Composition of Functions with Accumulating Parameters[†]

Janis Voigtländer[*] Armin Kühnemann

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

Abstract

A class of recursive functions with accumulating parameters can be modeled by macro tree transducers. We present a program transformation technique that can be used to solve the efficiency problems due to creation and consumption of intermediate data structures in compositions of such functions, where classical deforestation techniques fail.

In order to do so, we construct for two given macro tree transducers, under appropriate restrictions, a single macro tree transducer that implements the composition of the two original ones. The imposed restrictions are more liberal than those in the literature on macro tree transducer composition, thus generalizing previous results.

1 Introduction

An important style of writing programs in a functional language is to define new functions by composition of existing ones. Thus, the result of a function application is passed as argument to another function. This modular programming technique of solving an overall problem by combining solutions of partial problems—which might be solved by predefined or self-written functions—simplifies the design and verification of programs, and encourages the reuse of existing programming solutions. Unfortunately, modular programs often lack efficiency compared to other—often less understandable—programs, which solve the same tasks. If the created intermediate results are structured objects—for example lists or trees—their creation and eventual destruction will consume time and memory space. Furthermore, it is possible that, e.g., lists are traversed more often, than would really be necessary for solving the overall problem. Thus, we would like to have program transformation techniques, which enable us to optimize functions written in the modular style, by eliminating intermediate data structures. Several such techniques have been studied in the literature, e.g., the *fold/unfold-technique* from [BD77], its algorithmic instances *supercompilation* [Tur86, SGJ96, SS99] and *deforestation* [Wad90, Chi94], *catamorphism fusion* [Mal89, MFP91] and *shortcut deforestation* [GLP93, Gil96].

In this paper we follow an approach for eliminating intermediate results, which is based on the theory of tree transducers [FV98]. Particularly, we consider *macro tree transducers* [Eng80], which represent a large class of functional programs. Macro tree transducers are extended schemes of primitive recursion, allowing simultaneous definition of several functions and nested function calls in parameter positions. Composition of functions then corresponds to consecutive execution of macro tree transducers. The goal of avoiding intermediate data structures corresponds to the problem of replacing such a composition of macro tree transducers by a single macro tree transducer, which computes the same function. It is well-known that this is not possible in general, because the class of macro

[†]A revised version appeared in *Journal of Functional Programming*, 14(3):317–363, © CUP, 2004.
[*]This author was supported by the DFG under grant KU 1290/2-1.

tree transductions is not closed under composition [EV85]. For subclasses, however, such constructions can be found.

For illustration of the problem of intermediate results and as informal introduction to the concept of macro tree transducers, consider the following example. Assume given a representation of arithmetic terms, built from two variables and the binary operations addition and multiplication, as trees with nullary constructor symbols A and B and binary constructor symbols $+$ and $*$ (this representation corresponds to an algebraic data type in a functional language, e.g., `data Term = A | B | + Term Term | * Term Term` in Haskell-notation). Further, assume given a function pre for computing the prefix-notation of such a term as monadic tree of now unary symbols A, B, $+$, $*$, and a nullary ϵ as end symbol. In most functional languages, one would use a built-in type of lists instead, but for our illustration this representation of lists as monadic trees is more handy. Our techniques are also applicable to, e.g., polymorphic lists, as demonstrated in [KV01] by choosing appropriate constructor symbols. As a macro tree transducer, the function pre can be given by the following rules:

$$\begin{array}{lll}
\text{(i)} & : pre(+(x_1, x_2), y) & \rightarrow +(pre(x_1, pre(x_2, y))) \\
\text{(ii)} & : pre(*(x_1, x_2), y) & \rightarrow *(pre(x_1, pre(x_2, y))) \\
\text{(iii)} & : pre(A, y) & \rightarrow A(y) \\
\text{(iv)} & : pre(B, y) & \rightarrow B(y)
\end{array}$$

The function is defined by pattern matching on its first argument and uses an additional accumulating parameter, which here will be called context parameter. In order to compute the prefix-notation of a term t, the rules (i)–(iv) are used to exhaustively rewrite the initial expression $pre(t, \epsilon)$.

Now, consider the problem of computing, for a given term as above, a sequence of instructions for a stack-machine with two registers and instructions for addition and multiplication (represented as monadic tree, labeled with instructions $LOAD_A$, $LOAD_B$, ADD and MUL). Instead of writing a program for this problem from scratch, we might realize that we can solve the task by reversing the prefix-notation of the given term and replacing labels A, B, $+$ and $*$ by $LOAD_A$, $LOAD_B$, ADD and MUL, respectively. Thus, we define the following function aux:

$$\begin{array}{lll}
\text{(v)} & : aux(A(x), z) & \rightarrow aux(x, LOAD_A(z)) \\
\text{(vi)} & : aux(B(x), z) & \rightarrow aux(x, LOAD_B(z)) \\
\text{(vii)} & : aux(+(x), z) & \rightarrow aux(x, ADD(z)) \\
\text{(viii)} & : aux(*(x), z) & \rightarrow aux(x, MUL(z)) \\
\text{(ix)} & : aux(\epsilon, z) & \rightarrow z
\end{array}$$

On a built-in list type, this auxiliary function might be obtained from fusion of a `map`-function and the standard list `reverse`-function. The instruction sequence for a given term t can now be computed by composing the functions pre and aux, namely by rewriting the composite expression $aux(pre(t, \epsilon), \epsilon)$ with rules (i)–(ix).

This modular solution, however, is inefficient, because it creates and consumes an intermediate result. Depending on the used rewriting strategy, respectively the evaluation order of our functional language, this intermediate data structure might never exist as a whole, but nevertheless, for all of its nodes, memory cells have to be allocated and later deallocated. Also, even with a lazy evaluation strategy, this program performs a superfluous traversal on the intermediate list. To avoid these inefficiencies, we could instead have given the following function definition:

$$
\begin{array}{lll}
\text{(x)} & : ins(+(x_1, x_2), z) & \rightarrow ins(x_2, ins(x_1, ADD(z))) \\
\text{(xi)} & : ins(*(x_1, x_2), z) & \rightarrow ins(x_2, ins(x_1, MUL(z))) \\
\text{(xii)} & : ins(A, z) & \rightarrow LOAD_A(z) \\
\text{(xiii)} & : ins(B, z) & \rightarrow LOAD_B(z)
\end{array}
$$

If now, for a given term t, we use rules (x)–(xiii) on $ins(t, \epsilon)$, we will calculate the same instruction sequence as before with the modular solution, but without creating and traversing the intermediate list, thus requiring less rewriting steps. For the programmer, however, it would probably be more difficult (and, arguably, bad non-modular style) to come up with this more efficient solution to the problem from scratch, instead of reusing the already existing function for the prefix-notation of a term together with a simple list processing function.

Consequently, it would be worthwhile to have a technique, which can transform the modular solution (i)–(ix) into the efficient solution (x)–(xiii). To the best of our knowledge, techniques such as deforestation and catamorphism fusion cannot perform the optimization that we want to achieve here. In particular, classical deforestation fails due to its well-known problem of not reaching accumulating parameters [Chi94]. An approach that is applicable to our example is based on *attribute grammars* [Knu68] and was proposed independently in [Küh97, Küh98] and [CDPR98, CDPR99]. The idea is to transform [CF82] the two functions (in our formalism represented by two restricted macro tree transducers), which we want to compose, into attribute grammars, respectively *attributed tree transducers* [Fül81], which are an abstraction of attribute grammars. If the first attributed tree transducer fulfills the *single-use* restriction (cf. the *syntactic single-use requirement* in [Gie88]), which essentially means that every attribute instance in a tree may be used at most once in calculating the values of other attribute instances, the two transducers can be composed into a single attributed tree transducer (based on composition results from [Gan83, GG84, Gie88]). Applying a construction based on [Fra82], this attributed tree transducer can then be transformed into a macro tree transducer, thus giving a functional program for the composition of the two original functions, but without producing and consuming the intermediate result. This and related techniques of combining results from the theory of tree transducers for functional program optimization are presented in a uniform framework in [KV01]. A restricted instance of attribute grammar composition is also handled in [KGF01], where a single local rule is used to eliminate intermediate lists in compositions of `map`-style list transformers.

The approach of transformation by composition of attribute grammars—respectively, attributed tree transducers—does not work for all macro tree transducers, because only restricted macro tree transducers can be transformed into attributed tree transducers. One such restriction is the property of a macro tree transducer to be *weakly single-use* [Küh98], which roughly speaking means that at every occurrence in the input tree, recursive calls of functions on subtrees are restricted to appear at most once[1]. Moreover, in order to obtain a single-use attributed tree transducer—such that the composition result for attributed tree transducers becomes applicable—, the first macro tree transducer has to be further restricted to be *single-use*, which in addition to the restriction of being weakly single-use means that context parameters cannot be copied. The latter restriction is called *non-copying*. Figure 1 shows the transformation steps we have to take, where arrows indicate

[1] Also other restrictions of macro tree transducers are known that allow the transformation into attributed tree transducers, e.g., the *restricted-use* condition [KV01], which generalizes the weakly single-use property, and the *well-presented* restriction [CF82].

transformations and the semicolon stands for composition of tree transduction classes. Here, the class of functions computable by macro tree transducers is denoted as MAC, the class of attributed tree transductions as ATT, and the restrictions single-use and weakly single-use are indicated by subscripts su and wsu, respectively.

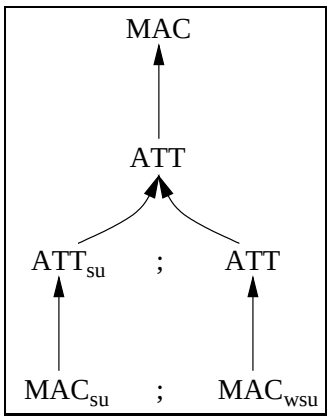

Figure 1: Indirect composition

There are several reasons, why we are interested in a direct construction for composing macro tree transducers, without the above indirection. Firstly, there would be benefits for an implementation, both, because a direct construction could be implemented more efficiently, and because the implementor—e.g., a compiler constructor—could work directly on functional programs, instead of having to consider the formalism of attribute grammars, just for optimization's sake. Secondly, there is a chance that a direct construction would produce better program code than the indirection by several transformations, because—for sake of generality—every single program transformation tends to introduce a certain "ballast" into the program, like, e.g., superfluous function parameters. Thirdly, a direct composition construction on the level of functional programs might prove more accessible to formal efficiency comparisons with other program transformation techniques. Finally, we want to broaden the applicability of macro tree transducer composition by generalizing the result $MAC_{su}; MAC_{wsu} \subseteq MAC$.

As we already noted, we cannot expect to be able to compose two arbitrary macro tree transducers. But, we can aim for weaker restrictions on the macro tree transducers than necessary so far, which still allow a composition. The success of this intention is limited, if we stick to the indirect construction via attributed tree transducers, because then we can only compose macro tree transducers that can be transformed into attributed tree transducers[2], which is not always possible, because we have a strict inclusion $ATT \subset MAC$ [Eng80, Fra82]. A direct composition construction has no such a priori limitation. In fact, we prove—with MAC_{nc} being the class of functions computable by non-copying macro tree transducers—the characterization $MAC_{nc}; MAC_{wsu} = MAC$, which is a generalization of the result quoted above, because single-use macro tree transducers are, by definition, non-copying. We will observe that the direct composition construction—developed and proven correct in [Voi01], and described in a simplified way in the present paper—allows us to further weaken the restriction on the second macro

[2]Another possibility is a new way of applying results from [Küh98], by first decomposing tree transducers, thus *introducing* new intermediate results, but creating new opportunities for compositions, which altogether can still lead to an optimization.

tree transducer by requiring the weakly single-use condition only for those of its functions that have at least one context parameter, thus giving us an even more general result, which is not available by the indirection via attributed tree transducers, respectively, attribute grammars. The presented construction is also applicable if one of the two macro tree transducers has no context parameters—i.e., is a top-down tree transducer [Rou70, Tha70, Eng75]—and the other one is an unrestricted macro tree transducer, thus incorporating in one construction two results from [Eng81, EV85].

The direct composition construction together with two post-processing constructions developed in [Voi01] will be able to transform the modular program—rules (i) to (ix)—from the introductory example into the more efficient program corresponding to rules (x)–(xiii).

This paper is divided into five sections and an appendix. In Section 2 we define necessary notations and introduce the basic concepts of macro tree transducers. In Section 3 we present the direct construction for composing two macro tree transducers—the first of which is non-copying and the second one weakly single-use—to a single macro tree transducer. In Section 4 we first consider inefficiencies that might still be present in a macro tree transducer obtained by the composition construction, and how to remove them. We will also show that a symmetric composition—a general construction for composing a weakly single-use macro tree transducer and a non-copying one in this order—cannot exist. As a corollary of the composition result, we show a size-height property for non-copying macro tree transducers. We will also present an implementation of the direct composition construction. Finally, Section 5 contains future research topics. The correctness proof for the composition construction was moved to the Appendix.

2 Preliminaries

We denote by $\mathbf{N}$ the set of natural numbers including 0, and for $n \in \mathbf{N}$, by $[n]$ the set $\{1, \ldots, n\}$. We set $\mathbf{N}_+ = \mathbf{N} \backslash \{0\}$. For a finite, non-empty set S of natural numbers, we denote by $max(S)$ the maximum of all its elements. We will use several sets of lowercase variables. As a convention, we denote for $k \in \mathbf{N}$, by X_k the finite set $\{x_1, \ldots, x_k\}$ of variables; analogous for Y and Z. For a set S, we denote by S^* the set of finite sequences of elements of S, where ε will denote the empty sequence. $\mathcal{P}(S)$ will denote the power set of a given set S. $|S|$ will denote the number of elements of a finite set S. For a set S and a binary relation $\lhd \subseteq S \times S$, we denote by $\lhd^+$ the transitive and by $\lhd^*$ the reflexive, transitive closure of $\lhd$, respectively.

For simplicity, we define substitution as substitution over strings (elements of S^* for some finite set S). For a string v and two lists of strings $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$ (with $n \in \mathbf{N}$), such that no two strings u_i and u_j $(i \neq j)$ overlap in v, we denote by $v[u_i \leftarrow v_i, i \in [n]]$ the string obtained from v by replacing all occurrences of every u_i in v by v_i. We will also use the alternative notation $v[u_1, \ldots, u_n \leftarrow v_1, \ldots, v_n]$, and appropriate multi-line notations for long substitutions. We write substitutions left-associative.

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*. For every $k \in \mathbf{N}$, we define $\Sigma^{(k)} = \{\sigma \in \Sigma \mid rank_\Sigma(\sigma) = k\}$. The rank k of a symbol σ will also be denoted by writing $\sigma^{(k)}$. For a ranked alphabet Σ, we denote the set of all its ranks as $rank(\Sigma) = \{k \in \mathbf{N} \mid \exists \sigma \in \Sigma : rank_\Sigma(\sigma) = k\}$. For a ranked alphabet Σ and a set A disjoint from Σ, we define the set $T_\Sigma(A)$ of *trees over* Σ *indexed by* A as the smallest set T such that

(i) $A \subseteq T$ and (ii) if $k \in \mathbf{N}$, $\sigma \in \Sigma^{(k)}$ and $t_1, \ldots, t_k \in T$, then also $\sigma(t_1, \ldots, t_k) \in T$. For nullary symbols, we simply write α instead of $\alpha()$. We denote the set $T_\Sigma(\emptyset)$ by T_Σ.

Let Σ be a ranked alphabet. A *rewrite rule* over Σ is a rule of the form $lhs \to rhs$ with $lhs, rhs \in T_\Sigma(V)$ for some set of variables V, such that lhs does not contain two occurrences of the same variable and every variable occurring in rhs is also contained in lhs. A set R of such rules induces a *(term) rewriting system* (see for example [DJ90, BN98]), denoted (Σ, R) (usually Σ, as well as V, will be clear from the context and we will only mention R), with 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 $c \in T_\Sigma(\{v\})$ (with $v \notin V$), which contains v exactly once, and there exist $n \in \mathbf{N}$, trees $t_1, \ldots, t_n \in T_\Sigma$ and variables $v_1, \ldots, v_n \in V$, such that:

$$t = c[v \leftarrow lhs\,[v_i \leftarrow t_i, i \in [n]]]$$
$$t' = c[v \leftarrow rhs[v_i \leftarrow t_i, i \in [n]]]$$

A reduction relation $\Rightarrow_R$ is called *confluent*, if for every $t, t_1, t_2 \in T_\Sigma$ with $t \Rightarrow_R^* t_1$ and $t \Rightarrow_R^* t_2$, there exists $t' \in T_\Sigma$ with $t_1 \Rightarrow_R^* t'$ and $t_2 \Rightarrow_R^* t'$. A reduction relation $\Rightarrow_R$ is called *terminating*, if there is no infinite chain $t_1 \Rightarrow_R t_2 \Rightarrow_R t_3 \Rightarrow_R \cdots$. If $t \Rightarrow_R^* t'$ and there is no t'' with $t' \Rightarrow_R t''$, then t' is called a *normal form* of t with respect to $\Rightarrow_R$. If $\Rightarrow_R$ is confluent and terminating, then every tree $t \in T_\Sigma$ has a unique normal form, denoted as $nf(\Rightarrow_R, t)$.

2.1 Functions on Trees

Let Σ, Δ and Ω be ranked alphabets. We call a total function $\tau : T_\Sigma \to T_\Delta$ a *tree transduction from T_Σ to T_Δ*. We define the composition of two tree transductions $\tau_1 : T_\Sigma \to T_\Delta$ and $\tau_2 : T_\Delta \to T_\Omega$, denoted by $\tau_1; \tau_2$, as $(\tau_1; \tau_2)(t) = \tau_2(\tau_1(t))$, for every $t \in T_\Sigma$. Further, we denote the composition of two classes $\mathcal{T}_1$ and $\mathcal{T}_2$ of tree transductions by $\mathcal{T}_1; \mathcal{T}_2 = \{\tau_1; \tau_2 \mid \tau_1 \in \mathcal{T}_1, \tau_2 \in \mathcal{T}_2\}$.

Let A be a set disjoint from Σ. We will need the *set of occurrences* in a tree, given by the function $occ : T_\Sigma(A) \to \mathcal{P}(\mathbf{N}^*)$, which is defined by structural recursion as follows: (i) if $t \in \Sigma^{(0)} \cup A$, then $occ(t) = \{\varepsilon\}$, and (ii) if $t = \sigma(t_1, \ldots, t_k)$ with $k \in \mathbf{N}_+$, $\sigma \in \Sigma^{(k)}$, $t_1, \ldots, t_k \in T_\Sigma(A)$, then $occ(t) = \{\varepsilon\} \cup \{i\nu \mid i \in [k], \nu \in occ(t_i)\}$.

We will also need the *label at an occurrence in a tree*, given by the mapping $lab : \{(t, o) \mid t \in T_\Sigma(A), o \in occ(t)\} \to \Sigma \cup A$, defined by: (i) if $t \in \Sigma^{(0)} \cup A$, then $lab(t, \varepsilon) = t$, and (ii) if $t = \sigma(t_1, \ldots, t_k)$ with $k \in \mathbf{N}_+$, $\sigma \in \Sigma^{(k)}$, $t_1, \ldots, t_k \in T_\Sigma(A)$, then $lab(t, \varepsilon) = \sigma$ and $lab(t, i\nu) = lab(t_i, \nu)$, for $i \in [k]$ and $\nu \in occ(t_i)$. The *subtree at an occurrence in a tree* is given by the function $sub : \{(t, o) \mid t \in T_\Sigma(A), o \in occ(t)\} \to T_\Sigma(A)$, defined by: (i) $sub(t, \varepsilon) = t$, for $t \in T_\Sigma(A)$, and (ii) if $t = \sigma(t_1, \ldots, t_k)$ with $k \in \mathbf{N}_+$, $\sigma \in \Sigma^{(k)}$, $t_1, \ldots, t_k \in T_\Sigma(A)$, then $sub(t, i\nu) = sub(t_i, \nu)$, for $i \in [k]$ and $\nu \in occ(t_i)$.

We define the *height of a tree* by the function $height : T_\Sigma(A) \to \mathbf{N}$ as follows: (i) if $t \in \Sigma^{(0)} \cup A$, then $height(t) = 0$, and (ii) if $t = \sigma(t_1, \ldots, t_k)$ with $k \in \mathbf{N}_+$, $\sigma \in \Sigma^{(k)}$, $t_1, \ldots, t_k \in T_\Sigma(A)$, then $height(t) = 1 + max(\{height(t_i) \mid i \in [k]\})$. Finally, we have the notion of *size of a tree*, defined by the mapping $size : T_\Sigma(A) \to \mathbf{N}$ with: (i) if $t \in \Sigma^{(0)} \cup A$, then $size(t) = 1$, and (ii) if $t = \sigma(t_1, \ldots, t_k)$ with $k \in \mathbf{N}_+$, $\sigma \in \Sigma^{(k)}$, $t_1, \ldots, t_k \in T_\Sigma(A)$, then $size(t) = 1 + \sum_{i \in [k]} size(t_i)$.

2.2 Macro Tree Transducers

We will define macro tree transducers of a slightly more general form than in [FV98], in that we allow for a more general initial expression than just the call of some state with fixed environment. Firstly, we describe the possible shapes of right-hand sides of rules.

Definition 2.1 Let Q and Δ be ranked alphabets, $k \in \mathbf{N}$ and V a set. We define the set $RHS(Q, \Delta, k, V)$ of *right-hand sides over Q and Δ, with k recursion variables and set of context variables V*, as the smallest set $RHS \subseteq T_{Q \cup \Delta}(X_k \cup V)$ satisfying the following conditions:

(1) $V \subseteq RHS$
(2) for every $\delta \in \Delta^{(a)}$ with $a \in \mathbf{N}$, and $\phi_1, \ldots, \phi_a \in RHS$ holds: $\delta(\phi_1, \ldots, \phi_a) \in RHS$
(3) for every $q \in Q^{(a+1)}$ with $a \in \mathbf{N}$, $x_i \in X_k$ and $\phi_1, \ldots, \phi_a \in RHS$ holds: $q(x_i, \phi_1, \ldots, \phi_a) \in RHS$ □

Now, we can define the components of a macro tree transducer.

Definition 2.2 A *macro tree transducer* (for short *mtt*) M is a tuple $(Q, \Sigma, \Delta, e, R)$ with:

(1) a ranked alphabet Q of *states*, where $Q^{(0)} = \emptyset$
(2) a ranked alphabet Σ of *input symbols*
(3) a ranked alphabet Δ of *output symbols*, where $\Delta^{(0)} \neq \emptyset$ and $Q \cap (\Sigma \cup \Delta) = \emptyset$
(4) an *initial expression* $e \in RHS(Q, \Delta, 1, \emptyset)$
(5) R contains for every $k, r \in \mathbf{N}$, $\sigma \in \Sigma^{(k)}$ and $q \in Q^{(r+1)}$, exactly one *rule* of the form $q(\sigma(x_1, \ldots, x_k), y_1, \ldots, y_r) \to rhs_{q,\sigma}$, with $rhs_{q,\sigma} \in RHS(Q, \Delta, k, Y_r)$ □

For an input symbol σ, we refer to the rules in R with left-hand side $q(\sigma(\cdots), \cdots)$ for some $q \in Q$, as *σ-rules*. The first argument of a state q is called *recursion argument*, the others are called *context parameters*. Of course, the actual variable names used in mtt rules are not fixed to come from X_k and Y_r for some $k, r \in \mathbf{N}$; consistent renaming is allowed.

Example 2.3 Let $Q_{pre} = \{pre^{(2)}\}$, $\Sigma_{term} = \{+^{(2)}, *^{(2)}, A^{(0)}, B^{(0)}\}$ and $\Delta_{list} = \{+^{(1)}, *^{(1)}, A^{(1)}, B^{(1)}, \epsilon^{(0)}\}$. We define the mtt $M_{pre} = (Q_{pre}, \Sigma_{term}, \Delta_{list}, e_{pre}, R_{pre})$, where R_{pre} contains the rules (i)–(iv) from the introduction, and $e_{pre} = pre(x_1, \epsilon)$. □

For further examples also see [KV01], which shows how well-known Haskell functions on, e.g., polymorphic lists—such as `map`, `reverse`, `++` and `foldr`—are handled as mtts.

The semantics of an mtt is a function from trees over the input ranked alphabet to trees over the output ranked alphabet. It is given by substituting the input tree for x_1 in the initial expression e of an mtt, and then calculating the normal form of this expression with respect to the reduction relation induced by the set R of rules. This normal form exists and is unique, because the rules of an mtt induce a confluent and terminating reduction relation (see, e.g., [FV98]).

Definition 2.4 We define the *tree transduction induced by an mtt $M = (Q, \Sigma, \Delta, e, R)$* as the total function $\tau(M) : T_\Sigma \to T_\Delta$, which assigns to every tree $t \in T_\Sigma$ the value $nf(\Rightarrow_R, e[x_1 \leftarrow t])$. We denote the class of *tree transductions induced by mtts* as *MAC*. □

Although the mtts presented here have a more general initial expression than those used in [FV98], our tree transduction class *MAC* coincides with the corresponding one in [FV98], because mtts in the two representations can be transformed into each other. Note that

the same need not be true for restricted classes of tree transductions, because our representation is more flexible.

Example 2.5 We apply the mtt from Example 2.3 to the input term $+(A, B)$, that is, we calculate the normal form of $e_{pre}[x_1 \leftarrow +(A, B)]$ with respect to $\Rightarrow_{R_{pre}}$:
$pre(+(A, B), \epsilon) \Rightarrow_{R_{pre}} +(pre(A, pre(B, \epsilon))) \Rightarrow_{R_{pre}} +(A(pre(B, \epsilon))) \Rightarrow_{R_{pre}} +(A(B(\epsilon)))$ □

We introduce two important syntactic properties of mtts (cf. [Küh98]) and the corresponding computability classes.

Definition 2.6 An mtt is called *non-copying*, if in the right-hand side of every rule there is no more than one occurrence of every context variable. We denote the class of *tree transductions induced by non-copying mtts* as MAC_{nc}. □

Definition 2.7 An mtt $M = (P, \Delta, \Omega, e, R)$ is called *weakly single-use*, if the following two conditions hold:

(i) For every $k \in \mathbf{N}_+$, $\delta \in \Delta^{(k)}$, $i \in [k]$ and $p \in P$, a call of the form $p(x_i, \cdots)$ occurs in a right-hand side of at most one δ-rule and there only once.

(ii) For every $p \in P$, the initial expression e contains at most one occurrence of a call $p(x_1, \cdots)$.

We denote the class of *tree transductions induced by weakly single-use mtts* as MAC_{wsu}. □

We also identify mtts, which have both of the above properties.

Definition 2.8 An mtt is called *single-use*, if it is both non-copying and weakly single-use. We denote the class of *tree transductions induced by single-use mtts* as MAC_{su}. □

Another important restriction of mtts is that of having no context parameters at all, which gives us top-down tree transducers [Rou70, Tha70, Eng75].

Definition 2.9 An mtt is called a *top-down tree transducer*, if all its states have rank one, that is, there are no context parameters present. We denote the class of *tree transductions induced by top-down tree transducers* as TOP. □

Example 2.10

- Let $\Omega_{ins} = \{ADD^{(1)}, MUL^{(1)}, LOAD_A^{(1)}, LOAD_B^{(1)}, \epsilon^{(0)}\}$. Then the mtt $M_{aux} = (\{aux^{(2)}\}, \Delta_{list}, \Omega_{ins}, aux(x_1, \epsilon), R_{aux})$, where R_{aux} contains the rules (v)–(ix) from the introduction, is both non-copying and weakly single-use, thus it is also single-use.
- Let $\Sigma_{mon} = \{A^{(1)}, B^{(1)}, E^{(0)}\}$. Then $M_{top} = (\{p^{(1)}, q^{(1)}\}, \Sigma_{mon}, \Sigma_{mon}, p(x_1), R_{top})$ with set of rules R_{top}:
  (i) : $p(A(x_1)) \rightarrow A(p(x_1))$
  (ii) : $p(B(x_1)) \rightarrow q(x_1)$
  (iii) : $p(E) \rightarrow E$
  (iv) : $q(A(x_1)) \rightarrow p(x_1)$
  (v) : $q(B(x_1)) \rightarrow B(q(x_1))$
  (vi) : $q(E) \rightarrow E$
  is a top-down tree transducer and thus trivially non-copying, but it is not weakly single-use, because a call $p(x_1)$ occurs twice in right-hand sides of A-rules.

- The mtt $M_{bin} = (\{q^{(2)}\}, \Sigma_{mon}, \{\alpha^{(2)}, \beta^{(2)}, \epsilon^{(0)}\}, q(x_1, \epsilon), R_{bin})$ with set of rules R_{bin}:

  (i) : $q(A(x_1), y_1) \rightarrow q(x_1, \alpha(y_1, y_1))$
  (ii) : $q(B(x_1), y_1) \rightarrow q(x_1, \beta(y_1, y_1))$
  (iii) : $q(E, y_1) \rightarrow y_1$

  is not non-copying. □

3 The Construction

The aim of this paper is to construct for two given mtts—the first of which is non-copying and the second one weakly single-use—a single new mtt that implements the composition of the two original ones. We can give an a priori justification[3] of the feasibility of this aim by semantic reasoning with the help of other tree transduction classes [Fül81, Küh97], which are not covered in this paper. We only mention that ATT, $YIELD$ and ATT_{su} denote the classes of attributed tree transductions, *yield*-functions and tree transductions induced by single-use attributed tree transducers, respectively.

Theorem 3.1 $MAC_{nc}; MAC_{wsu} = MAC$ □

Proof:

$MAC_{nc}; MAC_{wsu}$
$\subseteq TOP; ATT_{su}; MAC_{wsu}$ (an analogue of Lemma 5.3 in [Küh98])
$\subseteq TOP; ATT_{su}; ATT$ (Theorem 7.1 in [Küh98])
$\subseteq TOP; ATT$ (Lemma 6.4 in [Küh97])
$\subseteq TOP; MAC$ (cf. [Fra82], also Lemma 6.1 in [FV98])
$\subseteq MAC$ (remark following Theorem 3.1 in [Eng81])
$\subseteq TOP; YIELD$ (Theorem 3 in [Eng80])
$\subseteq MAC_{nc}; MAC_{wsu}$ ($TOP \subseteq MAC_{nc}$ trivially; Example 4.5 in [EV85], Lemma 6.10 in [Küh97]) □

The same kind of semantic reasoning can be applied to obtain a result on well-presented mtts [CF82, FV98]—namely, $MAC_{nc}; MAC_{wp} = MAC$ [4]—, as well as on classes of macro attributed tree transducers [KV94], which are not subject of the present paper.

3.1 The Idea

Before giving the formal construction for composing restricted mtts, we will try to provide an intuitive explanation of the underlying ideas:

The development of the construction first goes along the same lines as the classical construction for $MAC; TOP \subseteq MAC$ from [EV85]. Given two mtts M_1 (from T_Σ to T_Δ, with rules R_1) and M_2 (from T_Δ to T_Ω, with rules R_2) with respective sets of states $Q = \{q_1, \ldots, q_n\}$ and $P = \{p_1, \ldots, p_m\}$, we want to create an mtt $M_{1;2}$ (from T_Σ to T_Ω) with paired states $\{(q, p) \mid q \in Q, p \in P\}$. The basic idea behind this pairing construction is that a nested call of the form $p(q(t, y_1, \ldots, y_r), z_1, \ldots, z_s)$—with concrete trees for the $y_1, \ldots, y_r$ and $z_1, \ldots, z_s$—, which might arise from running M_1 and M_2 independently (having started with a composite expression, like $aux(pre(t, \epsilon), \epsilon)$ in the introductory example),

[3] Although it was obtained in [Voi01] only *after* establishing the composition construction; the original aim of the former paper was to give a direct construction for the weaker result $MAC_{su}; MAC_{wsu} \subseteq MAC$.

[4] $MAC_{nc}; MAC_{wp} \subseteq TOP; ATT_{su}; ATT \subseteq MAC \subseteq TOP; YIELD \subseteq MAC_{nc}; MAC_{wp}$, cf. Theorem 5.10 in [CF82] and the construction from Example 4.5 in [EV85]

should correspond to a construct $(q,p)(t,?,\ldots,?)$ during the computation of $M_{1;2}$. Now, how to fill the question marks, respectively decide what rank the new states should have? Clearly, we will keep the context parameters $z_1,\ldots,z_s$ of the outer call to $p \in P^{(s+1)}$. The situation is more involved for the context parameters of $q \in Q^{(r+1)}$, because they are not just copied into the final output. Rather, they will occur in the intermediate result after reducing $q(t,y_1,\ldots,y_r)$, and during the computation of $p(\bullet,z_1,\ldots,z_s)$ on that intermediate result—substituted for the bullet—, they are likely to be met by some state of the second mtt. Following the construction from [EV85], we will thus, for every context parameter y of q, hold available its m translations for all the states $p_1,\ldots,p_m$ of M_2. Accordingly, $p(q(t,y_1,\ldots,y_r),z_1,\ldots,z_s)$ will be replaced by $(q,p)(t,y_{1,p_1},\ldots,y_{r,p_m},z_1,\ldots,z_s)$, assuming that in the $y_{1,p_1},\ldots,y_{r,p_m}$-positions we are provided with all necessary translations of context parameters.

Now, that we settled the question of the ranks of our new states, we have to think about how to build the rules for them. Namely, given some input symbol $\sigma \in \Sigma^{(k)}$, what will be the right-hand side for $(q,p)(\sigma(x_1,\ldots,x_k),y_{1,p_1},\ldots,y_{r,p_m},z_1,\ldots,z_s)$? According to our idea, this should be corresponding to $p(\underline{q(\sigma(x_1,\ldots,x_k),y_1,\ldots,y_r)},z_1,\ldots,z_s)$, so we will take the right-hand side of the first mtt's rule $q(\sigma(x_1,\ldots,x_k),y_1,\ldots,y_r) \to rhs_{q,\sigma}$ and translate it with p by reducing $p(\underline{rhs_{q,\sigma}},z_1,\ldots,z_s)$. Of course, in general $rhs_{q,\sigma}$ will *not* only consist of output symbols of the first mtt—which are readily processed by states of M_2—, so it is not obvious how to do this reduction. In fact, while reducing $p(rhs_{q,\sigma},z_1,\ldots,z_s)$, we have to take care of three different cases:

1. some state $p' \in P$ meets an output symbol δ of M_1 (which therefore is an input symbol for the second mtt)
2. some state $p' \in P$ meets a context parameter y_h of q
3. some state $p' \in P$ meets a call to some state $q' \in Q$ of the first mtt

In the first case, we will just apply the rule given by M_2 for p' at δ, and it will be exactly this application, which leads to the elimination of intermediate data structures. In the second case, we remember that the call to (q,p) was provided with translations of the context parameters with all possible states of P, so we will just select the right one and output $y_{h,p'}$. In the third case, we will have met some situation $p'(q'(x_i,\phi_1,\ldots,\phi_a),\eta_1,\ldots,\eta_{s'})$ with $p' \in P^{(s'+1)}$, $q' \in Q^{(a+1)}$, furthermore $\phi_1,\ldots,\phi_a$ being part of the first mtt's right-hand side $rhs_{q,\sigma}$ and $\eta_1,\ldots,\eta_{s'}$ being parts of right-hand side expressions for $M_{1;2}$, which have been built up during the reduction of $rhs_{q,\sigma}$ with p. By our underlying idea of pairing states, we know that we should replace this with something like $(q',p')(x_i,?,\ldots,?,\eta_1,\ldots,\eta_{s'})$, where for the question marks, we have to fill in the translations of the context parameters of q'—namely $\phi_1,\ldots,\phi_a$—with all possible states of M_2 (such that the state (q',p') can later use these provided translations). Thus, we get:

(*) $$(q',p')(x_i,p_1(\phi_1,?,\ldots,?),\ldots,p_m(\phi_a,?,\ldots,?),\eta_1,\ldots,\eta_{s'}).$$

Note that here the construction "reaches" intermediate data structures in accumulating (context) parameters of the function (state) q', as opposed to classical deforestation. In the construction for the result $MAC;TOP \subseteq MAC$, we would be fine now, because then all the states of the second mtt would have rank one, so there would be no question marks left in our expression and we could go on by translating the $\phi_1,\ldots,\phi_a$ with states $p_1,\ldots,p_m$, just in the same way as we started processing $rhs_{q,\sigma}$ with p.

Instead, as we are dealing with general mtts, we have to fill the question mark slots in the expression **(*)**. Now, what are we supposed to provide in the places of the question marks? Clearly, it should be the context parameter values, with which the states $p_1, \ldots, p_m$ are expected to "arrive" at occurrences of $\phi_1, \ldots, \phi_a$ during computation of the $p'(q'(x_i, \phi_1, \ldots, \phi_a), \eta_1, \ldots, \eta_{s'})$ met above (and thus will clearly depend on the tree substituted for x_i). However, it is far from obvious, whether we can always provide this information. So, we better first answer the following question in general (see Figure 2):

> "Given two states q' and p' of mtts M_1 and M_2 as above, and some input tree t for M_1, can we determine, for any state p'' of M_2, and any context parameter y_h from $y_1, \ldots, y_a$, which will be the values in context parameter positions in every occurrence of a call to p'' on y_h during the reduction of $p'(q'(t, y_1, \ldots, y_a), \eta_1, \ldots, \eta_{s'})$?"

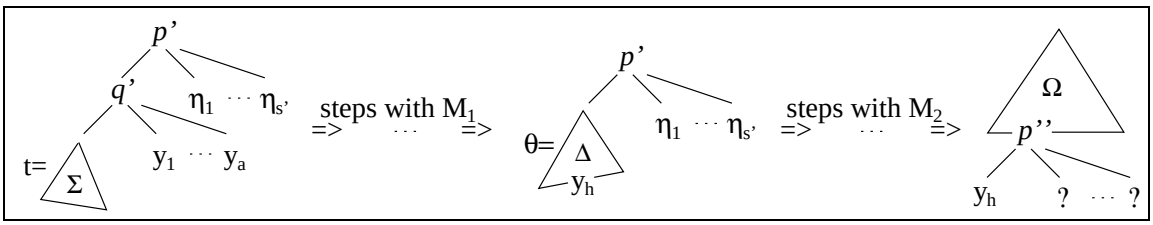

Figure 2: Can we uniquely determine the values in question mark positions?

To further illustrate the problem, we consider a—rather artificial—example for concrete M_1 and M_2.

Example 3.2 Let $\Sigma'_{mon} = \{\alpha^{(1)}, \beta^{(1)}, \epsilon^{(0)}\}$ and $M_1 = (\{q_1^{(2)}\}, \Sigma_{mon}, \Sigma'_{mon}, q_1(x_1, \epsilon), R_1)$ with set of rules R_1:

(i) : $q_1(A(x_1), y_1) \rightarrow \alpha(q_1(x_1, y_1))$
(ii) : $q_1(B(x_1), y_1) \rightarrow q_1(x_1, \beta(y_1))$
(iii) : $q_1(E, y_1) \rightarrow y_1$

Let $Nat = \{succ^{(1)}, zero^{(0)}\}$ and $M_2 = (\{p_1^{(2)}, p_2^{(2)}\}, \Sigma'_{mon}, Nat \cup \{\omega^{(2)}\}, p_2(x_1, zero), R_2)$ with set of rules R_2:

(iv) : $p_1(\alpha(x_1), z_1) \rightarrow p_1(x_1, p_2(x_1, z_1))$
(v) : $p_1(\beta(x_1), z_1) \rightarrow succ(z_1)$
(vi) : $p_1(\epsilon, z_1) \rightarrow z_1$
(vii) : $p_2(\alpha(x_1), z_1) \rightarrow z_1$
(viii) : $p_2(\beta(x_1), z_1) \rightarrow p_2(x_1, \omega(p_1(x_1, z_1), z_1))$
(ix) : $p_2(\epsilon, z_1) \rightarrow z_1$

Now, we might ask what will be the values in context parameter positions in every occurrence of a call to p_2 on y_1 during the reduction of $p_1(q_1(A(B(E)), y_1), z_1)$:

$p_1(q_1(A(B(E)), y_1), z_1) \Rightarrow_{R_1} p_1(\alpha(q_1(B(E), y_1)), z_1) \Rightarrow_{R_1} p_1(\alpha(q_1(E, \beta(y_1))), z_1) \Rightarrow_{R_1}$
$p_1(\alpha(\beta(y_1)), z_1) \Rightarrow_{R_2} p_1(\beta(y_1), p_2(\beta(y_1), z_1)) \Rightarrow_{R_2} p_1(\beta(y_1), \underline{p_2(y_1, \underline{\omega(p_1(y_1, z_1), z_1)})})$

Here, the answer is unique, because no other calls to p_2 on y_1 will occur during the further possible reduction. □

On the other hand, the following example shows that in general the answer to the question above Figure 2 is *no*!

Example 3.3

(a) consider rules $q'(\alpha, y_1) \to \delta(y_1, y_1)$ of M_1 and $p'(\delta(x_1, x_2)) \to \omega(p''(x_1, \alpha), p''(x_2, \beta))$ of M_2, giving for $t = \alpha$ the reduction:
$p'(q'(\alpha, y_1)) \Rightarrow_{R_1} p'(\delta(y_1, y_1)) \Rightarrow_{R_2} \omega(p''(y_1, \alpha), p''(y_1, \beta))$

(b) consider rules $q'(\alpha, y_1) \to \gamma(y_1)$ of M_1 and $p'(\gamma(x_1)) \to p''(x_1, p''(x_1, \alpha))$ of M_2, giving for $t = \alpha$ the reduction:
$p'(q'(\alpha, y_1)) \Rightarrow_{R_1} p'(\gamma(y_1)) \Rightarrow_{R_2} p''(y_1, p''(y_1, \alpha))$

In both cases, the reduction leads to occurrences of calls to p'' on y_1 with different values in context parameter positions. □

However, we claim that the question above Figure 2 can be answered positively, if M_1 is non-copying and M_2 is weakly single-use (which excludes the "counterexamples" from Example 3.3)—or, trivially, if one of them is a top-down tree transducer—, by reasoning as follows:

If M_1 is non-copying, then for every input $t \in T_\Sigma$, the normal form θ of $q'(t, y_1, \ldots, y_a)$ will contain at most one occurrence of every $y_1, \ldots, y_a$. If, moreover, M_2 is weakly single-use, then we can find out the values of context parameters in any occurrence of a call to some state p'' on, e.g., y_h during the reduction of $p'(\theta, \eta_1, \ldots, \eta_{s'})$, by "walking upwards in the tree θ" from the unique—if existing—occurrence of y_h in θ in the following manner:

(1) If y_h is at the root of θ, then either $p'' = p'$ and the sought context parameter values are $\eta_1, \ldots, \eta_{s'}$, or $p'' \neq p'$ and we can take some arbitrary dummy context parameters, because there will be no call to p'' on y_h during reduction of $p'(\theta, \eta_1, \ldots, \eta_{s'})$.

(2) If y_h is not at the root of θ, then it must occur below some output symbol of M_1, i.e., there is some occurrence o in θ and some symbol δ (which is an input symbol for M_2), such that $lab(\theta, o) = \delta$ and y_h occurs as child of this occurrence, say $lab(\theta, oi) = y_h$. Then there is at most one possible way, how y_h can be reached by a call to state p'', namely by the unique (since the second mtt is weakly single-use) occurrence of a $p''(x_i, \cdots)$-call in some right-hand side of a δ-rule of M_2. If we assume that this unique occurrence is in the right-hand side of a rule
$$p'''(\delta(\cdots), z_1, \ldots, z_{s'''}) \to \cdots p''(x_i, \psi_1, \ldots, \psi_{s''}) \cdots,$$
we could from $\psi_1, \ldots, \psi_{s''}$ determine our sought context parameters of calls to p'' on reaching the occurrence of y_h, if we knew the context parameters of calls to p''' on reaching the occurrence of δ above y_h (because the $\psi_1, \ldots, \psi_{s''}$ might contain occurrences of the $z_1, \ldots, z_{s'''}$). This occurrence can now again be either the root of θ or below some output symbol of M_1 in θ, so we either are in case (1) or again "walk upwards" in θ using this second case, until finally we reach the root.

This informal and incomplete explanation (the $\psi_1, \ldots, \psi_{s''}$ might also contain calls on other children of the δ-occurrence, or calls on the same child with another state, and we did not say, how to handle these) seems to suggest that the question above Figure 2 can be answered with *yes*. However, this does not immediately help our construction of $M_{1;2}$, because an mtt has to work locally and cannot follow our proposed procedure of "reduce $q'(t, y_1, \ldots, y_a)$ to its normal form θ, check where the context variables $y_1, \ldots, y_a$ are in θ, and walk your way up from there". Instead, we claim that the procedure suggested above *can* be implemented with an mtt, and exactly this is the "difficult part" of the composition construction.

For every two states q and p of M_1 and M_2, respectively, and every pair of context parameter positions h and l of q and p, respectively, we will introduce a new state (h_q, l_p)[5], which on a given tree t shall compute, what will be the lth context parameter in a call to p on the hth context variable of q, resulting from some reduction of $p'(q(t, \ldots, y_h, \ldots), \cdots)$, where p' is unknown, but unique (due to our reasoning of "walking upwards in the intermediate tree, using the weakly single-use property").

Coming back to **(*)**, we can now fill the question mark positions, using these new states:

$$\begin{aligned}\textbf{(**)}\qquad (q', p')(x_i, p_1(\phi_1, (1_{q'}, 1_{p_1})(x_i, ?, \ldots, ?), \ldots, (1_{q'}, s_{1p_1})(x_i, ?, \ldots, ?)),\\ \ldots,\\ p_m(\phi_a, (a_{q'}, 1_{p_m})(x_i, ?, \ldots, ?), \ldots, (a_{q'}, s_{mp_m})(x_i, ?, \ldots, ?)),\\ \eta_1, \ldots, \eta_{s'}),\end{aligned}$$

assuming that the states $p_1, \ldots, p_m$ of M_2 have $s_1, \ldots, s_m$ context parameter positions.

But, this produced new question marks, and before we can fill them, we have to think about what context parameters such a new state (h_q, l_p) requires, respectively, how its rules can be constructed. For every input symbol $\sigma \in \Sigma^{(k)}$ of M_1, we have to construct a rule

$$(h_q, l_p)(\sigma(x_1, \ldots, x_k), \cdots) \to ?,$$

aimed at computing the lth context parameter in a call to p on y_h, resulting from a reduction of $p'(q(\sigma(x_1, \ldots, x_k), \ldots, y_h, \ldots), \cdots)$ for unknown p'. Clearly, we should use our knowledge of the first mtt's rule $q(\sigma(x_1, \ldots, x_k), \ldots, y_h, \ldots) \to rhs_{q,\sigma}$. If $rhs_{q,\sigma}$ does not contain y_h, then there cannot result a call to any p on y_h from the reduction, so we are safe to choose some dummy right-hand side for (h_q, l_p) at σ. For the more general case that $rhs_{q,\sigma}$ *does* contain y_h, we will in this informal explanation only consider two examples, and hope they are somewhat illuminating our idea.

Example 3.4 Recall the mtts M_1 and M_2 from Example 3.2, but assume the simpler right-hand side $rhs_{q_1,B} = \beta(y_1)$ for rule (ii). Assume that we are interested in the first context parameter in a call to p_2 on the occurrence of y_1 in this right-hand side (that is, we want to construct the right-hand side for $(1_{q_1}, 1_{p_2})$ at B). We know that the only way to have a call to p_2 on the unique occurrence of y_1 is via a call to p_2 on x_1 in the right-hand side of some β-rule of M_2. The only such call occurs in the rule (viii): $p_2(\beta(x_1), z_1) \to p_2(x_1, \omega(p_1(x_1, z_1), z_1))$. Now, we know, what the first context parameter of p_2 on y_1 will be—namely $\omega(p_1(x_1, z_1), z_1)$—, but there is still x_1 present, for which we have to substitute the child of the β-occurrence in $rhs_{q_1,B}$. Thus, we get $\omega(p_1(y_1, z_1), z_1)$. □

The value obtained for the first context parameter in a call to p_2 on the occurrence of y_1 in $rhs_{q_1,B} = \beta(y_1)$ in the previous example, depends both on the p_1-translation of y_1 and on the first context parameter z_1 of p_2 at the root of $rhs_{q_1,B}$. Since, in general, we do not know, which states—here p_1 and p_2—will be concerned, we might need for computing (h_q, l_p) at σ:

1. all translations of context parameters of q with some state of M_2
2. all context parameters for every possible state of M_2 at the occurrence of $q(\sigma(x_1, \ldots, x_k), \ldots, y_h, \ldots)$ (respectively, at the root of $rhs_{q,\sigma}$)

[5]This shall not denote indexed integers, rather it is used as a compacter notation for the pair of pairs $((h, q), (l, p))$.

Thus, the rule discussed above Example 3.4 will indeed have the left-hand side:

$$(h_q, l_p)(\sigma(x_1, \ldots, x_k), y_{1,p_1}, \ldots, y_{r,p_m}, z_{p_1,1}, \ldots, z_{p_m,s_m}).$$

Note that—as follows from the reasoning below Example 3.3—for every input tree t, there will be at most one state p' of M_2, such that reducing $p'(q(t, \ldots, y_h, \ldots), \cdots)$ leads to a call to p on y_h. The result of reducing $(h_q, l_p)(t, y_{1,p_1}, \ldots, y_{r,p_m}, z_{p_1,1}, \ldots, z_{p_m,s_m})$, which shall compute the lth context parameter of this call to p on y_h, will then of the $z_{p_1,1}, \ldots, z_{p_m,s_m}$ only contain those parameters, which are associated with this unique p'. However, since for different input trees t also this p' might differ, it is unavoidable for the state (h_q, l_p) to have all the context parameters $z_{p_1,1}, \ldots, z_{p_m,s_m}$, though only a subset of them will be relevant in every concrete computation. Intuitively, in order to get the lth context parameter of a call $p(y_h, \cdots)$ resulting from reduction of $p'(q(t, \ldots, y_h, \ldots), \cdots)$, we have to compute $(h_q, l_p)(t, y_{1,p_1}, \ldots, y_{r,p_m}, z_{p_1,1}, \ldots, z_{p_m,s_m})$ and replace the $z_{p',1}, \ldots, z_{p',s'}$ (no other variables from $z_{p_1,1}, \ldots, z_{p_m,s_m}$ will occur in the result) by the context parameters of p' at the occurrence of $q(t, \ldots, y_h, \ldots)$.

In Example 3.4 we used a simplified right-hand side $rhs_{q_1,B}$, where no function call occurred. Continuing the example, we will now consider the original right-hand side of Example 3.2.

Example 3.5 Recall that we want to find the first context parameter in a call to p_2 on the occurrence of y_1 in the right-hand side $rhs_{q_1,B} = q_1(x_1, \beta(y_1))$ (cf. Example 3.2). As in Example 3.4, we have to start from the unique occurrence of y_1 in this right-hand side, which is the child of the β-occurrence. By the same reasoning as above we get $\omega(p_1(y_1, z_1), z_1)$, which contains the p_1-translation of y_1 and the first context parameter of p_2 on reaching the occurrence of β in $rhs_{q_1,B}$. Since we noted that $(1_{q_1}, 1_{p_2})$ will have as context parameters the translations of y_1 with all states of M_2, we can replace $p_1(y_1, z_1)$ with y_{1,p_1}. For z_1, however, we need to do further calculation, because it now does not stand for the context parameter of p_2 at the root of $rhs_{q_1,B}$, but rather for the one on reaching the β-occurrence in $rhs_{q_1,B}$. But this β occurs at the root of a context parameter position of q_1, so we know that we *can* compute our needed value for z_1 by calling $(1_{q_1}, 1_{p_2})$ on x_1 (thus computing the first context parameter of p_2 on reaching the first context parameter position of q_1 during a reduction on $q_1(x_1, \cdots)$ with some state of M_2). Thus, we get $\omega(y_{1,p_1}, (1_{q_1}, 1_{p_2})(x_1, ?, \ldots, ?))$. How to fill the context parameter positions of this call, is another problem that we do not want to discuss at this point (rather, we postpone this discussion until having formally introduced the construction). □

Now, we know a bit better, how to fill the question mark slots in **(**)**. Recall that we introduced the calls to $(1_{q'}, 1_{p_1}), \ldots, (a_{q'}, s_{mp_m})$ on x_i, in order to compute the context parameter values, with which the states $p_1, \ldots, p_m$ are expected to "arrive" at occurrences of $\phi_1, \ldots, \phi_a$ during the reduction of $p'(q'(x_i, \phi_1, \ldots, \phi_a), \eta_1, \ldots, \eta_{s'})$ for instantiated x_i. Further, recall that—by the discussion below Example 3.4—every state $(h_{q'}, l_p) \in \{(1_{q'}, 1_{p_1}), \ldots, (a_{q'}, s_{mp_m})\}$ expects in the $z_{p_1,1}, \ldots, z_{p_m,s_m}$-positions of its call on x_i, the context parameters of all possible states of M_2 at the occurrence of $q'(x_i, \phi_1, \ldots, \phi_a)$. However, only those of the translations $p_1(\phi_1, \cdots), \ldots, p_m(\phi_a, \cdots)$ will be needed during the above computation of state (q', p') on x_i, which actually occur during the reduction of $p'(q'(x_i, \phi_1, \ldots, \phi_a), \eta_1, \ldots, \eta_{s'})$ for instantiated x_i. For such a p and ϕ_h, we know (by the discussion above Example 3.5) that a call to state $(h_{q'}, l_p)$ on the concrete tree substituted for x_i, will depend only on those of its $z_{p_1,1}, \ldots, z_{p_m,s_m}$-positions, which are associated

with p'. We also gave the intuition that for these $z_{p',1}, \ldots, z_{p',s'}$-positions, we have to provide the context parameters of p' at the occurrence of $q'(x_i, \ldots, \phi_h, \ldots)$. Thus, we will fill the $z_{p',1}, \ldots, z_{p',s'}$-positions with $\eta_1, \ldots, \eta_{s'}$, and the others with a dummy nullary output symbol nil (these positions will *not* influence the computation, as we will demonstrate later on Example 3.7). This leaves us with the following expression:

$$\begin{aligned}(***)\quad (q',p')(x_i, p_1(\phi_1, (1_{q'}, 1_{p_1})(x_i, ?, \ldots, ?, nil, \ldots, nil, \eta_1, \ldots, \eta_{s'}, nil, \ldots, nil),\\ \ldots,\\ (1_{q'}, s_{1_{p_1}})(x_i, ?, \ldots, ?, nil, \ldots, nil, \eta_1, \ldots, \eta_{s'}, nil, \ldots, nil)),\\ \ldots,\\ p_m(\phi_a, (a_{q'}, 1_{p_m})(x_i, ?, \ldots, ?, nil, \ldots, nil, \eta_1, \ldots, \eta_{s'}, nil, \ldots, nil),\\ \ldots,\\ (a_{q'}, s_{mp_m})(x_i, ?, \ldots, ?, nil, \ldots, nil, \eta_1, \ldots, \eta_{s'}, nil, \ldots, nil)),\\ \eta_1, \ldots, \eta_{s'}),\end{aligned}$$

still having to fill the $y_{1,p_1}, \ldots, y_{a,p_m}$-positions of calls to $(1_{q'}, 1_{p_1}), \ldots, (a_{q'}, s_{mp_m})$. Here, we have to provide $p_1, \ldots, p_m$-translations of the $\phi_1, \ldots, \phi_a$, which might be needed during the computations of these states. This looks suspiciously like a circular construction, because we would use $p_1(\phi_1, \cdots)$ within $p_1(\phi_1, \cdots)$ and the like. However, we claim that we can "cut" these cycles by keeping track of the nesting (similar to the unfolding in the construction for $ATT \subseteq MAC$, cf. [Fra82, FV98]), because the positions, where such cyclic dependencies would occur, can for no possible input tree influence the computation. The intuitive reason for this is that otherwise we would meet a situation corresponding to the "counterexample" (b) in Example 3.3, like $p''(y_1, p''(y_1, \alpha))$, which we claimed cannot occur, because all occurrences of $p''(y_1, \cdots)$ must have identical context parameters, if M_1 is non-copying and M_2 is weakly single-use.

3.2 Formal Development

We give the construction with the help of term rewriting systems R_2, Pre and $Pair$, where R_2 contains the rules of the second given mtt, and Pre and $Pair$ are introduced in the construction:

- R_2 describes the application of a rule of M_2, corresponding to the first case on page 10.
- Pre describes the use of provided translations of context parameters of M_1 with states of M_2, corresponding to the second case on page 10.
- $Pair$ describes the pairing of states, corresponding to the third case on page 10.

Further, functions $nest_q$ and $ctx_{p,\phi}$ are defined for building up nested translations of context parameters of M_1 with states of M_2, and for building up right-hand sides for (h_q, l_p)-states, respectively:

For some state q of M_1, the $nest_q$-function initiates the preparation of translated context parameters of q, necessary for the paired states as discussed in the previous subsection. Its first argument is the position of the context parameter to be translated, its second argument is the state of M_2, with which to translate. The third argument is a set, which keeps track of nested parameter-state-combinations, in order to perform the "cutting" of cycles during the nesting process, as mentioned at the end of the previous subsection.

For some state p of M_2 and a right-hand side ϕ of M_1, the $ctx_{p,\phi}$-function performs the "walking upwards" in the right-hand side ϕ, necessary for giving rules to the (h_q, l_p)-states. The intuitive reading is as follows: from $ctx_{p,\phi}(o, l)$ we will obtain a right-hand side expression for $M_{1;2}$, which computes the lth context parameter in a call to state p on reaching the occurrence o in the first mtt's right-hand side expression ϕ (having started at the root of ϕ with some unknown state). We will give an explanation of the defining cases of $ctx_{p,\phi}$-functions (which actually are parameterized by additional subscripts k and r that are only necessary to determine correct types for them) below the construction.

It is straightforward—though tiresome and thus omitted here—to establish that the construction is well-defined, by showing well-definedness of the introduced functions and confluence and termination of the reduction relation induced by the term rewriting system $R_2 \cup Pre \cup Pair$. In particular, this gives us a guaranty for termination of our program transformation technique, without involving a bookkeeping of the transformation history that is necessary for ensuring termination of, e.g., classical deforestation and supercompilation.

Construction 3.6 Let $M_1 = (Q, \Sigma, \Delta, e_1, R_1)$ and $M_2 = (P, \Delta, \Omega, e_2, R_2)$ be mtts, such that M_1 is non-copying and M_2 is weakly single-use. We construct the mtt $M_{1;2} = (S, \Sigma, \Omega, e_{1;2}, R_{1;2})$ as follows:

Let $m = |P|$ and fix some ordering of states in P, such that $P = \{p_1, \ldots, p_m\}$. For $i \in [m]$, let $s_i \in \mathbf{N}$ be such that $p_i \in P^{(s_i+1)}$. Additionally, let $nil \in \Omega^{(0)}$ be some arbitrary output symbol and $Z_P = \{z_{p_1,1}, \ldots, z_{p_1,s_1}, \ldots, z_{p_m,1}, \ldots, z_{p_m,s_m}\}$. Then, the components of $M_{1;2}$ are obtained as follows:

- $S = \{(q,p)^{(r*m+s+1)} \mid r, s \in \mathbf{N}, q \in Q^{(r+1)}, p \in P^{(s+1)}\}$
  $\cup\ \{(h_q, l_p)^{(r*m+|Z_P|+1)} \mid r, s \in \mathbf{N}_+, q \in Q^{(r+1)}, h \in [r], p \in P^{(s+1)}, l \in [s]\}$
- $e_{1;2} = nf(\Rightarrow_{R_2 \cup Pair}, e_2[x_1 \leftarrow e_1])$
- $R_{1;2}$ contains:
  - for every $r, s, k \in \mathbf{N}$, $q \in Q^{(r+1)}$, $p \in P^{(s+1)}$ and $\sigma \in \Sigma^{(k)}$, the rule:
    $(q,p)(\sigma(x_1, \ldots, x_k), y_{1,p_1}, \ldots, y_{r,p_m}, z_1, \ldots, z_s)$
    $\rightarrow nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, p(rhs_{q,\sigma}, z_1, \ldots, z_s))$,
  - for every $k \in \mathbf{N}$, $r, s \in \mathbf{N}_+$, $q \in Q^{(r+1)}$, $h \in [r]$, $p \in P^{(s+1)}$, $l \in [s]$ and $\sigma \in \Sigma^{(k)}$, the rule:
    $(h_q, l_p)(\sigma(x_1, \ldots, x_k), y_{1,p_1}, \ldots, y_{r,p_m}, z_{p_1,1}, \ldots, z_{p_m,s_m})$
    $\rightarrow nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, \psi)$,
    where $\psi = nil$, if $rhs_{q,\sigma}$ does not contain the context variable y_h; otherwise $\psi = ctx_{p,rhs_{q,\sigma},k,r}(o, l)$ with $o \in occ(rhs_{q,\sigma})$ as the unique occurrence of y_h in $rhs_{q,\sigma}$ (M_1 is non-copying!)

We define the rewriting system Pre, which contains for every $h, s \in \mathbf{N}$ with $1 \leq h < max(rank(Q))$, and every $p \in P^{(s+1)}$, the rule

$p(y_h, z_1, \ldots, z_s) \rightarrow y_{h,p}$.

Note that here only the $z_1, \ldots, z_s$ are considered as variables of the rewriting system, whereas the ys are treated as ordinary symbols.

We define the rewriting system $Pair$, which contains for every $a, s \in \mathbf{N}$, $q \in Q^{(a+1)}$ and $p \in P^{(s+1)}$, the rule

$$p(q(x, y_1, \ldots, y_a), z_1, \ldots, z_s) \to (q,p)(x, nest_q(1, p_1, \emptyset), \ldots, nest_q(a, p_m, \emptyset), z_1, \ldots, z_s) \\ [z_{p_1,1}, \ldots, z_{p_m,s_m} \leftarrow \eta_{p_1,1}, \ldots, \eta_{p_m,s_m}],$$

where for every $s' \in \mathbf{N}$, $p' \in P^{(s'+1)}$ and $l' \in [s']$, we have $\eta_{p',l'} = (\text{if } p' = p \text{ then } z_{l'} \text{ else } nil)$. Note that here x, $y_1, \ldots, y_a$ and $z_1, \ldots, z_s$ are considered as variables of the rewriting system.

Further, note that there are no *critical pairs* [DJ90] in $R_2 \cup Pre \cup Pair$, hence the reduction relation $\Rightarrow_{R_2 \cup Pre \cup Pair}$ is confluent. It is also terminating, hence unique normal forms exist.

Before we introduce the $nest_q$- and $ctx_{p,\phi}$-functions, we define a set of sentential forms, which for now is only necessary in order to give correct types for the introduced functions, but will also be used later in the Appendix. For given sets V_1, Φ and V_2, the set $SEN(V_1, \Phi, V_2)$ is defined as the smallest set SEN satisfying the following conditions:

(1) $V_2 \subseteq SEN$
(2) for every $\omega \in \Omega^{(n)}$ with $n \in \mathbf{N}$, and $\xi_1, \ldots, \xi_n \in SEN$ holds: $\omega(\xi_1, \ldots, \xi_n) \in SEN$
(3) for every $p \in P^{(n+1)}$ with $n \in \mathbf{N}$, $\phi \in \Phi$ and $\xi_1, \ldots, \xi_n \in SEN$ holds: $p(\phi, \xi_1, \ldots, \xi_n) \in SEN$
(4) for every $f \in S^{(n+1)}$ with $n \in \mathbf{N}$, $x \in V_1$ and $\xi_1, \ldots, \xi_n \in SEN$ holds: $f(x, \xi_1, \ldots, \xi_n) \in SEN$

For every $a \in \mathbf{N}_+$, $q \in Q^{(a+1)}$, the function $nest_q : [a] \times P \times \mathcal{P}([a] \times P) \to SEN(\{x\}, Y_a, Z_P)$ is defined as follows. For every $h \in [a]$, $s \in \mathbf{N}$, $p \in P^{(s+1)}$ and $\mathcal{C} \subseteq [a] \times P$:

- $nest_q(h, p, \mathcal{C}) = nil$,
  if $(h,p) \in \mathcal{C}$

- $nest_q(h, p, \mathcal{C}) =$
  $p(y_h, (h_q, 1_p)(x, nest_q(1, p_1, \mathcal{C} \cup \{(h,p)\}), \ldots, nest_q(a, p_m, \mathcal{C} \cup \{(h,p)\}), z_{p_1,1}, \ldots, z_{p_m,s_m}), \ldots, (h_q, s_p)(x, nest_q(1, p_1, \mathcal{C} \cup \{(h,p)\}), \ldots, nest_q(a, p_m, \mathcal{C} \cup \{(h,p)\}), z_{p_1,1}, \ldots, z_{p_m,s_m}))$,
  if $(h,p) \notin \mathcal{C}$

Using the functions defined above, we additionally define for every $k \in \mathbf{N}$, $r \in \mathbf{N}_+$ with $(r+1) \in rank(Q)$, and $\phi \in RHS(Q, \Delta, k, Y_r)$, the following set of functions by mutual recursion:

$$\{ctx_{p,\phi,k,r} : \{o \in occ(\phi) \mid lab(\phi, o) \notin X_k\} \times [s] \to SEN(X_k, RHS(Q, \Delta, k, Y_r), Z_P) \\ \mid s \in \mathbf{N}_+, p \in P^{(s+1)}\}$$

For every $s \in \mathbf{N}_+$, $p \in P^{(s+1)}$, the function $ctx_{p,\phi,k,r}$ is defined as follows. For every $l \in [s]$:

- $ctx_{p,\phi,k,r}(\varepsilon, l) = z_{p,l}$

- For every $i \in \mathbf{N}_+$ and $oi \in occ(\phi)$ with $lab(\phi, oi) \notin X_k$:
  $ctx_{p,\phi,k,r}(oi, l) =$
  - in case $lab(\phi, o) = \delta$ with $a \in \mathbf{N}$, $\delta \in \Delta^{(a)}$ and $i \in [a]$:
    If, with $s' \in \mathbf{N}$, $p' \in P^{(s'+1)}$ and $\psi_1, \ldots, \psi_s \in RHS(P, \Omega, a, Z_{s'})$, the only occurrence of a $p(x_i, \cdots)$-call in the δ-rules of the weakly single-use mtt M_2 looks as follows:

    $$p'(\delta(x_1, \ldots, x_a), z_1, \ldots, z_{s'}) \to \cdots p(x_i, \psi_1, \ldots, \psi_s) \cdots,$$

    then the right-hand side of the equation is:

    $$\psi_l[x_1, \ldots, x_a \leftarrow sub(\phi, o1), \ldots, sub(\phi, oa),\\ z_1, \ldots, z_{s'} \leftarrow ctx_{p',\phi,k,r}(o, 1), \ldots, ctx_{p',\phi,k,r}(o, s')].$$

    If no such call exists in the δ-rules of M_2, then the right-hand side of the equation is nil.
  - in case $lab(\phi, o) = q$ with $a \in \mathbf{N}$, $q \in Q^{(a+1)}$, $2 \leq i \leq a+1$ and $lab(\phi, o1) = x_b \in X_k$:

    $$((i-1)_q, l_p)(x, nest_q(1, p_1, \{(i-1, p)\}), \ldots, nest_q(a, p_m, \{(i-1, p)\}),\\ z_{p_1,1}, \ldots, z_{p_m,s_m})\\ [x \leftarrow x_b,\\ y_1, \ldots, y_a \leftarrow sub(\phi, o2), \ldots, sub(\phi, o(a+1)),\\ z_{p_1,1}, \ldots, z_{p_m,s_m} \leftarrow ctx_{p_1,\phi,k,r}(o, 1), \ldots, ctx_{p_m,\phi,k,r}(o, s_m)].$$

Then, we claim: $\tau(M_1); \tau(M_2) = \tau(M_{1;2})$. □

We will show the correctness of the previous claim in the Appendix.

Note that in the construction we used of the weakly single-use restriction for M_2 only condition (i) from Definition 2.7 and this only for states of M_2 with rank greater than one. In fact, we could generalize the weakly single-use property in Definition 2.7 by dropping condition (ii), and requiring condition (i) only for states p that do have context parameters. This would require a change neither to Construction 3.6 nor to the proofs in the Appendix! The indirect semantic reasoning from the beginning of this section (cf. Theorem 3.1), however, would not be applicable for such a generalized class. Also, the informal explanations from the previous subsection would be more involved and less intuitive for this more general case. That is why we kept to the stronger restriction weakly single-use from the literature [Küh98] and only mention the possible generalization here.

Further, note that we used the non-copying restriction of M_1 and the weakly single-use restriction of M_2 only in the construction of rules for the (h_q, l_p)-states. Since no such states are created if one of the two original mtts is a top-down tree transducer, the composition construction is also applicable if either M_1 or M_2 is a top-down tree transducer and the other one is an unrestricted mtt. In these two special cases, Construction 3.6 corresponds to Transformation 11 in [Küh99] (if M_1 is a top-down tree transducer), respectively, to the construction in the proof of Theorem 4.12 in [EV85] (if M_2 is a top-down tree transducer).

As promised, we now want to explain the crucial part of the construction, namely the working of the $ctx_{p,\phi}$-functions. Recall that—for appropriate k and r—$ctx_{p,\phi,k,r}(o, l)$ shall—modulo reductions with $R_2 \cup Pre \cup Pair$—give a right-hand side expression for $M_{1;2}$, which computes the lth context parameter in a call to state p, reaching the occurrence o in the first mtt's right-hand side expression ϕ. In constructing the right-hand side expression for $M_{1;2}$, we are allowed to use the context parameters $z_{p_1,1}, \ldots, z_{p_m,s_m}$ of all

possible states of M_2 at the root of ϕ, as well as $y_{1,p_1}, \ldots, y_{r,p_m}$, denoting translations of the context parameters $y_1, \ldots, y_r$—possibly occurring in ϕ—with states of M_2. For every $k \in \mathbf{N}$, $r, s \in \mathbf{N}_+$ with $(r+1) \in rank(Q)$, every $\phi \in RHS(Q, \Delta, k, Y_r)$ and $p \in P^{(s+1)}$, the function $ctx_{p,\phi,k,r}$ is defined by case analysis on its first argument, an occurrence in the tree ϕ (the second argument is some $l \in [s]$). If this occurrence is:

- the root of ϕ,
  then the result is the appropriate choice from the $z_{p,1}, \ldots, z_{p,s}$-parameters.
- the ith child of the occurrence of an output symbol $\delta \in \Delta^{(a)}$ in ϕ,
  then we have to look for an occurrence of a $p(x_i, \cdots)$-call in the δ-rules of M_2, because this is the only way, how a call to state p can reach the ith child position of the occurrence of δ. If there is no such call, we can safely output some dummy symbol, because then we are in a position, which cannot be relevant for the computation. If however, there is such a—then necessarily unique—call in a rule $p'(\delta(x_1, \ldots, x_a), z_1, \ldots, z_{s'}) \to \cdots p(x_i, \psi_1, \ldots, \psi_s) \cdots$, then our sought lth context parameter of a call to state p, on reaching the ith child position of δ, is essentially $\psi_l \in RHS(P, \Omega, a, Z_{s'})$, except that it might contain references to the children of the occurrence of δ and to context parameters of the p'-call at δ. The former are obtained as the direct subtrees of the occurrence of δ in ϕ, while the latter can be computed by using the $ctx_{p',\phi,k,r}$-function on the occurrence of δ in ϕ. Everything is plugged together by substitution, and we are done with this case.
- the $(i-1)$st context parameter position of the occurrence of a call to some state $q \in Q^{(a+1)}$ on some $x_b \in X_k$ in ϕ,
  then the result can be computed by calling the state $((i-1)_q, l_p)$ on x_b, because this state was *designed* to compute the lth context parameter in a call to state p, reaching the $(i-1)$st context parameter position of a call to state q. In order to do so, the state $((i-1)_q, l_p)$ has to be provided with translations of q's context parameters—at the particular occurrence of the call to q in ϕ—with states of M_2, and furthermore, with the context parameter values of the second mtt's states at this occurrence (this discussion fulfills the promise from the end of Example 3.5). The latter ones are computed by $ctx_{p_1,\phi,k,r}, \ldots, ctx_{p_m,\phi,k,r}$-function calls on the occurrence of q in ϕ. The translations of q's context parameters with states of M_2 are obtained by replacing in the result of function calls to $nest_q$, the context variables $y_1, \ldots, y_a$ with the subtrees of ϕ in context parameter positions of the occurrence of the call to q. The $nest_q$-function takes care of proper (and finite!) nesting of such translations. Note that the nesting by $nest_q$ is not started with the empty set of "cut-positions", but with $\{(i-1, p)\}$, because we know that the context parameters in a call to state p on reaching the $(i-1)$st context parameter position of a call to state q cannot depend on the p-translation of this $(i-1)$st context parameter (cf. the discussion at the end of Subsection 3.1).

Now, we give as an example the application of the presented construction to the two mtts from Example 3.2. In order to make explicit the positions, where the dummy symbol *nil* is used—indicating that these positions will not influence the computation—, we add *nil* to the output ranked alphabet, instead of using some existing nullary symbol, as we usually would do. The fact that these positions indeed cannot influence the computation, is then verified by arguing that for every possible input, the output tree computed by $M_{1;2}$ will contain no *nil*-symbols.

Example 3.7 Consider M_1 and M_2 from Example 3.2. Obviously, M_1 is non-copying and M_2 is weakly single-use, thus we can apply Construction 3.6 to obtain $M_{1;2} = (S, \Sigma_{mon}, Nat \cup \{\omega^{(2)}, nil^{(0)}\}, e_{1;2}, R_{1;2})$ with components as follows (where $m = 2$, $s_1 = s_2 = 1$ and $Z_P = \{z_{p_1,1}, z_{p_2,1}\}$):

- $S = \{(q_1, p_1)^{(4)}, (q_1, p_2)^{(4)}\} \cup \{(1_{q_1}, 1_{p_1})^{(5)}, (1_{q_1}, 1_{p_2})^{(5)}\}$
- Pre contains the rules:
  $p_1(y_1, z_1) \to y_{1,p_1}$
  $p_2(y_1, z_1) \to y_{1,p_2}$
- $Pair$ contains the rules:
  $p_1(q_1(x, y_1), z_1) \to (q_1, p_1)(x, nest_{q_1}(1, p_1, \emptyset), nest_{q_1}(1, p_2, \emptyset), z_1)[z_{p_1,1}, z_{p_2,1} \leftarrow z_1, nil]$
  $=$
  $(q_1, p_1)(x,$
  $p_1(y_1, (1_{q_1}, 1_{p_1})(x, nil, p_2(y_1, (1_{q_1}, 1_{p_2})(x, nil, nil, z_1, nil)), z_1, nil)),$
  $p_2(y_1, (1_{q_1}, 1_{p_2})(x, p_1(y_1, (1_{q_1}, 1_{p_1})(x, nil, nil, z_1, nil)), nil, z_1, nil)),$
  $z_1)$
  $p_2(q_1(x, y_1), z_1) \to (q_1, p_2)(x, nest_{q_1}(1, p_1, \emptyset), nest_{q_1}(1, p_2, \emptyset), z_1)[z_{p_1,1}, z_{p_2,1} \leftarrow nil, z_1]$
  $=$
  $(q_1, p_2)(x,$
  $p_1(y_1, (1_{q_1}, 1_{p_1})(x, nil, p_2(y_1, (1_{q_1}, 1_{p_2})(x, nil, nil, nil, z_1)), nil, z_1)),$
  $p_2(y_1, (1_{q_1}, 1_{p_2})(x, p_1(y_1, (1_{q_1}, 1_{p_1})(x, nil, nil, nil, z_1)), nil, nil, z_1)),$
  $z_1)$
- $e_{1;2} = nf(\Rightarrow_{R_2 \cup Pair}, p_2(q_1(x_1, \epsilon), zero))$
  $=$ (by $\Rightarrow_{Pair}$)
  $nf(\Rightarrow_{R_2 \cup Pair}, (q_1, p_2)(x_1, p_1(\epsilon, (1_{q_1}, 1_{p_1})(x_1, nil, p_2(\epsilon, (1_{q_1}, 1_{p_2})(x_1, nil, nil, nil, zero)), nil, zero)),$
  $p_2(\epsilon, (1_{q_1}, 1_{p_2})(x_1, p_1(\epsilon, (1_{q_1}, 1_{p_1})(x_1, nil, nil, nil, zero)), nil, nil, zero)),$
  $zero))$
  $=$ (by $\Rightarrow^*_{R_2}$)
  $(q_1, p_2)(x_1, (1_{q_1}, 1_{p_1})(x_1, nil, (1_{q_1}, 1_{p_2})(x_1, nil, nil, nil, zero), nil, zero),$
  $(1_{q_1}, 1_{p_2})(x_1, (1_{q_1}, 1_{p_1})(x_1, nil, nil, nil, zero), nil, nil, zero),$
  $zero)$
- For the rules in $R_{1;2}$, we compute only two examples here:
  - $(q_1, p_1)(A(x_1), y_{1,p_1}, y_{1,p_2}, z_1) \to nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, p_1(rhs_{q_1,A}, z_1))$, with:
    $p_1(\alpha(q_1(x_1, y_1)), z_1)$
    $\Downarrow_{R_2}$
    $p_1(q_1(x_1, y_1), p_2(q_1(x_1, y_1), z_1))$

$\Downarrow_{Pair}$

$(q_1,p_1)(x_1, p_1(y_1,(1_{q_1},1_{p_1})(x_1, nil,$
$\quad p_2(y_1,(1_{q_1},1_{p_2})(x_1,nil,nil,p_2(q_1(x_1,y_1),z_1),nil)),$
$\quad p_2(q_1(x_1,y_1),z_1),nil)),$
$\quad p_2(y_1,(1_{q_1},1_{p_2})(x_1,p_1(y_1,(1_{q_1},1_{p_1})(x_1,nil,nil,p_2(q_1(x_1,y_1),z_1),nil)),$
$\quad nil,$
$\quad p_2(q_1(x_1,y_1),z_1),nil)),$
$\quad p_2(q_1(x_1,y_1),z_1))$

$\Downarrow_{*Pre}$

$(q_1,p_1)(x_1,y_{1,p_1},y_{1,p_2},p_2(q_1(x_1,y_1),z_1))$

$\Downarrow_{Pair}$

$(q_1,p_1)(x_1,y_{1,p_1},y_{1,p_2},$
$\quad (q_1,p_2)(x_1,p_1(y_1,(1_{q_1},1_{p_1})(x_1,nil,p_2(y_1,(1_{q_1},1_{p_2})(x_1,nil,nil,nil,z_1)),$
$\quad nil,z_1)),$
$\quad p_2(y_1,(1_{q_1},1_{p_2})(x_1,p_1(y_1,(1_{q_1},1_{p_1})(x_1,nil,nil,nil,z_1)),nil,$
$\quad nil,z_1)),$
$\quad z_1))$

$\Downarrow_{*Pre}$

$(q_1,p_1)(x_1,y_{1,p_1},y_{1,p_2},(q_1,p_2)(x_1,y_{1,p_1},y_{1,p_2},z_1))$

– $(1_{q_1},1_{p_2})(B(x_1),y_{1,p_1},y_{1,p_2},z_{p_1,1},z_{p_2,1}) \rightarrow nf(\Rightarrow_{R_2\cup Pre\cup Pair}, ctx_{p_2,rhs_{q_1,B},1,1}(21,1))$
with:

$ctx_{p_2,q_1(x_1,\beta(y_1)),1,1}(21,1)$
$\quad =$ (because of rule $p_2(\beta(x_1),z_1)\rightarrow p_2(x_1,\omega(p_1(x_1,z_1),z_1))$ in R_2)
$\omega(p_1(x_1,z_1),z_1)[x_1,z_1 \leftarrow y_1, ctx_{p_2,q_1(x_1,\beta(y_1)),1,1}(2,1)]$
$\quad =$
$\omega(p_1(x_1,z_1),z_1)$
$\quad [x_1 \leftarrow y_1,$
$\quad z_1 \leftarrow (1_{q_1},1_{p_2})(x, nest_{q_1}(1,p_1,\{(1,p_2)\}), nest_{q_1}(1,p_2,\{(1,p_2)\}), z_{p_1,1}, z_{p_2,1})$
$\quad\quad [x,y_1 \leftarrow x_1,\beta(y_1),$
$\quad\quad z_{p_1,1},z_{p_2,1} \leftarrow ctx_{p_1,q_1(x_1,\beta(y_1)),1,1}(\varepsilon,1), ctx_{p_2,q_1(x_1,\beta(y_1)),1,1}(\varepsilon,1)]]$
$\quad =$
$\omega(p_1(x_1,z_1),z_1)$
$\quad [x_1 \leftarrow y_1,$
$\quad z_1 \leftarrow (1_{q_1},1_{p_2})(x, p_1(y_1,(1_{q_1},1_{p_1})(x,nil,nil,z_{p_1,1},z_{p_2,1})), nil, z_{p_1,1}, z_{p_2,1})$
$\quad\quad [x,y_1,z_{p_1,1},z_{p_2,1} \leftarrow x_1,\beta(y_1),z_{p_1,1},z_{p_2,1}]]$
$\quad =$
$\omega(p_1(x_1,z_1),z_1)$
$\quad [x_1 \leftarrow y_1,$
$\quad z_1 \leftarrow (1_{q_1},1_{p_2})(x_1, p_1(\beta(y_1),(1_{q_1},1_{p_1})(x_1,nil,nil,z_{p_1,1},z_{p_2,1})), nil, z_{p_1,1}, z_{p_2,1})]$

And, by reduction:

$\omega(p_1(y_1,(1_{q_1},1_{p_2})(x_1,p_1(\beta(y_1),(1_{q_1},1_{p_1})(x_1,nil,nil,z_{p_1,1},z_{p_2,1})),nil,z_{p_1,1},z_{p_2,1})),$
$\quad (1_{q_1},1_{p_2})(x_1,p_1(\beta(y_1),(1_{q_1},1_{p_1})(x_1,nil,nil,z_{p_1,1},z_{p_2,1})),nil,z_{p_1,1},z_{p_2,1}))$
$\quad \Downarrow_{Pre}$

$\omega(y_{1,p_1}, (1_{q_1}, 1_{p_2})(x_1, p_1(\beta(y_1), (1_{q_1}, 1_{p_1})(x_1, nil, nil, z_{p_1,1}, z_{p_2,1})), nil, z_{p_1,1}, z_{p_2,1}))$
$\Downarrow_{R_2}$
$\omega(y_{1,p_1}, (1_{q_1}, 1_{p_2})(x_1, succ((1_{q_1}, 1_{p_1})(x_1, nil, nil, z_{p_1,1}, z_{p_2,1})), nil, z_{p_1,1}, z_{p_2,1}))$

Note the correspondences between this calculation and the informal explanation in Examples 3.4 and 3.5.

Altogether, we get the following set of rules:

$$
\begin{aligned}
(q_1,p_1)(A(x_1), y_{1,p_1}, y_{1,p_2}, z_1) &\to (q_1,p_1)(x_1, y_{1,p_1}, y_{1,p_2}, (q_1,p_2)(x_1, y_{1,p_1}, y_{1,p_2}, z_1))\\
(q_1,p_1)(B(x_1), y_{1,p_1}, y_{1,p_2}, z_1) &\to (q_1,p_1)(x_1, succ((1_{q_1}, 1_{p_1})(x_1, nil, y_{1,p_2}, z_1, nil)), y_{1,p_2}, z_1)\\
(q_1,p_1)(E, y_{1,p_1}, y_{1,p_2}, z_1) &\to y_{1,p_1}\\
(q_1,p_2)(A(x_1), y_{1,p_1}, y_{1,p_2}, z_1) &\to z_1\\
(q_1,p_2)(B(x_1), y_{1,p_1}, y_{1,p_2}, z_1) &\to (q_1,p_2)(x_1, succ((1_{q_1}, 1_{p_1})(x_1, nil, y_{1,p_2}, nil, z_1)), y_{1,p_2}, z_1)\\
(q_1,p_2)(E, y_{1,p_1}, y_{1,p_2}, z_1) &\to y_{1,p_2}\\
(1_{q_1}, 1_{p_1})(A(x_1), y_{1,p_1}, y_{1,p_2}, z_{p_1,1}, z_{p_2,1}) &\to (1_{q_1}, 1_{p_1})(x_1, nil, y_{1,p_2},\\
&\qquad (q_1,p_2)(x_1, y_{1,p_1}, y_{1,p_2}, z_{p_1,1}), z_{p_1,1})\\
(1_{q_1}, 1_{p_1})(B(x_1), y_{1,p_1}, y_{1,p_2}, z_{p_1,1}, z_{p_2,1}) &\to (1_{q_1}, 1_{p_2})(x_1,\\
&\qquad succ((1_{q_1}, 1_{p_1})(x_1, nil, nil, z_{p_1,1}, z_{p_2,1})),\\
&\qquad nil, z_{p_1,1}, z_{p_2,1})\\
(1_{q_1}, 1_{p_1})(E, y_{1,p_1}, y_{1,p_2}, z_{p_1,1}, z_{p_2,1}) &\to z_{p_1,1}\\
(1_{q_1}, 1_{p_2})(A(x_1), y_{1,p_1}, y_{1,p_2}, z_{p_1,1}, z_{p_2,1}) &\to (1_{q_1}, 1_{p_2})(x_1, y_{1,p_1}, nil,\\
&\qquad (q_1,p_2)(x_1, y_{1,p_1}, y_{1,p_2}, z_{p_1,1}), z_{p_1,1})\\
(1_{q_1}, 1_{p_2})(B(x_1), y_{1,p_1}, y_{1,p_2}, z_{p_1,1}, z_{p_2,1}) &\to \omega(y_{1,p_1}, (1_{q_1}, 1_{p_2})(x_1,\\
&\qquad succ((1_{q_1}, 1_{p_1})(x_1, nil, nil, z_{p_1,1}, z_{p_2,1})),\\
&\qquad nil, z_{p_1,1}, z_{p_2,1}))\\
(1_{q_1}, 1_{p_2})(E, y_{1,p_1}, y_{1,p_2}, z_{p_1,1}, z_{p_2,1}) &\to z_{p_2,1}
\end{aligned}
$$

□

Now, we want to use Example 3.7 to illustrate some points that have been discussed abstractly in Subsection 3.1.

1. Firstly, we focus on the rule constructed for (q_1, p_1) at B, with right-hand side:
$$(q_1,p_1)(x_1, succ((1_{q_1}, 1_{p_1})(x_1, nil, y_{1,p_2}, z_1, \underline{nil})), y_{1,p_2}, z_1),$$
containing two occurrences of nil. The second one—in the $z_{p_2,1}$-position of state $(1_{q_1}, 1_{p_1})$—is of the kind introduced into the expression **(***)** on page 15, where we promised the example given now. We can by no means assume that state $(1_{q_1}, 1_{p_1})$ will never need its last context parameter position, in fact we have that $(1_{q_1}, 1_{p_1})(B(E), y_{1,p_1}, y_{1,p_2}, z_{p_1,1}, z_{p_2,1})$ reduces to $z_{p_2,1}$. Instead, we have to argue according to the place, where the $(1_{q_1}, 1_{p_1})$-call occurs, namely in the first context parameter position of a (q_1, p_1)-call on the same tree argument. Thus, we need to explain why there can be no input tree t, such that (q_1, p_1) called on t depends on its y_{1,p_1}-position and $(1_{q_1}, 1_{p_1})$ called on t depends on its $z_{p_2,1}$-position. We reason intuitively as follows. If the normal form of $(q_1, p_1)(t, y_{1,p_1}, y_{1,p_2}, z_1)$ contains y_{1,p_1}, this means that reducing $p_1(q_1(t, y_1), z_1)$ will lead to a call to p_1 on y_1. By the weakly single-use property of M_2, we have argued that there can be only one state p' such that reducing $p'(q_1(t, y_1), \cdots)$ leads to a p_1-call on y_1. For our particular t, we know that $p' = p_1$. But then, $(1_{q_1}, 1_{p_1})$ called on t (which computes the first context parameter in a call to p_1 on the first context variable of q_1, resulting from reduction of $p'(q_1(t, y_1), \cdots)$) will not depend on $z_{p_2,1}$, because

$p' \neq p_2$, and—as indicated in the discussion above Example 3.5—the state $(1_{q_1}, 1_{p_1})$ called on t will of its $z_{p_1,1}, z_{p_2,1}$-positions only depend on those, which are associated with p'. Note that—with $p' = p_2$—the situation is analogous for the *nil*-symbol in the $z_{p_1,1}$-position of the $(1_{q_1}, 1_{p_1})$-call in the right-hand side for state (q_1, p_2) at B: $(q_1, p_2)(x_1, succ((1_{q_1}, 1_{p_1})(x_1, nil, y_{1,p_2}, \underline{nil}, z_1)), y_{1,p_2}, z_1)$.

2. Now, we want to consider the *nil*-symbols, which occur as first context parameter of $(1_{q_1}, 1_{p_1})$-calls in several rules and the initial expression:
$$(q_1, p_2)(x_1, (1_{q_1}, 1_{p_1})(x_1, \underline{nil}, (1_{q_1}, 1_{p_2})(x_1, nil, nil, nil, zero), nil, zero), (1_{q_1}, 1_{p_2})(x_1, (1_{q_1}, 1_{p_1})(x_1, \underline{nil}, nil, nil, zero), nil, nil, zero), zero)$$
These are justified by the fact that for no input tree t, the state $(1_{q_1}, 1_{p_1})$ depends on its y_{1,p_1}-position (analogous for state $(1_{q_1}, 1_{p_2})$ and y_{1,p_2}). This proposition holds, because $(1_{q_1}, 1_{p_1})$ on t computes the first context parameter position of a p_1-call on y_1, resulting from reduction of some $p'(q_1(t, y_1), \cdots)$. If this value would depend on the p_1-translation of y_1, we would have met during the reduction a situation like $p_1(y_1, \cdots p_1(y_1, \cdots) \cdots)$, which we argued—at the end of Subsection 3.1—to be impossible for non-copying M_1 and weakly single-use M_2.

3. Now, we explain the *nil*-symbol in the y_{1,p_2}-position of the second $(1_{q_1}, 1_{p_1})$-call in the initial expression:
$$(q_1, p_2)(x_1, (1_{q_1}, 1_{p_1})(x_1, nil, (1_{q_1}, 1_{p_2})(x_1, nil, nil, nil, zero), nil, zero), (1_{q_1}, 1_{p_2})(x_1, (1_{q_1}, 1_{p_1})(x_1, nil, \underline{nil}, nil, zero), nil, nil, zero), zero),$$
which stems from the "cutting" of cycles during the nesting process, described at the end of Subsection 3.1. Note that in general $(1_{q_1}, 1_{p_1})$ might very well depend on its second context parameter, e.g., $(1_{q_1}, 1_{p_1})(A(E), y_{1,p_1}, y_{1,p_2}, z_{p_1,1}, z_{p_2,1})$ reduces to y_{1,p_2}. But, we argue that there can be no input tree t, such that $(1_{q_1}, 1_{p_2})$ called on t depends on its y_{1,p_1}-position and $(1_{q_1}, 1_{p_1})$ called on t depends on its y_{1,p_2}-position, by reasoning intuitively as follows. If the normal form of $(1_{q_1}, 1_{p_2})(t, y_{1,p_1}, y_{1,p_2}, z_{p_1,1}, z_{p_2,1})$ contains y_{1,p_1}, this means that the first context parameter of every p_2-call on y_1, resulting from reduction of $p'(q_1(t, y_1), \cdots)$ for some p', depends on the p_1-translation of y_1, in fact a situation like $p_2(y_1, \cdots p_1(y_1, ?) \cdots)$ is met during this reduction. The call to $(1_{q_1}, 1_{p_1})$ on t computes exactly the context parameter of this call to p_1 on y_1, i.e., the value in the question mark position. But, if this value would depend on the y_{1,p_2}-position of $(1_{q_1}, 1_{p_1})$, then the question mark position would in fact contain some value depending on the p_2-translation of y_1, giving as result of the above reduction a normal form containing a situation like $p_2(y_1, \cdots p_1(y_1, \cdots p_2(y_1, \cdots) \cdots) \cdots)$. This clearly contradicts the statement that—if M_1 is non-copying and M_2 is weakly single-use—every two calls to p_2 on y_1 will have identical context parameters. Analogous reasoning applies to the second *nil*-symbol in the initial expression.

4. Finally, we turn to the right-hand side for state $(1_{q_1}, 1_{p_2})$ at B and the *nil*-symbol, which occurs there in the second context parameter position of the recursive call to $(1_{q_1}, 1_{p_1})$:
$$\omega(y_{1,p_1}, (1_{q_1}, 1_{p_2})(x_1, succ((1_{q_1}, 1_{p_1})(x_1, nil, \underline{nil}, z_{p_1,1}, z_{p_2,1})), nil, z_{p_1,1}, z_{p_2,1}))$$
To obtain this right-hand side, we computed $ctx_{p_2,q_1(x_1,\beta(y_1)),1,1}(2, 1)$ as defined in

the last case for the computation of $ctx_{p,\phi}$-functions in Construction 3.6. The *nil*-symbol is created, because the nesting is started with the set of "cut-positions" $\{(1, p_2)\}$, as justified in the last case of explaining the working of $ctx_{p,\phi}$-functions above Example 3.7. It will never appear in the output, because—as argued above in point 3.—there is no input tree t, such that $(1_{q_1}, 1_{p_2})$ called on t depends on its first context parameter and $(1_{q_1}, 1_{p_1})$ called on t depends on its second context parameter. Similarly for the right-hand side for state $(1_{q_1}, 1_{p_1})$ at B.

After having discussed the previous rather artificial example at such length, let us conclude this section with a more practical example, namely the one from the introduction.

Example 3.8 Composition of M_{pre} and M_{aux} from Examples 2.3 and 2.10, respectively, according to Construction 3.6 yields $M_{pre;aux} = (S_{pre;aux}, \Sigma_{term}, \Omega_{ins}, e_{pre;aux}, R_{pre;aux})$ with $S_{pre;aux} = \{(pre, aux)^{(3)}, (1_{pre}, 1_{aux})^{(3)}\}$, set of rules $R_{pre;aux}$:

$$
\begin{array}{lll}
\text{(i)} & : (pre, aux)(+(x_1, x_2), y, z) & \to (pre, aux)(x_1, (pre, aux)(x_2, y, \\
 & & \qquad (1_{pre}, 1_{aux})(x_1, \epsilon, ADD(z))), ADD(z)) \\
\text{(ii)} & : (pre, aux)(*(x_1, x_2), y, z) & \to (pre, aux)(x_1, (pre, aux)(x_2, y, \\
 & & \qquad (1_{pre}, 1_{aux})(x_1, \epsilon, MUL(z))), MUL(z)) \\
\text{(iii)} & : (pre, aux)(A, y, z) & \to y \\
\text{(iv)} & : (pre, aux)(B, y, z) & \to y \\
\text{(v)} & : (1_{pre}, 1_{aux})(+(x_1, x_2), y, z) & \to (1_{pre}, 1_{aux})(x_2, \epsilon, (1_{pre}, 1_{aux})(x_1, \epsilon, ADD(z))) \\
\text{(vi)} & : (1_{pre}, 1_{aux})(*(x_1, x_2), y, z) & \to (1_{pre}, 1_{aux})(x_2, \epsilon, (1_{pre}, 1_{aux})(x_1, \epsilon, MUL(z))) \\
\text{(vii)} & : (1_{pre}, 1_{aux})(A, y, z) & \to LOAD_A(z) \\
\text{(viii)} & : (1_{pre}, 1_{aux})(B, y, z) & \to LOAD_B(z)
\end{array}
$$

and $e_{pre;aux} = (pre, aux)(x_1, (1_{pre}, 1_{aux})(x_1, \epsilon, \epsilon), \epsilon)$. □

4 Aftermath

This section is divided into four subsections. The first considers inefficiencies that might still be present in an mtt obtained by the composition construction. Then, we will show that a symmetric composition result $MAC_{wsu}; MAC_{nc} \subseteq MAC$ does not hold. The third subsection uses Theorem 3.1 to show that for every non-copying mtt, the size of the output tree associated to an input tree is exponentially bounded by the input tree's height. Finally, we will show an example application of an implementation of the composition construction and post-processing steps in a program transformation system.

4.1 Tidying up

The mtt $M_{pre;aux}$, as constructed in Example 3.8, looks rather complicated. In particular, it is not quite the optimized program that we promised in the introduction. The reason is twofold. Firstly, we observe that the construction introduces context parameters, which are *superfluous*, in the sense that they will never (for no possible input tree) influence the output generated by a state. This is the case for parameter z of state (pre, aux) and parameter y of state $(1_{pre}, 1_{aux})$. Secondly, the state (pre, aux) never really performs any meaningful computation. No matter, on which input tree it is called, it will always just project on its first context parameter. We call a state showing such a behavior a *copy-state*, because this phenomenon is similar to superfluous data traversals due to *copy rules* of attribute grammars in [CDPR99].

The first problem is caused by the need for the composition construction to be as general as possible. There might actually be more complicated mtts, where the additionally created context parameters are really needed. However, the presence of these superfluous context parameters does not influence the efficiency of our constructed mtts, if lazy evaluation is used and we take the number of reduction steps as efficiency measure. Nevertheless, we still want to get rid of them, because they obscure the programs and, moreover, *do* influence the efficiency, if more detailed measures are used, such as taking into account the cost of allocating memory for representing function closures.

The second problem is more serious, as it leads to superfluous traversals through the input data structure and so contradicts our aim of *optimizing* functional programs by eliminating intermediate data structures.

In our example $M_{pre;aux}$, it is pretty obvious how to solve the two problems, but in general, both, superfluous context parameters and copy-states, are more difficult to detect. In [Voi01], two constructions (based on finding finite fixpoints) were developed for post-processing and optimizing mtts obtained from the composition construction, which allow to detect and remove all superfluous context parameters and copy-states, respectively.

Example 4.1 Using the two proposed constructions from [Voi01] on $M_{pre;aux}$, we obtain the mtt $M'_{pre;aux} = (S'_{pre;aux}, \Sigma_{term}, \Omega_{ins}, e'_{pre;aux}, R'_{pre;aux})$ with $S'_{pre;aux} = \{(1_{pre}, 1_{aux})^{(2)}\}$, set of rules $R'_{pre;aux}$:

$$\begin{array}{lll}
\text{(i)} & : & (1_{pre}, 1_{aux})(+(x_1, x_2), z) \rightarrow (1_{pre}, 1_{aux})(x_2, (1_{pre}, 1_{aux})(x_1, ADD(z))) \\
\text{(ii)} & : & (1_{pre}, 1_{aux})(*(x_1, x_2), z) \rightarrow (1_{pre}, 1_{aux})(x_2, (1_{pre}, 1_{aux})(x_1, MUL(z))) \\
\text{(iii)} & : & (1_{pre}, 1_{aux})(A, z) \rightarrow LOAD_A(z) \\
\text{(iv)} & : & (1_{pre}, 1_{aux})(B, z) \rightarrow LOAD_B(z)
\end{array}$$

and $e'_{pre;aux} = (1_{pre}, 1_{aux})(x_1, \epsilon)$. □

Note that this indeed corresponds to the optimized program of rules (x)–(xiii) in the introduction. Strictly speaking, the elimination of copy-states is sound only for computations on finite trees, because when used with infinite data structures—as they are possible in lazy functional languages—it can transform non-terminating programs into terminating ones (for a discussion, see [Voi01]).

4.2 The symmetric Result does not hold

In Subsection 3.2 we have presented a construction, which composes a non-copying mtt and a weakly single-use one to a single mtt. We might try to intuitively understand that this is possible, because the first mtt forbids the copying of context information, whereas the second one forbids multiple use of recursive computations on subtrees. It is thus naturally to ask, whether these two restrictions of controlling the copying of context information and recursive computations on subtrees, respectively, also work the other way round, that is, whether we also have $MAC_{wsu}; MAC_{nc} \subseteq MAC$. In this subsection, we will show that this is not the case, by giving a counterexample. In fact, we will even show the stronger result that $MAC_{su}; MAC_{nc} \not\subseteq MAC$.

Example 4.2

- Let $\Sigma_{bin} = \{\sigma^{(2)}, \alpha^{(0)}, \beta^{(0)}\}$ and consider the single-use mtt $M_{\Sigma_{bin}} = (\{s^{(2)}\}, \Sigma_{bin}, Nat, s(x_1, zero), R_{\Sigma_{bin}})$ with set of rules $R_{\Sigma_{bin}}$:

(i) : $s(\sigma(x_1, x_2), y_1) \to succ(s(x_1, s(x_2, y_1)))$
(ii) : $s(\alpha, y_1) \to succ(y_1)$
(iii) : $s(\beta, y_1) \to succ(y_1)$

We claim that for every $t \in T_{\Sigma_{bin}}$, we have: $height(\tau(M_{\Sigma_{bin}})(t)) = size(t)$.

- Consider the non-copying mtt $M_{exp} = (\{exp^{(2)}\}, Nat, Nat, exp(x_1, zero), R_{exp})$ with set of rules R_{exp}:

  (i) : $exp(succ(x_1), y_1) \to exp(x_1, exp(x_1, y_1))$
  (ii) : $exp(zero, y_1) \to succ(y_1)$

  We claim that for every $t \in T_{Nat}$, we have: $height(\tau(M_{exp})(t)) = 2^{height(t)}$.

The claim about $M_{\Sigma_{bin}}$ is an instance of a more general construction for arbitrary ranked alphabets in Subsection 4.3. The claim about M_{exp} can easily be proven by induction. □

Here, we quote a classical result on mtts (Theorem 3.24 in [EV85]), namely that the heights of their output trees are exponentially bounded by the heights of their input trees.

Lemma 4.3 Let $M = (Q, \Sigma, \Delta, e, R)$ be an mtt. There exists a constant $c \in \mathbf{N}$, such that for every input tree $t \in T_\Sigma$ holds: $height(\tau(M)(t)) \leq c^{height(t)}$. □

The result can now be shown by counterexample.

Theorem 4.4 $MAC_{su}; MAC_{nc} \not\subseteq MAC$ □

Proof: Consider the mtts $M_{\Sigma_{bin}}$ and M_{exp} from Example 4.2, which are single-use and non-copying, respectively, and assume the existence of an mtt M, such that $\tau(M) = \tau(M_{\Sigma_{bin}}); \tau(M_{exp})$. Then, for every $t \in T_{\Sigma_{bin}}$ holds: $\tau(M)(t) = \tau(M_{exp})(\tau(M_{\Sigma_{bin}})(t))$. According to Lemma 4.3, there must exist a constant $c \in \mathbf{N}$, such that $height(\tau(M)(t)) \leq c^{height(t)}$ for every such t. We calculate:

$height(\tau(M)(t)) =$
$height(\tau(M_{exp})(\tau(M_{\Sigma_{bin}})(t))) =$
$2^{height(\tau(M_{\Sigma_{bin}})(t))} =$
$2^{size(t)}$

Thus, there has to be a constant $c \in \mathbf{N}$, such that for every $t \in T_{\Sigma_{bin}}$, we have $2^{size(t)} \leq c^{height(t)}$. Taking for every $h \in \mathbf{N}$, for t the fully balanced binary tree of height h over the ranked alphabet Σ_{bin}, we get $size(t) = 2^{h+1} - 1$ and thus $2^{(2^{h+1}-1)} \leq c^h$. But, due to standard function analysis, no c exists such that this inequation holds for every $h \in \mathbf{N}$. □

4.3 A Size-Height Property for non-copying Macro Tree Transducers

In the previous subsection we quoted the classical result that for every tree transduction induced by an mtt, the height of the output tree is exponentially bounded by the height of the input tree. Lemmata of the same style—relating input- and output-height or -size—are also available for restricted mtts (and will be summarized in a table below), but not yet for the class of tree transductions induced by non-copying mtts. It would be useful to know such a lemma, because—as demonstrated in the proof of Theorem 4.4—these kinds of results can help in reasoning about the expressiveness of various classes of tree transducers.

By using the composition result $MAC_{nc}; MAC_{wsu} \subseteq MAC$, we are going to show that for non-copying mtts, the output-*size* is exponentially bounded by the input-*height*. In

order to do so, we will construct, for every ranked alphabet Δ, a weakly single-use mtt M_Δ, which computes the sizes of trees over Δ as natural numbers in monadic representation over the ranked alphabet *Nat*.

Construction 4.5 Let Δ be a ranked alphabet. We construct the mtt $M_\Delta = (\{s^{(2)}\}, \Delta, Nat, s(x_1, zero), R_\Delta)$, where for every $k \in \mathbf{N}$, $\delta \in \Delta^{(k)}$, the set R_Δ contains the following rule:

$$s(\delta(x_1, \ldots, x_k), y_1) \to succ(\zeta_k(k, y_1)).$$

For $k \in \mathbf{N}$, the function $\zeta_k : (\{0\} \cup [k]) \times RHS(\{s^{(2)}\}, Nat, k, Y_1) \to RHS(\{s^{(2)}\}, Nat, k, Y_1)$ is defined as follows. For every $\phi \in RHS(\{s^{(2)}\}, Nat, k, Y_1)$:

$$\begin{aligned} \zeta_k(0, \phi) &= \phi \\ \zeta_k(n, \phi) &= \zeta_k(n-1, s(x_n, \phi)), \qquad \text{if } n \in [k] \end{aligned}$$

We claim that M_Δ is weakly single-use (it is also non-copying and thus single-use, but we do not need this fact here) and that for every $t \in T_\Delta$, we have: $height(\tau(M_\Delta)(t)) = size(t)$. The claims can be proven by a straightforward induction, to be found in [Voi01]. □

Example 4.6 For the ranked alphabet Σ_{bin}, we obtain the mtt $M_{\Sigma_{bin}}$ in Example 4.2. □

Now, we can give the size-height property for non-copying mtts.

Lemma 4.7 Let $M_{nc} = (Q, \Sigma, \Delta, e, R)$ be a non-copying mtt. There exists a constant $c \in \mathbf{N}$, such that for every input tree $t \in T_\Sigma$ holds: $size(\tau(M_{nc})(t)) \leq c^{height(t)}$. □

Proof: Since M_{nc} is non-copying and M_Δ is weakly single-use, there exists—by Theorem 3.1, respectively, Construction 3.6—an mtt M, such that we have $\tau(M_{nc}); \tau(M_\Delta) = \tau(M)$, with $\tau(M) : T_\Sigma \to T_{Nat}$. For M we know, by Lemma 4.3, that there exists a constant $c \in \mathbf{N}$, such that for every $t \in T_\Sigma$ holds:

$$height(\tau(M)(t)) \leq c^{height(t)}.$$

This gives us, by $\tau(M)(t) = \tau(M_\Delta)(\tau(M_{nc})(t))$:

$$height(\tau(M_\Delta)(\tau(M_{nc})(t))) \leq c^{height(t)}.$$

And, by $height(\tau(M_\Delta)(t')) = size(t')$ for every $t' \in T_\Delta$:

$$size(\tau(M_{nc})(t)) \leq c^{height(t)}.$$ □

Thus, we get the following summary of input-output boundedness for classes of tree transductions:

MAC	: output-*height*	*exponentially*	bounded by input-*height*
MAC_{nc}	: output-*size*	*exponentially*	bounded by input-*height*
MAC_{wsu}	: output-*height*	*linearly*	bounded by input-*size*
MAC_{su}	: output-*size*	*linearly*	bounded by input-*size*
TOP	: output-*height*	*linearly*	bounded by input-*height*

The results for TOP and MAC are known from [EV85]. The bound for MAC_{wsu} follows from Theorem 7.1 of [Küh98] and the corresponding boundedness property for attributed tree transducers [Fül81]. The result for MAC_{su} is a consequence of Theorem 7.2 of [Küh98] and a boundedness property for single-use attributed tree transducers [Gie88, Küh97]. Finally, the bound for non-copying mtts has been proven in this subsection.

4.4 An Implementation

Construction 3.6 has been implemented in the Haskell$^+$ program transformation system [Les99, HVM$^+$01]. As an example for applying the system, we consider the following program.

Example 4.8

```
begindata Data
  data List = A List | B List | E
enddata

beginmag App [Mac,Mat,Su,Swp,Tl,Wp,Wsu,Xlin,Xnd,Ylin,Ynd]
  input Data
  syn app :: List -> List -> List
  app (A x) y = A (app x y)
  app (B x) y = B (app x y)
  app E     y = y
endmag

{- COMPOSITION: replace function calls as follows
  app (app x1 y1) z1 = (comp_app_o_app x1 (app y1 (comp_1app_o_1app x1 E z1)) z1)
-}
{- REMOVE SUPERFLUOUS CONTEXT PARAMETERS: replace function calls as follows
  comp_app_o_app   t y1 y2 = comp_app_o_app_   t y1
  comp_1app_o_1app t y1 y2 = comp_1app_o_1app_ t y2
-}
{- ELIMINATE COPY-STATES: replace function calls as follows
  comp_1app_o_1app_ t y = y
-}
beginmag App_o_App [Mac]
  input Data
  syn comp_app_o_app_ :: List -> List -> List
  comp_app_o_app_ (A x1) y1 = A (comp_app_o_app_ x1 y1)
  comp_app_o_app_ (B x1) y1 = B (comp_app_o_app_ x1 y1)
  comp_app_o_app_ E      y1 = y1
endmag
```

□

The input to the system consisted just of the first lines from `begindata` to `endmag`, containing the definition of a Haskell data type `List` and a function `app`, which computes concatenation on the `List` type. The program also contains special keywords like `enddata`, `beginmag`, `input` and `syn` that are used in the Haskell$^+$ language to specify ranked alphabets (in `begindata`–`enddata` blocks) and tree transducers (in `beginmag`–`endmag` blocks).

The system was instructed to analyze the input program and to compose the mtt `App` with itself, followed by post-processing steps as mentioned in Subsection 4.1. The analyzing phase outputs a list of properties behind the "`beginmag App`" statement. In particular, the list items `Mac`, `Ylin` and `Wsu` tell us that we have an mtt, which is non-copying and weakly single-use, respectively. Thus, the composition construction presented in Subsection 3.2 was indeed applicable and led to the introduction of a new mtt `App_o_App` into the output program. Also, the Haskell$^+$ system outputs—in comments above the resulting mtt—a list of equations that can be used as rewrite rules in order to take advantage of the composition and post-processing transformations. An optimizing compiler would have to detect appropriate places in the program, where these equations should be applied. For

example, one could use *rule pragmas* of the Glasgow Haskell Compiler [PTH01] to leave this task to the built-in simplifier.

Note that in the `COMPOSITION`-comment, the equation's right-hand side contains an occurrence of the state `app` (originating from the second of the two composed mtts), because its left-hand side is more general than just a composite of two mtts' initial expressions, in that it is not only variable over the input tree `x1`, but also over the context parameters `y1` and `z1`. This equation corresponds to the rewriting system *Pair* from Construction 3.6. Its correctness can be justified by appealing to statement **(II)(a)(i)** in the main Lemma A.12 of the Appendix.

Applying the equations introduced in the comments, from left to right, we get the following calculation:

```
app (app x1 y1) z1
  = (by the comment COMPOSITION)
(comp_app_o_app x1 (app y1 (comp_1app_o_1app x1 E z1)) z1)
  = (by the comment REMOVE SUPERFLUOUS CONTEXT PARAMETERS)
(comp_app_o_app_ x1 (app y1 (comp_1app_o_1app_ x1 z1)))
  = (by the comment ELIMINATE COPY-STATES)
(comp_app_o_app_ x1 (app y1 z1))
```

Since the defining equations, which were constructed for `comp_app_o_app_`, are essentially the same as those for `app`, this corresponds to a well-known optimizing transformation, namely making use of the associativity of the concatenation function.

5 Future Work

In this paper, we have presented a direct construction that composes a non-copying macro tree transducer and a weakly single-use macro tree transducer, which can be regarded as special functional programs. Since every macro tree transduction can also be realized by the consecutive execution of a non-copying macro tree transducer and a weakly single-use one, we obtain the characterization $MAC = MAC_{nc}; MAC_{wsu}$.

Thus, we have broadened the applicability of a technique—first proposed in [Küh97, Küh98]—for eliminating intermediate data structures—including those built up in accumulating parameters—in functional programs. However, we have not formally considered efficiency concerns in this paper. In fact, there are examples, for which a lazy evaluation strategy on the original program—using an intermediate result—performs less reduction steps than on the program obtained by eliminating the intermediate result with the proposed techniques. It is thus important for future work to develop decision procedures that determine, when a composition technique should be applied, or to identify subclasses of macro tree transducers, for which the application of a certain composition technique is guaranteed to improve the efficiency of programs. For the composition results $TOP; MAC \subseteq MAC$ and $MAC; TOP \subseteq MAC$, such a systematic study has begun in [Küh99] and [Höf99], where also a formal comparison between these techniques and classical deforestation [Wad90, Chi94] is drawn. Also other transformation techniques, such as shortcut deforestation [GLP93, Gil96] and *program calculation* techniques based on *recursion operators* [MFP91, SF93, BdM96, HIT96, Jür00], are interesting for a comparison.

Another possibility of further investigation is to further relax the restrictions needed to compose two macro tree transducers or to find other restrictions that enable such a composition. The observation below Construction 3.6 is one step into that direction. We think that the question raised in Figure 2 should be central in looking for sufficient conditions that allow a composition. Also, we have already noted that for well-presented macro tree transducers [CF82, FV98], we get the characterization $MAC_{nc}; MAC_{wp} = MAC$. We believe that a direct composition construction for this setting can be given based on a similar idea as the one exploited in the main construction of the present paper.

Also, we consider it fruitful to further investigate composition constructions for macro attributed tree transducers [KV94, Küh98]. These could give us interesting transformation techniques for higher-order functional programs, the basic idea being that—under appropriate restrictions—functions that use context parameters of higher-order type can be transformed into macro attributed tree transducers, similarly to direct translations of functions with context parameters of first-order type into attributed tree transducers [CF82, Höf98, EM99].

Further, we would like to integrate the presented composition construction and related techniques into an optimizing functional compiler. Compared to an implementation of classical deforestation, this would require the overhead of an analysis phase to detect macro tree transducers in source programs, but advantageously the composition construction itself is local and does not need a bookkeeping of the transformation history for ensuring termination.

Acknowledgment

We would like to thank Andreas Maletti for providing an implementation of the composition construction in the Haskell$^+$ system.

References

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

[BdM96] R.S. Bird and O. de Moor. *Algebra of Programming.* International Series in Computer Science. Prentice Hall, 1996.

[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, Paris, France, Proceedings*, volume 1702 of *LNCS*, pages 360–377. Springer-Verlag, September 1999.

[CF82] B. Courcelle and P. Franchi–Zannettacci. Attribute grammars and recursive program schemes. *Theoret. Comput. Sci.*, 17:163–191, 235–257, 1982.

[Chi94] W.N. Chin. Safe fusion of functional expressions II: Further improvements. *J. Funct. Prog.*, 4:515–555, 1994.

[DJ90] N. Dershowitz and J.P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 6, 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. *Inform. and Comput.*, 154:34–91, 1999. Also: Leiden University Technical Report, 98-09, August 1998.

[Eng75] J. Engelfriet. Bottom–up and top–down tree transformations — a comparison. *Math. Syst. 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. New York, Academic Press, 1980.

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

[EV85] J. Engelfriet and H. Vogler. Macro tree transducers. *J. Comput. Syst. Sci.*, 31:71–145, 1985.

[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.* Monographs in Theoretical Computer Science, An EATCS Series. Springer-Verlag, 1998.

[Gan83] H. Ganzinger. Increasing modularity and language–independency in automatically generated compilers. *Sci. of Comput. Prog.*, 3:223–278, 1983.

[GG84] H. Ganzinger and R. Giegerich. Attribute coupled grammars. In *Symposium on Compiler Construction, Montreal, Canada, Proceedings*, volume 19 of *ACM SIGPLAN Notices*, pages 157–170, June 1984.

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

[Gil96] A. Gill. *Cheap Deforestation for Non-strict Functional Languages.* PhD thesis, University of Glasgow, 1996.

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

[HIT96] Z. Hu, H. Iwasaki, and M. Takeichi. Deriving structural hylomorphisms from recursive definitions. In *International Conference on Functional Programming, Philadelphia, Pennsylvania, Proceedings*, pages 73–82. ACM Press, May 1996.

[Höf98] M. Höff. Vergleich der Berechnungsstärken von Klassen eingeschränkter Tree Transducer. Großer Beleg, Dresden University of Technology, 1998.

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

[HVM+01] M. Höff, R. Vater, A. Maletti, A. Kühnemann, and J. Voigtländer. Tree transducer based program transformations for Haskell$^+$, March 2001. Progress report, Dresden University of Technology.
http://wwwtcs.inf.tu-dresden.de/∼voigt/hpgreport.ps.gz.

[Jür00] C. Jürgensen. A formalization of hylomorphism based deforestation with an application to an extended typed λ-calculus. Technical Report TUD-FI00-13, Dresden University of Technology, 2000.
http://wwwtcs.inf.tu-dresden.de/∼claus/publications/tud00-13.ps.gz.

[KGF01] K. Kakehi, R. Glück, and Y. Futamura. A note on deforesting parameters of accumulating maps. Manuscript, Waseda University, Tokyo, Japan, 2001.

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

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

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

[Küh99] A. Kühnemann. Comparison of deforestation techniques for functional programs and for tree transducers. In *Fuji International Symposium on Functional and Logic Programming, Tsukuba, Japan, Proceedings*, volume 1722 of *LNCS*, pages 114–130. Springer-Verlag, November 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.
http://wwwtcs.inf.tu-dresden.de/∼voigt/TUD-FI01-07.ps.gz.

[Les99] C. Lescher. Entwurf und Implementierung einer Eingabesprache für ein System zur Erzeugung syntaxgesteuerter Editoren. Großer Beleg, Dresden University of Technology, 1999.

[Mal89] G. Malcolm. Homomorphisms and promotability. In *Mathematics of Program Construction, Groningen, The Netherlands, Proceedings*, volume 375 of *LNCS*, pages 335–347. Springer-Verlag, June 1989.

[MFP91] E. Meijer, M. Fokkinga, and R. Paterson. Functional programming with bananas, lenses, envelopes and barbed wire. In *Functional Programming Languages and Computer Architecture, Cambridge, Massachusetts, Proceedings*, volume 523 of *LNCS*, pages 124–144. Springer-Verlag, August 1991.

[PTH01] S.L. Peyton Jones, A. Tolmach, and T. Hoare. Playing by the rules: Rewriting as a practical optimisation technique in GHC. 2001. To appear at the ACM SIGPLAN Haskell Workshop.

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

[SF93] T. Sheard and L. Fegaras. A fold for all seasons. In *Functional Programming Languages and Computer Architecture, Copenhagen, Denmark, Proceedings*, pages 233–242. ACM Press, June 1993.

[SGJ96] M.H. Sørensen, R. Glück, and N.D. Jones. A positive supercompiler. *J. Funct. Prog.*, 6:811–838, 1996.

[SS99] J.P. Secher and M.H. Sørensen. On perfect supercompilation. In *Perspectives of System Informatics, Novosibirsk, Russia, Proceedings*, volume 1755 of *LNCS*, pages 113–127. Springer-Verlag, July 1999.

[Tha70] J.W. Thatcher. Generalized^2 sequential machine maps. *J. Comput. Syst. Sci.*, 4:339–367, 1970.

[Tur86] V.F. Turchin. The concept of a supercompiler. *ACM Trans. on Prog. Lang. and Systems*, 8:292–325, 1986.

[Voi01] J. Voigtländer. Composition of restricted macro tree transducers. Master's thesis, Dresden University of Technology, March 2001.
http://wwwtcs.inf.tu-dresden.de/∼voigt/mscthesis.ps.gz (revised version).

[Wad90] P. Wadler. Deforestation: Transforming programs to eliminate trees. *Theoret. Comput. Sci.*, 73:231–248, 1990.

Appendix: Correctness Proof

In this appendix we show that Construction 3.6 indeed produces an mtt, which is equivalent to the composition of the two given ones. We proceed in two stages, similar to the way we introduced the construction. Firstly, we will formalize the idea of "walking upwards" in the intermediate result to obtain the context parameters of calls to the second mtt's states on the context parameters of the first mtt, as presented in Subsection 3.1. Therefore, we introduce functions that we claim give answers to the question marks in Figure 2. We will prove this claim and present some properties of the mentioned functions. We will also introduce an ordering relation, which will prove useful in the "cutting" of potential cycles mentioned at the end of Subsection 3.1. As second step, we will then use the introduced notions and properties to show that the underlying idea, presented in the informal discussion before and now formalized exactly, can in fact be implemented as an mtt and that Construction 3.6 is correct in that it constructs an mtt $M_{1;2}$, which induces the composition of the two tree transductions induced by the given mtts M_1 and M_2.

In the following, let M_1 and M_2 be two fixed mtts as in Construction 3.6. We will also use the notations and names introduced there. For technical reasons, we assume without loss of generality that there are no name conflicts between M_1 and M_2, i.e., all involved ranked alphabets are pairwise disjoint. This can always be achieved by renaming and guarantees, e.g., that rewrite rules of M_1 and M_2 can be applied in arbitrary order ($\Rightarrow_{R_1 \cup R_2}$ is confluent).

As noted below Construction 3.6, we could generalize the weakly single-use property in Definition 2.7 by dropping condition (ii), and requiring condition (i) only for states p that do have context parameters. In fact, the proofs in this appendix will from Definition 2.7 only use condition (i) and this only for states of M_2 with rank greater than one. Also, the non-copying restriction of M_1 and the weakly single-use restriction of M_2 will not be needed, if one of the two is a top-down tree transducer. Thus, the correctness proof for Construction 3.6 also incorporates proofs for the known results $TOP; MAC \subseteq MAC$ [Eng81] and $MAC; TOP \subseteq MAC$ [EV85].

Firstly, we introduce functions that can be used to answer questions of the kind posed in Figure 2.

Definition A.1 We define the following set of functions by mutual recursion:

$$\{\kappa_{h,q,l,p} : T_\Sigma \to SEN(\emptyset, Y_r, Z_P) \mid r, s \in \mathbf{N}_+, q \in Q^{(r+1)}, p \in P^{(s+1)}, h \in [r], l \in [s]\}$$
$$\cup \{\pi_{p,\phi,k,r} : \underbrace{T_\Sigma \times \cdots \times T_\Sigma}_{k-times} \times \{o \in occ(\phi) \mid lab(\phi, o) \notin X_k\} \times [s] \to SEN(\emptyset, Y_r, Z_P)$$
$$\mid r, s \in \mathbf{N}_+, (r+1) \in rank(Q), k \in \mathbf{N}, \phi \in RHS(Q, \Delta, k, Y_r), p \in P^{(s+1)}\}$$

For every $r, s \in \mathbf{N}_+$, $q \in Q^{(r+1)}$, $p \in P^{(s+1)}$, $h \in [r]$ and $l \in [s]$, the function $\kappa_{h,q,l,p}$ is defined as follows. For every $k \in \mathbf{N}$, $\sigma \in \Sigma^{(k)}$ and $t_1, \ldots, t_k \in T_\Sigma$, we have: if $rhs_{q,\sigma} \in RHS(Q, \Delta, k, Y_r)$ does not contain the context variable y_h, then $\kappa_{h,q,l,p}(\sigma(t_1, \ldots, t_k)) = nil$; otherwise $\kappa_{h,q,l,p}(\sigma(t_1, \ldots, t_k)) = \pi_{p,rhs_{q,\sigma},k,r}(t_1, \ldots, t_k, o, l)$ with $o \in occ(rhs_{q,\sigma})$ as the unique occurrence of y_h in $rhs_{q,\sigma}$ (M_1 is non-copying!).

For every $r, s \in \mathbf{N}_+$ with $(r+1) \in rank(Q)$, $k \in \mathbf{N}$, $\phi \in RHS(Q, \Delta, k, Y_r)$ and $p \in P^{(s+1)}$, the function $\pi_{p,\phi,k,r}$ is defined as follows. For every $t_1, \ldots, t_k \in T_\Sigma$ and $l \in [s]$, we have:

- $\pi_{p,\phi,k,r}(t_1,\ldots,t_k,\varepsilon,l) = z_{p,l}$
- For every $i \in \mathbf{N}_+$ and $oi \in occ(\phi)$ with $lab(\phi,oi) \notin X_k$:
  $\pi_{p,\phi,k,r}(t_1,\ldots,t_k,oi,l) =$
  - in case $lab(\phi,o) = \delta$ with $a \in \mathbf{N}$, $\delta \in \Delta^{(a)}$ and $i \in [a]$:
    If, with $s' \in \mathbf{N}$, $p' \in P^{(s'+1)}$ and $\psi_1,\ldots,\psi_s \in RHS(P,\Omega,a,Z_{s'})$, the only occurrence of a $p(x_i,\cdots)$-call in the δ-rules of the weakly single-use mtt M_2 looks as follows:
    $$p'(\delta(x_1,\ldots,x_a),z_1,\ldots,z_{s'}) \to \cdots p(x_i,\psi_1,\ldots,\psi_s)\cdots,$$
    then the right-hand side of the equation is:
    $$nf(\Rightarrow_{R_2},\psi_l[x_u \leftarrow nf(\Rightarrow_{R_1},(sub(\phi,ou))[x_v \leftarrow t_v, v\in[k]]), u\in[a],\\ z_w \leftarrow \pi_{p',\phi,k,r}(t_1,\ldots,t_k,o,w), w\in[s']]).$$
    If no such call exists in the δ-rules of M_2, then the right-hand side of the equation is nil.
  - in case $lab(\phi,o) = q$ with $a \in \mathbf{N}$, $q \in Q^{(a+1)}$, $2 \le i \le a+1$ and $lab(\phi,o1) = x_b \in X_k$:
    $$(nf(\Rightarrow_{R_2},(\kappa_{i-1,q,l,p}(t_b))[y_u \leftarrow nf(\Rightarrow_{R_1},(sub(\phi,o(u+1)))[x_v \leftarrow t_v, v\in[k]]),\\ u\in[a]]))\\ [z_{p',w} \leftarrow \pi_{p',\phi,k,r}(t_1,\ldots,t_k,o,w),\ s'\in\mathbf{N}, p'\in P^{(s'+1)}, w\in[s']]$$ □

Note that the sentential forms produced by $\kappa_{h,q,l,p}$- and $\pi_{p,\phi,k,r}$-functions, respectively, are always $\Rightarrow_{R_2}$-normal forms. Also, note the similarities between the definitions of $\kappa_{h,q,l,p}$-functions and right-hand sides for states (h_q,l_p), and between the definitions of $\pi_{p,\phi,k,r}$-functions and $ctx_{p,\phi,k,r}$-functions, respectively. These correspondences will be made precise in Lemma A.12.

Example A.2 We refer to the mtts M_1 and M_2 from Example 3.2 and compute the value $\kappa_{1,q_1,1,p_2}(A(B(E)))$ as follows:

$\kappa_{1,q_1,1,p_2}(A(B(E)))$
= (by definition of $\kappa_{1,q_1,1,p_2}$, with $rhs_{q_1,A} = \alpha(q_1(x_1,\underline{y_1}))$)
$\pi_{p_2,\alpha(q_1(x_1,y_1)),1,1}(B(E),12,1)$
= (by definition of $\pi_{p_2,\alpha(q_1(x_1,y_1)),1,1}$)
$(nf(\Rightarrow_{R_2},(\kappa_{1,q_1,1,p_2}(B(E)))[y_1 \leftarrow nf(\Rightarrow_{R_1},y_1[x_1 \leftarrow B(E)])]))$
$[z_{p_1,1},z_{p_2,1} \leftarrow \pi_{p_1,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1),\pi_{p_2,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1)]$
= (by definition of $\kappa_{1,q_1,1,p_2}$, with $rhs_{q_1,B} = q_1(x_1,\beta(\underline{y_1}))$)
$(nf(\Rightarrow_{R_2},(\pi_{p_2,q_1(x_1,\beta(y_1)),1,1}(E,21,1))[y_1 \leftarrow y_1]))$
$[z_{p_1,1},z_{p_2,1} \leftarrow \pi_{p_1,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1),\pi_{p_2,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1)]$
= (by definition of $\pi_{p_2,q_1(x_1,\beta(y_1)),1,1}$, with $p_2(\beta(x_1),z_1)\to p_2(x_1,\omega(p_1(x_1,z_1),z_1))\in R_2$)
$(nf(\Rightarrow_{R_2},nf(\Rightarrow_{R_2},\omega(p_1(x_1,z_1),z_1)[x_1 \leftarrow nf(\Rightarrow_{R_1},y_1[x_1\leftarrow E]),$
$z_1 \leftarrow \pi_{p_2,q_1(x_1,\beta(y_1)),1,1}(E,2,1)])))$
$[z_{p_1,1},z_{p_2,1} \leftarrow \pi_{p_1,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1),\pi_{p_2,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1)]$
= (by substitution and normal form)
$(nf(\Rightarrow_{R_2},nf(\Rightarrow_{R_2},\omega(p_1(y_1,z_1),z_1)[z_1 \leftarrow \pi_{p_2,q_1(x_1,\beta(y_1)),1,1}(E,2,1)])))$
$[z_{p_1,1},z_{p_2,1} \leftarrow \pi_{p_1,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1),\pi_{p_2,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1)]$

$=$ (by definition of $\pi_{p_2,q_1(x_1,\beta(y_1)),1,1}$)

$(nf(\Rightarrow_{R_2}, nf(\Rightarrow_{R_2}, \omega(p_1(y_1,z_1),z_1)$
$[z_1 \leftarrow (nf(\Rightarrow_{R_2}, (\kappa_{1,q_1,1,p_2}(E))[y_1 \leftarrow nf(\Rightarrow_{R_1}, \beta(y_1)[x_1 \leftarrow E])]))$
$[z_{p_1,1}, z_{p_2,1} \leftarrow \pi_{p_1,q_1(x_1,\beta(y_1)),1,1}(E,\varepsilon,1), \pi_{p_2,q_1(x_1,\beta(y_1)),1,1}(E,\varepsilon,1)]]))))$
$[z_{p_1,1}, z_{p_2,1} \leftarrow \pi_{p_1,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1), \pi_{p_2,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1)]$

$=$ (by definition of $\pi_{p_1,q_1(x_1,\beta(y_1)),1,1}$ and $\pi_{p_2,q_1(x_1,\beta(y_1)),1,1}$)

$(nf(\Rightarrow_{R_2}, nf(\Rightarrow_{R_2}, \omega(p_1(y_1,z_1),z_1)$
$[z_1 \leftarrow (nf(\Rightarrow_{R_2}, (\kappa_{1,q_1,1,p_2}(E))[y_1 \leftarrow \beta(y_1)]))$
$[z_{p_1,1}, z_{p_2,1} \leftarrow z_{p_1,1}, z_{p_2,1}]])))$
$[z_{p_1,1}, z_{p_2,1} \leftarrow \pi_{p_1,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1), \pi_{p_2,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1)]$

$=$ (by definition of $\kappa_{1,q_1,1,p_2}$, with $rhs_{q_1,E} = y_1$)

$(nf(\Rightarrow_{R_2}, nf(\Rightarrow_{R_2}, \omega(p_1(y_1,z_1),z_1)[z_1 \leftarrow (nf(\Rightarrow_{R_2}, (\pi_{p_2,y_1,0,1}(\varepsilon,1))[y_1 \leftarrow \beta(y_1)]))]))$
$[z_{p_1,1}, z_{p_2,1} \leftarrow \pi_{p_1,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1), \pi_{p_2,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1)]$

$=$ (by definition of $\pi_{p_2,y_1,0,1}$)

$(nf(\Rightarrow_{R_2}, nf(\Rightarrow_{R_2}, \omega(p_1(y_1,z_1),z_1)[z_1 \leftarrow (nf(\Rightarrow_{R_2}, z_{p_2,1}[y_1 \leftarrow \beta(y_1)]))]))$
$[z_{p_1,1}, z_{p_2,1} \leftarrow \pi_{p_1,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1), \pi_{p_2,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1)]$

$=$ (by substitution and normal form)

$(\omega(p_1(y_1, z_{p_2,1}), z_{p_2,1}))$
$[z_{p_1,1}, z_{p_2,1} \leftarrow \pi_{p_1,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1), \pi_{p_2,\alpha(q_1(x_1,y_1)),1,1}(B(E),1,1)]$

$=$ (by definition of $\pi_{p_2,\alpha(q_1(x_1,y_1)),1,1}$, with $p_1(\alpha(x_1),z_1) \to p_1(x_1,\underline{p_2(x_1,z_1)}) \in R_2$)

$(\omega(p_1(y_1, z_{p_2,1}), z_{p_2,1}))$
$[z_{p_2,1} \leftarrow nf(\Rightarrow_{R_2}, z_1[x_1 \leftarrow nf(\Rightarrow_{R_1}, q_1(x_1,y_1)[x_1 \leftarrow B(E)]),$
$z_1 \leftarrow \pi_{p_1,\alpha(q_1(x_1,y_1)),1,1}(B(E),\varepsilon,1)])]$

$=$ (by definition of $\pi_{p_1,\alpha(q_1(x_1,y_1)),1,1}$)

$(\omega(p_1(y_1, z_{p_2,1}), z_{p_2,1}))$
$[z_{p_2,1} \leftarrow nf(\Rightarrow_{R_2}, z_1[x_1 \leftarrow nf(\Rightarrow_{R_1}, q_1(x_1,y_1)[x_1 \leftarrow B(E)]),$
$z_1 \leftarrow z_{p_1,1}])]$

$=$ (by substitution and normal form)

$\omega(p_1(y_1, z_{p_1,1}), z_{p_1,1})$

In Example A.5 we will establish how this result corresponds to the twofold underlined subterm in Example 3.2. □

The following theorem establishes the key property of the functions introduced in Definition A.1.

Theorem A.3 For every $t \in T_\Sigma$, $r, s \in \mathbf{N}_+$, $s' \in \mathbf{N}$, $q \in Q^{(r+1)}$, $p \in P^{(s+1)}$, $p' \in P^{(s'+1)}$, $h \in [r]$ and $l \in [s]$: if during a reduction of $p'(q(t, y_1, \ldots, y_r), z_1, \ldots, z_{s'})$ with $\Rightarrow_{R_1 \cup R_2}$ a call of the form $p(y_h, \xi_1, \ldots, \xi_s)$ occurs, then we have $\kappa_{h,q,l,p}(t) \in SEN(\emptyset, Y_r, \{z_{p',1}, \ldots, z_{p',s'}\})$ and $nf(\Rightarrow_{R_1 \cup R_2}, \xi_l) = (\kappa_{h,q,l,p}(t))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$. □

For proving Theorem A.3, we will use the following principle of proof by simultaneous induction [EV85, KV94, FV98].

Definition A.4 Let Σ be a ranked alphabet. Given two statements **(I)** and **(II)**, where **(I)** has a free variable $\mathbf{t} \in \mathbf{T_\Sigma}$ and **(II)** has free variables $\mathbf{k} \in \mathbf{N}$ and $\mathbf{t_1}, \ldots, \mathbf{t_k} \in \mathbf{T_\Sigma}$, we say that **(I)** and **(II)** are *proven by simultaneous induction*, if we can prove the following two induction steps:

(I) $\Leftarrow$ (II) : for every $k \in \mathbf{N}$, $\sigma \in \Sigma^{(k)}$ and $t_1, \ldots, t_k \in T_\Sigma$, such that **(II)** holds, we can show that **(I)** holds for $t = \sigma(t_1, \ldots, t_k)$

(II) $\Leftarrow$ (I) : for every $k \in \mathbf{N}$ and $t_1, \ldots, t_k \in T_\Sigma$, such that **(I)** holds for each of the $t_1, \ldots, t_k$, we can show that **(II)** holds

Then, we have proven that **(I)** holds for every $t \in T_\Sigma$ and that **(II)** holds for every $k \in \mathbf{N}$ and $t_1, \ldots, t_k \in T_\Sigma$. □

Proof of Theorem A.3: In order to describe the context parameters, with which states of M_2 reach certain occurrences in a tree, we need the notion of *marking* a tree. Therefore, we define a function *mark* on arbitrary trees, which adds to every symbol as a superscript its occurrence in the tree. For example, we have $mark(\delta(q(x,y),\alpha)) = \delta^{\varepsilon}(q^1(x^{11}, y^{12}), \alpha^2)$. All operations on marked trees work in principle like on ordinary trees. Markings are ignored, e.g., in matching a tree against the left-hand side of a rewrite rule, but markings are preserved by, e.g., rewriting, substitutions or the *sub*-function. If there is, e.g., a rule $p(\delta(x_1, x_2)) \to \gamma(p(x_2))$, then this induces the rewriting step $p(\delta^{\varepsilon}(q^1(x^{11}, y^{12}), \alpha^2)) \Rightarrow \gamma(p(\alpha^2))$. We will use the placeholder $\bullet^o$ for an arbitrary tree, which has a root symbol marked with occurrence o. Now, we prove the following two statements by simultaneous induction.

(I) For $\mathbf{t \in T_\Sigma}$:
For every $r, s \in \mathbf{N}_+$, $s' \in \mathbf{N}$, $q \in Q^{(r+1)}$, $p \in P^{(s+1)}$, $p' \in P^{(s'+1)}$, $h \in [r]$ and $l \in [s]$:
If during a reduction of $p'(q(t, y_1, \ldots, y_r), z_1, \ldots, z_{s'})$ with $\Rightarrow_{R_1 \cup R_2}$ a call of the form $p(y_h, \xi_1, \ldots, \xi_s)$ occurs, then we have $\kappa_{h,q,l,p}(t) \in SEN(\emptyset, Y_r, \{z_{p',1}, \ldots, z_{p',s'}\})$ and:
$nf(\Rightarrow_{R_1 \cup R_2}, \xi_l) = (\kappa_{h,q,l,p}(t))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$.

(II) For $\mathbf{k \in \mathbf{N}, t_1, \ldots, t_k \in T_\Sigma}$:
For every $r \in \mathbf{N}_+$ with $(r+1) \in rank(Q)$, $\phi \in RHS(Q, \Delta, k, Y_r)$, $o \in occ(\phi)$, such that $lab(\phi, o) \notin X_k$, $s \in \mathbf{N}_+$, $s' \in \mathbf{N}$, $p \in P^{(s+1)}$, $p' \in P^{(s'+1)}$ and $l \in [s]$:
If during a reduction of $p'((mark(\phi))[x_v \leftarrow t_v, v \in [k]], z_1, \ldots, z_{s'})$ with $\Rightarrow_{R_1 \cup R_2}$ a call of the form $p(\bullet^o, \xi_1, \ldots, \xi_s)$ occurs, then we have $\pi_{p,\phi,k,r}(t_1, \ldots, t_k, o, l) \in SEN(\emptyset, Y_r, \{z_{p',1}, \ldots, z_{p',s'}\})$ and:
$nf(\Rightarrow_{R_1 \cup R_2}, \xi_l) = (\pi_{p,\phi,k,r}(t_1, \ldots, t_k, o, l))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$.

(I) $\Leftarrow$ (II) : Let $t = \sigma(t_1, \ldots, t_k)$ with $k \in \mathbf{N}$, $\sigma \in \Sigma^{(k)}$ and $t_1, \ldots, t_k \in T_\Sigma$, then:

If during a reduction of $p'(q(\sigma(t_1, \ldots, t_k), y_1, \ldots, y_r), z_1, \ldots, z_{s'})$ with $\Rightarrow_{R_1 \cup R_2}$ a call of the form $p(y_h, \xi_1, \ldots, \xi_s)$ occurs, then it must be the case that this call occurs during a reduction of $p'(rhs_{q,\sigma}[x_v \leftarrow t_v, v \in [k]], z_1, \ldots, z_{s'})$ with $rhs_{q,\sigma} \in RHS(Q, \Delta, k, Y_r)$. This is only possible, if there is a—then unique, because M_1 is non-copying—occurrence o in $rhs_{q,\sigma}$ with $lab(rhs_{q,\sigma}, o) = y_h$. Consequently, the $\xi_1, \ldots, \xi_s$ appear in an occurrence $p(y_h{}^o, \xi_1, \ldots, \xi_s)$, resulting from a reduction of $p'((mark(rhs_{q,\sigma}))[x_v \leftarrow t_v, v \in [k]], z_1, \ldots, z_{s'})$. By induction hypothesis **(II)**, this means that $\pi_{p,rhs_{q,\sigma},k,r}(t_1, \ldots, t_k, o, l) \in SEN(\emptyset, Y_r, \{z_{p',1}, \ldots, z_{p',s'}\})$ and:

$$nf(\Rightarrow_{R_1 \cup R_2}, \xi_l) = (\pi_{p,rhs_{q,\sigma},k,r}(t_1, \ldots, t_k, o, l))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$$

By the definition of $\kappa_{h,q,l,p}$ for $t = \sigma(t_1, \ldots, t_k)$, this amounts to $\kappa_{h,q,l,p}(\sigma(t_1, \ldots, t_k)) \in SEN(\emptyset, Y_r, \{z_{p',1}, \ldots, z_{p',s'}\})$ and:

$$nf(\Rightarrow_{R_1 \cup R_2}, \xi_l) = (\kappa_{h,q,l,p}(\sigma(t_1, \ldots, t_k)))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$$

(II) ⇐ (I) : For fixed r and ϕ, we prove the claim by induction on the prefix-order of occurrences in ϕ:

- In the base case ε, we have a call of the form $p(\bullet^{\varepsilon}, \xi_1, \ldots, \xi_s)$ during reduction of $p'((mark(\phi))[x_v \leftarrow t_v, v \in [k]], z_1, \ldots, z_{s'})$. This is only possible, if $p = p'$ and $\xi_1, \ldots, \xi_s = z_1, \ldots, z_{s'}$, in which case we have (by Definition A.1):
  $\pi_{p,\phi,k,r}(t_1, \ldots, t_k, \varepsilon, l) = z_{p,l} \in SEN(\emptyset, Y_r, \{z_{p',1}, \ldots, z_{p',s'}\})$ and:

  $nf(\Rightarrow_{R_1 \cup R_2}, \xi_l)$
  $\quad = $ (by $\xi_1, \ldots, \xi_s = z_1, \ldots, z_{s'}$)
  z_l
  $\quad = $ (by $p = p'$ and substitution)
  $z_{p,l}[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$
  $\quad = $ (by definition of $\pi_{p,\phi,k,r}$)
  $(\pi_{p,\phi,k,r}(t_1, \ldots, t_k, \varepsilon, l))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$

- In the inductive case $oi \in occ(\phi)$—with $o \in occ(\phi)$, $i \in \mathbf{N}_+$ and $lab(\phi, oi) \notin X_k$—the reasoning is as follows. If during a reduction of $p'((mark(\phi))[x_v \leftarrow t_v, v \in [k]], z_1, \ldots, z_{s'})$ with $\Rightarrow_{R_1 \cup R_2}$ a call of the form $p(\bullet^{oi}, \xi_1, \ldots, \xi_s)$ occurs, then there exist $s'' \in \mathbf{N}$, $p'' \in P^{(s''+1)}$ and $\rho_1, \ldots, \rho_{s''}$, such that reducing $p'((mark(\phi))[x_v \leftarrow t_v, v \in [k]], z_1, \ldots, z_{s'})$ leads to an occurrence of $p''((sub(mark(\phi), o))[x_v \leftarrow t_v, v \in [k]], \rho_1, \ldots, \rho_{s''})$ and from this call to p'' on $(sub(mark(\phi), o))[x_v \leftarrow t_v, v \in [k]]$ results—by reduction—a call of the form $p(\bullet^{oi}, \xi_1, \ldots, \xi_s)$. This situation is shown in Figure 3.

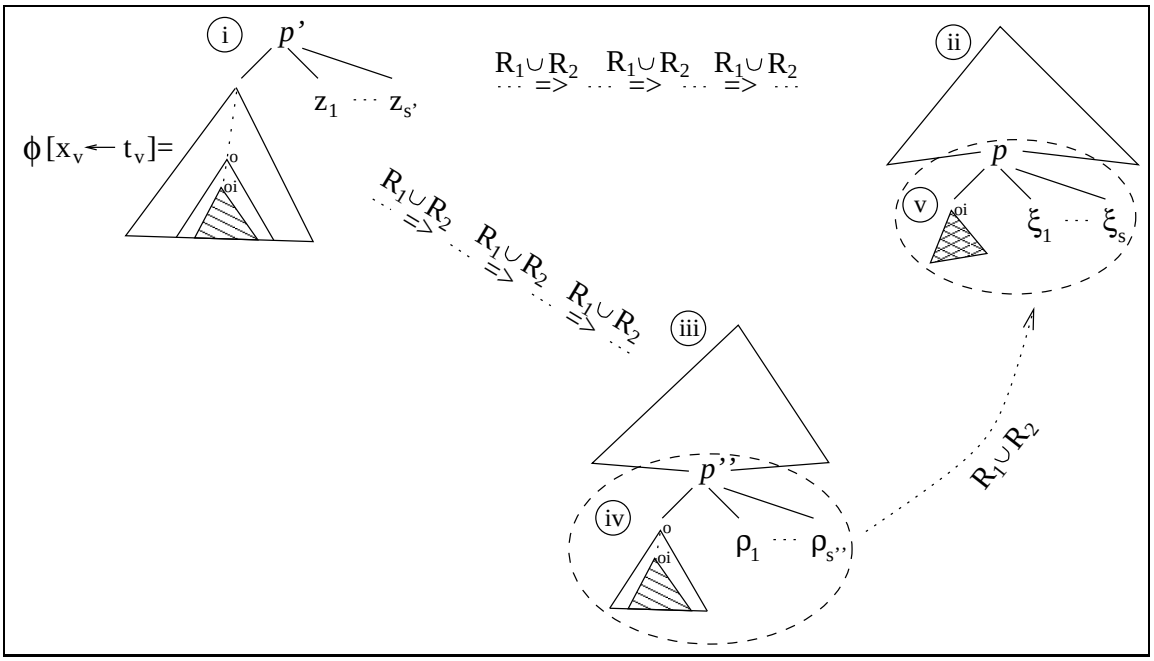

Figure 3: If **i** reduces to **ii**, then exists **iii**, such that a reduction of **iv** contains a **v**.

By applying the induction hypothesis for o, we know that for every $w \in [s'']$, we have $\pi_{p'',\phi,k,r}(t_1, \ldots, t_k, o, w) \in SEN(\emptyset, Y_r, \{z_{p',1}, \ldots, z_{p',s'}\})$ and:
$nf(\Rightarrow_{R_1 \cup R_2}, \rho_w) = (\pi_{p'',\phi,k,r}(t_1, \ldots, t_k, o, w))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$

We proceed by case distinction on $lab(\phi, o)$:

– $lab(\phi, o) = \delta$ with $a \in \mathbf{N}$, $\delta \in \Delta^{(a)}$ and $i \in [a]$:
We know that from reduction of the call
$p''(\delta^o(sub(mark(\phi), o1), \ldots, sub(mark(\phi), oa))[x_v \leftarrow t_v, v \in [k]], \rho_1, \ldots, \rho_{s''})$
results a call of the form $p(\bullet^{oi}, \xi_1, \ldots, \xi_s)$. Consequently, there must be a rule

$$p''(\delta(x_1, \ldots, x_a), z_1, \ldots, z_{s''}) \to \cdots p(x_i, \psi_1, \ldots, \psi_s) \cdots$$

in R_2 (with $\psi_1, \ldots, \psi_s \in RHS(P, \Omega, a, Z_{s''})$) and:

$$\begin{aligned}
&nf(\Rightarrow_{R_1 \cup R_2}, \xi_l) = nf(\Rightarrow_{R_1 \cup R_2}, \psi_l[x_u \leftarrow (sub(\phi, ou))[x_v \leftarrow t_v, v \in [k]], u \in [a],\\
&\qquad\qquad z_w \leftarrow \rho_w, w \in [s'']])\\
&\quad = \text{ (by confluence considerations)}\\
&nf(\Rightarrow_{R_2}, \psi_l[x_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, ou))[x_v \leftarrow t_v, v \in [k]]), u \in [a],\\
&\qquad z_w \leftarrow nf(\Rightarrow_{R_1 \cup R_2}, \rho_w), w \in [s'']])\\
&\quad = \text{ (by the information about the normal forms of } \rho_1, \ldots, \rho_{s''} \text{ gained above)}\\
&nf(\Rightarrow_{R_2}, \psi_l[x_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, ou))[x_v \leftarrow t_v, v \in [k]]), u \in [a],\\
&\qquad z_w \leftarrow (\pi_{p'', \phi, k, r}(t_1, \ldots, t_k, o, w))\\
&\qquad\qquad [z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}], w \in [s'']])\\
&\quad = \text{ (since the only occurrences of } z_{p',1}, \ldots, z_{p',s'} \text{ are in the expressions}\\
&\qquad \text{substituted for the } z_w)\\
&(nf(\Rightarrow_{R_2}, \psi_l[x_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, ou))[x_v \leftarrow t_v, v \in [k]]), u \in [a],\\
&\qquad z_w \leftarrow \pi_{p'', \phi, k, r}(t_1, \ldots, t_k, o, w), w \in [s'']]))\\
&\quad [z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]
\end{aligned}$$

Taking into account that the occurrence of $p(x_i, \psi_1, \ldots, \psi_s)$ in the rule from R_2 given above must be the unique occurrence of a $p(x_i, \cdots)$-call in the δ-rules of the weakly single-use mtt M_2, we have by definition of $\pi_{p,\phi,k,r}$:
$\pi_{p,\phi,k,r}(t_1, \ldots, t_k, oi, l) =$
$nf(\Rightarrow_{R_2}, \psi_l[x_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, ou))[x_v \leftarrow t_v, v \in [k]]), u \in [a],$
$\qquad z_w \leftarrow \pi_{p'', \phi, k, r}(t_1, \ldots, t_k, o, w), w \in [s'']])$
Since for every $w \in [s'']$, we know by the induction hypothesis for o that $\pi_{p'', \phi, k, r}(t_1, \ldots, t_k, o, w) \in SEN(\emptyset, Y_r, \{z_{p',1}, \ldots, z_{p',s'}\})$, and since for every $u \in [a]$, we have $nf(\Rightarrow_{R_1}, (sub(\phi, ou))[x_v \leftarrow t_v, v \in [k]]) \in T_\Delta(Y_r)$ (because $\phi \in RHS(Q, \Delta, k, Y_r)$), this implies the following:
$\pi_{p,\phi,k,r}(t_1, \ldots, t_k, oi, l) \in SEN(\emptyset, Y_r, \{z_{p',1}, \ldots, z_{p',s'}\})$.
Together with the previously computed equivalent of $nf(\Rightarrow_{R_1 \cup R_2}, \xi_l)$, we also have:

$$\begin{aligned}
&(\pi_{p,\phi,k,r}(t_1, \ldots, t_k, oi, l))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]\\
&\quad =\\
&(nf(\Rightarrow_{R_2}, \psi_l[x_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, ou))[x_v \leftarrow t_v, v \in [k]]), u \in [a],\\
&\qquad z_w \leftarrow \pi_{p'', \phi, k, r}(t_1, \ldots, t_k, o, w), w \in [s'']]))\\
&\quad [z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]\\
&\quad =\\
&nf(\Rightarrow_{R_1 \cup R_2}, \xi_l)
\end{aligned}$$

This proves the claim.

– $lab(\phi, o) = q$ with $a \in \mathbf{N}$, $q \in Q^{(a+1)}$, $2 \leq i \leq a+1$ and $lab(\phi, o1) = x_b \in X_k$:
We know that from reduction of the call
$p''(q^o(x_b{}^{o1}, sub(mark(\phi), o2), \ldots, sub(mark(\phi), o(a+1)))[x_v \leftarrow t_v, v \in [k]],$
$\rho_1, \ldots, \rho_{s''})$
results a call of the form $p(\bullet^{oi}, \xi_1, \ldots, \xi_s)$. This means that a call to p reaches the $(i-1)$st context parameter position of the call of q^o on t_b. By the induction hypothesis **(I)** for t_b, we know that for a call of the form $p(y_{i-1}, \xi'_1, \ldots, \xi'_s)$, resulting from a reduction of $p''(q(t_b, y_1, \ldots, y_a), z_1, \ldots, z_{s''})$, we have $\kappa_{i-1,q,l,p}(t_b) \in SEN(\emptyset, Y_a, \{z_{p'',1}, \ldots, z_{p'',s''}\})$ and:
$nf(\Rightarrow_{R_1 \cup R_2}, \xi'_l) = (\kappa_{i-1,q,l,p}(t_b))[z_{p'',1}, \ldots, z_{p'',s''} \leftarrow z_1, \ldots, z_{s''}]$
For our call $p(\bullet^{oi}, \xi_1, \ldots, \xi_s)$, resulting from reducing
$p''(q^o(t_b, (sub(mark(\phi), o2))[x_v \leftarrow t_v, v \in [k]],$
$\ldots,$
$(sub(mark(\phi), o(a+1)))[x_v \leftarrow t_v, v \in [k]]),$
$\rho_1, \ldots, \rho_{s''})$,
this implies the following calculation:

$$\begin{aligned}
nf(\Rightarrow_{R_1 \cup R_2}, \xi_l) = nf(\Rightarrow_{R_1 \cup R_2}, \xi'_l[y_u \leftarrow (sub(\phi, o(u+1)))[x_v \leftarrow t_v, v \in [k]], u \in [a],\\
z_w \leftarrow \rho_w, w \in [s'']])\\
= \text{ (by confluence considerations)}\\
nf(\Rightarrow_{R_1 \cup R_2}, (nf(\Rightarrow_{R_1 \cup R_2}, \xi'_l))[y_u \leftarrow (sub(\phi, o(u+1)))[x_v \leftarrow t_v, v \in [k]], u \in [a],\\
z_w \leftarrow \rho_w, w \in [s'']])\\
= \text{ (by } nf(\Rightarrow_{R_1 \cup R_2}, \xi'_l) = (\kappa_{i-1,q,l,p}(t_b))[z_{p'',1}, \ldots, z_{p'',s''} \leftarrow z_1, \ldots, z_{s''}])\\
nf(\Rightarrow_{R_1 \cup R_2}, (\kappa_{i-1,q,l,p}(t_b))[z_{p'',1}, \ldots, z_{p'',s''} \leftarrow z_1, \ldots, z_{s''}]\\
[y_u \leftarrow (sub(\phi, o(u+1)))[x_v \leftarrow t_v, v \in [k]], u \in [a],\\
z_w \leftarrow \rho_w, w \in [s'']])\\
= \text{ (by confluence considerations)}\\
nf(\Rightarrow_{R_2}, (\kappa_{i-1,q,l,p}(t_b))[z_{p'',1}, \ldots, z_{p'',s''} \leftarrow z_1, \ldots, z_{s''}]\\
[y_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, o(u+1)))[x_v \leftarrow t_v, v \in [k]]), u \in [a],\\
z_w \leftarrow nf(\Rightarrow_{R_1 \cup R_2}, \rho_w), w \in [s'']])\\
= \text{ (by the information about the normal forms of } \rho_1, \ldots, \rho_{s''} \text{ gained above)}\\
nf(\Rightarrow_{R_2}, (\kappa_{i-1,q,l,p}(t_b))[z_{p'',1}, \ldots, z_{p'',s''} \leftarrow z_1, \ldots, z_{s''}]\\
[y_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, o(u+1)))[x_v \leftarrow t_v, v \in [k]]), u \in [a],\\
z_w \leftarrow (\pi_{p'',\phi,k,r}(t_1, \ldots, t_k, o, w))\\
[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}], w \in [s'']])\\
= \text{ (by composing substitutions)}\\
nf(\Rightarrow_{R_2}, (\kappa_{i-1,q,l,p}(t_b))[y_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, o(u+1)))[x_v \leftarrow t_v, v \in [k]]), u \in [a],\\
z_{p'',w} \leftarrow (\pi_{p'',\phi,k,r}(t_1, \ldots, t_k, o, w))\\
[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}],\\
w \in [s'']])
\end{aligned}$$

We obtained above that $\kappa_{i-1,q,l,p}(t_b) \in SEN(\emptyset, Y_a, \{z_{p'',1}, \ldots, z_{p'',s''}\})$. Thus, $\kappa_{i-1,q,l,p}(t_b)$ does not contain variables from $Z_P \backslash \{z_{p'',1}, \ldots, z_{p'',s''}\}$, and we can transform the previous expression into the following equivalent[6]:

[6] It does not matter, which values we replace for the variables from $Z_P \backslash \{z_{p'',1}, \ldots, z_{p'',s''}\}$, because they anyway do not occur in $\kappa_{i-1,q,l,p}(t_b)$. We only need to ensure that for the $z_{p'',1}, \ldots, z_{p'',s''}$ the same values are substituted as in the previous expression.

$$nf(\Rightarrow_{R_2}, (\kappa_{i-1,q,l,p}(t_b))[y_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, o(u+1)))[x_v \leftarrow t_v, v \in [k]]), u \in [a],$$
$$z_{p''',w} \leftarrow (\pi_{p''',\phi,k,r}(t_1, \ldots, t_k, o, w))$$
$$[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}],$$
$$s''' \in \mathbf{N}, p''' \in P^{(s'''+1)}, w \in [s''']])$$

$=$ (by substitution and confluence considerations)

$$(nf(\Rightarrow_{R_2}, (\kappa_{i-1,q,l,p}(t_b))$$
$$[y_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, o(u+1)))[x_v \leftarrow t_v, v \in [k]]), u \in [a]]))$$
$$[z_{p''',w} \leftarrow nf(\Rightarrow_{R_2}, (\pi_{p''',\phi,k,r}(t_1, \ldots, t_k, o, w))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]),$$
$$s''' \in \mathbf{N}, p''' \in P^{(s'''+1)}, w \in [s''']]$$

$=$ (since the $\pi_{p''',\phi,k,r}(t_1, \ldots, t_k, o, w)$ are $\Rightarrow_{R_2}$-normal forms, by the remark below Definition A.1)

$$(nf(\Rightarrow_{R_2}, (\kappa_{i-1,q,l,p}(t_b))$$
$$[y_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, o(u+1)))[x_v \leftarrow t_v, v \in [k]]), u \in [a]]))$$
$$[z_{p''',w} \leftarrow (\pi_{p''',\phi,k,r}(t_1, \ldots, t_k, o, w))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}],$$
$$s''' \in \mathbf{N}, p''' \in P^{(s'''+1)}, w \in [s''']]$$

$=$ (by composition of substitutions)

$$(nf(\Rightarrow_{R_2}, (\kappa_{i-1,q,l,p}(t_b))$$
$$[y_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, o(u+1)))[x_v \leftarrow t_v, v \in [k]]), u \in [a]]))$$
$$[z_{p''',w} \leftarrow \pi_{p''',\phi,k,r}(t_1, \ldots, t_k, o, w),\ s''' \in \mathbf{N}, p''' \in P^{(s'''+1)}, w \in [s''']]$$
$$[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$$

By definition of $\pi_{p,\phi,k,r}$, we have:

$$\pi_{p,\phi,k,r}(t_1, \ldots, t_k, oi, l) =$$
$$(nf(\Rightarrow_{R_2}, (\kappa_{i-1,q,l,p}(t_b))$$
$$[y_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, o(u+1)))[x_v \leftarrow t_v, v \in [k]]), u \in [a]]))$$
$$[z_{p''',w} \leftarrow \pi_{p''',\phi,k,r}(t_1, \ldots, t_k, o, w),\ s''' \in \mathbf{N}, p''' \in P^{(s'''+1)}, w \in [s''']]$$

Since we reasoned above that $\kappa_{i-1,q,l,p}(t_b) \in SEN(\emptyset, Y_a, \{z_{p'',1}, \ldots, z_{p'',s''}\})$, and since for every $w \in [s'']$, we know by the induction hypothesis for o that $\pi_{p'',\phi,k,r}(t_1, \ldots, t_k, o, w) \in SEN(\emptyset, Y_r, \{z_{p',1}, \ldots, z_{p',s'}\})$, and since for every $u \in [a]$, we have $nf(\Rightarrow_{R_1}, (sub(\phi, o(u+1)))[x_v \leftarrow t_v, v \in [k]]) \in T_\Delta(Y_r)$ (by $\phi \in RHS(Q, \Delta, k, Y_r)$), this implies the following:

$$\pi_{p,\phi,k,r}(t_1, \ldots, t_k, oi, l) \in SEN(\emptyset, Y_r, \{z_{p',1}, \ldots, z_{p',s'}\}).$$

Together with the previously computed equivalent of $nf(\Rightarrow_{R_1 \cup R_2}, \xi_l)$, we also have:

$$(\pi_{p,\phi,k,r}(t_1, \ldots, t_k, oi, l))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$$
$=$
$$(nf(\Rightarrow_{R_2}, (\kappa_{i-1,q,l,p}(t_b))$$
$$[y_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, o(u+1)))[x_v \leftarrow t_v, v \in [k]]), u \in [a]]))$$
$$[z_{p''',w} \leftarrow \pi_{p''',\phi,k,r}(t_1, \ldots, t_k, o, w),\ s''' \in \mathbf{N}, p''' \in P^{(s'''+1)}, w \in [s''']]$$
$$[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$$
$=$
$$nf(\Rightarrow_{R_1 \cup R_2}, \xi_l)$$

This proves the claim. □

Above Figure 2 we posed the following question:

> "Given two states $q' \in Q^{(a+1)}$ and $p' \in P^{(s'+1)}$ and some input tree t for M_1, can we determine, for any state p'' of M_2, and any context parameter y_h from $y_1, \ldots, y_a$, which will be the values in context parameter positions in every occurrence of a call to p'' on y_h during the reduction of $p'(q'(t, y_1, \ldots, y_a), \eta_1, \ldots, \eta_{s'})$?"

Because of Theorem A.3, we can now answer this question positively by stating that the value in the lth context parameter position of every such call to p'' on y_h will be equal—modulo further reductions—to $(\kappa_{h,q',l,p''}(t))[z_{p',1}, \ldots, z_{p',s'} \leftarrow \eta_1, \ldots, \eta_{s'}]$.

Example A.5 For the mtts M_1 and M_2 from Example 3.2, we computed in Example A.2 the following:

$\kappa_{1,q_1,1,p_2}(A(B(E))) = \omega(p_1(y_1, z_{p_1,1}), z_{p_1,1}) \in SEN(\emptyset, Y_1, \{z_{p_1,1}\})$

Theorem A.3 now says that if during a reduction of $p_1(q_1(A(B(E)), y_1), z_1)$ with $\Rightarrow_{R_1 \cup R_2}$ a call of the form $p_2(y_1, \xi_1)$ occurs, then we have:

$nf(\Rightarrow_{R_1 \cup R_2}, \xi_1) = \omega(p_1(y_1, z_{p_1,1}), z_{p_1,1})[z_{p_1,1} \leftarrow z_1]$

In fact, we have seen in Example 3.2:

$p_1(q_1(A(B(E)), y_1), z_1) \Rightarrow^*_{R_1 \cup R_2} p_1(\beta(y_1), \underline{p_2(y_1, \underline{\omega(p_1(y_1, z_1), z_1)})})$ □

Now, we define for every state and every input tree of M_1, a binary relation on pairs, consisting of a context parameter position of the given state from M_1, and some state of the second mtt. This relation is intended to represent the nesting of occurrences of such pairs that can result from processing the intermediate result generated by M_1 (with the given state on the given input tree), with states of M_2.

Definition A.6 For every $r \in \mathbf{N}$, $q \in Q^{(r+1)}$ and $t \in T_\Sigma$, we define a binary relation $\prec_{q,t}$ on $[r] \times P$ as follows. For every $h_1, h_2 \in [r]$ and $p_1, p_2 \in P$, we have $(h_1, p_1) \prec_{q,t} (h_2, p_2)$ iff there exists an $l \in [rank_P(p_1) - 1]$, such that $\kappa_{h_1,q,l,p_1}(t)$ contains an occurrence of a $p_2(y_{h_2}, \cdots)$-call. For some subset $\mathcal{C} \subseteq [r] \times P$, we write $(h_1, p_1) \prec^*_{q,t} \mathcal{C}$ iff there exists an $(h', p') \in \mathcal{C}$ with $(h_1, p_1) \prec^*_{q,t} (h', p')$. Analogous for $\prec^+_{q,t}$. □

At the end of Subsection 3.1, we claimed that for non-copying M_1 and weakly single-use M_2, we cannot reach a situation, where two occurrences of one and the same such pair are nested, which would correspond to the "counterexample" (b) in Example 3.3. Now, we can formalize this claim as follows.

Claim A.7 For every $r \in \mathbf{N}$, $q \in Q^{(r+1)}$ and $t \in T_\Sigma$, there are no $h \in [r]$ and $p \in P$ with $(h, p) \prec^+_{q,t} (h, p)$. □

We will not formally prove this claim here. Instead, we show the following slightly weaker lemma, which in fact covers all cases that are important for the correctness of the mtt $M_{1;2}$.

Lemma A.8 For every $r, s \in \mathbf{N}$, $q \in Q^{(r+1)}$, $t \in T_\Sigma$, $p \in P^{(s+1)}$ and $h \in [r]$, the following holds: if there exist $s' \in \mathbf{N}$ and $p' \in P^{(s'+1)}$, such that during a reduction of $p'(q(t, y_1, \ldots, y_r), z_1, \ldots, z_{s'})$ with $\Rightarrow_{R_1 \cup R_2}$ a call of the form $p(y_h, \cdots)$ occurs, then we do not have $(h, p) \prec^+_{q,t} (h, p)$. □

Proof: The proof is by contradiction. Let the call of the form $p(y_h, \cdots)$ resulting from reducing $p'(q(t, y_1, \ldots, y_r), z_1, \ldots, z_{s'})$ be $p(y_h, \xi_1, \ldots, \xi_s)$, and assume, there were $n \in \mathbf{N}$ and $(h_0, p_0), \ldots, (h_n, p_n) \in [r] \times P$ with $(h, p) \prec_{q,t} (h_0, p_0) \prec_{q,t} \ldots \prec_{q,t} (h_n, p_n) = (h, p)$. Firstly, we show by induction that then for every $0 \leq k \leq n$ holds: $(h, p) \prec_{q,t} (h_k, p_k)$.

- $k = 0$:
  $(h, p) \prec_{q,t} (h_0, p_0)$ holds by assumption.

- $k \in [n]$:
  By induction hypothesis, we know that $(h, p) \prec_{q,t} (h_{k-1}, p_{k-1})$ and by assumption also $(h_{k-1}, p_{k-1}) \prec_{q,t} (h_k, p_k)$. With $p_{k-1} \in P^{(s_1+1)}$ and $p_k \in P^{(s_2+1)}$ (for $s_1, s_2 \in \mathbf{N}$), this means—by Definition A.6—that there exist $l_1 \in [s]$ and $l_2 \in [s_1]$, such that $\kappa_{h,q,l_1,p}(t)$ contains an occurrence of a $p_{k-1}(y_{h_{k-1}}, \cdots)$-call and $\kappa_{h_{k-1},q,l_2,p_{k-1}}(t)$ contains an occurrence of a $p_k(y_{h_k}, \cdots)$-call.
  By Theorem A.3, we have $nf(\Rightarrow_{R_1 \cup R_2}, \xi_{l_1}) = (\kappa_{h,q,l_1,p}(t))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$ with $\kappa_{h,q,l_1,p}(t) \in SEN(\emptyset, Y_r, \{z_{p',1}, \ldots, z_{p',s'}\})$, hence this normal form must contain an occurrence of a $p_{k-1}(y_{h_{k-1}}, \rho_1, \ldots, \rho_{s_1})$-call with $\rho_1, \ldots, \rho_{s_1} \in SEN(\emptyset, Y_r, Z_{s'})$. But, this implies that there is a reduction of $p'(q(t, y_1, \ldots, y_r), z_1, \ldots, z_{s'})$ with $\Rightarrow_{R_1 \cup R_2}$ (namely by reducing the ξ_{l_1} in $p(y_h, \xi_1, \ldots, \xi_s)$) leading to an occurrence of a $p_{k-1}(y_{h_{k-1}}, \rho_1, \ldots, \rho_{s_1})$-call and thus again by Theorem A.3: $nf(\Rightarrow_{R_1 \cup R_2}, \rho_{l_2}) = (\kappa_{h_{k-1},q,l_2,p_{k-1}}(t))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$.
  Since—as a subtree of $nf(\Rightarrow_{R_1 \cup R_2}, \xi_{l_1})$—$\rho_{l_2}$ is a $\Rightarrow_{R_1 \cup R_2}$-normal form, we now know that it is equal to the expression $(\kappa_{h_{k-1},q,l_2,p_{k-1}}(t))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$ and thus contains an occurrence of a $p_k(y_{h_k}, \cdots)$-call. Since ρ_{l_2} for its part occurred as a subtree of $nf(\Rightarrow_{R_1 \cup R_2}, \xi_{l_1}) = (\kappa_{h,q,l_1,p}(t))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$, we have that there is an occurrence of a $p_k(y_{h_k}, \cdots)$-call in $\kappa_{h,q,l_1,p}(t)$, which indeed gives us $(h, p) \prec_{q,t} (h_k, p_k)$.

Consequently, we have $(h, p) \prec_{q,t} (h, p)$. By Definition A.6, this means that there must be an $l \in [s]$, such that $\kappa_{h,q,l,p}(t)$ contains an occurrence of a $p(y_h, \cdots)$-call. By Theorem A.3, we have $nf(\Rightarrow_{R_1 \cup R_2}, \xi_l) = (\kappa_{h,q,l,p}(t))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$. This normal form now must contain a $p(y_h, \xi'_1, \ldots, \xi'_s)$-call with $\xi'_1, \ldots, \xi'_s \in SEN(\emptyset, Y_r, Z_{s'})$. Since this call does occur in a $\Rightarrow_{R_1 \cup R_2}$-reduction of $p'(q(t, y_1, \ldots, y_r), z_1, \ldots, z_{s'})$, we have again by Theorem A.3: $nf(\Rightarrow_{R_1 \cup R_2}, \xi'_l) = (\kappa_{h,q,l,p}(t))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}]$. Since—as a subtree of $nf(\Rightarrow_{R_1 \cup R_2}, \xi_l)$—$\xi'_l$ is a $\Rightarrow_{R_1 \cup R_2}$-normal form, this implies that $\xi'_l = (\kappa_{h,q,l,p}(t))[z_{p',1}, \ldots, z_{p',s'} \leftarrow z_1, \ldots, z_{s'}] = nf(\Rightarrow_{R_1 \cup R_2}, \xi_l)$. But ξ'_l was shown above to be a proper subtree of $nf(\Rightarrow_{R_1 \cup R_2}, \xi_l)$, which leads to a contradiction. □

The previous lemma is weaker than Claim A.7, because it only tells something about calls $p(y_h, \cdots)$ that can actually result from reduction of $p'(q(t, y_1, \ldots, y_r), z_1, \ldots, z_{s'})$ for some state p' of M_2. The technique from the previous proof (using the semantics of $\kappa_{h,q,l,p}$-functions) could not be used for proving Claim A.7, because Theorem A.3 is not sufficient as complete characterization of the $\kappa_{h,q,l,p}$-functions, as it is also only concerned with constellations that actually can occur for given t and q. On the other hand, the proposition of Lemma A.8 is strong enough for our purposes, because for the mtt $M_{1;2}$ under construction it is, of course, sufficient to simulate correctly the computations of M_1 and M_2 in all actually occurring constellations. In fact, we could replace all later uses of Claim A.7 with references to Lemma A.8 by imposing appropriate restrictions. This, however would require a considerable overhead of auxiliary notions and, more disturbingly, blur the already complex proof of our main Lemma A.12 beyond recognition, so we will instead use Claim A.7 and refer to the proof of Lemma A.8 for justification.
We will use the following property—based on Definition A.6—of the rewriting system Pre.

Proposition A.9 For every $a \in \mathbf{N}$, $q \in Q^{(a+1)}$, $t \in T_\Sigma$, $h_1, h_2 \in [a]$ and $p_1, p_2 \in P$, if for some $l \in [rank_P(p_1) - 1]$, the normal form $nf(\Rightarrow_{Pre}, \kappa_{h_1,q,l,p_1}(t))$ contains y_{h_2,p_2}, then $(h_1, p_1) \prec_{q,t} (h_2, p_2)$. □

Now, we prove an important technical lemma about the correctness of precomputing translations of the first mtt's context parameter values with states of M_2 (given the correct context parameter values of these states on reaching M_1's context parameter positions), and then providing these translations in the appropriate positions of the result obtained by computing in ignorance of M_1's concrete context parameter values. The intuition, why this lemma holds, stems from the $\kappa_{h,q,1,p'}, \ldots, \kappa_{h,q,s',p'}$-functions computing the question mark positions as given in Figure 4.

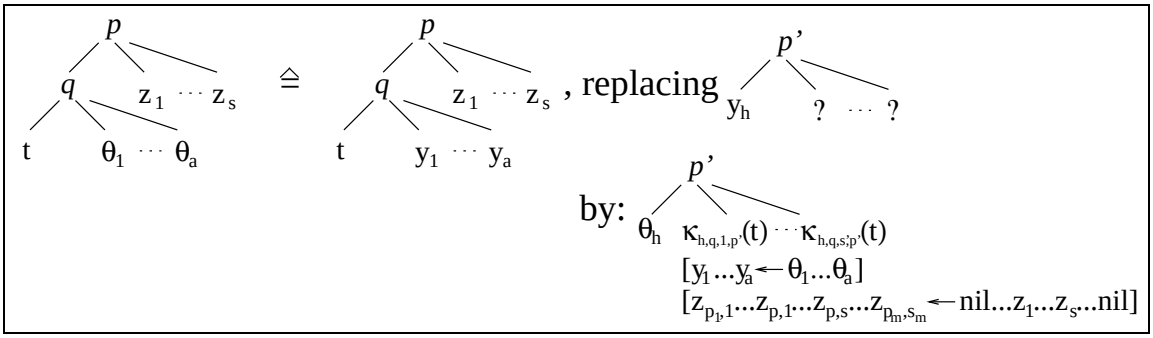

Figure 4: The intuition behind Lemma A.10 (modulo normal forms)

Lemma A.10 For every $a, s, r \in \mathbf{N}$, $q \in Q^{(a+1)}$, $p \in P^{(s+1)}$, $\theta_1, \ldots, \theta_a \in T_\Delta(Y_r)$, $t \in T_\Sigma$ and $\eta_{p_1,1}, \ldots, \eta_{p_m,s_m}$, such that for every $s' \in \mathbf{N}$, $p' \in P^{(s'+1)}$ and $l' \in [s']$, we have $\eta_{p',l'} =$ (if $p' = p$ then $z_{l'}$ else nil), the following holds:
$(nf(\Rightarrow_{Pre}, nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, q(t, y_1, \ldots, y_a)), z_1, \ldots, z_s))))$
$\quad [y_{h,p'} \leftarrow (nf(\Rightarrow_{R_2}, p'(y_h, \kappa_{h,q,1,p'}(t), \ldots, \kappa_{h,q,s',p'}(t))[y_1, \ldots, y_a \leftarrow \theta_1, \ldots, \theta_a]))$
$\qquad\qquad [z_{p_1,1}, \ldots, z_{p_m,s_m} \leftarrow \eta_{p_1,1}, \ldots, \eta_{p_m,s_m}],$
$\qquad\qquad h \in [a], s' \in \mathbf{N}, p' \in P^{(s'+1)}] =$
$nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, q(t, \theta_1, \ldots, \theta_a)), z_1, \ldots, z_s))$ □

Proof:

- Firstly, consider the normal form $nf(\Rightarrow_{R_1 \cup R_2}, p(q(t, y_1, \ldots, y_a), z_1, \ldots, z_s))$. We are going to prove by structural induction that for every subtree $\xi \in SEN(\emptyset, Y_a, Z_s)$ of this normal form, the following holds:
  $(nf(\Rightarrow_{Pre}, \xi))$
  $\quad [y_{h,p'} \leftarrow (nf(\Rightarrow_{R_2}, p'(y_h, \kappa_{h,q,1,p'}(t), \ldots, \kappa_{h,q,s',p'}(t))[y_1, \ldots, y_a \leftarrow \theta_1, \ldots, \theta_a]))$
  $\qquad\qquad [z_{p_1,1}, \ldots, z_{p_m,s_m} \leftarrow \eta_{p_1,1}, \ldots, \eta_{p_m,s_m}],$
  $\qquad\qquad h \in [a], s' \in \mathbf{N}, p' \in P^{(s'+1)}] =$
  $nf(\Rightarrow_{R_2}, \xi[y_1, \ldots, y_a \leftarrow \theta_1, \ldots, \theta_a])$
  The cases $\xi \in Z_s$ and $\xi = \omega(\xi_1, \ldots, \xi_n)$ (for $n \in \mathbf{N}$, $\omega \in \Omega^{(n)}$ and $\xi_1, \ldots, \xi_n \in SEN(\emptyset, Y_a, Z_s)$) are straightforward and thus omitted here.
  In the case $\xi = p''(y_{h'}, \xi_1, \ldots, \xi_{s''})$ with $y_{h'} \in Y_a$, $s'' \in \mathbf{N}$, $p'' \in P^{(s''+1)}$ and $\xi_1, \ldots, \xi_{s''} \in SEN(\emptyset, Y_a, Z_s)$, we reason as follows. Since $p''(y_{h'}, \xi_1, \ldots, \xi_{s''})$ is a subtree of $nf(\Rightarrow_{R_1 \cup R_2}, p(q(t, y_1, \ldots, y_a), z_1, \ldots, z_s))$, we can apply Theorem A.3 to obtain that for every $l \in [s'']$, we have $\kappa_{h',q,l,p''}(t) \in SEN(\emptyset, Y_a, \{z_{p,1}, \ldots, z_{p,s}\})$ and $nf(\Rightarrow_{R_1 \cup R_2}, \xi_l) = (\kappa_{h',q,l,p''}(t))[z_{p,1}, \ldots, z_{p,s} \leftarrow z_1, \ldots, z_s]$. As subtree of a normal

form, ξ_l is itself a normal form, and we can calculate for every $l \in [s'']$:
$(\kappa_{h',q,l,p''}(t))[y_1,\ldots,y_a \leftarrow \theta_1,\ldots,\theta_a][z_{p_1,1},\ldots,z_{p_m,s_m} \leftarrow \eta_{p_1,1},\ldots,\eta_{p_m,s_m}]$
$=$ (since the $\theta_1,\ldots,\theta_a$ do not contain any variables from Z_P,
$\kappa_{h',q,l,p''}(t) \in SEN(\emptyset, Y_a, \{z_{p,1},\ldots,z_{p,s}\})$ contains no variables from
$Z_P \backslash \{z_{p,1},\ldots,z_{p,s}\}$, and $\eta_{p,1},\ldots,\eta_{p,s} = z_1,\ldots,z_s$)
$(\kappa_{h',q,l,p''}(t))[z_{p,1},\ldots,z_{p,s} \leftarrow z_1,\ldots,z_s][y_1,\ldots,y_a \leftarrow \theta_1,\ldots,\theta_a]$
$=$ (by $\xi_l = (\kappa_{h',q,l,p''}(t))[z_{p,1},\ldots,z_{p,s} \leftarrow z_1,\ldots,z_s]$)
$\xi_l[y_1,\ldots,y_a \leftarrow \theta_1,\ldots,\theta_a]$

Now, we can calculate:
$(nf(\Rightarrow_{Pre}, p''(y_{h'},\xi_1,\ldots,\xi_{s''})))$
$[y_{h,p'} \leftarrow (nf(\Rightarrow_{R_2}, p'(y_h, \kappa_{h,q,1,p'}(t),\ldots,\kappa_{h,q,s',p'}(t))[y_1,\ldots,y_a \leftarrow \theta_1,\ldots,\theta_a]))$
$[z_{p_1,1},\ldots,z_{p_m,s_m} \leftarrow \eta_{p_1,1},\ldots,\eta_{p_m,s_m}],$
$h \in [a], s' \in \mathbf{N}, p' \in P^{(s'+1)}]$
$=$ (by $nf(\Rightarrow_{Pre}, p''(y_{h'},\xi_1,\ldots,\xi_{s''})) = y_{h',p''}$)
$(nf(\Rightarrow_{R_2}, p''(y_{h'}, \kappa_{h',q,1,p''}(t),\ldots,\kappa_{h',q,s'',p''}(t))[y_1,\ldots,y_a \leftarrow \theta_1,\ldots,\theta_a]))$
$[z_{p_1,1},\ldots,z_{p_m,s_m} \leftarrow \eta_{p_1,1},\ldots,\eta_{p_m,s_m}]$
$=$ (by substitution and since the $\eta_{p_1,1},\ldots,\eta_{p_m,s_m}$ are $\Rightarrow_{R_2}$-normal forms)
$nf(\Rightarrow_{R_2}, p''(\theta_{h'}, (\kappa_{h',q,1,p''}(t))[y_1,\ldots,y_a \leftarrow \theta_1,\ldots,\theta_a]$
$[z_{p_1,1},\ldots,z_{p_m,s_m} \leftarrow \eta_{p_1,1},\ldots,\eta_{p_m,s_m}],$
$\ldots,$
$(\kappa_{h',q,s'',p''}(t))[y_1,\ldots,y_a \leftarrow \theta_1,\ldots,\theta_a]$
$[z_{p_1,1},\ldots,z_{p_m,s_m} \leftarrow \eta_{p_1,1},\ldots,\eta_{p_m,s_m}]))$
$=$ (by the equivalence shown above for $\xi_l[y_1,\ldots,y_a \leftarrow \theta_1,\ldots,\theta_a]$, $l \in [s'']$)
$nf(\Rightarrow_{R_2}, p''(\theta_{h'}, \xi_1[y_1,\ldots,y_a \leftarrow \theta_1,\ldots,\theta_a],$
$\ldots,$
$\xi_{s''}[y_1,\ldots,y_a \leftarrow \theta_1,\ldots,\theta_a]))$
$=$ (by substitution)
$nf(\Rightarrow_{R_2}, p''(y_{h'},\xi_1,\ldots,\xi_{s''})[y_1,\ldots,y_a \leftarrow \theta_1,\ldots,\theta_a])$

- By confluence considerations, we know that:
  $nf(\Rightarrow_{R_1\cup R_2}, p(q(t,y_1,\ldots,y_a), z_1,\ldots,z_s)) =$
  $nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, q(t,y_1,\ldots,y_a)), z_1,\ldots,z_s))$

  The lemma is now proven by the following calculation:
  $(nf(\Rightarrow_{Pre}, nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, q(t,y_1,\ldots,y_a)), z_1,\ldots,z_s))))$
  $[y_{h,p'} \leftarrow (nf(\Rightarrow_{R_2}, p'(y_h, \kappa_{h,q,1,p'}(t),\ldots,\kappa_{h,q,s',p'}(t))[y_1,\ldots,y_a \leftarrow \theta_1,\ldots,\theta_a]))$
  $[z_{p_1,1},\ldots,z_{p_m,s_m} \leftarrow \eta_{p_1,1},\ldots,\eta_{p_m,s_m}],$
  $h \in [a], s' \in \mathbf{N}, p' \in P^{(s'+1)}]$
  $=$ (by the proposition proven in the first point,
  for $\xi = nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, q(t,y_1,\ldots,y_a)), z_1,\ldots,z_s))$)
  $nf(\Rightarrow_{R_2}, (nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, q(t,y_1,\ldots,y_a)), z_1,\ldots,z_s)))[y_1,\ldots,y_a \leftarrow \theta_1,\ldots,\theta_a])$
  $=$ (by confluence considerations)
  $nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, q(t,\theta_1,\ldots,\theta_a)), z_1,\ldots,z_s))$

□

The following claim states a similar property as the previous lemma, but for results of $\kappa_{h,q,l,p}$-functions. A restricted version of this claim could be proven using the semantics of $\kappa_{h,q,l,p}$-functions given by Theorem A.3 (both the necessary restriction and the proof technique of repeatedly applying Theorem A.3—to determine the appropriate normal forms—are very similar to Lemma A.8).

Claim A.11 For every $a, r \in \mathbf{N}$, $s \in \mathbf{N}_+$, $q \in Q^{(a+1)}$, $p \in P^{(s+1)}$, $h \in [a]$, $l \in [s]$, $\theta_1, \ldots, \theta_a \in T_\Delta(Y_r)$ and $t \in T_\Sigma$, the following holds:
$(nf(\Rightarrow_{Pre}, \kappa_{h,q,l,p}(t)))$
$[y_{h',p'} \leftarrow nf(\Rightarrow_{R_2}, p'(y_{h'}, \kappa_{h',q,1,p'}(t), \ldots, \kappa_{h',q,s',p'}(t))[y_1, \ldots, y_a \leftarrow \theta_1, \ldots, \theta_a]),$
$h' \in [a], s' \in \mathbf{N}, p' \in P^{(s'+1)}] =$
$nf(\Rightarrow_{R_2}, (\kappa_{h,q,l,p}(t))[y_1, \ldots, y_a \leftarrow \theta_1, \ldots, \theta_a])$ □

The remarks given below the proof of Lemma A.8 also apply here, so we will use the unrestricted version of Claim A.11 henceforth.

Now, we are going to present and prove the main lemma of this appendix. It consists of six statements that will be proven by nested simultaneous induction. In order to illustrate the proof structure, the statements are arranged in a nested fashion. We have a simultaneous induction of statements **(I)** and **(II)** for trees over the ranked alphabet Σ. Nested inside the first part of statement **(II)**, we have two statements **(i)** and **(ii)** that will also be proven by simultaneous induction, but for all possible right-hand sides built from states and output symbols of M_1 instead of all input trees. The principle of proof by simultaneous induction (cf. Definition A.4) can be straightforwardly extended to this setting, namely by following the recursive structure of right-hand sides, as given in Definition 2.1. The proof structure and dependencies used in the inductive argument for Lemma A.12 are shown in Figure 5.

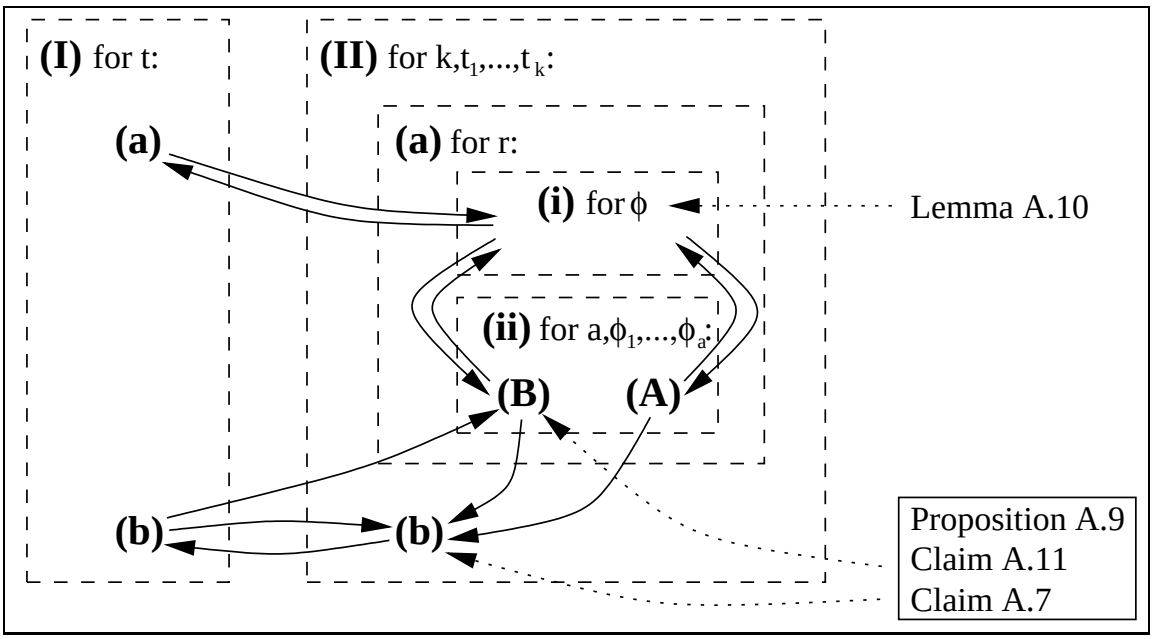

Figure 5: Proof structure for Lemma A.12

Lemma A.12 The following statements hold.
(I) For $\mathbf{t} \in \mathbf{T_\Sigma}$:
(a) for every $r, s \in \mathbf{N}$, $q \in Q^{(r+1)}$ and $p \in P^{(s+1)}$ holds:
$nf(\Rightarrow_{R_{1;2}}, (q,p)(t, y_{1,p_1}, \ldots, y_{r,p_m}, z_1, \ldots, z_s)) =$
$nf(\Rightarrow_{Pre}, nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, q(t, y_1, \ldots, y_r)), z_1, \ldots, z_s)))$
(b) for every $r, s \in \mathbf{N}_+$, $q \in Q^{(r+1)}$, $p \in P^{(s+1)}$, $h \in [r]$ and $l \in [s]$ holds:
$nf(\Rightarrow_{R_{1;2}}, (h_q, l_p)(t, y_{1,p_1}, \ldots, y_{r,p_m}, z_{p_1,1}, \ldots, z_{p_m,s_m})) = nf(\Rightarrow_{Pre}, \kappa_{h,q,l,p}(t))$
(II) For $\mathbf{k} \in \mathbf{N}, \mathbf{t_1}, \ldots, \mathbf{t_k} \in \mathbf{T_\Sigma}$:
(a) for every $r \in \mathbf{N}$ holds:
(i) for $\phi \in \mathbf{RHS}(\mathbf{Q}, \mathbf{\Delta}, \mathbf{k}, \mathbf{Y_r})$:
for every $s \in \mathbf{N}$ and $p \in P^{(s+1)}$ holds:
$nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, p(\phi, z_1, \ldots, z_s)))[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]) =$
$nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, \phi[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]), z_1, \ldots, z_s))$
(ii) for $\mathbf{a} \in \mathbf{N}, \phi_\mathbf{1}, \ldots, \phi_\mathbf{a} \in \mathbf{RHS}(\mathbf{Q}, \mathbf{\Delta}, \mathbf{k}, \mathbf{Y_r})$:
(A) for every $s \in \mathbf{N}$ (with $a \in rank(\Delta)$, $s+1 \in rank(P)$ or $a = 1$, $s = 0$) and $\psi \in RHS(P, \Omega, a, Z_s)$ holds:
$nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, \psi[x_u \leftarrow \phi_u, u \in [a]]))[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]) =$
$nf(\Rightarrow_{R_2}, \psi[x_u \leftarrow nf(\Rightarrow_{R_1}, \phi_u[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]), u \in [a]])$
(B) for every $i \in [k]$, $q \in Q^{(a+1)}$, $\mathcal{C} \subseteq [a] \times P$, $h \in [a]$, $s \in \mathbf{N}$ and $p \in P^{(s+1)}$, if not $(h,p) \prec^*_{q,t_i} \mathcal{C}$, then:
$nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(h,p,\mathcal{C}))[y_1, \ldots, y_a \leftarrow \phi_1, \ldots, \phi_a]))$
$[x \leftarrow x_i][x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]) =$
$nf(\Rightarrow_{R_2}, p(y_h, \kappa_{h,q,1,p}(t_i), \ldots, \kappa_{h,q,s,p}(t_i))$
$[y_u \leftarrow nf(\Rightarrow_{R_1}, \phi_u[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]), u \in [a]])$
(b) for every $r \in \mathbf{N}_+$ with $(r+1) \in rank(Q)$, $\phi \in RHS(Q, \Delta, k, Y_r)$, $o \in occ(\phi)$, such that $lab(\phi, o) \notin X_k$, for every $s \in \mathbf{N}_+$, $p \in P^{(s+1)}$ and $l \in [s]$ holds:
$nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, ctx_{p,\phi,k,r}(o,l)))[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]) =$
$\pi_{p,\phi,k,r}(t_1, \ldots, t_k, o, l)$ □

Proof: We prove **(I)** and **(II)** by simultaneous induction:
(I) $\Leftarrow$ (II) : Let $t = \sigma(t_1, \ldots, t_k)$ with $k \in \mathbf{N}$, $\sigma \in \Sigma^{(k)}$ and $t_1, \ldots, t_k \in T_\Sigma$, then:
(I)(a) :
$nf(\Rightarrow_{R_{1;2}}, (q,p)(\sigma(t_1, \ldots, t_k), y_{1,p_1}, \ldots, y_{r,p_m}, z_1, \ldots, z_s))$
$=$ (by $\Rightarrow_{R_{1;2}}$)
$nf(\Rightarrow_{R_{1;2}}, rhs_{(q,p),\sigma}[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$
$=$ (by definition of $rhs_{(q,p),\sigma}$ in Construction 3.6)
$nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, p(rhs_{q,\sigma}, z_1, \ldots, z_s)))[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$
$=$ (by confluence considerations)
$nf(\Rightarrow_{Pre}, nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, p(rhs_{q,\sigma}, z_1, \ldots, z_s)))[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]))$
$=$ (by induction hypothesis **(II)(a)(i)** for $\phi = rhs_{q,\sigma} \in RHS(Q, \Delta, k, Y_r)$)
$nf(\Rightarrow_{Pre}, nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, rhs_{q,\sigma}[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]), z_1, \ldots, z_s)))$
$=$ (by $\Rightarrow_{R_1}$)
$nf(\Rightarrow_{Pre}, nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, q(\sigma(t_1, \ldots, t_k), y_1, \ldots, y_r)), z_1, \ldots, z_s)))$
(I)(b) : We have by $\Rightarrow_{R_{1;2}}$:
$nf(\Rightarrow_{R_{1;2}}, (h_q, l_p)(\sigma(t_1, \ldots, t_k), y_{1,p_1}, \ldots, y_{r,p_m}, z_{p_1,1}, \ldots, z_{p_m,s_m})) =$
$nf(\Rightarrow_{R_{1;2}}, rhs_{(h_q,l_p),\sigma}[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$

Now, there are two possible cases:

- If $rhs_{q,\sigma} \in RHS(Q, \Delta, k, Y_r)$ does *not* contain the context variable y_h, then by Construction 3.6, we have $rhs_{(h_q,l_p),\sigma} = nil$ and thus:
  $nf(\Rightarrow_{R_{1;2}}, rhs_{(h_q,l_p),\sigma}[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$
  $= $ (by $rhs_{(h_q,l_p),\sigma} = nil = nf(\Rightarrow_{Pre}, nil)$)
  $nf(\Rightarrow_{Pre}, nil)$
  $= $ (by definition of $\kappa_{h,q,l,p}$)
  $nf(\Rightarrow_{Pre}, \kappa_{h,q,l,p}(\sigma(t_1, \ldots, t_k)))$

- If $rhs_{q,\sigma} \in RHS(Q, \Delta, k, Y_r)$ *does* contain the context variable y_h, and the unique such occurrence (M_1 is non-copying!) is $o \in occ(rhs_{q,\sigma})$, then we have:
  $nf(\Rightarrow_{R_{1;2}}, rhs_{(h_q,l_p),\sigma}[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$
  $= $ (by definition of $rhs_{(h_q,l_p),\sigma}$ in Construction 3.6)
  $nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pre \cup Pair}, ctx_{p,rhs_{q,\sigma},k,r}(o, l)))[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$
  $= $ (by confluence considerations)
  $nf(\Rightarrow_{Pre}, nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, ctx_{p,rhs_{q,\sigma},k,r}(o, l)))[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]))$
  $= $ (by induction hypothesis **(II)(b)** for $\phi = rhs_{q,\sigma}$)
  $nf(\Rightarrow_{Pre}, \pi_{p,rhs_{q,\sigma},k,r}(t_1, \ldots, t_k, o, l))$
  $= $ (by definition of $\kappa_{h,q,l,p}$)
  $nf(\Rightarrow_{Pre}, \kappa_{h,q,l,p}(\sigma(t_1, \ldots, t_k)))$

(II) $\Leftarrow$ **(I) :** We first prove **(a)** by simultaneous induction of **(i)** and **(ii)**, then we can use **(a)** for proving **(b)**. In doing so, we can assume **(I)** for the fixed $t_1, \ldots, t_k$.

(II)(a) : simultaneous induction of **(i)** and **(ii)**

(i) $\Leftarrow$ **(ii) :** by case analysis on $\phi \in RHS(Q, \Delta, k, Y_r)$:

- $\phi = y_h \in Y_r$:
  $nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, p(y_h, z_1, \ldots, z_s)))[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$
  $= $ (by substitution and normal form)
  $p(y_h, z_1, \ldots, z_s)$
  $= $ (by substitution and normal form)
  $nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, y_h[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]), z_1, \ldots, z_s))$

- $\phi = \delta(\phi_1, \ldots, \phi_a), a \in \mathbf{N}, \delta \in \Delta^{(a)}, \phi_1, \ldots, \phi_a \in RHS(Q, \Delta, k, Y_r)$:
  $nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, p(\delta(\phi_1, \ldots, \phi_a), z_1, \ldots, z_s)))[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$
  $= $ (with rule $p(\delta(x_1, \ldots, x_a), z_1, \ldots, z_s) \to rhs_{p,\delta}$ in R_2)
  $nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, rhs_{p,\delta}[x_1, \ldots, x_a \leftarrow \phi_1, \ldots, \phi_a]))[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$
  $= $ (by induction hypothesis **(ii)(A)** for $\psi = rhs_{p,\delta} \in RHS(P, \Omega, a, Z_s)$)
  $nf(\Rightarrow_{R_2}, rhs_{p,\delta}[x_u \leftarrow nf(\Rightarrow_{R_1}, \phi_u[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]), u \in [a]])$
  $= $ (by the above rule in R_2)
  $nf(\Rightarrow_{R_2}, p(\delta(nf(\Rightarrow_{R_1}, \phi_1[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]),$
  $\ldots,$
  $nf(\Rightarrow_{R_1}, \phi_a[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])), z_1, \ldots, z_s))$
  $= $ (by substitution and normal form)
  $nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, \delta(\phi_1, \ldots, \phi_a)[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]),$
  $z_1, \ldots, z_s))$

- $\phi = q(x_i, \phi_1, \ldots, \phi_a), a \in \mathbf{N}, q \in Q^{(a+1)}, x_i \in X_k, \phi_1, \ldots, \phi_a \in RHS(Q, \Delta, k, Y_r)$:
  By definition of $Pair$, we have:
  $nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, p(q(x_i, \phi_1, \ldots, \phi_a), z_1, \ldots, z_s)))$
  $\quad [x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]) =$
  $nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, (q,p)(x, nest_q(1, p_1, \emptyset), \ldots, nest_q(a, p_m, \emptyset), z_1, \ldots, z_s)$
  $\quad [z_{p_1,1}, \ldots, z_{p_m,s_m} \leftarrow \eta_{p_1,1}, \ldots, \eta_{p_m,s_m}]$
  $\quad [x, y_1, \ldots, y_a \leftarrow x_i, \phi_1, \ldots, \phi_a]))$
  $\quad [x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]),$
  where for every $s' \in \mathbf{N}$, $p' \in P^{(s'+1)}$ and $l' \in [s']$, we have $\eta_{p',l'} =$ (if $p' = p$ then $z_{l'}$ else nil). By substitution and confluence considerations, this is equivalent to:

  $nf(\Rightarrow_{R_{1;2}}, (q,p)(x_i, (nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(1, p_1, \emptyset))[y_1, \ldots, y_a \leftarrow \phi_1, \ldots, \phi_a]))$
  $\quad [x \leftarrow x_i][z_{p_1,1}, \ldots, z_{p_m,s_m} \leftarrow \eta_{p_1,1}, \ldots, \eta_{p_m,s_m}],$
  $\quad \ldots,$
  $\quad (nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(a, p_m, \emptyset))[y_1, \ldots, y_a \leftarrow \phi_1, \ldots, \phi_a]))$
  $\quad [x \leftarrow x_i][z_{p_1,1}, \ldots, z_{p_m,s_m} \leftarrow \eta_{p_1,1}, \ldots, \eta_{p_m,s_m}],$
  $\quad z_1, \ldots, z_s)$
  $\quad [x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$
  $=$ (by confluence considerations and since the
  $\quad \eta_{p_1,1}, \ldots, \eta_{p_m,s_m}$ are $\Rightarrow_{R_{1;2}}$-normal forms that do not contain $x_1, \ldots, x_k$)
  $(nf(\Rightarrow_{R_{1;2}}, (q,p)(t_i, y_{1,p_1}, \ldots, y_{a,p_m}, z_1, \ldots, z_s)))$
  $\quad [y_{h,p'} \leftarrow (nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(h, p', \emptyset))[y_1, \ldots, y_a \leftarrow \phi_1, \ldots, \phi_a]))$
  $\quad [x \leftarrow x_i][x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]))$
  $\quad [z_{p_1,1}, \ldots, z_{p_m,s_m} \leftarrow \eta_{p_1,1}, \ldots, \eta_{p_m,s_m}],$
  $\quad h \in [a], s' \in \mathbf{N}, p' \in P^{(s'+1)}]$
  $=$ (by induction hypothesis **(I)(a)** for t_i and
  $\quad$ induction hypothesis **(ii)(B)** on the normal forms replaced for the $y_{h,p'}$,
  $\quad$ using that not $(h, p') \prec^*_{q,t_i} \emptyset$)
  $(nf(\Rightarrow_{Pre}, nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, q(t_i, y_1, \ldots, y_a)), z_1, \ldots, z_s))))$
  $\quad [y_{h,p'} \leftarrow (nf(\Rightarrow_{R_2}, p'(y_h, \kappa_{h,q,1,p'}(t_i), \ldots, \kappa_{h,q,s',p'}(t_i))$
  $\quad\quad [y_u \leftarrow nf(\Rightarrow_{R_1}, \phi_u[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]), u \in [a]]))$
  $\quad [z_{p_1,1}, \ldots, z_{p_m,s_m} \leftarrow \eta_{p_1,1}, \ldots, \eta_{p_m,s_m}],$
  $\quad h \in [a], s' \in \mathbf{N}, p' \in P^{(s'+1)}]$
  $=$ (by Lemma A.10 with $\theta_u = nf(\Rightarrow_{R_1}, \phi_u[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]) \in T_\Delta(Y_r)$,
  $\quad$ for every $u \in [a]$)
  $nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, q(t_i, nf(\Rightarrow_{R_1}, \phi_1[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]),$
  $\quad \ldots,$
  $\quad nf(\Rightarrow_{R_1}, \phi_a[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]))),$
  $\quad z_1, \ldots, z_s))$
  $=$ (by substitution and confluence of $\Rightarrow_{R_1}$)
  $nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, q(x_i, \phi_1, \ldots, \phi_a)[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]),$
  $\quad z_1, \ldots, z_s))$

(ii) $\Leftarrow$ (i) :

(ii)(A) : a straightforward structural induction on $\psi \in RHS(P, \Omega, a, Z_s)$, using induction hypotheses **(i)** on ϕ_i in the case $\psi = p(x_i, \cdots)$ for $i \in [a]$

(ii)(B) : For fixed $i \in [k]$ and $q \in Q^{(a+1)}$, we proceed by proving the statement for every $\mathcal{C} \in \mathcal{P}([a] \times P)$, by induction on the reversed subset-order:

- $\mathcal{C} = [a] \times P$:
  In this case, we have $(h,p) \prec^*_{q,t_i} \mathcal{C}$ for every $h \in [a]$ and $p \in P$, thus nothing remains to prove.

- $\mathcal{C} \subset [a] \times P$:
  If not $(h,p) \prec^*_{q,t_i} \mathcal{C}$, then we know $(h,p) \notin \mathcal{C}$ and thus:

$$
\begin{aligned}
&nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(h,p,\mathcal{C}))[y_1,\ldots,y_a \leftarrow \phi_1,\ldots,\phi_a])) \\
&\qquad [x \leftarrow x_i][x_1,\ldots,x_k \leftarrow t_1,\ldots,t_k]) \\
&\quad = \text{ (by definition of } nest_q) \\
&nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, p(y_h, (h_q, 1_p)(x, nest_q(1,p_1,\mathcal{C} \cup \{(h,p)\}), \\
&\qquad\qquad \ldots, \\
&\qquad\qquad nest_q(a,p_m,\mathcal{C} \cup \{(h,p)\}), \\
&\qquad\qquad z_{p_1,1},\ldots,z_{p_m,s_m}), \\
&\qquad \ldots, \\
&\qquad (h_q,s_p)(x, nest_q(1,p_1,\mathcal{C} \cup \{(h,p)\}), \\
&\qquad\qquad \ldots, \\
&\qquad\qquad nest_q(a,p_m,\mathcal{C} \cup \{(h,p)\}), \\
&\qquad\qquad z_{p_1,1},\ldots,z_{p_m,s_m})) \\
&\qquad [y_1,\ldots,y_a \leftarrow \phi_1,\ldots,\phi_a])) \\
&\qquad [x \leftarrow x_i][x_1,\ldots,x_k \leftarrow t_1,\ldots,t_k]) \\
&\quad = \text{ (by substitution and confluence considerations)} \\
&nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, p(\phi_h, z_1,\ldots,z_s))) \\
&\quad [z_l \leftarrow (h_q,l_p)(x, nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(1,p_1,\mathcal{C} \cup \{(h,p)\}))[y_u \leftarrow \phi_u, u \in [a]]), \\
&\qquad\qquad \ldots, \\
&\qquad\qquad nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(a,p_m,\mathcal{C} \cup \{(h,p)\}))[y_u \leftarrow \phi_u, u \in [a]]), \\
&\qquad\quad z_{p_1,1},\ldots,z_{p_m,s_m}), \\
&\qquad l \in [s]] \\
&\quad [x \leftarrow x_i][x_1,\ldots,x_k \leftarrow t_1,\ldots,t_k]) \\
&\quad = \text{ (by substitution and confluence considerations)} \\
&(nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, p(\phi_h, z_1,\ldots,z_s)))[x_1,\ldots,x_k \leftarrow t_1,\ldots,t_k])) \\
&\quad [z_l \leftarrow nf(\Rightarrow_{R_{1;2}}, (h_q,l_p)(x_i, \\
&\qquad\quad (nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(1,p_1,\mathcal{C} \cup \{(h,p)\}))[y_u \leftarrow \phi_u, u \in [a]]))[x \leftarrow x_i], \\
&\qquad\qquad \ldots, \\
&\qquad\quad (nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(a,p_m,\mathcal{C} \cup \{(h,p)\}))[y_u \leftarrow \phi_u, u \in [a]]))[x \leftarrow x_i], \\
&\qquad\quad z_{p_1,1},\ldots,z_{p_m,s_m})[x_1,\ldots,x_k \leftarrow t_1,\ldots,t_k]), \\
&\qquad l \in [s]] \\
&\quad = \text{ (by induction hypothesis } \textbf{(i)} \text{ for } \phi_h) \\
&(nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, \phi_h[x_1,\ldots,x_k \leftarrow t_1,\ldots,t_k]), z_1,\ldots,z_s))) \\
&\quad [z_l \leftarrow nf(\Rightarrow_{R_{1;2}}, (h_q,l_p)(x_i, \\
&\qquad\quad (nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(1,p_1,\mathcal{C} \cup \{(h,p)\}))[y_u \leftarrow \phi_u, u \in [a]]))[x \leftarrow x_i], \\
&\qquad\qquad \ldots, \\
&\qquad\quad (nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(a,p_m,\mathcal{C} \cup \{(h,p)\}))[y_u \leftarrow \phi_u, u \in [a]]))[x \leftarrow x_i], \\
&\qquad\quad z_{p_1,1},\ldots,z_{p_m,s_m})[x_1,\ldots,x_k \leftarrow t_1,\ldots,t_k]), \\
&\qquad l \in [s]]
\end{aligned}
$$

We name this expression **(+)** and continue to calculate on the terms substituted for the z_l with $l \in [s]$:

$nf(\Rightarrow_{R_{1;2}}, (h_q, l_p)(x_i,$
$\quad (nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(1, p_1, \mathcal{C} \cup \{(h,p)\}))[y_u \leftarrow \phi_u, u \in [a]]))[x \leftarrow x_i],$
$\quad\quad \ldots,$
$\quad (nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(a, p_m, \mathcal{C} \cup \{(h,p)\}))[y_u \leftarrow \phi_u, u \in [a]]))[x \leftarrow x_i],$
$\quad z_{p_1,1}, \ldots, z_{p_m,s_m})[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$
$\quad =$ (by substitution and confluence considerations)
$(nf(\Rightarrow_{R_{1;2}}, (h_q, l_p)(t_i, y_{1,p_1}, \ldots, y_{a,p_m}, z_{p_1,1}, \ldots, z_{p_m,s_m})))$
$\quad [y_{h',p'} \leftarrow nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(h', p', \mathcal{C} \cup \{(h,p)\}))[y_u \leftarrow \phi_u, u \in [a]]))$
$\quad\quad\quad\quad\quad\quad\quad\quad [x \leftarrow x_i][x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]),$
$\quad\quad\quad h' \in [a], s' \in \mathbf{N}, p' \in P^{(s'+1)}]$
$\quad =$ (by induction hypothesis **(I)(b)** for t_i)
$(nf(\Rightarrow_{Pre}, \kappa_{h,q,l,p}(t_i)))$
$\quad [y_{h',p'} \leftarrow nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(h', p', \mathcal{C} \cup \{(h,p)\}))[y_u \leftarrow \phi_u, u \in [a]]))$
$\quad\quad\quad\quad\quad\quad\quad\quad [x \leftarrow x_i][x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]),$
$\quad\quad\quad h' \in [a], s' \in \mathbf{N}, p' \in P^{(s'+1)}]$

Now, we want to use the induction hypothesis on $\mathcal{C} \cup \{(h,p)\}$, as expressed below[7], in order to show that for every $l \in [s]$, this is equivalent to the following expression **(++)**:

$(nf(\Rightarrow_{Pre}, \kappa_{h,q,l,p}(t_i)))$
$\quad [y_{h',p'} \leftarrow nf(\Rightarrow_{R_2}, p'(y_{h'}, \kappa_{h',q,1,p'}(t_i), \ldots, \kappa_{h',q,s',p'}(t_i))$
$\quad\quad\quad\quad\quad\quad\quad\quad [y_u \leftarrow nf(\Rightarrow_{R_1}, \phi_u[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]), u \in [a]]),$
$\quad\quad\quad h' \in [a], s' \in \mathbf{N}, p' \in P^{(s'+1)}]$

Therefore, we need to establish that for every $y_{h',p'}$ occurring in $nf(\Rightarrow_{Pre}, \kappa_{h,q,l,p}(t_i))$, we do not have $(h',p') \prec^*_{q,t_i} \mathcal{C} \cup \{(h,p)\}$. This can be shown by contradiction. Assume that for such a $y_{h',p'}$—for which by Proposition A.9 must hold: $(h,p) \prec_{q,t_i} (h',p')$—we have $(h',p') \prec^*_{q,t_i} \mathcal{C} \cup \{(h,p)\}$. This would imply $(h,p) \prec^+_{q,t_i} \mathcal{C} \cup \{(h,p)\}$, which cannot be true, because by the assumption at the very beginning of the present case, we know that not $(h,p) \prec^*_{q,t_i} \mathcal{C}$, and by Claim A.7 we know that not $(h,p) \prec^+_{q,t_i} (h,p)$. Let us now return to expression **(+)**. Replacing the terms substituted there for z_l (with $l \in [s]$) by the terms **(++)** (shown above to be equivalent), gives us the following equivalent of **(+)**:

$(nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, \phi_h[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]), z_1, \ldots, z_s)))$
$\quad [z_l \leftarrow (nf(\Rightarrow_{Pre}, \kappa_{h,q,l,p}(t_i)))$
$\quad\quad [y_{h',p'} \leftarrow nf(\Rightarrow_{R_2}, p'(y_{h'}, \kappa_{h',q,1,p'}(t_i), \ldots, \kappa_{h',q,s',p'}(t_i))$
$\quad\quad\quad\quad\quad\quad [y_u \leftarrow nf(\Rightarrow_{R_1}, \phi_u[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]),$
$\quad\quad\quad\quad\quad\quad\quad\quad u \in [a]]),$
$\quad\quad\quad\quad h' \in [a], s' \in \mathbf{N}, p' \in P^{(s'+1)}],$
$\quad\quad l \in [s]]$

[7] For every $h' \in [a]$, $s' \in \mathbf{N}$ and $p' \in P^{(s'+1)}$, if not $(h',p') \prec^*_{q,t_i} \mathcal{C} \cup \{(h,p)\}$, then:
$nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(h', p', \mathcal{C} \cup \{(h,p)\}))[y_u \leftarrow \phi_u, u \in [a]]))$
$\quad\quad [x \leftarrow x_i][x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]) =$
$nf(\Rightarrow_{R_2}, p'(y_{h'}, \kappa_{h',q,1,p'}(t_i), \ldots, \kappa_{h',q,s',p'}(t_i))[y_u \leftarrow nf(\Rightarrow_{R_1}, \phi_u[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]), u \in [a]])$

$=$ (by Claim A.11 on the normal forms replaced for $z_1, \ldots, z_s$,
with $\theta_u = nf(\Rightarrow_{R_1}, \phi_u[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]) \in T_\Delta(Y_r)$, $u \in [a]$)
$(nf(\Rightarrow_{R_2}, p(nf(\Rightarrow_{R_1}, \phi_h[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]), z_1, \ldots, z_s)))$
$[z_l \leftarrow nf(\Rightarrow_{R_2}, (\kappa_{h,q,l,p}(t_i))[y_u \leftarrow nf(\Rightarrow_{R_1}, \phi_u[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]), u \in [a]]),$
$l \in [s]]$
$=$ (by confluence considerations and substitution)
$nf(\Rightarrow_{R_2}, p(y_h, \kappa_{h,q,1,p}(t_i), \ldots, \kappa_{h,q,s,p}(t_i))$
$[y_u \leftarrow nf(\Rightarrow_{R_1}, \phi_u[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]), u \in [a]])$

(II)(b) : In proving **(II)(b)**, we may use the statements from **(II)(a)(ii)**, which have previously been proven for every $a \in \mathbf{N}$, $\phi_1, \ldots, \phi_a \in RHS(Q, \Delta, k, Y_r)$, but for the fixed $t_1, \ldots, t_k$ under consideration in **(II)(b)**. For fixed $r \in \mathbf{N}_+$ with $(r+1) \in rank(Q)$, and $\phi \in RHS(Q, \Delta, k, Y_r)$, we proceed by induction on the prefix-order of occurrences in ϕ:

- In the base case ε, we have:
  $nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, ctx_{p,\phi,k,r}(\varepsilon, l)))[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$
  $=$ (by definition of $ctx_{p,\phi,k,r}$)
  $z_{p,l}$
  $=$ (by definition of $\pi_{p,\phi,k,r}$)
  $\pi_{p,\phi,k,r}(t_1, \ldots, t_k, \varepsilon, l)$

- The inductive case $oi \in occ(\phi)$—with $o \in occ(\phi)$, $i \in \mathbf{N}_+$ and $lab(\phi, oi) \notin X_k$—is shown by case distinction on $lab(\phi, o)$:

  - $lab(\phi, o) = \delta$ with $a \in \mathbf{N}$, $\delta \in \Delta^{(a)}$ and $i \in [a]$:

    * If there is no $p(x_i, \cdots)$-call in the δ-rules of M_2, then we have:
      $nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, ctx_{p,\phi,k,r}(oi, l)))[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$
      $=$ (by definition of $ctx_{p,\phi,k,r}$)
      nil
      $=$ (by definition of $\pi_{p,\phi,k,r}$)
      $\pi_{p,\phi,k,r}(t_1, \ldots, t_k, oi, l)$
    * Otherwise, if the unique such call (remember that M_2 is weakly single-use) looks, with $s' \in \mathbf{N}$, $p' \in P^{(s'+1)}$ and $\psi_1, \ldots, \psi_s \in RHS(P, \Omega, a, Z_{s'})$, as follows:
      $$p'(\delta(x_1, \ldots, x_a), z_1, \ldots, z_{s'}) \to \cdots p(x_i, \psi_1, \ldots, \psi_s) \cdots,$$
      then we can calculate:
      $nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, ctx_{p,\phi,k,r}(oi, l)))[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$
      $=$ (by definition of $ctx_{p,\phi,k,r}$)
      $nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair},$
      $\psi_l[x_1, \ldots, x_a \leftarrow sub(\phi, o1), \ldots, sub(\phi, oa),$
      $z_1, \ldots, z_{s'} \leftarrow ctx_{p',\phi,k,r}(o, 1), \ldots, ctx_{p',\phi,k,r}(o, s')]))$
      $[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$
      $=$ (by substitution and confluence considerations)

$(nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, \psi_l[x_1, \ldots, x_a \leftarrow sub(\phi, o1), \ldots, sub(\phi, oa)]))$
$[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]))$
$[z_w \leftarrow nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, ctx_{p',\phi,k,r}(o, w)))$
$[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]),$
$w \in [s']]$
$=$ (by the induction hypothesis on o)
$(nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, \psi_l[x_1, \ldots, x_a \leftarrow sub(\phi, o1), \ldots, sub(\phi, oa)]))$
$[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]))$
$[z_w \leftarrow \pi_{p',\phi,k,r}(t_1, \ldots, t_k, o, w), w \in [s']]$
$=$ (by the previously proven statement **(a)(ii)(A)** for
$sub(\phi, o1), \ldots, sub(\phi, oa) \in RHS(Q, \Delta, k, Y_r)$,
$s' + 1 \in rank(P)$ and $\psi_l \in RHS(P, \Omega, a, Z_{s'})$)
$(nf(\Rightarrow_{R_2}, \psi_l[x_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, ou))[x_v \leftarrow t_v, v \in [k]]), u \in [a]]))$
$[z_w \leftarrow \pi_{p',\phi,k,r}(t_1, \ldots, t_k, o, w), w \in [s']]$
$=$ (since the $\pi_{p',\phi,k,r}(t_1, \ldots, t_k, o, w)$, for $w \in [s']$, are
$\Rightarrow_{R_2}$-normal forms, by the remark below Definition A.1)
$nf(\Rightarrow_{R_2}, \psi_l[x_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, ou))[x_v \leftarrow t_v, v \in [k]]), u \in [a]]$
$[z_w \leftarrow \pi_{p',\phi,k,r}(t_1, \ldots, t_k, o, w), w \in [s']])$
$=$ (by composing substitutions, using that for every $w \in [s']$, $u \in [a]$,
no z_w occurs in $nf(\Rightarrow_{R_1}, (sub(\phi, ou))[x_v \leftarrow t_v, v \in [k]]))$
$nf(\Rightarrow_{R_2}, \psi_l[x_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, ou))[x_v \leftarrow t_v, v \in [k]]), u \in [a],$
$z_w \leftarrow \pi_{p',\phi,k,r}(t_1, \ldots, t_k, o, w), w \in [s']])$
$=$ (by definition of $\pi_{p,\phi,k,r}$, taking into account the unique
$p(x_i, \cdots)$-call in the δ-rules of M_2 as given above)
$\pi_{p,\phi,k,r}(t_1, \ldots, t_k, oi, l)$

– $lab(\phi, o) = q$ with $a \in \mathbf{N}$, $q \in Q^{(a+1)}$, $2 \leq i \leq a + 1$ and $lab(\phi, o1) = x_b \in X_k$:
We calculate:
$nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, ctx_{p,\phi,k,r}(oi, l)))[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$
$=$ (by definition of $ctx_{p,\phi,k,r}$)
$nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair},$
$((i-1)_q, l_p)(x, nest_q(1, p_1, \{(i-1, p)\}), \ldots, nest_q(a, p_m, \{(i-1, p)\}),$
$z_{p_1,1}, \ldots, z_{p_m,s_m})$
$[x \leftarrow x_b,$
$y_1, \ldots, y_a \leftarrow sub(\phi, o2), \ldots, sub(\phi, o(a+1)),$
$z_{p_1,1}, \ldots, z_{p_m,s_m} \leftarrow ctx_{p_1,\phi,k,r}(o, 1), \ldots, ctx_{p_m,\phi,k,r}(o, s_m)]))$
$[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k])$
$=$ (by substitution and confluence considerations)
$(nf(\Rightarrow_{R_{1;2}}, ((i-1)_q, l_p)(t_b, y_{1,p_1}, \ldots, y_{a,p_m}, z_{p_1,1}, \ldots, z_{p_m,s_m})))$
$[y_{h,p'} \leftarrow nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(h, p', \{(i-1, p)\}))$
$[y_u \leftarrow sub(\phi, o(u+1)), u \in [a]]))$
$[x \leftarrow x_b][x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]),$
$h \in [a], s' \in \mathbf{N}, p' \in P^{(s'+1)}]$
$[z_{p',w} \leftarrow nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, ctx_{p',\phi,k,r}(o, w)))[x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]),$
$s' \in \mathbf{N}, p' \in P^{(s'+1)}, w \in [s']]$

$= $ (by induction hypothesis **(I)(b)** for t_b and
the induction hypothesis on o)

$(nf(\Rightarrow_{Pre}, \kappa_{i-1,q,l,p}(t_b)))$
$[y_{h,p'} \leftarrow nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, (nest_q(h,p',\{(i-1,p)\}))$
$[y_u \leftarrow sub(\phi, o(u+1)), u \in [a]]))$
$[x \leftarrow x_b][x_1, \ldots, x_k \leftarrow t_1, \ldots, t_k]),$
$h \in [a], s' \in \mathbf{N}, p' \in P^{(s'+1)}]$
$[z_{p',w} \leftarrow \pi_{p',\phi,k,r}(t_1, \ldots, t_k, o, w), s' \in \mathbf{N}, p' \in P^{(s'+1)}, w \in [s']]$

Now, we claim that for every variable $y_{h,p'}$ that occurs in $nf(\Rightarrow_{Pre}, \kappa_{i-1,q,l,p}(t_b))$, we do not have $(h,p') \prec^*_{q,t_b} (i-1,p)$. The reasoning is by contradiction. If for such a $y_{h,p'}$—for which by Proposition A.9 must hold: $(i-1,p) \prec_{q,t_b} (h,p')$—we would have $(h,p') \prec^*_{q,t_b} (i-1,p)$, this would imply $(i-1,p) \prec^+_{q,t_b} (i-1,p)$, which by Claim A.7 is impossible. This fact allows us to apply—on the normal forms replaced for the $y_{h,p'}$—the previously proven statement **(a)(ii)(B)** for $sub(\phi, o2), \ldots, sub(\phi, o(a+1)) \in RHS(Q, \Delta, k, Y_r)$ and obtain:

$(nf(\Rightarrow_{Pre}, \kappa_{i-1,q,l,p}(t_b)))$
$[y_{h,p'} \leftarrow nf(\Rightarrow_{R_2}, p'(y_h, \kappa_{h,q,1,p'}(t_b), \ldots, \kappa_{h,q,s',p'}(t_b))$
$[y_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, o(u+1)))[x_v \leftarrow t_v, v \in [k]]),$
$u \in [a]]),$
$h \in [a], s' \in \mathbf{N}, p' \in P^{(s'+1)}]$
$[z_{p',w} \leftarrow \pi_{p',\phi,k,r}(t_1, \ldots, t_k, o, w), s' \in \mathbf{N}, p' \in P^{(s'+1)}, w \in [s']]$
$= $ (by Claim A.11 with
$\theta_u = nf(\Rightarrow_{R_1}, (sub(\phi, o(u+1)))[x_v \leftarrow t_v, v \in [k]]) \in T_\Delta(Y_r),\ u \in [a])$
$(nf(\Rightarrow_{R_2}, (\kappa_{i-1,q,l,p}(t_b))[y_u \leftarrow nf(\Rightarrow_{R_1}, (sub(\phi, o(u+1)))[x_v \leftarrow t_v, v \in [k]]),$
$u \in [a]]))$
$[z_{p',w} \leftarrow \pi_{p',\phi,k,r}(t_1, \ldots, t_k, o, w),\ s' \in \mathbf{N}, p' \in P^{(s'+1)}, w \in [s']]$
$= $ (by definition of $\pi_{p,\phi,k,r}$)
$\pi_{p,\phi,k,r}(t_1, \ldots, t_k, oi, l)$ □

Using the previous lemma, we can now prove the correctness of Construction 3.6.

Theorem A.13 $\tau(M_1); \tau(M_2) = \tau(M_{1;2})$ □

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

$\tau(M_{1;2})(t)$
$= $ (by definition of $\tau(M_{1;2})$)
$nf(\Rightarrow_{R_{1;2}}, e_{1;2}[x_1 \leftarrow t])$
$= $ (by definition of $e_{1;2}$ in Construction 3.6)
$nf(\Rightarrow_{R_{1;2}}, (nf(\Rightarrow_{R_2 \cup Pair}, e_2[x_1 \leftarrow e_1]))[x_1 \leftarrow t])$
$= $ (by statement **(II)(a)(ii)(A)** of Lemma A.12 for $k=1$, $t_1 = t \in T_\Sigma$, $r=0$, $a=1$, $\phi_1 = e_1 \in RHS(Q, \Delta, 1, \emptyset)$, $s=0$ and $\psi = e_2 \in RHS(P, \Omega, 1, \emptyset)$)
$nf(\Rightarrow_{R_2}, e_2[x_1 \leftarrow nf(\Rightarrow_{R_1}, e_1[x_1 \leftarrow t])])$
$= $ (by definition of $\tau(M_1)$)
$nf(\Rightarrow_{R_2}, e_2[x_1 \leftarrow \tau(M_1)(t)])$
$= $ (by definition of $\tau(M_2)$)
$\tau(M_2)(\tau(M_1)(t))$ □