Formal Efficiency Analysis for Tree Transducer Composition*

Janis Voigtländer†

Department of Computer Science, Dresden University of Technology,
01062 Dresden, Germany. E-mail: voigt@tcs.inf.tu-dresden.de

Abstract

We study the question of efficiency improvement or deterioration for a semantics-preserving program transformation technique for (lazy) functional languages, based on composition of restricted macro tree transducers. By annotating programs to reflect the intensional property "computation time" explicitly in the computed output and by manipulating such annotations, we formally prove syntactic conditions under which the composed program is guaranteed to be not less efficient than the original program with respect to the number of call-by-need reduction steps required to reach normal form. The criteria developed can be checked automatically and efficiently, and thus are suitable for integration into an optimizing compiler.

1 Introduction

Automatic transformation of programs is a key technology in software engineering, as it enables programmers to work at a higher level of abstraction than would otherwise be possible and thus raises programmer productivity. Prominent among its many applications to lazy functional languages in particular is the elimination of intermediate results, with the aim of mitigating the tension between modularity—as desired for the sake of reliability, maintainability, and reusability—and efficiency of programs. Under the name *deforestation*, automatic elimination of structured intermediate results (i.e. trees) was pioneered in [Wad90] through an algorithmic instance of the *unfold/fold-technique* [BD77] that is suitable for integration into an optimizing compiler. This unfold/fold-based approach has seen many extensions—e.g. in [HJ92], [MW93], and [Chi94]—, which we will collectively refer to as *classical deforestation* in the following. A fundamental weakness of classical deforestation is its inability to eliminate intermediate results that are built inside so-called *accumulating parameters*. This problem is also shared by *shortcut deforestation* techniques [GLP93, Chi99, Joh02, Sve02], based on denoting producers and consumers of intermediate results in terms of certain higher-order, polymorphic combinators.

An accumulating parameter is one that may grow in recursive function calls. For example, in the following rules defining a function g with two parameters:

$$g\ (\delta\ v_1\ v_2)\ z_1 \rightarrow \delta\ (g\ v_1\ (g\ v_2\ z_1))$$

*Author's version. Original publication in *Theory of Computing Systems*, 41:619–689, 2007.
†Research supported by the "Deutsche Forschungsgemeinschaft" under grant KU 1290/2-4.

$$\begin{array}{ll} g\ (\alpha\ v_1) & z_1 \to \alpha\ (g\ v_1\ z_1) \\ g\ (\beta\ v_1) & z_1 \to \beta\ (g\ v_1\ z_1) \\ g\ \epsilon & z_1 \to \epsilon\ z_1 \end{array}$$

every recursive call performs a strict descent in the first parameter, whereas the second parameter is accumulating (cf. the first rule). These rules together with an initial expression $g\ v_1\ \gamma$ constitute the generally favored solution to the task of computing (as a monadic output tree, i.e., essentially as a list) the prefix traversal of a tree built from binary symbol δ, unary symbols α and β, and nullary symbol ϵ. In fact, accumulating style is a quite common programming idiom in functional languages (cf. e.g. Chapter 6 of [FFFK01]). As a consequence, the problem of eliminating intermediate results produced inside accumulating parameters has received much attention [Küh98, Küh99, CDPR99, KGF02a, KGF02b, Nis02, VK04a, Voi04b, Nis04]. All solutions proposed so far derive from restricting the class of input programs to certain extended schemes of primitive recursion, which can be formalized as (deterministic) *macro tree transducers* [Eng80].

A macro tree transducer (for short *mtt*) translates trees over a ranked alphabet of input symbols into trees over a ranked alphabet of output symbols. For this translation process an mtt uses a set of functions which are defined by rewrite rules. More precisely, every function is defined by pattern matching on the root symbol of its first argument t. The right-hand side of every rule may contain the other function arguments, output symbols, and recursive function calls—potentially to other functions of the mtt—, the first arguments of which must be variables that refer to subtrees of t. The above program for prefix traversal is a simple example of an mtt. Since many typical functions on algebraic data types are defined by structural descent on a distinguished argument, mtts represent a large class of functional programs using accumulating parameters, which are called *context parameters* in the scope of mtts. For example, manipulation of abstract syntax trees in compilers often follows the recursion scheme of mtts [Vog91, FV98], and the "tree transformation core" of XML processing languages can be compiled into compositions of mtts [EM03a, MBPS05].

By transforming functional programs corresponding to restricted mtts into *attribute grammars* [Knu68] or *attributed tree transducers* [Fül81], performing a composition construction on that level, and transforming the result back into a functional program, [Küh98] and [CDPR99] were the first to achieve elimination of intermediate results produced inside accumulating parameters. The transformation technique consists of a combination of constructions from [Fül81, Fra82, CF82, Gan83, GG84, Gie88] and relies on certain syntactic restrictions imposed on the original mtts, viz. *context-linearity* (meaning that context parameters are never copied, imposed on the producing mtt) and *weakly single-useness* (as introduced in [Küh98], imposed on both the producing and the consuming mtt). In [Voi01, VK04a] a direct construction without detour to the "attribute world" was presented that is applicable to a pair of less restricted mtts. In particular, it successfully performs a composition also if the producer of the intermediate result is an arbitrary mtt, while the consumer is a *top-down tree transducer* (for short *tdtt*) [Rou70, Tha70, Eng75], i.e., an

mtt without context parameters. Thus, it captures a result from [EV85] that was used in [Küh99] as another application of tree transducer theory to accumulating parameter deforestation. In fact, except for [Nis04] (allowing a restricted use of stacks in input programs), none of the techniques proposed to date for eliminating intermediate results produced inside accumulating parameters handles a richer class of programs than the construction from [VK04a]. The latter's direct nature made it suitable for integration into an optimizing compiler for the lazy functional language Haskell [Reu03].

Beside preservation of program semantics (which for the technique from [VK04a] was proved in [Voi01, VK04b]) more intensional properties are in focus when considering program transformations. The key question usually to be asked about a source-to-source transformation implemented in a compiler is whether it will make the compiled program more efficient in some sense. Elimination of intermediate results aims in this direction by avoiding memory allocations and deallocations, but this alone is not sufficient. In fact, none of the deforestation techniques mentioned above is guaranteed, when applied to arbitrary input programs, to produce output programs that are faster—or at least not slower—than the original ones. The standard result about classical deforestation is that it does not deteriorate the efficiency of input programs consisting solely of functions that never copy any of their arguments, where efficiency is measured by counting *call-by-need* reduction steps. Examples show that for tree transducer composition such linearity of input programs is neither a sufficient nor a necessary condition for efficiency nondeterioration with respect to this measure. For the special case that one of the mtts to be composed is a tdtt sufficient conditions were developed in [Küh99] and [Höf99] using rather ad-hoc reasoning. In [Voi02] we have studied this case using a more systematic approach, which in the work presented here is extended to the full construction from [VK04a], thus providing the first formal efficiency guarantees for elimination of intermediate results by general tree transducer composition. The criteria obtained improve on previous results and cover many typical examples. They are decidable and we present an efficient decision procedure (also suitable for integration into an optimizing compiler), correctness of which was proved in [Voi05].

The general idea behind our approach to efficiency analysis is that of annotating programs to reflect computation time in the observable output. A certain "decomposition" of the respective annotation of the composed program in terms of a suitable annotation of the original program allows to capture the time behavior of the composed program without ever explicitly producing it. Combining and manipulating annotations (of the original program only) then leads to our syntactic conditions for efficiency nondeterioration. For a more detailed outline, please consult Section 4.1.

The current paper is derived from the technical report [Voi04a]; it contains exactly the same results about call-by-need efficiency, but less proof details. The thesis [Voi05] subsumes the mentioned technical report and thus also the work presented here. Additionally, it studies further (efficiency and semantic) aspects of tree transducer composition as a program transformation. As mentioned above, the earlier work [Voi02] studied more restricted composition instances. While part of the

efficiency nondeterioration results from that study is reproduced in the present work, this coverage is not complete. Hence, one theorem from [Voi02] will simply be repeated. The remainder of the paper is organized as follows (plus a proof appendix). Section 2 defines necessary notions and introduces the basic concepts of mtts. Section 3 recalls the composition construction to be studied. Section 4 develops our efficiency analysis. Section 5 considers related work. Section 6 concludes.

2 Preliminaries

We denote by $\mathbb{N}$ the set of natural numbers including 0 and by $\mathbb{Z}$ the set of all integers. We set $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. For every $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. The mapping *max* gives for every finite, nonempty subset of $\mathbb{N}$ the maximum of that subset's elements.

Let S be a set. We denote by S^* the set of finite sequences of elements of S, where ε denotes the empty sequence. The power set of S is denoted by $\mathcal{P}(S)$. If S is finite, then the number of its elements is denoted by $|S|$. The Cartesian product of sets $S_1, \ldots, S_n$ is denoted by $S_1 \times \cdots \times S_n$. The n-fold Cartesian product of one set S is denoted by S^n. Given a binary relation $\Rightarrow \in \mathcal{P}(S^2)$, we write $s \Rightarrow s'$ for $(s, s') \in \Rightarrow$. We denote by $\Rightarrow^n$ (with $n \in \mathbb{N}$) the n-fold composition, by $\Rightarrow^?$ the reflexive, and by $\Rightarrow^*$ the reflexive, transitive closure of $\Rightarrow$, respectively (all in $\mathcal{P}(S^2)$). If for every $s, s_1, s_2 \in S$ with $s \Rightarrow^* s_1$ and $s \Rightarrow^* s_2$ there exists $s' \in S$ with $s_1 \Rightarrow^* s'$ and $s_2 \Rightarrow^* s'$, then $\Rightarrow$ is called *confluent*. If there is no infinite chain $s_1 \Rightarrow s_2 \Rightarrow s_3 \Rightarrow \cdots$, then $\Rightarrow$ is called *terminating*. If $s \Rightarrow^* s'$ and there is no s'' with $s' \Rightarrow s''$, then s' is called a *normal form* of s with respect to $\Rightarrow$. If $\Rightarrow$ is confluent and terminating, then every $s \in S$ has a unique normal form, denoted by $nf(\Rightarrow, s)$.

We denote by *id* the identity function on every set. We use two symmetric versions of function composition, namely for all sets S_1, S_2, S_3, and functions $f : S_1 \longrightarrow S_2$ and $g : S_2 \longrightarrow S_3$ we define $f;g = g \cdot f : S_1 \longrightarrow S_3$ such that for every $s \in S_1$: $(g \cdot f)(s) = g(f(s))$. For every function $f : S \longrightarrow S$ and $n \in \mathbb{N}$ the n-fold composition of f is denoted by f^n, where $f^0 = id$. The Cartesian product of functions $f_1 : S_1 \longrightarrow S_1', \ldots, f_n : S_n \longrightarrow S_n'$ is denoted by $f_1 \times \cdots \times f_n : S_1 \times \cdots \times S_n \longrightarrow S_1' \times \cdots \times S_n'$ and defined by $(f_1 \times \cdots \times f_n)(s_1, \ldots, s_n) = (f_1(s_1), \ldots, f_n(s_n))$. For two functions $f : S \longrightarrow \mathbb{Z}$ and $g : S \longrightarrow \mathbb{Z}$ we write $f \leq g$ iff $f(s) \leq g(s)$ for every $s \in S$, and we write $f - g : S \longrightarrow \mathbb{Z}$ for the function that maps every $s \in S$ to $f(s) - g(s)$.

A *ranked alphabet* is a pair $(\Sigma, rank_\Sigma)$, where Σ is a finite set of symbols and $rank_\Sigma$ assigns to each of these symbols a natural number, its *rank*. In the following, we drop the $rank_\Sigma$-function from the denotation and only mention Σ when referring to a ranked alphabet. For every $p \in \mathbb{N}$ we define $\Sigma^{(p)} = \{\sigma \in \Sigma \mid rank_\Sigma(\sigma) = p\}$. The rank p of a symbol σ is also denoted by writing $\sigma^{(p)}$. A *nullary* symbol is one of rank 0; a *unary* symbol one of rank 1. For the sake of brevity, quantifications over a symbol in a ranked alphabet implicitly quantify also its rank. E.g., we write "for every $\sigma \in \Sigma^{(p)}$" instead of "for every $p \in \mathbb{N}$, $\sigma \in \Sigma^{(p)}$" and "there exists

$f \in F^{(r+1)}$" instead of "there exist $r \in \mathbb{N}$ and $f \in F^{(r+1)}$". For a ranked alphabet Σ we denote the set of all its ranks as $rank(\Sigma) = \{p \in \mathbb{N} \mid \exists \sigma \in \Sigma.\ rank_\Sigma(\sigma) = p\}$. We use several sets of lowercase variables. We denote by U the set $\{u_1, u_2, u_3, \ldots\}$ of variables, and for every $p \in \mathbb{N}$ by U_p the finite set $\{u_1, \ldots, u_p\} \subseteq U$; analogously for V, Y, Z, and $Y' = \{y'_1, y'_2, y'_3, \ldots\}$. For a ranked alphabet Σ and a set X of variables disjoint from Σ we define the set $T_\Sigma(X)$ of *trees over* Σ *indexed by* X as the smallest set $T \subseteq (\Sigma \cup X \cup \{(,)\})^*$ such that (i) $X \subseteq T$ and (ii) for every $\sigma \in \Sigma^{(p)}$ and $t_1, \ldots, t_p \in T$: $(\sigma\ t_1 \cdots t_p) \in T$. If readability allows, outer brackets of trees are omitted. Every symbol $\sigma \in \Sigma^{(p)}$ can also be considered as a function of type $T_\Sigma(X)^p \longrightarrow T_\Sigma(X)$, mapping $(t_1, \ldots, t_p)$ to $\sigma\ t_1 \cdots t_p$. We denote $T_\Sigma(\emptyset)$ by T_Σ.

We need the *set of paths in a tree*, the *label at a path in a tree*, and the *subtree at a path in a tree*, given by the functions $paths : T_\Sigma(X) \longrightarrow \mathcal{P}((\mathbb{N}_+)^*)$, $lab : \{(t, \pi) \mid t \in T_\Sigma(X), \pi \in paths(t)\} \longrightarrow \Sigma \cup X$, and $sub : \{(t, \pi) \mid t \in T_\Sigma(X), \pi \in paths(t)\} \longrightarrow T_\Sigma(X)$, which are defined, with $x \in X$, $\sigma \in \Sigma^{(p)}$, $i \in \mathbb{N}_+$, $\pi \in (\mathbb{N}_+)^*$, and $t, t_1, \ldots, t_p \in T_\Sigma(X)$, by the equations $paths(x) = \{\varepsilon\}$, $paths(\sigma\ t_1 \cdots t_p) = \{\varepsilon\} \cup \{i\pi \mid i \in [p], \pi \in paths(t_i)\}$, $lab(x, \varepsilon) = x$, $lab(\sigma\ t_1 \cdots t_p, \varepsilon) = \sigma$, $lab(\sigma\ t_1 \cdots t_p, i\pi) = lab(t_i, \pi)$, $sub(t, \varepsilon) = t$, and $sub(\sigma\ t_1 \cdots t_p, i\pi) = sub(t_i, \pi)$. The label $lab(t, \varepsilon)$ is called the *root symbol* of the tree t. We also need the *number of occurrences of some symbol* $s \in \Sigma \cup X$ *in a tree*, given by the function $|\,.\,|_s : T_\Sigma(X) \longrightarrow \mathbb{N}$, which is defined by $|t|_s = |\{\pi \in paths(t) \mid lab(t, \pi) = s\}|$.

We use two kinds of substitution over trees, written postfix and left-associative. For pairwise different variables $x_1, \ldots, x_n \in X$ and trees $t'_1, \ldots, t'_n \in T_{\Sigma'}(X')$ for a ranked alphabet Σ' and a set X' of variables such that $(\Sigma \cup \Sigma') \cap (X \cup X') = \emptyset$, the *first-order substitution* $.[x_1, \ldots, x_n \leftarrow t'_1, \ldots, t'_n] : T_\Sigma(X) \longrightarrow T_{\Sigma\cup\Sigma'}((X \setminus \{x_1, \ldots, x_n\}) \cup X')$ is defined, with $x \in X \setminus \{x_1, \ldots, x_n\}$, $i \in [n]$, $\sigma \in \Sigma^{(p)}$, and $t_1, \ldots, t_p \in T_\Sigma(X)$, by the equations $x[\cdots] = x$, $x_i[\cdots] = t'_i$, and $(\sigma\ t_1 \cdots t_p)[\cdots] = \sigma\ t_1[\cdots]\ \cdots\ t_p[\cdots]$. For pairwise different symbols $\sigma_1 \in \Sigma^{(p_1)}, \ldots, \sigma_n \in \Sigma^{(p_n)}$ and functions $f_1 : T_{\Sigma\cup\Sigma'}(X)^{p_1} \longrightarrow T_{\Sigma\cup\Sigma'}(X), \ldots, f_n : T_{\Sigma\cup\Sigma'}(X)^{p_n} \longrightarrow T_{\Sigma\cup\Sigma'}(X)$ for a ranked alphabet Σ' such that $\Sigma' \cap X = \emptyset$, the *second-order substitution* $.[\sigma_1, \ldots, \sigma_n \twoheadleftarrow f_1, \ldots, f_n] : T_\Sigma(X) \longrightarrow T_{\Sigma\cup\Sigma'}(X)$ is defined, with $x \in X$, $\sigma \in \Sigma^{(p)} \setminus \{\sigma_1, \ldots, \sigma_n\}$, $i \in [n]$, and $t_j \in T_\Sigma(X)$ for $j \in \mathbb{N}$, by the equations $x[\cdots] = x$, $(\sigma\ t_1 \cdots t_p)[\cdots] = \sigma\ t_1[\cdots]\ \cdots\ t_p[\cdots]$, and $(\sigma_i\ t_1 \cdots t_{p_i})[\cdots] = f_i(t_1[\cdots], \ldots, t_{p_i}[\cdots])$. We also use alternative notations similar to set comprehensions, e.g. $.[x_i \leftarrow t'_i \mid i \in [n]]$, and appropriate multi-line notations for long substitutions.

A *rewrite rule* over Σ and X is a rule of the form $lhs \to rhs$ with $lhs, rhs \in T_\Sigma(X)$ such that the left-hand side lhs does not contain two occurrences of the same variable and every variable occurring in the right-hand side rhs is also contained in lhs. A set R of rewrite rules over Σ and X is called a *rewrite system* [DJ90, BN98] (over Σ and X, but also over $\Sigma' \supseteq \Sigma$ and $X' \supseteq X$ if $\Sigma' \cap X' = \emptyset$). For every $\Sigma' \supseteq \Sigma$ (where Σ' and X are not necessarily disjoint) it induces a binary *reduction relation* $\Rightarrow_R \subseteq T_{\Sigma'} \times T_{\Sigma'}$ such that $t \Rightarrow_R t'$ iff R contains a rule $lhs \to rhs$, there is a tree $c \in T_{\Sigma'}(\{x\})$ (with $x \notin \Sigma' \cup X$) that contains x exactly once, and there exist $n \in \mathbb{N}$, trees $t_1, \ldots, t_n \in T_{\Sigma'}$, and pairwise different variables $x_1, \ldots, x_n \in X$ such that $t = c[x \leftarrow lhs[x_1, \ldots, x_n \leftarrow t_1, \ldots, t_n]]$ and $t' = c[x \leftarrow rhs[x_1, \ldots, x_n \leftarrow t_1, \ldots, t_n]]$.

Definition 2.1 (macro tree transducer, *RHS*)
A *macro tree transducer* (for short *mtt*) M is a tuple $(F, \Sigma, \Delta, e, R)$ consisting of:

- a ranked alphabet F of *states*, where $F^{(0)} = \emptyset$ and $F \neq \emptyset$,
- a ranked alphabet Σ of *input symbols*, where $\Sigma^{(0)} \neq \emptyset$ and $F \cap \Sigma = \emptyset$,
- a ranked alphabet Δ of *output symbols*, where $\Delta^{(0)} \neq \emptyset$ and $F \cap \Delta = \emptyset$,
- an *initial expression* $e \in RHS(F, \Delta, U_1, \emptyset)$, and
- a set R containing for every $f \in F^{(r+1)}$ and $\sigma \in \Sigma^{(p)}$ exactly one *rule* of the form $f\ (\sigma\ u_1 \cdots u_p)\ y_1 \cdots y_r \to rhs_{M,f,\sigma}$, where $rhs_{M,f,\sigma} \in RHS(F, \Delta, U_p, Y_r)$.

Here for variable sets X and X' the set $RHS(F, \Delta, X, X')$ is the smallest set $RHS \subseteq T_{F \cup \Delta}(X \cup X')$ satisfying the following conditions:

- $X' \subseteq RHS$,
- for every $\delta \in \Delta^{(q)}$ and $\phi_1, \ldots, \phi_q \in RHS$: $(\delta\ \phi_1 \cdots \phi_q) \in RHS$, and
- for every $f \in F^{(r+1)}$, $x \in X$, $\phi_1, \ldots, \phi_r \in RHS$: $(f\ x\ \phi_1 \cdots \phi_r) \in RHS$. □

Note that R in the above definition is a rewrite system over $F \cup \Sigma \cup \Delta$ and $U \cup Y$. A *rule right-hand side* of M is the right-hand side of some rule in R. For $\sigma \in \Sigma$, every rule in R of the form $f\ (\sigma\ \cdots)\ \cdots \to \cdots$ for some $f \in F$ is called a *σ-rule*. A subtree of the form $(f\ t\ \cdots)$ is referred to as a *call* to f on t. The first argument of a state f is called *recursion argument*; the others *context parameters*. Correspondingly, variables from U are called *recursion variables* and variables from Y are called *context variables*. However, the actual variable names used in mtt rules are not fixed to come from U_p and Y_r for some $p, r \in \mathbb{N}$; consistent renaming is allowed. For example, we later use recursion variables from V and context variables from Z for the second mtt in the composition construction (with states in G, input symbols in Δ, and output symbols in Ω), in the way we have already done for the introductory example.

Example 2.2 (two mtts, to be used as running example)
Let $\Sigma_{mon} = \{\alpha^{(1)}, \beta^{(1)}, \epsilon^{(0)}\}$, $\Delta_{tree} = \{\delta^{(2)}, \alpha^{(1)}, \beta^{(1)}, \epsilon^{(0)}\}$, and $\Omega_{mon} = \{\delta^{(1)}, \alpha^{(1)}, \beta^{(1)}, \epsilon^{(1)}, \gamma^{(0)}\}$. We define the mtts $M_{spine} = (\{f_1^{(2)}, f_2^{(3)}\}, \Sigma_{mon}, \Delta_{tree}, f_1\ u_1\ \epsilon, R_{spine})$ with the set of rules

$$
\begin{array}{lll}
R_{spine}\colon & f_1\ (\alpha\ u_1)\ y_1 \to \alpha\ (f_1\ u_1\ y_1) & f_2\ (\alpha\ u_1)\ y_1\ y_2 \to \alpha\ (f_1\ u_1\ (\delta\ y_1\ y_2)) \\
& f_1\ (\beta\ u_1)\ y_1 \to f_2\ u_1\ y_1\ (\beta\ \epsilon) & f_2\ (\beta\ u_1)\ y_1\ y_2 \to f_2\ u_1\ y_1\ (\beta\ (\beta\ y_2)) \\
& f_1\ \epsilon \quad\ y_1 \to y_1 & f_2\ \epsilon \quad\ y_1\ y_2 \to \delta\ y_1\ y_2
\end{array}
$$

and $M_{pfx} = (\{g^{(2)}\}, \Delta_{tree}, \Omega_{mon}, g\ v_1\ \gamma, R_{pfx})$, where R_{pfx} contains the rules for g given in the introduction. □

The output computed by an mtt for a particular input tree is obtained by substituting the input tree for u_1 (or for v_1 if recursion variables are drawn from V) in

the mtt's initial expression and then computing the normal form with respect to the reduction relation induced by the mtt's set of rules. This normal form exists and is unique because the rules of any mtt induce a confluent and terminating reduction relation (cf. e.g. [FV98]).

Definition 2.3 (semantics of an mtt)
The *tree transduction induced by an mtt* $M = (F, \Sigma, \Delta, e, R)$ is the total function $\tau_M : T_\Sigma \longrightarrow T_\Delta$ that assigns to every tree $t \in T_\Sigma$ the value $nf(\Rightarrow_R, e[u_1 \leftarrow t])$. □

The tree transduction induced by the mtt M_{spine} from Example 2.2 maps every element of $T_{\Sigma_{mon}}$ to one of $T_{\Delta_{tree}}$ as depicted in Figure 1. While that transformation might seem a bit artificial in comparison to the more straightforward examples in [KV01] and [VK04a], the mtts M_{spine} and M_{pfx} together serve as a good running example here because they seem to constitute about the simplest reasonable composition pair embodying all the phenomena we later want to illustrate when developing our efficiency analysis. The example also has the advantage of requiring a quite disciplined, general approach rather than tempting us to found the analysis on too specific assumptions about the program to be transformed. In particular, M_{spine} is complex enough to defeat simplistic analysis attempts based solely on the size of—i.e. the number of symbols in—the intermediate result compared to that of the original input.

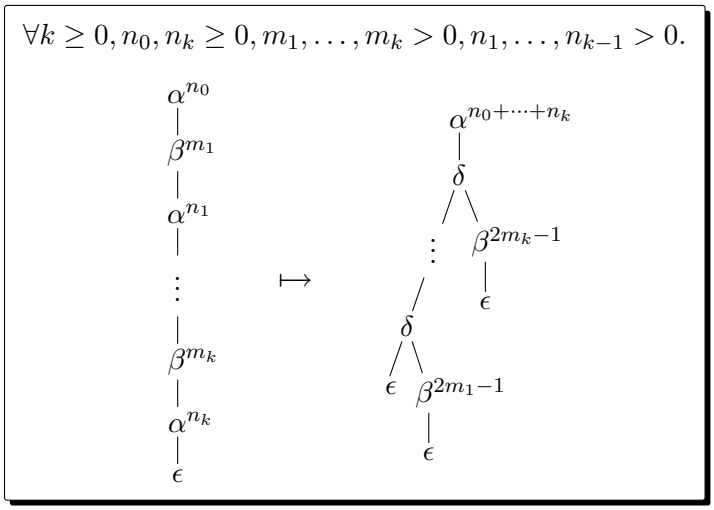

Figure 1: $\tau_{M_{spine}}$

Definition 2.4 (syntactic restrictions of mtts)
An mtt $M = (F, \Sigma, \Delta, e, R)$ is:

- a *top-down tree transducer* (for short *tdtt*) if $F = F^{(1)}$
- *basic* [Vog87] if e and the right-hand sides of the rules in R do not contain any nested calls, i.e. subtrees of the form $(f\ \cdots(f'\ \cdots)\cdots)$ with $f, f' \in F$
- *weakly single-use* if for every $\sigma \in \Sigma^{(p)}$, $i \in [p]$, and $f \in F \setminus F^{(1)}$ at most one call to f on u_i occurs in all right-hand sides of σ-rules in R

- *recursion-linear* if for every $f \in F$, $\sigma \in \Sigma^{(p)}$, and $i \in [p]$: $|rhs_{M,f,\sigma}|_{u_i} \leq 1$
- *context-linear* if for every $f \in F^{(r+1)}$, $\sigma \in \Sigma$, and $k \in [r]$: $|rhs_{M,f,\sigma}|_{y_k} \leq 1$
- *linear* if it is recursion- and context-linear
- *atmost* if it is recursion-linear and it is context-linear or basic
- *recursion-nondeleting* if for every $f \in F$, $\sigma \in \Sigma^{(p)}$, and $i \in [p]$: $|rhs_{M,f,\sigma}|_{u_i} \geq 1$
- *context-nondeleting* if for every $f \in F^{(r+1)}$, $\sigma \in \Sigma$, and $k \in [r]$: $|rhs_{M,f,\sigma}|_{y_k} \geq 1$
- *nondeleting* if it is recursion-nondeleting and context-nondeleting
- *atleast* if it is recursion-nondeleting and it is context-nondeleting or basic. □

The weakly single-use property as defined above is slightly less restrictive than the one originally introduced in [Küh98], where occurrence of two or more calls to f on u_i in right-hand sides of σ-rules (for given σ and i) was forbidden for all states f, not just for nonunary ones, and where an analogous ban was imposed also regarding the initial expression (the "right-hand side of the rule for the initial synthesized function at the root symbol" in the terminology of [Küh98]). It was, however, observed in [VK04a] that the property as liberalized above actually suffices to ensure applicability and semantic correctness of the construction to be recalled in the next section.

The following lemma was proved in [Voi04a]. Since it is straightforward, we do not include the proof here. Similar lemmas also appear in [EM99] and [EM03b].

Lemma 2.5 (properties of context-linear and -nondeleting mtts, resp.) Let $M = (F, \Sigma, \Delta, e, R)$ be an mtt. For every $t \in T_\Sigma$, $f \in F^{(r+1)}$, and $k \in [r]$:

1. if M is context-linear, then $|nf(\Rightarrow_R, f\ t\ y_1 \cdots y_r)|_{y_k} \leq 1$
2. if M is context-nondeleting, then $|nf(\Rightarrow_R, f\ t\ y_1 \cdots y_r)|_{y_k} \geq 1$. □

The following lemma, which is proved in the appendix, will also be useful.

Lemma 2.6 (auxiliary, actual vs. formal parameters) Let $M = (F, \Sigma, \Delta, e, R)$ be an mtt. For every $t \in T_\Sigma$, $f \in F^{(r+1)}$, and $\theta_1, \ldots, \theta_r \in T_\Delta$:

$$nf(\Rightarrow_R, f\ t\ \theta_1 \cdots \theta_r) = nf(\Rightarrow_R, f\ t\ y_1 \cdots y_r)[y_k \leftarrow \theta_k \mid k \in [r]].$$ □

3 Tree transducer composition

In [Voi01, VK04a] a direct composition construction is presented that given two mtts M_1 and M_2, where the output ranked alphabet of M_1 is the input ranked alphabet of M_2 and where one of M_1 and M_2 is a tdtt, or M_1 is context-linear and M_2 is weakly single-use, produces an mtt $\overline{M_1;M_2}$ such that $\tau_{\overline{M_1;M_2}} = \tau_{M_1} ; \tau_{M_2}$. Before recalling the formal construction (with notations adapted to the setting of the present paper),

we briefly consider its ingredients on an intuitive level. For a more detailed step-by-step development of the underlying ideas, please consult Section 4 of [VK04a]. The brave-hearted may also want to study the formal correctness proof supplied in [VK04b].

The key to avoiding the production and consumption of an intermediate result is that whenever a state g of M_2 would recurse on the output generated by a state f of M_1, the final output is instead computed more directly through a call to a new state $\overline{fg}$ of $\overline{M_1;M_2}$ on f's recursion argument. This means that every nesting of the form

$$g\ (f\ t\ \phi_1 \cdots \phi_r)\ \eta_1 \cdots \eta_s \tag{1}$$

is replaced by an appropriate call to $\overline{fg}$ on t. Given that the (intermediate) output generated by f is essentially built up by patching together right-hand sides of M_1's rules at the symbols occurring in t, the rules for such a new state $\overline{fg}$ are obtained by "translating" the right-hand sides of rules for f using M_2, starting with g. More precisely, for every potential root symbol σ of t, $rhs_{\overline{M_1;M_2},\overline{fg},\sigma}$ is determined from $g\ rhs_{M_1,f,\sigma}\ z_1 \cdots z_s$—where $z_1 \cdots z_s$ are the context variables of $\overline{fg}$ that correspond to the context parameters of the outer g-call in (1)—by exhaustively rewriting with the rules of M_2. Thus, the symbols in $rhs_{M_1,f,\sigma}$ that would otherwise build part of the intermediate data structure passed at runtime from the inner f-call to the outer g-call in (1) are consumed at transformation time. Since $rhs_{M_1,f,\sigma}$ may contain recursive calls, one may during the rewriting also encounter situations corresponding to a nesting like (1) above (possibly for some $f' \neq f$ and $g' \neq g$). To ensure that all parts of $rhs_{M_1,f,\sigma}$—including those in context parameter positions of recursive calls—are effectively reached by the transformation, any call to a new state replacing such a nesting takes as arguments beside the unchanged context parameters of the outer state from M_2 also "translated" versions of the context parameters of the inner state from M_1 with all states of M_2. Under this strategy, the replacement for a call like (1) takes the following form (assuming that M_2 has exactly $\mu \in \mathbb{N}_+$ states $g_1, \ldots, g_\mu$):

$$\overline{fg}\ t\ (g_1\ \phi_1\ \boxed{\cdots})\ \cdots\ (g_\mu\ \phi_1\ \boxed{\cdots})\ \cdots\ (g_1\ \phi_r\ \boxed{\cdots})\ \cdots\ (g_\mu\ \phi_r\ \boxed{\cdots})\ \eta_1 \cdots \eta_s \tag{2}$$

Since $\overline{fg}$ is thus provided with precomputed translations of f's context parameters, any call to some state of M_2 on some context variable of f surfacing during the processing of $g\ rhs_{M_1,f,\sigma}\ z_1 \cdots z_s$ can be resolved by simply selecting the corresponding parameter of $\overline{fg}$, modeled by a rewrite system Pre in the formal construction below.

Of course, the crux in this strategy of sending the states of M_2 into the context parameters of calls to M_1's states is to determine what should go into the boxes in the replacement (2) for (1) above. Here the preconditions on M_1 and M_2 enter the picture. If one of the two mtts is actually a tdtt, then no boxes are present because either $r = 0$ or none of the $g_1, \ldots, g_\mu$ takes any context parameters. Otherwise, context-linearity of M_1 and weakly single-useness of M_2 ensure that the context parameter values with which the states $g_1, \ldots, g_\mu$ may reach the context parameters of the inner f-call during reduction of (1) are unambiguous. Moreover, these values can themselves be computed—without leaving the recursion scheme of mtts—by

new states $\overline{k_f l_{g'}}$ for every $k \in [r]$, $g' \in \{g_1, \ldots, g_\mu\}$, and context parameter position l of g', as indicated in Figure 2. Hence, these auxiliary states can be used to provide the information missing in (2). In the formal construction the precise shape of the thus completed replacement for (1) is specified in the rewrite system *Pair*, built up using the function $nest_f$ to arrange a repeated nesting of $g_1, \ldots, g_\mu$-translations of f's context parameters interspersed with appropriate calls to $\overline{k_f l_{g'}}$-states. The reasons why this nesting is necessary but can be kept of finite depth are discussed at length in [VK04a]. There it is also motivated which context parameters the auxiliary states require and how their rules can be constructed following an intricate scheme of "walking upwards" from occurrences of context variables in M_1's right-hand sides, using *par*-functions that assemble information about context parameters of recursive calls in M_2's rules at symbols encountered on the way to the root.

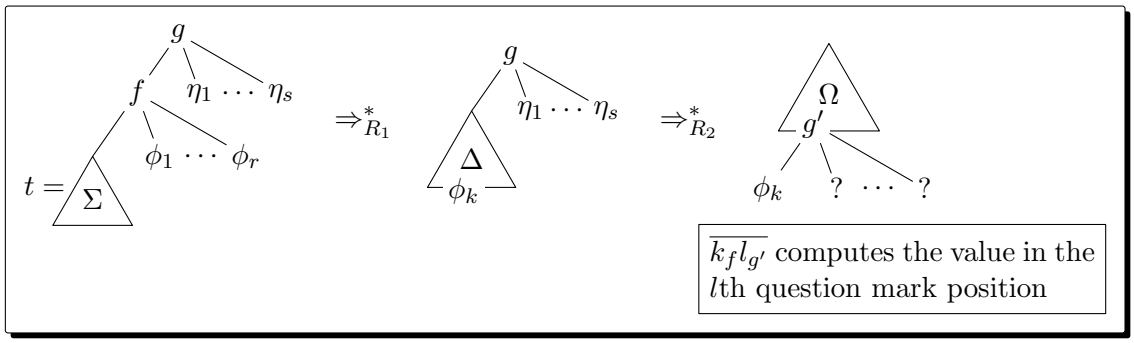

Figure 2: Role of the auxiliary states.

The following construction differs from Construction 5.1 in [VK04a] only in that the required dummy symbol *nil* is explicitly added to the output ranked alphabet rather than randomly picking an existing nullary symbol.

Construction 3.1 (direct composition of restricted mtts)
Let $M_1 = (F, \Sigma, \Delta, e_{M_1}, R_1)$ and $M_2 = (G, \Delta, \Omega, e_{M_2}, R_2)$ be mtts such that one of the two is a tdtt, or M_1 is context-linear and M_2 is weakly single-use. Without loss of generality, assume that F and G are disjoint and that M_1 uses recursion variables from U and context variables from Y, whereas M_2 uses V and Z, respectively. Let $\mu = |G|$ and fix some ordering of the states in G such that $G = \{g_1, \ldots, g_\mu\}$. For every $n \in [\mu]$ let $s_n \in \mathbb{N}$ such that $g_n \in G^{(s_n+1)}$. Additionally, set $r_{max} = max(rank(F)) - 1$, $Z_G = \{z_{g_1,1}, \ldots, z_{g_1,s_1}, \ldots, z_{g_\mu,1}, \ldots, z_{g_\mu,s_\mu}\}$, and let $nil \notin \Omega$ be some arbitrary symbol. The mtt $\overline{M_1;M_2} = (H, \Sigma, \Omega \cup \{nil^{(0)}\}, e_{\overline{M_1;M_2}}, \overline{R_1;R_2})$, using recursion variables from U and context variables from $\{y_{k,g} \mid k \in [r_{max}], g \in G\} \cup Z \cup Z_G$, is obtained as follows:

$$H = \{\overline{fg}^{(r*\mu+s+1)} \mid f \in F^{(r+1)}, g \in G^{(s+1)}\} \cup \{\overline{k_f l_g}^{(r*\mu+|Z_G|+1)} \mid f \in F^{(r+1)}, g \in G^{(s+1)}, k \in [r], l \in [s]\},$$

$e_{\overline{M_1;M_2}} = nf(\Rightarrow_{R_2 \cup Pair}, e_{M_2}[v_1 \leftarrow e_{M_1}])$, and

$\overline{R_1;R_2}$ contains:

- for every $f \in F^{(r+1)}$, $g \in G^{(s+1)}$, and $\sigma \in \Sigma^{(p)}$ the rule

  $$\overline{fg}\ (\sigma\ u_1 \cdots u_p)\ y_{1,g_1} \cdots y_{r,g_\mu}\ z_1 \cdots z_s \to nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, g\ rhs_{M_1,f,\sigma}\ z_1 \cdots z_s)$$

- and for every $f \in F^{(r+1)}$, $g \in G^{(s+1)}$, $k \in [r]$, $l \in [s]$, and $\sigma \in \Sigma^{(p)}$ the rule

  $$\overline{k_f l_g}\ (\sigma\ u_1 \cdots u_p)\ y_{1,g_1} \cdots y_{r,g_\mu}\ z_{g_1,1} \cdots z_{g_\mu,s_\mu} \to nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, \varrho)\,,$$

  where $\varrho = nil$ if $rhs_{M_1,f,\sigma}$ does not contain the context variable y_k; otherwise, $\varrho = par_{M_2, rhs_{M_1,f,\sigma}}(\pi, g, l)$, where $\pi \in paths(rhs_{M_1,f,\sigma})$ is the path of the unique (because M_1 is context-linear by the construction's preconditions if the ranges of the above quantifications of f, g, k, and l are nonempty, given that then neither M_1 nor M_2 can be a tdtt) occurrence of y_k in $rhs_{M_1,f,\sigma}$.

The rewrite systems Pre (over $G \cup \{y_k^{(0)} \mid k \in [r_{max}]\} \cup \{y_{k,g}^{(0)} \mid k \in [r_{max}], g \in G\}$ and Z) and $Pair$ (over $F \cup G \cup H \cup \{nil\}$ and $\{u\} \cup Y' \cup Z$) are defined as follows:

Pre : $g\ y_k\ z_1 \cdots z_s$	$\to y_{k,g}$	$\forall g \in G^{(s+1)}, k \in [r_{max}]$
$Pair$: $g\ (f\ u\ y'_1 \cdots y'_r)\ z_1 \cdots z_s \to$		$\forall g \in G^{(s+1)}, f \in F^{(r+1)}$
$\overline{fg}\ u\ nest_f(1, g_1, \emptyset)[z_{g,l} \leftarrow z_l \mid l \in [s],$		
$z_{g',l} \leftarrow nil \mid g' \in G^{(s'+1)} \setminus \{g\}, l \in [s']]$		
$\cdots$		
$nest_f(r, g_\mu, \emptyset)[z_{g,l} \leftarrow z_l \mid l \in [s],$		
$z_{g',l} \leftarrow nil \mid g' \in G^{(s'+1)} \setminus \{g\}, l \in [s']]$		
$z_1 \cdots z_s$		

For every $f \in F^{(r+1)}$ the function $nest_f : [r] \times G \times \mathcal{P}([r] \times G) \longrightarrow T_{G \cup H \cup \{nil\}}(\{u\} \cup Y'_r \cup Z_G)$ is defined by setting for every $k \in [r]$, $g' \in G^{(s'+1)}$, and $\mathcal{C} \subseteq [r] \times G$:

$$nest_f(k, g', \mathcal{C}) = \begin{cases} nil & \text{if } (k, g') \in \mathcal{C} \\ g'\ y'_k\ (\overline{k_f 1_{g'}}\ u\ nest_f(1, g_1, \mathcal{C} \cup \{(k, g')\}) \cdots nest_f(r, g_\mu, \mathcal{C} \cup \{(k, g')\}) & \\ \qquad z_{g_1,1} \cdots z_{g_\mu,s_\mu}) & \\ \qquad \cdots & \\ \qquad (\overline{k_f s'_{g'}}\ u\ nest_f(1, g_1, \mathcal{C} \cup \{(k, g')\}) \cdots nest_f(r, g_\mu, \mathcal{C} \cup \{(k, g')\}) & \\ \qquad z_{g_1,1} \cdots z_{g_\mu,s_\mu}) & \text{otherwise.} \end{cases}$$

With these definitions, the reduction relations $\Rightarrow_{R_2 \cup Pair}$ and $\Rightarrow_{R_2 \cup Pre \cup Pair}$ are confluent and terminating, which justifies the above uses of respective normal forms.

For every $p \in \mathbb{N}$, $r \in \mathbb{N}_+$ with $(r+1) \in rank(F)$, and $\phi \in RHS(F, \Delta, U_p, Y_r)$ the function $par_{M_2,\phi} : \{(\pi, g, l) \mid \pi \in paths(\phi), lab(\phi, \pi) \notin U_p, g \in G^{(s+1)}, l \in [s]\} \longrightarrow T_{F \cup G \cup H \cup \Delta \cup \Omega \cup \{nil\}}(U_p \cup Y_r \cup Z_G)$ is defined by induction over the prefix-order of paths in ϕ as follows. For every $g \in G^{(s+1)}$ and $l \in [s]$:

– $par_{M_2,\phi}(\varepsilon, g, l) = z_{g,l}$

– For every $j \in \mathbb{N}_+$ and $\pi j \in paths(\phi)$ with $lab(\phi, \pi j) \notin U_p$ by case distinction:

Case a: $lab(\phi, \pi) = f$ for some $f \in F^{(r'+1)}$, where $j - 1 \in [r']$. Then:

$$\begin{aligned} par_{M_2,\phi}(\pi j, g, l) = (\overline{(j-1)_f l_g}\ u\ & nest_f(1, g_1, \{(j-1, g)\}) \\ & \cdots \\ & nest_f(r', g_\mu, \{(j-1, g)\})\ z_{g_1,1} \cdots z_{g_\mu,s_\mu}) \\ [u & \leftarrow lab(\phi, \pi 1), \\ y'_1, \ldots, y'_{r'} & \leftarrow sub(\phi, \pi 2), \ldots, sub(\phi, \pi(r'+1)), \\ z_{g_1,1}, \ldots, z_{g_\mu,s_\mu} & \leftarrow par_{M_2,\phi}(\pi, g_1, 1), \ldots, par_{M_2,\phi}(\pi, g_\mu, s_\mu)]. \end{aligned}$$

Case b: $lab(\phi, \pi) = \delta$ for some $\delta \in \Delta^{(q)}$, where $j \in [q]$. If no call to g on v_j exists in any of the right-hand sides of the δ-rules of M_2, then $par_{M_2,\phi}(\pi j, g, l) = nil$. Otherwise, there must be exactly one δ-rule containing such a call and that call must be unique in the rule's right-hand side because M_2 is weakly single-use by the construction's preconditions if the ranges of the above quantifications of r, g, and l are nonempty, given that then neither M_1 nor M_2 can be a tdtt, and because $g \in G^{(s+1)}$ and $l \in [s]$ imply that g is a nonunary state. Hence, there are unique $g' \in G^{(s'+1)}$ and $\psi_1, \ldots, \psi_s \in RHS(G, \Omega, V_q, Z_{s'})$ such that a rule of the form

$$g'\ (\delta\ v_1 \cdots v_q)\ z_1 \cdots z_{s'} \to \cdots (g\ v_j\ \psi_1 \cdots \psi_s) \cdots$$

is in R_2. Then:

$$\begin{aligned} par_{M_2,\phi}(\pi j, g, l) = \psi_l[v_1, \ldots, v_q & \leftarrow sub(\phi, \pi 1), \ldots, sub(\phi, \pi q), \\ z_1, \ldots, z_{s'} & \leftarrow par_{M_2,\phi}(\pi, g', 1), \ldots, par_{M_2,\phi}(\pi, g', s')]. \end{aligned}$$ □

The following theorem was proved in [Voi01]; a more detailed account is in [VK04b].

Theorem 3.2 (correctness of Constr. 3.1; Theorem 5.2 of [VK04a])
$\tau_{M_1} ; \tau_{M_2} = \tau_{\overline{M_1;M_2}}$ □

For a detailed protocol of the presented construction in action for two concrete mtts, please consult Section 5.2 of [VK04a]. Here we give only the values of *nest*-functions that are different from *nil* in its application to the running example, and the corresponding composition result.

Example 3.3 (composition of the mtts from Example 2.2)
When applying Construction 3.1 to the mtts M_{spine} and M_{pfx}, the following non-*nil* values of *nest*-functions are relevant:

$$\begin{aligned} nest_{f_1}(1, g, \emptyset) &= g\ y'_1\ (\overline{1_{f_1} 1_g}\ u\ nil\ z_{g,1}) \\ nest_{f_2}(1, g, \emptyset) &= g\ y'_1\ (\overline{1_{f_2} 1_g}\ u\ nil\ (g\ y'_2\ (\overline{2_{f_2} 1_g}\ u\ nil\ nil\ z_{g,1}))\ z_{g,1}) \\ nest_{f_2}(1, g, \{(2, g)\}) &= g\ y'_1\ (\overline{1_{f_2} 1_g}\ u\ nil\ nil\ z_{g,1}) \\ nest_{f_2}(2, g, \emptyset) &= g\ y'_2\ (\overline{2_{f_2} 1_g}\ u\ (g\ y'_1\ (\overline{1_{f_2} 1_g}\ u\ nil\ nil\ z_{g,1}))\ nil\ z_{g,1}) \\ nest_{f_2}(2, g, \{(1, g)\}) &= g\ y'_2\ (\overline{2_{f_2} 1_g}\ u\ nil\ nil\ z_{g,1}). \end{aligned}$$

The resulting mtt is $\overline{M_{spine};M_{pfx}} = (\{\overline{f_1 g}^{(3)}, \overline{f_2 g}^{(4)}, \overline{1_{f_1} 1_g}^{(3)}, \overline{1_{f_2} 1_g}^{(4)}, \overline{2_{f_2} 1_g}^{(4)}\}, \Sigma_{mon}, \Omega_{mon} \cup \{nil^{(0)}\}, e_{\overline{M_{spine};M_{pfx}}}, \overline{R_{spine};R_{pfx}})$ with rules

$\overline{R_{spine};R_{pfx}}$:

$$
\begin{array}{lllll}
\overline{f_1 g} & (\alpha\ u_1)\ y_{1,g}\ z_1 & & \to \alpha\ (\overline{f_1 g}\ u_1\ y_{1,g}\ z_1) \\
\overline{f_1 g} & (\beta\ u_1)\ y_{1,g}\ z_1 & & \to \overline{f_2 g}\ u_1\ y_{1,g}\ (\beta\ (\epsilon\ (\overline{2_{f_2} 1_g}\ u_1\ y_{1,g}\ nil\ z_1)))\ z_1 \\
\overline{f_1 g} & \epsilon & y_{1,g}\ z_1 & \to y_{1,g} \\
\overline{f_2 g} & (\alpha\ u_1)\ y_{1,g}\ y_{2,g}\ z_1 & & \to \alpha\ (\overline{f_1 g}\ u_1\ (\delta\ y_{1,g})\ z_1) \\
\overline{f_2 g} & (\beta\ u_1)\ y_{1,g}\ y_{2,g}\ z_1 & & \to \overline{f_2 g}\ u_1\ y_{1,g}\ (\beta\ (\beta\ y_{2,g}))\ z_1 \\
\overline{f_2 g} & \epsilon & y_{1,g}\ y_{2,g}\ z_1 & \to \delta\ y_{1,g} \\
\overline{1_{f_1} 1_g} & (\alpha\ u_1)\ y_{1,g}\ z_{g,1} & & \to \overline{1_{f_1} 1_g}\ u_1\ nil\ z_{g,1} \\
\overline{1_{f_1} 1_g} & (\beta\ u_1)\ y_{1,g}\ z_{g,1} & & \to \overline{1_{f_2} 1_g}\ u_1\ nil\ (\beta\ (\epsilon\ (\overline{2_{f_2} 1_g}\ u_1\ nil\ nil\ z_{g,1})))\ z_{g,1} \\
\overline{1_{f_1} 1_g} & \epsilon & y_{1,g}\ z_{g,1} & \to z_{g,1} \\
\overline{1_{f_2} 1_g} & (\alpha\ u_1)\ y_{1,g}\ y_{2,g}\ z_{g,1} & & \to y_{2,g} \\
\overline{1_{f_2} 1_g} & (\beta\ u_1)\ y_{1,g}\ y_{2,g}\ z_{g,1} & & \to \overline{1_{f_2} 1_g}\ u_1\ nil\ (\beta\ (\beta\ y_{2,g}))\ z_{g,1} \\
\overline{1_{f_2} 1_g} & \epsilon & y_{1,g}\ y_{2,g}\ z_{g,1} & \to y_{2,g} \\
\overline{2_{f_2} 1_g} & (\alpha\ u_1)\ y_{1,g}\ y_{2,g}\ z_{g,1} & & \to \overline{1_{f_1} 1_g}\ u_1\ nil\ z_{g,1} \\
\overline{2_{f_2} 1_g} & (\beta\ u_1)\ y_{1,g}\ y_{2,g}\ z_{g,1} & & \to \overline{2_{f_2} 1_g}\ u_1\ y_{1,g}\ nil\ z_{g,1} \\
\overline{2_{f_2} 1_g} & \epsilon & y_{1,g}\ y_{2,g}\ z_{g,1} & \to z_{g,1}
\end{array}
$$

and initial expression $e_{\overline{M_{spine};M_{pfx}}} = \overline{f_1 g}\ u_1\ (\epsilon\ (\overline{1_{f_1} 1_g}\ u_1\ nil\ \gamma))\ \gamma$. □

The mtt obtained in the previous example is unnecessarily complicated because—for the sake of generality—the composition construction tends to introduce context parameters that are superfluous in the sense that they will never (for no possible input tree) influence the output generated by a state. Here this is the case for the second context parameter of $\overline{f_1 g}$, the second and third context parameters of $\overline{f_2 g}$, the first context parameter of $\overline{1_{f_1} 1_g}$, the first and third context parameters of $\overline{1_{f_2} 1_g}$, and the first and second context parameters of $\overline{2_{f_2} 1_g}$. In a practical implementation such as [Reu03] the mtt resulting from a composition should be post-processed to eliminate such ballast. An appropriate transformation detecting and removing all superfluous context parameters of an mtt was developed already in Section 4.1 of [Voi01], based on a straightforward fixpoint construction. For $\overline{M_{spine};M_{pfx}}$ it yields an mtt with states of reduced ranks, initial expression $\overline{f_1 g}\ u_1\ (\epsilon\ (\overline{1_{f_1} 1_g}\ u_1\ \gamma))$, and simplified rules—explicitly listed in [Voi04a]—obtained by eliminating the context parameters indicated above from all calls in left- and right-hand sides.

4 Formal efficiency analysis

We want to formally relate the efficiency of a composed program obtained by Construction 3.1 (plus post-processing by eliminating superfluous context parameters) to the efficiency of the original program. From the construction it is obvious that intermediate data structures (over the ranked alphabet Δ) produced by M_1 in the

original program do not appear in the composed program. Thus, no memory cells for this intermediate result have to be allocated in the heap and later be deallocated by the garbage collector. To make sure that this beneficial effect on **space** usage is not in vain, the composed program should not take more **time** to compute its output than the original program. A reasonable notion for the "cost" of a computation to consider is therefore the number of reduction steps required to reach the final output for a given input. While the reduction relation used to provide the semantics of an mtt in Section 2 is in general nondeterministic (though confluent), implementations of functional languages are typically based on deterministic reduction strategies. Since we are interested in applying tree transducer composition to lazy languages, we study *call-by-need reduction* [Wad71] (leftmost-outermost reduction with sharing). Even though the pure number of reduction steps is a somewhat coarse efficiency measure, abstracting from the fact that not every step necessarily requires the same amount of time in a concrete implementation, it is usually sufficient in practice. For example, classical deforestation is successfully applied in a production compiler for the lazy functional language Clean [AGS03], despite the fact that the only known formal efficiency statement regarding this transformation is that it does not increase the number of call-by-need reduction steps for linear programs (the number of *call-by-name* steps, respectively, for arbitrary programs [San96a], where call-by-name means leftmost-outermost reduction without sharing, which in turn is similar to outside-in or OI derivation, not enforcing left-to-right evaluation, in the tree transducer literature [ES77]).

Assume that the functions $cbn_{M_1;M_2} : T_\Sigma \longrightarrow \mathbb{N}$ and $cbn_{\overline{M_1;M_2}} : T_\Sigma \longrightarrow \mathbb{N}$ associate to every input tree $t \in T_\Sigma$ the number of call-by-need steps required by the original and the composed program, respectively, to compute the corresponding output tree $(\tau_{M_1} ; \tau_{M_2})(t) = \tau_{\overline{M_1;M_2}}(t)$. That is, $cbn_{M_1;M_2}(t)$ denotes the number of call-by-need steps required to reach this output using rules from $R_1 \cup R_2$ on $e_{M_2}[v_1 \leftarrow e_{M_1}[u_1 \leftarrow t]]$, while $cbn_{\overline{M_1;M_2}}(t)$ denotes the number of call-by-need steps required to reach it using rules from $\overline{R_1;R_2}$ on $e_{\overline{M_1;M_2}}[u_1 \leftarrow t]$. The (original) program consisting of rules $R_1 \cup R_2$ and "initial expression" $e_{M_2}[v_1 \leftarrow e_{M_1}]$ will also be referred to as $M_1 ; M_2$ in the following, and similarly for annotated versions of M_1 and M_2.

Example 4.1 (call-by-need reductions for the running example)
Figure 3 shows the call-by-need reduction sequence for $M_{spine} ; M_{pfx}$ on a particular input. The encircled areas correspond to memory cells that have to be allocated for the intermediate data structure. Figure 4 shows the call-by-need reduction sequence for the corresponding composed and post-processed program on the same input. Note that no intermediate data structure is created and that one reduction step less is required. □

Since the elimination of superfluous context parameters has no effect with respect to the number of reduction steps, we need not pay special attention to the post-processing in the efficiency analysis and will instead study the immediate outcome of the composition construction itself. The aim is to find conditions under which

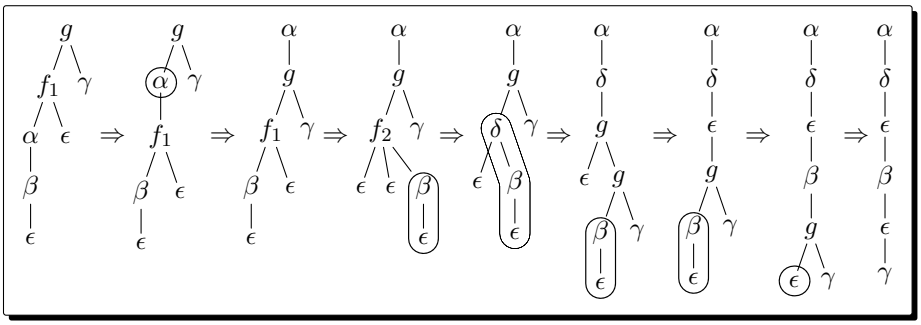

Figure 3: $cbn_{M_{spine};M_{pfx}}(\alpha\ (\beta\ \epsilon)) = 8$.

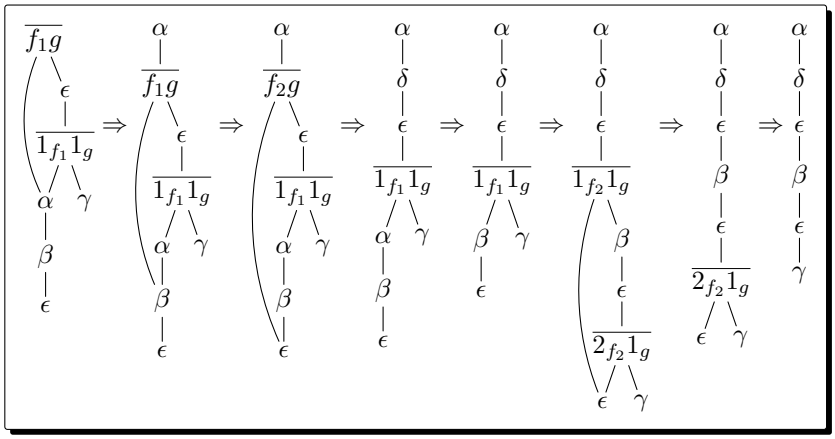

Figure 4: $cbn_{\overline{M_{spine};M_{pfx}}}(\alpha\ (\beta\ \epsilon)) = 7$.

for every $t \in T_\Sigma$ we have $cbn_{\overline{M_1;M_2}}(t) \leq cbn_{M_1;M_2}(t)$. Actually, it would also be satisfactory to find some small constant $d \in \mathbb{N}$ such that for every possible input tree t the following holds:

$$cbn_{\overline{M_1;M_2}}(t) \leq cbn_{M_1;M_2}(t) + d \tag{3}$$

This would mean that, whatever the input is, the composed program performs at most d additional reduction steps but avoids the allocation and eventual deallocation of an intermediate data structure of arbitrary size. Note that the time costs associated to the prevented memory allocations and deallocations are neglected in our analysis, but would anyway weigh in on the plus side for our construction in practice. A negative proviso with respect to practical relevance of our analysis results is the imprecision caused by abstracting from the different costs potentially inherent in different (kinds of) reduction steps. Measurements in [Reu03] demonstrated that such imprecision is particularly manifest in the presence of *tail calls*, potentially leading to the situation that the composed program is slower than the original one, even though the total number of reduction steps performed by it is not bigger. To maintain the applicability of our analysis results in an optimizing compiler, we have shown in [Voi05] how to adjust the criteria produced by our efficiency study so that tail calls are taken into account appropriately. Another aspect that is considered in [Voi05] is the possibility that the final output of the program to be transformed is to be evaluated only partially (rather than until normal form is reached). In the remainder of the current paper, however, we will simply take (3) as criterion for efficiency nondeterioration. As mentioned above, this agrees with what has been done for classical deforestation.

4.1 Outline of the analysis

In the following subsections we will work towards two theorems (Theorems 4.19 and 4.39) providing conditions as desired. Since computation time is an intensional property of a program, i.e., it cannot be determined by solely observing which output is generated for which input according to the program's (extensional) semantics, methods developed to reason about extensional properties of programs are not readily applicable to study it. For example, the correctness proof for our composition construction in [VK04b] makes heavy use of reordering of normal form computations, justified by the fact that the reduction relation induced by the rules of any mtt is confluent. But since the efficiency analysis we are aiming at here is concerned with a particular fixed reduction strategy, we cannot make use of confluence in this way now. A seemingly naïve—but often quite effective, see related work in Section 5.3—strategy in such a situation is to externalize the intensional property, e.g. by transforming or annotating programs, thus making it more accessible. Our strategy can be summarized as follows:

– First, we enrich the original and the composed program, respectively, by annotating their rules with special symbols and adding new rules in such a way

that the output trees computed for a particular input tree reflect the numbers of *call-by-need* reduction steps performed by the programs under consideration for that input. This follows the development in [Voi02].

– Then, we devise another annotation for the original program such that applying the composition construction to the thus annotated original program yields essentially the annotated version of the composed program. What was a simple exercise for the special case considered in [Voi02] requires some imagination for the general case. In fact, we consider the presented annotation (and the associated proof establishing its suitability) a key contribution because on an intuitive level it exposes the essence of the rather complicated construction from [VK04a] in a surprisingly elegant way, while on the technical side it allows to completely remove the details of the composition construction from concern in the remainder of the analysis.

– Left with annotated versions of the original program only, we can combine and manipulate annotations, trading those special symbols that correspond to steps of the composed program against those that correspond to steps of the original program. The aim is to arrive at an annotation for which simple conditions suffice to guarantee that for every input the number of symbols produced of the former kind is outweighed by that of symbols produced of the latter kind. Some of the involved manipulations will be safe—i.e., will not lead to an overestimation of the number of steps performed by the original program or an underestimation of the number of steps performed by the composed program—only under certain conditions that will also be recorded.

4.2 Annotating programs to reflect computational costs

For the remainder of the paper, let $M_1 = (F, \Sigma, \Delta, e_{M_1}, R_1)$ and $M_2 = (G, \Delta, \Omega, e_{M_2}, R_2)$ be two fixed mtts to which Construction 3.1 is applicable, yielding the mtt $\overline{M_1;M_2} = (H, \Sigma, \Omega \cup \{nil\}, e_{\overline{M_1;M_2}}, \overline{R_1;R_2})$. We use the notions and variable naming conventions from Construction 3.1. Special unary symbols $\diamond$, $\bullet$, $\circ$, and $\star$, assumed not to occur in $\Sigma \cup \Delta \cup \Omega \cup F \cup G$, will be used in annotated versions of the mtts under consideration.

The natural way to keep track of reduction steps in the output is by generating a dedicated "tick"-symbol whenever some rewrite rule is applied, e.g. by replacing every rule

$$f\ (\sigma\ u_1 \cdots u_p)\ y_1 \cdots y_r \to rhs$$

by a corresponding one with an additional symbol on top of its right-hand side:

$$f\ (\sigma\ u_1 \cdots u_p)\ y_1 \cdots y_r \to \diamond\ rhs.$$

The dedicated symbol then appears scattered over the output tree generated by computing with such annotated rules. Compared to an attempt of gathering all the information about performed reduction steps in one place, e.g. at the root of

the complete output or as an additional function result, this has the advantage that the origin of computational costs is remembered. In particular, discarding a piece of output produced inside an unused argument position also removes from consideration any computational costs associated with it. Moreover, the locational information associated with the positions of $\diamond$-symbols is important if the output produced by the annotated rules above is not yet the final output. Recall that in the original program—computing with rules $R_1 \cup R_2$ on an instantiation of $e_{M_2}[v_1 \leftarrow e_{M_1}]$—the rules R_1 are used to produce an intermediate result then consumed by states of M_2. The latter must therefore not only record their own reduction steps properly in the final output, but also propagate information about the computational costs of M_1 for producing the intermediate result. By simply including in their count any $\diamond$-symbols they come across, they can count exactly those reduction steps of M_1 that produce parts of the intermediate result actually demanded during computation of the final output. The following definition gives annotated versions of M_1 and M_2 along these lines.

Definition 4.2 ($M_1^{\rightarrow\diamond}$ and $M_2^{\diamond\rightarrow\bullet}$)
The mtts $M_1^{\rightarrow\diamond} = (F, \Sigma, \Delta \cup \{\diamond\}, e_{M_1}, R_1^{\rightarrow\diamond})$ and $M_2^{\diamond\rightarrow\bullet} = (G, \Delta \cup \{\diamond\}, \Omega \cup \{\bullet\}, e_{M_2}, R_2^{\diamond\rightarrow\bullet})$ have rules as follows:

$R_1^{\rightarrow\diamond}$:		
$f\ (\sigma\ u_1 \cdots u_p)\ y_1 \cdots y_r$	$\rightarrow \diamond\ rhs_{M_1,f,\sigma}$	$\forall f \in F^{(r+1)}, \sigma \in \Sigma^{(p)}$
$R_2^{\diamond\rightarrow\bullet}$:		
$g\ (\delta\ v_1 \cdots v_q)\ z_1 \cdots z_s$	$\rightarrow \bullet\ rhs_{M_2,g,\delta}$	$\forall g \in G^{(s+1)}, \delta \in \Delta^{(q)}$
$g\ (\diamond\ v_1)\ z_1 \cdots z_s$	$\rightarrow \bullet\ (g\ v_1\ z_1 \cdots z_s)$	$\forall g \in G^{(s+1)}$

Note that the initial expressions remain unchanged. □

If no further restrictions are imposed, then $M_1^{\rightarrow\diamond}; M_2^{\diamond\rightarrow\bullet}$ counts the **call-by-name** steps of $M_1; M_2$, rather than call-by-need steps, because information about sharing is lost in the annotated output tree. An easy way to circumvent this would be to restrict input programs to be linear, so that no sharing can occur. In fact, this is what is typically done in efficiency arguments about classical deforestation, and it also underlies the results from [Küh99]. But the particular recursion scheme of mtts allows for more liberal conditions to be used here. Since the difference between call-by-name and call-by-need shows only in situations where a call-by-name step would duplicate a subexpression containing a function call by substituting it for several occurrences of the same variable in the right-hand side of a rewrite rule, it suffices to outlaw such situations. With respect to potential duplication of calls nested in context parameter positions of other calls, this can obviously be done by requiring both M_1 and M_2 to be context-linear or basic. To avoid potential duplication of calls to states of M_1 that appear in a partially evaluated intermediate result which is recursed by M_2, we additionally impose recursion-linearity on the latter. Under the combined restrictions we obtain the following, corresponding to Lemma 2 in [Voi02]. Recall that $|\cdot|_\bullet$ is a function returning the number of $\bullet$-symbols in a tree.

Lemma 4.3 ($M_1^{\to\diamond}\,;M_2^{\diamond\to\bullet}$ counts the call-by-need steps of $M_1\,;M_2$)
If M_1 is context-linear or basic and M_2 is atmost, then:

$$cbn_{M_1;M_2} = \tau_{M_1^{\to\diamond}}\,;\tau_{M_2^{\diamond\to\bullet}}\,;|\,\cdot\,|_\bullet.$$ □

In [Voi04a] a formal proof is given by explicitly defining call-by-name reduction (in Section 4.1), establishing that $\tau_{M_1^{\to\diamond}}\,;\tau_{M_2^{\diamond\to\bullet}}\,;|\,\cdot\,|_\bullet$ computes exactly the number of steps in a call-by-name reduction of the original program for a particular input (in Lemma 4.5), and showing that for context-linear or basic M_1 and atmost M_2 no duplication of function calls can take place during such a reduction (in Section 4.9). Since those definitions and proofs hold no big surprises, we omit them here and leave it at the intuitive arguments above. The reader wanting to follow up on the development in [Voi04a] (or [Voi05]), however, should note that there the $cbn_{M_1;M_2}$- and $cbn_{\overline{M_1;M_2}}$-functions refer to call-by-name rather than call-by-need reduction. A further (somewhat subtle) point to note is that both in the case of call-by-name and in the case of call-by-need reduction we count steps performed starting with an instantiation of $e_{M_2}[v_1 \leftarrow e_{M_1}]$. One could argue that this is not entirely appropriate in the call-by-need case because the substitution of e_{M_1} for v_1 leads to copies of the former if e_{M_2} contains several occurrences of v_1, thus forgoing potential benefit from sharing. This could be circumvented by revising the definition of recursion-linearity, and thus also the definition of atmostness, by additionally imposing linearity of the initial expression in the input variable. For harmony and simplicity, we refrain from doing so or replacing the "textual substitution" in $e_{M_2}[v_1 \leftarrow e_{M_1}]$ with a shared representation. On the formal side, this is correct because for $cbn_{M_1;M_2}$ as defined in the present paper Lemma 4.3 holds without additional provisos. On the practical side, the issue is a non-issue because a compiler will usually only apply the transformation starting with initial expressions containing exactly one call from each mtt, anyway. (For example, this is what the implementation [Reu03] does.)

Example 4.4 (counting call-by-need steps for the running example)
According to Definition 4.2, the mtts $M_{spine}^{\to\diamond}$ and $M_{pfx}^{\diamond\to\bullet}$ have rules

$$\begin{array}{lll}
R_{spine}^{\to\diamond}\colon & f_1\ (\alpha\ u_1)\ y_1 \to \diamond\ (\alpha\ (f_1\ u_1\ y_1)) & f_2\ (\alpha\ u_1)\ y_1\ y_2 \to \diamond\ (\alpha\ (f_1\ u_1\ (\delta\ y_1\ y_2)))\\
& f_1\ (\beta\ u_1)\ y_1 \to \diamond\ (f_2\ u_1\ y_1\ (\beta\ \epsilon)) & f_2\ (\beta\ u_1)\ y_1\ y_2 \to \diamond\ (f_2\ u_1\ y_1\ (\beta\ (\beta\ y_2)))\\
& f_1\ \epsilon\ \ \ y_1 \to \diamond\ y_1 & f_2\ \epsilon\ \ \ y_1\ y_2 \to \diamond\ (\delta\ y_1\ y_2)
\end{array}$$

and

$$\begin{array}{ll}
R_{pfx}^{\diamond\to\bullet}\colon & g\ (\delta\ v_1\ v_2)\ z_1 \to \bullet\ (\delta\ (g\ v_1\ (g\ v_2\ z_1)))\\
& g\ (\alpha\ v_1)\ \ \ z_1 \to \bullet\ (\alpha\ (g\ v_1\ z_1))\\
& g\ (\beta\ v_1)\ \ \ z_1 \to \bullet\ (\beta\ (g\ v_1\ z_1))\\
& g\ \epsilon\ \ \ z_1 \to \bullet\ (\epsilon\ z_1)\\
& g\ (\diamond\ v_1)\ \ \ z_1 \to \bullet\ (g\ v_1\ z_1)
\end{array}$$

respectively. Figure 5 shows an example computation using these annotated rules, starting from the same input as in Figure 3. Recall that the notation $\bullet^n$ stands for the n-fold composition of $\bullet$ (in its interpretation as a function on trees). Then,

note that the number of •-symbols in the final output is the same as the number of call-by-need reduction steps required in Figure 3. Further note that, of course, this final output is independent of the chosen reduction strategy: any other order of applying rules from $R^{\rightarrow\diamond}_{spine}$ and $R^{\diamond\rightarrow\bullet}_{pfx}$ would have yielded exactly the same normal form! □

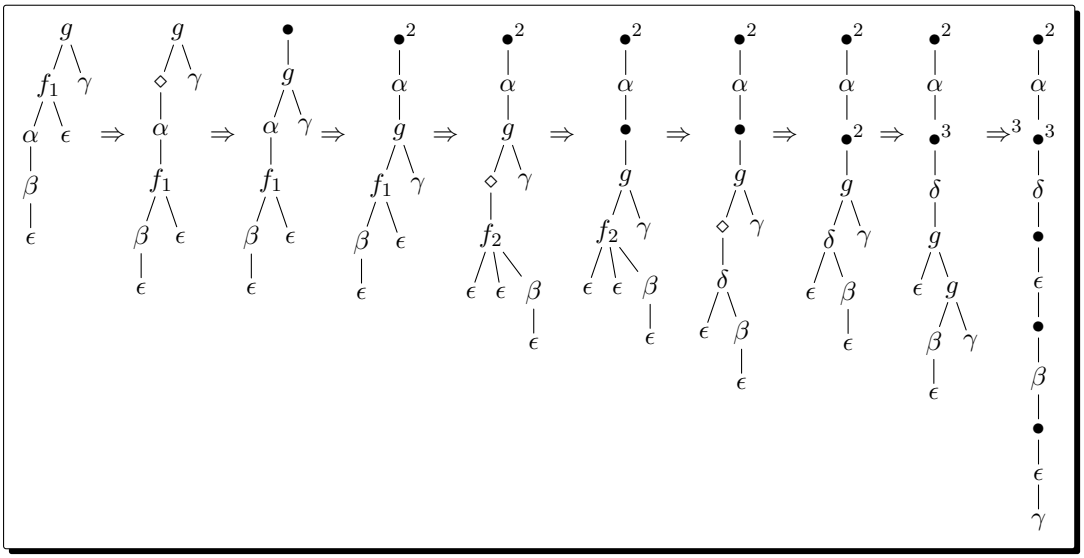

Figure 5: Reduction sequence using $\Rightarrow_{R^{\rightarrow\diamond}_{spine} \cup R^{\diamond\rightarrow\bullet}_{pfx}}$.

For the composed program things are simpler than above because it consists of only a single mtt, so no propagation of annotated symbols from an intermediate to the final result is necessary, and because we can be satisfied with obtaining a pessimistic approximation of its computation cost. By just putting another special symbol on top of the right-hand side of every rule of $\overline{M_1;M_2}$ we obtain the following.

Definition 4.5 ($\overline{M_1;M_2}^{\rightarrow\circ}$)
The mtt $\overline{M_1;M_2}^{\rightarrow\circ} = (H, \Sigma, \Omega \cup \{nil, \circ\}, e_{\overline{M_1;M_2}}, \overline{R_1;R_2}^{\rightarrow\circ})$ has rules as follows:

$$\frac{}{\overline{R_1;R_2}^{\rightarrow\circ}:\quad h\ (\sigma\ u_1 \cdots u_p)\ \cdots \rightarrow \circ\ rhs_{\overline{M_1;M_2},h,\sigma} \quad \forall h \in H, \sigma \in \Sigma^{(p)}}$$

where on the left-hand sides the same context variables occur (in the same order) as in the corresponding rules of $\overline{M_1;M_2}$. □

Again, $\tau_{\overline{M_1;M_2}^{\rightarrow\circ}}; |\,.\,|_\circ$ computes exactly the number of **call-by-name** steps performed by the composed program for a particular input (cf. Lemma 4.8 of [Voi04a]). Note that this, of course, does not depend on the particular form of $\overline{M_1;M_2}$ (e.g., also $M_1^{\rightarrow\diamond}$ counts the call-by-name steps of M_1). Since call-by-need always performs at most as many reduction steps as a call-by-name strategy, we obtain the following.

Lemma 4.6 ($\overline{M_1;M_2}^{\rightarrow\circ}$ **approximates the call-by-need steps of** $\overline{M_1;M_2}$)
$cbn_{\overline{M_1;M_2}} \leq \tau_{\overline{M_1;M_2}^{\rightarrow\circ}}; |\,.\,|_\circ$ □

It seems appropriate to discuss the consequences of settling for an approximation here. Since we essentially compare call-by-name efficiency of the composed program to call-by-need efficiency of the original program, we clearly introduce a certain imprecision into our analysis, but not more so than is the case in comparable efficiency studies for other transformation techniques. In fact, it seems questionable whether a more faithful analysis of the call-by-need efficiency of the composed program is at all possible in general, given that the composition construction itself proceeds intrinsically call-by-name and loses a certain amount of sharing that cannot be recovered without leaving the framework of mtts. However, it will often be the case that the inequality from Lemma 4.6 turns into an equality. For the running example it is indeed so, as is easy to see because the right-hand sides of the rules after post-processing (given explicitly on page 15 of [Voi04a]) are linear in all variables. We do not attempt here to track down conditions on the original program under which we get an equational version of Lemma 4.6 because anyway we will see more inequalities as the analysis proceeds. For most of those, in particular for the ones in Lemmas 4.23, 4.24, 4.33, 4.35, 4.36, and 4.37, we have in [Voi04a] proved that equality holds under certain additional conditions. While this exercise served to develop some sense of just how frequently the pessimistic approximations will manifest in practice, it clearly does not allow for a single additional instance of tree transducer composition to be flagged as efficiency nondeteriorating; hence, we will not repeat it in the present paper. In conclusio: (i) for showing that a composed program is more efficient than the original one, a count based on inequalities (in the proper direction) is sufficient, and (ii) the primary concern when evaluating the usefulness of sufficient conditions is how many typical examples they cover in practice. In Section 5.2 we will argue that our efficiency analysis fares pretty well in the latter respect.

4.3 Pushing annotation through composition

Through the characterization of $cbn_{M_1;M_2}$ and $cbn_{\overline{M_1;M_2}}$ in terms of $\tau_{M_1^{\rightarrow\diamond}}$, $\tau_{M_2^{\diamond\rightarrow\bullet}}$, and $\tau_{\overline{M_1;M_2}^{\rightarrow\circ}}$ we have freed ourselves from concerns about a particular reduction strategy. Since the tree transduction induced by an mtt is defined without imposing any policy on reductions computing the normal forms of instantiations of its initial expression, we can reason about the outputs generated by the mtts $M_1^{\rightarrow\diamond}$, $M_2^{\diamond\rightarrow\bullet}$, and $\overline{M_1;M_2}^{\rightarrow\circ}$ with much more ease than about the lengths of specific reduction sequences involving M_1, M_2, and $\overline{M_1;M_2}$. The externalization will thus prove fruitful.

One hurdle for analyzing the relation between the number of $\circ$-symbols produced by $\overline{M_1;M_2}^{\rightarrow\circ}$ and the number of $\bullet$-symbols produced by $M_1^{\rightarrow\diamond}$ followed by $M_2^{\diamond\rightarrow\bullet}$ is that the rule structure of the former mtt seems rather unconnected to that of the latter two on first sight. But of course, there is a tight connection in that $M_1^{\rightarrow\diamond}$ and $M_2^{\diamond\rightarrow\bullet}$ are annotated versions of the original mtts M_1 and M_2 while $\overline{M_1;M_2}^{\rightarrow\circ}$ was obtained by annotating the rules produced by our composition construction, given those original mtts as input. The most promising approach to exploit this connection is by trying to get an alternative perspective on $\overline{M_1;M_2}^{\rightarrow\circ}$: rather than viewing it as an annotated version of the composed mtt, can we also view it as the

composition of annotated versions of the original mtts? This would allow us to shift further attention completely to the level of (annotations of) the original program, after all the base on which we aim to formulate sufficient conditions.

The quest for appropriately annotated versions of M_1 and M_2 whose composition according to Construction 3.1 would differ from $\overline{M_1;M_2}$ essentially only by having an added $\circ$-symbol on top of every non-*nil* rule right-hand side (while the effect on rules with the dummy symbol *nil* as right-hand side does not need to meet any particular criteria, given that these rules are irrelevant for the composed mtt's computation) required a more thorough analysis of the internal workings of the composition construction and the interplay between its ingredients than the necessarily cursory explanations in Section 3. The result, however, is surprisingly simple and elegant, and comes with a pretty straightforward proof that it indeed has the desired characteristic. Moreover, it sheds new light on the roles of the states of $\overline{M_1;M_2}$.

Recall that a state $\overline{fg}$ of $\overline{M_1;M_2}$ essentially computes the output of state g of M_2 run on an intermediate result produced by state f of M_1. It consumes its recursion input with the same granularity as f does, namely by pattern-matching against the root symbol. Hence, to chart steps of $\overline{fg}$ in terms of the original program, it makes sense to mark portions of the intermediate result produced by a single step of f (reusing the $\diamond$-symbol):

$$f\ (\sigma\ u_1 \cdots u_p)\ y_1 \cdots y_r \to \diamond\ (\cdots)$$

and to record the consumption of such a portion by g as follows in the final output:

$$g\ (\diamond\ v_1)\ z_1 \cdots z_s \to \circ\ (g\ v_1\ z_1 \cdots z_s).$$

So far, the approach is the same as in [Voi02] for the special case that one of the involved mtts is a tdtt; but for $\overline{k_f l_{g'}}$-states, which were not present in that special case, things become considerably more intricate. These states are used to determine the values of context parameters of M_2's state g' on reaching the positions of context parameters of M_1's state f in an intermediate result produced by f for a given input. To model such behavior, we explicitly record positions in the intermediate result where a context parameter of f was used by marking all context variables in right-hand sides of rules for f with yet another dedicated symbol $\star$. The cost associated in the composed program with accessing a context parameter of g' on a thus marked context parameter of a state of M_1 is reflected by adding the following rule:

$$g'\ (\star\ v_1)\ z_1 \cdots z_{s'} \to g'\ v_1\ (\circ\ z_1)\ \cdots\ (\circ\ z_{s'}).$$

Its effect is illustrated in Figure 6; compare that to Figure 2. The chosen scheme ensures that if some of the context parameters of g' are not required for its computation on a concrete instantiation of v_1, e.g. on ϕ_k in the figure, then the $\circ$-symbols stored in the corresponding positions will not be included in the final count as they will be discarded along with the values they mark. That this is exactly what we want hinges on the fact that a $\overline{k_f l_{g'}}$-computation is necessary only if the corresponding value is really required in the replacement (2) for (1) in Section 3.

Summarizing, we obtain the following annotated versions of M_1 and M_2.

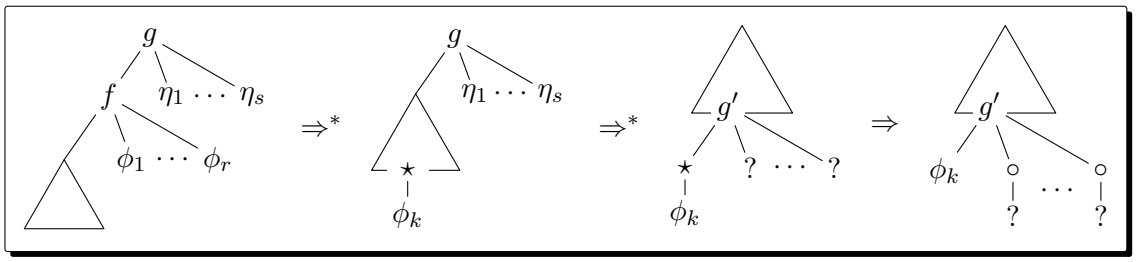

Figure 6: Effects of the $\star$-annotation and associated rules for the second mtt.

Definition 4.7 ($M_1^{\to\diamond\star}$ and $M_2^{\diamond\star\to\circ}$)
The mtts $M_1^{\to\diamond\star} = (F, \Sigma, \Delta \cup \{\diamond, \star\}, e_{M_1}, R_1^{\to\diamond\star})$ and $M_2^{\diamond\star\to\circ} = (G, \Delta \cup \{\diamond, \star\}, \Omega \cup \{\circ\}, e_{M_2}, R_2^{\diamond\star\to\circ})$ have rules as follows:

$R_1^{\to\diamond\star}$:		
$f\ (\sigma\ u_1 \cdots u_p)\ y_1 \cdots y_r$	$\to \diamond\ rhs_{M_1,f,\sigma}[y_k \leftarrow \star\ y_k \mid k \in [r]]$	$\forall f \in F^{(r+1)}, \sigma \in \Sigma^{(p)}$
$R_2^{\diamond\star\to\circ}$:		
$g\ (\delta\ v_1 \cdots v_q)\ z_1 \cdots z_s$	$\to rhs_{M_2,g,\delta}$	$\forall g \in G^{(s+1)}, \delta \in \Delta^{(q)}$
$g\ (\diamond\ v_1)\ z_1 \cdots z_s$	$\to \circ\ (g\ v_1\ z_1 \cdots z_s)$	$\forall g \in G^{(s+1)}$
$g\ (\star\ v_1)\ z_1 \cdots z_s$	$\to g\ v_1\ (\circ\ z_1)\ \cdots\ (\circ\ z_s)$	$\forall g \in G^{(s+1)}$

Note that $M_1^{\to\diamond\star}$ is a tdtt iff M_1 is; likewise for context-linearity. Further, $M_2^{\diamond\star\to\circ}$ is a tdtt iff M_2 is; likewise for weakly single-useness. □

By the remarks in Definition 4.7, applicability of Construction 3.1 to M_1 and M_2 implies its applicability also to $M_1^{\to\diamond\star}$ and $M_2^{\diamond\star\to\circ}$, yielding the mtt $\overline{M_1^{\to\diamond\star};M_2^{\diamond\star\to\circ}} = (H, \Sigma, \Omega \cup \{nil, \circ\}, e_{\overline{M_1^{\to\diamond\star};M_2^{\diamond\star\to\circ}}}, \overline{R_1^{\to\diamond\star};R_2^{\diamond\star\to\circ}})$, which we aim to show equivalent to $\overline{M_1;M_2}^{\to\circ}$. Note that the state set of $\overline{M_1^{\to\diamond\star};M_2^{\diamond\star\to\circ}}$ is the same H as for $\overline{M_1;M_2}$ (and thus for $\overline{M_1;M_2}^{\to\circ}$) because its definition in Construction 3.1 depends only on F and G. Likewise, the $nest_f$-functions and the rewrite systems Pre and $Pair$ are unchanged between the two applications of the composition construction to M_1 and M_2, respectively to $M_1^{\to\diamond\star}$ and $M_2^{\diamond\star\to\circ}$, assuming the same ordering of states in G is used in both cases. The following two auxiliary lemmas study the (non-)impact of the differences which do exist between the rule sets of the mtts involved in those two invocations of the composition construction on certain computations to be performed when constructing the rule right-hand sides for the composed program.

Lemma 4.8 (auxiliary)
For every $f \in F^{(r+1)}$, $g \in G^{(s+1)}$, and $\sigma \in \Sigma$:

$$\begin{aligned} &nf(\Rightarrow_{R_2^{\diamond\star\to\circ} \cup Pre \cup Pair}, g\ rhs_{M_1,f,\sigma}[y_k \leftarrow \star\ y_k \mid k \in [r]]\ z_1 \cdots z_s) \\ =\ &nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, g\ rhs_{M_1,f,\sigma}\ z_1 \cdots z_s). \end{aligned}$$ □

The proof, which can be found in the appendix, relies on the facts that, for every $\delta \in \Delta$, the δ-rules do not differ between $R_2^{\diamond\star\to\circ}$ and R_2, that $rhs_{M_1,f,\sigma}[y_k \leftarrow \star\ y_k \mid k \in [r]]$ contains no $\diamond$-symbols, that it contains $\star$-symbols only atop context variables y_k, and that a reduction $g\ (\star\ y_k)\ \varrho_1 \cdots \varrho_s \Rightarrow_{R_2^{\diamond\star\to\circ}} g\ y_k\ (\circ\ \varrho_1)\ \cdots\ (\circ\ \varrho_s) \Rightarrow_{Pre} y_{k,g}$ for arbitrary

g, y_k, and $\varrho_1, \ldots, \varrho_s$ has the same outcome as a $\Rightarrow_{Pre}$-reduction for a corresponding call to g on (unannotated) y_k.

Lemma 4.9 (auxiliary)
For every $f \in F^{(r+1)}$, $g \in G^{(s+1)}$, $\sigma \in \Sigma$, $\pi \in paths(rhs_{M_1,f,\sigma})$ with $lab(rhs_{M_1,f,\sigma}, \pi) \in Y_r$, and $l \in [s]$:

$$\begin{aligned} &nf(\Rightarrow_{R_2^{\diamond\star\to\circ} \cup Pre \cup Pair}, par_{M_2^{\diamond\star\to\circ}, rhs_{M_1^{\to\diamond\star},f,\sigma}}(1\pi, g, l)) \\ = \; &nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, par_{M_2, rhs_{M_1,f,\sigma}}(\pi, g, l)). \end{aligned}$$ □

The proof can be found in the appendix. Beside the facts that, for every $\delta \in \Delta$, the δ-rules are the same in $R_2^{\diamond\star\to\circ}$ and R_2, that no $\star$-symbols occur between the root of $rhs_{M_1^{\to\diamond\star},f,\sigma} = \diamond\, rhs_{M_1,f,\sigma}[y_k \leftarrow \star\, y_k \mid k \in [r]]$ and the $\star$-symbol at path 1π, and that the use of substitution $\cdot[y_k \leftarrow \star\, y_k \mid k \in [r]]$ makes for no difference between computations using states from G with reduction relations $\Rightarrow_{R_2^{\diamond\star\to\circ} \cup Pre \cup Pair}$ and $\Rightarrow_{R_2 \cup Pre \cup Pair}$, respectively (as in Lemma 4.8), the key is that the recursive calls in $R_2^{\diamond\star\to\circ}$-rules at $\diamond$ are performed with unchanged context parameters.

We can now show that except for rules whose right-hand side is *nil* the mtts $\overline{M_1^{\to\diamond\star}; M_2^{\diamond\star\to\circ}}$ and $\overline{M_1;M_2}^{\to\circ}$ are syntactically the same, taking into account that by Definition 4.5 the initial expression of the latter mtt is $e_{\overline{M_1;M_2}}$ and that by Construction 3.1 and Definition 4.7 the left-hand sides of the rules in $\overline{R_1^{\to\diamond\star}; R_2^{\diamond\star\to\circ}}$ coincide with the left-hand sides of the $\overline{R_1;R_2}$-rules and thus by Definition 4.5 also with those of the $\overline{R_1;R_2}^{\to\circ}$-rules, i.e., not only the state set H and input symbol alphabet Σ, but also the recursion and context variables used are identical between the rule sets.

Lemma 4.10 ($\overline{M_1^{\to\diamond\star}; M_2^{\diamond\star\to\circ}}$ and $\overline{M_1;M_2}^{\to\circ}$ are equal up to dummy rules)

1. $e_{\overline{M_1^{\to\diamond\star}; M_2^{\diamond\star\to\circ}}} = e_{\overline{M_1;M_2}}$
2. For every $h \in H$ and $\sigma \in \Sigma$, if $rhs_{\overline{M_1^{\to\diamond\star}; M_2^{\diamond\star\to\circ}}, h, \sigma}$ is not *nil*, then it is equal to $rhs_{\overline{M_1;M_2}^{\to\circ}, h, \sigma}$.

Proof

1. By Construction 3.1 and Definition 4.7 we have $e_{\overline{M_1^{\to\diamond\star}; M_2^{\diamond\star\to\circ}}} = nf(\Rightarrow_{R_2^{\diamond\star\to\circ} \cup Pair}, e_{M_2}[v_1 \leftarrow e_{M_1}])$ and $e_{\overline{M_1;M_2}} = nf(\Rightarrow_{R_2 \cup Pair}, e_{M_2}[v_1 \leftarrow e_{M_1}])$. Since $e_{M_2}[v_1 \leftarrow e_{M_1}] \in T_{F \cup G \cup \Delta \cup \Omega}(U_1)$ contains no $\diamond$- or $\star$-symbols and since, for every $\delta \in \Delta$, the δ-rules in $R_2^{\diamond\star\to\circ}$ coincide with those in R_2 by Definition 4.7, the two normal forms are equal.

2. By case analysis on $h \in H$:

   Case a: $h = \overline{fg}$ for some $f \in F^{(r+1)}$ and $g \in G^{(s+1)}$. Then:

   $rhs_{\overline{M_1^{\to\diamond\star}; M_2^{\diamond\star\to\circ}}, \overline{fg}, \sigma}$
   $= $ (by Construction 3.1 and Definition 4.7)
   $nf(\Rightarrow_{R_2^{\diamond\star\to\circ} \cup Pre \cup Pair}, g\ (\diamond\, rhs_{M_1,f,\sigma}[y_k \leftarrow \star\, y_k \mid k \in [r]])\ z_1 \cdots z_s)$
   $= $ (by $\Rightarrow_{R_2^{\diamond\star\to\circ}}$, using that $rhs_{M_2^{\diamond\star\to\circ}, g, \diamond} = \circ\ (g\ v_1\ z_1 \cdots z_s)$)

$\circ\ nf(\Rightarrow_{R_2^{\diamond\star\to\circ}\cup Pre\cup Pair}, g\ rhs_{M_1,f,\sigma}[y_k \leftarrow \star\ y_k \mid k \in [r]]\ z_1 \cdots z_s)$
$\quad =$ (by Lemma 4.8)
$\circ\ nf(\Rightarrow_{R_2\cup Pre\cup Pair}, g\ rhs_{M_1,f,\sigma}\ z_1 \cdots z_s)$
$\quad =$ (by Definition 4.5 and Construction 3.1)
$rhs_{\overline{M_1;M_2}^{\to\circ},\overline{fg},\sigma}$

Case b: $h = \overline{k_f l_g}$ for some $f \in F^{(r+1)}$, $g \in G^{(s+1)}$, $k \in [r]$, and $l \in [s]$. Note that $rhs_{M_1^{\to\diamond\star},f,\sigma} = \diamond\ rhs_{M_1,f,\sigma}[y_{k'} \leftarrow \star\ y_{k'} \mid k' \in [r]]$ by Definition 4.7.

- If $rhs_{M_1,f,\sigma}$ does not contain the context variable y_k, then neither does $rhs_{M_1^{\to\diamond\star},f,\sigma}$, hence $rhs_{\overline{M_1^{\to\diamond\star};M_2^{\diamond\star\to\circ}},\overline{k_f l_g},\sigma} = nil$ by Construction 3.1.
- Otherwise, if $lab(rhs_{M_1,f,\sigma},\pi) = y_k$—and thus $sub(rhs_{M_1^{\to\diamond\star},f,\sigma}, 1\pi) = \star\ y_k$—for some $\pi \in paths(rhs_{M_1,f,\sigma})$, then:

  $rhs_{\overline{M_1^{\to\diamond\star};M_2^{\diamond\star\to\circ}},\overline{k_f l_g},\sigma}$
  $\quad =$ (by Construction 3.1)
  $nf(\Rightarrow_{R_2^{\diamond\star\to\circ}\cup Pre\cup Pair}, par_{M_2^{\diamond\star\to\circ},rhs_{M_1^{\to\diamond\star},f,\sigma}}(1\pi 1, g, l))$
  $\quad =$ (by definition of $par_{M_2^{\diamond\star\to\circ},rhs_{M_1^{\to\diamond\star},f,\sigma}}$, using that
  $\qquad rhs_{M_2^{\diamond\star\to\circ},g,\star} = g\ v_1\ (\circ\ z_1)\ \cdots\ (\circ\ z_s)$)
  $nf(\Rightarrow_{R_2^{\diamond\star\to\circ}\cup Pre\cup Pair}, \circ\ par_{M_2^{\diamond\star\to\circ},rhs_{M_1^{\to\diamond\star},f,\sigma}}(1\pi, g, l))$
  $\quad =$ (by Lemma 4.9)
  $\circ\ nf(\Rightarrow_{R_2\cup Pre\cup Pair}, par_{M_2,rhs_{M_1,f,\sigma}}(\pi, g, l))$
  $\quad =$ (by Definition 4.5 and Construction 3.1)
  $rhs_{\overline{M_1;M_2}^{\to\circ},\overline{k_f l_g},\sigma}$ □

Using Lemma 4.10 and the fact that $\overline{M_1^{\to\diamond\star};M_2^{\diamond\star\to\circ}}$ was obtained by our **semantics-preserving** composition construction from $M_1^{\to\diamond\star}$ and $M_2^{\diamond\star\to\circ}$, we can express the semantics of $\overline{M_1;M_2}^{\to\circ}$ directly in terms of the semantics of these annotated versions of the original mtts.

Lemma 4.11 ($M_1^{\to\diamond\star}\,;M_2^{\diamond\star\to\circ}$ simulates $\overline{M_1;M_2}^{\to\circ}$)
$\tau_{\overline{M_1;M_2}^{\to\circ}} = \tau_{M_1^{\to\diamond\star}}\,;\tau_{M_2^{\diamond\star\to\circ}}$

Proof
The lemma follows from an instantiation of Theorem 3.2,

$$\tau_{\overline{M_1^{\to\diamond\star};M_2^{\diamond\star\to\circ}}} = \tau_{M_1^{\to\diamond\star}}\,;\tau_{M_2^{\diamond\star\to\circ}} : T_\Sigma \longrightarrow T_{\Omega\cup\{\circ\}}\,,$$

if for every $t \in T_\Sigma$ we can show that

$$\tau_{\overline{M_1^{\to\diamond\star};M_2^{\diamond\star\to\circ}}}(t) = nf(\Rightarrow_{\overline{R_1^{\to\diamond\star};R_2^{\diamond\star\to\circ}}}, e_{\overline{M_1^{\to\diamond\star};M_2^{\diamond\star\to\circ}}}[u_1 \leftarrow t])$$

is equal to

$$\tau_{\overline{M_1;M_2}^{\to\circ}}(t) = nf(\Rightarrow_{\overline{R_1;R_2}^{\to\circ}}, e_{\overline{M_1;M_2}}[u_1 \leftarrow t]).$$

Since the range of $\tau_{\overline{M_1^{\to\diamond\star};M_2^{\diamond\star\to\circ}}}$ contains no trees with nil-symbols, the former normal form can be obtained by applying only rules from $\overline{R_1^{\to\diamond\star};R_2^{\diamond\star\to\circ}}$ whose right-hand side

is not equal to *nil*. (As, in particular, in a call-by-name reduction sequence application of a rule with right-hand side *nil* would lead to an output tree containing *nil*.) But by Lemma 4.10(2) all rules from $\overline{R_1^{\to\diamond\star};R_2^{\diamond\star\to\circ}}$ with non-*nil* right-hand side are identically in $\overline{R_1;R_2}^{\to\circ}$, which implies $e_{\overline{M_1^{\to\diamond\star};M_2^{\diamond\star\to\circ}}}[u_1 \leftarrow t] \Rightarrow^*_{\overline{R_1;R_2}^{\to\circ}} \tau_{\overline{M_1^{\to\diamond\star};M_2^{\diamond\star\to\circ}}}(t)$. Hence, the desired equality follows from Lemma 4.10(1) and the fact that no rules from $\overline{R_1;R_2}^{\to\circ}$ are applicable to an element of $T_{\Omega\cup\{\circ\}}$. □

The previous lemma (in combination with Lemma 4.6) implies that we do not even need to perform Construction 3.1 to analyze the efficiency of a composed program resulting from it. While the externalization by annotation in the previous subsection freed us from further concerns about the specifics of call-by-need reduction, we have thus eliminated any necessity to consider the composition construction itself in the further development.

Example 4.12 (counting steps for composed program in running ex.) The number of $\circ$-symbols produced using the rules in $R^{\to\diamond\star}_{spine}$ and $R^{\diamond\star\to\circ}_{pfx}$ in the example computation in Figure 7 is the same as the number of call-by-need reduction steps required in Figure 4 for the same input. In general we will only have a (pessimistic) approximation of the number of steps of the composed program, but for the running example we actually get an exact measure (cf. the discussion below Lemma 4.6). □

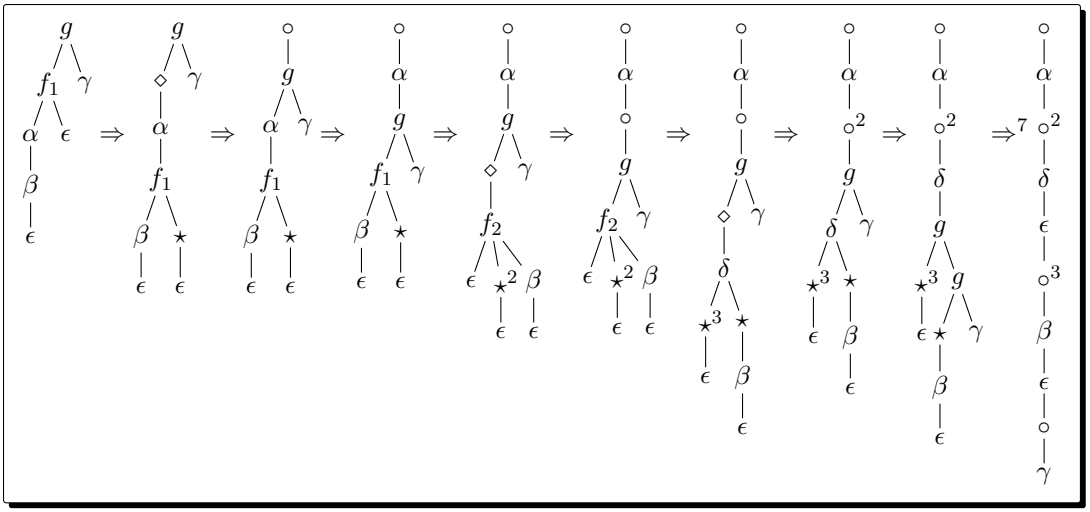

Figure 7: Reduction sequence using $\Rightarrow_{R^{\to\diamond\star}_{spine}\cup R^{\diamond\star\to\circ}_{pfx}}$.

4.4 Combining annotations, and a first efficiency result

Summarizing the results from the previous two subsections, $M_1^{\to\diamond\star};M_2^{\diamond\star\to\circ}$ pessimistically approximates the number of call-by-need steps of the composed program while, for context-linear or basic M_1 and atmost M_2, $M_1^{\to\diamond};M_2^{\diamond\to\bullet}$ counts exactly the call-by-need steps of the original program. In the remainder of the analysis we are seeking sufficient conditions on a pair of mtts M_1 and M_2 under which the

number of $\circ$-symbols in the output of $M_1^{\to\diamond\star}; M_2^{\diamond\star\to\circ}$ is less than or equal to the number of $\bullet$-symbols in the output of $M_1^{\to\diamond}; M_2^{\diamond\to\bullet}$ (plus some small constant) for every input. To compare those numbers it is clearly beneficial to work with only one annotated version for each of M_1 and M_2. Quite naturally, "overlaying" the different annotation schemes leads to the following rules for $f \in F^{(r+1)}$, $\sigma \in \Sigma^{(p)}$, $g \in G^{(s+1)}$, and $\delta \in \Delta^{(q)}$:

$$
\begin{array}{lll}
f\ (\sigma\ u_1 \cdots u_p)\ y_1 \cdots y_r & \to \diamond\ rhs_{M_1,f,\sigma}[y_k \leftarrow \star\ y_k \mid k \in [r]] \\
g\ (\delta\ v_1 \cdots v_q)\ \ z_1 \cdots z_s & \to \bullet\ rhs_{M_2,g,\delta} \\
g\ (\diamond\ v_1) \qquad\quad z_1 \cdots z_s & \to \circ\ (\bullet\ (g\ v_1\ z_1 \cdots z_s)) \\
g\ (\star\ v_1) \qquad\quad z_1 \cdots z_s & \to g\ v_1\ (\circ\ z_1)\ \cdots\ (\circ\ z_s)
\end{array}
$$

Since we are only interested in whether there are more occurrences of one of the symbols $\circ$ and $\bullet$ than of the other, rather than in their absolute numbers, the adjacent $\circ$ and $\bullet$ in the g-rules at $\diamond$ can be omitted. Since the new g-rules at $\diamond$ will simply delete any $\diamond$-symbols appearing in the intermediate result, we can equivalently annotate as follows:

$$
\begin{array}{lll}
f\ (\sigma\ u_1 \cdots u_p)\ y_1 \cdots y_r & \to rhs_{M_1,f,\sigma}[y_k \leftarrow \star\ y_k \mid k \in [r]] \\
g\ (\delta\ v_1 \cdots v_q)\ \ z_1 \cdots z_s & \to \bullet\ rhs_{M_2,g,\delta} \\
g\ (\star\ v_1) \qquad\quad z_1 \cdots z_s & \to g\ v_1\ (\circ\ z_1)\ \cdots\ (\circ\ z_s)
\end{array}
$$

For the later analysis it will be beneficial to keep the δ-rules unchanged from M_2, i.e. devoid of any annotation. To achieve this, we instead of producing a $\bullet$-symbol whenever such a rule is applied arrange for every $\delta \in \Delta$ in the intermediate result to be explicitly marked with a $\bullet$-symbol that is then reproduced separately using appropriate new rules. This leads to the annotated versions of M_1 and M_2 given below. Regarding the notation $\bullet \cdot \delta$, recall that $\bullet$ and δ can be interpreted as functions on (tuples of) trees and that $\cdot$ is standard function composition, i.e., $\bullet \cdot \delta$ for a symbol δ of rank q maps trees $(t_1, \ldots, t_q)$ to $\bullet\ (\delta\ t_1 \cdots t_q)$. Hence, the second-order substitution $\blacksquare[\delta \twoheadleftarrow \bullet \cdot \delta \mid \delta \in \Delta]$ realizes exactly the addition of a $\bullet$-symbol above every symbol from Δ in a tree.

Definition 4.13 ($M_1^{\to\star\bullet}$ and $M_2^{\star\bullet\to\circ\bullet}$)
The mtts $M_1^{\to\star\bullet} = (F, \Sigma, \Delta \cup \{\star, \bullet\}, e_{M_1^{\to\star\bullet}}, R_1^{\to\star\bullet})$ and $M_2^{\star\bullet\to\circ\bullet} = (G, \Delta \cup \{\star, \bullet\}, \Omega \cup \{\circ, \bullet\}, e_{M_2}, R_2^{\star\bullet\to\circ\bullet})$ are given as follows:

$R_1^{\to\star\bullet}$:

$$
f\ (\sigma\ u_1 \cdots u_p)\ y_1 \cdots y_r \to rhs_{M_1,f,\sigma}[\delta \twoheadleftarrow \bullet \cdot \delta \mid \delta \in \Delta]\ [y_k \leftarrow \star\ y_k \mid k \in [r]] \qquad \forall f \in F^{(r+1)}, \sigma \in \Sigma^{(p)}
$$

$$
e_{M_1^{\to\star\bullet}} = e_{M_1}[\delta \twoheadleftarrow \bullet \cdot \delta \mid \delta \in \Delta]
$$

$R_2^{\star\bullet\to\circ\bullet}$:

$$
\begin{array}{lll}
g\ (\delta\ v_1 \cdots v_q)\ \ z_1 \cdots z_s & \to rhs_{M_2,g,\delta} & \forall g \in G^{(s+1)}, \delta \in \Delta^{(q)} \\
g\ (\star\ v_1) \qquad\quad z_1 \cdots z_s & \to g\ v_1\ (\circ\ z_1)\ \cdots\ (\circ\ z_s) & \forall g \in G^{(s+1)} \\
g\ (\bullet\ v_1) \qquad\quad z_1 \cdots z_s & \to \bullet\ (g\ v_1\ z_1 \cdots z_s) & \forall g \in G^{(s+1)}
\end{array}
$$

Note that $M_2^{\star\bullet\to\circ\bullet}$ is context-linear iff M_2 is. $\square$

To show that the output computed by $M_1^{\rightarrow\star\bullet}\,;M_2^{\star\bullet\rightarrow\circ\bullet}$ correctly reflects the difference between the number of $\circ$-symbols produced by $M_1^{\rightarrow\diamond\star}\,;M_2^{\diamond\star\rightarrow\circ}$ on the one hand and the number of $\bullet$-symbols produced by $M_1^{\rightarrow\diamond}\,;M_2^{\diamond\rightarrow\bullet}$ on the other, we first need two auxiliary lemmas. The statements of both lemmas use the second-order substitutions $\,.\,[\star \twoheadleftarrow id]$ and $\,.\,[\diamond \twoheadleftarrow id]$ to express deletion of $\star$- and $\diamond$-symbols, respectively, in a tree. The second lemma is formulated using the function $|\,.\,|_{\circ-\bullet} = |\,.\,|_\circ - |\,.\,|_\bullet$.

Lemma 4.14 (auxiliary for M_1)
For every $t \in T_\Sigma$:

1. $\tau_{M_1^{\rightarrow\diamond}}(t) = \tau_{M_1^{\rightarrow\diamond\star}}(t)[\star \twoheadleftarrow id]$
2. $\tau_{M_1^{\rightarrow\star\bullet}}(t) = \tau_{M_1^{\rightarrow\diamond\star}}(t)[\diamond \twoheadleftarrow id][\delta \twoheadleftarrow \bullet\cdot\delta \mid \delta \in \Delta]$.

Proof
Point 1 is immediate from the facts that $M_1^{\rightarrow\diamond\star}$ and $M_1^{\rightarrow\diamond}$ have the same initial expression containing no $\star$-symbols and that for every $f \in F$ and $\sigma \in \Sigma$, $rhs_{M_1^{\rightarrow\diamond},f,\sigma} = rhs_{M_1^{\rightarrow\diamond\star},f,\sigma}[\star \twoheadleftarrow id]$. Similarly, point 2 follows from the facts that $e_{M_1^{\rightarrow\star\bullet}}$ can be obtained from the initial expression of $M_1^{\rightarrow\diamond\star}$ (e_{M_1}, containing no $\diamond$-symbols) via the substitutions $[\diamond \twoheadleftarrow id][\delta \twoheadleftarrow \bullet\cdot\delta \mid \delta \in \Delta]$ and that for every $f \in F$ and $\sigma \in \Sigma$, $rhs_{M_1^{\rightarrow\star\bullet},f,\sigma} = rhs_{M_1^{\rightarrow\diamond\star},f,\sigma}[\diamond \twoheadleftarrow id][\delta \twoheadleftarrow \bullet\cdot\delta \mid \delta \in \Delta]$. □

Lemma 4.15 (auxiliary for M_2)
For every $t' \in T_{\Delta\cup\{\diamond,\star\}}$:

$$|\tau_{M_2^{\diamond\star\rightarrow\circ}}(t')|_\circ - |\tau_{M_2^{\diamond\rightarrow\bullet}}(t'[\star \twoheadleftarrow id])|_\bullet = |\tau_{M_2^{\star\bullet\rightarrow\circ\bullet}}(t'[\diamond \twoheadleftarrow id][\delta \twoheadleftarrow \bullet\cdot\delta \mid \delta \in \Delta])|_{\circ-\bullet}.$$ □

The proof can be found in the appendix; it holds no big surprises. Plugging the two auxiliary lemmas together, we obtain the following.

Lemma 4.16 ($M_1^{\rightarrow\star\bullet}\,;M_2^{\star\bullet\rightarrow\circ\bullet}$ counts the difference of $\circ$- and $\bullet$-symbols)
$(\tau_{M_1^{\rightarrow\diamond\star}}\,;\tau_{M_2^{\diamond\star\rightarrow\circ}}\,;|\,.\,|_\circ) - (\tau_{M_1^{\rightarrow\diamond}}\,;\tau_{M_2^{\diamond\rightarrow\bullet}}\,;|\,.\,|_\bullet) = \tau_{M_1^{\rightarrow\star\bullet}}\,;\tau_{M_2^{\star\bullet\rightarrow\circ\bullet}}\,;|\,.\,|_{\circ-\bullet}$

Proof
For every $t \in T_\Sigma$ we calculate:

$$\begin{aligned}
&|\tau_{M_2^{\diamond\star\rightarrow\circ}}(\tau_{M_1^{\rightarrow\diamond\star}}(t))|_\circ - |\tau_{M_2^{\diamond\rightarrow\bullet}}(\tau_{M_1^{\rightarrow\diamond}}(t))|_\bullet \\
&\quad = \text{(by Lemma 4.14(1))} \\
&|\tau_{M_2^{\diamond\star\rightarrow\circ}}(\tau_{M_1^{\rightarrow\diamond\star}}(t))|_\circ - |\tau_{M_2^{\diamond\rightarrow\bullet}}(\tau_{M_1^{\rightarrow\diamond\star}}(t)[\star \twoheadleftarrow id])|_\bullet \\
&\quad = \text{(by Lemma 4.15 with } t' = \tau_{M_1^{\rightarrow\diamond\star}}(t)) \\
&|\tau_{M_2^{\star\bullet\rightarrow\circ\bullet}}(\tau_{M_1^{\rightarrow\diamond\star}}(t)[\diamond \twoheadleftarrow id][\delta \twoheadleftarrow \bullet\cdot\delta \mid \delta \in \Delta])|_{\circ-\bullet} \\
&\quad = \text{(by Lemma 4.14(2))} \\
&|\tau_{M_2^{\star\bullet\rightarrow\circ\bullet}}(\tau_{M_1^{\rightarrow\star\bullet}}(t))|_{\circ-\bullet}
\end{aligned}$$ □

Example 4.17 (instantiation of Lemma 4.16 to the running example)
Compare the output computed using the rules in $R_{spine}^{\rightarrow\star\bullet}$ and $R_{pfx}^{\star\bullet\rightarrow\circ\bullet}$ (and the new initial expression $e_{M_{spine}^{\rightarrow\star\bullet}} = f_1\ u_1\ (\bullet\ \epsilon)$, cf. Definition 4.13) in Figure 8 to that from Figures 5 and 7 (for the same input). As before, there is one more $\bullet$-symbol than there are $\circ$-symbols. □

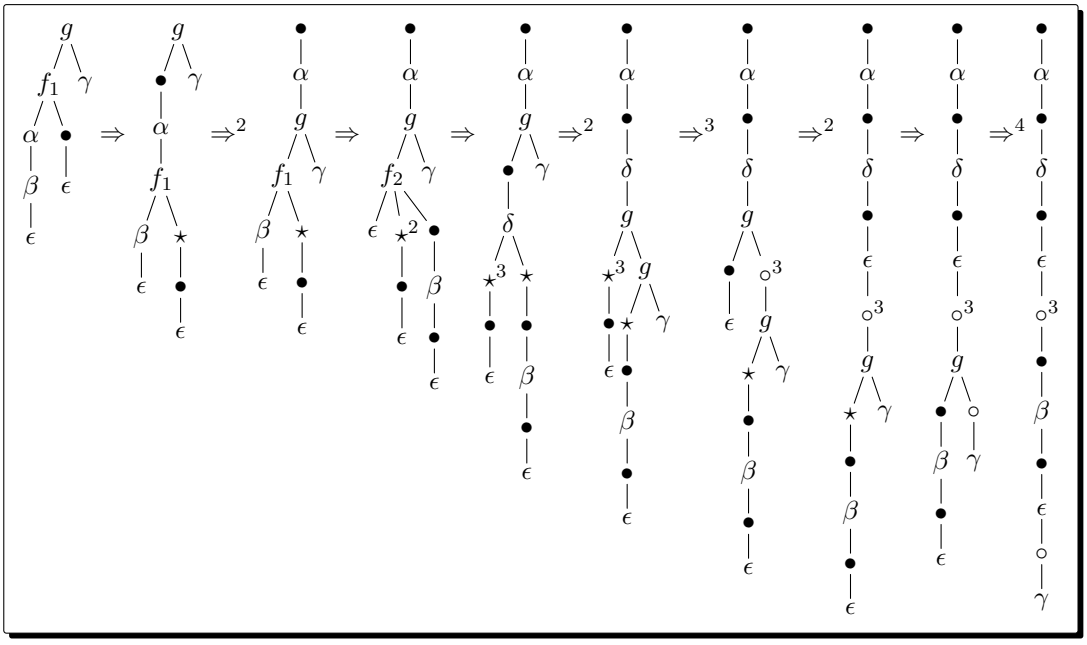

Figure 8: Reduction sequence using $\Rightarrow_{R^{\rightarrow\star\bullet}_{spine} \cup R^{\star\bullet\rightarrow\circ\bullet}_{pfx}}$.

It is easy to see that the output computed by $M_1^{\rightarrow\star\bullet}; M_2^{\star\bullet\rightarrow\circ\bullet}$ will contain no $\circ$-symbols if one of the two mtts involved is a tdtt. This yields a first useful result of our efficiency analysis, albeit for very special cases. It corresponds to Theorem 1 of [Voi02], except for the different formalization of initial expressions in that paper. Since part of the proof will be of further use later, we channel it into an extra lemma first, obtained by a straightforward combination of Lemmas 4.3, 4.6, 4.11, and 4.16.

Lemma 4.18 (intermediate wrap-up)
If M_1 is context-linear or basic and M_2 is atmost, then:

$$cbn_{\overline{M_1;M_2}} - cbn_{M_1;M_2} \leq \tau_{M_1^{\rightarrow\star\bullet}} ; \tau_{M_2^{\star\bullet\rightarrow\circ\bullet}} ; |\,\blacksquare\,|_{\circ-\bullet}.$$ □

Theorem 4.19 (efficiency nondeterioration if M_1 or M_2 is a tdtt)
If M_1 is a tdtt and M_2 is atmost, or if M_1 is context-linear or basic and M_2 is a linear tdtt, then:

$$cbn_{\overline{M_1;M_2}} \leq cbn_{M_1;M_2}.$$

Proof
Lemma 4.18 (using the preconditions on M_1 and M_2) implies that for every $t \in T_\Sigma$:

$$cbn_{\overline{M_1;M_2}}(t) \leq cbn_{M_1;M_2}(t) + |\tau_{M_2^{\star\bullet\rightarrow\circ\bullet}}(\tau_{M_1^{\rightarrow\star\bullet}}(t))|_\circ - |\tau_{M_2^{\star\bullet\rightarrow\circ\bullet}}(\tau_{M_1^{\rightarrow\star\bullet}}(t))|_\bullet.$$

The only possibility for $\tau_{M_2^{\star\bullet\rightarrow\circ\bullet}}(\tau_{M_1^{\rightarrow\star\bullet}}(t))$ to contain $\circ$-symbols is via applications of $\star$-rules of $M_2^{\star\bullet\rightarrow\circ\bullet}$. If M_1 is a tdtt, then $\tau_{M_1^{\rightarrow\star\bullet}}(t)$ contains no $\star$-symbols because there are no context variables in M_1's rule right-hand sides in this case. If M_2 is a tdtt, then the $\star$-rules of $M_2^{\star\bullet\rightarrow\circ\bullet}$ produce no $\circ$-symbols at all because $G = G^{(1)}$ in this case. In both cases we obtain $|\tau_{M_2^{\star\bullet\rightarrow\circ\bullet}}(\tau_{M_1^{\rightarrow\star\bullet}}(t))|_\circ = 0$. Since $|\tau_{M_2^{\star\bullet\rightarrow\circ\bullet}}(\tau_{M_1^{\rightarrow\star\bullet}}(t))|_\bullet$ is always nonnegative, the theorem follows. □

4.5 Unifying the treatment of ∘- and •-symbols

For the general case, i.e. when neither of the two mtts to be composed is a tdtt, we have reduced the analysis problem to comparing—for each input—the numbers of ∘- and •-symbols appearing in the final output produced by $M_1^{\rightarrow\star\bullet}\,;M_2^{\star\bullet\rightarrow\circ\bullet}$. While the emergence of •-symbols by adding one atop every symbol from Δ in the intermediate result and then simply reproducing them using rules

$$g\ (\bullet\ v_1)\ z_1\cdots z_s \rightarrow \bullet\ (g\ v_1\ z_1\cdots z_s)$$

for $g \in G^{(s+1)}$ is already quite handy, the completely different manner in which ∘-symbols are produced, namely through rules of the form

$$g\ (\star\ v_1)\ z_1\cdots z_s \rightarrow g\ v_1\ (\circ\ z_1)\ \cdots\ (\circ\ z_s) \tag{4}$$

from ⋆-symbols in the intermediate result, is somewhat unwieldy. Clearly, it would facilitate the comparison if ∘-symbols were also to appear directly in right-hand sides of rules of the first mtt and to be reproduced by the second mtt using rules analogous to the ones for •. A first step in that direction is to move all ∘-symbols out of the context parameter positions in ⋆-rules as in the following replacement for (4):

$$g\ (\star\ v_1)\ z_1\cdots z_s \rightarrow \underbrace{\circ\ (\cdots\circ}_{s\text{ times}}\ (g\ v_1\ z_1\cdots z_s)\cdots) \tag{5}$$

We have to be careful, though, because this might change the total number of ∘-symbols produced. If, for example, g (called on a concrete instantiation of v_1) deletes one of its context parameters, then the ∘-symbol in that position in the right-hand side of (4) is discarded while the corresponding outwards moved symbol in (5) is unnecessarily counted. Conversely, if g copies one of its context parameters, then a ∘-symbol that is rightfully counted several times when using (4) is counted only once when using (5) instead. While overestimating the number of ∘-symbols and thus the number of call-by-need steps performed by the composed program is acceptable when looking for sufficient (rather than necessary) conditions guaranteeing efficiency nondeterioration through our construction, any underestimation in that respect is of course unsafe. It is avoided by performing the replacement of (4) with (5) only if $M_2^{\star\bullet\rightarrow\circ\bullet}$ is context-linear.

The next step is to directly have sequences of ∘-symbols instead of single ⋆-symbols in the right-hand sides of rules of the first mtt and to reproduce those ∘-symbols one by one with an appropriate rule of the second mtt. Since states in G can have different numbers of context parameters, to be on the safe side we have to replace every ⋆ by s_{max} times ∘, where s_{max} is the maximal such number. This yields for every $f \in F^{(r+1)}$ and $\sigma \in \Sigma^{(p)}$ the following rule:

$$f\ (\sigma\ u_1\cdots u_p)\ y_1\cdots y_r \rightarrow rhs_{M_1,f,\sigma}[\delta \twoheadleftarrow \bullet\cdot\delta \mid \delta\in\Delta][y_k \leftarrow \circ^{s_{max}}\ y_k \mid k\in[r]].$$

In the next subsection our analysis will proceed by finding conditions under which pairs of ∘- and •-symbols appearing in one and the same annotated rule right-hand

side of M_1 can be caused to cancel each other out without compromising safety. To gain flexibility in that endeavor, it would be beneficial if the ∘-symbols in question were not always fixed to appear cascaded above all context variables. Without changing the computed annotated intermediate result (as proved in Lemma 4.22 below), this can be achieved by, e.g., moving some of the s_{max} ∘-symbols from the top of all occurrences of context variable y_k in every right-hand side for f to the kth context parameter position of every call to f throughout right-hand sides and the initial expression. To provide for maximal freedom of choice, the amount of ∘-symbols to be relocated thus can differ for different f and k, as modeled in the following definition through parameterization over the mapping κ. Regarding the second-order substitution $\cdot[f' \leftarrow f' \cdot (id \times \circ^{\kappa_{f',1}} \times \cdots \times \circ^{\kappa_{f',r'}}) \mid f' \in F^{(r'+1)}]$, recall the explanation of $\cdot[\delta \leftarrow \bullet \cdot \delta \mid \delta \in \Delta]$ above Definition 4.13 and the fact that $\times$ is used to denote the Cartesian product of functions, so that $f' \cdot (id \times \circ^{\kappa_{f',1}} \times \cdots \times \circ^{\kappa_{f',r'}})$ for $f' \in F^{(r'+1)}$ maps trees $(t, t_1, \ldots, t_{r'})$ to $f'\ t\ (\circ^{\kappa_{f',1}}\ t_1)\ \cdots\ (\circ^{\kappa_{f',r'}}\ t_{r'})$.

Definition 4.20 ($M_1^{\to\circ\bullet,\kappa}$ and $M_2^{\circ\bullet\to\circ\bullet}$)
Let $s_{max} = max(rank(G)) - 1$ and let $\kappa : \{(f,k) \mid f \in F^{(r+1)}, k \in [r]\} \longrightarrow \{0, \ldots, s_{max}\}$ be arbitrary. For every $f \in F^{(r+1)}$ and $k \in [r]$ abbreviate $\kappa_{f,k} = \kappa(f,k)$ and $\overline{\kappa}_{f,k} = s_{max} - \kappa(f,k)$. The mtts $M_1^{\to\circ\bullet,\kappa} = (F, \Sigma, \Delta \cup \{\circ, \bullet\}, e_{M_1^{\to\circ\bullet,\kappa}}, R_1^{\to\circ\bullet,\kappa})$ and $M_2^{\circ\bullet\to\circ\bullet} = (G, \Delta \cup \{\circ, \bullet\}, \Omega \cup \{\circ, \bullet\}, e_{M_2}, R_2^{\circ\bullet\to\circ\bullet})$ are given as follows:

$R_1^{\to\circ\bullet,\kappa}$:

$$f\ (\sigma\ u_1 \cdots u_p)\ y_1 \cdots y_r \to rhs_{M_1,f,\sigma}\ [\delta \leftarrow \bullet \cdot \delta \mid \delta \in \Delta]\ [f' \leftarrow f' \cdot (id \times \circ^{\kappa_{f',1}} \times \cdots \times \circ^{\kappa_{f',r'}}) \mid f' \in F^{(r'+1)}]\ [y_k \leftarrow \circ^{\overline{\kappa}_{f,k}}\ y_k \mid k \in [r]] \quad \forall f \in F^{(r+1)}, \sigma \in \Sigma^{(p)}$$

$$e_{M_1^{\to\circ\bullet,\kappa}} = e_{M_1}[\delta \leftarrow \bullet \cdot \delta \mid \delta \in \Delta][f' \leftarrow f' \cdot (id \times \circ^{\kappa_{f',1}} \times \cdots \times \circ^{\kappa_{f',r'}}) \mid f' \in F^{(r'+1)}]$$

$R_2^{\circ\bullet\to\circ\bullet}$:

$$g\ (\delta\ v_1 \cdots v_q)\ z_1 \cdots z_s \to rhs_{M_2,g,\delta} \quad \forall g \in G^{(s+1)}, \delta \in \Delta^{(q)}$$

$$g\ (\circ\ v_1)\ z_1 \cdots z_s \to \circ\ (g\ v_1\ z_1 \cdots z_s) \quad \forall g \in G^{(s+1)}$$

$$g\ (\bullet\ v_1)\ z_1 \cdots z_s \to \bullet\ (g\ v_1\ z_1 \cdots z_s) \quad \forall g \in G^{(s+1)}$$

Note that $M_2^{\circ\bullet\to\circ\bullet}$ is recursion-linear iff M_2 is; likewise for recursion-nondeleting, context-linear and -nondeleting, basic, atmost, and atleast. □

Example 4.21 ($M_{spine}^{\to\circ\bullet,\kappa}$ for $s_{max} = 1$ and a concrete κ)
Choose κ so that $\kappa_{f_1,1} = \kappa_{f_2,2} = 1$ and $\kappa_{f_2,1} = 0$. Then, the mtt $M_{spine}^{\to\circ\bullet,\kappa} = (\{f_1^{(2)}, f_2^{(3)}\}, \Sigma_{mon}, \Delta_{tree} \cup \{\circ, \bullet\}, e_{M_{spine}^{\to\circ\bullet,\kappa}}, R_{spine}^{\to\circ\bullet,\kappa})$ has the rules

$$\begin{aligned} R_{spine}^{\to\circ\bullet,\kappa}\colon\ & f_1\ (\alpha\ u_1)\ y_1 \to \bullet\ (\alpha\ (f_1\ u_1\ (\circ\ y_1))) \\ & f_1\ (\beta\ u_1)\ y_1 \to f_2\ u_1\ y_1\ (\circ\ (\bullet\ (\beta\ (\bullet\ \epsilon)))) \\ & f_1\ \epsilon\ y_1 \to y_1 \\ & f_2\ (\alpha\ u_1)\ y_1\ y_2 \to \bullet\ (\alpha\ (f_1\ u_1\ (\circ\ (\bullet\ (\delta\ (\circ\ y_1)\ y_2))))) \\ & f_2\ (\beta\ u_1)\ y_1\ y_2 \to f_2\ u_1\ (\circ\ y_1)\ (\circ\ (\bullet\ (\beta\ (\bullet\ (\beta\ y_2))))) \\ & f_2\ \epsilon\ y_1\ y_2 \to \bullet\ (\delta\ (\circ\ y_1)\ y_2) \end{aligned}$$

and the initial expression $e_{M_{spine}^{\to\circ\bullet,\kappa}} = f_1\ u_1\ (\circ\ (\bullet\ \epsilon))$. □

To show that—irrespective of the choice of κ—the difference between the numbers of $\circ$- and $\bullet$-symbols in the output computed by $M_1^{\to\circ\bullet,\kappa}\,;M_2^{\circ\bullet\to\circ\bullet}$ for context-linear M_2 is greater than or equal to the corresponding difference in the output produced by $M_1^{\to\star\bullet}\,;M_2^{\star\bullet\to\circ\bullet}$, we first need two auxiliary lemmas.

Lemma 4.22 (auxiliary for M_1)
For every $\kappa : \{(f,k) \mid f \in F^{(r+1)}, k \in [r]\} \longrightarrow \{0,\ldots,s_{max}\}$ and $t \in T_\Sigma$:

$$\tau_{M_1^{\to\circ\bullet,\kappa}}(t) = \tau_{M_1^{\to\star\bullet}}(t)[\star \leftarrow \circ^{s_{max}}].$$ □

The proof can be found in the appendix. The key observation is that when a call $f\ t\ (\circ^{\kappa_{f,1}}\ y_1)\ \cdots\ (\circ^{\kappa_{f,r}}\ y_r)$ is reduced using a rule from $R_1^{\to\circ\bullet,\kappa}$, additional $\circ$-symbols are put on top of each context parameter by applying $\circ^{\overline{\kappa}_{f,k}}$ to $(\circ^{\kappa_{f,k}}\ y_k)$ for every $k \in [r]$. Since $\overline{\kappa}_{f,k} + \kappa_{f,k} = s_{max}$ for every $k \in [r]$, this corresponds to the $\circ^{s_{max}}$ substituted eventually for the single $\star$-symbol put onto each context parameter when reducing a call to f on t using a rule from $R_1^{\to\star\bullet}$.

Lemma 4.23 (auxiliary for M_2)
If M_2 is context-linear, then for every $t' \in T_{\Delta\cup\{\star,\bullet\}}$:

$$|\tau_{M_2^{\star\bullet\to\circ\bullet}}(t')|_{\circ-\bullet} \leq |\tau_{M_2^{\circ\bullet\to\circ\bullet}}(t'[\star \leftarrow \circ^{s_{max}}])|_{\circ-\bullet}.$$ □

The proof can be found in the appendix. The key idea is to quantify how often the $\circ$-symbols produced in a step $g\ (\star\ t_1')\ z_1 \cdots z_s \Rightarrow_{R_2^{\star\bullet\to\circ\bullet}} g\ t_1'\ (\circ\ z_1)\ \cdots\ (\circ\ z_s)$ will appear in the final output in terms of the number of occurrences of $z_1,\ldots,z_s$ in $nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, g\ t_1'\ z_1 \cdots z_s)$—which can be approximated using Lemma 2.5(1)—and to compare this to the number of $\circ$-symbols produced when reducing a call to g on $(\star\ t_1')[\star \leftarrow \circ^{s_{max}}]$ using (s_{max} times) the rule for g at $\circ$ in $R_2^{\circ\bullet\to\circ\bullet}$.

Plugging the two auxiliary lemmas together, we obtain the following.

Lemma 4.24 ($M_1^{\to\circ\bullet,\kappa}\,;M_2^{\circ\bullet\to\circ\bullet}$ computes a safe approximation)
If M_2 is context-linear, then for every $\kappa : \{(f,k) \mid f \in F^{(r+1)}, k \in [r]\} \longrightarrow \{0,\ldots,s_{max}\}$:

$$\tau_{M_1^{\to\star\bullet}}\,;\tau_{M_2^{\star\bullet\to\circ\bullet}}\,;|\,\cdot\,|_{\circ-\bullet} \leq \tau_{M_1^{\to\circ\bullet,\kappa}}\,;\tau_{M_2^{\circ\bullet\to\circ\bullet}}\,;|\,\cdot\,|_{\circ-\bullet}.$$

Proof
For every $t \in T_\Sigma$, $|\tau_{M_2^{\star\bullet\to\circ\bullet}}(\tau_{M_1^{\to\star\bullet}}(t))|_{\circ-\bullet} \leq |\tau_{M_2^{\circ\bullet\to\circ\bullet}}(\tau_{M_1^{\to\circ\bullet,\kappa}}(t))|_{\circ-\bullet}$ by Lemma 4.23 with $t' = \tau_{M_1^{\to\star\bullet}}(t)$ and Lemma 4.22. □

Example 4.25 (instantiation of Lemma 4.24 to the running example)
To observe the effect of the transition from $M_1^{\to\star\bullet}$ and $M_2^{\star\bullet\to\circ\bullet}$ to $M_1^{\to\circ\bullet,\kappa}$ and $M_2^{\circ\bullet\to\circ\bullet}$ for the running example with the concrete κ from Example 4.21, compare the computation in Figure 9 to that from Figure 8. In particular, note how as a consequence of replacing (4) with (5), $\circ$-symbols have "floated upwards" in the computed output, as indicated by the dashed arrows. □

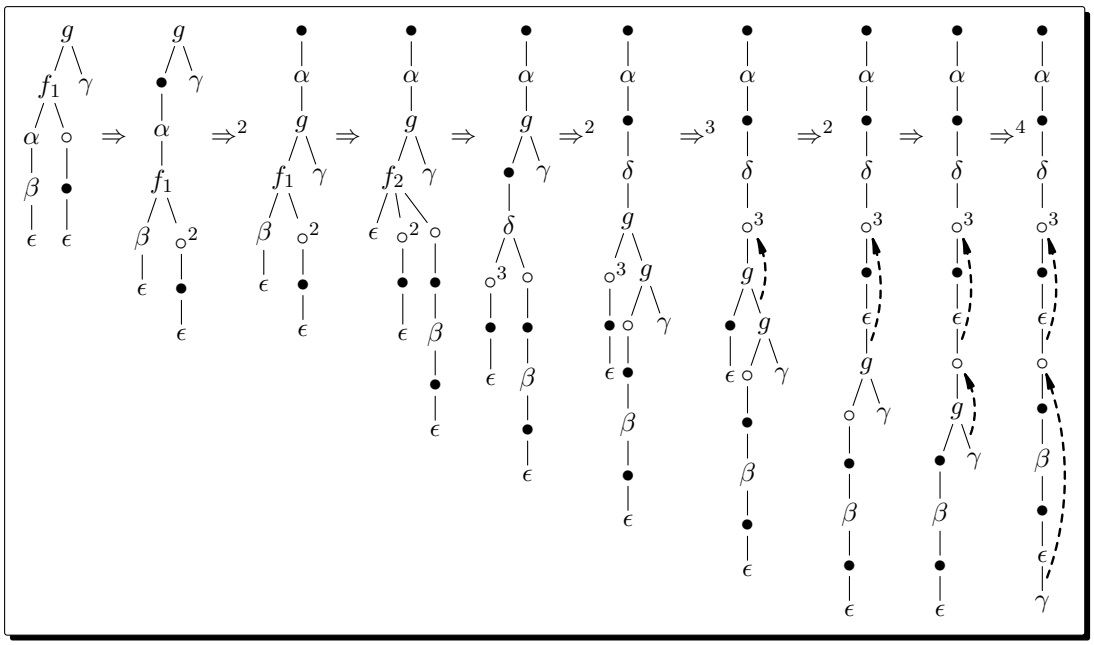

Figure 9: Reduction sequence using $\Rightarrow_{R_{spine}^{\rightarrow\circ\bullet,\kappa} \cup R_{pfx}^{\circ\bullet\rightarrow\circ\bullet}}$.

4.6 Moving •-symbols to cancel out ∘-symbols

Our ultimate goal is now to put a constant upper bound, for every input, on the difference between the numbers of ∘- and •-symbols reproduced by $M_2^{\circ\bullet\rightarrow\circ\bullet}$ from an intermediate result produced by some $M_1^{\rightarrow\circ\bullet,\kappa}$. That could in principle be achieved by correspondingly bounding only the number of ∘-symbols because the subtracted number of •-symbols is always nonnegative. Clearly, it is unlikely that there is indeed a constant upper bound for the number of ∘-symbols because the number of applications of rules of $M_1^{\rightarrow\circ\bullet,\kappa}$ in whose right-hand sides such symbols appear depends on the input, and so consequently does the number of ∘-symbols in the final output. We would stand much better chances if all ∘-symbols were eliminated from the rules of the mtt producing the annotated intermediate result. Fortunately, we can make this happen by continuing to manipulate the first mtt's annotation, as long as this does not lead to an underestimation of the difference between the numbers of ∘- and •-symbols. One obviously safe manipulation is to once again remove pairs of adjacent ∘- and •-symbols. If such symbols do not appear next to each other, we can attempt to first bring them closer together by moving either kind of symbols. The following definition lists the sensible choices for moving •-symbols around in annotated right-hand sides of rules of the first mtt. It turned out that dually moving also ∘-symbols does not add strength to the analysis.

Definition 4.26 (symmetric up- and down-rules for •)
For every $f \in F^{(r+1)}$ and $k \in [r]$ rewrite rules $\downarrow_{f,k}$ and $\uparrow_{f,k}$, and for every $\delta \in \Delta^{(q)}$ and $j \in [q]$ rewrite rules $\downarrow_{\delta,j}$ and $\uparrow_{\delta,j}$ are given as follows:

$$\begin{array}{lll}
\downarrow_{f,k} & : \bullet\ (f\ u\ v_1 \cdots v_r) & \to f\ u\ v_1 \cdots (\bullet\ v_k) \cdots v_r \\
\uparrow_{f,k} & : f\ u\ v_1 \cdots (\bullet\ v_k) \cdots v_r & \to \bullet\ (f\ u\ v_1 \cdots v_r) \\
\downarrow_{\delta,j} & : \bullet\ (\delta\ v_1 \cdots v_q) & \to \delta\ v_1 \cdots (\bullet\ v_j) \cdots v_q \\
\uparrow_{\delta,j} & : \delta\ v_1 \cdots (\bullet\ v_j) \cdots v_q & \to \bullet\ (\delta\ v_1 \cdots v_q)
\end{array}$$

□

Such ↑- and ↓-rules cannot be applied unconstrained. If, for example, $M_2^{\circ\bullet\to\circ\bullet}$ is not recursion-linear, then the application of a $\downarrow_{\delta,j}$-rule to move a •-symbol below an output symbol in the right-hand side of some rule of the first mtt—and as a consequence also in the annotated intermediate result—can cause that •-symbol to be counted more often after the manipulation than it was previously, in case the jth recursion variable occurs more than once in some δ-rule of M_2. That is, in the presence of a rule

$$g\ (\delta\ v_1\ v_2) \to \omega\ (g\ v_1)\ (g\ v_1)$$

of M_2 an application of the $\downarrow_{\delta,1}$-rule can have the effect that an $\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}$-reduction

$$g\ (\bullet\ (\delta\ \theta_1\ \theta_2)) \Rightarrow \bullet\ (g\ (\delta\ \theta_1\ \theta_2)) \Rightarrow \bullet\ (\omega\ (g\ \theta_1)\ (g\ \theta_1))$$

for some $\theta_1, \theta_2 \in T_{\Delta\cup\{\circ,\bullet\}}$ is replaced by the following:

$$g\ (\delta\ (\bullet\ \theta_1)\ \theta_2) \Rightarrow \omega\ (g\ (\bullet\ \theta_1))\ (g\ (\bullet\ \theta_1)) \Rightarrow^2 \omega\ (\bullet\ (g\ \theta_1))\ (\bullet\ (g\ \theta_1)),$$

thus overestimating the number of •-symbols. In addition to avoiding unlegitimate duplication of a •-symbol through several recursive calls on the subtree into which it was moved, also more subtle ways of duplication via context parameters must be barred. If, for example, M_2 has a rule

$$g\ (\delta\ v_1\ v_2) \to g'\ v_1\ (g\ v_2) \tag{6}$$

then an application of the $\downarrow_{\delta,2}$-rule can have the effect that an $\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}$-reduction

$$g\ (\bullet\ (\delta\ \theta_1\ \theta_2)) \Rightarrow \bullet\ (g\ (\delta\ \theta_1\ \theta_2)) \Rightarrow \bullet\ (g'\ \theta_1\ (g\ \theta_2))$$

for some $\theta_1, \theta_2 \in T_{\Delta\cup\{\circ,\bullet\}}$ is replaced by the following:

$$g\ (\delta\ \theta_1\ (\bullet\ \theta_2)) \Rightarrow g'\ \theta_1\ (g\ (\bullet\ \theta_2)) \Rightarrow g'\ \theta_1\ (\bullet\ (g\ \theta_2)).$$

If g' happens to copy its context parameter, then one •-symbol is turned into more than one. To avoid this problem, we can either outlaw nested calls such as the one in the right-hand side of (6), or require that context parameters are never copied. Syntactic restrictions of mtts to these effects are (either of) basicness and context-linearity from Definition 2.4; the proper combination with recursion-linearity to guarantee a safe application of $\downarrow_{\delta,j}$-rules is given by what we dubbed **atmost** there. Completely dually, M_2 must be **atleast** in order to apply $\uparrow_{\delta,j}$-rules safely. In the case of $\downarrow_{f,k}$- and $\uparrow_{f,k}$-rules for a state $f \in F$, we additionally need to require context-linearity and -nondeletingness, respectively, of the first mtt to prevent unlegitimate proliferation of •-symbols in the annotated intermediate result already. The following definition captures the gained insights by specifying a permissible subset $\mathcal{E}_{M_1,M_2}$ of the possible rewrite rules for moving •-symbols and eliminating ∘-symbols.

Definition 4.27 ($\mathcal{E}_{M_1,M_2}$)
The rewrite system $\mathcal{E}_{M_1,M_2}$ contains the following rules:

- $\bullet\ (\circ\ v) \to v$,
- $\circ\ (\bullet\ v) \to v$,
- $\downarrow_{f,k}$ for every $f \in F^{(r+1)}$ and $k \in [r]$ if M_1 is context-linear and M_2 is atmost,
- $\uparrow_{f,k}$ for every $f \in F^{(r+1)}$ and $k \in [r]$ if M_1 is context-nondeleting and M_2 is atleast,
- $\downarrow_{\delta,j}$ for every $\delta \in \Delta^{(q)}$ and $j \in [q]$ if M_2 is atmost, and
- $\uparrow_{\delta,j}$ for every $\delta \in \Delta^{(q)}$ and $j \in [q]$ if M_2 is atleast. □

Example 4.28 ($\mathcal{E}_{M_{spine},M_{pfx}}$, cf. Example 2.2)
Since M_{spine} is context-linear and context-nondeleting and M_{pfx} is atmost and atleast (even linear and nondeleting), the rewrite system $\mathcal{E}_{M_{spine},M_{pfx}}$ contains all potential $\downarrow$- and $\uparrow$-rules. □

We will later allow the rewrite rules from $\mathcal{E}_{M_1,M_2}$ to be applied arbitrarily often to rule right-hand sides, but also to the initial expression, of annotated versions of M_1. For the sake of proving that this indeed does not lead to an underestimation of the difference between the numbers of ∘- and •-symbols in the final output, however, it is more convenient to first consider at most one rewrite step on the initial expression and each right-hand side. This is described by a relation $\rightsquigarrow_{\mathcal{E}_{M_1,M_2}}$ on a set containing all $M_1^{\to\circ\bullet,\kappa}$ and all mtts possibly obtained from them via the rewrite rules in $\mathcal{E}_{M_1,M_2}$.

Definition 4.29 ($\mathcal{M}_1^{\to\circ\bullet}$ and $\rightsquigarrow_{\mathcal{E}_{M_1,M_2}} \subseteq \mathcal{M}_1^{\to\circ\bullet} \times \mathcal{M}_1^{\to\circ\bullet}$)
Let $\mathcal{M}_1^{\to\circ\bullet}$ be the set of all mtts $M = (F, \Sigma, \Delta \cup \{\circ, \bullet\}, e, R)$ for which $e[\circ, \bullet \twoheadleftarrow id, id] = e_{M_1}$ and for every $f \in F$ and $\sigma \in \Sigma$, $rhs_{M,f,\sigma}[\circ, \bullet \twoheadleftarrow id, id] = rhs_{M_1,f,\sigma}$. Note that every $M_1^{\to\circ\bullet} \in \mathcal{M}_1^{\to\circ\bullet}$ is context-linear iff M_1 is; likewise for context-nondeleting. For $M_1^{\to\circ\bullet} = (F, \Sigma, \Delta \cup \{\circ, \bullet\}, e_{M_1^{\to\circ\bullet}}, R_1^{\to\circ\bullet})$ and $M_1^{\to\circ\bullet'} = (F, \Sigma, \Delta \cup \{\circ, \bullet\}, e_{M_1^{\to\circ\bullet'}}, R_1^{\to\circ\bullet'})$ in $\mathcal{M}_1^{\to\circ\bullet}$ we write $M_1^{\to\circ\bullet} \rightsquigarrow_{\mathcal{E}_{M_1,M_2}} M_1^{\to\circ\bullet'}$ if $e_{M_1^{\to\circ\bullet}} \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} e_{M_1^{\to\circ\bullet'}}$ and for every $f \in F$ and $\sigma \in \Sigma$, $rhs_{M_1^{\to\circ\bullet},f,\sigma} \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} rhs_{M_1^{\to\circ\bullet'},f,\sigma}$. □

Example 4.30 (rewriting $M_{spine}^{\to\circ\bullet,\kappa}$ from Example 4.21 with $\rightsquigarrow_{\mathcal{E}_{M_{spine},M_{pfx}}}$)
By rewriting the initial expression and each right-hand side of a rule of $M_{spine}^{\to\circ\bullet,\kappa}$ with at most one step of $\Rightarrow_{\mathcal{E}_{M_{spine},M_{pfx}}}$, we obtain $M_{spine}^{\to\circ\bullet,\kappa} \rightsquigarrow_{\mathcal{E}_{M_{spine},M_{pfx}}} M_{spine}^{\to\circ\bullet'} = (\{f_1^{(2)}, f_2^{(3)}\}, \Sigma_{mon}, \Delta_{tree} \cup \{\circ, \bullet\}, f_1\ u_1\ (\circ\ (\bullet\ \epsilon)), R_{spine}^{\to\circ\bullet'})$ with rules

$$
\begin{array}{lll}
R_{spine}^{\to\circ\bullet'}\colon & f_1\ (\alpha\ u_1)\ y_1 & \to \bullet\ (\alpha\ (f_1\ u_1\ (\circ\ y_1))) \\
& f_1\ (\beta\ u_1)\ y_1 & \to f_2\ u_1\ y_1\ (\beta\ (\bullet\ \epsilon)) \\
& f_1\ \epsilon\ \ \ \ y_1 & \to y_1 \\
& f_2\ (\alpha\ u_1)\ y_1\ y_2 & \to \bullet\ (\alpha\ (f_1\ u_1\ (\circ\ (\delta\ (\bullet\ (\circ\ y_1))\ y_2)))) \\
& f_2\ (\beta\ u_1)\ y_1\ y_2 & \to f_2\ u_1\ (\circ\ y_1)\ (\circ\ (\bullet\ (\beta\ (\bullet\ (\beta\ y_2))))) \\
& f_2\ \epsilon\ \ \ \ y_1\ y_2 & \to \delta\ (\bullet\ (\circ\ y_1))\ y_2
\end{array}
$$

□

To show that rewriting an annotated version of M_1 according to $\rightsquigarrow_{\mathcal{E}_{M_1,M_2}}$ does not lead to a decrease in the difference between the numbers of $\circ$- and $\bullet$-symbols reproduced by $M_2^{\circ\bullet\to\circ\bullet}$ from the annotated intermediate result, we first prove two lemmas. The first one captures how the omission of $\bullet$-symbols from an annotated intermediate result is reflected in the final output produced by a state of an atmost and of an atleast M_2, respectively. In fact, the properties given here motivated the choice of names "atmost" and "atleast". Note that the slightly counter-intuitive difference in the orders in which θ and θ' appear in the left- and right-hand sides of the given inequations is caused by the fact that $\bullet$-symbols are counted negatively by $|\,.\,|_{\circ-\bullet}$ but positively by $|\,.\,|_\bullet$.

Lemma 4.31 (essential properties of atmost and atleast mtts, resp.)
For every $\theta, \theta' \in T_{\Delta\cup\{\circ,\bullet\}}$ with $\theta \Rightarrow^*_{\{\bullet\, v\to v\}} \theta'$ and every $g \in G^{(s+1)}$:

1. if M_2 is atmost, then:
   $|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta'\ z_1 \cdots z_s)|_{\circ-\bullet} - |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta\ z_1 \cdots z_s)|_{\circ-\bullet} \leq |\theta|_\bullet - |\theta'|_\bullet$
2. if M_2 is atleast, then:
   $|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta'\ z_1 \cdots z_s)|_{\circ-\bullet} - |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta\ z_1 \cdots z_s)|_{\circ-\bullet} \geq |\theta|_\bullet - |\theta'|_\bullet$. □

The proof (using Lemma 2.5) can be found in the appendix. It uses $|\,.\,|_{\bullet-\circ} = |\,.\,|_\bullet - |\,.\,|_\circ$ instead of $|\,.\,|_{\circ-\bullet}$ to later enable a "recycling" of the performed inductions for the proof of Lemma 4.36.

Lemma 4.31 is used in the following lemma to establish in which sense the $\uparrow$- and $\downarrow$-rules we have chosen to put into $\mathcal{E}_{M_1,M_2}$ are permissible.

Lemma 4.32 (essential properties of the rewrite system $\mathcal{E}_{M_1,M_2}$)
For every $g \in G^{(s+1)}$:

1. for every $\delta \in \Delta^{(q)}$, $j \in [q]$, and $\theta_1, \ldots, \theta_q \in T_{\Delta\cup\{\circ,\bullet\}}$:
   (a) if $\downarrow_{\delta,j} \in \mathcal{E}_{M_1,M_2}$, then:
   $|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ (\delta\ \theta_1 \cdots \theta_q)\ z_1 \cdots z_s)|_{\circ-\bullet}$
   $\leq 1 + |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ (\delta\ \theta_1 \cdots (\bullet\ \theta_j) \cdots \theta_q)\ z_1 \cdots z_s)|_{\circ-\bullet}$
   (b) if $\uparrow_{\delta,j} \in \mathcal{E}_{M_1,M_2}$, then:
   $|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ (\delta\ \theta_1 \cdots \theta_q)\ z_1 \cdots z_s)|_{\circ-\bullet}$
   $\geq 1 + |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ (\delta\ \theta_1 \cdots (\bullet\ \theta_j) \cdots \theta_q)\ z_1 \cdots z_s)|_{\circ-\bullet}$
2. for every $M_1^{\to\circ\bullet} = (F, \Sigma, \Delta\cup\{\circ,\bullet\}, e_{M_1^{\to\circ\bullet}}, R_1^{\to\circ\bullet}) \in \mathcal{M}_1^{\to\circ\bullet}$, $f \in F^{(r+1)}$, $k \in [r]$, $t \in T_\Sigma$, and $\theta_1, \ldots, \theta_r \in T_{\Delta\cup\{\circ,\bullet\}}$:
   (a) if $\downarrow_{f,k} \in \mathcal{E}_{M_1,M_2}$, then:
   $|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet}}, f\ t\ \theta_1 \cdots \theta_r)\ z_1 \cdots z_s)|_{\circ-\bullet}$
   $\leq 1 + |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet}}, f\ t\ \theta_1 \cdots (\bullet\ \theta_k) \cdots \theta_r)\ z_1 \cdots z_s)|_{\circ-\bullet}$
   (b) if $\uparrow_{f,k} \in \mathcal{E}_{M_1,M_2}$, then:
   $|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet}}, f\ t\ \theta_1 \cdots \theta_r)\ z_1 \cdots z_s)|_{\circ-\bullet}$
   $\geq 1 + |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet}}, f\ t\ \theta_1 \cdots (\bullet\ \theta_k) \cdots \theta_r)\ z_1 \cdots z_s)|_{\circ-\bullet}$.

Proof
Since from $\downarrow_{\delta,j} \in \mathcal{E}_{M_1,M_2}$ follows that M_2 is atmost, and from $\uparrow_{\delta,j} \in \mathcal{E}_{M_1,M_2}$ follows that M_2 is atleast, the points 1a and 1b follow from Lemma 4.31 with $\theta = \delta\ \theta_1 \cdots (\bullet\ \theta_j) \cdots \theta_q$ and $\theta' = \delta\ \theta_1 \cdots \theta_q$, using that $\theta \Rightarrow_{\{\bullet\ v \to v\}} \theta'$ and $|\theta|_\bullet - |\theta'|_\bullet = 1$.

Since from $\downarrow_{f,k} \in \mathcal{E}_{M_1,M_2}$ follows that M_1 is context-linear and M_2 is atmost, and from $\uparrow_{f,k} \in \mathcal{E}_{M_1,M_2}$ follows that M_1 is context-nondeleting and M_2 is atleast, the points 2a and 2b follow from Lemma 4.31 with $\theta = nf(\Rightarrow_{R_1^{\to\circ\bullet}}, f\ t\ \theta_1 \cdots (\bullet\ \theta_k) \cdots \theta_r)$ and $\theta' = nf(\Rightarrow_{R_1^{\to\circ\bullet}}, f\ t\ \theta_1 \cdots \theta_r)$, using three auxiliary statements:

1. $\theta \Rightarrow^*_{\{\bullet\ v \to v\}} \theta'$,

2. $|\theta|_\bullet - |\theta'|_\bullet \leq 1$ if M_1, and hence also $M_1^{\to\circ\bullet}$, is context-linear, and

3. $|\theta|_\bullet - |\theta'|_\bullet \geq 1$ if M_1, and hence also $M_1^{\to\circ\bullet}$, is context-nondeleting.

These statements, in turn, follow immediately from the following equalities for θ and θ' (which hold due to Lemma 2.6 for the mtt $M_1^{\to\circ\bullet}$):

$$\begin{aligned}\theta &= nf(\Rightarrow_{R_1^{\to\circ\bullet}}, f\ t\ y_1 \cdots y_r)[y_1, \ldots, y_r \leftarrow \theta_1, \ldots, \bullet\ \theta_k, \ldots, \theta_r]\\ \theta' &= nf(\Rightarrow_{R_1^{\to\circ\bullet}}, f\ t\ y_1 \cdots y_r)[y_1, \ldots, y_r \leftarrow \theta_1, \ldots, \theta_r]\end{aligned}$$

and Lemma 2.5 for the mtt $M_1^{\to\circ\bullet}$. □

Now, we are in a position to prove that rewriting according to $\rightsquigarrow_{\mathcal{E}_{M_1,M_2}}$ does not lead to an unlegitimate underestimation of the number of ∘-symbols or overestimation of the number of •-symbols.

Lemma 4.33 (applying $\rightsquigarrow_{\mathcal{E}_{M_1,M_2}}$ leads to a safe approximation)
Let $M_1^{\to\circ\bullet}, M_1^{\to\circ\bullet'} \in \mathcal{M}_1^{\to\circ\bullet}$. If $M_1^{\to\circ\bullet} \rightsquigarrow_{\mathcal{E}_{M_1,M_2}} M_1^{\to\circ\bullet'}$, then:

$$\tau_{M_1^{\to\circ\bullet}}\ ;\tau_{M_2^{\circ\bullet\to\circ\bullet}}\ ;|\,\blacksquare\,|_{\circ-\bullet} \leq \tau_{M_1^{\to\circ\bullet'}}\ ;\tau_{M_2^{\circ\bullet\to\circ\bullet}}\ ;|\,\blacksquare\,|_{\circ-\bullet}.$$ □

The proof (using Lemma 4.32) can be found in the appendix. At its heart is an analysis of the different ways in which a rule right-hand side or initial expression of $M_1^{\to\circ\bullet}$ can be rewritten (or not) using $\Rightarrow^?_{\mathcal{E}_{M_1,M_2}}$. This leads to eleven cases to consider, but a lot of reuse is possible in resolving the associated proof obligations.

Example 4.34 (instantiation of Lemma 4.33 to the running example)
To observe the effect of the concrete application of $\rightsquigarrow_{\mathcal{E}_{M_{spine},M_{pfx}}}$ in Example 4.30, compare the computation in Figure 10 to that from Figure 9. In particular, note how as a consequence of rewriting the right-hand side of the rule for f_2 at ϵ, one •-symbol has "floated downwards" in the computed output, as indicated by the dashed arrow, and how the removal of adjacent ∘- and •-symbols in the right-hand side of the rule for f_1 at β has led to the elimination of one such pair also from the computed output at the ×-marked position. □

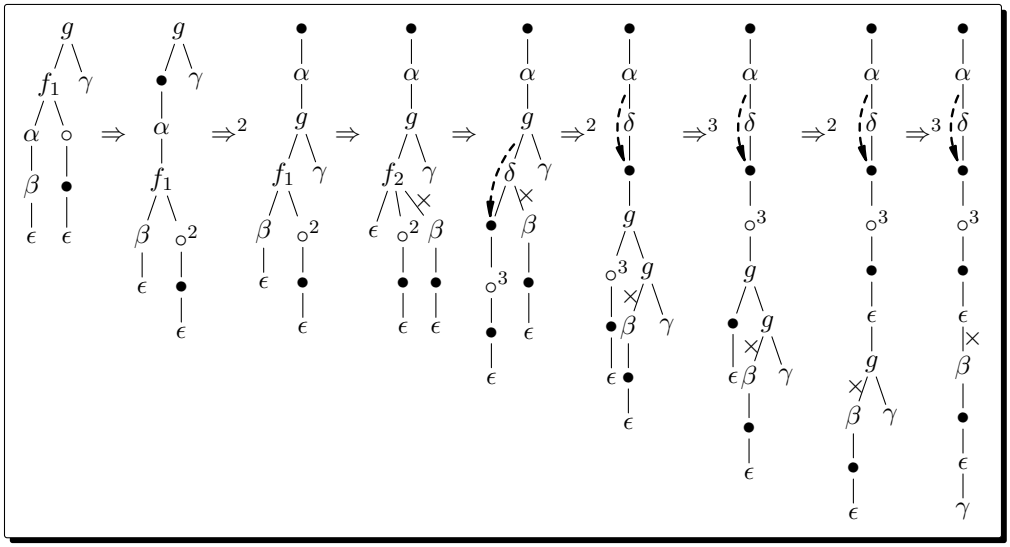

Figure 10: Reduction sequence using $\Rightarrow_{R^{\rightarrow\circ\bullet\prime}_{spine} \cup R^{\circ\bullet\rightarrow\circ\bullet}_{pfx}}$.

4.7 Bounding the number of ∘-symbols

As motivated at the beginning of the previous subsection, we want to manipulate some $M_1^{\rightarrow\circ\bullet,\kappa}$ (by eliminating ∘- and •-symbols via $\rightsquigarrow_{\varepsilon_{M_1,M_2}}$) until we obtain an $M_1^{\rightarrow\circ\bullet} \in \mathcal{M}_1^{\rightarrow\circ\bullet}$ for which we can put a constant upper bound on $|\tau_{M_2^{\circ\bullet\rightarrow\circ\bullet}}(\tau_{M_1^{\rightarrow\circ\bullet}}(t))|_\circ$ for every $t \in T_\Sigma$. The most natural approach to establishing such an upper bound is to first consider conditions under which the number of ∘-symbols in the intermediate result $\tau_{M_1^{\rightarrow\circ\bullet}}(t)$ is bounded. Requiring that no ∘-symbols occur anymore in right-hand sides of rules of $M_1^{\rightarrow\circ\bullet}$ is not quite enough because there might still be ∘-symbols present in its initial expression. Hence, we additionally impose context-linearity to be sure that such ∘-symbols are not duplicated during the computation. The following lemma capturing this simple idea was proved in [Voi04a] (using Lemma 2.5 rather than the actual syntactic definition of context-linearity); here it is stated without explicit proof.

Lemma 4.35 (bounding the number of ∘-symbols produced by $M_1^{\rightarrow\circ\bullet}$)
Let $M_1^{\rightarrow\circ\bullet} = (F, \Sigma, \Delta \cup \{\circ, \bullet\}, e_{M_1^{\rightarrow\circ\bullet}}, R_1^{\rightarrow\circ\bullet}) \in \mathcal{M}_1^{\rightarrow\circ\bullet}$. If the right-hand sides of the rules in $R_1^{\rightarrow\circ\bullet}$ contain no ∘-symbols and M_1 is context-linear, then for every $t \in T_\Sigma$:

$$|\tau_{M_1^{\rightarrow\circ\bullet}}(t)|_\circ \leq |e_{M_1^{\rightarrow\circ\bullet}}|_\circ. \qquad \square$$

Given a constant bound for the number of ∘-symbols in $\tau_{M_1^{\rightarrow\circ\bullet}}(t)$, we could obtain a corresponding bound for $\tau_{M_2^{\circ\bullet\rightarrow\circ\bullet}}(\tau_{M_1^{\rightarrow\circ\bullet}}(t))$ if we knew that $M_2^{\circ\bullet\rightarrow\circ\bullet}$ duplicates ∘-symbols found in its input by at most a constant factor. One reason for unbounded reproduction could be recursion-nonlinearity because then a given ∘-symbol could be reached arbitrarily many times, depending on its depth in the input tree. Another source for unbounded duplication could be that the output of a call that is nested inside a context parameter position of another call is duplicated when the outer

call copies its parameters. This can be avoided by requiring basicness or context-linearity, just as we did in the previous subsection when discussing conditions for the safe applicability of $\downarrow_{\delta,j}$-rules. In fact, the restriction to atmost mtts guarantees that each $\circ$-symbol found in the input is reproduced in the output at most as many times as the input is plugged into the initial expression of $M_2^{\circ\bullet\to\circ\bullet}$.

Lemma 4.36 (bounding the duplication of $\circ$-symbols through $M_2^{\circ\bullet\to\circ\bullet}$)
If M_2 is atmost, then for every $\theta \in T_{\Delta\cup\{\circ,\bullet\}}$:

$$|\tau_{M_2^{\circ\bullet\to\circ\bullet}}(\theta)|_\circ \leq |e_{M_2}|_{v_1} * |\theta|_\circ. \qquad \square$$

The proof (by essentially "recycling" the inductions from the proof of Lemma 4.31) can be found in the appendix. The close connection of this lemma to Lemma 4.31, which might not be apparent on first sight, stems from the fact that the omission of $\circ$-symbols from an input tree for an atmost $M_2^{\circ\bullet\to\circ\bullet}$ is reflected in the final output just in the same manner as in the case of $\bullet$-symbols.

4.8 Assembling the efficiency analysis

Together with Lemma 4.33, the two lemmas from the previous subsection can under appropriate preconditions be used to safely approximate the difference between the numbers of $\circ$- and $\bullet$-symbols reproduced by $M_2^{\circ\bullet\to\circ\bullet}$ from an intermediate result produced by $M_1^{\to\circ\bullet,\kappa}$ for an input tree t in terms of $|\tau_{M_2^{\circ\bullet\to\circ\bullet}}(\tau_{M_1^{\to\circ\bullet}}(t))|_\bullet$ and the initial expressions of $M_1^{\to\circ\bullet}$ and $M_2^{\circ\bullet\to\circ\bullet}$ for some $M_1^{\to\circ\bullet}$ obtained from $M_1^{\to\circ\bullet,\kappa}$ via $\rightsquigarrow^*_{\mathcal{E}_{M_1,M_2}}$.

Lemma 4.37 (putting together some pieces)
Let $\kappa : \{(f,k) \mid f \in F^{(r+1)}, k \in [r]\} \longrightarrow \{0,\ldots,s_{max}\}$ and $M_1^{\to\circ\bullet} = (F, \Sigma, \Delta \cup \{\circ,\bullet\}, e_{M_1^{\to\circ\bullet}}, R_1^{\to\circ\bullet}) \in \mathcal{M}_1^{\to\circ\bullet}$. If $M_1^{\to\circ\bullet,\kappa} \rightsquigarrow^*_{\mathcal{E}_{M_1,M_2}} M_1^{\to\circ\bullet}$, the rules in $R_1^{\to\circ\bullet}$ contain no $\circ$-symbols, M_1 is context-linear, and M_2 is atmost, then for every $t \in T_\Sigma$:

$$|\tau_{M_2^{\circ\bullet\to\circ\bullet}}(\tau_{M_1^{\to\circ\bullet,\kappa}}(t))|_{\circ-\bullet} \leq |e_{M_2}|_{v_1} * |e_{M_1^{\to\circ\bullet}}|_\circ - |\tau_{M_2^{\circ\bullet\to\circ\bullet}}(\tau_{M_1^{\to\circ\bullet}}(t))|_\bullet.$$

Proof
From repeated applications of Lemma 4.33 follows by reflexivity and transitivity of $\leq$ that $\tau_{M_1^{\to\circ\bullet,\kappa}} ; \tau_{M_2^{\circ\bullet\to\circ\bullet}} ; |\,\cdot\,|_{\circ-\bullet} \leq \tau_{M_1^{\to\circ\bullet}} ; \tau_{M_2^{\circ\bullet\to\circ\bullet}} ; |\,\cdot\,|_{\circ-\bullet}$, and hence for every $t \in T_\Sigma$ that $|\tau_{M_2^{\circ\bullet\to\circ\bullet}}(\tau_{M_1^{\to\circ\bullet,\kappa}}(t))|_{\circ-\bullet} \leq |\tau_{M_2^{\circ\bullet\to\circ\bullet}}(\tau_{M_1^{\to\circ\bullet}}(t))|_\circ - |\tau_{M_2^{\circ\bullet\to\circ\bullet}}(\tau_{M_1^{\to\circ\bullet}}(t))|_\bullet$. By Lemma 4.36, with $\theta = \tau_{M_1^{\to\circ\bullet}}(t)$, the minuend is less than or equal to $|e_{M_2}|_{v_1} * |\tau_{M_1^{\to\circ\bullet}}(t)|_\circ$, which by Lemma 4.35—and using that the factor $|e_{M_2}|_{v_1}$ is nonnegative—is less than or equal to $|e_{M_2}|_{v_1} * |e_{M_1^{\to\circ\bullet}}|_\circ$. $\square$

Example 4.38 (rewriting $M_{spine}^{\to\circ\bullet'}$ from Example 4.30 with $\rightsquigarrow^*_{\mathcal{E}_{M_{spine},M_{pfx}}}$)
By rewriting the initial expression and each right-hand side of a rule of $M_{spine}^{\to\circ\bullet'}$ arbitrarily often with rules from $\mathcal{E}_{M_{spine},M_{pfx}}$, we obtain $M_{spine}^{\to\circ\bullet'} \rightsquigarrow^*_{\mathcal{E}_{M_{spine},M_{pfx}}} M_{spine}^{\to\circ\bullet} = (\{f_1^{(2)}, f_2^{(3)}\}, \Sigma_{mon}, \Delta_{tree} \cup \{\circ,\bullet\}, e_{M_{spine}^{\to\circ\bullet}}, R_{spine}^{\to\circ\bullet})$ with $e_{M_{spine}^{\to\circ\bullet}} = f_1\ u_1\ \epsilon$ and rules

$$R^{\to\circ\bullet}_{spine}\colon \begin{array}{ll} f_1\ (\alpha\ u_1)\ y_1 \to \alpha\ (f_1\ u_1\ y_1) & f_2\ (\alpha\ u_1)\ y_1\ y_2 \to \alpha\ (f_1\ u_1\ (\delta\ y_1\ y_2)) \\ f_1\ (\beta\ u_1)\ y_1 \to f_2\ u_1\ y_1\ (\beta\ (\bullet\ \epsilon)) & f_2\ (\beta\ u_1)\ y_1\ y_2 \to f_2\ u_1\ y_1\ (\beta\ (\beta\ y_2)) \\ f_1\ \epsilon\ \quad y_1 \to y_1 & f_2\ \epsilon\ \quad y_1\ y_2 \to \delta\ y_1\ y_2 \end{array}$$

As predicted by Lemma 4.37, the negative number of $\bullet$-symbols in the output computed in Figure 11 using the above rules safely approximates (actually is equal to) the difference of the numbers of $\circ$- and $\bullet$-symbols in the output of Figure 9. □

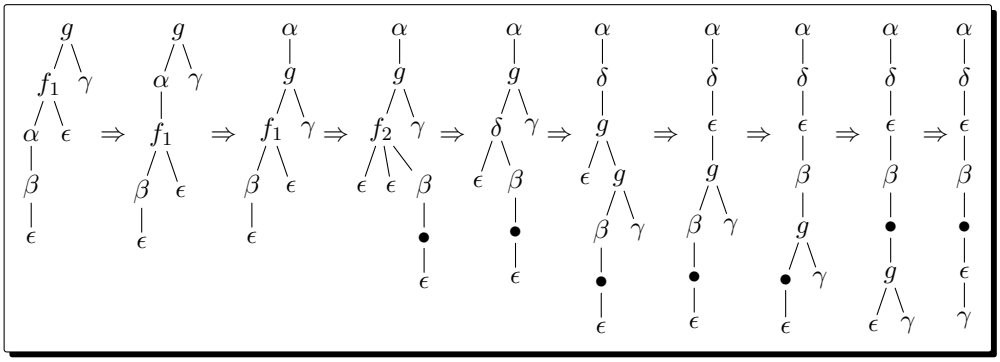

Figure 11: Reduction sequence using $\Rightarrow_{R^{\to\circ\bullet}_{spine} \cup R^{\circ\bullet\to\circ\bullet}_{pfx}}$.

Finally, we prove the main (new, as opposed to Theorem 4.19) theorem of this paper.

Theorem 4.39 (efficiency nondeterioration conditions for general mtts) Let $M_1 = (F, \Sigma, \Delta, e_{M_1}, R_1)$ and $M_2 = (G, \Delta, \Omega, e_{M_2}, R_2)$ be mtts to which Construction 3.1 is applicable. If M_1 is context-linear, M_2 is linear, and there exists $\kappa : \{(f,k) \mid f \in F^{(r+1)}, k \in [r]\} \longrightarrow \{0, \ldots, s_{max}\}$ such that the right-hand sides of all rules in $R_1^{\to\circ\bullet,\kappa}$ can be rewritten with $\Rightarrow^*_{\mathcal{E}_{M_1,M_2}}$ such that all $\circ$-symbols are eliminated, then there exists a constant $d \in \mathbb{N}$ such that for every $t \in T_\Sigma$:

$$cbn_{\overline{M_1;M_2}}(t) \leq cbn_{M_1;M_2}(t) + d.$$

(The reader looking at this theorem without having followed the developments in the previous subsections is referred to Definitions 4.20, 4.26, and 4.27 for the material needed to make sense of its statement.)

Proof

Lemmas 4.18 (using that M_1 is context-linear and M_2 is atmost because it is linear) and 4.24 imply that for every $\kappa : \{(f,k) \mid f \in F^{(r+1)}, k \in [r]\} \longrightarrow \{0, \ldots, s_{max}\}$ and $t \in T_\Sigma$:

$$cbn_{\overline{M_1;M_2}}(t) - cbn_{M_1;M_2}(t) \leq |\tau_{M_2^{\circ\bullet\to\circ\bullet}}(\tau_{M_1^{\to\circ\bullet,\kappa}}(t))|_{\circ-\bullet}.$$

The statement about the particular κ whose existence is a precondition of the theorem implies that there is an mtt $M_1^{\to\circ\bullet} = (F, \Sigma, \Delta \cup \{\circ, \bullet\}, e_{M_1^{\to\circ\bullet}}, R_1^{\to\circ\bullet})$ meeting the requirements of Lemma 4.37. As a consequence, we obtain that for this κ and every $t \in T_\Sigma$:

$$|\tau_{M_2^{\circ\bullet\to\circ\bullet}}(\tau_{M_1^{\to\circ\bullet,\kappa}}(t))|_{\circ-\bullet} \leq |e_{M_2}|_{v_1} * |e_{M_1^{\to\circ\bullet}}|_\circ - |\tau_{M_2^{\circ\bullet\to\circ\bullet}}(\tau_{M_1^{\to\circ\bullet}}(t))|_\bullet.$$

Since $|\tau_{M_2^\circ\bullet\to\circ\bullet}(\tau_{M_1^{\to\circ\bullet}}(t))|_\bullet$ is always nonnegative, the previous two inequalities imply that for every $t \in T_\Sigma$, $cbn_{\overline{M_1;M_2}}(t) \leq cbn_{M_1;M_2}(t) + |e_{M_2}|_{v_1} * |e_{M_1^{\to\circ\bullet}}|_\circ$. □

Note that the preconditions of the previous theorem do not mention $e_{M_1^{\to\circ\bullet,\kappa}}$ in any way. Nevertheless, it makes sense to eliminate—using $\Rightarrow^*_{\mathcal{E}_{M_1,M_2}}$—as many $\circ$-symbols as possible also from $e_{M_1^{\to\circ\bullet,\kappa}}$ because that lowers the constant d up to which we can establish efficiency nondeterioration.

Example 4.40 (efficiency analysis outcome for the running example) Given that M_{spine} is context-linear and M_{pfx} is linear, and in the light of the rule right-hand sides of the mtt $M_{spine}^{\to\circ\bullet}$ from Example 4.38, Theorem 4.39 yields that there exists a constant $d \in \mathbb{N}$ such that $cbn_{\overline{M_{spine};M_{pfx}}}(t) \leq cbn_{M_{spine};M_{pfx}}(t) + d$ for every $t \in T_{\Sigma_{mon}}$. In fact, the constant here is 0 because $|e_{M_{spine}^{\to\circ\bullet}}|_\circ = 0$. Given the observation regarding the running example below Lemma 4.6 and the statements about exactness of approximations additionally proved in [Voi04a] for Lemmas 4.24 and 4.37, we could even conclude the stronger statement that for a given input t the composed program is more efficient than the original program by exactly $|\tau_{M_{pfx}^{\circ\bullet}\to\circ\bullet}(\tau_{M_{spine}^{\to\circ\bullet}}(t))|_\bullet$ steps. This is witnessed, for a concrete input, by the lengths of the call-by-need reduction sequences in Figures 3 and 4 on the one hand and the output computed in Figure 11 on the other. By analyzing $|\tau_{M_{pfx}^{\circ\bullet}\to\circ\bullet}(\tau_{M_{spine}^{\to\circ\bullet}}(t))|_\bullet$, it is also not difficult to see that in general the amount of improvement corresponds to the number of β-blocks in the input tree (i.e. k in a decomposition as in Figure 1). To evaluate the benefits of our approach to efficiency analysis, the interested reader may ponder about how easy or difficult it would have been to work this out directly from the definitions of the original and the composed program as given in Examples 2.2 and 3.3, and whether such an ad-hoc reasoning could be entrusted to a compiler. □

In the course of our analysis we have, where possible, modularized proofs by "factoring out" essential properties guaranteed by certain syntactic restrictions. This provides potential levers for generalization. For example, the references in Definition 4.27 to restrictions from Definition 2.4 could be replaced with references to laxer restrictions as long as Lemmas 2.5 and 4.31 are still satisfied for those. Similarly, the linearity condition on M_2 in Theorem 4.39 is actually a requirement on M_2 to be atmost and context-linear, where the latter requirement could be understood in a "dynamic" sense, meaning that only the inequality from Lemma 2.5(1) must be fulfilled (for every f, t, and k), rather than the syntactic restriction as given in Definition 2.4. The dynamic property is called *finite copying in the parameters with parameter copying bound* 1 in [EM99], and following the strategy from the proof of Lemma 4.10(ii) in [EM03b] it is decidable for an mtt whether or not it is fulfilled. A more liberal inclusion of $\uparrow$- and $\downarrow$-rules into $\mathcal{E}_{M_1,M_2}$ would also be acceptable without requiring a new proof for Theorem 4.39, as long as Lemma 4.32 holds.

4.9 Effectiveness and efficiency of the analysis

At the heart of applying Theorem 4.39 (and of applying Theorem 2 of [Voi02], which is repeated as Theorem 5.1 in Section 5.2 below) to establish call-by-need efficiency

nondeterioration for a particular application of our composition construction is the need to check, given a rewrite system $\mathcal{E}$ containing $\bullet\ (\circ\ v) \to v$, $\circ\ (\bullet\ v) \to v$, and some of the rules from Definition 4.26, whether certain $\circ$- and $\bullet$-annotated versions of rule right-hand sides of M_1 can be rewritten with $\Rightarrow_{\mathcal{E}}^*$ such that all $\circ$-symbols are eliminated. That is, we have to be able to check the following predicate for some $\phi \in RHS(F, \Delta \cup \{\circ, \bullet\}, U, Y)$:

$$\exists \phi' \in RHS(F, \Delta \cup \{\bullet\}, U, Y).\ \phi \Rightarrow_{\mathcal{E}}^* \phi' \tag{7}$$

Since in general $\Rightarrow_{\mathcal{E}}$ is neither confluent nor terminating, we cannot simply solve this task by computing a unique normal form. However, in Chapter 5 of [Voi05] a procedure is proved correct which decides (7) in only a single traversal over ϕ. Using the standard interpretation of arithmetic operations and comparison relations on $\mathbb{Z} \cup \{-\infty\}$, it consists of the following function:

$$\begin{array}{lcl}
sup_{\mathcal{E}} : RHS(F, \Delta \cup \{\circ, \bullet\}, U, Y) \longrightarrow \mathbb{Z} \cup \{-\infty\} & & \\
sup_{\mathcal{E}}(y_k) & = & 0 \\
sup_{\mathcal{E}}(\bullet\ \phi_1) & = & sup_{\mathcal{E}}(\phi_1) + 1 \\
sup_{\mathcal{E}}(\circ\ \phi_1) & = & sup_{\mathcal{E}}(\phi_1) - 1 \\
sup_{\mathcal{E}}(\delta\ \phi_1 \cdots \phi_q) & = & \sum\limits_{j \in [q]} \begin{cases} -\infty & \text{if } \downarrow_{\delta,j} \notin \mathcal{E} \wedge 0 > sup_{\mathcal{E}}(\phi_j) \\ 0 & \text{if } \uparrow_{\delta,j} \notin \mathcal{E} \wedge 0 \leq sup_{\mathcal{E}}(\phi_j) \\ sup_{\mathcal{E}}(\phi_j) & \text{otherwise} \end{cases} \\
sup_{\mathcal{E}}(f\ u_i\ \phi_1 \cdots \phi_r) & = & \sum\limits_{k \in [r]} \begin{cases} -\infty & \text{if } \downarrow_{f,k} \notin \mathcal{E} \wedge 0 > sup_{\mathcal{E}}(\phi_k) \\ 0 & \text{if } \uparrow_{f,k} \notin \mathcal{E} \wedge 0 \leq sup_{\mathcal{E}}(\phi_k) \\ sup_{\mathcal{E}}(\phi_k) & \text{otherwise} \end{cases}
\end{array}$$

which gives a nonnegative result for a particular ϕ iff (7) is fulfilled. The inductive characterization becomes possible by generalizing the task to be solved. In addition to determining whether a given ϕ can be rewritten using $\Rightarrow_{\mathcal{E}}^*$ such that all $\circ$-symbols are eliminated, $sup_{\mathcal{E}}$ also determines the maximal number of extra $\bullet$-symbols that a ϕ can "supply" at its root (in case $sup_{\mathcal{E}}(\phi) \geq 0$) or the minimal number of $\bullet$-symbols to be added on top of ϕ in order to allow elimination of all $\circ$-symbols (the negative result in case $-\infty < sup_{\mathcal{E}}(\phi) < 0$). It returns $-\infty$ if ϕ cannot be rewritten such that all $\circ$-symbols are eliminated, no matter how many additional $\bullet$-symbols are provided at the root. For further details and motivation, please consult [Voi05].

5 Related work

In [VK04a] and [Voi05] the presented composition construction is compared with classical and shortcut deforestation and with *lazy composition* [Voi04b] on a qualitative level and based on examples. Here we relate it to (other) composition techniques for tree transducers (and for attribute grammars). We also discuss previous work on efficiency analysis for tree transducer composition, and for functional programs in general.

5.1 Tree transducer composition

Composition of tree transducers was first considered in [Rou70] for two tdtts, to be composed into a single one. Using a (reversible) decomposition of mtts into tdtts and certain substitution devices [Eng80], the composition of tdtts was reused in [Eng81] to compose a tdtt and an mtt (in this order) into a single mtt. A more direct construction for this composition was later given in [Küh99]. In [EV85] an mtt and a tdtt (in this order) are composed into a single mtt. Our Construction 3.1 generalizes all these composition results. More precisely, if both M_1 and M_2 are tdtts, then it corresponds to the construction in the proof of Theorem 2 in [Rou70]; if M_1 is a tdtt, then it corresponds to Transformation 11 in [Küh99]; and if M_2 is a tdtt, then it corresponds to the construction in the proof of Theorem 4.12 in [EV85].

The potential application of tree transducer composition for optimizing functional programs was first recognized in [Küh97] and [Küh98]. This was also the first work to deal with the composition of two mtts neither of which is a tdtt. It was realized using an indirection via attributed tree transducers (for short *atts*). Essentially the same approach was developed independently in [CDPR98, CDPR99], though not in the syntactical framework of tree transducers. The key idea is to transform the functions from the original program—corresponding to restricted mtts—into atts or attribute grammars (either using a further indirection via tdtts and substitution devices, or using a direct construction in the spirit of [CF82]), to compose these using a construction based on ideas from [Fül81, Gan83, GG84, Gie88], and to transform the result back into an mtt using a construction from [Fra82]. Our direct construction is applicable to more programs because the restrictions on the original mtts are less severe. More precisely, the technique from [Küh97, Küh98] requires M_1 to be context-linear and weakly single-use, and M_2 to be weakly single-use, in both cases using the stronger notion of weakly single-use mentioned below Definition 2.4. In [KV01] the indirect transformation was generalized by weakening the condition on M_2 to a certain *restricted-use* restriction. This condition is incomparable with—i.e., neither strictly more nor strictly less general than—our (weaker) notion of weakly single-use, but it is not too hard to see that Construction 3.1 is also applicable (with a slightly different argument establishing the unambiguity of the definition of $par_{M_2,\phi}(\pi j, g, l)$ in case $lab(\phi, \pi) = \delta$) if M_1 is (only) context-linear and M_2 fulfills a straightforward generalization (by dropping the condition on the initial expression from the definition of the restricted-use property) of both restricted-use and our notion of weakly single-use.

Interestingly, the annotation scheme from Definition 4.7 works also for the indirect transformation via atts, i.e., an analogon of Lemma 4.10 holds, implying that any conditions for efficiency nondeterioration derived using our approach apply to the indirect transformation as well. From a different perspective, the appropriateness of our annotation scheme also for the indirect transformation means that in terms of the number of call-by-name reduction steps the output programs produced by the indirect and the direct composition construction, respectively, perform identically. This gives a negative answer to the question, implicitly raised in [KV01],

whether the fact that the output programs can be syntactically different provides a practical motivation for holding on to the indirect transformation via atts, even though a direct construction (with the obvious attendant benefits, e.g. for an implementor) is available. With respect to call-by-name efficiency it is now clear that there is no point in doing so. Any advantage that a program obtained by the indirect construction might hypothetically enjoy under call-by-need evaluation would thus have to come from a somehow better use of sharing than in the program produced by the direct construction. But such an advantage is highly unlikely if the transformation result is to be an mtt in both cases. Instead, one would have to deviate from the principle that the final program should be an mtt again, by using a different mapping from atts back to functional programs. The most promising candidate would be the mapping into a circular program applied in [Joh87] and [KS87]. We conjecture that this would lead to an overall transformation corresponding to our lazy composition transformation from [Voi04b] (cf. also the related technique in [Nis02, Nis04]).

In Theorem 2 of [Man03] the impressive result is proved that a composition of arbitrary many mtts can be transformed into a single one if the function computed by the composition is of linear size increase, which is decidable. The heart of the associated algorithm is to compose two mtts, the first of which is *finite copying* [EM99] and the second of which is recursion-linear. This is again done using an indirection via atts, applying essentially the composition construction from [Küh98] (cum grano salis, as expressed in [Man03], because the atts are additionally equipped with regular look-ahead). It would be interesting to investigate a direct composition construction also for this setting. Formal efficiency analysis of the final mtt vs. the original program, however, would be complicated by the fact that before we even get hands at the finite copying and the recursion-linear mtt to compose, a fair number of constructions from [Man02], [EV85], and [EM03b] have to be applied. Note that neither is the composition technique from [Man03] more general than our Construction 3.1, nor the other way round. While in [Man03] more than two mtts can be composed and there are no a priori syntactic restrictions on their rules, our construction can also handle mtts the composition of which computes a function that is not of linear size increase. Further note that there is not much room for generalizing the result from [Man03]: the introduction of [KV94] describes a tree transduction that (i) is of **quadratic** size increase, (ii) can be computed by the composition of two mtts, but (iii) cannot be computed by a single mtt. Facts (i) and (ii) are easy to establish; fact (iii) can be considered folklore, in particular in light of the use that is made in [KV94] of the tree transduction in question as the motivating example for generalizing mtts by introducing a more powerful class of tree transducers. A variation of the mentioned tree transduction (by adding the path nondeterministically to an arbitrary leaf rather than to all leaves of the input tree) indicates that the result from [Man03] about the collapse of the composition hierarchy for **nondeterministic** mtts under the assumptions that the relation computed by a composition is a function and is of linear size increase cannot be generalized by dropping the first assumption.

5.2 Efficiency analysis for tree transducer composition

In [Voi02] we have studied the impact of tree transducer composition on call-by-need efficiency for the special case that one of the mtts to be composed is a tdtt. Beside a theorem corresponding to Theorem 4.19 from the present paper, that former study obtained further results by considering only the steps performed by the **second** mtt in the original program, which allows weaker restrictions than context-linearity or basicness of M_1 and atmostness of M_2 to still guarantee, using a simpler annotation, that call-by-need steps are counted. The same approach could be followed if neither of the involved mtts is a tdtt, but we have refrained from doing so. We only repeat the outcome for the special case considered in [Voi02].

Theorem 5.1 (essentially Theorem 2 of [Voi02])
Let $M_1 = (F, \Sigma, \Delta, e_{M_1}, R_1)$ and $M_2 = (G, \Delta, \Omega, e_{M_2}, R_2)$ be mtts, at least one of which is a tdtt. If M_2 is context-linear or basic, $e_{M_1} = f\ u_1\ \theta_1 \cdots \theta_r$ and $e_{M_2} = g\ v_1\ \eta_1 \cdots \eta_s$ for some $f \in F^{(r+1)}$, $g \in G^{(s+1)}$, $\theta_1, \ldots, \theta_r \in T_\Delta$, and $\eta_1, \ldots, \eta_s \in T_\Omega$, and there exists $\lambda \in \{0, 1\}$ such that the right-hand sides of all rules in

$$R_1^{\to\circ\bullet,\lambda}: \quad f\ (\sigma\ u_1 \cdots u_p)\ y_1 \cdots y_r \to \circ^{(1-\lambda)}\ rhs_{M_1,f,\sigma}\ [\delta \twoheadleftarrow \bullet \cdot \delta \mid \delta \in \Delta]\ [f' \twoheadleftarrow \circ^\lambda \cdot f' \mid f' \in F] \qquad \forall f \in F^{(r+1)}, \sigma \in \Sigma^{(p)}$$

can be rewritten with $\Rightarrow^*_{\mathcal{E}_{M_1,M_2}}$ (cf. Definitions 4.26 and 4.27) such that no $\circ$-symbols remain, then:

$$cbn_{\overline{M_1;M_2}} \leq cbn_{M_1;M_2}. \qquad \square$$

Actually, in [Voi02] no $\downarrow_{f,k}$-rules are available to rewrite right-hand sides of rules in $R_1^{\to\circ\bullet,\lambda}$. However, since we have proved Lemma 4.33 (corresponding to Lemma 7 in [Voi02]) for the general case that $\mathcal{E}_{M_1,M_2}$ may contain $\downarrow_{f,k}$-rules, the theorem is also valid as given here. In any case, by Definition 4.27 such rules are contained in $\mathcal{E}_{M_1,M_2}$ only if M_1 is context-linear and M_2 is atmost, and under these conditions we already know by Theorem 4.19 that $cbn_{\overline{M_1;M_2}} \leq cbn_{M_1;M_2}$.

Theorems 4.19, 4.39, and 5.1 are sufficient to formally establish call-by-need efficiency nondeterioration for many typical examples of tree transducer composition for which previously efficiency nondeterioration was assured by ad-hoc reasoning or by experiments only. In particular, this is the case for the various such examples considered in [VK04a]. Our results also capture all the examples for tree transducer composition from [KV01], where the pure elimination of intermediate results was considered as success, without regard to actual numbers of reduction steps. Moreover, they improve on formal efficiency considerations performed in [Küh99] and [Höf99] for the special case that one of the mtts to be composed is a tdtt. More precisely, the respective results of [Küh99], namely that the composed program is at least as efficient as the original program if M_1 is a linear tdtt and M_2 is linear (Corollary 21), or if M_1 is linear, but not a tdtt, and M_2 is a linear tdtt with exactly one state (Theorem 23), are covered and generalized by Theorem 1 of [Voi02], and thus by Theorem 4.19. The same is true for Claim 3.13 and Conjecture 3.21 of [Höf99], together stating an efficiency improvement if M_1 is a tdtt and M_2 is

linear. Theorem 4.34 of [Höf99] states an efficiency improvement for the case that M_1 is context-nondeleting, every rule right-hand side of M_1 contains at least one output symbol, and M_2 is a nondeleting tdtt with exactly one state that is not linear. Theorem 2 of [Voi02]—repeated as Theorem 5.1 above—gives (with $\lambda = 0$, using the fact that all $\uparrow$-rules are in $\mathcal{E}_{M_1,M_2}$ if M_1 is context-nondeleting and M_2 is nondeleting) a more general result in that M_2 is allowed to be any nondeleting tdtt. The only efficiency result proved about tree transducer composition in [Höf99] that goes beyond our theorems is Theorem 4.21 of that work, stating an efficiency improvement if M_1 is context-nondeleting and M_2 is a linear, nondeleting tdtt with exactly one state. We conjecture that context-nondeletingness of M_1 and nondeletingness of M_2 can be dropped here.

In [Voi05] it is argued that Theorem 4.19 remains true even if in addition to the pure number of reduction steps the aspects of *tail calls* and *partial demand* are taken into account. Further, adjustments to Theorems 4.39 and 5.1 are proposed to maintain their applicability in the same setting. This supports the practical relevance of our efficiency results.

5.3 Efficiency analysis of functional programs

The idea to externalize an intensional property of a computation is not new. For example, in [Ros89] for every first-order functional program a step-counting version is produced that—when called with the same arguments—returns the number of *call-by-value* reduction steps performed by the original program. Then, an abstract interpretation of this step-counting program is used to express the worst-case complexity (in dependence on the input size) by recurrence equations. The ultimate goal is to transform these equations into a closed expression, which unfortunately is feasible only for subclasses of programs. In [LG01] more detailed measures than just the number of reduction steps are used by counting different primitive operations separately, thus allowing to compare evaluation costs (for concrete inputs) along different dimensions.

These techniques cannot easily be adapted to call-by-name or call-by-need reduction because for those evaluation strategies the number of reduction steps performed for a certain program expression is not a compositional property. Since the computation time invested in evaluating a function argument depends on how much of the argument's value is actually "needed", the total cost of a function application cannot be understood solely from the cost properties of its ingredients in isolation. In [Wad88] step-counting versions computing upper bounds for the number of call-by-need steps are derived by formally describing this "need" using strictness analysis (see also the related approach in [BH89]). To avoid the problem of having to compute exact demand information, [San95] developed a more operational approach leading to a theory of call-by-name cost equivalence, an axiomatization of which is given by annotating functional programs with a special identity function that represents a single "tick" of computation time and by stating a set of laws that can be used to derive statements about the relative efficiency of program expressions. With a sharp

eye, some of these laws can be recognized in the annotation schemes (for counting the reduction steps of the original and of the composed program) from Section 4.2. In [GS01] syntactic means similar to tick-symbols—so-called *space gadgets*—are used in reasoning about asymptotic (i.e., only up to constant factors) **space** complexity.

While the papers mentioned in this subsection so far are concerned with deriving explicit cost bounds for a given program, or with comparing the efficiency of two concrete programs that compute the same result but are otherwise unrelated (at least not related in a formal, systematic way), our aim was to analyze on a general level the relative efficiency for pairs of two closely connected programs, namely the input and output programs of Construction 3.1. In this respect our aim is similar to that of [San96b], where a call-by-name improvement relation (a nonsymmetric counterpart to the cost equivalence of [San95]) is used to prove the correctness—and, more as a side product, also call-by-name efficiency nondeterioration—of unfold/fold-transformations. In [San96a] that machinery is applied to classical deforestation. A key difference separating our work from that of Sands is that, unlike unfold/fold-transformations, our construction cannot be understood as a sequence of locally equivalence-preserving steps, but rather must be considered as a whole in one go. As a consequence, also the "bookkeeping" of tick-symbols for the transformed program cannot be performed in a similarly modular fashion as in [San96b]. Instead, we had to very carefully analyze the composition construction to invent the annotation scheme in Section 4.3 so that eventually Lemma 4.10 could be proved. The same aspect prevents the application—to our construction—of the call-by-need variant of the approach from [San96b] developed in [MS99].

6 Conclusion

We have developed sufficient conditions for efficiency improvement by tree transducer composition in lazy functional languages. The key was to annotate programs thus that their outputs reflect the numbers of performed reduction steps. Together with the presented "decomposition" of the respective annotation of the composed program in terms of a suitable annotation of the original program, this allows to study the effect of the composition construction on efficiency based solely on the outputs computed by two different annotated versions of the original program. For the sake of proceeding with the analysis we have made compromises on generality and accepted pessimistic approximations at some places. Nevertheless, the sufficient conditions obtained are met in many relevant cases. Moreover, they are suitable to be checked in an optimizing compiler. In fact, for the special case that one of the mtts to be composed is a tdtt the conditions (from [Voi02]) have already been implemented in [Reu03], using a slightly less general variant of the procedure presented in Section 4.9. It is planned to integrate also the conditions for the general case of tree transducer composition presented in this paper. For a more fine-grained analysis it would be worthwhile to investigate efficiency measures where not every reduction step is assumed to incur exactly the same cost, even beyond the consideration of

tail calls. Experiments using the implementation from [Reu03] can hopefully provide help in determining necessary weights.

Another topic of interest would be the setting of functional languages based on call-by-value reduction. Automatic elimination of intermediate results in such languages has scarcely been studied before, and so the same of course holds with respect to efficiency analysis for appropriate transformation techniques. While tree transducer composition is applicable and semantics-preserving irrespective of the chosen reduction strategy, our efficiency analysis would not be easily transferable to the call-by-value setting: the annotation by scattering symbols representing reduction steps over the output tree as in Section 4.2 has an inherent call-by-name flavor.

Acknowledgments

I would like to thank Armin Kühnemann for his comments on a draft version and for (literally) proof reading. Sebastian Maneth pointed me to the decidability of the finite copying in the parameters property. I would also like to thank the anonymous referees and the editor, Joost Engelfriet, for their suggestions, in particular regarding the presentation of the paper and the simplification of some of the proofs in the appendix.

References

[AGS03] D.F.R. van Arkel, J.H.G. van Groningen, and S. Smetsers. Fusion in practice. In *2002 International Workshop on Implementation of Functional Languages, Revised Selected Papers*, volume 2670 of *LNCS*, pages 51–67. Springer-Verlag, 2003.

[BD77] R.M. Burstall and J. Darlington. A transformation system for developing recursive programs. *Journal of the ACM*, 24:44–67, 1977.

[BH89] B. Bjerner and S. Holmström. A compositional approach to time analysis of first order lazy functional programs. In *Functional Programming Languages and Computer Architecture, Proceedings*, pages 157–165. ACM Press, 1989.

[BN98] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[CDPR98] L. Correnson, E. Duris, D. Parigot, and G. Roussel. Symbolic composition. Technical Report 3348, Unité de recherche INRIA Rocquencourt, 1998.

[CDPR99] L. Correnson, E. Duris, D. Parigot, and G. Roussel. Declarative program transformation: A deforestation case-study. In *Principles and Practice of Declarative Programming, Proceedings*, volume 1702 of *LNCS*, pages 360–377. Springer-Verlag, 1999.

[CF82] B. Courcelle and P. Franchi-Zannettacci. Attribute grammars and recursive program schemes. *Theoretical Computer Science*, 17:163–191, 235–257, 1982.

[Chi94] W.N. Chin. Safe fusion of functional expressions II: Further improvements. *Journal of Functional Programming*, 4:515–555, 1994.

[Chi99] O. Chitil. Type inference builds a short cut to deforestation. In *International Conference on Functional Programming, Proceedings*, pages 249–260. ACM Press, 1999.

[DJ90] N. Dershowitz and J.P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–320. Elsevier Science Publishers B.V., 1990.

[EM99] J. Engelfriet and S. Maneth. Macro tree transducers, attribute grammars, and MSO definable tree translations. *Information and Computation*, 154:34–91, 1999.

[EM03a] J. Engelfriet and S. Maneth. A comparison of pebble tree transducers with macro tree transducers. *Acta Informatica*, 39:613–698, 2003.

[EM03b] J. Engelfriet and S. Maneth. Macro tree translations of linear size increase are MSO definable. *SIAM Journal on Computing*, 32:950–1006, 2003.

[Eng75] J. Engelfriet. Bottom-up and top-down tree transformations — A comparison. *Mathematical Systems Theory*, 9:198–231, 1975.

[Eng80] J. Engelfriet. Some open questions and recent results on tree transducers and tree languages. In R.V. Book, editor, *Formal language theory; perspectives and open problems*, pages 241–286. Academic Press, 1980.

[Eng81] J. Engelfriet. Tree transducers and syntax directed semantics. Technical Report Memorandum 363, Technische Hogeschool Twente, 1981.

[ES77] J. Engelfriet and E.M. Schmidt. IO and OI. *Journal of Computer and System Sciences*, 15:328–353, 1977. Continued *Ibid.*, 16:67–99, 1978.

[EV85] J. Engelfriet and H. Vogler. Macro tree transducers. *Journal of Computer and System Sciences*, 31:71–146, 1985.

[FFFK01] M. Felleisen, R.B. Findler, M. Flatt, and S. Krishnamurthi. *How to Design Programs — An Introduction to Programming and Computing.* MIT Press, 2001.

[Fra82] P. Franchi-Zannettacci. *Attributs sémantiques et schémas de programmes.* PhD thesis, Université de Bordeaux I, 1982.

[Fül81] Z. Fülöp. On attributed tree transducers. *Acta Cybernetica*, 5:261–279, 1981.

[FV98] Z. Fülöp and H. Vogler. *Syntax-Directed Semantics — Formal Models Based on Tree Transducers*. Springer-Verlag, 1998.

[Gan83] H. Ganzinger. Increasing modularity and language-independency in automatically generated compilers. *Science of Computer Programming*, 3:223–278, 1983.

[GG84] H. Ganzinger and R. Giegerich. Attribute coupled grammars. In *Symposium on Compiler Construction, Proceedings*, pages 157–170. ACM Press, 1984.

[Gie88] R. Giegerich. Composition and evaluation of attribute coupled grammars. *Acta Informatica*, 25:355–423, 1988.

[GLP93] A. Gill, J. Launchbury, and S.L. Peyton Jones. A short cut to deforestation. In *Functional Programming Languages and Computer Architecture, Proceedings*, pages 223–232. ACM Press, 1993.

[GS01] J. Gustavsson and D. Sands. Possibilities and limitations of call-by-need space improvement. In *International Conference on Functional Programming, Proceedings*, pages 265–276. ACM Press, 2001.

[HJ92] G.W. Hamilton and S.B. Jones. Extending deforestation for first order functional programs. In *1991 Glasgow Workshop on Functional Programming, Proceedings*, pages 134–145. Springer-Verlag, 1992.

[Höf99] M. Höff. Vergleich von Verfahren zur Elimination von Zwischenergebnissen bei funktionalen Programmen. Master's thesis, Dresden University of Technology, 1999.

[Joh87] T. Johnsson. Attribute grammars as a functional programming paradigm. In *Functional Programming Languages and Computer Architecture, Proceedings*, volume 274 of *LNCS*, pages 154–173. Springer-Verlag, 1987.

[Joh02] P. Johann. A generalization of short-cut fusion and its correctness proof. *Higher-Order and Symbolic Computation*, 15:273–300, 2002.

[KGF02a] K. Kakehi, R. Glück, and Y. Futamura. An extension of shortcut deforestation for accumulative list folding. *IEICE Transactions on Information and Systems*, E85-D:1372–1383, 2002.

[KGF02b] K. Kakehi, R. Glück, and Y. Futamura. On deforesting parameters of accumulating maps. In *2001 International Workshop on Logic Based Program Synthesis and Transformation, Selected Papers*, volume 2372 of *LNCS*, pages 46–56. Springer-Verlag, 2002.

[Knu68] D.E. Knuth. Semantics of context-free languages. *Mathematical Systems Theory*, 2:127–145, 1968. Corrections *Ibid.*, 5:95–96, 1971.

[KS87] M.F. Kuiper and S.D. Swierstra. Using attribute grammars to derive efficient functional programs. In *Computing Science in the Netherlands, Proceedings*. SION, 1987.

[Küh97] A. Kühnemann. *Berechnungsstärken von Teilklassen primitiv-rekursiver Programmschemata*. PhD thesis, Dresden University of Technology, 1997.

[Küh98] A. Kühnemann. Benefits of tree transducers for optimizing functional programs. In *Foundations of Software Technology and Theoretical Computer Science, Proceedings*, volume 1530 of *LNCS*, pages 146–157. Springer-Verlag, 1998.

[Küh99] A. Kühnemann. Comparison of deforestation techniques for functional programs and for tree transducers. In *Functional and Logic Programming, Proceedings*, volume 1722 of *LNCS*, pages 114–130. Springer-Verlag, 1999.

[KV94] A. Kühnemann and H. Vogler. Synthesized and inherited functions — A new computational model for syntax-directed semantics. *Acta Informatica*, 31:431–477, 1994.

[KV01] A. Kühnemann and J. Voigtländer. Tree transducer composition as deforestation method for functional programs. Technical Report TUD-FI01-07, Dresden University of Technology, 2001.

[LG01] Y.A. Liu and G. Gómez. Automatic accurate cost-bound analysis for high-level languages. *IEEE Transactions on Computers*, 50:1295–1309, 2001.

[Man02] S. Maneth. The complexity of compositions of deterministic tree transducers. In *Foundations of Software Technology and Theoretical Computer Science, Proceedings*, volume 2556 of *LNCS*, pages 265–276. Springer-Verlag, 2002.

[Man03] S. Maneth. The macro tree transducer hierarchy collapses for functions of linear size increase. In *Foundations of Software Technology and Theoretical Computer Science, Proceedings*, volume 2914 of *LNCS*, pages 326–337. Springer-Verlag, 2003.

[MBPS05] S. Maneth, A. Berlea, T. Perst, and H. Seidl. XML type checking with macro tree transducers. In *Principles of Database Systems, Proceedings*, pages 283–294. ACM Press, 2005.

[MS99] A.K. Moran and D. Sands. Improvement in a lazy context: An operational theory for call-by-need. In *Principles of Programming Languages, Proceedings*, pages 43–56. ACM Press, 1999.

[MW93] S. Marlow and P. Wadler. Deforestation for higher-order functions. In *1992 Glasgow Workshop on Functional Programming, Proceedings*, pages 154–165. Springer-Verlag, 1993.

[Nis02] S. Nishimura. Deforesting in accumulating parameters via type-directed transformations. In *Asian Workshop on Programming Languages and Systems, Informal Proceedings*, pages 145–159, 2002.

[Nis04] S. Nishimura. Fusion with stacks and accumulating parameters. In *Partial Evaluation and Program Manipulation, Proceedings*, pages 101–112. ACM Press, 2004.

[Reu03] S. Reuther. Implementing tree transducer composition for the Glasgow Haskell Compiler. Master's thesis, Dresden University of Technology, 2003.

[Ros89] M. Rosendahl. Automatic complexity analysis. In *Functional Programming Languages and Computer Architecture, Proceedings*, pages 144–156. ACM Press, 1989.

[Rou70] W.C. Rounds. Mappings and grammars on trees. *Mathematical Systems Theory*, 4:257–287, 1970.

[San95] D. Sands. A naïve time analysis and its theory of cost equivalence. *Journal of Logic and Computation*, 5:495–541, 1995.

[San96a] D. Sands. Proving the correctness of recursion-based automatic program transformations. *Theoretical Computer Science*, 167:193–233, 1996.

[San96b] D. Sands. Total correctness by local improvement in the transformation of functional programs. *ACM Transactions on Programming Languages and Systems*, 18:175–234, 1996.

[Sve02] J. Svenningsson. Shortcut fusion for accumulating parameters & zip-like functions. In *International Conference on Functional Programming, Proceedings*, pages 124–132. ACM Press, 2002.

[Tha70] J.W. Thatcher. Generalized^2 sequential machine maps. *Journal of Computer and System Sciences*, 4:339–367, 1970.

[VK04a] J. Voigtländer and A. Kühnemann. Composition of functions with accumulating parameters. *Journal of Functional Programming*, 14:317–363, 2004.

[VK04b] J. Voigtländer and A. Kühnemann. Proof appendix of [VK04a], 2004. Available from *http://wwwtcs.inf.tu-dresden.de/∼voigt/JFP-appendix.pdf*.

[Vog87] H. Vogler. Basic tree transducers. *Journal of Computer and System Sciences*, 34:87–128, 1987.

[Vog91] H. Vogler. Functional description of the contextual analysis in block-structured programming languages: A case study of tree transducers. *Science of Computer Programming*, 16:251–275, 1991.

[Voi01] J. Voigtländer. Composition of restricted macro tree transducers. Master's thesis, Dresden University of Technology, 2001.

[Voi02] J. Voigtländer. Conditions for efficiency improvement by tree transducer composition. In *Rewriting Techniques and Applications, Proceedings*, volume 2378 of *LNCS*, pages 222–236. Springer-Verlag, 2002.

[Voi04a] J. Voigtländer. Formal efficiency analysis for tree transducer composition. Technical Report TUD-FI04-08, Dresden University of Technology, 2004.

[Voi04b] J. Voigtländer. Using circular programs to deforest in accumulating parameters. *Higher-Order and Symbolic Computation*, 17:129–163, 2004.

[Voi05] J. Voigtländer. *Tree Transducer Composition as Program Transformation*. PhD thesis, Dresden University of Technology, 2005.

[Wad71] C.P. Wadsworth. *Semantics and Pragmatics of the Lambda Calculus*. PhD thesis, Oxford University, 1971.

[Wad88] P. Wadler. Strictness analysis aids time analysis. In *Principles of Programming Languages, Proceedings*, pages 119–132. ACM Press, 1988.

[Wad90] P. Wadler. Deforestation: Transforming programs to eliminate trees. *Theoretical Computer Science*, 73:231–248, 1990.

A Proof appendix

The predominant proof principle to be used in this appendix works as follows.

Proof principle (simultaneous induction; cf. e.g. [EV85, FV98, VK04b]) Let Σ be a ranked alphabet. For a tree $t \in T_\Sigma$, a subtree $sub(t, \pi)$ with $\pi \in paths(t)$ is called *proper* if $\pi \neq \varepsilon$ and *direct* if $\pi \in \mathbb{N}_+$. Let (a) and (b) be statements, called *induction hypotheses*, where (a) has a free variable $t \in T_\Sigma$ and (b) has free variables $p \in \mathbb{N}$ and $t_1, \ldots, t_p \in T_\Sigma$. If

(a) ⇐ (b) : for every $t \in T_\Sigma$ such that (a) holds for every proper subtree of t and (b) holds for direct subtrees $t_1, \ldots, t_p$ of t, $p \in \mathbb{N}$, we can prove that (a) holds, and

(b) ⇐ (a) : for every $p \in \mathbb{N}$ and $t_1, \ldots, t_p \in T_\Sigma$ such that (a) holds for each of the $t_1, \ldots, t_p$, we can prove that (b) holds,

then we have proved that (a) holds for every $t \in T_\Sigma$ and that (b) holds for every $p \in \mathbb{N}$ and $t_1, \ldots, t_p \in T_\Sigma$. □

Note the somewhat unusual reference to (a) from within the induction step **(a) ⇐ (b)**, which gives more—and needed—freedom in later proofs. Since it refers only to proper subtrees of the one under consideration, the soundness of the proof principle can still be argued along similar lines as in Section 2.5 of [EV85]. Further note that ordinary structural induction over trees is a special case of simultaneous induction, with (b) being trivially true. Simultaneous and structural induction will also be used for trees with a more refined structure, in particular for elements of $RHS(\cdots)$-sets (cf. Definition 2.1). Obvious properties of normal forms will often be used without mentioning; similarly for substitutions.

Proof of Lemma 2.6
We prove the following two statements (a) and (b):

(a) for every $t \in T_\Sigma$:
For every $f \in F^{(r+1)}$ and $\theta_1, \ldots, \theta_r \in T_{F \cup \Sigma \cup \Delta}(Y)$:
$nf(\Rightarrow_R, f\ t\ \theta_1 \cdots \theta_r) = nf(\Rightarrow_R, f\ t\ y_1 \cdots y_r)[y_k \leftarrow nf(\Rightarrow_R, \theta_k) \mid k \in [r]]$.

(b) for every $p \in \mathbb{N}$ and $t_1, \ldots, t_p \in T_\Sigma$:
For every $r \in \mathbb{N}$, $\theta_1, \ldots, \theta_r \in T_{F \cup \Sigma \cup \Delta}(Y)$, and $\phi \in RHS(F, \Delta, U_p, Y_r)$:
$nf(\Rightarrow_R, \phi[u_1, \ldots, u_p, y_1, \ldots, y_r \leftarrow t_1, \ldots, t_p, \theta_1, \ldots, \theta_r])$
$= nf(\Rightarrow_R, \phi[u_i \leftarrow t_i \mid i \in [p]])[y_k \leftarrow nf(\Rightarrow_R, \theta_k) \mid k \in [r]]$.

by simultaneous induction. Lemma 2.6 then follows from statement (a), taking into account that elements of $T_\Delta \subseteq T_{F \cup \Sigma \cup \Delta}(Y)$ are normal forms with respect to $\Rightarrow_R$.

(a) ⇐ (b) : Assume $t = \sigma\ t_1 \cdots t_p$ for $\sigma \in \Sigma^{(p)}$ and $t_1, \ldots, t_p \in T_\Sigma$. The statement follows from induction hypothesis (b) with $\phi = rhs_{M,f,\sigma}$.

(b) ⇐ (a) : for fixed $r \in \mathbb{N}$ and $\theta_1, \ldots, \theta_r \in T_{F \cup \Sigma \cup \Delta}(Y)$ by structural induction on $\phi \in RHS(F, \Delta, U_p, Y_r)$. The cases $\phi \in Y_r$ and $lab(\phi, \varepsilon) \in \Delta$ are straightforward. The validity in the remaining case is proved as follows.

$\phi = f\ u_{i'}\ \phi_1 \cdots \phi_{r'}$ for some $f \in F^{(r'+1)}$, $u_{i'} \in U_p$, $\phi_1, \ldots, \phi_{r'} \in RHS(F, \Delta, U_p, Y_r)$:

$nf(\Rightarrow_R, (f\ u_{i'}\ \phi_1 \cdots \phi_{r'})[u_1, \ldots, u_p, y_1, \ldots, y_r \leftarrow t_1, \ldots, t_p, \theta_1, \ldots, \theta_r])$
$=$ (by substitution, induction hypothesis (a) for $t_{i'}$, and the induction hypotheses for the $\phi_1, \ldots, \phi_{r'}$)
$nf(\Rightarrow_R, f\ t_{i'}\ y_1 \cdots y_{r'})[y_{k'} \leftarrow nf(\Rightarrow_R, \phi_{k'}[u_i \leftarrow t_i \mid i \in [p]])$
$[y_k \leftarrow nf(\Rightarrow_R, \theta_k) \mid k \in [r]] \mid k' \in [r']]$

$$
\begin{aligned}
&= \text{(by composition of substitutions)}\\
&nf(\Rightarrow_R, f\ t_{i'}\ y_1 \cdots y_{r'})[y_{k'} \leftarrow nf(\Rightarrow_R, \phi_{k'}[u_i \leftarrow t_i \mid i \in [p]]) \mid k' \in [r']]\\
&\qquad\qquad [y_k \leftarrow nf(\Rightarrow_R, \theta_k) \mid k \in [r]]\\
&= \text{(by substitution and induction hypothesis (a) for } t_{i'})\\
&nf(\Rightarrow_R, (f\ u_{i'}\ \phi_1 \cdots \phi_{r'})[u_i \leftarrow t_i \mid i \in [p]])[y_k \leftarrow nf(\Rightarrow_R, \theta_k) \mid k \in [r]]
\end{aligned}
$$

□

Proof of Lemma 4.8

For fixed $p, r \in \mathbb{N}$ we prove the following two statements:

(a) for every $\phi \in RHS(F, \Delta, U_p, Y_r)$:
For every $g \in G^{(s+1)}$ and $\eta_1, \ldots, \eta_s, \eta_1^\star, \ldots, \eta_s^\star \in T_{F \cup G \cup H \cup \Delta \cup \Omega \cup \{\diamond, \star, \circ, nil\}}(U_p \cup Y_r \cup Z \cup Z_G)$ with $nf(\Rightarrow_{R_2^{\diamond\star \to \circ} \cup Pre \cup Pair}, \eta_l^\star) = nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, \eta_l)$ for every $l \in [s]$:
$nf(\Rightarrow_{R_2^{\diamond\star \to \circ} \cup Pre \cup Pair}, g\ \phi[y_k \leftarrow \star\ y_k \mid k \in [r]]\ \eta_1^\star \cdots \eta_s^\star)$
$= nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, g\ \phi\ \eta_1 \cdots \eta_s)$.

(b) for every $r' \in \mathbb{N}$ and $\phi_1, \ldots, \phi_{r'} \in RHS(F, \Delta, U_p, Y_r)$:
For every $f' \in F^{(r'+1)}$, $u_i \in U_p$, and $\eta_{g_1,1}, \ldots, \eta_{g_\mu,s_\mu}, \eta^\star_{g_1,1}, \ldots, \eta^\star_{g_\mu,s_\mu} \in T_{F \cup G \cup H \cup \Delta \cup \Omega \cup \{\diamond, \star, \circ, nil\}}(U_p \cup Y_r \cup Z \cup Z_G)$ with $nf(\Rightarrow_{R_2^{\diamond\star \to \circ} \cup Pre \cup Pair}, \eta^\star_{g',l'}) = nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, \eta_{g',l'})$ for every $g' \in G^{(s'+1)}$ and $l' \in [s']$, for every $\mathcal{C} \subseteq [r'] \times G$, $k' \in [r']$, and $g'' \in G$:

$$
\begin{aligned}
&nf(\Rightarrow_{R_2^{\diamond\star \to \circ} \cup Pre \cup Pair}, nest_{f'}(k', g'', \mathcal{C})[u \quad \leftarrow u_i,\\
&\qquad\qquad\qquad\qquad\qquad\qquad y'_b \quad \leftarrow \phi_b[y_k \leftarrow \star\ y_k \mid k \in [r]] \mid b \in [r'],\\
&\qquad\qquad\qquad\qquad\qquad\qquad z_{g',l'} \leftarrow \eta^\star_{g',l'} \mid g' \in G^{(s'+1)}, l' \in [s']])\\
&= nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, nest_{f'}(k', g'', \mathcal{C})[u \quad \leftarrow u_i,\\
&\qquad\qquad\qquad\qquad\qquad\qquad y'_b \quad \leftarrow \phi_b \quad \mid b \in [r'],\\
&\qquad\qquad\qquad\qquad\qquad\qquad z_{g',l'} \leftarrow \eta_{g',l'} \mid g' \in G^{(s'+1)}, l' \in [s']]).
\end{aligned}
$$

by simultaneous induction. Lemma 4.8 then follows from statement (a) with $\phi = rhs_{M_1,f,\sigma}$ and $\eta_l = \eta_l^\star = z_l$ for every $l \in [s]$.

(a) ⇐ (b) : by case analysis on $\phi \in RHS(F, \Delta, U_p, Y_r)$:

$\phi \in Y_r$:
straightforward, using that $rhs_{M_2^{\diamond\star \to \circ}, g, \star} = g\ v_1\ (\circ\ z_1)\ \cdots\ (\circ\ z_s)$ and the form of rules in *Pre*.

$\phi = \delta\ \phi_1 \cdots \phi_q$ for some $\delta \in \Delta^{(q)}$ and $\phi_1, \ldots, \phi_q \in RHS(F, \Delta, U_p, Y_r)$:
straightforward, using that for every $\psi \in RHS(G, \Omega, V_q, Z_s)$, and thus in particular for $rhs_{M_2^{\diamond\star \to \circ}, g, \delta} = rhs_{M_2, g, \delta}$,

$$
\begin{aligned}
&nf(\Rightarrow_{R_2^{\diamond\star \to \circ} \cup Pre \cup Pair}, \psi[v_j \leftarrow \phi_j[y_k \leftarrow \star\ y_k \mid k \in [r]] \mid j \in [q],\\
&\qquad\qquad z_l \leftarrow \eta_l^\star \mid l \in [s]])\\
&= nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, \psi[v_1, \ldots, v_q, z_1, \ldots, z_s \leftarrow \phi_1, \ldots, \phi_q, \eta_1, \ldots, \eta_s])\,,
\end{aligned}
$$

which can be proved by structural induction. The validity in the case $\psi \in Z_s$ follows from the precondition on the $\eta_1, \ldots, \eta_s, \eta_1^\star, \ldots, \eta_s^\star$. The case $lab(\psi, \varepsilon) \in \Omega$ is

straightforward. In the remaining case, $\psi = g'\ v_j\ \psi_1 \cdots \psi_{s'}$ for some $g' \in G^{(s'+1)}$, $v_j \in V_q$, and $\psi_1, \ldots, \psi_{s'} \in RHS(G, \Omega, V_q, Z_s)$, the induction hypotheses for the $\psi_1, \ldots, \psi_{s'}$ establish the precondition necessary to apply induction hypothesis (a) for ϕ_j, which suffices.

$\underline{\phi = f'\ u_i\ \phi_1 \cdots \phi_{r'}}$ for some $f' \in F^{(r'+1)}$, $u_i \in U_p$, $\phi_1, \ldots, \phi_{r'} \in RHS(F, \Delta, U_p, Y_r)$:

$$
\begin{aligned}
&nf(\Rightarrow_{R_2^{\diamond\star\to\circ} \cup Pre \cup Pair}, g\ (f'\ u_i\ \phi_1 \cdots \phi_{r'})[y_k \leftarrow \star\ y_k \mid k \in [r]]\ \eta_1^\star \cdots \eta_s^\star) \\
&\quad = \text{(by } \Rightarrow_{Pair}) \\
&nf(\Rightarrow_{R_2^{\diamond\star\to\circ} \cup Pre \cup Pair}, \\
&\quad \overline{f'g}\ u_i\ nest_{f'}(1, g_1, \emptyset)[u \leftarrow u_i, \\
&\qquad\qquad y'_b \leftarrow \phi_b[y_k \leftarrow \star\ y_k \mid k \in [r]] \mid b \in [r'], \\
&\qquad\qquad z_{g,l} \leftarrow \eta_l^\star \mid l \in [s], \\
&\qquad\qquad z_{g',l'} \leftarrow nil \mid g' \in G^{(s'+1)} \setminus \{g\}, l' \in [s']] \\
&\qquad \cdots \\
&\qquad nest_{f'}(r', g_\mu, \emptyset)[u \leftarrow u_i, \\
&\qquad\qquad y'_b \leftarrow \phi_b[y_k \leftarrow \star\ y_k \mid k \in [r]] \mid b \in [r'], \\
&\qquad\qquad z_{g,l} \leftarrow \eta_l^\star \mid l \in [s], \\
&\qquad\qquad z_{g',l'} \leftarrow nil \mid g' \in G^{(s'+1)} \setminus \{g\}, l' \in [s']] \\
&\qquad \eta_1^\star \cdots \eta_s^\star) \\
&\quad = \text{(see below)} \\
&nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, \overline{f'g}\ u_i\ nest_{f'}(1, g_1, \emptyset)[u \leftarrow u_i, \\
&\qquad\qquad y'_b \leftarrow \phi_b \mid b \in [r'], \\
&\qquad\qquad z_{g,l} \leftarrow \eta_l \mid l \in [s], \\
&\qquad\qquad z_{g',l'} \leftarrow nil \mid g' \in G^{(s'+1)} \setminus \{g\}, l' \in [s']] \\
&\qquad \cdots \\
&\qquad nest_{f'}(r', g_\mu, \emptyset)[u \leftarrow u_i, \\
&\qquad\qquad y'_b \leftarrow \phi_b \mid b \in [r'], \\
&\qquad\qquad z_{g,l} \leftarrow \eta_l \mid l \in [s], \\
&\qquad\qquad z_{g',l'} \leftarrow nil \mid g' \in G^{(s'+1)} \setminus \{g\}, l' \in [s']] \\
&\qquad \eta_1 \cdots \eta_s) \\
&\quad = \text{(by } \Rightarrow_{Pair}) \\
&nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, g\ (f'\ u_i\ \phi_1 \cdots \phi_{r'})\ \eta_1 \cdots \eta_s)
\end{aligned}
$$

Given the precondition that $nf(\Rightarrow_{R_2^{\diamond\star\to\circ} \cup Pre \cup Pair}, \eta_l^\star) = nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, \eta_l)$ for every $l \in [s]$, the above gap can be closed if we establish for every $k' \in [r']$ and $g'' \in G$:

$$
\begin{aligned}
&nf(\Rightarrow_{R_2^{\diamond\star\to\circ} \cup Pre \cup Pair}, nest_{f'}(k', g'', \emptyset)[u \leftarrow u_i, \\
&\qquad\qquad y'_b \leftarrow \phi_b[y_k \leftarrow \star\ y_k \mid k \in [r]] \mid b \in [r'], \\
&\qquad\qquad z_{g,l} \leftarrow \eta_l^\star \mid l \in [s], \\
&\qquad\qquad z_{g',l'} \leftarrow nil \mid g' \in G^{(s'+1)} \setminus \{g\}, l' \in [s']]) \\
&= nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, nest_{f'}(k', g'', \emptyset)[u \leftarrow u_i, \\
&\qquad\qquad y'_b \leftarrow \phi_b \mid b \in [r'], \\
&\qquad\qquad z_{g,l} \leftarrow \eta_l \mid l \in [s], \\
&\qquad\qquad z_{g',l'} \leftarrow nil \mid g' \in G^{(s'+1)} \setminus \{g\}, l' \in [s']]).
\end{aligned}
$$

But this is an instance of induction hypothesis (b), given the precondition on the $\eta_1, \ldots, \eta_s, \eta_1^\star, \ldots, \eta_s^\star$.

(b) $\Leftarrow$ (a) : for fixed $f' \in F^{(r'+1)}$, $u_i \in U_p$, and $\eta_{g_1,1}, \ldots, \eta_{g_\mu,s_\mu}, \eta^\star_{g_1,1}, \ldots, \eta^\star_{g_\mu,s_\mu} \in T_{F\cup G\cup H\cup\Delta\cup\Omega\cup\{\diamond,\star,\circ,nil\}}(U_p \cup Y_r \cup Z \cup Z_G)$ with the associated precondition, by induction over the reversed subset-order on $\mathcal{P}([r'] \times G)$:

$\underline{\mathcal{C} = [r'] \times G}$:
In this base case we have for every $k' \in [r']$ and $g'' \in G$ that $(k', g'') \in \mathcal{C}$ and thus $nest_{f'}(k', g'', \mathcal{C}) = nil$, which makes the desired equation trivially true.

$\underline{\mathcal{C} \subset [r'] \times G}$:
For every $k' \in [r']$ and $g'' \in G$ with $(k', g'') \notin \mathcal{C}$—the other case is again trivial—we have, setting $s'' = rank_G(g'') - 1$:

$$
\begin{aligned}
&nf(\Rightarrow_{R_2^{\diamond\star\to\circ}\cup Pre\cup Pair}, nest_{f'}(k', g'', \mathcal{C})[u \quad \leftarrow u_i,\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad y'_b \quad \leftarrow \phi_b[y_k \leftarrow \star\, y_k \mid k \in [r]] \mid b \in [r'],\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad z_{g',l'} \leftarrow \eta^\star_{g',l'} \mid g' \in G^{(s'+1)}, l' \in [s']])\\
&\quad = \text{(by definition of } nest_{f'} \text{ and substitution)}\\
&nf(\Rightarrow_{R_2^{\diamond\star\to\circ}\cup Pre\cup Pair},\\
&\quad g''\ \phi_{k'}[y_k \leftarrow \star\, y_k \mid k \in [r]]\\
&\qquad (\overline{k'_{f'}1_{g''}}\ u\ nest_{f'}(1, g_1, \mathcal{C} \cup \{(k', g'')\})\ \cdots\ nest_{f'}(r', g_\mu, \mathcal{C} \cup \{(k', g'')\})\\
&\qquad\qquad z_{g_1,1} \cdots z_{g_\mu,s_\mu})[u \quad \leftarrow u_i,\\
&\qquad\qquad\qquad\qquad\qquad\qquad y'_b \quad \leftarrow \phi_b[y_k \leftarrow \star\, y_k \mid k \in [r]] \mid b \in [r'],\\
&\qquad\qquad\qquad\qquad\qquad\qquad z_{g',l'} \leftarrow \eta^\star_{g',l'} \mid g' \in G^{(s'+1)}, l' \in [s']]\\
&\qquad \cdots\\
&\qquad (\overline{k'_{f'}s''_{g''}}\ u\ nest_{f'}(1, g_1, \mathcal{C} \cup \{(k', g'')\})\ \cdots\ nest_{f'}(r', g_\mu, \mathcal{C} \cup \{(k', g'')\})\\
&\qquad\qquad z_{g_1,1} \cdots z_{g_\mu,s_\mu})[u \quad \leftarrow u_i,\\
&\qquad\qquad\qquad\qquad\qquad\qquad y'_b \quad \leftarrow \phi_b[y_k \leftarrow \star\, y_k \mid k \in [r]] \mid b \in [r'],\\
&\qquad\qquad\qquad\qquad\qquad\qquad z_{g',l'} \leftarrow \eta^\star_{g',l'} \mid g' \in G^{(s'+1)}, l' \in [s']])\\
&\quad = \text{(see below)}\\
&nf(\Rightarrow_{R_2\cup Pre\cup Pair},\\
&\quad g''\ \phi_{k'}\ (\overline{k'_{f'}1_{g''}}\ u\ nest_{f'}(1, g_1, \mathcal{C} \cup \{(k', g'')\})\ \cdots\ nest_{f'}(r', g_\mu, \mathcal{C} \cup \{(k', g'')\})\\
&\qquad\qquad z_{g_1,1} \cdots z_{g_\mu,s_\mu})[u \quad \leftarrow u_i,\\
&\qquad\qquad\qquad\qquad\qquad\qquad y'_b \quad \leftarrow \phi_b \quad \mid b \in [r'],\\
&\qquad\qquad\qquad\qquad\qquad\qquad z_{g',l'} \leftarrow \eta_{g',l'} \mid g' \in G^{(s'+1)}, l' \in [s']]\\
&\qquad \cdots\\
&\qquad (\overline{k'_{f'}s''_{g''}}\ u\ nest_{f'}(1, g_1, \mathcal{C} \cup \{(k', g'')\})\ \cdots\ nest_{f'}(r', g_\mu, \mathcal{C} \cup \{(k', g'')\})\\
&\qquad\qquad z_{g_1,1} \cdots z_{g_\mu,s_\mu})[u \quad \leftarrow u_i,\\
&\qquad\qquad\qquad\qquad\qquad\qquad y'_b \quad \leftarrow \phi_b \quad \mid b \in [r'],\\
&\qquad\qquad\qquad\qquad\qquad\qquad z_{g',l'} \leftarrow \eta_{g',l'} \mid g' \in G^{(s'+1)}, l' \in [s']])\\
&\quad = \text{(by definition of } nest_{f'} \text{ and substitution)}\\
&nf(\Rightarrow_{R_2\cup Pre\cup Pair}, nest_{f'}(k', g'', \mathcal{C})[u \quad \leftarrow u_i,\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad y'_b \quad \leftarrow \phi_b \quad \mid b \in [r'],\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad z_{g',l'} \leftarrow \eta_{g',l'} \mid g' \in G^{(s'+1)}, l' \in [s']])
\end{aligned}
$$

The above gap can be closed by using induction hypothesis (a) for $\phi_{k'}$ if we establish for every $l \in [s'']$:

$$\begin{aligned}
&nf(\Rightarrow_{R_2^{\diamond\star\to\circ}\cup Pre\cup Pair},\\
&\quad(\overline{k'_{f'}l_{g''}}\ u\ nest_{f'}(1,g_1,\mathcal{C}\cup\{(k',g'')\})\ \cdots\ nest_{f'}(r',g_\mu,\mathcal{C}\cup\{(k',g'')\})\\
&\qquad z_{g_1,1}\cdots z_{g_\mu,s_\mu})[u \leftarrow u_i,\\
&\qquad\qquad y'_b \leftarrow \phi_b[y_k \leftarrow\star\ y_k \mid k\in[r]] \mid b\in[r'],\\
&\qquad\qquad z_{g',l'} \leftarrow \eta^\star_{g',l'} \mid g'\in G^{(s'+1)}, l'\in[s']]\\
=\ &nf(\Rightarrow_{R_2\cup Pre\cup Pair},\\
&\quad(\overline{k'_{f'}l_{g''}}\ u\ nest_{f'}(1,g_1,\mathcal{C}\cup\{(k',g'')\})\ \cdots\ nest_{f'}(r',g_\mu,\mathcal{C}\cup\{(k',g'')\})\\
&\qquad z_{g_1,1}\cdots z_{g_\mu,s_\mu})[u \leftarrow u_i,\\
&\qquad\qquad y'_b \leftarrow \phi_b \mid b\in[r'],\\
&\qquad\qquad z_{g',l'} \leftarrow \eta_{g',l'} \mid g'\in G^{(s'+1)}, l'\in[s']]).
\end{aligned}$$

But this follows directly from the precondition on the $\eta_{g_1,1},\ldots,\eta_{g_\mu,s_\mu},\eta^\star_{g_1,1},\ldots,\eta^\star_{g_\mu,s_\mu}$ and applications of the induction hypothesis for $\mathcal{C}\cup\{(k',g'')\}$. □

Proof of Lemma 4.9
For fixed $\sigma\in\Sigma^{(p)}$ and $f\in F^{(r+1)}$ with $r\in\mathbb{N}_+$ we prove by induction over the prefix-order of paths $\pi\in paths(rhs_{M_1,f,\sigma})$ with $lab(rhs_{M_1,f,\sigma},\pi)\notin U_p$ that for every $g\in G^{(s+1)}$ and $l\in[s]$:

$$\begin{aligned}
&nf(\Rightarrow_{R_2^{\diamond\star\to\circ}\cup Pre\cup Pair}, par_{M_2^{\diamond\star\to\circ},\diamond\ rhs_{M_1,f,\sigma}[y_k\leftarrow\star\ y_k\mid k\in[r]]}(1\pi,g,l))\\
=\ &nf(\Rightarrow_{R_2\cup Pre\cup Pair}, par_{M_2,rhs_{M_1,f,\sigma}}(\pi,g,l)).
\end{aligned}$$

Lemma 4.9 then follows by $Y_r\cap U_p=\emptyset$ and the definition of $rhs_{M_1^{\to\diamond\star},f,\sigma}$.

The base case ε is straightforward, using the definition of *par*-functions and the fact that $rhs_{M_2^{\diamond\star\to\circ},g,\diamond}=\circ\ (g\ v_1\ z_1\cdots z_s)$. The inductive case $\pi j\in paths(rhs_{M_1,f,\sigma})$ with $j\in\mathbb{N}_+$ and $lab(rhs_{M_1,f,\sigma},\pi j)\notin U_p$ is shown by case analysis on $lab(rhs_{M_1,f,\sigma},\pi)$ $= lab(\diamond\ rhs_{M_1,f,\sigma}[y_k\leftarrow\star\ y_k\mid k\in[r]],1\pi)$:

$\underline{lab(rhs_{M_1,f,\sigma},\pi)=f'}$ for some $f'\in F^{(r'+1)}$, where $j-1\in[r']$:

$$\begin{aligned}
&nf(\Rightarrow_{R_2^{\diamond\star\to\circ}\cup Pre\cup Pair}, par_{M_2^{\diamond\star\to\circ},\diamond\ rhs_{M_1,f,\sigma}[y_k\leftarrow\star\ y_k\mid k\in[r]]}(1\pi j,g,l))\\
&\quad=\text{ (by definition of } par_{M_2^{\diamond\star\to\circ},\diamond\ rhs_{M_1,f,\sigma}[y_k\leftarrow\star\ y_k\mid k\in[r]]} \text{ and properties of } sub)\\
&nf(\Rightarrow_{R_2^{\diamond\star\to\circ}\cup Pre\cup Pair},\\
&\quad(\overline{(j-1)_{f'}l_g}\ u\ nest_{f'}(1,g_1,\{(j-1,g)\})\ \cdots\ nest_{f'}(r',g_\mu,\{(j-1,g)\})\\
&\qquad z_{g_1,1}\cdots z_{g_\mu,s_\mu})\\
&\quad[u \leftarrow lab(\diamond\ rhs_{M_1,f,\sigma}[y_k\leftarrow\star\ y_k\mid k\in[r]],1\pi 1),\\
&\quad\ y'_b \leftarrow sub(rhs_{M_1,f,\sigma},\pi(b+1))[y_k\leftarrow\star\ y_k\mid k\in[r]]\mid b\in[r'],\\
&\quad\ z_{g',l'} \leftarrow par_{M_2^{\diamond\star\to\circ},\diamond\ rhs_{M_1,f,\sigma}[y_k\leftarrow\star\ y_k\mid k\in[r]]}(1\pi,g',l') \mid g'\in G^{(s'+1)}, l'\in[s']])\\
&\quad=\text{ (see below)}
\end{aligned}$$

$$
\begin{aligned}
&nf(\Rightarrow_{R_2\cup Pre\cup Pair},\\
&\quad \underline{((j-1)_{f'} l_g\ u\ nest_{f'}(1,g_1,\{(j-1,g)\})\ \cdots\ nest_{f'}(r',g_\mu,\{(j-1,g)\})}\\
&\qquad\qquad z_{g_1,1}\cdots z_{g_\mu,s_\mu})\\
&\quad [u \quad \leftarrow lab(rhs_{M_1,f,\sigma},\pi 1),\\
&\quad\ y'_b \quad \leftarrow sub(rhs_{M_1,f,\sigma},\pi(b+1)) \mid b\in[r'],\\
&\quad\ z_{g',l'} \leftarrow par_{M_2,rhs_{M_1,f,\sigma}}(\pi,g',l') \quad \mid g'\in G^{(s'+1)}, l'\in[s']])\\
&\quad = \text{(by definition of } par_{M_2,rhs_{M_1,f,\sigma}})\\
&nf(\Rightarrow_{R_2\cup Pre\cup Pair}, par_{M_2,rhs_{M_1,f,\sigma}}(\pi j,g,l))
\end{aligned}
$$

Since $lab(\diamond\ rhs_{M_1,f,\sigma}[y_k \leftarrow \star\ y_k \mid k\in[r]],1\pi 1) = lab(rhs_{M_1,f,\sigma},\pi 1)\in U_p$ and for every $g'\in G^{(s'+1)}$ and $l'\in[s']$,

$$
\begin{aligned}
&nf(\Rightarrow_{R_2^{\diamond\star\to\circ}\cup Pre\cup Pair}, par_{M_2^{\diamond\star\to\circ},\diamond\ rhs_{M_1,f,\sigma}[y_k\leftarrow\star\ y_k\mid k\in[r]]}(1\pi,g',l'))\\
&= nf(\Rightarrow_{R_2\cup Pre\cup Pair}, par_{M_2,rhs_{M_1,f,\sigma}}(\pi,g',l'))
\end{aligned}
$$

by the induction hypothesis for π, the above gap can be closed if we establish for every $k'\in[r']$ and $g''\in G$:

$$
\begin{aligned}
&nf(\Rightarrow_{R_2^{\diamond\star\to\circ}\cup Pre\cup Pair},\\
&\quad nest_{f'}(k',g'',\{(j-1,g)\})\\
&\quad\ [u \quad \leftarrow lab(\diamond\ rhs_{M_1,f,\sigma}[y_k\leftarrow\star\ y_k\mid k\in[r]],1\pi 1),\\
&\quad\ y'_b \quad \leftarrow sub(rhs_{M_1,f,\sigma},\pi(b+1))[y_k\leftarrow\star\ y_k\mid k\in[r]]\mid b\in[r'],\\
&\quad\ z_{g',l'} \leftarrow par_{M_2^{\diamond\star\to\circ},\diamond\ rhs_{M_1,f,\sigma}[y_k\leftarrow\star\ y_k\mid k\in[r]]}(1\pi,g',l') \quad \mid g'\in G^{(s'+1)}, l'\in[s']])\\
&= nf(\Rightarrow_{R_2\cup Pre\cup Pair}, nest_{f'}(k',g'',\{(j-1,g)\})\\
&\qquad\qquad [u \quad \leftarrow lab(rhs_{M_1,f,\sigma},\pi 1),\\
&\qquad\qquad\ y'_b \quad \leftarrow sub(rhs_{M_1,f,\sigma},\pi(b+1))\mid b\in[r'],\\
&\qquad\qquad\ z_{g',l'} \leftarrow par_{M_2,rhs_{M_1,f,\sigma}}(\pi,g',l') \quad \mid g'\in G^{(s'+1)}, l'\in[s']]).
\end{aligned}
$$

But this is an instance of statement (b) from the proof of Lemma 4.8 for $sub(rhs_{M_1,f,\sigma},\pi 2),\ldots,sub(rhs_{M_1,f,\sigma},\pi(r'+1))\in RHS(F,\Delta,U_p,Y_r)$, where the statements obtained by the induction hypothesis for π establish the necessary precondition on the values substituted for the $z_{g',l'}$.

$\underline{lab(rhs_{M_1,f,\sigma},\pi)=\delta}$ for some $\delta\in\Delta^{(q)}$, where $j\in[q]$:

If there is no $(g\ v_j\ \cdots)$-call in the δ-rules of M_2 and hence also no such call in the δ-rules of $M_2^{\diamond\star\to\circ}$, then both sides of the equation are equal to nil by the definitions of $par_{M_2^{\diamond\star\to\circ},\diamond\ rhs_{M_1,f,\sigma}[y_k\leftarrow\star\ y_k\mid k\in[r]]}$ and $par_{M_2,rhs_{M_1,f,\sigma}}$.

Otherwise, if the unique such call—in both the δ-rules of M_2 and of $M_2^{\diamond\star\to\circ}$—looks, with $g'\in G^{(s'+1)}$ and $\psi_1,\ldots,\psi_s\in RHS(G,\Omega,V_q,Z_{s'})$, as follows:

$$g'\ (\delta\ v_1\cdots v_q)\ z_1\cdots z_{s'}\to\cdots(g\ v_j\ \psi_1\cdots\psi_s)\cdots,$$

then:

$$
\begin{aligned}
&nf(\Rightarrow_{R_2^{\diamond\star\to\circ}\cup Pre\cup Pair}, par_{M_2^{\diamond\star\to\circ},\diamond\, rhs_{M_1,f,\sigma}[y_k\leftarrow\star\, y_k\mid k\in[r]]}(1\pi j,g,l))\\
&\quad = \text{(by definition of } par_{M_2^{\diamond\star\to\circ},\diamond\, rhs_{M_1,f,\sigma}[y_k\leftarrow\star\, y_k\mid k\in[r]]} \text{ and properties of } sub)\\
&nf(\Rightarrow_{R_2^{\diamond\star\to\circ}\cup Pre\cup Pair}, \psi_l[v_{j'} \leftarrow sub(rhs_{M_1,f,\sigma}, \pi j')[y_k\leftarrow\star\, y_k \mid k\in[r]] \quad \mid j'\in[q],\\
&\qquad\qquad z_{l'} \leftarrow par_{M_2^{\diamond\star\to\circ},\diamond\, rhs_{M_1,f,\sigma}[y_k\leftarrow\star\, y_k\mid k\in[r]]}(1\pi,g',l') \mid l'\in[s']])\\
&\quad = \text{(see below)}\\
&nf(\Rightarrow_{R_2\cup Pre\cup Pair}, \psi_l[v_{j'} \leftarrow sub(rhs_{M_1,f,\sigma}, \pi j') \quad \mid j'\in[q],\\
&\qquad\qquad z_{l'} \leftarrow par_{M_2,rhs_{M_1,f,\sigma}}(\pi,g',l') \mid l'\in[s']])\\
&\quad = \text{(by definition of } par_{M_2,rhs_{M_1,f,\sigma}})\\
&nf(\Rightarrow_{R_2\cup Pre\cup Pair}, par_{M_2,rhs_{M_1,f,\sigma}}(\pi j,g,l))
\end{aligned}
$$

Above we used that for every $\psi \in RHS(G,\Omega,V_q,Z_{s'})$:

$$
\begin{aligned}
&nf(\Rightarrow_{R_2^{\diamond\star\to\circ}\cup Pre\cup Pair}, \psi[v_{j'} \leftarrow sub(rhs_{M_1,f,\sigma}, \pi j')[y_k\leftarrow\star\, y_k \mid k\in[r]] \quad \mid j'\in[q],\\
&\qquad\qquad z_{l'} \leftarrow par_{M_2^{\diamond\star\to\circ},\diamond\, rhs_{M_1,f,\sigma}[y_k\leftarrow\star\, y_k\mid k\in[r]]}(1\pi,g',l') \mid l'\in[s']])\\
=\ &nf(\Rightarrow_{R_2\cup Pre\cup Pair}, \psi[v_{j'} \leftarrow sub(rhs_{M_1,f,\sigma}, \pi j') \quad \mid j'\in[q],\\
&\qquad\qquad z_{l'} \leftarrow par_{M_2,rhs_{M_1,f,\sigma}}(\pi,g',l') \mid l'\in[s']]),
\end{aligned}
$$

which can be proved by structural induction. The validity in the case $\psi \in Z_{s'}$ follows from the induction hypothesis for π. The case $lab(\psi,\varepsilon)\in\Omega$ is straightforward. In the remaining case, $\psi = g''\ v_d\ \psi'_1\cdots\psi'_{s''}$ for some $g''\in G^{(s''+1)}$, $v_d\in V_q$, and $\psi'_1,\ldots,\psi'_{s''}\in RHS(G,\Omega,V_q,Z_{s'})$, the induction hypotheses for the $\psi'_1,\ldots,\psi'_{s''}$ establish the precondition necessary to apply statement (a) from the proof of Lemma 4.8 for $sub(rhs_{M_1,f,\sigma},\pi d)\in RHS(F,\Delta,U_p,Y_r)$, which suffices. □

Proof of Lemma 4.15
We prove the following two statements:

(a) for every $t'\in T_{\Delta\cup\{\diamond,\star\}}$:
For every $g\in G^{(s+1)}$ and $\eta_1^\circ,\ldots,\eta_s^\circ,\eta_1^\bullet,\ldots,\eta_s^\bullet,\eta_1^{\circ-\bullet},\ldots,\eta_s^{\circ-\bullet}\in T_{G\cup\Delta\cup\Omega\cup\{\diamond,\star,\circ,\bullet\}}$ with $|nf(\Rightarrow_{R_2^{\diamond\star\to\circ}},\eta_l^\circ)|_\circ - |nf(\Rightarrow_{R_2^{\diamond\to\bullet}},\eta_l^\bullet)|_\bullet = |nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}},\eta_l^{\circ-\bullet})|_{\circ-\bullet}$ for every $l\in[s]$:
$|nf(\Rightarrow_{R_2^{\diamond\star\to\circ}}, g\ t'\ \eta_1^\circ\cdots\eta_s^\circ)|_\circ - |nf(\Rightarrow_{R_2^{\diamond\to\bullet}}, g\ t'[\star\twoheadleftarrow id]\ \eta_1^\bullet\cdots\eta_s^\bullet)|_\bullet$
$= |nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, g\ t'[\diamond\twoheadleftarrow id][\delta\twoheadleftarrow\bullet\cdot\delta \mid \delta\in\Delta]\ \eta_1^{\circ-\bullet}\cdots\eta_s^{\circ-\bullet})|_{\circ-\bullet}$.

(b) for every $q\in\mathbb{N}$ and $t'_1,\ldots,t'_q\in T_{\Delta\cup\{\diamond,\star\}}$:
For every $s\in\mathbb{N}$ and $\eta_1^\circ,\ldots,\eta_s^\circ,\eta_1^\bullet,\ldots,\eta_s^\bullet,\eta_1^{\circ-\bullet},\ldots,\eta_s^{\circ-\bullet}\in T_{G\cup\Delta\cup\Omega\cup\{\diamond,\star,\circ,\bullet\}}$ with $|nf(\Rightarrow_{R_2^{\diamond\star\to\circ}},\eta_l^\circ)|_\circ - |nf(\Rightarrow_{R_2^{\diamond\to\bullet}},\eta_l^\bullet)|_\bullet = |nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}},\eta_l^{\circ-\bullet})|_{\circ-\bullet}$ for every $l\in[s]$, for every $\psi\in RHS(G,\Omega,V_q,Z_s)$:
$|nf(\Rightarrow_{R_2^{\diamond\star\to\circ}},\psi[v_1,\ldots,v_q,z_1,\ldots,z_s\leftarrow t'_1,\ldots,t'_q,\eta_1^\circ,\ldots,\eta_s^\circ])|_\circ$
$- |nf(\Rightarrow_{R_2^{\diamond\to\bullet}},\psi[v_j\leftarrow t'_j[\star\twoheadleftarrow id]\mid j\in[q],\ z_l\leftarrow\eta_l^\bullet \mid l\in[s]])|_\bullet$
$= |nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}},\psi[v_j\leftarrow t'_j[\diamond\twoheadleftarrow id][\delta\twoheadleftarrow\bullet\cdot\delta\mid\delta\in\Delta]\mid j\in[q],\ z_l\leftarrow\eta_l^{\circ-\bullet}\mid l\in[s]])|_{\circ-\bullet}$.

by simultaneous induction. Lemma 4.15 then follows from statement (b) with $q = 1$, $t'_1 = t'$, $s = 0$, and $\psi = e_{M_2}$.

(a) $\Leftarrow$ (b) : by case analysis on $t' \in T_{\Delta\cup\{\diamond,\star\}}$:

$\underline{t' = \diamond\ t'_1}$ for some $t'_1 \in T_{\Delta\cup\{\diamond,\star\}}$:
straightforward, using that $rhs_{M_2^{\diamond\star\to\circ},g,\diamond} = \circ\ (g\ v_1\ z_1 \cdots z_s)$ and $rhs_{M_2^{\diamond\to\bullet},g,\diamond} = \bullet\ (g\ v_1\ z_1 \cdots z_s)$, and applying induction hypothesis (a) for t'_1.

$\underline{t' = \star\ t'_1}$ for some $t'_1 \in T_{\Delta\cup\{\diamond,\star\}}$:

$|nf(\Rightarrow_{R_2^{\diamond\star\to\circ}}, g\ (\star\ t'_1)\ \eta_1^\circ \cdots \eta_s^\circ)|_\circ - |nf(\Rightarrow_{R_2^{\diamond\to\bullet}}, g\ (\star\ t'_1)[\star \twoheadleftarrow id]\ \eta_1^\bullet \cdots \eta_s^\bullet)|_\bullet$
$= $ (by $\Rightarrow_{R_2^{\diamond\star\to\circ}}$, using that $rhs_{M_2^{\diamond\star\to\circ},g,\star} = g\ v_1\ (\circ\ z_1)\ \cdots\ (\circ\ z_s)$, and by substitution)
$|nf(\Rightarrow_{R_2^{\diamond\star\to\circ}}, g\ t'_1\ (\circ\ \eta_1^\circ)\ \cdots\ (\circ\ \eta_s^\circ))|_\circ - |nf(\Rightarrow_{R_2^{\diamond\to\bullet}}, g\ t'_1[\star \twoheadleftarrow id]\ \eta_1^\bullet \cdots \eta_s^\bullet)|_\bullet$
$= $ (by induction hypothesis (a) for t'_1, using that for every $l \in [s]$, $|nf(\Rightarrow_{R_2^{\diamond\star\to\circ}}, \circ\ \eta_l^\circ)|_\circ - |nf(\Rightarrow_{R_2^{\diamond\to\bullet}}, \eta_l^\bullet)|_\bullet = |nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, \circ\ \eta_l^{\circ-\bullet})|_{\circ-\bullet}$, which follows from the precondition on the $\eta_1^\circ, \ldots, \eta_s^\circ, \eta_1^\bullet, \ldots, \eta_s^\bullet, \eta_1^{\circ-\bullet}, \ldots, \eta_s^{\circ-\bullet}$ by properties of $|\,.\,|_\circ$ and $|\,.\,|_{\circ-\bullet}$)
$|nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, g\ t'_1[\diamond \twoheadleftarrow id][\delta \twoheadleftarrow \bullet\cdot\delta \mid \delta \in \Delta]\ (\circ\ \eta_1^{\circ-\bullet})\ \cdots\ (\circ\ \eta_s^{\circ-\bullet}))|_{\circ-\bullet}$
$= $ (by $\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}$, using that $rhs_{M_2^{\star\bullet\to\circ\bullet},g,\star} = g\ v_1\ (\circ\ z_1)\ \cdots\ (\circ\ z_s)$)
$|nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, g\ (\star\ t'_1)[\diamond \twoheadleftarrow id][\delta \twoheadleftarrow \bullet\cdot\delta \mid \delta \in \Delta]\ \eta_1^{\circ-\bullet} \cdots \eta_s^{\circ-\bullet})|_{\circ-\bullet}$

$\underline{t' = \delta'\ t'_1 \cdots t'_q}$ for some $\delta' \in \Delta^{(q)}$ and $t'_1, \ldots, t'_q \in T_{\Delta\cup\{\diamond,\star\}}$:
straightforward, using that $rhs_{M_2^{\diamond\star\to\circ},g,\delta'} = rhs_{M_2,g,\delta'}$, $rhs_{M_2^{\diamond\to\bullet},g,\delta'} = \bullet\ rhs_{M_2,g,\delta'}$, $rhs_{M_2^{\star\bullet\to\circ\bullet},g,\bullet} = \bullet\ (g\ v_1\ z_1 \cdots z_s)$, and $rhs_{M_2^{\star\bullet\to\circ\bullet},g,\delta'} = rhs_{M_2,g,\delta'}$, and applying induction hypothesis (b) with $\psi = rhs_{M_2,g,\delta'}$.

(b) $\Leftarrow$ (a) : for fixed $s \in \mathbb{N}$, $\eta_1^\circ, \ldots, \eta_s^\circ, \eta_1^\bullet, \ldots, \eta_s^\bullet, \eta_1^{\circ-\bullet}, \ldots, \eta_s^{\circ-\bullet} \in T_{G\cup\Delta\cup\Omega\cup\{\diamond,\star,\circ,\bullet\}}$ with the associated precondition, by structural induction on $\psi \in RHS(G, \Omega, V_q, Z_s)$. The validity in the case $\psi \in Z_s$ follows from the precondition on the $\eta_1^\circ, \ldots, \eta_s^\circ, \eta_1^\bullet, \ldots, \eta_s^\bullet, \eta_1^{\circ-\bullet}, \ldots, \eta_s^{\circ-\bullet}$. The case $lab(\psi, \varepsilon) \in \Omega$ is straightforward. In the remaining case, $\psi = g\ v_j\ \psi_1 \cdots \psi_{s'}$ for some $g \in G^{(s'+1)}$, $v_j \in V_q$, and $\psi_1, \ldots, \psi_{s'} \in RHS(G, \Omega, V_q, Z_s)$, the induction hypotheses for the $\psi_1, \ldots, \psi_{s'}$ establish the precondition necessary to apply induction hypothesis (a) for t'_j, which suffices. □

Proof of Lemma 4.22
For fixed $\kappa : \{(f,k) \mid f \in F^{(r+1)}, k \in [r]\} \longrightarrow \{0, \ldots, s_{max}\}$ we prove the following two statements:

(a) $\underline{\text{for every } t \in T_\Sigma}$:
$\overline{\text{For every } f \in F^{(r+1)}}$:
$nf(\Rightarrow_{R_1^{\to\circ\bullet},\kappa}, f\ t\ (\circ^{\kappa_{f,1}}\ y_1)\ \cdots\ (\circ^{\kappa_{f,r}}\ y_r)) = nf(\Rightarrow_{R_1^{\to\star\bullet}}, f\ t\ y_1 \cdots y_r)[\star \twoheadleftarrow \circ^{s_{max}}]$.

(b) for every $p \in \mathbb{N}$ and $t_1, \ldots, t_p \in T_\Sigma$:
For every $r \in \mathbb{N}$ and $\phi \in RHS(F, \Delta \cup \{\bullet\}, U_p, Y_r)$:
$nf(\Rightarrow_{R_1^{\to\circ\bullet,\kappa}}, \phi[f' \leftarrow f' \cdot (id \times \circ^{\kappa_{f',1}} \times \cdots \times \circ^{\kappa_{f',r'}}) \mid f' \in F^{(r'+1)}]$
$[u_i \leftarrow t_i \mid i \in [p]])$
$= nf(\Rightarrow_{R_1^{\to\star\bullet}}, \phi[u_i \leftarrow t_i \mid i \in [p]])[\star \leftarrow \circ^{s_{max}}]$.

by simultaneous induction. Lemma 4.22 then follows from statement (b) with $p = 1$, $t_1 = t$, $r = 0$, and $\phi = e_{M_1}[\delta \leftarrow \bullet \cdot \delta \mid \delta \in \Delta]$, taking into account the definitions of $e_{M_1^{\to\circ\bullet,\kappa}}$ and $e_{M_1^{\to\star\bullet}}$.

(a) $\Leftarrow$ (b) : Assume $t = \sigma\ t_1 \cdots t_p$ for $\sigma \in \Sigma^{(p)}$ and $t_1, \ldots, t_p \in T_\Sigma$. We calculate:

$nf(\Rightarrow_{R_1^{\to\circ\bullet,\kappa}}, f\ (\sigma\ t_1 \cdots t_p)\ (\circ^{\kappa_{f,1}}\ y_1)\ \cdots\ (\circ^{\kappa_{f,r}}\ y_r))$
$=$ (by $\Rightarrow_{R_1^{\to\circ\bullet,\kappa}}$, using the definition of $rhs_{M_1^{\to\circ\bullet,\kappa},f,\sigma}$)
$nf(\Rightarrow_{R_1^{\to\circ\bullet,\kappa}}, rhs_{M_1,f,\sigma}[\delta \leftarrow \bullet \cdot \delta \mid \delta \in \Delta]$
$[f' \leftarrow f' \cdot (id \times \circ^{\kappa_{f',1}} \times \cdots \times \circ^{\kappa_{f',r'}}) \mid f' \in F^{(r'+1)}]$
$[u_i \leftarrow t_i \mid i \in [p],$
$y_k \leftarrow \circ^{\overline{\kappa}_{f,k}}\ (\circ^{\kappa_{f,k}}\ y_k) \mid k \in [r]])$
$=$ (by statement (b) from the proof of Lemma 2.6 for the mtt $M_1^{\to\circ\bullet,\kappa}$, with
$\phi = rhs_{M_1,f,\sigma}[\delta \leftarrow \bullet \cdot \delta \mid \delta \in \Delta]$
$[f' \leftarrow f' \cdot (id \times \circ^{\kappa_{f',1}} \times \cdots \times \circ^{\kappa_{f',r'}}) \mid f' \in F^{(r'+1)}]$,
and $\overline{\kappa}_{f,k} + \kappa_{f,k} = s_{max}$ for every $k \in [r]$)
$nf(\Rightarrow_{R_1^{\to\circ\bullet,\kappa}}, rhs_{M_1,f,\sigma}[\delta \leftarrow \bullet \cdot \delta \mid \delta \in \Delta]$
$[f' \leftarrow f' \cdot (id \times \circ^{\kappa_{f',1}} \times \cdots \times \circ^{\kappa_{f',r'}}) \mid f' \in F^{(r'+1)}]$
$[u_i \leftarrow t_i \mid i \in [p]])[y_k \leftarrow \circ^{s_{max}}\ y_k \mid k \in [r]]$
$=$ (by induction hypothesis (b) with $\phi = rhs_{M_1,f,\sigma}[\delta \leftarrow \bullet \cdot \delta \mid \delta \in \Delta]$)
$nf(\Rightarrow_{R_1^{\to\star\bullet}}, rhs_{M_1,f,\sigma}[\delta \leftarrow \bullet \cdot \delta \mid \delta \in \Delta][u_i \leftarrow t_i \mid i \in [p]])[\star \leftarrow \circ^{s_{max}}]$
$[y_k \leftarrow \circ^{s_{max}}\ y_k \mid k \in [r]]$
$=$ (by statement (b) from the proof of Lemma 2.6 for the mtt $M_1^{\to\star\bullet}$, with
$\phi = rhs_{M_1,f,\sigma}[\delta \leftarrow \bullet \cdot \delta \mid \delta \in \Delta]$, and properties of substitution)
$nf(\Rightarrow_{R_1^{\to\star\bullet}}, rhs_{M_1,f,\sigma}[\delta \leftarrow \bullet \cdot \delta \mid \delta \in \Delta][u_i \leftarrow t_i \mid i \in [p],$
$y_k \leftarrow \star\ y_k \mid k \in [r]])[\star \leftarrow \circ^{s_{max}}]$
$=$ (by $\Rightarrow_{R_1^{\to\star\bullet}}$, using the definition of $rhs_{M_1^{\to\star\bullet},f,\sigma}$)
$nf(\Rightarrow_{R_1^{\to\star\bullet}}, f\ (\sigma\ t_1 \cdots t_p)\ y_1 \cdots y_r)[\star \leftarrow \circ^{s_{max}}]$

(b) $\Leftarrow$ (a) : for fixed $r \in \mathbb{N}$ by structural induction on $\phi \in RHS(F, \Delta \cup \{\bullet\}, U_p, Y_r)$. The cases $\phi \in Y_r$ and $lab(\phi, \varepsilon) \in \Delta \cup \{\bullet\}$ are straightforward. The validity in the remaining case is proved as follows.

$\phi = f\ u_{i'}\ \phi_1 \cdots \phi_{r''}$ for some $f \in F^{(r''+1)}$, $u_{i'} \in U_p$, and $\phi_1, \ldots, \phi_{r''} \in RHS(F, \Delta \cup \{\bullet\}, U_p, Y_r)$:

$nf(\Rightarrow_{R_1^{\to\circ\bullet,\kappa}}, (f\ u_{i'}\ \phi_1 \cdots \phi_{r''})[f' \leftarrow f' \cdot (id \times \circ^{\kappa_{f',1}} \times \cdots \times \circ^{\kappa_{f',r'}}) \mid f' \in F^{(r'+1)}]$
$[u_i \leftarrow t_i \mid i \in [p]])$

$$
\begin{aligned}
&= \text{(by properties of substitution)}\\
&nf(\Rightarrow_{R_1^{\rightarrow\circ\bullet,\kappa}}, (f\ u_{i'}\ (\circ^{\kappa_{f,1}}\ y_1)\ \cdots\ (\circ^{\kappa_{f,r''}}\ y_{r''}))\\
&\qquad [u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\ y_k \leftarrow \phi_k[f' \twoheadleftarrow f' \cdot (id \times \circ^{\kappa_{f',1}} \times \cdots \times \circ^{\kappa_{f',r'}}) \mid f' \in F^{(r'+1)}]\\
&\qquad\qquad [u_i \leftarrow t_i \mid i \in [p]] \mid k \in [r'']])\\
&= \text{(by statement (b) from the proof of Lemma 2.6 for the mtt } M_1^{\rightarrow\circ\bullet,\kappa}\text{, with}\\
&\quad\ \phi = f\ u_{i'}\ (\circ^{\kappa_{f,1}}\ y_1)\ \cdots\ (\circ^{\kappa_{f,r''}}\ y_{r''}))\\
&nf(\Rightarrow_{R_1^{\rightarrow\circ\bullet,\kappa}}, (f\ u_{i'}\ (\circ^{\kappa_{f,1}}\ y_1)\ \cdots\ (\circ^{\kappa_{f,r''}}\ y_{r''}))[u_i \leftarrow t_i \mid i \in [p]])\\
&\ [y_k \leftarrow nf(\Rightarrow_{R_1^{\rightarrow\circ\bullet,\kappa}}, \phi_k[f' \twoheadleftarrow f' \cdot (id \times \circ^{\kappa_{f',1}} \times \cdots \times \circ^{\kappa_{f',r'}}) \mid f' \in F^{(r'+1)}]\\
&\qquad\qquad [u_i \leftarrow t_i \mid i \in [p]]) \mid k \in [r'']]\\
&= \text{(by substitution, induction hypothesis (a) for } t_{i'}\text{, and the induction}\\
&\quad\ \text{hypotheses for the } \phi_1, \ldots, \phi_{r''})\\
&nf(\Rightarrow_{R_1^{\rightarrow\star\bullet}}, f\ t_{i'}\ y_1 \cdots y_{r''})[\star \twoheadleftarrow \circ^{s_{max}}]\\
&\ [y_k \leftarrow nf(\Rightarrow_{R_1^{\rightarrow\star\bullet}}, \phi_k[u_i \leftarrow t_i \mid i \in [p]])[\star \twoheadleftarrow \circ^{s_{max}}] \mid k \in [r'']]\\
&= \text{(by properties of substitution and statement (a) from the proof of}\\
&\quad\ \text{Lemma 2.6 for the mtt } M_1^{\rightarrow\star\bullet}\text{)}\\
&nf(\Rightarrow_{R_1^{\rightarrow\star\bullet}}, (f\ u_{i'}\ \phi_1 \cdots \phi_{r''})[u_i \leftarrow t_i \mid i \in [p]])[\star \twoheadleftarrow \circ^{s_{max}}]
\end{aligned}
$$

□

The following two auxiliary lemmas are needed in the proofs of Lemmas 4.23, 4.31, and 4.33 (and thus indirectly also in that of Lemma 4.36).

Lemma A.1 (auxiliary, later to be instantiated for $M_2^{\star\bullet\rightarrow\circ\bullet}$ and $M_2^{\circ\bullet\rightarrow\circ\bullet}$)
Let $M_2' = (G, \Delta', \Omega \cup \{\circ, \bullet\}, e, R_2')$ be an mtt that uses context variables from Z. For every $g \in G^{(s+1)}$, $t' \in T_{\Delta'}$, and $\eta_1, \ldots, \eta_s \in T_{G \cup \Delta' \cup \Omega \cup \{\circ,\bullet\}}(Z)$:[1]

$$
\begin{aligned}
&|nf(\Rightarrow_{R_2'}, g\ t'\ \eta_1 \cdots \eta_s)|_{\circ-\bullet}\\
&= |nf(\Rightarrow_{R_2'}, g\ t'\ z_1 \cdots z_s)|_{\circ-\bullet} + \sum_{l \in [s]} |nf(\Rightarrow_{R_2'}, g\ t'\ z_1 \cdots z_s)|_{z_l} * |nf(\Rightarrow_{R_2'}, \eta_l)|_{\circ-\bullet}.
\end{aligned}
$$

The same holds with $|\,.\,|_\circ$ or $|\,.\,|_{\bullet-\circ} = |\,.\,|_\bullet - |\,.\,|_\circ$ instead of $|\,.\,|_{\circ-\bullet} = |\,.\,|_\circ - |\,.\,|_\bullet$.

Proof
By statement (a) from the proof of Lemma 2.6 for the mtt M_2', and obvious properties of $|\,.\,|_{\circ-\bullet}$, $|\,.\,|_\circ$, $|\,.\,|_{\bullet-\circ}$, and substitution. □

Lemma A.2 (auxiliary)
Let $M_2^{\star\bullet\rightarrow\circ\bullet} = (G, \Delta \cup \{\star, \bullet\}, \Omega \cup \{\circ, \bullet\}, e_{M_2}, R_2^{\star\bullet\rightarrow\circ\bullet})$ and $M_2^{\circ\bullet\rightarrow\circ\bullet} = (G, \Delta \cup \{\circ, \bullet\}, \Omega \cup \{\circ, \bullet\}, e_{M_2}, R_2^{\circ\bullet\rightarrow\circ\bullet})$ be the mtts from Definitions 4.13 and 4.20, respectively. For every $g \in G^{(s+1)}$ and $l \in [s]$:

1. for every $t' \in T_{\Delta \cup \{\star,\bullet\}}$ and $t'' \in T_{\Delta \cup \{\circ,\bullet\}}$ with $t'[\star \twoheadleftarrow id] = t''[\circ \twoheadleftarrow id]$:
   $|nf(\Rightarrow_{R_2^{\star\bullet\rightarrow\circ\bullet}}, g\ t'\ z_1 \cdots z_s)|_{z_l} = |nf(\Rightarrow_{R_2^{\circ\bullet\rightarrow\circ\bullet}}, g\ t''\ z_1 \cdots z_s)|_{z_l}$

[1] To minimize the need for brackets when writing sum formulae, we will use the conventions that $*$ binds stronger than a $\sum$ to the left of it, while $\sum$ binds stronger than $+$, so that, e.g., an expression of the form $\sum a * b + c * \sum d * e$ is parsed as $\big(\sum(a * b)\big) + c * \big(\sum(d * e)\big)$.

2. for every $t', t'' \in T_{\Delta\cup\{\circ,\bullet\}}$ with $t'[\bullet \twoheadleftarrow id] = t''[\bullet \twoheadleftarrow id]$:
   $|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ t'\ z_1 \cdots z_s)|_{z_l} = |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ t''\ z_1 \cdots z_s)|_{z_l}$.

Proof

1. $|nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, g\ t'\ z_1 \cdots z_s)|_{z_l}$
   $=$ (since, due to the rules in $R_2^{\star\bullet\to\circ\bullet}$,
   $nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, g\ t'[\star \twoheadleftarrow id]\ z_1 \cdots z_s) = nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, g\ t'\ z_1 \cdots z_s)[\circ \twoheadleftarrow id]$)
   $|nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, g\ t'[\star \twoheadleftarrow id]\ z_1 \cdots z_s)|_{z_l}$
   $=$ (by $t'[\star \twoheadleftarrow id] = t''[\circ \twoheadleftarrow id] \in T_{\Delta\cup\{\bullet\}}$ and the fact that restricting $R_2^{\star\bullet\to\circ\bullet}$ and $R_2^{\circ\bullet\to\circ\bullet}$, respectively, to rules at elements of $\Delta \cup \{\bullet\}$ leads to identical rule sets)
   $|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ t''[\circ \twoheadleftarrow id]\ z_1 \cdots z_s)|_{z_l}$
   $=$ (since, due to the rules in $R_2^{\circ\bullet\to\circ\bullet}$,
   $nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ t''[\circ \twoheadleftarrow id]\ z_1 \cdots z_s) = nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ t''\ z_1 \cdots z_s)[\circ \twoheadleftarrow id]$)
   $|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ t''\ z_1 \cdots z_s)|_{z_l}$

2. Similar, but slightly simpler. □

Proof of Lemma 4.23
For a context-linear mtt M_2 we prove the following two statements:

(a) for every $t' \in T_{\Delta\cup\{\star,\bullet\}}$:
For every $g \in G^{(s+1)}$:
$|nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, g\ t'\ z_1 \cdots z_s)|_{\circ-\bullet} \le |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ t'[\star \twoheadleftarrow \circ^{s_{max}}]\ z_1 \cdots z_s)|_{\circ-\bullet}$.

(b) for every $q \in \mathbb{N}$ and $t'_1, \ldots, t'_q \in T_{\Delta\cup\{\star,\bullet\}}$:
For every $s \in \mathbb{N}$ and $\psi \in RHS(G, \Omega, V_q, Z_s)$:
$|nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, \psi[v_j \leftarrow t'_j \mid j \in [q]])|_{\circ-\bullet}$
$\le |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \psi[v_j \leftarrow t'_j[\star \twoheadleftarrow \circ^{s_{max}}] \mid j \in [q]])|_{\circ-\bullet}$.

by simultaneous induction. Lemma 4.23 then follows from statement (b) with $q = 1$, $t'_1 = t'$, $s = 0$, and $\psi = e_{M_2}$.

(a) ⇐ (b) : by case analysis on $t' \in T_{\Delta\cup\{\star,\bullet\}}$:

$t' = \star\ t'_1$ for some $t'_1 \in T_{\Delta\cup\{\star,\bullet\}}$:

$|nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, g\ (\star\ t'_1)\ z_1 \cdots z_s)|_{\circ-\bullet}$
$=$ (by $\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}$, using that $rhs_{M_2^{\star\bullet\to\circ\bullet},g,\star} = g\ v_1\ (\circ\ z_1)\ \cdots\ (\circ\ z_s)$)
$|nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, g\ t'_1\ (\circ\ z_1)\ \cdots\ (\circ\ z_s))|_{\circ-\bullet}$
$=$ (by Lemma A.1 for the mtt $M_2^{\star\bullet\to\circ\bullet}$)
$|nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, g\ t'_1\ z_1 \cdots z_s)|_{\circ-\bullet} + \sum_{l\in[s]} |nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, g\ t'_1\ z_1 \cdots z_s)|_{z_l} * |nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, \circ\ z_l)|_{\circ-\bullet}$
$\le$ (by Lemma 2.5(1) for the mtt $M_2^{\star\bullet\to\circ\bullet}$ and properties of $|\,\cdot\,|_{\circ-\bullet}$)

$|nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, g\ t'_1\ z_1 \cdots z_s)|_{\circ-\bullet} + \sum_{l\in[s]} 1$
$\leq$ (by induction hypothesis (a) for t'_1 and $\sum_{l\in[s]} 1 = s \leq s_{max}$)
$|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ t'_1[\star \nleftarrow \circ^{s_{max}}]\ z_1 \cdots z_s)|_{\circ-\bullet} + s_{max}$
$=$ (by substitution and $\Rightarrow^{s_{max}}_{R_2^{\circ\bullet\to\circ\bullet}}$, using that $rhs_{M_2^{\circ\bullet\to\circ\bullet},g,\circ} = \circ\ (g\ v_1\ z_1 \cdots z_s)$, and by properties of $|\,.\,|_{\circ-\bullet}$)
$|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ (\star\ t'_1)[\star \nleftarrow \circ^{s_{max}}]\ z_1 \cdots z_s)|_{\circ-\bullet}$

$\underline{t' = \bullet\ t'_1}$ for some $t'_1 \in T_{\Delta\cup\{\star,\bullet\}}$:
straightforward, using that $rhs_{M_2^{\star\bullet\to\circ\bullet},g,\bullet} = rhs_{M_2^{\circ\bullet\to\circ\bullet},g,\bullet} = \bullet\ (g\ v_1\ z_1 \cdots z_s)$ and applying induction hypothesis (a) for t'_1.

$\underline{t' = \delta\ t'_1 \cdots t'_q}$ for some $\delta \in \Delta^{(q)}$ and $t'_1, \ldots, t'_q \in T_{\Delta\cup\{\star,\bullet\}}$:
straightforward, using that $rhs_{M_2^{\star\bullet\to\circ\bullet},g,\delta} = rhs_{M_2^{\circ\bullet\to\circ\bullet},g,\delta} = rhs_{M_2,g,\delta}$ and applying induction hypothesis (b) with $\psi = rhs_{M_2,g,\delta}$.

(b) $\Leftarrow$ (a) : for fixed $s \in \mathbb{N}$ by structural induction on $\psi \in RHS(G,\Omega,V_q,Z_s)$. The cases $\psi \in Z_s$ and $lab(\psi,\varepsilon) \in \Omega$ are straightforward. The validity in the remaining case is proved as follows.

$\underline{\psi = g\ v_{j'}\ \psi_1 \cdots \psi_{s'}}$ for some $g \in G^{(s'+1)}$, $v_{j'} \in V_q$, $\psi_1, \ldots, \psi_{s'} \in RHS(G,\Omega,V_q,Z_s)$:

$|nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, (g\ v_{j'}\ \psi_1 \cdots \psi_{s'})[v_j \leftarrow t'_j \mid j \in [q]])|_{\circ-\bullet}$
$=$ (by substitution and Lemma A.1 for the mtt $M_2^{\star\bullet\to\circ\bullet}$)
$|nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, g\ t'_{j'}\ z_1 \cdots z_{s'})|_{\circ-\bullet}$
$+ \sum_{l'\in[s']} |nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, g\ t'_{j'}\ z_1 \cdots z_{s'})|_{z_{l'}} * |nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, \psi_{l'}[v_j \leftarrow t'_j \mid j \in [q]])|_{\circ-\bullet}$
$\leq$ (by induction hypothesis (a) for $t'_{j'}$ and Lemma A.2(1))
$|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ t'_{j'}[\star \nleftarrow \circ^{s_{max}}]\ z_1 \cdots z_{s'})|_{\circ-\bullet}$
$+ \sum_{l'\in[s']} |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ t'_{j'}[\star \nleftarrow \circ^{s_{max}}]\ z_1 \cdots z_{s'})|_{z_{l'}} * |nf(\Rightarrow_{R_2^{\star\bullet\to\circ\bullet}}, \psi_{l'}[v_j \leftarrow t'_j \mid j \in [q]])|_{\circ-\bullet}$
$\leq$ (by the induction hypotheses for the $\psi_1, \ldots, \psi_{s'}$)
$|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ t'_{j'}[\star \nleftarrow \circ^{s_{max}}]\ z_1 \cdots z_{s'})|_{\circ-\bullet}$
$+ \sum_{l'\in[s']} |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ t'_{j'}[\star \nleftarrow \circ^{s_{max}}]\ z_1 \cdots z_{s'})|_{z_{l'}} * |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \psi_{l'}[v_j \leftarrow t'_j[\star \nleftarrow \circ^{s_{max}}] \mid j \in [q]])|_{\circ-\bullet}$
$=$ (by substitution and Lemma A.1 for the mtt $M_2^{\circ\bullet\to\circ\bullet}$)
$|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, (g\ v_{j'}\ \psi_1 \cdots \psi_{s'})[v_j \leftarrow t'_j[\star \nleftarrow \circ^{s_{max}}] \mid j \in [q]])|_{\circ-\bullet}$ □

Proof of Lemma 4.31
For an atmost mtt M_2 we prove the following two statements:

(a) $\underline{\text{for every } \theta \in T_{\Delta\cup\{\circ,\bullet\}}:}$
For every $\theta' \in T_{\Delta\cup\{\circ,\bullet\}}$ with $\theta \Rightarrow^*_{\{\bullet\ v\to v\}} \theta'$ and every $g \in G^{(s+1)}$:
$|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta\ z_1 \cdots z_s)|_{\bullet-\circ} - |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta'\ z_1 \cdots z_s)|_{\bullet-\circ} \leq |\theta|_\bullet - |\theta'|_\bullet$.

(b) for every $q \in \mathbb{N}$ and $\theta_1, \ldots, \theta_q \in T_{\Delta \cup \{\circ,\bullet\}}$:
For every $\theta'_1, \ldots, \theta'_q \in T_{\Delta\cup\{\circ,\bullet\}}$ with $\theta_j \Rightarrow^*_{\{\bullet\, v \to v\}} \theta'_j$ for every $j \in [q]$, for every $s \in \mathbb{N}$ and $\psi \in RHS(G, \Omega, V_q, Z_s)$, if M_2 is context-linear or ψ contains no nested calls, then:
$$\begin{aligned}
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \psi[v_j \leftarrow \theta_j \mid j \in [q]])|_{\bullet-\circ} \\
&- |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \psi[v_j \leftarrow \theta'_j \mid j \in [q]])|_{\bullet-\circ} \\
&\leq \sum_{j\in[q]} |\psi|_{v_j} * (|\theta_j|_\bullet - |\theta'_j|_\bullet).
\end{aligned}$$

by simultaneous induction. The first statement of Lemma 4.31 then follows from statement (a), taking into account that $|\varrho|_{\bullet-\circ} = -|\varrho|_{\circ-\bullet}$ for every $\varrho \in T_{\Omega\cup\{\circ,\bullet\}}(Z)$. The second statement of Lemma 4.31, for atleast M_2, is obtained correspondingly after replacing all occurrences of $\leq$ and "-linear" by $\geq$ and "-nondeleting", respectively, in the above statements and the following proof.

(a) $\Leftarrow$ (b) : by case analysis on $\theta \in T_{\Delta\cup\{\circ,\bullet\}}$ and $\theta \Rightarrow^*_{\{\bullet\, v\to v\}} \theta'$:

$\theta = \bullet\ \theta_1$, $\theta' = \theta'_1$ for some $\theta_1, \theta'_1 \in T_{\Delta\cup\{\circ,\bullet\}}$ with $\theta_1 \Rightarrow^*_{\{\bullet\, v\to v\}} \theta'_1$:

$$\begin{aligned}
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ (\bullet\ \theta_1)\ z_1 \cdots z_s)|_{\bullet-\circ} - |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta'_1\ z_1 \cdots z_s)|_{\bullet-\circ} \\
&\quad = \text{(by } \Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}} \text{ and properties of } |\,.\,|_{\bullet-\circ}) \\
&1 + |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta_1\ z_1 \cdots z_s)|_{\bullet-\circ} - |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta'_1\ z_1 \cdots z_s)|_{\bullet-\circ} \\
&\quad \leq \text{(by induction hypothesis (a) for } \theta_1 \Rightarrow^*_{\{\bullet\, v\to v\}} \theta'_1) \\
&1 + |\theta_1|_\bullet - |\theta'_1|_\bullet \\
&\quad = \text{(by properties of } |\,.\,|_\bullet) \\
&|\bullet\ \theta_1|_\bullet - |\theta'_1|_\bullet
\end{aligned}$$

$\theta = \circ\ \theta_1$, $\theta' = \circ\ \theta'_1$ for some $\theta_1, \theta'_1 \in T_{\Delta\cup\{\circ,\bullet\}}$ with $\theta_1 \Rightarrow^*_{\{\bullet\, v\to v\}} \theta'_1$:
straightforward, using that $rhs_{M_2^{\circ\bullet\to\circ\bullet},g,\circ} = \circ\ (g\ v_1\ z_1 \cdots z_s)$ and applying induction hypothesis (a) for $\theta_1 \Rightarrow^*_{\{\bullet\, v\to v\}} \theta'_1$.

$\theta = \bullet\ \theta_1$, $\theta' = \bullet\ \theta'_1$ for some $\theta_1, \theta'_1 \in T_{\Delta\cup\{\circ,\bullet\}}$ with $\theta_1 \Rightarrow^*_{\{\bullet\, v\to v\}} \theta'_1$:
analogous to the previous case.

$\theta = \delta\ \theta_1 \cdots \theta_q$, $\theta' = \delta\ \theta'_1 \cdots \theta'_q$ for some $\delta \in \Delta^{(q)}$ and $\theta_1, \ldots, \theta_q, \theta'_1, \ldots, \theta'_q \in T_{\Delta\cup\{\circ,\bullet\}}$ with $\theta_j \Rightarrow^*_{\{\bullet\, v\to v\}} \theta'_j$ for every $j \in [q]$:

$$\begin{aligned}
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ (\delta\ \theta_1 \cdots \theta_q)\ z_1 \cdots z_s)|_{\bullet-\circ} \\
&-|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ (\delta\ \theta'_1 \cdots \theta'_q)\ z_1 \cdots z_s)|_{\bullet-\circ} \\
&\quad = \text{(by (twice) } \Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}\text{, using that } rhs_{M_2^{\circ\bullet\to\circ\bullet},g,\delta} = rhs_{M_2,g,\delta}) \\
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, rhs_{M_2,g,\delta}[v_j \leftarrow \theta_j \mid j \in [q]])|_{\bullet-\circ} \\
&-|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, rhs_{M_2,g,\delta}[v_j \leftarrow \theta'_j \mid j \in [q]])|_{\bullet-\circ} \\
&\quad \leq \text{(by induction hypothesis (b) with } \psi = rhs_{M_2,g,\delta}\text{, using that } M_2 \text{ is context-} \\
&\qquad \text{linear or } M_2 \text{ is basic, in which case } rhs_{M_2,g,\delta} \text{ contains no nested calls)} \\
&\sum_{j\in[q]} |rhs_{M_2,g,\delta}|_{v_j} * (|\theta_j|_\bullet - |\theta'_j|_\bullet) \\
&\quad \leq \text{(by } |rhs_{M_2,g,\delta}|_{v_j} \leq 1 \text{ for every } j \in [q] \text{ (since } M_2 \text{ is recursion-linear),}
\end{aligned}$$

using that $|\theta_j|_\bullet - |\theta'_j|_\bullet \geq 0$ due to $\theta_j \Rightarrow^*_{\{\bullet\, v \to v\}} \theta'_j$ for every $j \in [q]$)

$\sum_{j\in[q]} (|\theta_j|_\bullet - |\theta'_j|_\bullet)$

$=$ (by properties of $|\,.\,|_\bullet$)

$|\delta\ \theta_1 \cdots \theta_q|_\bullet - |\delta\ \theta'_1 \cdots \theta'_q|_\bullet$

(b) $\Leftarrow$ (a) : for fixed $\theta'_1, \ldots, \theta'_q \in T_{\Delta\cup\{\circ,\bullet\}}$ with the associated precondition, for fixed $s \in \mathbb{N}$ by structural induction on $\psi \in RHS(G, \Omega, V_q, Z_s)$. The cases $\psi \in Z_s$ and $lab(\psi, \varepsilon) \in \Omega$ are straightforward. The remaining case is proved as follows.

$\underline{\psi = g\ v_{j'}\ \psi_1 \cdots \psi_{s'}}$ for some $g \in G^{(s'+1)}$, $v_{j'} \in V_q$, $\psi_1, \ldots, \psi_{s'} \in RHS(G, \Omega, V_q, Z_s)$:

$|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, (g\ v_{j'}\ \psi_1 \cdots \psi_{s'})[v_j \leftarrow \theta_j \mid j \in [q]])|_{\bullet-\circ}$
$-|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, (g\ v_{j'}\ \psi_1 \cdots \psi_{s'})[v_j \leftarrow \theta'_j \mid j \in [q]])|_{\bullet-\circ}$

$=$ (by substitution and Lemma A.1 for the mtt $M_2^{\circ\bullet\to\circ\bullet}$, twice)

$|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta_{j'}\ z_1 \cdots z_{s'})|_{\bullet-\circ} - |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta'_{j'}\ z_1 \cdots z_{s'})|_{\bullet-\circ}$
$+ \sum_{l'\in[s']} |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta_{j'}\ z_1 \cdots z_{s'})|_{z_{l'}} * |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \psi_{l'}[v_j \leftarrow \theta_j \mid j \in [q]])|_{\bullet-\circ}$
$- \sum_{l'\in[s']} |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta'_{j'}\ z_1 \cdots z_{s'})|_{z_{l'}} * |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \psi_{l'}[v_j \leftarrow \theta'_j \mid j \in [q]])|_{\bullet-\circ}$

$\leq$ (by induction hypothesis (a) for $\theta_{j'} \Rightarrow^*_{\{\bullet\, v \to v\}} \theta'_{j'}$ and Lemma A.2(2))

$|\theta_{j'}|_\bullet - |\theta'_{j'}|_\bullet + \sum_{l'\in[s']} |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta_{j'}\ z_1 \cdots z_{s'})|_{z_{l'}}$
$* (|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \psi_{l'}[v_j \leftarrow \theta_j \mid j \in [q]])|_{\bullet-\circ}$
$-|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \psi_{l'}[v_j \leftarrow \theta'_j \mid j \in [q]])|_{\bullet-\circ})$

$\leq$ (by the induction hypotheses for the $\psi_1, \ldots, \psi_{s'}$, using that if $\psi = g\ v_{j'}\ \psi_1 \cdots \psi_{s'}$ contains no nested calls, then neither does any of the $\psi_1, \ldots, \psi_{s'}$)

$|\theta_{j'}|_\bullet - |\theta'_{j'}|_\bullet + \sum_{l'\in[s']} |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta_{j'}\ z_1 \cdots z_{s'})|_{z_{l'}} * \sum_{j\in[q]} |\psi_{l'}|_{v_j} * (|\theta_j|_\bullet - |\theta'_j|_\bullet)$

$\leq$ (by sum manipulation and $|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta_{j'}\ z_1 \cdots z_{s'})|_{z_{l'}} * |\psi_{l'}|_{v_j} \leq |\psi_{l'}|_{v_j}$ for every $l' \in [s']$ and $j \in [q]$ (see below), using that $|\theta_j|_\bullet - |\theta'_j|_\bullet \geq 0$ due to $\theta_j \Rightarrow^*_{\{\bullet\, v \to v\}} \theta'_j$)

$|\theta_{j'}|_\bullet - |\theta'_{j'}|_\bullet + \sum_{l'\in[s']} \sum_{j\in[q]} |\psi_{l'}|_{v_j} * (|\theta_j|_\bullet - |\theta'_j|_\bullet)$

$=$ (by properties of $|\,.\,|_{v_j}$ and sum manipulation)

$\sum_{j\in[q]} |g\ v_{j'}\ \psi_1 \cdots \psi_{s'}|_{v_j} * (|\theta_j|_\bullet - |\theta'_j|_\bullet)$

The statement, used above, that for every $l' \in [s']$ and $j \in [q]$,

$$|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta_{j'}\ z_1 \cdots z_{s'})|_{z_{l'}} * |\psi_{l'}|_{v_j} \leq |\psi_{l'}|_{v_j}\,,$$

follows from the precondition that M_2, and hence also $M_2^{\circ\bullet\to\circ\bullet}$, is context-linear or $\psi = g\ v_{j'}\ \psi_1 \cdots \psi_{s'}$ contains no nested calls, since in the former case,

$$|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta_{j'}\ z_1 \cdots z_{s'})|_{z_{l'}} \leq 1$$

by Lemma 2.5 for $M_2^{\circ\bullet\to\circ\bullet}$, while in the latter case $|\psi_{l'}|_{v_j} = 0$. □

Proof of Lemma 4.33
Let $M_1^{\to\circ\bullet} = (F, \Sigma, \Delta \cup \{\circ, \bullet\}, e_{M_1^{\to\circ\bullet}}, R_1^{\to\circ\bullet})$ and $M_1^{\to\circ\bullet'} = (F, \Sigma, \Delta \cup \{\circ, \bullet\}, e_{M_1^{\to\circ\bullet'}}, R_1^{\to\circ\bullet'})$ be in $\mathcal{M}_1^{\to\circ\bullet}$, where $M_1^{\to\circ\bullet} \rightsquigarrow_{\mathcal{E}_{M_1,M_2}} M_1^{\to\circ\bullet'}$. We prove the following two statements (a) and (b):

(a) for every $t \in T_\Sigma$:
For every $f \in F^{(r+1)}$ and $\theta_1, \ldots, \theta_r, \theta'_1, \ldots, \theta'_r \in T_{\Delta\cup\{\circ,\bullet\}}$,
if for every $k \in [r]$, $g \in G^{(s+1)}$, and $\eta_1, \ldots, \eta_s, \eta'_1, \ldots, \eta'_s \in T_{G\cup\Omega\cup\Delta\cup\{\circ,\bullet\}}$ from $|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \eta_l)|_{\circ-\bullet} \leq |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \eta'_l)|_{\circ-\bullet}$ for every $l \in [s]$ follows that

$$|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta_k\ \eta_1 \cdots \eta_s)|_{\circ-\bullet} \leq |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta'_k\ \eta'_1 \cdots \eta'_s)|_{\circ-\bullet},$$

then for every $g \in G^{(s+1)}$ and $\eta_1, \ldots, \eta_s, \eta'_1, \ldots, \eta'_s \in T_{G\cup\Omega\cup\Delta\cup\{\circ,\bullet\}}$ with $|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \eta_l)|_{\circ-\bullet} \leq |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \eta'_l)|_{\circ-\bullet}$ for every $l \in [s]$:

$$\begin{aligned} &|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet}}, f\ t\ \theta_1 \cdots \theta_r)\ \eta_1 \cdots \eta_s)|_{\circ-\bullet} \\ \leq\ &|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, f\ t\ \theta'_1 \cdots \theta'_r)\ \eta'_1 \cdots \eta'_s)|_{\circ-\bullet}. \end{aligned}$$

(b) for every $p \in \mathbb{N}$ and $t_1, \ldots, t_p \in T_\Sigma$:
For every $r \in \mathbb{N}$ and $\theta_1, \ldots, \theta_r, \theta'_1, \ldots, \theta'_r \in T_{\Delta\cup\{\circ,\bullet\}}$,
if for every $k \in [r]$, $g \in G^{(s+1)}$, and $\eta_1, \ldots, \eta_s, \eta'_1, \ldots, \eta'_s \in T_{G\cup\Omega\cup\Delta\cup\{\circ,\bullet\}}$ from $|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \eta_l)|_{\circ-\bullet} \leq |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \eta'_l)|_{\circ-\bullet}$ for every $l \in [s]$ follows that

$$|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta_k\ \eta_1 \cdots \eta_s)|_{\circ-\bullet} \leq |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta'_k\ \eta'_1 \cdots \eta'_s)|_{\circ-\bullet},$$

then the following statements (i) and (ii) hold:

(i) for every $\phi \in RHS(F, \Delta \cup \{\circ, \bullet\}, U_p, Y_r)$:
For every $\phi' \in RHS(F, \Delta\cup\{\circ, \bullet\}, U_p, Y_r)$ with $\phi \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \phi'$, for every $g \in G^{(s+1)}$ and $\eta_1, \ldots, \eta_s, \eta'_1, \ldots, \eta'_s \in T_{G\cup\Omega\cup\Delta\cup\{\circ,\bullet\}}$ with $|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \eta_l)|_{\circ-\bullet}$ $\leq |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \eta'_l)|_{\circ-\bullet}$ for every $l \in [s]$:

$$\begin{aligned} &|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet}}, \phi[u_i \leftarrow t_i \mid i \in [p],\\ &\qquad\qquad y_k \leftarrow \theta_k \mid k \in [r]])\ \eta_1 \cdots \eta_s)|_{\circ-\bullet} \\ \leq\ &|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, \phi'[u_i \leftarrow t_i \mid i \in [p],\\ &\qquad\qquad y_k \leftarrow \theta'_k \mid k \in [r]])\ \eta'_1 \cdots \eta'_s)|_{\circ-\bullet}. \end{aligned}$$

(ii) for every $q \in \mathbb{N}$ and $\phi_1, \ldots, \phi_q \in RHS(F, \Delta \cup \{\circ, \bullet\}, U_p, Y_r)$:
For every $\phi'_1, \ldots, \phi'_q \in RHS(F, \Delta\cup\{\circ, \bullet\}, U_p, Y_r)$ with $\phi_j \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \phi'_j$ for every $j \in [q]$, for every $s \in \mathbb{N}$ and $\eta_1, \ldots, \eta_s, \eta'_1, \ldots, \eta'_s \in T_{G\cup\Omega\cup\Delta\cup\{\circ,\bullet\}}$ with $|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \eta_l)|_{\circ-\bullet} \leq |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \eta'_l)|_{\circ-\bullet}$ for every $l \in [s]$, for every $\psi \in RHS(G, \Omega, V_q, Z_s)$:

$$\begin{aligned} &|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \psi[v_j \leftarrow nf(\Rightarrow_{R_1^{\to\circ\bullet}}, \phi_j[u_i \leftarrow t_i \mid i \in [p],\\ &\qquad\qquad y_k \leftarrow \theta_k \mid k \in [r]]) \mid j \in [q],\\ &\qquad\qquad z_l \leftarrow \eta_l \mid l \in [s]])|_{\circ-\bullet} \\ \leq\ &|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \psi[v_j \leftarrow nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, \phi'_j[u_i \leftarrow t_i \mid i \in [p],\\ &\qquad\qquad y_k \leftarrow \theta'_k \mid k \in [r]]) \mid j \in [q],\\ &\qquad\qquad z_l \leftarrow \eta'_l \mid l \in [s]])|_{\circ-\bullet}. \end{aligned}$$

by simultaneous induction, where the two statements (i) and (ii) nested inside (b) are also proved by simultaneous induction. Lemma 4.33 then follows from statement (b)(ii) with $p = 1$, $r = 0$, $q = 1$, $\phi_1 = e_{M_1^{\to\circ\bullet}}$, $\phi_1' = e_{M_1^{\to\circ\bullet'}}$, $s = 0$, and $\psi = e_{M_2}$.

(a) $\Leftarrow$ (b) : Assume $t = \sigma\ t_1 \cdots t_p$ for $\sigma \in \Sigma^{(p)}$ and $t_1, \ldots, t_p \in T_\Sigma$. We calculate:

$$
\begin{aligned}
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet}}, f\ (\sigma\ t_1 \cdots t_p)\ \theta_1 \cdots \theta_r)\ \eta_1 \cdots \eta_s)|_{\circ-\bullet} \\
&\quad = \text{(by } \Rightarrow_{R_1^{\to\circ\bullet}}\text{)} \\
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet}}, rhs_{M_1^{\to\circ\bullet},f,\sigma}[u_i \leftarrow t_i \mid i \in [p], \\
&\qquad\qquad y_k \leftarrow \theta_k \mid k \in [r]])\ \eta_1 \cdots \eta_s)|_{\circ-\bullet} \\
&\quad \leq \text{(by statement (i) of induction hypothesis (b)} \\
&\qquad \text{with } \phi = rhs_{M_1^{\to\circ\bullet},f,\sigma} \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} rhs_{M_1^{\to\circ\bullet'},f,\sigma} = \phi') \\
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, rhs_{M_1^{\to\circ\bullet'},f,\sigma}[u_i \leftarrow t_i \mid i \in [p], \\
&\qquad\qquad y_k \leftarrow \theta_k' \mid k \in [r]])\ \eta_1' \cdots \eta_s')|_{\circ-\bullet} \\
&\quad = \text{(by } \Rightarrow_{R_1^{\to\circ\bullet'}}\text{)} \\
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, f\ (\sigma\ t_1 \cdots t_p)\ \theta_1' \cdots \theta_r')\ \eta_1' \cdots \eta_s')|_{\circ-\bullet}
\end{aligned}
$$

(b) $\Leftarrow$ (a) : for fixed $r \in \mathbb{N}$ and $\theta_1, \ldots, \theta_r, \theta_1', \ldots, \theta_r' \in T_{\Delta\cup\{\circ,\bullet\}}$ with the associated precondition, by simultaneous induction of (i) and (ii):

(i) $\Leftarrow$ (ii) : by case analysis on $\phi \in RHS(F, \Delta \cup \{\circ, \bullet\}, U_p, Y_r)$ and $\phi \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \phi'$:

$\underline{\phi = \phi' = y_k \in Y_r}$:

$$
\begin{aligned}
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet}}, y_k[u_i \leftarrow t_i \mid i \in [p], \\
&\qquad\qquad y_k \leftarrow \theta_k \mid k \in [r]])\ \eta_1 \cdots \eta_s)|_{\circ-\bullet} \\
&\quad = \text{(by substitution and since } \theta_k \in T_{\Delta\cup\{\circ,\bullet\}} \text{ is a normal form w.r.t. } \Rightarrow_{R_1^{\to\circ\bullet}}\text{)} \\
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta_k\ \eta_1 \cdots \eta_s)|_{\circ-\bullet} \\
&\quad \leq \text{(by the preconditions for } \theta_k, \theta_k' \text{ and for } \eta_1, \ldots, \eta_s, \eta_1', \ldots, \eta_s'\text{)} \\
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ \theta_k'\ \eta_1' \cdots \eta_s')|_{\circ-\bullet} \\
&\quad = \text{(by substitution and since } \theta_k' \in T_{\Delta\cup\{\circ,\bullet\}} \text{ is a normal form w.r.t. } \Rightarrow_{R_1^{\to\circ\bullet'}}\text{)} \\
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, y_k[u_i \leftarrow t_i \mid i \in [p], \\
&\qquad\qquad y_k \leftarrow \theta_k' \mid k \in [r]])\ \eta_1' \cdots \eta_s')|_{\circ-\bullet}
\end{aligned}
$$

$\underline{\phi = \circ\ \phi_1,\ \phi' = \circ\ \phi_1'}$ for $\phi_1, \phi_1' \in RHS(F, \Delta \cup \{\circ, \bullet\}, U_p, Y_r)$ with $\phi_1 \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \phi_1'$: straightforward, using that $rhs_{M_2^{\circ\bullet\to\circ\bullet},g,\circ} = \circ\ (g\ v_1\ z_1 \cdots z_s)$ and applying induction hypothesis (i) for $\phi_1 \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \phi_1'$.

$\underline{\phi = \bullet\ \phi_1,\ \phi' = \bullet\ \phi_1'}$ for $\phi_1, \phi_1' \in RHS(F, \Delta \cup \{\circ, \bullet\}, U_p, Y_r)$ with $\phi_1 \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \phi_1'$: analogous to the previous case.

$\underline{\phi = \delta\ \phi_1 \cdots \phi_q,\ \phi' = \delta\ \phi_1' \cdots \phi_q'}$ for some $\delta \in \Delta^{(q)}$ and $\phi_1, \ldots, \phi_q, \phi_1', \ldots, \phi_q' \in RHS(F, \Delta \cup \{\circ, \bullet\}, U_p, Y_r)$ with $\phi_j \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \phi_j'$ for every $j \in [q]$: straightforward, using that $rhs_{M_2^{\circ\bullet\to\circ\bullet},g,\delta} = rhs_{M_2,g,\delta}$ and applying induction hypothesis (ii) with $\psi = rhs_{M_2,g,\delta}$.

$\phi = f\ u_{i'}\ \phi_1 \cdots \phi_{r'}$, $\phi' = f\ u_{i'}\ \phi'_1 \cdots \phi'_{r'}$ for $f \in F^{(r'+1)}$, $u_{i'} \in U_p$, $\phi_1, \ldots, \phi_{r'}$, $\phi'_1, \ldots, \phi'_{r'} \in RHS(F, \Delta \cup \{\circ, \bullet\}, U_p, Y_r)$ with $\phi_k \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \phi'_k$ for every $k \in [r']$: the induction hypotheses (i) for $\phi_1 \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \phi'_1$, …, $\phi_{r'} \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \phi'_{r'}$ establish the precondition necessary to apply induction hypothesis (a) for $t_{i'}$, which suffices.

$\phi = \bullet\ (\circ\ \phi_1)$, $\phi' = \phi_1$ for some $\phi_1 \in RHS(F, \Delta \cup \{\circ, \bullet\}, U_p, Y_r)$:
straightforward, using that $rhs_{M_2^{\circ\bullet\to\circ\bullet},g,\bullet} = \bullet\ (g\ v_1\ z_1 \cdots z_s)$ and $rhs_{M_2^{\circ\bullet\to\circ\bullet},g,\circ} = \circ\ (g\ v_1\ z_1 \cdots z_s)$, and applying induction hypothesis (i) for $\phi_1 \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \phi_1$.

$\phi = \circ\ (\bullet\ \phi_1)$, $\phi' = \phi_1$ for some $\phi_1 \in RHS(F, \Delta \cup \{\circ, \bullet\}, U_p, Y_r)$:
analogous to the previous case.

$\phi = \bullet\ (\delta\ \phi_1 \cdots \phi_q)$, $\phi' = \delta\ \phi_1 \cdots (\bullet\ \phi_j) \cdots \phi_q$ for some $\delta \in \Delta^{(q)}$, $\phi_1, \ldots, \phi_q \in RHS(F, \Delta \cup \{\circ, \bullet\}, U_p, Y_r)$, and $j \in [q]$ with $\downarrow_{\delta,j} \in \mathcal{E}_{M_1,M_2}$:

$$
\begin{aligned}
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet}}, (\bullet\ (\delta\ \phi_1 \cdots \phi_q))[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta_k \mid k \in [r]])\ \eta_1 \cdots \eta_s)|_{\circ-\bullet}\\
&\quad = \text{(by } \Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}\text{, using that } rhs_{M_2^{\circ\bullet\to\circ\bullet},g,\bullet} = \bullet\ (g\ v_1\ z_1 \cdots z_s)\text{, and by}\\
&\qquad \text{properties of } |\,\cdot\,|_{\circ-\bullet})\\
&-1 + |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet}}, (\delta\ \phi_1 \cdots \phi_q)[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta_k \mid k \in [r]])\ \eta_1 \cdots \eta_s)|_{\circ-\bullet}\\
&\quad \le \text{(by induction hypothesis (i) for } \delta\ \phi_1 \cdots \phi_q \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \delta\ \phi_1 \cdots \phi_q)\\
&-1 + |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, (\delta\ \phi_1 \cdots \phi_q)[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta'_k \mid k \in [r]])\ \eta'_1 \cdots \eta'_s)|_{\circ-\bullet}\\
&\quad \le \text{(by Lemma A.1 for the mtt } M_2^{\circ\bullet\to\circ\bullet} \text{ and Lemma 4.32(1a))}\\
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ (\delta\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, \phi_1[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta'_k \mid k \in [r]])\\
&\qquad \cdots\\
&\qquad (\bullet\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, \phi_j[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta'_k \mid k \in [r]]))\\
&\qquad \cdots\\
&\qquad nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, \phi_q[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta'_k \mid k \in [r]]))\ z_1 \cdots z_s)|_{\circ-\bullet}\\
&+ \sum_{l \in [s]} |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, (\delta\ \phi_1 \cdots \phi_q)[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta'_k \mid k \in [r]])\ z_1 \cdots z_s)|_{z_l}\\
&\qquad * |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \eta'_l)|_{\circ-\bullet}\\
&\quad = \text{(by Lemma A.1 for the mtt } M_2^{\circ\bullet\to\circ\bullet} \text{ and Lemma A.2(2))}\\
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, (\delta\ \phi_1 \cdots (\bullet\ \phi_j) \cdots \phi_q)[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta'_k \mid k \in [r]])\\
&\qquad \eta'_1 \cdots \eta'_s)|_{\circ-\bullet}
\end{aligned}
$$

$\phi = \delta\ \phi_1 \cdots (\bullet\ \phi_j) \cdots \phi_q$, $\phi' = \bullet\ (\delta\ \phi_1 \cdots \phi_q)$ for some $\delta \in \Delta^{(q)}$, $\phi_1, \ldots, \phi_q \in RHS(F, \Delta \cup \{\circ, \bullet\}, U_p, Y_r)$, and $j \in [q]$ with $\uparrow_{\delta,j} \in \mathcal{E}_{M_1,M_2}$:

$$
\begin{aligned}
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet}}, (\delta\ \phi_1 \cdots (\bullet\ \phi_j) \cdots \phi_q)[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta_k \mid k \in [r]])\\
&\qquad \eta_1 \cdots \eta_s)|_{\circ-\bullet}
\end{aligned}
$$

$\leq$ (straightforward, using that $rhs_{M_2^{\circ\bullet\to\circ\bullet},g,\delta} = rhs_{M_2,g,\delta}$ and applying induction hypothesis (ii) with $\psi = rhs_{M_2,g,\delta}$, taking into account that $\phi_1 \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \phi_1, \ldots, \bullet\ \phi_j \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \bullet\ \phi_j, \ldots, \phi_q \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \phi_q$)

$$|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, (\delta\ \phi_1 \cdots (\bullet\ \phi_j) \cdots \phi_q)[u_i \leftarrow t_i \mid i \in [p], y_k \leftarrow \theta'_k \mid k \in [r]])\ \eta'_1 \cdots \eta'_s)|_{\circ-\bullet}$$

$\leq$ (by Lemma A.1 for the mtt $M_2^{\circ\bullet\to\circ\bullet}$ and Lemma 4.32(1b))

$$\begin{aligned}-1 + |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ (\delta\ &nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, \phi_1[u_i \leftarrow t_i \mid i \in [p], y_k \leftarrow \theta'_k \mid k \in [r]])\\ &\cdots\\ &nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, \phi_q[u_i \leftarrow t_i \mid i \in [p], y_k \leftarrow \theta'_k \mid k \in [r]]))\ z_1 \cdots z_s)|_{\circ-\bullet}\\ + \sum_{l\in[s]} |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ &nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, (\delta\ \phi_1 \cdots (\bullet\ \phi_j) \cdots \phi_q)[u_i \leftarrow t_i \mid i \in [p], y_k \leftarrow \theta'_k \mid k \in [r]])\\ &z_1 \cdots z_s)|_{z_l} * |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \eta'_l)|_{\circ-\bullet}\end{aligned}$$

$=$ (by Lemma A.1 for the mtt $M_2^{\circ\bullet\to\circ\bullet}$, Lemma A.2(2), by $\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}$, using that $rhs_{M_2^{\circ\bullet\to\circ\bullet},g,\bullet} = \bullet\ (g\ v_1\ z_1 \cdots z_s)$, and by properties of $|\,.\,|_{\circ-\bullet}$)

$$|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, (\bullet\ (\delta\ \phi_1 \cdots \phi_q))[u_i \leftarrow t_i \mid i \in [p], y_k \leftarrow \theta'_k \mid k \in [r]])\ \eta'_1 \cdots \eta'_s)|_{\circ-\bullet}$$

$\phi = f\ u_{i'}\ \phi_1 \cdots (\bullet\ \phi_{k'}) \cdots \phi_{r'}$, $\phi' = \bullet\ (f\ u_{i'}\ \phi_1 \cdots \phi_{r'})$ for some $f \in F^{(r'+1)}$, $u_{i'} \in U_p$, $\phi_1, \ldots, \phi_{r'} \in RHS(F, \Delta \cup \{\circ, \bullet\}, U_p, Y_r)$, and $k' \in [r']$ with $\uparrow_{f,k'} \in \mathcal{E}_{M_1,M_2}$:

$$|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet}}, (f\ u_{i'}\ \phi_1 \cdots (\bullet\ \phi_{k'}) \cdots \phi_{r'})[u_i \leftarrow t_i \mid i \in [p], y_k \leftarrow \theta_k \mid k \in [r]])\ \eta_1 \cdots \eta_s)|_{\circ-\bullet}$$

$\leq$ (by substitution, properties of nf, and induction hypothesis (a) for $t_{i'}$, using the induction hypotheses (i) for $\phi_1 \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \phi_1, \ldots, \bullet\ \phi_{k'} \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \bullet\ \phi_{k'}, \ldots, \phi_{r'} \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} \phi_{r'}$ to establish the necessary precondition)

$$\begin{aligned}|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, &g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, f\ t_{i'}\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, \phi_1[u_i \leftarrow t_i \mid i \in [p], y_k \leftarrow \theta'_k \mid k \in [r]])\\ &\cdots\\ &nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, (\bullet\ \phi_{k'})[u_i \leftarrow t_i \mid i \in [p], y_k \leftarrow \theta'_k \mid k \in [r]])\\ &\cdots\\ &nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, \phi_{r'}[u_i \leftarrow t_i \mid i \in [p], y_k \leftarrow \theta'_k \mid k \in [r]]))\ \eta'_1 \cdots \eta'_s)|_{\circ-\bullet}\end{aligned}$$

$\leq$ (by Lemma A.1 for the mtt $M_2^{\circ\bullet\to\circ\bullet}$, Lemma 4.32(2b) for the mtt $M_1^{\to\circ\bullet'}$, by substitution and properties of nf)

$$
\begin{aligned}
&-1 + |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}},\\
&\qquad g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, (f\ u_{i'}\ \phi_1 \cdots \phi_{r'})[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta'_k \mid k \in [r]])\ z_1 \cdots z_s)|_{\circ-\bullet}\\
&+ \sum_{l \in [s]} |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}},\\
&\qquad g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, (f\ u_{i'}\ \phi_1 \cdots (\bullet\ \phi_{k'}) \cdots \phi_{r'})[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta'_k \mid k \in [r]])\\
&\qquad z_1 \cdots z_s)|_{zl} * |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \eta'_l)|_{\circ-\bullet}\\
&\quad = \text{(by Lemma A.1 for the mtt } M_2^{\circ\bullet\to\circ\bullet}\text{, Lemma A.2(2), by } \Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}\text{, using}\\
&\qquad \text{that } rhs_{M_2^{\circ\bullet\to\circ\bullet},g,\bullet} = \bullet\ (g\ v_1\ z_1 \cdots z_s)\text{, and by properties of } |\blacksquare|_{\circ-\bullet})\\
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}},\\
&\quad g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, (\bullet\ (f\ u_{i'}\ \phi_1 \cdots \phi_{r'}))[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta'_k \mid k \in [r]])\ \eta'_1 \cdots \eta'_s)|_{\circ-\bullet}
\end{aligned}
$$

$\underline{\phi = \bullet\ (f\ u_{i'}\ \phi_1 \cdots \phi_{r'}),\ \phi' = f\ u_{i'}\ \phi_1 \cdots (\bullet\ \phi_{k'}) \cdots \phi_{r'}}$ for some $f \in F^{(r'+1)}$, $u_{i'} \in U_p$, $\phi_1, \ldots, \phi_{r'} \in RHS(F, \Delta \cup \{\circ, \bullet\}, U_p, Y_r)$, and $k' \in [r']$ with $\downarrow_{f,k'} \in \mathcal{E}_{M_1,M_2}$:

$$
\begin{aligned}
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet}}, (\bullet\ (f\ u_{i'}\ \phi_1 \cdots \phi_{r'}))[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta_k \mid k \in [r]])\ \eta_1 \cdots \eta_s)|_{\circ-\bullet}\\
&\quad = \text{(by } \Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}\text{, using that } rhs_{M_2^{\circ\bullet\to\circ\bullet},g,\bullet} = \bullet\ (g\ v_1\ z_1 \cdots z_s)\text{, and by}\\
&\qquad \text{properties of } |\blacksquare|_{\circ-\bullet})\\
&-1 + |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet}}, (f\ u_{i'}\ \phi_1 \cdots \phi_{r'})[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta_k \mid k \in [r]])\\
&\qquad \eta_1 \cdots \eta_s)|_{\circ-\bullet}\\
&\quad \leq \text{(by induction hypothesis (i) for } f\ u_{i'}\ \phi_1 \cdots \phi_{r'} \Rightarrow^?_{\mathcal{E}_{M_1,M_2}} f\ u_{i'}\ \phi_1 \cdots \phi_{r'})\\
&-1 + |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, (f\ u_{i'}\ \phi_1 \cdots \phi_{r'})[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta'_k \mid k \in [r]])\\
&\qquad \eta'_1 \cdots \eta'_s)|_{\circ-\bullet}\\
&\quad \leq \text{(by Lemma A.1 for the mtt } M_2^{\circ\bullet\to\circ\bullet}\text{, substitution, properties of } nf\text{, and}\\
&\qquad \text{Lemma 4.32(2a) for the mtt } M_1^{\to\circ\bullet'})\\
&|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, f\ t_{i'}\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, \phi_1[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta'_k \mid k \in [r]])\\
&\qquad\qquad \cdots\\
&\qquad\qquad (\bullet\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, \phi_{k'}[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad\qquad y_k \leftarrow \theta'_k \mid k \in [r]]))\\
&\qquad\qquad \cdots\\
&\qquad\qquad nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, \phi_{r'}[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad\qquad y_k \leftarrow \theta'_k \mid k \in [r]]))\\
&\qquad z_1 \cdots z_s)|_{\circ-\bullet}\\
&+ \sum_{l \in [s]} |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}}, (f\ u_{i'}\ \phi_1 \cdots \phi_{r'})[u_i \leftarrow t_i \mid i \in [p],\\
&\qquad\qquad y_k \leftarrow \theta'_k \mid k \in [r]])\\
&\qquad z_1 \cdots z_s)|_{zl} * |nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}}, \eta'_l)|_{\circ-\bullet}\\
&\quad = \text{(by Lemma A.1 for the mtt } M_2^{\circ\bullet\to\circ\bullet}\text{, substitution, properties of } nf\text{, and}\\
&\qquad \text{Lemma A.2(2))}
\end{aligned}
$$

$$|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}},\ g\ nf(\Rightarrow_{R_1^{\to\circ\bullet'}},(f\ u_{i'}\ \phi_1\cdots(\bullet\ \phi_{k'})\cdots\phi_{r'})[u_i\leftarrow t_i\mid i\in[p],\ y_k\leftarrow\theta'_k\mid k\in[r]])\ \eta'_1\cdots\eta'_s)|_{\circ-\bullet}$$

(ii) $\Leftarrow$ (i) : for fixed $\phi'_1,\ldots,\phi'_q\in RHS(F,\Delta\cup\{\circ,\bullet\},U_p,Y_r)$ with $\phi_j\Rightarrow^?_{\mathcal{E}_{M_1,M_2}}\phi'_j$ for every $j\in[q]$, for fixed $s\in\mathbb{N}$ and $\eta_1,\ldots,\eta_s,\eta'_1,\ldots,\eta'_s\in T_{G\cup\Omega\cup\Delta\cup\{\circ,\bullet\}}$ with $|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}},\eta_l)|_{\circ-\bullet}\leq|nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}},\eta'_l)|_{\circ-\bullet}$ for every $l\in[s]$, by structural induction on $\psi\in RHS(G,\Omega,V_q,Z_s)$. The validity in the case $\psi\in Z_s$ follows from the precondition on the $\eta_1,\ldots,\eta_s,\eta'_1,\ldots,\eta'_s$. The case $lab(\psi,\varepsilon)\in\Omega$ is straightforward. In the remaining case, $\psi=g\ v_j\ \psi_1\cdots\psi_{s'}$ for some $g\in G^{(s'+1)}$, $v_j\in V_q$, and $\psi_1,\ldots,\psi_{s'}\in RHS(G,\Omega,V_q,Z_s)$, the induction hypotheses for the $\psi_1,\ldots,\psi_{s'}$ establish the precondition necessary to apply induction hypothesis (i) for $\phi_j\Rightarrow^?_{\mathcal{E}_{M_1,M_2}}\phi'_j$, which suffices. □

Proof of Lemma 4.36

For atmost M_2, we take the statements (a) and (b) and the simultaneous induction establishing their validity in the proof of Lemma 4.31. We replace all occurrences of $\Rightarrow^*_{\{\bullet\ v\to v\}}$, $|\,.\,|_\bullet$, and $|\,.\,|_{\bullet-\circ}$ by $\Rightarrow^*_{\{\circ\ v\to v\}}$, $|\,.\,|_\circ$, and $|\,.\,|_\circ$, respectively, and all occurrences of $\bullet\ \theta_1$ by $\circ\ \theta_1$ in the first case of **(a) $\Leftarrow$ (b)**. This yields again a valid proof. We obtain

$$|\tau_{M_2^{\circ\bullet\to\circ\bullet}}(\theta)|_\circ-|\tau_{M_2^{\circ\bullet\to\circ\bullet}}(\theta[\circ\twoheadleftarrow id])|_\circ\leq|e_{M_2}|_{v_1}*|\theta|_\circ$$

from the thus adjusted statement (b) with $q=1$, $\theta_1=\theta$, $\theta'_1=\theta[\circ\twoheadleftarrow id]$, $s=0$, and $\psi=e_{M_2}$, taking into account that M_2 is context-linear or M_2 is basic, in which case e_{M_2} contains no nested calls, and that $|\theta[\circ\twoheadleftarrow id]|_\circ=0$. Lemma 4.36 then follows because $|\tau_{M_2^{\circ\bullet\to\circ\bullet}}(\theta[\circ\twoheadleftarrow id])|_\circ=0$, given that the initial expression $e_{M_2}\in RHS(G,\Omega,V_1,\emptyset)$ of $M_2^{\circ\bullet\to\circ\bullet}$ contains no $\circ$-symbols and hence such symbols in $\tau_{M_2^{\circ\bullet\to\circ\bullet}}(\theta[\circ\twoheadleftarrow id])=nf(\Rightarrow_{R_2^{\circ\bullet\to\circ\bullet}},e_{M_2}[v_1\leftarrow\theta[\circ\twoheadleftarrow id]])$ could only originate from right-hand sides of $\circ$-rules in $R_2^{\circ\bullet\to\circ\bullet}$, which however are never applied due to the fact that $\theta[\circ\twoheadleftarrow id]$ contains no $\circ$-symbols. □