k \vdash t_1 : \xi_j \Rightarrow V_j \text{ for each } j \in \{1, \ldots, \ell\}\end{aligned}$$

  By the induction hypotheses and (Tr2-Set), we have:

  $$\begin{aligned}&\Xi \vdash [s/x]t_0 : \xi_1 \wedge \cdots \wedge \xi_\ell \to \xi \Rightarrow [U_1/x_{\xi_1}, \ldots, U_k/x_{\xi_k}]u_0\\ &\Xi \vdash [s/x]t_1 : \xi_j \Rightarrow [U_1/x_{\xi_1}, \ldots, U_k/x_{\xi_k}]V_j\end{aligned}$$

  Hence, we have the required result by (Tr2-App).

□

Next we show that the transformation is a right-to-left forward simulation.

▶ **Lemma 38** (subject reduction). *Suppose $\mathcal{G} \Rightarrow \mathcal{G}'$. If $u \longrightarrow_{\mathcal{G}'} u'$ with $\vdash t : \xi \Rightarrow u$ and $\xi :: \mathtt{o}$, then there exists t' such that $t \longrightarrow^*_{\mathcal{G}} t'$ with $\vdash t' : \xi \Rightarrow u'$.*

Proof. This follows by induction on the derivation of $\vdash t : \xi \Rightarrow u$. We perform case analysis on the shape of t.

- Case $t = \mathtt{br}\, t_0\, t_1$: We first consider the case where t_0 or t_1 has type $\mathtt{o}_\epsilon$. Since $u \longrightarrow_{\mathcal{G}'} u'$, we have:

  $$\begin{array}{l} t = \mathtt{br}\, t_0\, t_1 \\ \xi = \xi_i = \mathtt{o}_+ \qquad \xi_{1-i} = \mathtt{o}_\epsilon \\ \vdash t_i : \mathtt{o}_+ \Rightarrow u \end{array}$$

  for some $i \in \{0,1\}$. By the induction hypothesis, there exists t'_i such that $t_i \longrightarrow^*_{\mathcal{G}} t'_i$ and $\vdash t'_i : \mathtt{o}_+ \Rightarrow u'$. The required result holds for $\mathtt{br}\, t'_0\, t'_1$, where $t'_{1-i} := t_{1-i}$.
  In the other case, for some $i \in \{0,1\}$, we have:

  $$\begin{array}{l} u = \mathtt{br}\, u_0\, u_1 \\ u_i \longrightarrow_{\mathcal{G}'} u'_i \qquad u'_{1-i} = u_{1-i} \\ u' = \mathtt{br}\, u'_0\, u'_1 \\ \vdash t_j : \mathtt{o}_+ \Rightarrow u_j \text{ for each } j \in \{0,1\} \end{array}$$

  By the induction hypothesis, there exists t'_i such that $\vdash t'_i : \mathtt{o}_+ \Rightarrow u'_i$ and $t_i \longrightarrow^*_{\mathcal{G}} t'_i$. Let $t'_{1-i} = t_{1-i}$. Then, the result holds for $t' = \mathtt{br}\, t'_0\, t'_1$.
- $t = A\, t_1\, \cdots\, t_k$: In this case, we have:

  $$\begin{array}{l} u = A_{\xi'}\, U_{1,1}\, \cdots\, U_{1,\ell_1}\, \cdots\, U_{k,1}\, \cdots\, U_{k,\ell_k} \\ \vdash A : \xi' \Rightarrow A_{\xi'} \\ \vdash t_i : \xi_{i,j} \Rightarrow U_{i,j} \text{ for each } i \in \{1,\dots,k\}, j \in \{1,\dots,\ell_i\} \\ \xi' = \bigwedge_j \xi_{1,j} \to \cdots \to \bigwedge_j \xi_{k,j} \to \xi \\ \big(A'_\xi\, (x_1)_{\xi_{1,1}} \cdots (x_1)_{\xi_{1,\ell_1}} \cdots (x_k)_{\xi_{k,1}} \cdots (x_k)_{\xi_{k,\ell_k}} \to u_0\big) \in \mathcal{G}' \\ u' \in [U_{i,j}/(x_i)_{\xi_{i,j}}]_{i \in \{1,\dots,k\}, j \in \{1,\dots,\ell_i\}} u_0 \end{array}$$

  By the condition $\big(A'_\xi\, (x_1)_{\xi_{1,1}} \cdots (x_1)_{\xi_{1,\ell_1}} \cdots (x_k)_{\xi_{k,1}} \cdots (x_k)_{\xi_{k,\ell_k}} \to u_0\big) \in \mathcal{G}'$, we have

  $$\begin{array}{l} (A\, x_1\, \cdots\, x_k \to t_0) \in \mathcal{G} \\ x_1 : \xi_{1,1}, \dots, x_1 : \xi_{1,\ell_1}, \dots, x_k : \xi_{k,1}, \dots, x_k : \xi_{k,\ell_k} \vdash t_0 : \xi \Rightarrow u_0. \end{array}$$

  Let $t' = [t_1/x_1, \dots, t_k/x_k]t_0$. By Lemma 37, we have $\vdash t' : \xi \Rightarrow u'$ and $t \longrightarrow^*_{\mathcal{G}} t'$ as required.

□

▶ **Lemma 39.** *If $\vdash t : \mathtt{o}_\epsilon \Rightarrow u$, then $t \longrightarrow^*_{\mathcal{G}} \pi$ for some π such that $\textbf{leaves}(\pi) \in \mathtt{e}^*$.*

Proof. We define the unary logical relation $\mathcal{R}_\xi$ by:

- $\mathcal{R}_{\mathtt{o}_\epsilon}(t)$ if $t \longrightarrow^*_{\mathcal{G}} \pi$ for some π such that $\textbf{leaves}(\pi) \in \mathtt{e}^*$.
- $\mathcal{R}_{\mathtt{o}_+}(t)$ if $t \longrightarrow^*_{\mathcal{G}} \pi$ for some π such that $\textbf{leaves}(\pi) \notin \mathtt{e}^*$.
- $\mathcal{R}_{\xi_1 \wedge \cdots \wedge \xi_k \to \xi}(t)$ if, whenever $\mathcal{R}_{\xi_i}(s)$ for every $i \in \{1,\dots,k\}$, $\mathcal{R}_\xi(ts)$.

For a substitution $\theta = [s_1/x_1, \dots, s_\ell/x_\ell]$, we write $\theta \models \Xi$ if $\Xi = \{x_i : \xi_{i,j} \mid i \in \{1,\dots,\ell\}, j \in \{1,\dots,m_i\}\}$ and $\forall i \in \{1,\dots,\ell\}. \forall j \in \{1,\dots,m_i\}. \mathcal{R}_{\xi_{i,j}}(s_i)$. We write $\Xi \models t : \xi$ when $\theta \models \Xi$ implies $\mathcal{R}_\xi(\theta t)$. We show that, for every applicative term t, $\Xi \vdash t : \xi \Rightarrow u$ implies $\Xi \models t : \xi$, from which the result follows. The proof proceeds by induction on the derivation of $\Xi \vdash t : \xi \Rightarrow u$, with case analysis on the last rule used.

- Cases for (Tr2-Var), (Tr2-Const0), and (Tr2-Const1): trivial.
- Case for (Tr2-Const2): In this case, we have:

$$t = \mathtt{br}\, t_0\, t_1 \qquad \Xi \vdash t_i : \xi_i \Rightarrow u_i \qquad \xi_i = \begin{cases} \mathtt{o}_+ & (\exists i.\, \xi_i = \mathtt{o}_+) \\ \mathtt{o}_\epsilon & (\forall i.\, \xi_i = \mathtt{o}_\epsilon) \end{cases}$$

  Then, $\Xi \models \mathtt{br}\, t_0\, t_1 : \xi$ follows from the induction hypotheses $\Xi \models t_i : \xi_i$.
- Case for (Tr2-NT): In this case, we have:

$$\begin{array}{l} t = A \qquad \xi = \bigwedge_{j\in\{1,\dots,\ell_1\}} \xi_{1,j} \to \cdots \to \bigwedge_{j\in\{1,\dots,\ell_k\}} \xi_{k,j} \to \xi_0 \qquad \xi_0 = \mathtt{o}_+ \text{ or } \mathtt{o}_\epsilon \\ A\, x_1\, \cdots\, x_k \to t_0 \in \mathcal{G} \\ \{x_i : \xi_{i,j} \mid i \in \{1,\dots,k\}, j \in \{1,\dots,\ell_i\}\} \vdash t_0 : \xi_0 \Rightarrow u \end{array}$$

  To show $\mathcal{R}_\xi(A)$, suppose $\mathcal{R}_{\xi_{i,j}}(s_i)$ for every $i \in \{1,\dots,k\}$ and $j \in \{1,\dots,\ell_i\}$. By applying the induction hypothesis to the last condition, we have

$$\{x_i : \xi_{i,j} \mid i \in \{1,\dots,k\}, j \in \{1,\dots,\ell_i\}\} \models t_0 : \xi_0.$$

  (Note that the derivation for $\{x_i : \xi_{i,j} \mid i \in \{1,\dots,k\}, j \in \{1,\dots,\ell_i\}\} \vdash t_0 : \xi_0 \Rightarrow u$ is a sub-derivation for $\vdash A : \xi \Rightarrow u$; this is the reason why we have included the rightmost premise in the rule (Tr2-NT).) Therefore, we have $\mathcal{R}_{\xi_0}([s_1/x_1, \dots, s_k/x_k]t_0)$, which implies $\mathcal{R}_{\xi_0}(A\, s_1\, \cdots\, s_k)$ as required.
- Case for (Tr2-App): In this case, we have:

$$\begin{array}{l} t = t_0\, t_1 \\ \Xi \vdash t_0 : \xi_1 \wedge \cdots \wedge \xi_k \to \xi \Rightarrow u_0 \\ \Xi \vdash t_1 : \xi_i \Rightarrow U_i \end{array}$$

  Suppose $\theta \models \Xi$. We need to show $\mathcal{R}_\xi(\theta t)$. By applying the induction hypothesis to the transformation judgments for t_0 and t_1, we have $\Xi \models t_0 : \xi_1 \wedge \cdots \wedge \xi_k \to \xi$ and $\Xi \models t_1 : \xi_i$ for every $i \in \{1,\dots,k\}$. Thus, we have also $\mathcal{R}_{\xi_1\wedge\cdots\wedge\xi_k\to\xi}(\theta t_0)$ and $\mathcal{R}_{\xi_i}(\theta t_1)$. Therefore, we have $\mathcal{R}_\xi(\theta t)$ as required.

□

▶ **Lemma 40.** *If $\vdash t : \xi \Rightarrow \pi'$, then there exists π such that $t \longrightarrow^*_{\mathcal{G}} \pi$ and*

$$\begin{array}{lll} \pi' \neq \mathtt{e} & \boldsymbol{leaves}(\pi){\uparrow_{\mathtt{e}}} = \boldsymbol{leaves}(\pi') & (\textit{if } \xi = \mathtt{o}_+) \\ \pi' = \mathtt{e} & \boldsymbol{leaves}(\pi){\uparrow_{\mathtt{e}}} = \varepsilon & (\textit{if } \xi = \mathtt{o}_\epsilon). \end{array}$$

Proof. This follows by induction on the derivation of $\vdash t : \xi \Rightarrow \pi'$. Since the output of transformation is a tree, the last rule used to derive $\vdash t : \xi \Rightarrow \pi'$ must be (Tr2-Const0), (Tr2-Const1), or (Tr2-Const2). The cases for (Tr2-Const0) and (Tr2-Const1) are trivial. If the last rule is (Tr2-Const2), we have: $t = \mathtt{br}\, t_0\, t_1$ with $\vdash t_i : \xi_i \Rightarrow u_i$ for $i \in \{0, 1\}$. We perform case analysis on ξ_0 and ξ_1.

- Case $\xi_0 = \xi_1 = \mathtt{o}_+$: In this case, $\pi' = \mathtt{br}\, u_0\, u_1$ and $\xi = \mathtt{o}_+$. For each $i \in \{0,1\}$, by the induction hypothesis there exists π_i such that $t_i \longrightarrow^*_{\mathcal{G}} \pi_i$ and $\mathbf{leaves}(\pi_i)\!\uparrow_{\mathtt{e}} = \mathbf{leaves}(u_i)$. Thus, we have $t \longrightarrow^*_{\mathcal{G}} \pi$, $\pi' \neq \mathtt{e}$, and $\mathbf{leaves}(\pi)\!\uparrow_{\mathtt{e}} = \mathbf{leaves}(\pi')$ for $\pi = \mathtt{br}\, \pi_0\, \pi_1$.
- Case $\xi_i = \mathtt{o}_+$ and $\xi_{1-i} = \mathtt{o}_\epsilon$ for some $i \in \{0,1\}$. In this case, $\pi' = u_i$ and $\xi = \mathtt{o}_+$. By the induction hypothesis, there exists π_i such that $t_i \longrightarrow^*_{\mathcal{G}} \pi_i$, $\pi' \neq \mathtt{e}$, and $\mathbf{leaves}(\pi_i)\!\uparrow_{\mathtt{e}} = \mathbf{leaves}(\pi')$. By Lemma 39, there exists π_{1-i} such that $t_{1-i} \longrightarrow^*_{\mathcal{G}} \pi_{1-i}$ and $\mathbf{leaves}(\pi_{1-i}) \in \mathtt{e}^*$. Thus, we have $t \longrightarrow^*_{\mathcal{G}} \pi$, $\pi' \neq \mathtt{e}$, and $\mathbf{leaves}(\pi)\!\uparrow_{\mathtt{e}} = \mathbf{leaves}(\pi')$ for $\pi = \mathtt{br}\, \pi_0\, \pi_1$.
- Case $\xi_0 = \xi_1 = \mathtt{o}_\epsilon$. In this case, $\pi' = \mathtt{e}$ and $\xi = \mathtt{o}_\epsilon$. By Lemma 39, for each $i \in \{0,1\}$ there exists π_i such that $t_i \longrightarrow^*_{\mathcal{G}} \pi_i$ and $\mathbf{leaves}(\pi_i) \in \mathtt{e}^*$. Thus, we have $t \longrightarrow^*_{\mathcal{G}} \pi$, $\pi' = \mathtt{e}$, and $\mathbf{leaves}(\pi)\!\uparrow_{\mathtt{e}} = \varepsilon$ for $\pi = \mathtt{br}\, \pi_0\, \pi_1$.

$\square$

We are now ready to prove soundness of the transformation.

▶ **Lemma 41** (soundness). *If $\mathcal{G} \Rightarrow \mathcal{G}'$, then $\mathcal{L}_{\mathtt{leaf}}(\mathcal{G})\!\uparrow_{\mathtt{e}} \supseteq \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$.*

Proof. Suppose $w \in \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$. Then there exist $\xi \in \{\mathtt{o}_+, \mathtt{o}_\epsilon\}$ and π' such that $S_\xi \longrightarrow^*_{\mathcal{G}'} \pi'$ and

$$\begin{aligned} &w = \mathbf{leaves}(\pi') && (\text{if } \pi' \neq \mathtt{e}) \\ &w = \varepsilon && (\text{if } \pi' = \mathtt{e}). \end{aligned}$$

Now $S_\xi \longrightarrow_{\mathcal{G}'} u \longrightarrow^*_{\mathcal{G}'} \pi'$ for some u, and so $(S_\xi \longrightarrow u) \in \mathcal{G}'$. By (Tr2-Rule), there exists t such that $(S \longrightarrow t) \in \mathcal{G}$ and $\vdash t : \xi \Rightarrow u$. Hence, by (Tr2-NT), we have $\vdash S : \xi \Rightarrow S_\xi$. By repeated applications of Lemma 38, we have $S \longrightarrow^*_{\mathcal{G}} t$ and $\vdash t : \xi \Rightarrow \pi'$. By Lemma 40, $\xi = \mathtt{o}_\epsilon$ iff $\pi' = \mathtt{e}$, and we have $t \longrightarrow^*_{\mathcal{G}} \pi$ and $\mathbf{leaves}(\pi)\!\uparrow_{\mathtt{e}} = w$ for some π. Thus, we have $S \longrightarrow^*_{\mathcal{G}} \pi$ and $\mathbf{leaves}(\pi)\!\uparrow_{\mathtt{e}} = w$ as required. $\square$

Proof of Theorem 9.

The fact that $\mathcal{G}'$ is a tree grammar of order at most n follows immediately from Lemma 30. $\mathcal{L}_{\mathtt{leaf}}(\mathcal{G})\!\uparrow_{\mathtt{e}} = \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$ follows immediately from Lemmas 36 and 41. $\square$

C Proof of Theorem 5

First we give a formal definition of the construction for Theorem 5. Let $\mathcal{G}' = (\Sigma', \mathcal{N}', \mathcal{R}', S')$ be an order-n tree grammar such that no word in $\mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$ contains $\mathtt{e}$. We define a grammar $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}, S)$ as:

$$\begin{aligned} \Sigma &:= \{a \mapsto 1 \mid \Sigma'(a) = 0, a \neq \mathtt{e}\} \cup \{\mathtt{e} \mapsto 0\} \\ \mathcal{N} &:= \{A : \kappa^\circ \mid (A : \kappa) \in \mathcal{N}'\} \cup \{E : \mathtt{o} \to \mathtt{o},\ Br : (\mathtt{o} \to \mathtt{o}) \to (\mathtt{o} \to \mathtt{o}) \to (\mathtt{o} \to \mathtt{o}),\ S : \mathtt{o}\} \\ \mathcal{R} &:= \{A\, x_1 \cdots x_\ell\, x \to t^\circ x \mid (A\, x_1 \cdots x_\ell \to t) \in \mathcal{R}'\} \\ &\quad \cup \{E\, x \to x,\ Br\, f\, g\, x \to f\,(g\, x),\ S \to S'\mathtt{e}\} \end{aligned}$$

where E, Br, and S are fresh non-terminals, and $(-)^\circ$ is defined as follows.

$$\begin{aligned} &\mathtt{o}^\circ := \mathtt{o} \to \mathtt{o} \qquad (\kappa_1 \to \kappa_2)^\circ := {\kappa_1}^\circ \to {\kappa_2}^\circ \\ &x^\circ := x \qquad \mathtt{e}^\circ := E \qquad \mathtt{br}^\circ := Br \qquad a^\circ := a \quad (\Sigma'(a) = 0, a \neq \mathtt{e}) \\ &A^\circ := A \qquad (s\, t)^\circ := s^\circ\, t^\circ \end{aligned}$$

If $n = 0$, then the above $\mathcal{G}$ is an order-2 grammar. In this case, any occurrence of $\mathtt{br}$ in $\mathcal{G}'$ must be fully applied, hence we replace $\mathtt{br}^\circ := Br$ in the above definition with

$$(\mathtt{br}\, s\, t)^\circ := A_{s,t}$$

and add a fresh non-terminal $A_{s,t} : \mathtt{o} \to \mathtt{o}$ and a rule

$$A_{s,t}\, x \to s^\circ\, (t^\circ\, x)$$

for each s and t such that $\mathtt{br}\, s\, t$ occurs in the right hand side of some rule of $\mathcal{G}'$. We write this modified grammar as $\mathcal{G}_0$.

▶ **Lemma 42.** *For $(A\, x_1 \cdots x_\ell \to t) \in \mathcal{R}'$ where $\mathcal{N}'(A) = \kappa_1 \to \cdots \to \kappa_\ell \to \mathtt{o}$,*

$$\mathcal{N}, x_1 : {\kappa_1}^\circ, \ldots, x_\ell : {\kappa_\ell}^\circ, x : \mathtt{o} \vdash_{\mathtt{ST}} t^\circ\, x : \mathtt{o}.$$

Proof. By definition we have $\mathcal{N}', x_1 : \kappa_1, \ldots, x_\ell : \kappa_\ell \vdash_{\mathtt{ST}} t : \mathtt{o}$, and we can show by induction on t that $\mathcal{N}', x_1 : \kappa_1, \ldots, x_\ell : \kappa_\ell \vdash_{\mathtt{ST}} t : \kappa$ implies $\mathcal{N}, x_1 : \kappa_1{}^\circ, \ldots, x_\ell : \kappa_\ell{}^\circ \vdash_{\mathtt{ST}} t^\circ : \kappa^\circ$. ◻

By the above lemma, we can show that $\mathcal{G}$ is well-defined as a grammar, and clearly $\mathtt{order}(\kappa^\circ) = \mathtt{order}(\kappa) + 1$; thus $\mathcal{G}$ is an order-$(n+1)$ grammar if $n \geq 1$. When $n = 0$, similarly we can show that $\mathcal{G}_0$ is an order-1 grammar.

In the rest of this section, we show $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$ for $n \geq 0$; note that it is clear that $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}_{\mathtt{w}}(\mathcal{G}_0)$ since the above modification for defining $\mathcal{G}_0$ is just unfolding of the (unique) rule for Br.

To show the goal, we use a logical relation for cpo semantics (cf. e.g., [20] and the proof of Lemma 12 in Appendix A.1). First, recall from Appendix A.1 that we can embed a grammar $\mathcal{G}$ into a λY-calculus, which we write as $\lambda Y_{\mathcal{G}}$, and here we use binary choice operators $+_\kappa : \kappa \to \kappa \to \kappa$ to represent the finite nondeterminism; thus, $\lambda Y_{\mathcal{G}}$ is the λY-calculus that has the terminals of $\mathcal{G}$ and $+$ as constants. We extend the definition of $(-)^\circ$ to $\lambda Y_{\mathcal{G}'}$:

$$(\lambda x : \kappa. t)^\circ := \lambda x : \kappa^\circ. t^\circ \qquad (Y_\kappa)^\circ := Y_{\kappa^\circ} \qquad (+_\kappa)^\circ := +_{\kappa^\circ}$$

For a grammar $\mathcal{G}$, let $\mathbf{Tr}_{\mathcal{G}}$ be the set of all the trees which consist of terminals of $\mathcal{G}$, and we define $[\![-]\!]_{\mathcal{G}}$ as:

$$\begin{aligned} [\![\mathtt{o}]\!]_{\mathcal{G}} &:= (P(\mathbf{Tr}_{\mathcal{G}}), \subseteq) \\ [\![\kappa_1 \to \kappa_2]\!]_{\mathcal{G}} &:= [\![\kappa_1]\!]_{\mathcal{G}} \to [\![\kappa_2]\!]_{\mathcal{G}} \\ [\![a]\!]_{\mathcal{G}}(L_1) \cdots (L_n) &:= \{a\, \pi_1 \cdots \pi_n \mid \pi_i \in L_i\} \in P(\mathbf{Tr}_{\mathcal{G}}) \qquad (\Sigma(a) = n,\ L_i \in P(\mathbf{Tr}_{\mathcal{G}})) \end{aligned}$$

where $A \to B$ is the continuous function space from A to B. The adequacy theorem, i.e., $[\![t]\!]_{\mathcal{G}} = \{\pi \mid t \longrightarrow^* \pi\}$, holds as usual. ($[\![t]\!]_{\mathcal{G}} \supseteq \{\pi \mid t \longrightarrow^* \pi\}$ is obtained by showing that $t \longrightarrow t'$ implies $[\![t]\!]_{\mathcal{G}} \supseteq [\![t']\!]_{\mathcal{G}}$ by induction on the derivation of $t \longrightarrow t'$. The converse immediately follows from the abstraction theorem (i.e., $[\![t]\!]\, R_\kappa\, t$ for any $t \in \mathrm{Term}_\kappa$) of the logical relation R given in the proof of Lemma 12 in Appendix A.1 (cf. [20, Theorem 4.6]), with a trivial modification for the difference that here we are considering trees up to the equality rather than up to $\sim_{\mathrm{v}}$.)

We consider a logical relation $(R'_\kappa)_\kappa$ between two models for $\lambda Y_{\mathcal{G}'}$. One model is the model $[\![-]\!]_{\mathcal{G}'}$ and we define another model $[\![-]\!]^\circ$ for $\lambda Y_{\mathcal{G}'}$, a model reflecting the translation $(-)^\circ$. The interpretation of types is given by:

$$\begin{aligned} [\![\mathtt{o}]\!]^\circ &:= [\![\mathtt{o}^\circ]\!]_{\mathcal{G}} = P(\mathbf{Tr}_{\mathcal{G}}) \to P(\mathbf{Tr}_{\mathcal{G}}) \\ [\![\kappa_1 \to \kappa_2]\!]^\circ &:= [\![\kappa_1]\!]^\circ \to [\![\kappa_2]\!]^\circ. \end{aligned}$$

For a constant c, the interpretation is given as $[\![c]\!]^\circ := [\![c^\circ]\!]_{\mathcal{G}}$; specifically, for terminals,

$$\begin{aligned}
[\![\mathtt{e}]\!]^\circ &:= id : P(\mathbf{Tr}_{\mathcal{G}}) \to P(\mathbf{Tr}_{\mathcal{G}}) \\
[\![a]\!]^\circ &:= (L \mapsto \{a\,\pi \mid \pi \in L\}) : P(\mathbf{Tr}_{\mathcal{G}}) \to P(\mathbf{Tr}_{\mathcal{G}}) \qquad (\Sigma'(a) = 0, a \neq \mathtt{e}) \\
[\![\mathtt{br}]\!]^\circ &:= (f \mapsto (g \mapsto f \circ g)) : [\![\mathtt{o}]\!]^\circ \to [\![\mathtt{o}]\!]^\circ \to [\![\mathtt{o}]\!]^\circ.
\end{aligned}$$

Then, we can easily show that $[\![t^\circ]\!]_{\mathcal{G}} = [\![t]\!]^\circ$ for any term t, by induction on t.

The logical relation $(R'_\kappa)_\kappa$ is determined by $R'_{\mathtt{o}}$, which is defined as:

$$\begin{aligned}
&R'_{\mathtt{o}} \subseteq [\![\mathtt{o}]\!]_{\mathcal{G}'} \times [\![\mathtt{o}]\!]^\circ = P(\mathbf{Tr}_{\mathcal{G}'}) \times (P(\mathbf{Tr}_{\mathcal{G}}) \to P(\mathbf{Tr}_{\mathcal{G}})) \\
&L'\, R'_{\mathtt{o}}\, f \text{ if for any } L \in P(\mathbf{Tr}_{\mathcal{G}}), \\
&\{\mathbf{leaves}(\pi) \mid \pi \in L'\}{\uparrow_{\mathtt{e}}} \cdot \{a_1 \cdots a_n \mid a_1(\cdots(a_n\,\mathtt{e})\cdots) \in L\} \\
&\ = \{a_1 \cdots a_n \mid a_1(\cdots(a_n\,\mathtt{e})\cdots) \in f(L)\}
\end{aligned}$$

where $A \cdot B$ is the concatenation of word languages A and B. If we show the abstraction theorem (i.e., $[\![t]\!]_{\mathcal{G}'}\, R'_\kappa\, [\![t]\!]^\circ$ for any closed term t), then we can show our goal as follows: Since $[\![t]\!]^\circ = [\![t^\circ]\!]_{\mathcal{G}}$, $[\![S']\!]^\circ(\{\mathtt{e}\}) = [\![S'^\circ]\!]_{\mathcal{G}}([\![\mathtt{e}]\!]_{\mathcal{G}}) = [\![S']\!]_{\mathcal{G}}([\![\mathtt{e}]\!]_{\mathcal{G}}) = [\![S]\!]_{\mathcal{G}}$. Now we have $[\![S']\!]_{\mathcal{G}'}\, R'_{\mathtt{o}}\, [\![S']\!]^\circ$, which means—with $L = \{\mathtt{e}\}$ and the adequacy—that $\mathcal{L}_{\mathtt{leaf}}(\mathcal{G}'){\uparrow_{\mathtt{e}}} \cdot \{\varepsilon\} = \mathcal{L}_{\mathtt{w}}(\mathcal{G})$. By the assumption that no word in $\mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$ contains $\mathtt{e}$, we have $\mathcal{L}_{\mathtt{leaf}}(\mathcal{G}'){\uparrow_{\mathtt{e}}} = \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$.

To show the abstraction theorem, we only have to show $[\![c]\!]_{\mathcal{G}'}\, R'\, [\![c]\!]^\circ$ for each constant $c \in \{Y, \mathtt{e}, a, \mathtt{br}, +\}$. The cases of $\mathtt{e}$, a, and $+$ are clear. As usual, the case of Y follows from showing that $R'_{\mathtt{o}}$ is closed under the least upper bound of increasing chains and contains the bottom, which is clear. To show the case of $\mathtt{br}$, suppose that $L'_1\, R'_{\mathtt{o}}\, f_1$ and $L'_2\, R'_{\mathtt{o}}\, f_2$; then, for $L \in P(\mathbf{Tr}_{\mathcal{G}})$, we have:

$$\begin{aligned}
&\{\mathbf{leaves}(\pi) \mid \pi \in [\![\mathtt{br}]\!]_{\mathcal{G}'}(L'_1)(L'_2)\}{\uparrow_{\mathtt{e}}} \cdot \{a_1 \cdots a_n \mid a_1(\cdots(a_n\,\mathtt{e})\cdots) \in L\} \\
&= \{\mathbf{leaves}(\pi) \mid \pi \in L'_1\}{\uparrow_{\mathtt{e}}} \cdot \{\mathbf{leaves}(\pi) \mid \pi \in L'_2\}{\uparrow_{\mathtt{e}}} \cdot \{a_1 \cdots a_n \mid a_1(\cdots(a_n\,\mathtt{e})\cdots) \in L\} \\
&= \{\mathbf{leaves}(\pi) \mid \pi \in L'_1\}{\uparrow_{\mathtt{e}}} \cdot \{a_1 \cdots a_n \mid a_1(\cdots(a_n\,\mathtt{e})\cdots) \in f_2(L)\} \\
&= \{a_1 \cdots a_n \mid a_1(\cdots(a_n\,\mathtt{e})\cdots) \in f_1(f_2(L))\} \\
&= \{a_1 \cdots a_n \mid a_1(\cdots(a_n\,\mathtt{e})\cdots) \in [\![\mathtt{br}]\!]^\circ(f_1)(f_2)(L)\}.
\end{aligned}$$