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, obtained by allowing non-terminals may take trees or 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: $\kappa ::= \mathtt{o} \mid \kappa_1 \to \kappa_2$. The order and arity of a simple type κ, written $\mathtt{order}(\kappa)$ and $\mathtt{ar}(\kappa)$, are defined respectively by:

$$\mathtt{order}(\mathtt{o}) = 0 \qquad \mathtt{order}(\kappa_1 \to \kappa_2) = \max(\mathtt{order}(\kappa_1)+1, \mathtt{order}(\kappa_2))$$
$$\mathtt{ar}(\mathtt{o}) = 0 \qquad \mathtt{ar}(\kappa_1 \to \kappa_2) = 1 + \mathtt{ar}(\kappa_2)$$

The type $\mathtt{o}$ 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.

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

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

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 the case of the frontier language, we treat $\mathtt{e}$ as a special symbol and consider $\mathtt{e} \in \mathcal{L}_{\mathtt{leaf}}(\mathcal{G})$ as an empty word.

We note that the classes of order-0 and order-1 word languages coincides 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 $\mathcal{L}_{\mathtt{leaf}}(\mathcal{G}_2)$ 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 $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}')$.*

The converse of the above theorem also holds: for every order-n tree grammar $\mathcal{G}'$, there exists an order-$(n+1)$ word grammar $\mathcal{G}$ such that $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}')$. We omit the proof since it is trivial. For $n \geq 1$, the grammar $\mathcal{G}$ is obtained by (i) replacing each order-0 constant a with a new non-terminal A_a of type $\mathtt{o} \to \mathtt{o}$, defined by $A_a\, x \to \mathtt{a}\, x$, (ii) replacing the unique binary symbol $\mathtt{br}$ 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 $Br\, f\, g\, x \to f(g\, x)$, and (iii) applying η-expansion to add an order-0 argument. For example, given the grammar $\mathcal{G}_2$ above, the following grammar is obtained:

$$\begin{array}{l} S\,x \to F\,A\,x \qquad S\,x \to F\,B\,x \\ F\,f\,x \to Br\,f\,f\,x \qquad F\,f\,x \to F(Br\,A\,f)\,x \qquad F\,f\,x \to F(Br\,B\,f)\,x \\ A\,x \to \mathtt{a}\,x \qquad B\,x \to \mathtt{b}\,x \qquad Br\,f\,g\,x \to f(g\,x). \end{array}$$

Theorem 4 is proved by two-step grammar transformations, both of which are based on intersection types. In the first step, we first 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 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:

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

Here, $u_0\, u_1$ is interpreted as just a shorthand for $u_0\{u_1\}$. Intuitively, $\{u_1, \dots, u_k\}$ 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:

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

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:

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

where the substitution θu is defined by:

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

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 $\mathcal{G}' = (\{\mathtt{br} \mapsto 2, \mathtt{e} \mapsto 0\} \cup \{a \mapsto 0 \mid \Sigma(a) = 1\}$ such that $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}')\uparrow_{\mathtt{e}}$.

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 $\mathtt{br\ a\ e}$. 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 $\mathtt{br}\ A\ (\mathtt{br\ b\ e})$. 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: $A \to \mathtt{br\ a\ e}$. Thus, the term after the transformation generates the tree $\mathtt{br}\,(\mathtt{br\ a\,e})\,(\mathtt{br\,b\,e})$. 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 $\delta_1, \dots, \delta_k$, 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.

$$\frac{\delta_0 ::_{\mathrm{u}} \kappa \qquad \delta_i ::_{\mathrm{b}} \kappa \text{ (for each } i \in \{1,\dots,k\})}{\delta_0 \wedge \delta_1 \wedge \cdots \wedge \delta_k ::_{\mathrm{u}} \kappa} \qquad \frac{\delta_i ::_{\mathrm{b}} \kappa \text{ (for each } i \in \{1,\dots,k\})}{\delta_1 \wedge \cdots \wedge \delta_k ::_{\mathrm{b}} \kappa}$$

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

Intuitively, unbalanced types describe trees or closures that contain the end of a word (i.e., symbol $\mathtt{e}$). We write $\delta :: \kappa$ if $\delta ::_{\mathrm{b}} \kappa$ or $\delta ::_{\mathrm{u}} \kappa$. 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. 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 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.

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

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

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

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

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

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

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

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:

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

$$\frac{\mathcal{N}' = \{F_\delta : [\![\delta :: \kappa]\!] \mid \mathcal{N}(F) = \kappa \wedge \delta :: \kappa\} \qquad \mathcal{R}' = \{r' \mid \exists r \in \mathcal{R}.r \Rightarrow r'\}}{(\Sigma, \mathcal{N}, \mathcal{R}, S) \Rightarrow (\Sigma, \mathcal{N}', \mathcal{R}', S_{\mathtt{o}})} \qquad (\text{Tr1-Gram})$$

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

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

▶ **Example 5.** 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:

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

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 $(\top \to \mathtt{o}) \wedge (\mathtt{o} \to \mathtt{o})$.

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

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

▶ **Theorem 6.** *Let $\mathcal{G}$ be an order-$(n+1)$ word grammar. If $\mathcal{G} \Rightarrow \mathcal{G}''$, then $\mathcal{G}''$ is an order-n (extended) grammar. 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 to eliminate 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 $\mathtt{e}$, 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 translated 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 translation 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 translation 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:

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

Intuitively, $\mathtt{o}_\epsilon$ describes trees consisting of only $\mathtt{br}$ and $\mathtt{e}$, and $\mathtt{o}_+$ describes trees that have at least one non-$\mathtt{e}$ leaf. 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.

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

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

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

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

$$\frac{\Xi \vdash t : \xi \Rightarrow u_i \text{ (for each } i \in \{1, \ldots, k\}) \qquad k > 0}{\Xi \vdash t : \xi \Rightarrow \{u_1, \ldots, u_k\}} \quad (\textsc{Tr2-Set})$$

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

The transformation of rewriting rules and grammars is defined by:

$$\frac{\emptyset \vdash \lambda x_1.\cdots \lambda x_k.t : \xi \Rightarrow \lambda x_1.\cdots \lambda x_\ell.t' \qquad \xi :: \mathcal{N}(F)}{(F \to \lambda x_1.\cdots \lambda x_k.t) \Rightarrow (F_\xi \to \lambda x_1.\cdots \lambda x_\ell.t')} \quad (\textsc{Tr2-Rule})$$

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

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

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

We explain some key rules. In (Tr2-Var) and (Tr2-NT), 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 $\mathtt{e}$). 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 rightmost premise requires that the body of A 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 rightmost premise is 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 $S' \to S_{\mathtt{o}_\epsilon}, S' \to S_{\mathtt{o}_+}$. The rewriting rule for $S_{\mathtt{o}_\epsilon}$ ($S_{\mathtt{o}_+}$, resp.) 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 7.** 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:

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

It is the same as the grammar obtained in Example 5, except that redundant subscripts on non-terminals and variables have been removed. The body of the rule for A is transformed as follows.

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

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 the following rules would also be generated from $F\,f \to \mathtt{br}\,f\,(\mathtt{br}\,f\,\mathtt{e})$:

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

From $F\,f \to F\,(A\,f)$, many 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 6 and 8.

▶ **Theorem 8.** *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 also an order-n tree grammar, and $\mathcal{L}_{\mathtt{leaf}}(\mathcal{G}) \uparrow_{\mathtt{e}} = \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}')$.*

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 $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}')$. Note that $\mathcal{G}'$ is necessarily safe, since 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 $\mathcal{L}_{\mathtt{w}}(\mathcal{G}'') = \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}')$. By the construction of the backward transformation, $\mathcal{G}''$ is clearly a safe grammar: since the type of each term of $\mathcal{G}'$ is $\mathtt{o} \to \cdots \mathtt{o} \to \mathtt{o}$, the type of the corresponding term of $\mathcal{G}''$ is $(\mathtt{o} \to \mathtt{o}) \to \cdots (\mathtt{o} \to \mathtt{o}) \to (\mathtt{o} \to \mathtt{o})$. 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 contains a word language as a special case) to an order-n tree language, which 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.[4] Thus, it may actually be the case that their translation 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 $\mathcal{L}_{\mathtt{w}}(\mathcal{G})$. 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 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] They cite our work as "On Unsafe Tree and Leaf Languages, in preparation".
[4] This may be because, when the second author of [4] visited us, we have briefly told him the idea of replacing $t_1 t_2$ with $\mathtt{br}\ t_1^{\#}\ t_2^{\#}$ when t_2 is an order-0 term.

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 6

We give a proof of Theorem 6 in Section A.1 after preparing some basic definitions. Lemmas for the proof are given after the proof.

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 $s\, \overrightarrow{t_i}^{i \le n}$. Also, for type $\kappa, \kappa_1, \dots, \kappa_n$, we sometimes write an iterated function type $\kappa_1 \to \cdots \to \kappa_n \to \kappa$ as $\overrightarrow{\kappa_i \to}^{i \le n} \kappa$ and we omit the superscript if it is not important.

A.1 Proof of Theorem 6 and Basic Definitions and Lemmas

The extended terms can be embedded into the simply typed λY-calculus with non-determinism and with the same constant symbols as the terminals (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 value trees as the least congruence (w.r.t. the definition of value 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 π such that $C(\lambda x_1. \dots \lambda x_n.u) \longrightarrow^* \pi$, there exists π' such that $C(\lambda x_1. \dots \lambda x_n.u') \longrightarrow^* \pi'$ 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:

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

We also define $(a_1 \cdots a_n)^{\#}$ inductively by: $\epsilon^{\#} = \mathtt{e}$ and $(as)^{\#} = \mathtt{br}\, a\, s^{\#}$.

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 $\Gamma \vdash t : \delta \Rightarrow u$.

Now we prove Theorem 6, which states: Let $\mathcal{G}$ be an order-$(n+1)$ word grammar. If $\mathcal{G} \Rightarrow \mathcal{G}''$, then $\mathcal{G}''$ is an order-n (extended) grammar. Furthermore, $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}'') \uparrow_{\mathtt{e}}$.

Proof of Theorem 6. By the mutual induction on $\delta ::_{\mathrm{u}} \kappa$ and $\delta ::_{\mathrm{b}} \kappa$, we can show that $\mathtt{order}([\![\delta :: \kappa]\!]) = \mathtt{order}(\kappa) - 1$ if $\mathtt{order}(\kappa) \geq 1$. Also we can check that $\mathcal{G}''$ is well-defined as a grammar; especially, the well-typedness of the right hand side term of each rewriting rule can be proved by Lemma 24, forgetting the type-refinement.

Now we show $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}'') \uparrow_{\mathtt{e}}$. If $a_1 \cdots a_n \in \mathcal{L}_{\mathtt{w}}(\mathcal{G})$, then $S \longrightarrow^*_{\mathcal{G}} a_1(\cdots(a_n(\mathtt{e}))\cdots)$. By Lemma 17, we have $\vdash_{\mathrm{s}} a_1(\cdots(a_n(\mathtt{e}))\cdots) : \mathtt{o} \Rightarrow (a_1 \cdots a_n)^{\#}$. By Lemma 13, we have u such that $\vdash_{\mathrm{s}} S : \mathtt{o} \Rightarrow u$ with $u(\longrightarrow_{\mathcal{G}''}\gtrsim)^*(a_1 \cdots a_n)^{\#}$. By the transformation rule, u must be $S_{\mathtt{o}}$. Thus, we have $S_{\mathtt{o}}(\longrightarrow_{\mathcal{G}''}\gtrsim)^*(a_1 \cdots a_n)^{\#}$, which implies $a_1 \cdots a_n \in \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}'') \uparrow_{\mathtt{e}}$ as required.

Conversely, suppose $a_1 \cdots a_n \in \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}'') \uparrow_{\mathtt{e}}$. Then $S_{\mathtt{o}} \longrightarrow^*_{\mathcal{G}''} \pi$ with $\mathbf{leaves}(\pi) \uparrow_{\mathtt{e}} = a_1 \cdots a_n \mathtt{e}$. By repeating Lemma 18, we have $S \longrightarrow^*_{\mathcal{G}} s$ and $\vdash s : \mathtt{o} \Rightarrow \pi'$ with $\pi' \sim_{\mathrm{v}} \pi$. By Lemma 20, $s = a_1(\cdots(a_n(\mathtt{e}))\cdots)$. Thus, we have $a_1 \cdots a_n \in \mathcal{L}_{\mathtt{w}}(\mathcal{G})$ as required. ◻

▶ **Lemma 9** (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 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 value 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 interpretations of function types are given by the usual continuous function spaces. The interpretations of the constants are given as follows:

$$\begin{aligned} [\![\mathtt{br}]\!](D_1, D_2) &:= \{[\mathtt{br}\,\pi_1\,\pi_2]_{\sim_{\mathrm{v}}} \mid [\pi_i]_{\sim_{\mathrm{v}}} \in D_i\} \qquad (D_1, D_2 \in P(\mathbb{V})) \\ [\![\mathtt{a}]\!] &:= \{[\mathtt{a}]_{\sim_{\mathrm{v}}}\} \qquad (\Sigma(a) = 0). \end{aligned}$$

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:

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

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 10.** *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 a straightforward induction on t. ◻

▶ **Lemma 11.** *Given $\Gamma \vdash t : \delta \Rightarrow u$, the following holds.*

1. *For unbalanced types δ' and δ'', if $x_{\delta'}$ and $y_{\delta''}$ occur in u or if $x : \delta', y : \delta'' \in \Gamma$, then $x = y$ and $\delta' = \delta''$. Moreover, for any x and unbalanced δ', $x_{\delta'}$ occurs in u iff Γ has $x : \delta'$.*
2. *For $y \in \mathbf{FV}(u)$ there exists $x : \delta' \in \Gamma$ such that $y = x_{\delta'}$.*

Proof. By a straightforward induction on t. ☐

▶ **Lemma 12.** *1. For any u, u_1, u_2, and u_3,*

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

2. For any π_1 and π_2,

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

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

A.2 Lemmas for Forward Direction

The following lemma states that the transformation relation (up to $\mathtt{e}$-observational preorder) is a left-to-right backward simulation relation.

▶ **Lemma 13** (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 rule used last 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, 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 with $\vdash_{\mathrm{s}} t_1 : \mathtt{o} \Rightarrow u_1$ 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 rule used last for $t \longrightarrow_{\mathcal{G}} t'$ be

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

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 16, for each $j = n, \ldots, 1$, there exist $\Gamma^j, k^j, (\delta_i^j)_{i \le k^j}, (U_i^j)_{i \le k^j}, v^j$ such that

$$\begin{aligned}
\Gamma^j &= (\Gamma^{j+1}, x_j : \delta_1^j, \ldots, x_j : \delta_{k^j}^j) \\
&\vdash_{\mathrm{s}} t_j : \delta_i^j \Rightarrow U_i^j \qquad (i \le k^j) \\
\Gamma^j &\vdash_{\mathrm{s}} [t_{j'}/x_{j'}]_{j' \le j-1} s : \mathtt{o} \Rightarrow v^j \\
v^{j+1} &\lesssim [U_i^j / x_{\delta_i^j}]_{i \le k^j} v^j \,.
\end{aligned}$$

By Lemmas 14 and 15, since $v^2 \lesssim [U^1_i/x_{\delta^1_i}]_{i\le k^1} v^1$, we have $v^3 \lesssim [U^2_i/x_{\delta^2_i}]_{i\le k^2} v^2 \lesssim ([U^1_i/x_{\delta^1_i}]_{i\le k^1} \cup [U^2_i/x_{\delta^2_i}]_{i\le k^2}) v^1$. Iterating this reasoning, we have

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

Now we have $\Gamma^1 \vdash_{\mathrm{s}} [t_{j'}/x_{j'}]_{j'\le 0} s : \delta \Rightarrow v^1$, i.e.,

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

We define

$$v^0 := \begin{cases} [\mathsf{e}/(x_j)_{\mathsf{o}}] v^1 & (k_j = 1 \text{ for some (unique) } j \in \{m+1, \ldots, n\}) \\ v^1 & (\text{otherwise}). \end{cases} \tag{2}$$

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

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

and by iterating Tr1-Abs1, we have

$$\vdash_{\mathrm{s}} \lambda x_1 . \ldots \lambda x_n . s : \wedge_{i\le k^1} \delta^1_i \to \cdots \to \wedge_{i\le k^n} \delta^n_i \to \mathsf{o} \Rightarrow$$
$$\lambda x_{1\delta^1_1} . \ldots \lambda x_{1\delta^1_{k^1}} . \ldots \lambda x_{m\delta^m_1} . \ldots \lambda x_{m\delta^m_{k^m}} . v^0 \,.$$

Hence, by Tr1-Rule, we have

$$\vdash_{\mathrm{s}} (A\, x_1 \ldots\, x_n \longrightarrow_{\mathcal{G}} s) \Rightarrow (A_\delta\, \overrightarrow{x_{1\delta^1_i}}^{i\le k^1} \cdots \overrightarrow{x_{m\delta^m_i}}^{i\le k^m} \longrightarrow_{\mathcal{G}''} v^0) \tag{3}$$

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 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

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

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

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

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

$$\begin{aligned} & (A_\delta\, \overrightarrow{U^1_i}^{i\le k^1} \cdots \overrightarrow{U^m_i}^{i\le k^m}) * U^j_1 \\ \longrightarrow_{\mathcal{G}''}\ & \big(([U^1_i/x_{\delta^1_i}]_{i\le k^1} \cup \cdots \cup [U^m_i/x_{\delta^m_i}]_{i\le k^m}) v^0\big) * U^j_1 && \text{(by (3))} \\ =\ & \big([\mathsf{e}/(x_j)_{\mathsf{o}}]([U^1_i/x_{\delta^1_i}]_{i\le k^1} \cup \cdots \cup [U^m_i/x_{\delta^m_i}]_{i\le k^m}) v^1\big) * U^j_1 && \text{(by (2))} \\ \gtrsim\ & \big([\mathsf{e}/(x_j)_{\mathsf{o}}] v^{m+1}\big) * U^j_1 && \text{(by (1))} \\ \sim\ & [U^j_1/(x_j)_{\mathsf{o}}] v^{m+1} && \text{(Lemma 21)} \\ \gtrsim\ & v^{n+1} = u'. \end{aligned}$$

In the other case, we have

$$\begin{aligned}
&A_\delta\, \overrightarrow{U^1_i}^{i\le k^1} \cdots \overrightarrow{U^m_i}^{i\le k^m} \\
\longrightarrow_{\mathcal{G}''}\ & ([U^1_i/x_{\delta^1_i}]_{i\le k^1} \cup \cdots \cup [U^m_i/x_{\delta^m_i}]_{i\le k^m})v^0 && \text{(by (3))}\\
=\ & ([U^1_i/x_{\delta^1_i}]_{i\le k^1} \cup \cdots \cup [U^m_i/x_{\delta^m_i}]_{i\le k^m})v^1 && \text{(by (2))}\\
\gtrsim\ & v^{m+1} && \text{(by (1))}\\
=\ & v^{n+1} = u'.
\end{aligned}$$

◻

▶ **Lemma 14.** *If $u \lesssim u'$, then $\theta u \lesssim \theta u'$.*

Proof. The proof is trivial by transforming the substitution into a context. ◻

▶ **Lemma 15.**

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

Proof. The proof is given by a straightforward induction on u. ◻

▶ **Lemma 16** (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 $k \ge 0$, $(\delta_i)_{i\le k}$, $(U_i)_{i\le k}$, and $v^\bullet$ such that*

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

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

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

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

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

$$\frac{\begin{array}{c}\Gamma'_0 \vdash_{\mathrm{s}} [t/x]s' : \delta'_1 \wedge \cdots \wedge \delta'_{k'} \to \delta \Rightarrow v' \\ \Gamma'_i \vdash_{\mathrm{s}} [t/x]t' : \delta'_i \Rightarrow U'_i \ \text{(for each } i \in \{1,\dots,k'\})\end{array}}{\Gamma'_0 \cup \Gamma'_1 \cup \cdots \cup \Gamma'_{k'} \vdash_{\mathrm{s}} ([t/x]s')\,([t/x]t') : \delta \Rightarrow \begin{cases} v' U'_1 \cdots U'_{k'} & (\mathtt{order}(t') \ge 1 \vee k' = 0) \\ v' * U'_1 & (\mathtt{order}(t') = 0 \wedge k' = 1)\end{cases}}$$

where

$$\begin{aligned}
\Gamma &= \Gamma'_0 \cup \Gamma'_1 \cup \cdots \cup \Gamma'_{k'}\\
s &= s' t'\\
v &= \begin{cases} v' U'_1 \cdots U'_{k'} & (\mathtt{order}(t') \ge 1 \vee k' = 0) \\ v' * U'_1 & (\mathtt{order}(t') = 0 \wedge k' = 1)\,. \end{cases}
\end{aligned}$$

By Tr1-SetS,

$$\frac{\begin{array}{c}\Gamma'_i \vdash_{\mathrm{s}} [t/x]t' : \delta'_i \Rightarrow u'_{jh} \ \text{(for each } h \in \{1,\dots,k_j\}) \\ k = 1 \text{ if } \delta'_i \text{ is unbalanced}\end{array}}{\Gamma'_i \vdash_{\mathrm{s}} [t/x]t' : \delta'_i \Rightarrow \{u'_{j1},\dots,u'_{jk_j}\} (= U'_i)}$$

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

$$\Gamma_0', x:\delta_1^0,\dots,x:\delta_{k^0}^0 \vdash_{\mathrm{s}} s' : \delta_1' \wedge\cdots\wedge \delta_{k'}' \to \delta \Rightarrow v'^{\bullet}$$
$$\vdash_{\mathrm{s}} t : \delta_i^0 \Rightarrow U_i^0 \quad (i\leq k^0)$$
for each $i \leq k^0$, if δ_i^0 is unbalanced, U_i^0 is a singleton
$$v' \lesssim [U_i^0/x_{\delta_i^0}]_{i\leq k^0} v'^{\bullet}$$

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 $u_{jh}'^{\bullet}$ such that

$$\Gamma_i', x:\delta_1^{jh},\dots,x:\delta_{k^{jh}}^{jh} \vdash_{\mathrm{s}} t' : \delta_i' \Rightarrow u_{jh}'^{\bullet}$$
$$\vdash_{\mathrm{s}} t : \delta_i^{jh} \Rightarrow U_i^{jh} \quad (i\leq k^{jh})$$
for each $i \leq k^{jh}$, if δ_i^{jh} is unbalanced, U_i^{jh} is a singleton
$$u_{jh}' \lesssim [U_i^{jh}/x_{\delta_i^{jh}}]_{i\leq k^{jh}} u_{jh}'^{\bullet}\,.$$

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

$$\frac{\Gamma_i', x:\delta_1^{jh},\dots,x:\delta_{k^{jh}}^{jh} \vdash_{\mathrm{s}} t' : \delta_i' \Rightarrow u_{jh}'^{\bullet} \text{ (for each } h\in\{1,\dots,k_j\}) \quad k=1 \text{ if } \delta_i' \text{ is unbalanced}}{\Gamma_i' \cup \{x:\delta_i^{jh} \mid h\leq k_j, i\leq k^{jh}\} \vdash_{\mathrm{s}} t' : \delta_i' \Rightarrow \{u_{jh}'^{\bullet} \mid h \leq k_j\}}$$

Here the singleton condition is satisfied from well-formedness of types. We define

$$U_j'^{\bullet} := \{u_{jh}'^{\bullet} \mid h \leq k_j\}\,.$$

By Tr1-App1 or Tr1-App2, 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

$$v^{\bullet} := \begin{cases} v'^{\bullet}\,\overrightarrow{U_j'^{\bullet}}^{j\leq k'} & (\mathtt{order}(t') \geq 1 \vee k'=0) \\ v'^{\bullet} * U_1'^{\bullet} & (\mathtt{order}(t') = 0 \wedge k'=1) \end{cases}$$

Now we define k and $(\delta_i)_{i\leq k}$ by the following

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

Also, for each $i \leq k^0$, we define $\langle\!|i|\!\rangle \leq k$ such a number that $\delta_{\langle\!|i|\!\rangle} = \delta_i^0$ and for each $j \leq k'$, $h \leq K_j$, and $i \leq k^{jh}$, we define $\langle\!|j,h,i|\!\rangle \leq k$ such a number that $\delta_{\langle\!|j,h,i|\!\rangle} = \delta_i^{jh}$. 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 Tr1-SetS, for each $i \leq k$ we have $\vdash_{\mathrm{s}} t : \delta_i \Rightarrow U_i$. The singleton condition can be shown by using the well-formedness of types.

Now we show $v \lesssim [U_i/x_{\delta_i}]_{i\leq k} v^{\bullet}$.

Case where $\mathtt{order}(t') \geq 1 \vee k' = 0$:

$$\begin{aligned} v &= v'\,\overrightarrow{U_j'}^{j\leq k'} \\ &\lesssim ([U_i^0/\mathsf{e}]_{i\leq k^0} v'^{\bullet})\,\overrightarrow{\{[U_i^{jh}/\mathsf{e}]_{i\leq k^{jh}} u_{jh}'^{\bullet} | h\leq k_j\}}^{j\leq k'} \\ &\lesssim [U_i/\mathsf{e}]_{i\leq k}(v'^{\bullet}\,\overrightarrow{\{u_{jh}'^{\bullet} | h\leq k_j\}}^{j\leq k'}) \\ &= [U_i/x_{\delta_i}]_{i\leq k} v^{\bullet} \end{aligned}$$

Case where $\texttt{order}(t') = 0 \land k' = 1$:

$$\begin{aligned} v &= v' * U'_1 \\ &\lesssim ([U^0_i/x_{\delta^0_i}]_{i \le k^0} v'^{\bullet}) * ([U^{11}_i/x_{\delta^{11}_i}]_{i \le k^{11}} u'^{\bullet}_{11}) \\ &\lesssim [U_i/x_{\delta_i}]_{i \le k} v^{\bullet} \end{aligned}$$

□

▶ **Lemma 17.** $\vdash_{\mathrm{s}} a_1(a_2(\cdots(a_n\,\mathsf{e})\cdots)) : \mathsf{o} \Rightarrow (a_1 \cdots a_n)^{\#}$.

Proof. This follows by straightforward induction on n. □

A.3 Lemmas for Backward Direction

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

▶ **Lemma 18.** *Given $u \longrightarrow^p_{\mathcal{G}''} \pi$ where $p > 0$ and $\vdash t : \mathsf{o} \Rightarrow u$, there exist t', u', π' and $q < p$ such that $t \longrightarrow^*_{\mathcal{G}} t'$, $\vdash t' : \mathsf{o} \Rightarrow u'$, 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 $\texttt{br}$ or a non-terminal.

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

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

or Tr1-App2:

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

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

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

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

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

which is a contradiction from well-formedness. Hence, the head of v' is nullary terminal a or a non-terminal.

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

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:

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

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

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

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

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

Then we have the following derivation tree:

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

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

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

and

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

Thus we find that

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

Now

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

and hence

$$u = \mathtt{br}\, v'\, U' \longrightarrow \mathtt{br}\,([\mathtt{e}/x^{m'+1}_{\mathtt{o}}][U_i/x'_i]_{i \leq \ell} v_0)\, U' \longrightarrow^{p-1} \pi \tag{4}$$

while

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

Recall that $\vdash t'_1 : \mathtt{o} \Rightarrow U'$; hence by Lemma 19, we have

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

Thus we define

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

Now by Lemma 19

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

and hence by the key lemma (Lemma 21),

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

but from (4) furthermore we have

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

for some π'.

Case where the head of u is non-terminal: This case can be proved similarly to the above case that the head of v' is a non-terminal (and more easily: we do not need the key lemma).y
$\square$

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

▶ **Lemma 19** (substitution). *Given $\Gamma, x : \delta_1, \ldots, x : \delta_k \vdash s : \delta \Rightarrow v$ where $x \notin \mathit{dom}(\Gamma)$ and $k \geq 0$, and given $\vdash t : \delta_i \Rightarrow U_i$ for each $i \leq k$, we have*

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

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:

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

We have

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

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

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

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

$$\Gamma'_0\setminus x \vdash [t/x]s' : \delta'_1\wedge\cdots\wedge\delta'_{k'}\to\delta \Rightarrow [U_i/x_{\delta_i}]_{i\in\langle\Gamma'_0\rangle}v' \tag{5}$$

$$\Gamma'_j\setminus x \vdash [t/x]t' : \delta'_j \Rightarrow [U_i/x_{\delta_i}]_{i\in\langle\Gamma'_j\rangle}u'_{jh} \qquad (j\in\{1,\dots,k'\}, h\le k_j) \tag{6}$$

and by using Tr1-Set, from (6), we have

$$\Gamma'_j\setminus x \vdash [t/x]t' : \delta'_j \Rightarrow \cup_{h\le k_j}[U_i/x_{\delta_i}]_{i\in\langle\Gamma'_j\rangle}u'_{jh} \qquad (j\in\{1,\dots,k'\}). \tag{7}$$

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 latter equation is shown by Lemma 11-2. For any $v'^{\bullet}\in[U_i/x_{\delta_i}]_{i\in\langle\Gamma'_0\rangle}v'$, by Tr1-Set and (5), 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 (7), we have

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

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,

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

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

$$\Gamma'_0 \setminus x \vdash [t/x]s' : \mathtt{o} \to \delta \Rightarrow [U_i/x_{\delta_i}]_{i\in\langle\Gamma'_0\rangle} v'_h \qquad (h \le k_0) \tag{8}$$

$$\Gamma'_1 \setminus x \vdash [t/x]t' : \mathtt{o} \Rightarrow [U_i/x_{\delta_i}]_{i\in\langle\Gamma'_1\rangle} u'_h \qquad (h \le k_1) \tag{9}$$

and by using Tr1-Set (going and back), from (8) and (9), 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. ☐

▶ **Lemma 20.** *Let t be an applicative term. If $\vdash t : \mathtt{o} \Rightarrow \pi$ (where π consists of only terminals), then $t = a_1(a_2(\cdots(a_n\,\mathtt{e})))$, with $(a_1 a_2 \cdots a_n)^{\#} = \pi$.*

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 $(a_2 \cdots a_n)^\# = \pi_2$. Thus, we have $t = a_1(a_2(\cdots(a_n\,\mathtt{e})))$, with $(a_1 a_2 \cdots a_n)^\# = \pi$ as required.

□

A.4 The Key Lemma

▶ **Lemma 21.** *Given $x : \mathtt{o} \vdash s : \delta \Rightarrow v$ where $\mathtt{order}(\delta) \leq 1$ and $\vdash t : \mathtt{o} \Rightarrow U$,*

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

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

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

there exists π' such that

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

The above lemma is the key of the proof of Theorem 6. 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 may occur only at the rightmost position of a tree; $\mathtt{o}$ 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 types occurring below are well-formed.

A type environment Φ is a set of type bindings of the form $x : \rho$. 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, whenever $x : \rho \in \Phi_1 \cap \Phi_2$, ρ is balanced. We write $\mathbf{bal}(\Phi)$ and say Φ is *balanced* if ρ is balanced for every $x : \rho \in \Phi$.

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:

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

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

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

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

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

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

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

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

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

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

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

It is easy to see that if $\Phi, x : \rho \vdash u : \rho'$, then $\rho \to \rho'$ is well-formed, by the induction on the derivation.

We prepare some lemmas to prove the key lemma (Lemma 21).

▶ **Lemma 22** (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 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 $x' : \rho' \in \Phi_0$: Similar to the other cases.

◻

▶ **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 $(\Gamma)^\sharp_{\mathsf{o}} \vdash u : (\delta)^\sharp_{\mathsf{o}}$.
3. Given $\Gamma, x : \mathsf{o} \vdash t : \delta \Rightarrow u$, we have $(\Gamma)^\sharp_{\mathsf{o}} \vdash [\mathsf{e}/x_{\mathsf{o}}]u : (\delta)^\sharp_{\mathsf{o}}$.

Proof. Case of item 1: The proof is given by a straightforward induction on u; the base case is trivial, as every terminal and non-terminal has a typing rule for having a type of the form $(\delta)^\sharp_{\mathsf{o}}$. 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 24, $(\Gamma)^\sharp \vdash u : (\delta)^\sharp$. By item 1, we have $(\Gamma)^\sharp_{\mathsf{o}} \vdash u : (\delta)^\sharp_{\mathsf{o}}$.

Case of item 3: By item 2, we have $(\Gamma)^\sharp_{\mathsf{o}}, x_{\mathsf{o}} : \mathsf{o} \vdash u : (\delta)^\sharp_{\mathsf{o}}$. Since $\vdash \mathsf{e} : \mathsf{o}$, by Lemma 22, we have $(\Gamma)^\sharp_{\mathsf{o}} \vdash [\mathsf{e}/x_{\mathsf{o}}]u : (\delta)^\sharp_{\mathsf{o}}$. ◻

▶ **Lemma 24.** *For any $\Gamma \vdash s : \delta \Rightarrow v$, we have $(\Gamma)^\sharp \vdash v : (\delta)^\sharp$.*

Proof. The proof is a 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 is

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

  and this is obtained by RTy-Var.
- Case of Tr1-Const0:

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

  The goal is

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

  and this 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 is

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

and this is obtained by RTy-Alph.

- Case of Tr1-NT:

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

The goal is

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

and this 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

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

where the latter are obtained through Tr1-Set and RTy-Set. The goal is

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

and this 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

$$\begin{array}{l}(\Gamma_0)^\sharp \vdash V : \mathtt{o} \\ (\Gamma_1)^\sharp \vdash U : \mathtt{oR}.\end{array}$$

The goal is

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

and this 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 is

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

  and this is obtained by RTy-Set.

□

▶ **Lemma 25** (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 $U_1 = u_1 \longrightarrow u_1'$ 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 ($x_{\mathtt{o}} : \mathtt{oR} \vdash u_1 : \mathtt{o}$), this implies $\mathtt{oR} \to \mathtt{o}$ is well-formed, which is 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 $u = A_\delta\, U_1 \cdots U_\ell$, $u' \in [U_i/x_i']_{i \leq \ell} v$, and the rule used for $u \to u'$ be $A_\delta\, x_1' \cdots x_\ell' \longrightarrow v$. Suppose

$$\delta = \wedge_{i \leq k_1} \delta_i^1 \to \cdots \to \wedge_{i \leq k_m} \delta_i^m \to \delta^0$$
$$\delta^0 = \wedge_{i \leq k_{m+1}} \delta_i^{m+1} \to \cdots \to \wedge_{i \leq k_n} \delta_i^n \to \mathtt{o}$$

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

Let

$$\Phi := (x_{\mathtt{o}} : \mathtt{oR}) \quad \rho := \mathtt{oR} \quad \text{(in the case of item 1)}$$
$$\Phi := \emptyset \qquad \rho := \mathtt{o} \quad \text{(in the case of item 2).}$$

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

$$\vdash A_\delta : \rho_1 \to \cdots \to \rho_\ell \to \rho \tag{10}$$
$$\Phi_i \vdash U_i : \rho_i \quad (i \leq \ell) \tag{11}$$
$$\Phi = \Phi_1 \cup \cdots \cup \Phi_\ell\,. \tag{12}$$

The rule used for (10) is RTy-NtRight or RTy-NtALL: in the former case, we have

$$\begin{aligned}&\rho_1 \to \cdots \to \rho_\ell \to \rho = (\delta)^\sharp \\ &= (\delta^1_1)^\sharp \to \cdots \to (\delta^1_{k_1})^\sharp \to \cdots \to (\delta^m_1)^\sharp \to \cdots \to (\delta^m_{k_m})^\sharp \to (\delta^0)^\sharp\end{aligned}$$

i.e.,

$$(\rho_1, \dots, \rho_\ell) = \big((\delta^1_1)^\sharp, \dots, (\delta^1_{k_1})^\sharp, \dots, (\delta^m_1)^\sharp, \dots, (\delta^m_{k_m})^\sharp\big) \tag{13}$$

$$\rho = (\delta^0)^\sharp\,. \tag{14}$$

In the latter case, similarly we have

$$(\rho_1, \dots, \rho_\ell) = \big((\delta^1_1)^\sharp_{\mathtt{o}}, \dots, (\delta^1_{k_1})^\sharp_{\mathtt{o}}, \dots, (\delta^m_1)^\sharp_{\mathtt{o}}, \dots, (\delta^m_{k_m})^\sharp_{\mathtt{o}}\big) \tag{15}$$

$$\rho = (\delta^0)^\sharp_{\mathtt{o}}\,. \tag{16}$$

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

$$x_1 : \delta^1_1, \dots, x_1 : \delta^1_{k_1}, \dots, x_n : \delta^n_1, \dots, x_n : \delta^n_{k_n} \vdash s : \mathtt{o} \Rightarrow v' \tag{17}$$

$$v = v' \qquad \text{(when } \delta^0 \text{ is unbalanced)} \tag{18}$$

$$v = [\mathtt{e}/(x_{j_0})_{\mathtt{o}}]v' \quad \text{(when } \delta^0 \text{ is balanced)} \tag{19}$$

$$\big(x'_1, \dots, x'_\ell\big) = \big((x_1)_{\delta^1_1}, \dots, (x_1)_{\delta^1_{k_1}}, \dots, (x_m)_{\delta^m_1}, \dots, (x_m)_{\delta^m_{k_m}}\big)\,. \tag{20}$$

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

Case of item 1: Since $\rho = \mathtt{oR}$, we have (13) and (14). By (14), δ^0 is unbalanced, and so we have (18). By (17) and Lemma 24 with (13), (18), and (20), we have

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

Then, by (11), (12), and Lemma 22, 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

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

either by using (17), Lemma 23-2, and (18) when δ^0 is unbalanced, or by using (17), Lemma 23-3, and (19) when δ^0 is balanced. By (11) and Lemma 23-1, we have

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

By either (13) or (15), we have

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

Hence, by Lemma 22,

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

and by RTY-SET, we have

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

as required. ☐

Proof of Lemma 21. By the assumption $x : \mathtt{o} \vdash s : \delta \Rightarrow v$, v contains free variable $x_{\mathtt{o}}$, where $\mathtt{o}$ is unbalanced. Suppose $v \longrightarrow^* v' \not\longrightarrow$. Let us consider the following contexts:

$$E ::= [\,] \mid \pi * E \mid E * \pi$$

Now $v' = E[x_{\mathtt{o}}]$ for some E for the following reasons: Since $x_{\mathtt{o}}$ is treated as a linear variable, by Lemma 25, $x_{\mathtt{o}}$ must occur in v' just once. Now we show that, for any v' such that v' can no longer be reduced and $\mathbf{FV}(v') = \{x_{\mathtt{o}}\}$, there exists E such that $v' = E[x_{\mathtt{o}}]$; the proof is by induction on v'. The head of v' is either $x_{\mathtt{o}}$ or a terminal. If the head is terminal, it must has non-zero arity since $x_{\mathtt{o}}$ occurs in v'; thus $v' = V'_0 * V'_1$ for some V'_0 and V'_1. Since v' cannot be reduced, V'_0 and V'_1 are singletons $\{v'_0\}$ and $\{v'_1\}$, respectively. If $x_{\mathtt{o}}$ occurs in v'_i ($i = 0$ or 1) we can use the induction hypothesis for v'_i. Then v'_{1-i} is closed and cannot be reduced, which means that v'_{1-i} is a value tree.

By Lemma 24 we have $x_{\mathtt{o}} : \mathtt{oR} \vdash v : \mathtt{oR}$, and by Lemma 25, we have $x_{\mathtt{o}} : \mathtt{oR} \vdash E[x_{\mathtt{o}}] : \mathtt{oR}$. Now we show that if $x_{\mathtt{o}} : \mathtt{oR} \vdash E[x_{\mathtt{o}}] : \mathtt{oR}$ then there exists $\pi_1, \ldots, \pi_n$ ($n \ge 0$) such that $E[x_{\mathtt{o}}] = \pi_1 * (\pi_2 * \cdots (\pi_n * x_{\mathtt{o}}) \cdots)$, by induction on E.

- Case $E = [\,]$: In this case, $n := 0$.
- Case $E = \pi' * E'$: Now $E[x_{\mathtt{o}}] = \mathtt{br}\, \pi'\, E'[x_{\mathtt{o}}]$, and in the derivation tree of $x_{\mathtt{o}} : \mathtt{oR} \vdash \mathtt{br}\, \pi'\, E'[x_{\mathtt{o}}] : \mathtt{oR}$, the rule used for the occurrence of $\mathtt{br}$ must be RTY-BRRIGHT. Hence we have $x_{\mathtt{o}} : \mathtt{oR} \vdash E'[x_{\mathtt{o}}] : \mathtt{oR}$ and by the induction hypothesis for E',

  $$E'[x_{\mathtt{o}}] = \pi_1 * (\pi_2 * \cdots (\pi_n * x_{\mathtt{o}}) \cdots)$$

  for some $(\pi_i)_{i\le n}$. Hence,

  $$E[x_{\mathtt{o}}] = \pi' * (\pi_1 * (\pi_2 * \cdots (\pi_n * x_{\mathtt{o}}) \cdots)).$$

- Case $E = E' * v'$: Now $E[x_{\mathtt{o}}] = \mathtt{br}\, E'[x_{\mathtt{o}}]\, v'$, and by traversing the derivation tree of $x_{\mathtt{o}} : \mathtt{oR} \vdash \mathtt{br}\, E'[x_{\mathtt{o}}]\, v' : \mathtt{oR}$, we have $x_{\mathtt{o}} : \mathtt{oR} \vdash E'[x_{\mathtt{o}}] : \mathtt{o}$. But this implies that $\mathtt{oR} \to \mathtt{o}$ is well-formed, which is a contradiction.

  Now we prove the goal of the current lemma by using the context lemma (Lemma 9). Given $[U/x_{\mathtt{o}}]v \longrightarrow^* \pi$, there exists v' such that v' cannot be reduced anymore and

  $$v \longrightarrow^* v' \qquad [U/x_{\mathtt{o}}]v' \longrightarrow^* \pi.$$

Hence there exist $\pi_1, \ldots, \pi_n$ such that

$$v' = \pi_1 * (\pi_2 * \cdots (\pi_n * x_{\mathtt{o}}) \cdots).$$

Since

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

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' cannot be reduced any more and

$$v \longrightarrow^{p_2} v' \qquad [\mathsf{e}/x_\circ]v' \longrightarrow^{p_3} \pi'_0 \qquad p_2 + p_3 = p_0\,.$$

Hence there exist $\pi_1, \ldots, \pi_n$ such that

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

Since

$$[\mathsf{e}/x_\circ]v' = \pi_1 * (\pi_2 * \cdots (\pi_n * \mathsf{e}) \cdots) \longrightarrow^{p_3} \pi'_0,$$

we have

$$p_3 = 0 \qquad p_2 = p_0 \qquad \pi_1 * (\pi_2 * \cdots (\pi_n * \mathsf{e}) \cdots) = \pi'_0\,.$$

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.,

$$[U/x_\circ]v \longrightarrow^{p} \pi_1 * (\pi_2 * \cdots (\pi_n * \pi'_1) \cdots) \sim_{\mathrm{v}} \pi'_0 * \pi'_1 = \pi.$$

◻

B Proof of Theorem 8

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 $\overrightarrow{t_{i,j}}^{i\in\{1,\ldots,k\},j\in\{1,\ldots,m_i\}}$. We also write $[t_i/x_i]_{i\in S}$ for $[t_1/x_1, \ldots, t_k/x_k]$ when $S = \{1, \ldots, k\}$.

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

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

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

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

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

▶ **Lemma 28** (de-substitution)**.** *Let t be an applicative term. If $\Xi \vdash [s/x]t : \xi \Rightarrow u$, then*

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

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

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

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:

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

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

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

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 above case.
- Case $t = \mathtt{br}\, t_0\, t_1$. In this case, we have:

$$\Xi \vdash [s/x]t_i : \xi'_i \Rightarrow u_i \text{ for each } i \in \{0,1\}$$

$$(u, \xi) = \begin{cases} (\mathtt{br}\, u_0\, u_1, \mathtt{o}_+) & \text{if } \xi'_0 = \xi'_1 = \mathtt{o}_+ \\ (u_i, \mathtt{o}_+) & \text{if } \xi'_i = \mathtt{o}_+ \text{ and } \xi'_{1-i} = \mathtt{o}_\epsilon \\ (\mathtt{e}, \mathtt{o}_\epsilon) & \text{if } \xi'_0 = \xi'_1 = \mathtt{o}_\epsilon \end{cases}$$

By applying the induction hypothesis to $\Xi \vdash [s/x]t_i : \xi'_i \Rightarrow u_i$, we obtain:

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

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

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

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

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

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

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

By applying the induction hypothesis, we obtain:

$$\begin{array}{l}\Xi, x : \xi_{0,1}, \ldots, x : \xi_{0,k_0} \vdash t_0 : \xi'_1 \wedge \cdots \wedge \xi'_\ell \to \xi \Rightarrow u'_0 \\ u_0 \preceq\in [U_{0,1}/x_{\xi_{0,1}}, \ldots, U_{0,k_0}/x_{\xi_{0,k_0}}]u'_0 \\ \Xi \vdash s : \xi_{0,i} \Rightarrow U_{0,i} \text{ for each } i \in \{1, \ldots, k_0\} \\ \Xi, x : \xi_{j,1}, \ldots, x : \xi_{j,k_j} \vdash t_1 : \xi'_j \Rightarrow V'_j \\ V_j \preceq [U_{j,1}/x_{\xi_{j,1}}, \ldots, U_{j,k_j}/x_{\xi_{j,k_j}}]V'_j \\ \Xi \vdash s : \xi_{j,i} \Rightarrow U_{j,i} \text{ for each } i \in \{1, \ldots, k_j\}\end{array}$$

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

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

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

□

▶ **Lemma 29.** *Suppose $\mathcal{G} \Rightarrow \mathcal{G}'$. If $s \longrightarrow_{\mathcal{G}} s'$ with $\emptyset \vdash s' : \xi \Rightarrow u'$ and $\xi :: \mathtt{o}$, then there exist u and u'' such that $t \longrightarrow^*_{\mathcal{G}'} u''$ with $\emptyset \vdash s : \xi \Rightarrow u$ 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:

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

By Lemma 28 and $\vdash [s_1/x_1, \ldots, s_k/x_k]t : \xi \Rightarrow u'$, we have:

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

By the first condition, we have:

$$\vdash (A\,x_1\,\cdots\,x_k \to t) \Rightarrow (A_{\xi'}\,\overrightarrow{x_{i,\xi_{i,j}}}^{i\in\{1,\dots,k\},j\in\{1,\dots,m_i\}} \to v)$$

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 $u'' = [U_{i,j}/x_{i,\xi_{i,j}}]_{i\in\{1,\dots,k\},j\in\{1,\dots,m_i\}}v$.

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

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

  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

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

◻

▶ **Lemma 30.** *If $u_1 \preceq u_2$ and $u_1 \longrightarrow_{\mathcal{G}'} u'_1$, then $u_2 \longrightarrow_{\mathcal{G}'} u'_2$ and $u'_1 \preceq u'_2$ for some u'_2.*

Proof. This follows by straightforward induction on the derivation of $u_1 \longrightarrow_{\mathcal{G}'} u'_1$. ◻

▶ **Lemma 31.** *Suppose $\mathcal{G} \Rightarrow \mathcal{G}'$. If $\emptyset \vdash \pi : \xi \Rightarrow u$ and $\xi :: \mathtt{o}$, then t also consists of only terminal symbols and **leaves**$(\pi)\uparrow_{\mathtt{e}}=$ **leaves**(u). Furthermore, $u = \mathtt{e}$ iff $\xi = \mathtt{o}_\epsilon$.*

Proof. This follows by induction on the derivation of $\emptyset \vdash \pi : \xi \Rightarrow u$.

- Case Tr2-Const0: In this case, $\pi = u = a$ and $\xi = \mathtt{o}_+$. Thus, the result follows immediately.
- Case Tr2-Const1: In this case, $\pi = u = \mathtt{e}$ and $\xi = \mathtt{o}_\epsilon$. Thus, the result follows immediately.
- Case Tr1-Const2: In this case, we have:

  $$\begin{array}{l} \pi = \mathtt{br}\,\pi_0\,\pi_1 \\ \emptyset \vdash \pi_i : \xi_i \Rightarrow u_i \\ (u, \xi) = \begin{cases} (\mathtt{br}\,u_0\,u_1, \mathtt{o}_+) & \text{if } \xi_0 = \xi_1 = \mathtt{o}_+ \\ (u_i, \mathtt{o}_+) & \text{if } \xi_i = \mathtt{o}_+ \text{ and } \xi_{1-i} = \mathtt{o}_\epsilon \\ (\mathtt{e}, \mathtt{o}_\epsilon) & \text{if } \xi_0 = \xi_1 = \mathtt{o}_\epsilon \end{cases} \end{array}$$

  The result follows immediately from the induction hypothesis.

◻

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 32** (completeness of the transformation)**.** *If $\mathcal{G} \Rightarrow \mathcal{G}'$, then $\mathcal{L}_{\mathtt{leaf}}(\mathcal{G})\uparrow_{\mathtt{e}} \subseteq \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}')$.*

Proof. Suppose $w \in \mathcal{L}_{\mathtt{leaf}}(\mathcal{G})\uparrow_{\mathtt{e}}$, i.e., **leaves**$(\pi)\uparrow_{\mathtt{e}} \in \mathcal{L}_{\mathtt{leaf}}(\mathcal{G})\uparrow_{\mathtt{e}}$ for some tree π such that $S \longrightarrow^*_{\mathcal{G}} \pi$. By Lemma 31, there exists π' such that $\emptyset \vdash \pi : \xi \Rightarrow \pi'$ and **leaves**$(\pi)\uparrow_{\mathtt{e}}=$ **leaves**(π') with $\xi :: \mathtt{o}$. (By the translation rules, $\xi = \mathtt{o}_\epsilon$ if and only if $\pi' = \mathtt{e}$.) By repeated applications of Lemmas 29 and 30, we have $S_\xi \longrightarrow^*_{\mathcal{G}'} \pi''$ with $\pi' \preceq \pi''$. By the definition of $\preceq$, it must be the case that $\pi' = \pi''$. Thus, we have $w =$ **leaves**$(\pi') \in \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}')$ as required.
◻

We now turn to prove the soundness of the transformation (Lemma 37 below). We again prepare several lemmas.

▶ **Lemma 33.** *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,

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

  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 $[s/x]y = y = t$.
  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 $\Xi \vdash s : \xi_i \Rightarrow U_i$.

□

▶ **Lemma 34.** *Suppose $\mathcal{G} \Rightarrow \mathcal{G}'$. If $u \longrightarrow_{\mathcal{G}} u'$ with $\emptyset \vdash t : \xi \Rightarrow u$ and $\xi :: \mathtt{o}$, then there exists t' such that $t \longrightarrow^*_{\mathcal{G}} t'$ with $\emptyset \vdash t' : \xi \Rightarrow u'$.*

Proof. This follows by double induction on the derivations of $\emptyset \vdash t : \xi \Rightarrow u$ and $u \longrightarrow_{\mathcal{G}} u'$.

We first consider the case where $\emptyset \vdash t : \xi \Rightarrow u$ has been derived by rule (Tr2-Const2), with $\mathtt{o}_\epsilon \in \{\xi_0, \xi_1\}$. Since $t \longrightarrow_{\mathcal{G}} t'$, we have:

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

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

For the other cases, we perform case analysis on the shape of t.

- Case $t = \mathtt{br}\, t_0\, t_1$: In this case, for some $i \in \{0, 1\}$, we have:

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

  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:

  $$u = A_{\xi'}\, U_{1,1} \cdots U_{1,\ell_1} \cdots U_{k,1} \cdots U_{k,\ell_k} \\ \vdash A : \xi' \Rightarrow A_{\xi'} \\ \vdash t_i : \xi_{i,j} \Rightarrow U_{i,j} \text{ for each } i \in \{1, \ldots, k\}, j \in \{1, \ldots, \ell_k\} \\ \xi' = \bigwedge_j \xi_{1,j} \to \cdots \to \bigwedge_j \xi_{k,j} \to \xi \\ A'_\xi\, x_{1,\xi_{1,1}} \cdots x_{1,\xi_{1,\ell_1}} \cdots x_{k,\xi_{k,1}} \cdots x_{k,\xi_{k,\ell_k}} \to u_0 \in \mathcal{G}' \\ u' \in [U_{i,j}/x_{i,\xi_{ij}}]_{i \in \{1,\ldots,k\}, j \in \{1,\ldots,\ell_k\}} u_0$$

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

$$\begin{array}{l} A\, x_1 \cdots x_k \to t_0 \\ x_1 : \xi_{1,1}, \ldots, x_1 : \xi_{1,\ell_1}, \ldots, x_k : \xi_{k,1}, \ldots, x_1 : \xi_{k,\ell_k} \vdash t_0 : \xi \Rightarrow u_0. \end{array}$$

Let $t' = [t_1/x_1, \ldots, t_k/x_k]t_0$. By Lemma 33, we have $\emptyset \vdash t' : \xi \Rightarrow u'$ and $t \longrightarrow^*_{\mathcal{G}} t'$ as required.

□

▶ **Lemma 35.** *If $\vdash t : \mathtt{o}_\epsilon \Rightarrow u$, then $t \longrightarrow^*_{\mathcal{G}} \pi$ for some π such that* ***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 **leaves**$(\pi) \in \mathtt{e}^*$.
- $\mathcal{R}_{\mathtt{o}_+}(t)$ if $t \longrightarrow^*_{\mathcal{G}} \pi$ for some π such that **leaves**$(\pi) \notin \mathtt{e}^*$.
- $\mathcal{R}_{\xi_1 \wedge \cdots \xi_k \to \xi}(t)$ if, whenever $\mathcal{R}_{\xi_i}(s)$ for every $i \in \{1, \ldots, k\}$, $\mathcal{R}_\xi(ts)$.

For a substitution $\theta = [s_1/x_1, \ldots, s_\ell/x_\ell]$, we write $\theta \models \Xi$ if $\Xi = \{x_i : \xi_{i,j} \mid i \in \{1, \ldots, \ell\}, j \in \{1, \ldots, m_i\}\}$ and $\forall i \in \{1, \ldots, \ell\}.\forall j \in \{1, \ldots, m_i\}.\mathcal{R}_{\xi_{i,j}}(s_i)$. 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-NT): In this case, we have:

  $$\begin{array}{l} t = F \qquad \xi = \bigwedge_{j \in \{1,\ldots,\ell_1\}} \xi_{1,j} \to \cdots \to \bigwedge_{j \in \{1,\ldots,\ell_k\}} \xi_{k,j} \to \xi_0 \\ F\, x_1 \cdots x_k \to t_0 \in \mathcal{G} \\ \{x_i : \xi_{i,j} \mid i \in \{1, \ldots, k\}, j \in \{1, \ldots, \ell_k\}\} \vdash t_0 : \xi_0 \Rightarrow u \end{array}$$

  Suppose $\mathcal{R}_{\xi_{i,j}}(s_i)$ for every $i \in \{1, \ldots, k\}, j \in \{1, \ldots, \ell_k\}$. By applying the induction hypothesis to the last condition, we have

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

  (Note that the derivation for $\{x_i : \xi_{i,j} \mid i \in \{1, \ldots, k\}, j \in \{1, \ldots, \ell_k\}\} \vdash t_0 : \xi_0 \Rightarrow u$ is a sub-derivation for $\vdash F : \xi$ů; this is the reason why we have included the rightmost premise is required in the rule (Tr2-NT).) Therefore, we have $\mathcal{R}_{\xi_0}([s_1/x_1, \ldots, s_k/x_k]t_0)$, which implies $\mathcal{R}_{\xi_0}(F\, s_1 \cdots s_k)$ 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 $i \in \{1, \ldots, k\}$. 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 36.** *If $\emptyset \vdash t : \xi \Rightarrow \pi'$, then $t \longrightarrow^*_{\mathcal{G}} \pi$ and* ***leaves***$(\pi) \uparrow_{\mathtt{e}}=$ ***leaves***(π') *for some π.*

Proof. This follows by induction on the derivation of $\emptyset \vdash t : \xi \Rightarrow \pi'$. Since the output of transformation is a tree, the last rule used for deriving $\emptyset \vdash t : \xi \Rightarrow \pi'$ 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 $\emptyset \vdash t_i : \xi_i \Rightarrow u_i$ for $i \in \{0, 1\}$. Case analysis on ξ_0 and ξ_1.

- Case $\xi_0 = \xi_1 = \mathtt{o}_+$: In this case, $\pi' = \mathtt{br}\, u_0\, u_1$. By the induction hypothesis, $t_i \longrightarrow^*_{\mathcal{G}} u_i$ and $\mathbf{leaves}(\pi_i) \uparrow_{\mathtt{e}}= \mathbf{leaves}(\pi'_i)$ for each $i \in \{0, 1\}$. Thus, we have $t \longrightarrow^*_{\mathcal{G}} \pi$ 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, $\pi' = u_i$. By the induction hypothesis, $t_i \longrightarrow^*_{\mathcal{G}} \pi_i$ and $\mathbf{leaves}(\pi_i) \uparrow_{\mathtt{e}}= \mathbf{leaves}(\pi')$ for some π_i. By Lemma 35, we have $t_{1-i} \longrightarrow^*_{\mathcal{G}} \mathtt{e}$. Let $\pi_{1-i} = \mathtt{e}$ and $\pi = \mathtt{br}\, \pi_0\, \pi_1$. Thus, we have $t \longrightarrow^*_{\mathcal{G}} \pi$ and $\mathbf{leaves}(\pi) \uparrow_{\mathtt{e}}= \mathbf{leaves}(\pi')$ as required.
- Case $\xi_0 = \xi_1 = \mathtt{e}$. In this case, $\pi' = \mathtt{e}$. By Lemma 35, we have $t_i \longrightarrow^*_{\mathcal{G}} \mathtt{e}$ for each $i \in \{0, 1\}$. Thus, we have $t \longrightarrow^*_{\mathcal{G}} \pi$ and $\mathbf{leaves}(\pi) \uparrow_{\mathtt{e}}= \mathbf{leaves}(\pi')$ for $\pi = \mathtt{br\, e\, e}$ as required.

◻

We are now ready to prove soundness of the transformation.

▶ **Lemma 37** (soundness). *If $\mathcal{G} \Rightarrow \mathcal{G}'$, then $\mathcal{L}_{\mathtt{leaf}}(\mathcal{G}) \uparrow_{\mathtt{e}} \supseteq \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}')$.*

Proof. Suppose $w \in \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}')$. Then $S_\xi \longrightarrow^*_{\mathcal{G}'} \pi'$ with $\mathbf{leaves}(\pi') = w$. By repeated applications of Lemma 34, we have $S \longrightarrow^*_{\mathcal{G}} t$ and $\emptyset \vdash t : \xi \Rightarrow \pi'$. By Lemma 36, we have $t \longrightarrow^*_{\mathcal{G}} \pi$ and $\mathbf{leaves}(\pi) \uparrow_{\mathtt{e}}= \mathbf{leaves}(\pi')$ for some π. Thus, we have $S \longrightarrow^*_{\mathcal{G}} \pi'$ and $\mathbf{leaves}(\pi) \uparrow_{\mathtt{e}}= \mathbf{leaves}(\pi') = w$ as required. ◻

Proof of Theorem 8.

The fact that $\mathcal{G}'$ is an order-n grammar follows immediately from Lemma 26. $\mathcal{L}_{\mathtt{leaf}}(\mathcal{G}) \uparrow_{\mathtt{e}}= \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}')$ follows immediately from Lemmas 32 and 37. ◻