MATHEMATICAL ENGINEERING
TECHNICAL REPORTS

Composing Stack-Attributed Tree Transducers

Keisuke Nakano

(Communicated by Zhenjiang HU)

METR 2004–01 January 2004

DEPARTMENT OF MATHEMATICAL INFORMATICS
GRADUATE SCHOOL OF INFORMATION SCIENCE AND TECHNOLOGY
THE UNIVERSITY OF TOKYO
BUNKYO-KU, TOKYO 113-8656, JAPAN

WWW page: http://www.i.u-tokyo.ac.jp/mi/mi-e.htm

Composing Stack-Attributed Tree Transducers

Keisuke Nakano

Department of Mathmatical Informatics, University of Tokyo
ksk@mist.i.u-tokyo.ac.jp

January 2004

Abstract. This paper presents a composition method for stack-attributed tree transducers. Stack-attributed tree transducers extend attributed tree transducers with a pushdown stack device for attribute values. Stack-attributed tree transducers are more powerful than attributed tree transducers due to the stack mechanism. We extend the existing composition method for attributed tree transducers to the composition method for stack-attributed tree transducers. The composition method is proved to be correct and to enjoy a closure property.

1 Introduction

Attribute grammars were introduced by Knuth [Knu68, Knu71] to describe semantics of context-free languages. They specify the meaning of each derivation tree by allocating values to attributes associated with every node of the tree. The framework of attribute grammars has been utilized for the development of compilers [ASU86, Far84, RM89, KHZ82], editing environments [Rep84, HT85] and program transformations [Joh87, CDPR99].

Composition of attribute grammars is one of the well-studied issues on attribute grammars when attribute grammars are regarded as tree transformations. Ganzinger and Giegerich [Gan83, GG84, Gie88] invented *descriptional composition*, which enables a single attribute grammar to be synthesized from two attribute grammars under the so-called *single-use* condition. Descriptional composition, which was originally developed for composing compiler components, has since then been applied in a wider scope. For example, Kühnemann [Küh98] and Correnson et al. [CDPR99] independently applied descriptional composition to the transformation of functional programs by utilizing the fact that attribute grammars are closely connected to functional programs.

The class of attribute grammars to which descriptional composition can be applied is still limited, however. There are significant instances of transformations where the composition method cannot be applied. For instance, consider the transformation from the postfix representation of numerical formulae, *e.g.,* $2, 1, 2, +, \times$ in Figure 1(a), to the infix representation,

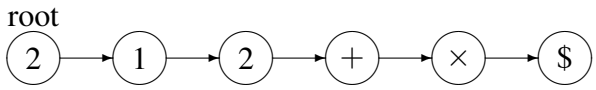

Figure 1(a): Postfix representation

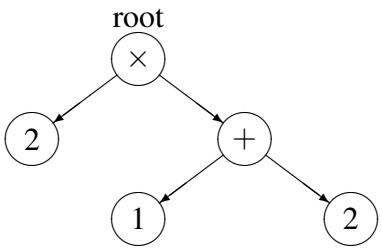

Figure 1(b): Infix representation

Figure 1: Two kinds of representations of numerical formulae

e.g., $2 \times (1+2)$ in Figure 1(b). The descriptional composition cannot deal with this transformation since it requires a pushdown stack device to store an unbounded number of subtrees of the output tree.

Figure 2 gives a definition of postfix-to-infix transformation as an attribute grammar. The symbols 1, 2, $+$, $\times$ and \$ correspond to terminal symbols *one*, *two*, *plus*, *multi* and *end*, respectively. For instance, the postfix representation in Figure 1(a) is expressed by $two(one(two(plus($ $multi(end)))))$ and the infix representation in Figure 1(b) by $multi(two, plus(one, two))$. The inherited attribute $\boldsymbol{s}$ associated with a nonterminal symbol T has stacks of trees as values. We write $[\,]$ for the empty stack, $t :: e$ for the stack obtained by pushing the tree t to the stack e, $hd(e)$ for the top-most element of the stack e and $tl(e)$ for the stack e whose top-most element is removed. The addition of these stack operations enhances the transformation power of the attribute grammar. Existing composition methods, however, cannot deal with attribute grammars that use a stack mechanism in particular when they appear as the first component of the composition.

We present a new method to compose attribute grammars that use a pushdown stack device. The method is formalized in the framework of attributed tree transducers [Fül81] (att), which is one of the formal computational models of attribute grammars. In this paper, we introduce *stack-attributed tree transducers* (satt) by extending atts with a stack device and propose a method to compose satts that extends descriptional composition. Satt's are more powerful than atts because of the pushdown storages. This relation is similar to that between pushdown automata and finite state automata: The class of languages accepted by the former, the context-free languages, is larger than the class of languages accepted by the latter, the regular languages [AU73].

$$
\begin{array}{lrcl}
S \to T: & S.a_0 & = & T.a_0 \\
 & T.\boldsymbol{s} & = & [\,] \\
T \to one(T_1): & T.a_0 & = & T_1.a_0 \\
 & T_1.\boldsymbol{s} & = & one :: T.\boldsymbol{s} \\
T \to two(T_1): & T.a_0 & = & T_1.a_0 \\
 & T_1.\boldsymbol{s} & = & two :: T.\boldsymbol{s} \\
T \to plus(T_1): & T.a_0 & = & T_1.a_0 \\
 & T_1.\boldsymbol{s} & = & (plus(hd(tl(T.\boldsymbol{s})), hd(T.\boldsymbol{s}))) \\
 & & & :: (tl(tl(T.\boldsymbol{s}))) \\
T \to multi(T_1): & T.a_0 & = & T_1.a_0 \\
 & T_1.\boldsymbol{s} & = & (multi(hd(tl(T.\boldsymbol{s})), hd(T.\boldsymbol{s}))) \\
 & & & :: (tl(tl(T.\boldsymbol{s}))) \\
T \to end: & T.a_0 & = & hd(T.\boldsymbol{s})
\end{array}
$$

Figure 2: An attribute grammar with a stack device for postfix-to-infix transformation

$$
\begin{aligned}
TDTT &\subsetneq ATT && (1)\\
ATT &\subsetneq MTT && (2)\\
MTT_{wsu} &\subsetneq ATT && (3)\\
ATT &\subsetneq SATT && (4)
\end{aligned}
\qquad
\begin{aligned}
TDTT \circ TDTT &= TDTT && (5)\\
ATT_{su} \circ ATT_{su} &= ATT_{su} && (6)\\
TDTT \circ ATT &= ATT && (7)\\
TDTT \circ MTT &= MTT && (8)\\
ATT \circ ATT_{su} &= ATT && (9)\\
MTT_{wsu} \circ MTT_{nc} &= MTT && (10)\\
ATT_{su} \circ SATT_{su} &= SATT_{su} && (11)\\
TDTT \circ SATT &= SATT && (12)\\
ATT \circ SATT_{su} &= SATT && (13)
\end{aligned}
$$

Figure 3: The inclusion relations of various classes of tree transformations and their compositions

We prove the correctness of our composition method by extending the composition method of atts. As in the case of atts, our composition method only works under certain restrictions. Roughly speaking an att is *single-use* if (the value of) each attribute is used at most once by the other attribute is used at most once by the other attributes, and a satt is single-use if, moreover, every element of (the value of) each attribute is used at most once by the other attribute is used at most once by the other attributes. In the case of atts it is known that the composition of an att M with an att M' works whenever M is single-use or M' has no inherited attributes. Moreover, if both atts are single-use, then so is the resulting att. In this paper we compose a satt M with an att M'. The main result of the paper is that our method works in the same cases as for atts: whenever M is single-use or M' has no inherited attributes; moreover, if both M and M' are single-use then so is the resulting satt.

Our composition method is proposed not only for theoretical interest. The method to compose stack-attributed tree transducers is useful in practice for transformations over structured documents such as XML [W3C]. Nakano and Nishimura [NN01] utilized descriptional composition to generate stream-to-stream document transformations from tree-to-tree document transformations, where a stream means just a string. Given a tree-to-tree document transformation T, a stream-to-stream document transformation can be obtained by the composition of three transformations *Unparse*, T and *Parse*: *Unparse* is an unparsing (tree-to-stream) transformation; *Parse* is a parsing (stream-to-tree) transformation. Defining each transformation by an attribute grammar, Nakano et al. applied descriptional composition to obtain a stream-to-stream transformation. In their framework, they faced the problem that parsing requires a stack mechanism, which cannot be dealt with by the existing descriptional composition method. They circumvented the problem by fixing the maximum depth of nesting of the input documents and by simulating the stack mechanism with a finite set of attributes. Our composition method provides a general solution of the above problem in a straightforward way. If we give a tree-to-tree transformation by a single-use att, we can get tree-to-stream transformation as an att by composing it with *Unparse*. If we give a tree-to-stream transformation by an att, we can get stream-to-stream transformation as a satt by our method because *Parse* can be represented by a single-use satt.

There are a number of investigations on tree transducer composition. Figure 3 shows the

inclusion relations of various classes of tree transformations and their compositions [FV98]. We write TDTT, ATT, MTT and SATT to represent the classes of tree transformations defined by top-down tree transducers [Rou70], attributed tree transducers [Fül81], macro tree transducers [Eng80, EV85] and stack-attributed tree transducers, respectively. The subscripts *su*, *wsu* and *nc* indicate the subsets of tree transducers restricted by the conditions *single-use* [Gie88], *weakly single-use* and *non-copying* [Küh98], respectively. For two classes T_1 and T_2 of tree transformations, $T_1 \circ T_2$ denotes the class $\{\tau \mid \tau(x) = \tau_1(\tau_2(x)), \tau_1 \in T_1, \tau_2 \in T_2\}$. The relations (1) and (7) are shown in [Fül81]; (2) and (8), in [EV85]; (3), in [Küh98]; (5), in [Rou70]; (6) and (9) in [GG84, Gie88]; (10), in [VK04]. The relations (4), (11), (12) and (13), involving SATT, are shown in this paper.

We say that the class T of tree transformations enjoys a *full closure property* if $T \circ T = T$. Equations (5) and (6) show that TDTT and ATT_{su} enjoy the full closure property. However, TDTT and ATT_{su} are not expressive enough.

We consider a weaker closure property than the full closure property. For two classes T_1 and T_2 of tree transformations, we call T_2 is *left closed* for T_1 if $T_1 \circ T_2 = T_2$. This paper shows that SATT, together with SATT_{su}, is one of the largest classes of tree transformations that enjoys left closure property. Equation (12) shows that SATT is left closed for TDTT as well as ATT and MTT seen in (7) and (8). Additionally, (11) shows that SATT_{su} is left closed for ATT_{su}.

Unfortunately, our result (13) does not show any closure porperty as well as (10) does not enjoy any closure property. However, it generalizes equation (9) and it is helpful to implement an XML transformation language intended for stream processing [Nak04], while (10) is helpful to the transformation of functional programs [VK04].

Outline. The paper is comprised of five sections, including this introduction. In Section 2, we define basic notions and notations. Section 3 introduces attributed tree transducers and stack-attributed tree transducers with simple examples and shows that stack-attributed tree transducers can be simulated by attributed tree transducers. Section 4 introduces the composition method for attributed tree transducers, presents the composition method for stack-attributed tree transducers and shows the correctness of the composition method. Finally Section 5 presents further work and concludes the paper.

2 Preliminaries

The empty set is denoted by $\emptyset$. We denote the set of non-negative integers including 0 by $\mathbb{N}$, the set of positive integers by $\mathbb{N}_+$ and the sets $\{1,\ldots,n\}$ by $[n]$ for $n \in \mathbb{N}$, in particular, $[0] = \emptyset$. The disjoint union of two sets P and Q is denoted by $P \uplus Q$ and the cartesian product of two sets P and Q is denoted by $P \times Q$, *i.e.,* $P \times Q = \{\langle p,q\rangle \mid p \in P, q \in Q\}$. We assume that the cartesian product is associative, *i.e.,* $(P \times Q) \times R$ and $P \times (Q \times R)$ are identified and are written $P \times Q \times R$. We denote the set of finite strings over a set P of symbols by P^*. The empty string is denoted by ε.

The designated symbol $\perp$ means an undefined value, which can be a symbol in any alphabet. A function f from a set P to a set Q is denoted by $f : P \to Q$, while $\mathit{dom}(f)$ and $\mathit{range}(f)$ denote the *domain* of f and the *range* of f, respectively, such that $\mathit{dom}(f) = \{x \in P \mid f(x) \text{ is defined}\}$ and $\mathit{range}(f) = \{f(x) \in Q \mid x \in \mathit{dom}(f)\}$. The composition of two functions f and g is defined by $f \circ g(x) = f(g(x))$ for all $x \in \mathit{dom}(g)$ with $g(x) \in \mathit{dom}(f)$. For sets of functions F and G, we

write $F \circ G$ for $\{f \circ g \mid f \in F, g \in G\}$.

A *reduction system* is a system $(A, \Rightarrow)$ where A is a set and $\Rightarrow$ is a binary relation over A. We write $a_1 \Rightarrow^n a_{n+1}$ if $a_i \Rightarrow a_{i+1} (i \in [n])$ for some $a_1, \ldots, a_{n+1} \in A$. In particular, $a \Rightarrow^0 a$. We also write $a \Rightarrow^* b$, $a \Rightarrow^+ b$ and $a \Rightarrow^? b$ when $a \Rightarrow^m b$ holds for some $m \in \mathbb{N}$, $m \in \mathbb{N}_+$ and $m \in \{0,1\}$, respectively. We say that $a \in A$ is *reducible* with respect to $\Rightarrow$ if there exists $b \in A$ such that $a \Rightarrow b$. Otherwise, $a \in A$ is *irreducible*. If $a \Rightarrow^* b$ and b is irreducible, we say that b is a *normal form* of a. If a has a unique normal form, we denote it by $nf(\Rightarrow, a)$. If every $a \in A$ has at most one element $b \in A$ such that $a \Rightarrow b$, we say that the reduction system is *deterministic*. In a deterministic reduction system $(A, \Rightarrow)$, every $a \in A$ has at most one normal form.

A *ranked set* Σ is a set in which every symbol is associated with a non-negative integer called its *rank*. A *ranked alphabet* is a finite ranked set. For every $n \in \mathbb{N}$, $\Sigma^{(n)}$ is the set of symbols of rank n. We denote the rank of a symbol σ by $rank(\sigma)$. We may write $\sigma^{(n)}$ to indicate that $\sigma \in \Sigma^{(n)}$. For a ranked alphabet Σ, the maximum rank of symbols in Σ is denoted by $maxr(\Sigma)$, *i.e.,* $maxr(\Sigma) = \max\{rank(\sigma) \mid \sigma \in \Sigma\}$. The designated symbol $\perp$ is of rank 0. Let Σ be a ranked alphabet and X be a set of variables disjoint with Σ. The set of Σ-*labeled trees indexed by* X, denoted by $T_\Sigma(X)$ (or T_Σ, if X is empty), is the smallest set T satisfying

- $X \subset T$ and
- $\sigma(t_1, \ldots, t_n) \in T$ for every $n \in \mathbb{N}$, $\sigma \in \Sigma^{(n)}$ and $t_1, \ldots, t_n \in T$,

We denote by $t[x := s]$ the *substitution* of all occurrences of the variable x in t by s. Let $t, s_1, \ldots, s_n, u_1, \ldots, u_n$ be trees in $T_\Sigma(X)$ such that every u_i for $i \in [n]$ is a subtree of t, provided that u_i is not a subtree of u_j for any $j \neq i$. The tree $t[u_1, \ldots, u_n := s_1, \ldots, s_n]$, or $t[u_i := s_i]_{i \in [n]}$ is obtained from t by simultaneously *replacing* all occurrences of the subtrees $u_1, \ldots, u_n$ by the trees $s_1, \ldots, s_n$, respectively. A $T_\Sigma(X)$-*context* $\mathcal{E}$ (or *context*, simply) is an element of $T_\Sigma(X \uplus \{\bullet\})$ where the symbol $\bullet$, called *hole*, occurs exactly once in $\mathcal{E}$. The tree $\mathcal{E}[t]$ with $t \in T_\Sigma(X)$ stands for $\mathcal{E}[\bullet := t] \in T_\Sigma(X)$.

The prefix-closed set of all *paths* of $t \in T_\Sigma(X)$, denoted by $path(t) (\subset \mathbb{N}_+^*)$, is defined by $path(\sigma(t_1, \ldots, t_n)) = \{\varepsilon\} \cup \{iw \mid i \in [n], w \in path(t_i)\}$ if $\sigma \in \Sigma^{(n)}$. Note that $path(\sigma^{(0)}) = \{\varepsilon\}$. We write $sub_t(w)$ for the subtree of a tree t at a path $w \in path(t)$. Every path $w \in path(t)$ refers to a corresponding label of t, denoted by $label_t(w)$, which is defined by $label_{\sigma(t_1,\ldots,t_n)}(\varepsilon) = \sigma$ and $label_{\sigma(t_1,\ldots,t_n)}(iw) = label_{t_i}(w)$ for every $i \in [n]$ and $w \in path(t_i)$. We use $\pi, \pi 1, \pi 2, \ldots$ for *path variables*. A substitution $[\pi, \pi 1, \pi 2, \ldots := w, w1, w2, \ldots]$ is simply denoted by $[\pi := w]$ for $w \in \mathbb{N}_+^*$.

3 Stack-Attributed Tree Transducers

In this section, we give the definition of attributed tree transducers (for short, att) and stack-attributed tree transducers (for short, satt). We also show that satts are simulated by atts when the set of input trees are restricted to a certain set. We follow the definition of atts in [Küh97, FV98] and give the definition of satts as an extension of atts.

3.1 Attributed Tree Transducers

Following [FV98], we start with a definition of the set of trees occurring in the right-hand sides of attribute rules in the specification of atts.

Definition 3.1 Let Δ be a ranked alphabet, *Syn* and *Inh* be countable sets of unary ranked symbols and $k \in \mathbb{N}$. The set $RHS(Syn, Inh, \Delta, k)$ of *right-hand sides* over *Syn*, *Inh* and Δ is the smallest subset *RHS* of $T_{\Delta \cup Syn \cup Inh}(\{\pi, \pi 1, \ldots, \pi k\})$ that satisfies the following conditions:

- For every $a \in Syn$ and $1 \leq i \leq k$, the term $a(\pi i)$ is in *RHS*.
- For every $b \in Inh$, the term $b(\pi)$ is in *RHS*.
- For every $\delta \in \Delta^{(l)}$ with $l \in \mathbb{N}$ and $\eta_1, \ldots, \eta_l \in RHS$, the term $\delta(\eta_1, \ldots, \eta_l)$ is in *RHS*.

□

Definition 3.2 An *attributed tree transducer* (att) is a septuple $M = (Syn, Inh, \Sigma, \Delta, in, \sharp, R)$ where

- *Syn* and *Inh* are countable sets of unary ranked symbols satisfying $Syn \cap Inh = \emptyset$, whose elements are called *synthesized attributes* and *inherited attributes*, respectively,
- Σ and Δ are ranked alphabets such that $(Syn \cup Inh) \cap (\Sigma \cup \Delta) = \emptyset$, called the *input alphabet* and the *output alphabet*, respectively,
- $in \in Syn$ is a designated attribute, called the *initial attribute*,
- $\sharp \notin \Sigma \cup Syn \cup Inh$ is a unary ranked symbol, called the *initial symbol*; we write Σ^+ for $\Sigma \uplus \{\sharp\}$,
- R is a set of *attribute rules* such that $R = \bigcup_{\sigma \in \Sigma^+} R^\sigma$ with finite sets R^σ of σ-*rules* satisfying the following conditions:
  - for every $a \in Syn$, the set R^σ contains exactly one attribute rule of the form $a(\pi) \xrightarrow{\sigma} \eta_\sigma$,
  - for every $b \in Inh$ and $i \in [rank(\sigma)]$, the set R^σ contains exactly one attribute rule of the form $b(\pi i) \xrightarrow{\sigma} \eta_\sigma$,

  where η_σ with $\sigma \in \Sigma$ is in $RHS(Syn, Inh, \Delta, rank(\sigma))$ and $\eta_\sharp$ is in $RHS(Syn, \emptyset, \Delta, 1)$.

Moreover, an att whose set $Syn \cup Inh$ of attributes is finite is called *finitary att*. □

Our definition is slightly different from [FV98] in how an att is specified. We use no environment describing values of inherited attributes of the root of the input tree. Instead, we define the values of those inherited attributes in the attribute rules for the initial symbol, following [Gie88, Küh97, Küh98]. We assume the input and output alphabets implicitly contain the designated symbol $\perp$ whose rank is 0. For that symbol, all atts implicitly have the set $R^\perp$ of attribute rules $a(\pi) \xrightarrow{\perp} \perp$ for every synthesized attribute a in the att. Note that the set of attributes is allowed to be countably infinite unlike usual definition. It will be helpful to simulate a stack-attributed tree transducer by an att as mentioned later.

The readers who are familiar with attribute grammars may understand the specification of atts by considering $a(\pi)$ and $b(\pi i)$ as attribute occurrences $P.a$ and $P_i.b$ in attribute rules for a production rule $P \to \sigma(P_1, \ldots, P_k)$, in the classical representation of attribute grammars where P and P_i are nonterminal symbols and σ is a terminal symbol representing the name of the

production. For example, an attribute rule $a(\pi) \xrightarrow{\sigma} \delta(b(\pi 1))$ with $rank(\sigma) = 2$ corresponds to an attribute rule $P.a = \delta(P_1.b)$ for a production rule $P \to \sigma(P_1, P_2)$. Below we give an example of an att and its attribute grammar counterpart.

Example 3.3 We give a (finitary) att M_{itop} for an infix-to-postfix conversion of simple numerical formulae, *e.g.,* $2 \times (1 + 2)$ to $2, 1, 2, +, \times$ (see Figure 1). For the sake of simplicity, we assume that they are built up from the integers 1 and 2 by using the two binary operators $+$ and $\times$. The infix representation is specified by a binary tree in which internal nodes and leaves are numerical operators and numbers, respectively. The postfix representation is specified by a monadic tree whose nodes are numerical operators or numbers. The att $M_{itop} = (Syn, Inh, \Sigma, \Delta, a_0, \sharp, R)$ is given as follows:

- $Syn = \{a_0\}$, $Inh = \{a_1\}$.
- $\Sigma = \{one^{(0)}, two^{(0)}, plus^{(2)}, multi^{(2)}\}$.
- $\Delta = \{one^{(1)}, two^{(1)}, plus^{(1)}, multi^{(1)}, end^{(0)}\}$.
- R is the following set of attribute rules:

$$a_0(\pi) \xrightarrow{\sharp} a_0(\pi 1) \tag{14}$$
$$a_1(\pi 1) \xrightarrow{\sharp} end \tag{15}$$
$$a_0(\pi) \xrightarrow{one} one(a_1(\pi)) \tag{16}$$
$$a_0(\pi) \xrightarrow{two} two(a_1(\pi)) \tag{17}$$
$$a_0(\pi) \xrightarrow{plus} a_0(\pi 1) \tag{18}$$
$$a_1(\pi 1) \xrightarrow{plus} a_0(\pi 2) \tag{19}$$
$$a_1(\pi 2) \xrightarrow{plus} plus(a_1(\pi)) \tag{20}$$
$$a_0(\pi) \xrightarrow{multi} a_0(\pi 1) \tag{21}$$
$$a_1(\pi 1) \xrightarrow{multi} a_0(\pi 2) \tag{22}$$
$$a_1(\pi 2) \xrightarrow{multi} multi(a_1(\pi)) \tag{23}$$

Figure 4 shows the attribute grammar counterpart of the att M_{itop}. The rules (14) and (15), namely $R^\sharp$, correspond to the set of two attribute rules for the production rule $S \to T$ in the attribute grammar. Similarly, R^{one}, R^{two}, R^{plus} and R^{multi} correspond to the attribute rules for the production rules $T \to one$, $T \to two$, $T \to plus(T_1, T_2)$ and $T \to multi(T_1, T_2)$, respectively. □

We define the semantics of atts. The computation of an att M for an input tree t is defined by a reduction system, whose definition is given below.

Definition 3.4 Let $M = (Syn, Inh, \Sigma, \Delta, in, \sharp, R)$ be an att, $t \in T_{\Sigma^+}$. The *derivation relation induced by M on t* is the smallest binary relation $\Rightarrow_{M,t}$ over $T_\Delta(\{a(w) \mid a \in Syn \cup Inh, w \in path(t)\})$ such that:

- $a(w) \Rightarrow_{M,t} \eta[\pi := w]$ where $a \in Syn$, $a(\pi) \xrightarrow{\sigma} \eta \in R$ and $\sigma = label_t(w)$.

- $b(wi) \Rightarrow_{M,t} \eta[\pi := w]$ where $b \in Inh$, $b(\pi i) \xrightarrow{\sigma} \eta \in R$ and $\sigma = label_t(w)$.
- $\delta(\eta_1,\ldots,\eta_i,\ldots,\eta_n) \Rightarrow_{M,t} \delta(\eta_1,\ldots,\eta_i',\ldots,\eta_n)$ where $\delta \in \Delta^{(n)}$, $\eta_i \Rightarrow_{M,t} \eta_i'$ and $\eta_k \in T_\Delta$ for every $k \in [i-1]$.

We may omit the subscripts M and t when they are clear from the context. Note that there is no clause such as "$\mathcal{E}[\eta_1] \Rightarrow \mathcal{E}[\eta_2]$ if $\eta_1 \Rightarrow \eta_2$". That implies that, for every tree s, there is at most one tree s' such that $s \Rightarrow s'$, *i.e.,* this reduction system is deterministic. Therefore a normal form, if it exists, must be unique.

The *attribute value* of $a(w)$ for an att M and an input tree $t \in T_\Sigma$ is defined by $nf(\Rightarrow_{M,\sharp(t)}, a(w))$, if it exists, where $a \in Syn \cup Inh$, $w \in path(\sharp(t))$ and $\langle a,w\rangle \notin Inh \times \{\varepsilon\}$. Note that $nf(\Rightarrow_{M,\sharp(t)}, a(w))$ is a tree over the output alphabet $\Delta(\cup\{\bot\})$ of M. This is because any term having subterms of the form $a'(w')$ with $a' \in Inh \cup Syn$ and a path w' is always reducible unless $a' \in Inh$ and $w' = \varepsilon$. If $\langle a,w\rangle \notin Inh \times \{\varepsilon\}$, $b(\varepsilon)$ with $b \in Inh$ does not occur in the derivation induced by M on $\sharp(t)$ with $t \in T_\Sigma$ because of the specification of $\eta_\sharp$ in Definition 3.2. □

We cannot always compute all attribute values for a given att M and an input tree t. For instance, an attribute value $a(w)$ cannot be computed in the case that we have a derivation $a(w) \Rightarrow^+_{M,\sharp(t)} \mathcal{E}[a(w)]$ for some context $\mathcal{E}$. In order to avoid the problem, we require the usual well-definedness (or noncircularity) property for an att.

Definition 3.5 Let $M = (Syn, Inh, \Sigma, \Delta, in, \sharp, R)$ be an att. The *semantics* of M is the function $[\![M]\!] : T_\Sigma \to T_\Delta$ such that for all $t \in T_\Sigma$, $[\![M]\!](t) = nf(\Rightarrow_{M,\sharp(t)}, in(\varepsilon))$ if it exists. The att M is *well-defined* if $nf(\Rightarrow_{M,\sharp(t)}, a(w))$ exists for every $t \in T_\Sigma$, $a \in Syn \cup Inh$ and $w \in path(\sharp(t))$ with $\langle a,w\rangle \notin Inh \times \{\varepsilon\}$. □

Example 3.6 Let M be the att given in Example 3.3. The computation of M for the input tree $t = multi(two, plus(one, two))$ is shown below. We obtain that

$$[\![M]\!](multi(two, plus(one, two))) = two(one(two(plus(multi(end)))))$$

$S \to T$:	$S.a_0 = T.a_0$	$T \to plus(T_1,T_2)$:	$T.a_0 = T_1.a_0$
	$T.a_1 = end$		$T_1.a_1 = T_2.a_0$
$T \to one$:	$T.a_0 = one(T.a_1)$		$T_2.a_1 = plus(T.a_1)$
$T \to two$:	$T.a_0 = two(T.a_1)$	$T \to multi(T_1,T_2)$:	$T.a_0 = T_1.a_0$
			$T_1.a_1 = T_2.a_0$
			$T_2.a_1 = multi(T.a_1)$

Figure 4: The attribute grammar counterpart of Example 3.3

by the following reduction steps:

$$
\begin{aligned}
a_0(\varepsilon) &\Rightarrow^{\sharp,\varepsilon} a_0(1) && (24)\\
&\Rightarrow^{multi,1} a_0(11) && (25)\\
&\Rightarrow^{two,11} two(a_1(11)) && (26)\\
&\Rightarrow^{multi,1} two(a_0(12)) && (27)\\
&\Rightarrow^{plus,12} two(a_0(121)) && (28)\\
&\Rightarrow^{one,121} two(one(a_1(121))) && (29)\\
&\Rightarrow^{plus,12} two(one(a_0(122))) && (30)\\
&\Rightarrow^{two,122} two(one(two(a_1(122)))) && (31)\\
&\Rightarrow^{plus,12} two(one(two(plus(a_1(12))))) && (32)\\
&\Rightarrow^{multi,1} two(one(two(plus(multi(a_1(1)))))) && (33)\\
&\Rightarrow^{\sharp,\varepsilon} two(one(two(plus(multi(end))))) && (34)
\end{aligned}
$$

where the superscripts of $\Rightarrow^{\sigma,w}$ indicate that the derivation relation is based on a σ-rule at the path w. For example, the reduction step $\Rightarrow^{plus,12}$ at (32) is derived by replacing π with the path 12 in $\sharp(t)$ in the *plus*-rule at (20). □

An att M is called a *top-down tree transducer* (tdtt) if there is no inherited attribute in M. We define the sets of tree transformations $ATT = \{[\![M]\!] \mid M \text{ is a well-defined finitary att}\}$ and $TDTT = \{[\![M]\!] \mid M \text{ is a finitary tdtt}\}$. It is well known that $TDTT \subsetneq ATT$[Fül81].

3.2 Stack-Attributed Tree Transducers

The *stack-attributed tree transducers* (satts) extend atts with a new type of attributes, called *stack attributes*. Thus, there are two types of attributes in a satt, output attributes and stack attributes. Output attributes have trees over the output alphabet as values, similar to attributes in atts. Stack attributes have pushdown stacks of trees over the output alphabet as values. The following definition introduces the notion of stack system specifying sets of output expressions and stack expressions which are used for the definition of a satt.

Definition 3.7 A *stack system over* Δ is a triple $\mathcal{S} = (X_o, X_s, \Delta)$ where X_o and X_s are disjoint sets of trees and Δ is a ranked alphabet. For a stack system $\mathcal{S}$, we define the set $EXP_o(\mathcal{S})$ of *output expressions* and the set $EXP_s(\mathcal{S})$ of *stack expressions* as the smallest set EXP_o and EXP_s of trees such that:

- $EXP_o \supset X_o$ and $EXP_s \supset X_s$.
- $\delta(e_1,\ldots,e_n) \in EXP_o$ if $e_i \in EXP_o (i \in [n])$ and $\delta \in \Delta^{(n)}$ with $n \in \mathbb{N}$.
- $Cons^{(2)}(e_1,e_2) \in EXP_s$ if $e_1 \in EXP_o$ and $e_2 \in EXP_s$.
- $Empty^{(0)} \in EXP_s$.
- $Head^{(1)}(e) \in EXP_o$ if $e \in EXP_s$.
- $Tail^{(1)}(e) \in EXP_s$ if $e \in EXP_s$. □

The four ranked symbols *Cons*, *Empty*, *Head* and *Tail* are called *stack operators*: $Cons(e_1, e_2)$ denotes the stack obtained by adding a value e_1 to the top of the stack e_2; *Empty* denotes the empty stack; $Head(e)$ denotes the value at the head of the stack e; $Tail(e)$ denotes the stack obtained by removing its head from the stack e.

We specify the set of right-hand sides of attribute rules in satts before giving the definition of satts. The right-hand sides are defined by output and stack expressions of a stack system over an output alphabet. We use bold-faced symbols, *e.g.,* $\boldsymbol{a}, \boldsymbol{a_1}, \ldots$, to represent stack attributes for the purpose of distinguishing them from output attributes.

Definition 3.8 Let Δ be a ranked alphabet, let *Syn*, *Inh*, *StSyn* and *StInh* be disjoint unary ranked alphabets, let $k \in \mathbb{N}$ and let $\mathcal{S}_{\Delta,k}$ be a stack system (X_o, X_s, Δ) with

$$X_o = \{a(\pi i) \mid a \in Syn, i \in [k]\} \cup \{b(\pi) \mid b \in Inh\} \tag{35}$$
$$X_s = \{\boldsymbol{a}(\pi i) \mid \boldsymbol{a} \in StSyn, i \in [k]\} \cup \{\boldsymbol{b}(\pi) \mid \boldsymbol{b} \in StInh\}. \tag{36}$$

- The set $RHS_o(Syn, Inh, StSyn, StInh, \Delta, k)$ of *output right-hand sides* over *Syn*, *Inh*, *StSyn*, *StInh* and Δ is $EXP_o(\mathcal{S}_{\Delta,k})$.
- The set $RHS_s(Syn, Inh, StSyn, StInh, \Delta, k)$ of *stack right-hand sides* over *Syn*, *Inh*, *StSyn*, *StInh* and Δ is $EXP_s(\mathcal{S}_{\Delta,k})$.

□

Definition 3.9 A *stack-attributed tree transducer* (satt) is a nonuple

$$M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, in, \sharp, R)$$

where

- *Syn*, *Inh*, Σ, Δ, *in* and $\sharp$ are the same as in Definition 3.2 and Σ^+ is defined by $\Sigma \uplus \{\sharp\}$. We call *Syn* and *Inh* *output synthesized attributes* and *output inherited attributes*, respectively.
- *StSyn* and *StInh* are unary ranked alphabets with $StSyn \cap StInh = \emptyset$, whose elements are called *stack synthesized attributes* and *stack inherited attributes*, respectively. They are also disjoint with *Syn*, *Inh*, Σ and Δ.
- R is a set of *attribute rules* such that $R = \bigcup_{\sigma \in \Sigma^+} R^\sigma$ with finite sets R^σ of σ-*rules* satisfying the following conditions. For every $\sigma \in \Sigma^+$ whose rank is $k \in \mathbb{N}$,
  - for every $a \in Syn$, the set R^σ contains exactly one attribute rule of the form $a(\pi) \xrightarrow{\sigma} \eta_\sigma$,
  - for every $b \in Inh$ and $i \in [k]$, the set R^σ contains exactly one attribute rule of the form $b(\pi i) \xrightarrow{\sigma} \eta_\sigma$,
  - for every $\boldsymbol{a} \in StSyn$, the set R^σ contains exactly one attribute rule of the form $\boldsymbol{a}(\pi) \xrightarrow{\sigma} \zeta_\sigma$,
  - for every $\boldsymbol{b} \in StInh$ and $i \in [k]$, the set R^σ contains exactly one attribute rule of the form $\boldsymbol{b}(\pi i) \xrightarrow{\sigma} \zeta_\sigma$,

where η_σ and ζ_σ with $\sigma \in \Sigma$ are in $RHS_o(Syn, Inh, StSyn, StInh, \Delta, k)$ and $RHS_s(Syn, Inh, StSyn, StInh, \Delta, k)$, respectively, and $\eta_\sharp$ and $\zeta_\sharp$ are in $RHS_o(Syn, \emptyset, StSyn, \emptyset, \Delta, 1)$ and $RHS_s(Syn, \emptyset, StSyn, \emptyset, \Delta, 1)$, respectively.

A satt whose set $Syn \cup Inh$ of attributes is finite is called *finitary satt*.

Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, in, \sharp, R)$ be a satt. We define $RHS_o(M)$ and $RHS_s(M)$ as follows:

$$RHS_o(M) = RHS_o(Syn, Inh, StSyn, StInh, \Delta, maxr(\Sigma)) \quad (37)$$

$$RHS_s(M) = RHS_s(Syn, Inh, StSyn, StInh, \Delta, maxr(\Sigma)) \quad (38)$$

□

As for atts, we assume that the input and output alphabets of satts implicitly contain the designate symbol $\perp$ and that the set $R^\perp$ of attribute rules $a(\pi) \xrightarrow{\perp} \perp$ for every synthesized output attribute a and attribute rules $\boldsymbol{a}(\pi) \xrightarrow{\perp} Empty$ for every synthesized stack attribute $\boldsymbol{a}$ is implicitly.

The following example M_{ptoi} represents the inverse transformation of the att M_{itop} defined in Example 3.3. The satt M_{ptoi} converts postfix representations into infix representations. This transformation requires a stack device.

Example 3.10 The postfix-to-infix conversion in Section 1 can be expressed by the (finitary) satt M_{ptoi}, which corresponds to the attribute grammar in Figure 2. The satt $M_{ptoi} = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, a_0, \sharp, R)$ is given as follows:

- $Syn = \{a_0\}$, $Inh = \emptyset$.
- $StSyn = \emptyset$, $StInh = \{\boldsymbol{s}\}$.
- $\Sigma = \{one^{(1)}, two^{(1)}, plus^{(1)}, multi^{(1)}, end^{(0)}\}$.
- $\Delta = \{one^{(0)}, two^{(0)}, plus^{(2)}, multi^{(2)}\}$.
- R is the following set of attribute rules:

$$a_0(\pi) \xrightarrow{\sharp} a_0(\pi 1) \quad (39)$$

$$\boldsymbol{s}(\pi 1) \xrightarrow{\sharp} Empty \quad (40)$$

$$a_0(\pi) \xrightarrow{one} a_0(\pi 1) \quad (41)$$

$$\boldsymbol{s}(\pi 1) \xrightarrow{one} Cons(one, \boldsymbol{s}(\pi)) \quad (42)$$

$$a_0(\pi) \xrightarrow{two} a_0(\pi 1) \quad (43)$$

$$\boldsymbol{s}(\pi 1) \xrightarrow{two} Cons(two, \boldsymbol{s}(\pi)) \quad (44)$$

$$a_0(\pi) \xrightarrow{plus} a_0(\pi 1) \quad (45)$$

$$\boldsymbol{s}(\pi 1) \xrightarrow{plus} Cons(plus(Head(Tail(\boldsymbol{s}(\pi))), Head(\boldsymbol{s}(\pi))), Tail(Tail(\boldsymbol{s}(\pi)))) \quad (46)$$

$$a_0(\pi) \xrightarrow{multi} a_0(\pi 1) \quad (47)$$

$$\boldsymbol{s}(\pi 1) \xrightarrow{multi} Cons(multi(Head(Tail(\boldsymbol{s}(\pi))), Head(\boldsymbol{s}(\pi))), Tail(Tail(\boldsymbol{s}(\pi)))) \quad (48)$$

$$a_0(\pi) \xrightarrow{end} Head(\boldsymbol{s}(\pi)) \quad (49)$$

□

We define the semantics of satts by a reduction system defined in much of the same way as that of atts in Definition 3.4.

Definition 3.11 Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, in, \sharp, R)$ be a satt, $t \in T_{\Sigma^+}$ and $\mathcal{S} = (X_o, X_s, \Delta)$ the stack system with

$$\begin{aligned} X_o &= \{a(w) \mid a \in Syn \cup Inh,\ w \in path(t)\} \\ X_s &= \{\boldsymbol{a}(w) \mid \boldsymbol{a} \in StSyn \cup StInh,\ w \in path(t)\}. \end{aligned}$$

The *derivation relation induced by M on t* is the smallest binary relation $\Rightarrow_{M,t} \subset (EXP_o(\mathcal{S}) \times EXP_o(\mathcal{S})) \cup (EXP_s(\mathcal{S}) \times EXP_s(\mathcal{S}))$ such that:

- $a(w)$ and $b(wi)$ with $a \in Syn$, $b \in Inh$, $w \in path(t)$ and $i \in \mathbb{N}_+$ are related by $\Rightarrow_{M,t}$ in the same way as in Definition 3.4.
- $\boldsymbol{a}(w) \Rightarrow_{M,t} \zeta[\pi := w]$ where $\boldsymbol{a} \in StSyn$, $\boldsymbol{a}(\pi) \xrightarrow{\sigma} \zeta \in R$ and $\sigma = label_t(w)$.
- $\boldsymbol{b}(wi) \Rightarrow_{M,t} \zeta[\pi := w]$ where $\boldsymbol{b} \in StInh$, $\boldsymbol{b}(\pi i) \xrightarrow{\sigma} \zeta \in R$ and $\sigma = label_t(w)$.
- $Head(\zeta) \Rightarrow_{M,t} \begin{cases} \eta' & (\text{if } \zeta = Cons(\eta', \zeta')) \\ \bot & (\text{if } \zeta = Empty) \\ Head(\zeta') & (\text{otherwise, if } \zeta \Rightarrow_{M,t} \zeta') \end{cases}$

  where $\eta' \in EXP_o(\mathcal{S})$ and $\zeta, \zeta' \in EXP_s(\mathcal{S})$.
- $Tail(\zeta) \Rightarrow_{M,t} \begin{cases} \zeta' & (\text{if } \zeta = Cons(\eta', \zeta')) \\ Empty & (\text{if } \zeta = Empty) \\ Tail(\zeta') & (\text{otherwise, if } \zeta \Rightarrow_{M,t} \zeta') \end{cases}$

  where $\eta' \in EXP_o(\mathcal{S})$ and $\zeta, \zeta' \in EXP_s(\mathcal{S})$.
- $Cons(\eta, \zeta) \Rightarrow_{M,t} \begin{cases} Cons(\eta', \zeta) & (\text{if } \eta \Rightarrow_{M,t} \eta') \\ Cons(\eta, \zeta') & (\text{if } \eta \in T_\Delta \text{ and } \zeta \Rightarrow_{M,t} \zeta') \end{cases}$

  where $\eta, \eta' \in EXP_o(\mathcal{S})$ and $\zeta, \zeta' \in EXP_s(\mathcal{S})$.
- $\delta(\eta_1, \ldots, \eta_i, \ldots, \eta_n) \Rightarrow_{M,t} \delta(\eta_1, \ldots, \eta_i', \ldots, \eta_n)$ where $\delta \in \Delta^{(n)}$, $\eta_i \Rightarrow_{M,t} \eta_i'$ and $\eta_k \in T_\Delta$ for every $k \in [i-1]$.

Similar to reduction systems for atts, uniqueness of the normal form is guaranteed since for every expression e there is at most one expression e' such that $e \Rightarrow_{M,t} e'$. We may omit the subscripts M and t when they are clear from the context. The *attribute value* of $a(w)$ for a satt M and an input tree t is defined by $nf(\Rightarrow_{M,\sharp(t)}, a(w))$, if it exists, where $a \in Inh \cup Syn \cup StSyn \cup StInh$ and $w \in path(\sharp(t))$ with $\langle a, w\rangle \notin (Inh \cup StInh) \times \{\varepsilon\}$. □

The following statement guarantees that all attribute values of output attributes range over output trees.

Proposition 3.12 *Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, in, \sharp, R)$ be a satt, $t \in T_\Sigma$, $a \in Syn \cup Inh$, and $w \in path(\sharp(t))$. If $nf(\Rightarrow_{M,\sharp(t)}, a(w))$ exists, then it is in T_Δ.* □

Proof. Let EXP_o and EXP_s be as given by $EXP_o(\mathcal{S})$ and $EXP_s(\mathcal{S})$ in Definition 3.11.

Suppose that there exists $t' = nf(\Rightarrow_{M,\sharp(t)}, a(w))$. By the definition of $\Rightarrow_{M,\sharp(t)}$, we have $t' \in EXP_o$. We prove the proposition by analyzing the structure of t'. First, we show that the pattern $a(w)$ with $a \in Syn \cup Inh \cup StSyn \cup StInh$ does not occur in t'. Second, we show that the patterns $Cons(\eta,\zeta)$ and $Empty$ do not occur in t'. Finally, we show that the patterns $Head(\zeta)$ and $Tail(\zeta)$ do not occur in t'. These facts imply that $t' \in T_\Delta$.

The pattern $a(w)$ with $a \in Syn \cup Inh \cup StSyn \cup StInh$ does not occur in t' since all terms having occurrences of $a(w)$ are reducible by the reduction rules based on the attribute rules.

It can be shown that the pattern $Cons(\eta,\zeta)$ does not occur in t' by contradiction. Assume that $t' = \mathcal{E}[Cons(\eta,\zeta)]$, provided that there is no occurrence of $Cons$ in $\mathcal{E}$. Since we have $t' \in EXP_o$ and $Cons(\eta,\zeta) \in EXP_s$, there exists a context $\mathcal{E}'$ such that either $\mathcal{E} = \mathcal{E}'[Head(\bullet)]$ or $\mathcal{E} = \mathcal{E}'[Tail(\bullet)]$. This contradicts with the fact that t' is irreducible. Thus the pattern $Cons(\eta,\zeta)$ does not occur in t'. Similarly, we can show that $Empty$ does not occur in t'.

It can be shown that the patterns $Head(\zeta)$ and $Tail(\zeta)$ do not occur in t' as follows. Since every stack expression has the form of $Cons(e)$, $Empty$ or $\boldsymbol{a}(w)$ with $\boldsymbol{a} \in StSyn \cup StInh$, we can assume that there is no occurrence of stack expressions in t' using the facts we have shown above. It implies that the patterns $Head$ and $Tail$ do not occur in t' because their arguments must be stack expressions. □

The semantics of satts is defined by the value of the initial attribute at the root in the same way as that of atts.

Definition 3.13 Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, in, \sharp, R)$ be a satt. The *semantics* of a satt M is the function $[\![M]\!] : T_\Sigma \to T_\Delta$ such that for all $t \in T_\Sigma$, $[\![M]\!](t) = nf(\Rightarrow_{M,\sharp(t)}, in(\varepsilon))$ if it exists. The satt M is *well-defined* if $nf(\Rightarrow_{M,\sharp(t)}, a(w))$ exists for every $t \in T_\Sigma$, $a \in Syn \cup Inh \cup StSyn \cup StInh$ and $w \in path(\sharp(t))$ with $\langle a, w\rangle \notin (Inh \cup StInh) \times \{\varepsilon\}$. □

Example 3.14 Let M be the satt given in Example 3.10. Figure 5 shows the reduction steps for deriving

$$[\![M]\!](two(one(two(plus(multi(end)))))) = multi(two, plus(one, two))$$

where $t = two(one(two(plus(multi(end)))))$ is the input tree and the superscripts of $\Rightarrow^{\sigma,w}$ indicate that the derivation relation is based on a σ-rule at the path w. The superscripts HC and TC indicate the applied rule in Definition 3.11, *i.e., HC* for $Head(Cons(\eta,\zeta))$ and TC for $Tail(Cons(\eta,\zeta))$. □

We write $SATT$ for the set of tree transformations $\{[\![M]\!] \mid M \text{ is a well-defined finitary satt}\}$. We can prove that $ATT \subsetneq SATT$ by showing the existence of a transformation that is not in ATT but in $SATT$, called *a parenthesis balance checker*.

Theorem 3.15 $ATT \subsetneq SATT$.

Proof. Since $ATT \subset SATT$ by the definitions of finitary att and finitary satt, it suffices to show that this inclusion is proper, *i.e.,* there is a satt M which cannot be simulated by any finitary att. Consider the transformation τ_{pbc} from T_Σ to T_Δ with $\Sigma = \{parenL^{(1)}, parenR^{(1)}, end^{(0)}\}$ and $\Delta = \{\top^{(0)}(,\bot^{(0)})\}$ such that τ_{pbc} returns $\top$ iff every right parenthesis $parenR$ has its left counterpart $parenL$ and vice versa in the input. This problem is a well-known example which

$$
\begin{aligned}
a_0(\varepsilon) &\Rightarrow^{\sharp,\varepsilon} && a_0(1)\\
&\Rightarrow^{two,1} && a_0(11)\\
&\Rightarrow^{one,11} && a_0(111)\\
&\Rightarrow^{two,111} && a_0(1111)\\
&\Rightarrow^{plus,1111} && a_0(11111)\\
&\Rightarrow^{multi,11111} && a_0(111111)\\
&\Rightarrow^{end,111111} && \mathit{Head}(\boldsymbol{s}(111111))\\
&\Rightarrow^{multi,11111} && \mathit{Head}(\mathit{Cons}(\mathit{multi}(\mathit{Head}(\mathit{Tail}(\boldsymbol{s}(11111))),\mathit{Head}(\boldsymbol{s}(11111))),\\
& && \qquad \mathit{Tail}(\mathit{Tail}(\boldsymbol{s}(11111)))))\\
&\Rightarrow^{HC} && \mathit{multi}(\mathit{Head}(\mathit{Tail}(\boldsymbol{s}(11111))),\mathit{Head}(\boldsymbol{s}(11111)))\\
&\Rightarrow^{plus,1111} && \mathit{multi}(\mathit{Head}(\mathit{Tail}(\mathit{Cons}(\mathit{plus}(\mathit{Head}(\mathit{Tail}(\boldsymbol{s}(1111)),\mathit{Head}(\boldsymbol{s}(1111))),\\
& && \qquad \mathit{Tail}(\mathit{Tail}(\boldsymbol{s}(1111))))))),\mathit{Head}(\boldsymbol{s}(11111)))\\
&\Rightarrow^{TC} && \mathit{multi}(\mathit{Head}(\mathit{Tail}(\mathit{Tail}(\boldsymbol{s}(1111)))),\mathit{Head}(\boldsymbol{s}(11111)))\\
&\Rightarrow^{two,111} && \mathit{multi}(\mathit{Head}(\mathit{Tail}(\mathit{Tail}(\mathit{Cons}(\mathit{two},\boldsymbol{s}(111))))),\mathit{Head}(\boldsymbol{s}(11111)))\\
&\Rightarrow^{TC} && \mathit{multi}(\mathit{Head}(\mathit{Tail}(\boldsymbol{s}(111))),\mathit{Head}(\boldsymbol{s}(11111)))\\
&\Rightarrow^{one,11} && \mathit{multi}(\mathit{Head}(\mathit{Tail}(\mathit{Cons}(\mathit{one},\boldsymbol{s}(11)))),\mathit{Head}(\boldsymbol{s}(11111)))\\
&\Rightarrow^{TC} && \mathit{multi}(\mathit{Head}(\boldsymbol{s}(11)),\mathit{Head}(\boldsymbol{s}(11111)))\\
&\Rightarrow^{two,1} && \mathit{multi}(\mathit{Head}(\mathit{Cons}(\mathit{two},\boldsymbol{s}(1))),\mathit{Head}(\boldsymbol{s}(11111)))\\
&\Rightarrow^{HC} && \mathit{multi}(\mathit{two},\mathit{Head}(\boldsymbol{s}(11111)))\\
&\Rightarrow^{plus,1111} && \mathit{multi}(\mathit{two},\mathit{Head}(\mathit{Cons}(\mathit{plus}(\mathit{Head}(\mathit{Tail}(\boldsymbol{s}(1111))),\mathit{Head}(\boldsymbol{s}(1111))),\\
& && \qquad \mathit{Tail}(\mathit{Tail}(\boldsymbol{s}(1111))))))\\
&\Rightarrow^{HC} && \mathit{multi}(\mathit{two},\mathit{plus}(\mathit{Head}(\mathit{Tail}(\boldsymbol{s}(1111))),\mathit{Head}(\boldsymbol{s}(1111))))\\
&\Rightarrow^{two,111} && \mathit{multi}(\mathit{two},\mathit{plus}(\mathit{Head}(\mathit{Tail}(\mathit{Cons}(\mathit{two},\boldsymbol{s}(111))))),\mathit{Head}(\boldsymbol{s}(1111))))\\
&\Rightarrow^{TC} && \mathit{multi}(\mathit{two},\mathit{plus}(\mathit{Head}(\boldsymbol{s}(111)),\mathit{Head}(\boldsymbol{s}(1111))))\\
&\Rightarrow^{one,11} && \mathit{multi}(\mathit{two},\mathit{plus}(\mathit{Head}(\mathit{Cons}(\mathit{one},\boldsymbol{s}(11))),\mathit{Head}(\boldsymbol{s}(1111))))\\
&\Rightarrow^{HC} && \mathit{multi}(\mathit{two},\mathit{plus}(\mathit{one},\mathit{Head}(\boldsymbol{s}(1111))))\\
&\Rightarrow^{two,111} && \mathit{multi}(\mathit{two},\mathit{plus}(\mathit{one},\mathit{Head}(\mathit{Cons}(\mathit{two},\boldsymbol{s}(111))))))\\
&\Rightarrow^{HC} && \mathit{multi}(\mathit{two},\mathit{plus}(\mathit{one},\mathit{two}))
\end{aligned}
$$

Figure 5: The reduction steps by the derivation relation $\Rightarrow_{M,t}$

cannot be solved by any finite state automaton. In this proof, we first illustrate that all atts from T_Σ to T_Δ have an equivalent finite state automaton to show that there is no att M such that $[\![M]\!] = \tau_{pbc}$. Finally, we illustrate that there is a satt M_{pbc} such that $[\![M_{pbc}]\!] = \tau_{pbc}$.

For any att from T_Σ to T_Δ, we can construct an equivalent finite state automaton that accepts $\{t \in T_\Sigma \mid [\![M]\!](t) = \top\}$. The construction is similar to the *finite state transition machine construction* introduced in [NN01], where a more detailed account for the construction is given. Let $M = (Syn, Inh, \Sigma, \Delta, a_0, \sharp, R)$ be a well-defined att where $Syn = \{a_0, a_1, \ldots, a_m\}$ and $Inh = \{b_1, \ldots, b_n\}$. Note that every symbol in Σ (and Δ) has rank at most 1 (and 0, respectively). Therefore the right-hand side of every rule in R is an element of $\Delta \cup \{a(\pi 1) \mid a \in Syn\} \cup \{b(\pi) \mid b \in Inh\}$. We define a function $rhs_\sigma : Syn \cup Inh \to \Delta \cup Syn \cup Inh$ with $\sigma \in \Sigma$ by removing any path information from the rules in R^σ. For instance, $rhs_\sigma(a_0) = \top$, $rhs_\sigma(a_1) = b_1$ and $rhs_\sigma(b_1) = a_0$ with attributes $a_0, a_1 \in Syn$ and $b_1 \in Inh$ are defined by $a_0(\pi) \xrightarrow{\sigma} \top$, $a_1(\pi) \xrightarrow{\sigma} b_1(\pi)$ and $b_1(\pi 1) \xrightarrow{\sigma} a_0(\pi 1)$ in R^σ, respectively.

The set of states of the automaton corresponding to M is $(\Delta \cup Syn)^{n+1}$ where n is the cardinality of Inh. It is finite since Δ, Syn and Inh are finite. The initial state is $\langle rhs_\sharp(a_0), rhs_\sharp(b_1), \ldots, rhs_\sharp(b_n)\rangle$. The set of final states is $\{\top\} \times (\Delta \cup Syn)^n$. The transition rules of the states are defined by the *disentangling algorithm*, Algorithm 2 in [NN01]. When the current state is $\langle v_0, v_1, \ldots, v_n\rangle$ and the input symbol σ is read, the next state is defined by consulting R^σ as follows. If v_0 is either $\top$ or $\bot$, then the state does not change. Otherwise v_0 is $a_k \in Syn$ with $k \in [m] \cup \{0\}$. Let us define two functions $f : Syn \cup Inh \to \Delta \cup Syn$ and $g : Inh \to \Delta \cup Syn$ such that

$$f(a) = \begin{cases} rhs_\sigma(a) & (\text{if } rhs_\sigma(a) \in \Delta \cup Syn) \\ g(rhs_\sigma(a)) & (\text{if } rhs_\sigma(a) \in Inh) \end{cases}$$

$$g(b_i) = \begin{cases} v_i & (\text{if } v_i \in \Delta) \\ f(v_i) & (\text{if } v_i \in Syn) \end{cases}$$

The next state is defined by $\langle f(a_k), f(b_1), \ldots, f(b_n)\rangle$. The above recursion always terminates because M is well-defined.

On the other hand, the transformation τ_{pbc} can be defined by the satt

$$M_{pbc} = (\{a_0\}, \emptyset, \{s\}, \emptyset, \Sigma, \Delta, a_0, \sharp, R)$$

with the set R of the following attribute rules:

$$\begin{array}{llll} a_0(\pi) \xrightarrow{\sharp} Head(s(\pi 1)) & & & \\ a_0(\pi) \xrightarrow{parenL} \bot & & s(\pi) \xrightarrow{parenL} Tail(s(\pi 1)) & \\ a_0(\pi) \xrightarrow{parenR} \bot & & s(\pi) \xrightarrow{parenR} Cons(\bot, s(\pi 1)) & \\ a_0(\pi) \xrightarrow{end} \bot & & s(\pi) \xrightarrow{end} Cons(\top, Empty) & \end{array}$$

□

3.3 Simulation of Stack-Attributed Tree Transducers with Attributed Tree Transducers

This section shows that every satt can be simulated by an att as long as the inputs are restricted. It helps us to prove the correctness of the composition of satts by using that of atts. For a given

$$\begin{array}{lcl} \langle \boldsymbol{a},1\rangle(\pi) & \xrightarrow{\sigma} & \eta \\ \langle \boldsymbol{a},2\rangle(\pi) & \xrightarrow{\sigma} & \langle \boldsymbol{a},1\rangle(\pi 1) \\ & \vdots & \\ \langle \boldsymbol{a},n\rangle(\pi) & \xrightarrow{\sigma} & \langle \boldsymbol{a},n-1\rangle(\pi 1) \end{array}$$

Figure 6(a): For $\boldsymbol{a}(\pi) \xrightarrow{\sigma} \mathit{Cons}(\eta,\boldsymbol{a}(\pi 1))$

$$\begin{array}{lcl} \langle \boldsymbol{a},1\rangle(\pi) & \xrightarrow{\sigma} & \langle \boldsymbol{a},2\rangle(\pi 1) \\ & \vdots & \\ \langle \boldsymbol{a},n-1\rangle(\pi) & \xrightarrow{\sigma} & \langle \boldsymbol{a},n\rangle(\pi 1) \\ \langle \boldsymbol{a},n\rangle(\pi) & \xrightarrow{\sigma} & \bot \end{array}$$

Figure 6(b): For $\boldsymbol{a}(\pi) \xrightarrow{\sigma} \mathit{Tail}(\boldsymbol{a}(\pi 1))$

Figure 6: Attribute rules of an att simulating those of a satt.

input tree, the depth of stacks involved in the derivation relation induced by a well-defined satt is finitely bounded. This is clear from the fact that the length of the derivation is finite and the increase of the depth of the stacks in every derivation step is finite. This implies that, once the input is fixed, a satt can be simulated by an att obtained by replacing stack attributes with a finite number of output attributes. For example, suppose that the number of trees to be stored in a stack is less than n. Then, an attribute rule $\boldsymbol{a}(\pi) \xrightarrow{\sigma} \mathit{Cons}(\eta,\boldsymbol{a}(\pi 1))$ with a synthesized stack attribute $\boldsymbol{a}$ in the satt can be replaced with the attribute rules in Figure 6(a), where each $\langle \boldsymbol{a},i\rangle (i \in [n])$ is a synthesized attribute of an att that indicates the i-th top-most stack element. An attribute rule $\boldsymbol{a}(\pi) \xrightarrow{\sigma} \mathit{Tail}(\boldsymbol{a}(\pi 1))$ in the satt is replaced with the attribute rules in Figure 6(b). Since any elements deeper than level n are not referred to by the stack, we can use $\bot$ instead of $\langle \boldsymbol{a},n+1\rangle(\pi 1)$ in the right-hand side of the last rule.

We first introduce two kinds of simulating functions α_o and α_s for a stack system $\mathcal{S}$. The two functions α_o and α_s are defined over $\mathit{EXP}_o(\mathcal{S})$ and $\mathit{EXP}_s(\mathcal{S}) \times \mathbb{N}_+$, respectively: α_o represents a map from $\mathit{EXP}_o(\mathcal{S})$ onto $T_\Delta(A)$ and α_s represents a map from $\mathit{EXP}_s(\mathcal{S}) \times \mathbb{N}_+$ onto $T_\Delta(A)$ where A is a set of variables. The value of $\alpha_s(e,i)$ corresponds to the value of the i-th element in the stack represented by e. Both simulating functions eliminate all occurrences of the stack operators – *Cons*, *Empty*, *Head* and *Tail*.

Definition 3.16 (Simulating functions for a stack system) Let $\mathcal{S} = (X_o, X_s, \Delta)$ be a stack system, $\Gamma_o : X_o \to A$ and $\Gamma_s : X_s \times \mathbb{N}_+ \to A$ be functions, called *base simulating functions*, where A is a set of variables, and $n \in \mathbb{N}_+$. We define two *simulating functions* $\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n} : \mathit{EXP}_o(\mathcal{S}) \to T_\Delta(A)$ and $\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n} : \mathit{EXP}_s(\mathcal{S}) \times \mathbb{N}_+ \to T_\Delta(A)$ as follows:

$$\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e) = \Gamma_o(e) \quad \text{if } e \in X_o. \tag{50}$$

$$\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e,i) = \Gamma_s(e,i) \quad \text{if } e \in X_s. \tag{51}$$

$$\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\delta(e_1,\ldots,e_m)) = \delta(\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e_1),\ldots,\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e_m)) \tag{52}$$

$$\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\mathit{Cons}(e_1,e_2),i) = \begin{cases} \alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e_1) & (\text{if } i=1) \\ \alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e_2,i-1) & (\text{otherwise}) \end{cases} \tag{53}$$

$$\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\mathit{Empty},i) = \bot \tag{54}$$

$$\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\mathit{Head}(e)) = \alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e,1) \tag{55}$$

$$\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\mathit{Tail}(e),i) = \begin{cases} \bot & (\text{if } i \geq n) \\ \alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e,i+1) & (\text{otherwise}) \end{cases} \tag{56}$$

We write $\mathcal{A}_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e)$ with $e \in \mathit{EXP}_o(\mathcal{S})$ for the subset of $\mathit{EXP}_o(\mathcal{S}) \cup (\mathit{EXP}_s(\mathcal{S}) \times [n])$ which

consists of all e'_o and $\langle e'_s, i' \rangle$ such that $\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e'_o)$ and $\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e'_s, i')$ occur in the computation of $\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e)$. Similarly, we write $\mathcal{A}_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e,i)$ with $e \in EXP_s(\mathcal{S})$ and $i \in [n]$ for the set of occurrences in the computation of $\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e,i)$. For example, for $n \geq 2$, $\mathcal{A}_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(Head(Tail(e))) = \{Head(Tail(e)), \langle Tail(e), 1\rangle\} \cup \mathcal{A}_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e,2)$. From the definition, e_1 and e_2 are subexpressions of e when $e_1 \in \mathcal{A}_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e)$ and $\langle e_2, i\rangle \in \mathcal{A}_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e)$. This property holds for $\mathcal{A}_s$ as well. We may omit the superscripts of α_o, α_s, $\mathcal{A}_o$ and $\mathcal{A}_s$ when they are clear from the context. □

We first show the following lemma describing that both simulating functions are context-independent.

Lemma 3.17 *Let $\mathcal{S} = (X_o, X_s, \Delta)$ be a stack system, let Γ_o and Γ_s be base simulating functions onto A where A is a set of variables, and let $=_\alpha$ be a binary relation over $EXP_o(\mathcal{S}) \cup EXP_s(\mathcal{S})$ such that $\eta_1 =_\alpha \eta_2$ only if one of the two following conditions holds, either $\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\eta_1) = \alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\eta_2)$ holds where $\eta_1, \eta_2 \in EXP_o(\mathcal{S})$ or $\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\eta_1, i) = \alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\eta_2, i)$ holds for any i where $\eta_1, \eta_2 \in EXP_s(\mathcal{S})$. Then we have*

$$\mathcal{E}[\eta_1] =_\alpha \mathcal{E}[\eta_2] \quad \textit{if} \quad \eta_1 =_\alpha \eta_2$$

for any $(EXP_o(\mathcal{S}) \cup EXP_s(\mathcal{S}))$-context $\mathcal{E}$.

Proof. This lemma can be proved by an easy induction on the structures of $\mathcal{E}$. □

A satt can be simulated by an att if the input tree is fixed. The simulation is defined as follows.

Definition 3.18 (n-depth simulation of a satt) Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, in, \sharp, R)$ be a satt, and $\mathcal{S} = (X_o, X_s, \Delta)$ be the stack system with

$$\begin{aligned} X_o &= \{a(\varphi) \mid a \in Syn \cup Inh,\ \varphi \in \Pi\} \quad \text{and} \\ X_s &= \{\boldsymbol{a}(\varphi) \mid \boldsymbol{a} \in StSyn \cup StInh,\ \varphi \in \Pi\} \end{aligned}$$

where $\Pi = \{\pi, \pi 1, \pi 2, \dots\}$. Let Γ_o be the function such that $\Gamma_o(a(\varphi)) = a(\varphi)$ for every $a(\varphi) \in X_o$ and Γ_s the function such that $\Gamma_s(\boldsymbol{a}(\varphi), i) = \langle \boldsymbol{a}, i\rangle(\varphi)$ for every $\boldsymbol{a}(\varphi) \in X_s$ and $i \in \mathbb{N}_+$. The *n-depth simulation* $sim_n(M)$ for $n \in \mathbb{N}_+$ is the following att:

$$sim_n(M) \ = \ (Syn', Inh', \Sigma, \Delta, in, \sharp, R')$$

where $Syn' = Syn \cup (StSyn \times [n])$, $Inh' = Inh \cup (StInh \times [n])$ and R' is defined by

$$\begin{aligned} R' \ = \ & \{a(\varphi) \xrightarrow{\sigma} \alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\eta) \mid a(\varphi) \xrightarrow{\sigma} \eta \in R\} \\ & \cup \{\langle \boldsymbol{a}, i\rangle(\varphi) \xrightarrow{\sigma} \alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\zeta, i) \mid i \in [n],\ \boldsymbol{a}(\varphi) \xrightarrow{\sigma} \zeta \in R\} \end{aligned}$$

□

Example 3.19 Let M_{ptoi} be the satt defined in Example 3.10. The 3-depth simulation of M_{ptoi} is specified by $sim_3(M_{ptoi}) = (Syn', Inh', \Sigma, \Delta, a_0, \sharp, R')$ where

- $Syn' = \{a_0\}$, $Inh' = \{\langle \boldsymbol{s}, 1\rangle, \langle \boldsymbol{s}, 2\rangle, \langle \boldsymbol{s}, 3\rangle\}$,

- Σ and Δ are the same as those of M_{ptoi},
- R' is the following set of attribute rules:

$$
\begin{array}{llll}
a_0(\pi) & \overset{\sharp}{\rightarrow} & a_0(\pi 1)) & \langle s,1\rangle(\pi 1) \overset{\sharp}{\rightarrow} \bot \\
 & & & \langle s,2\rangle(\pi 1) \overset{\sharp}{\rightarrow} \bot \\
 & & & \langle s,3\rangle(\pi 1) \overset{\sharp}{\rightarrow} \bot \\
a_0(\pi) & \overset{one}{\rightarrow} & a_0(\pi 1) & \langle s,1\rangle(\pi 1) \overset{one}{\rightarrow} one \\
 & & & \langle s,2\rangle(\pi 1) \overset{one}{\rightarrow} \langle s,1\rangle(\pi) \\
 & & & \langle s,3\rangle(\pi 1) \overset{one}{\rightarrow} \langle s,2\rangle(\pi) \\
a_0(\pi) & \overset{two}{\rightarrow} & a_0(\pi 1) & \langle s,1\rangle(\pi 1) \overset{two}{\rightarrow} two \\
 & & & \langle s,2\rangle(\pi 1) \overset{two}{\rightarrow} \langle s,1\rangle(\pi) \\
 & & & \langle s,3\rangle(\pi 1) \overset{two}{\rightarrow} \langle s,2\rangle(\pi) \\
a_0(\pi) & \overset{plus}{\rightarrow} & a_0(\pi 1) & \langle s,1\rangle(\pi 1) \overset{plus}{\rightarrow} plus(\langle s,2\rangle(\pi),\langle s,1\rangle(\pi)) \\
 & & & \langle s,2\rangle(\pi 1) \overset{plus}{\rightarrow} \langle s,3\rangle(\pi) \\
 & & & \langle s,3\rangle(\pi 1) \overset{plus}{\rightarrow} \bot \\
a_0(\pi) & \overset{multi}{\rightarrow} & a_0(\pi 1) & \langle s,1\rangle(\pi 1) \overset{multi}{\rightarrow} multi(\langle s,2\rangle(\pi),\langle s,1\rangle(\pi)) \\
 & & & \langle s,2\rangle(\pi 1) \overset{multi}{\rightarrow} \langle s,3\rangle(\pi) \\
 & & & \langle s,3\rangle(\pi 1) \overset{multi}{\rightarrow} \bot \\
a_0(\pi) & \overset{end}{\rightarrow} & \langle s,1\rangle(\pi) &
\end{array}
$$

Then we have

$$[\![sim_3(M_{ptoi})]\!](two(one(two(plus(multi(end)))))) = multi(two,plus(one,two))$$

just as for M_{ptoi}. On the other hand,

$$[\![sim_3(M_{ptoi})]\!](two(one(two(one(plus(multi(plus(end)))))))))$$

does not return the same value as M_{ptoi} because the depth of the stack is limited.

□

Let M be a satt and t a fixed input tree. If we take an n large enough, the derivations induced by $sim_n(M)$ and M on t coincide as shown by the following lemma and theorem. We consider the n-depth simulation for a derivation by M on t.

Definition 3.20 (n-depth simulation for derivations by a satt) Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, in, \sharp, R)$ be a satt, $t \in T_{\Sigma^+}$, $n \in \mathbb{N}_+$, let $\mathcal{S} = (X_o, X_s, \Delta)$ be the stack system defined in Definition 3.11, and let $\psi \in EXP_o(\mathcal{S})$ satisfy $in(\varepsilon) \Rightarrow^*_{M,t} \psi$. The *$n$-depth simulation* of ψ is $\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\psi)$ where Γ_o and Γ_s are the base simulating functions such that $\Gamma_o(a(w)) = a(w)$ for every $a(w) \in X_o$ and $\Gamma_s(\boldsymbol{a}(w),i) = \langle \boldsymbol{a}, i\rangle(w)$ for every $\boldsymbol{a}(w) \in X_s$ and $i \in \mathbb{N}_+$. □

Choosing the first branch of (56) in the computation of the n-depth simulation indicates that the stack depth is shorter than required. The following lemma shows that, if $\bot$ is derived by $\Rightarrow_{M,t}$, then $\bot$ is derived by $\Rightarrow_{sim_n(M),t}$ or the stack is overflowed.

Lemma 3.21 *Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, in, \sharp, R)$ be a satt, t be an input tree for M such that $[\![M]\!](t)$ can be defined, ψ be a term such that $in(\varepsilon) \Rightarrow^*_{M,\sharp(t)} \psi$, $n \in \mathbb{N}_+$ and $\mathcal{S}, \Gamma_o, \Gamma_s$ be as given in* Definition 3.20. *Let us write α_o and α_s for $\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}$ and $\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}$, respectively. Suppose that $n > \max\{i \mid \langle e, i\rangle \in \mathcal{A}_o(\psi)\}$. If $\alpha_o(\eta)$ and $\alpha_s(\eta, j)$ occur in the computation of $\alpha_o(\psi)$, then the following statements are true:*

(i) If $\alpha_o(\eta)$ is reducible by $\Rightarrow_{sim_n(M),\sharp(t)}$, then η is reducible by $\Rightarrow_{M,\sharp(t)}$.

(ii) If $\alpha_s(\zeta, k)$ with $k \in [n-1]$ is reducible by $\Rightarrow_{sim_n(M),\sharp(t)}$, then $Head(Tail^{k-1}(\zeta))$ is reducible by $\Rightarrow_{M,\sharp(t)}$.

where $Tail^0(e) = e$ and $Tail^k(e) = Tail(Tail^{k-1}(e))$ for $k \in \mathbb{N}_+$.

Proof. We prove these statements by induction on the structure of η and ζ.

(CASE $\eta = a(w)$ FOR (i) AND $\zeta = \boldsymbol{a}(w)$ FOR (ii)) It is clear that (i) and (ii) hold because $a(w)$ and $\boldsymbol{a}(w)$ are always reducible by the definition of $\Rightarrow_{M,\sharp(t)}$.

(CASE $\eta = \delta(\eta_1, \ldots, \eta_m)$ FOR (i)) Suppose $\alpha_o(\eta)$ is reducible by $\Rightarrow_{sim_n(M),\sharp(t)}$. Then $\alpha_o(\eta_k)$ is reducible by $\Rightarrow_{sim_n(M),\sharp(t)}$ by for some k. From the induction hypothesis, η_k is reducible by $\Rightarrow_{M,\sharp(t)}$. Hence η is reducible by $\Rightarrow_{M,\sharp(t)}$. Therefore (i) holds.

(CASE $\zeta = Cons(\eta', \zeta')$ FOR (ii)) If $k = 1$, then (ii) holds since $Head(\zeta) = Head(Cons(\eta', \zeta'))$ is reducible into η'. If $k > 1$, then (ii) holds since $Head(Tail^{k-1}(\zeta)) = Head(Tail^{k-2}(Tail(Cons(\eta', \zeta'))))$ is reducible into $Head(Tail^{k-2}(\zeta'))$.

(CASE $\zeta = Empty$ FOR (ii)) Since $Head(Tail^{k-1}(Empty))$ is reducible to $\bot$, (ii) holds.

(CASE $\eta = Head(\zeta')$ FOR (i)) Suppose that $\alpha_o(\eta) = \alpha_s(\zeta', 1)$ is reducible by $\Rightarrow_{sim_n(M),\sharp(t)}$. $Head(\zeta')$ is reducible from the induction hypothesis of (ii). Therefore (i) holds.

(CASE $\zeta = Tail(\zeta')$ FOR (ii)) We have $\alpha_s(\zeta, k) = \alpha_s(\zeta', k+1)$ from $k < n$ which is the assumption on n and k in this lemma. From the induction hypothesis, $Head(Tail^k(\zeta'))$ is reducible. Since $Head(Tail^k(\zeta')) = Head(Tail^{k-1}(\zeta))$, (ii) holds. □

The next theorem shows that given any satt with a fixed input, there exists an n-depth simulation for some n large enough such that it can mimic the original satt for the input.

Theorem 3.22 *Let M be a satt and t be an input tree for M. There exists $n_t \in \mathbb{N}$ such that $[\![sim_n(M)]\!](t) = [\![M]\!](t)$ for every $n \geq n_t$.*

Proof. Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, in, \sharp, R)$ be a satt, $t \in T_\Sigma$ be an input tree for M such that the output tree $t' = [\![M]\!](t)$ is defined and $n \in \mathbb{N}_+$. Let $\mathcal{S}$, Γ_o and Γ_s be as given in Definition 3.20. We use α_o, α_s for $\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}$ and $\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}$, respectively. Let $\mathcal{S}'$, Γ'_o and Γ'_s be $\mathcal{S}$, Γ_o and Γ_s as given in Definition 3.18, respectively. We use α'_o and α'_s for $\alpha_o^{\mathcal{S}',\Gamma'_o,\Gamma'_s,n}$ and $\alpha_s^{\mathcal{S}',\Gamma'_o,\Gamma'_s,n}$, respectively.

Let $\mathcal{A}_o(t)$ denote the union of all $\mathcal{A}_o(\psi)$ with $in(\varepsilon) \Rightarrow^*_{M,\sharp(t)} \psi$. Suppose that $n > \max\{i \mid \langle e, i\rangle \in \mathcal{A}_o(t)\}$. We show that, for $\psi \in EXP_o(\mathcal{S})$ and $\psi \in EXP_s(\mathcal{S})$, respectively,

(i) If $\psi \in \mathcal{A}_o(t)$ and $\psi \Rightarrow_{M,\sharp(t)} \psi'$, then $\alpha_o(\psi) \Rightarrow^?_{sim_n(M),\sharp(t)} \alpha_o(\psi')$.

(ii) If $\langle \psi, j\rangle \in \mathcal{A}_o(t)$, $\psi \Rightarrow_{M,\sharp(t)} \psi'$ and $j \in [n-1]$, then $\alpha_s(\psi, j) \Rightarrow^?_{sim_n(M),\sharp(t)} \alpha_s(\psi', j)$.

This implies $nf(\Rightarrow_{sim_n(M),\sharp(t)}, \alpha_o(in(\varepsilon))) = \alpha_o(nf(\Rightarrow_{M,\sharp(t)}, in(\varepsilon)))$ from Lemma 3.21. Since we have $\alpha_o(in(\varepsilon)) = in(\varepsilon)$ and $\alpha_o(t') = t'$ because of $t' \in T_\Delta$ and the definition of α_o, we can conclude $[\![sim_n(M)]\!](t) = [\![M]\!](t)$.

We prove the statements (i) and (ii) by induction on the structure of φ. The possible cases on φ are specified by left-hand sides of the derivation relation induced by $\Rightarrow_{M,\sharp(t)}$ in Definition 3.11.

(CASE $\varphi = a(w)$ WITH $a \in Syn$) We have $\varphi \Rightarrow_{M,\sharp(t)} \eta[\pi := w]$ where $a(\pi) \xrightarrow{\sigma} \eta \in R$. Since $sim_n(M)$ has a rule $a(\pi) \xrightarrow{\sigma} \alpha'_o(\eta)$, we have $a(w) \Rightarrow_{sim_n(M),\sharp(t)} \alpha'_o(\eta)[\pi := w]$. Now we can show $\alpha_o(\eta[\pi := w]) = \alpha'_o(\eta)[\pi := w]$ by simple induction on η with the definitions of α_o and α'_o. Since we have $\alpha_o(a(w)) = a(w)$, (i) holds.

(CASE $\varphi = b(w)$ WITH $b \in Inh$) Similar to the previous case.

(CASE $\varphi = \boldsymbol{a}(w)$ WITH $\boldsymbol{a} \in StSyn$) We have $\varphi \Rightarrow_{M,\sharp(t)} \zeta[\pi := w]$ where $\boldsymbol{a}(\pi) \xrightarrow{\sigma} \zeta \in R$. Since $sim_n(M)$ has a rule $\langle \boldsymbol{a}, j\rangle(\pi) \xrightarrow{\sigma} \alpha'_s(\zeta, j)$ for every $j \in [n]$, we have $\langle \boldsymbol{a}, j\rangle(w) \Rightarrow_{sim_n(M),\sharp(t)} \alpha'_s(\zeta, j)[\pi := w]$. Similar to the first case, we can show $\alpha'_s(\zeta, j)[\pi := w] = \alpha_s(\zeta[\pi := w], j)$. Then (ii) holds.

(CASE $\varphi = \boldsymbol{b}(wi)$ WITH $\boldsymbol{b} \in StInh$) Similar to the previous case.

(CASE $\varphi = Head(Cons(\eta, \zeta))$) We have $\varphi \Rightarrow_{M,\sharp(t)} \eta$. Since we have $\alpha_o(Head(Cons(\eta, \zeta))) = \alpha_o(\eta)$, (i) holds.

(CASE $\varphi = Head(Empty)$) We have $\varphi \Rightarrow_{M,\sharp(t)} \bot$. Since we have $\alpha_o(Head(Empty)) = \bot$ and $\alpha_o(\bot) = \bot$, (i) holds.

(CASE $\varphi = Head(\zeta)$ WITH $\zeta \neq Cons(\eta_1, \zeta_1)$ AND $\zeta \neq Empty$) We have $\varphi \Rightarrow_{M,\sharp(t)} Head(\zeta')$ where $\zeta \Rightarrow_{M,\sharp(t)} \zeta'$. We find $\alpha_s(\zeta, 1) \Rightarrow^?_{sim_n(M),\sharp(t)} \alpha_s(\zeta', 1)$ from the induction hypothesis. Since we have $\alpha_o(Head(\rho)) = \alpha_s(\rho, 1)$ with $\rho = \zeta, \zeta'$, (i) holds.

(CASE $\varphi = Tail(Cons(\eta, \zeta))$) We have $\varphi \Rightarrow_{M,\sharp(t)} \zeta$. Since we have $\alpha_s(Tail(Cons(\eta, \zeta)), j) = \alpha_s(\zeta, j)$ for $1 \leq j < n$ from the assumption on n, (ii) holds.

(CASE $\varphi = Tail(Empty)$) We have $\varphi \Rightarrow_{M,\sharp(t)} Empty$. Since $\alpha_s(Tail(Empty), j) = \alpha_s(Empty, j+1)(= \bot)$ for any j, (ii) holds.

(CASE $\varphi = Tail(\zeta)$ WITH $\zeta \neq Cons(\eta_1, \zeta_1)$ AND $\zeta \neq Empty$) We have $\varphi \Rightarrow_{M,\sharp(t)} Tail(\zeta')$ where $\zeta \Rightarrow_{M,\sharp(t)} \zeta'$. We find $\alpha_s(\zeta, j-1) \Rightarrow^?_{sim_n(M),\sharp(t)} \alpha_s(\zeta', j-1)$ with $j > n$ from the induction hypothesis and the assumption on n. Since $\alpha_s(Tail(\rho), j) = \alpha_s(\rho, j-1)$ with $\rho = \zeta, \zeta'$, (ii) holds.

(CASE $\varphi = \delta(\eta_1, \ldots, \eta_k, \ldots, \eta_m)$ WHERE $\eta_k \Rightarrow_{M,\sharp(t)} \eta'_k$ AND $\eta_1, \ldots, \eta_{k-1}$ ARE IRREDUCIBLE) We have $\delta(\eta_1, \ldots, \eta_k, \ldots, \eta_m) \Rightarrow_{M,\sharp(t)} \delta(\eta_1, \ldots, \eta'_k, \ldots, \eta_m)$ by the definition of $\Rightarrow_{M,\sharp(t)}$. By Lemma 3.21 (i), $\alpha_o(\eta_1), \ldots, \alpha_o(\eta_{k-1})$ are irreducible. We find $\alpha_o(\eta_k) \Rightarrow^?_{sim_n(M),\sharp(t)} \alpha_o(\eta'_k)$ from the induction hypothesis, Then we have $\delta(\alpha_o(\eta_1), \ldots, \alpha_o(\eta_k), \ldots, \alpha_o(\eta_m)) \Rightarrow^?_{sim_n(M),\sharp(t)} \delta(\alpha_o(\eta_1), \ldots, \alpha_o(\eta'_k), \ldots, \alpha_o(\eta_m))$. Therefore (i) holds. □

For a satt M and an input tree t, we write $lub(M, t)$ to denote the minimum possible number n_t in Theorem 3.22.

Corollary 3.23 *Let M be a satt. If $sim_n(M)$ is a well-defined att for any $n \in \mathbb{N}_+$, M is a well-defined satt.*

Proof. Let M be a satt $(Syn, Inh, StSyn, StInh, \Sigma, \Delta, in, \sharp, R)$. From Definition 3.13, it suffices to show that there exists $nf(\Rightarrow_{M,\sharp(t)}, in(\varepsilon))$ for every input $t \in T_\Sigma$ if $sim_n(M)$ is a well-defined att

for any $n \in \mathbb{N}_+$.

Let $t \in T_\Sigma$ and $n_t \geq lub(M, \sharp(t))$. Suppose that $sim_n(M)$ is a well-defined att for any $n \in \mathbb{N}_+$. Then there exists $nf(\Rightarrow_{sim_{n_t}(M),\sharp(t)}, in(\varepsilon))$ which is $[\![sim_{n_t}(M)]\!](t)$. Since $[\![M]\!](t) = [\![sim_{n_t}(M)]\!](t)$ holds from Theorem 3.22, there exists $nf(\Rightarrow_{M,\sharp(t)}, in(\varepsilon))$ which is $[\![M]\!](t)$. □

4 Composing Stack-Attributed Tree Transducers

We present a composition method of SATTs by extending a composition method of ATTs. In this section, we first review the *descriptional composition*[GG84, Gie88], a composition method for attribute grammars, in terms of ATTs. Next we introduce a composition method of SATTs by extending an algorithm of descriptional composition. Finally we prove the correctness of the algorithm and the closure property of the composition.

4.1 Descriptional Composition

We present an algorithm of descriptional composition following the presentation in [CDPR99]. Here, the algorithm is formalized in terms of ATTs. Let us first give a condition, called *syntactic single use requirement* in [GG84, Gie88], under which descriptional composition is successfully applied.

Definition 4.1 An ATT $M = (Syn, Inh, \Sigma, \Delta, a_0, \sharp, R)$ satisfies the *single use requirement*(sur) if there is no pair of rules $x_1 \xrightarrow{\sigma} \mathcal{E}_1[a(\varphi)]$ and $x_2 \xrightarrow{\sigma} \mathcal{E}_2[a(\varphi)]$ in R^σ for every $\sigma \in \Sigma \cup \{\sharp\}$, $a \in Syn \cup Inh$, and $\varphi \in \{\pi, \pi 1, \pi 2 \cdots\}$ where $\mathcal{E}_1$ and $\mathcal{E}_2$ are contexts. We write ATT_{su} for a set of tree transformers such that $ATT_{su} = \{[\![M]\!] \mid M \text{ is a well-defined sur-ATT.}\}$.

For given two ATTs M_1 and M_2, we write $M_1 \odot M_2$ to denote a single ATT which is the result of descriptional composition of M_1 and M_2. The ATT $M_1 \odot M_2$ computes a transformation equivalent to $[\![M_1]\!] \circ [\![M_2]\!]$, thus $M_1 \odot M_2$ takes a tree over the input alphabet of M_2 and returns a tree over the output alphabet of M_1. The descriptional composition is divided into three steps: projection, symbolic evaluation and renaming.

Definition 4.2 Let $M_1 = (Syn_1, Inh_1, \Sigma_1, \Delta_1, a_1, \sharp_1, R_1)$ and $M_2 = (Syn_2, Inh_2, \Sigma_2, \Delta_2, a_2, \sharp_2, R_2)$ be ATTs with $\Delta_2 \subset \Sigma_1$ and let $\Sigma_2' = \Sigma_2 \uplus \{\sharp_2\}$. The ATT $M_1 \odot M_2$ is obtained by $ren \circ se_{M_1} \circ proj_{M_1}(M_2)$, where three functions ren, se_{M_1} and $proj_{M_1}$ are defined as follows:

- $proj_{M_1}(M_2)$ returns $U = (Syn_2, Inh_2, \Sigma_2, \Delta_2, a_2, \sharp_2, R)$ where

$$
\begin{aligned}
R &= \{a(x) \xrightarrow{\gamma} a(\eta_{x,\gamma}) \mid a \in Syn_1,\ x \xrightarrow{\gamma} \eta \in R_2,\ \gamma \in \Sigma_2'\} \\
&\quad \cup \{b(\eta_{x,\gamma}) \xrightarrow{\gamma} b(x) \mid b \in Inh_1,\ x \xrightarrow{\gamma} \eta \in R_2,\ \gamma \in \Sigma_2'\} \\
\eta_{x,\gamma} &= \begin{cases} \sharp_1(\eta) & (\text{if } x = a_2(\pi) \text{ and } \gamma = \sharp_2) \\ \eta & (\text{otherwise}) \end{cases}
\end{aligned}
$$

  The calculation of $proj_{M_1}(M_2)$ is called *projection*. Note that U is just an intermediate representation and is not an ATT.

- $se_{M_1}(U)$ with $U = (Syn_2, Inh_2, \Sigma_2, \Delta_2, a_2, \sharp_2, \bigcup_{\gamma\in\Sigma_2'} R^\gamma)$ returns

$$(Syn_2, Inh_2, \Sigma_2, \Delta_2, a_2, \sharp_2, \bigcup_{\gamma\in\Sigma_2'} nf(\Rightarrow_{SE}, R^\gamma))$$

where the binary relation $\Rightarrow_{SE}$ is defined by the following clause. $P \Rightarrow_{SE} Q$ holds iff

$$\begin{aligned}
P &= \{b_i(\sigma(e_1,\cdots,e_n)) \xrightarrow{\gamma} \zeta_i \mid 1 \le i \le m\} \uplus R^\gamma_{misc} \\
Q &= \left\{ b_i(e_j) \xrightarrow{\gamma} \theta(\eta) \;\middle|\; \begin{array}{l} b_i(\pi j) \xrightarrow{\sigma} \eta \in R^\sigma_1, \\ 1 \le i \le m, 1 \le j \le n \end{array} \right\} \\
&\quad \cup \left\{ x \xrightarrow{\gamma} \rho^*(\eta) \;\middle|\; \begin{array}{l} x \xrightarrow{\gamma} \eta \in R^\gamma_{misc}, \\ \rho = [a_k(\sigma(e_1,\cdots,e_n)) := \theta(\psi_k)]_{1\le k\le l}, \\ a_k(\pi) \xrightarrow{\sigma} \psi_k \in R^\sigma_1, 1 \le k \le l \end{array} \right\} \\
\theta &= [b_i(\pi) := \zeta_i]_{1\le i\le m}[\pi j := e_j]_{1\le j\le n}
\end{aligned}$$

with $\gamma \in \Sigma_2'$, $\sigma \in \Delta_2^{(n)}$, $Syn_1 = \{a_1,\cdots,a_l\}$ and $Inh_1 = \{b_1,\cdots,b_m\}$. The calculation of $se_{M_1}(U)$ is called *symbolic evaluation*. Note that both sides of any rule in $nf(\Rightarrow_{SE}, R^\gamma)$ do not have occurrences of expressions of the form $a(\sigma(\eta_1,\cdots,\eta_n))$.

- $ren(U)$ with $U = (Syn_2, Inh_2, \Sigma_2, \Delta_2, a_2, \sharp_2, R)$ returns

$$(Syn, Inh, \Sigma_2, \Delta_1, \langle a_1, a_2\rangle, \sharp_2, \Theta(R) \cup R_{dmy})$$

where

$$\begin{aligned}
Syn &= \{\langle a, a'\rangle \mid \langle a, a'\rangle \in Syn_1 \times Syn_2 \cup Inh_1 \times Inh_2\} \\
Inh &= \{\langle a, a'\rangle \mid \langle a, a'\rangle \in Syn_1 \times Inh_2 \cup Inh_1 \times Syn_2\} \\
\Theta(R) &= \{x' \xrightarrow{\sigma} \eta' \mid x' = \theta(x), \eta' = \theta(\eta), x \xrightarrow{\sigma} \eta \in R\} \\
\theta &= [a(a'(\varphi)) := \langle a, a'\rangle(\varphi)]_{a\in Att_1, a'\in Att_2, \varphi\in\{\pi,\pi1,\pi2,\cdots\}}
\end{aligned}$$

with $Att_1 = Syn_1 \cup Inh_1$ and $Att_2 = Syn_2 \cup Inh_2$. R_{dmy} is a set of dummy rules which gives a rule $\langle a, a'\rangle(\varphi) \xrightarrow{\sigma} \perp$ for any rule $a(a'(\varphi)) \xrightarrow{\sigma} \zeta \notin R$, $a \in Att_1$, $a' \in Att_2$ and $\varphi \in \{\pi, \pi1, \pi2, \cdots\}$. The calculation of $ren(U)$ is called *renaming*.

The correctness of the descriptional composition method is guaranteed by the following theorem.

Theorem 4.3 ($\odot$-Correctness, Ganzinger[Gan83] and Giegerich[Gie88]) *If M_1 and M_2 are well-defined sur-ATTs, then $M_1 \odot M_2$ is a well-defined sur-ATT such that $[\![M_1]\!] \circ [\![M_2]\!] = [\![M_1 \odot M_2]\!]$.*

Corollary 4.4 $ATT_{su} \circ ATT_{su} = ATT_{su}$.

Proof. The statement follows immediately from the fact that ATT_{su} contains the identical tree transformation and that $ATT_{su} \circ ATT_{su} \subseteq ATT_{su}$ holds from Theorem 4.3. □

Consider the case where M_1 in Definition 4.2 is a TDTT, *i.e.,* M_1 has no inherited attribute. Then the descriptional composition method is equivalent to the composition method of a TDTT and an ATT presented in [Fül81]. Therefore we have the following theorem. This composition requires no condition such as the sur-condition required in Theorem 4.3.

Theorem 4.5 ($\odot$-Correctness, Fülöp[Fül81]) *If M_1 is a TDTT and M_2 is a well-defined ATT, then $M_1 \odot M_2$ is a well-defined ATT such that $[\![M_1]\!] \circ [\![M_2]\!] = [\![M_1 \odot M_2]\!]$.*

Corollary 4.6 $TDTT \circ ATT = ATT$.

Proof. The statement follows immediately from the fact that $TDTT$ contains the identical tree transformation and that $TDTT \circ ATT \subseteq ATT$ holds from Theorem 4.5. □

4.2 Extended Descriptional Composition

We extend the above descriptional composition to apply to SATTs. As mentioned in Section 1, we consider a composition of ATTs and SATTs. The result of the composition is obtained as a single SATT. We first present the condition under which the extended composition method is successfully applied before introducing the method. As the method is defined by an extension of the algorithm in Definition 4.2, the condition is also presented by an extension of the sur condition in Definition 4.1.

Definition 4.7 A SATT M is a *sur-SATT* if $sim_n(M)$ is a sur-ATT for any $n \in \mathbb{N}_+$. We write $SATT_{su}$ for a set of tree transformers such that $SATT_{su} = \{[\![M]\!] \mid M \text{ is a well-defined sur-SATT.}\}$.

The above definition is not directly applied to check if a SATT satisfies the sur-condition, since we need to check the sur-condition for infinitely many ATTs $sim_n(M)$ for any $n \in \mathbb{N}_+$. We do not discuss a checking method for the sur-condition of SATTs. However, there is a finitely checking method for the sur-condition of SATT. For instance, we can claim that M_{ptoi} in Example 3.10 is a sur-SATT. Although $s(\pi)$ is referred three times in a *plus*-rule (46), these three references do not overlap each other: $Head(s(\pi))$ represents a reference to the first element of the stack; $Head(Tail(s(\pi)))$ represents a reference to the second element of the stack; $Tail(Tail(s(\pi)))$ represents a reference to a stack comprised elements following the second element of the stack.

An algorithm of the extended descriptional composition is also divided into three steps. The intermediate results of the first and second step, projection and symbolic evaluation, do not have the form of SATTs similarly as in the original composition method for ATTs. A set of terms occurring at the both sides of attribute rules in the intermediate result is defined by the set BHS_o or BHS_s, as defined below.

Definition 4.8 Let $M_1 = (Syn_1, Inh_1, \Sigma_1, \Delta_1, a_1, \sharp_1, R_1)$ and $M_2 = (Syn_2, Inh_2, StSyn_2, StInh_2, \Sigma_2, \Delta_2, a_2, \sharp_2, R_2)$ be an ATT and a SATT with $\Sigma_1 \supset \Delta_2$, respectively, and let $\mathcal{S} = (X_o, X_s, \Delta_1)$ be a stack system with

$$\begin{aligned} X_o &= \{a(\eta) \mid a \in Syn_1 \cup Inh_1, \eta \in RHS_o(M_2)\} \\ &\quad \cup \{a(\sharp_1(\eta)) \mid a \in Syn_1 \cup Inh_1, \eta \in RHS_o(M_2)\} \\ X_s &= \{a(\zeta) \mid a \in Syn_1 \cup Inh_1, \zeta \in RHS_s(M_2)\}. \end{aligned}$$

The set $BHS_o(M_1, M_2)$ and the set $BHS_s(M_1, M_2)$ are defined by the following sets:

$$BHS_o(M_1, M_2) \stackrel{def}{=} EXP_o(\mathcal{S}) \quad \text{and} \quad BHS_s(M_1, M_2) \stackrel{def}{=} EXP_s(\mathcal{S}).$$

We present the extended descriptional composition method for SATTs below. The major difference from the original one is found in the step of symbolic evaluation. The original symbolic evaluation process is intend to eliminate occurrences of trees of the form $a(\sigma(e_1,\cdots,e_n))$. In addition to this, the extended one is intend to eliminate occurrences of trees of the forms $a(Cons(e_1,e_2))$, $a(Empty)$, $a(Head(e))$ and $a(Tail(e))$.

Definition 4.9 Let $M_1 = (Syn_1, Inh_1, \Sigma_1, \Delta_1, a_1, \sharp_1, R_1)$ and $M_2 = (Syn_2, Inh_2, StSyn_2, StInh_2, \Sigma_2, \Delta_2, a_2, \sharp_2, R_2)$ be an ATT and a SATT with $\Delta_2 \subset \Sigma_1$, respectively, let $\mathcal{R}$ be a set $\{\eta_1 \xrightarrow{\sigma} \eta_2 \mid \langle \eta_1, \eta_2 \rangle \in BHS_o(M_1,M_2) \times BHS_o(M_1,M_2) \cup BHS_s(M_1,M_2) \times BHS_s(M_1,M_2)\}$ and let $\Pi = \{\pi, \pi 1, \pi 2, \cdots\}$. The SATT $M_1 \hat{\odot} M_2$ is obtained by $\widehat{ren} \circ \widehat{se}_{M_1} \circ \widehat{proj}_{M_1}(M_2)$, where three functions $\widehat{ren}$, $\widehat{se}_{M_1}$ and $\widehat{proj}_{M_1}$ are defined as follows:

- $\widehat{proj}_{M_1}(M_2)$ returns $U = (Syn_2, Inh_2, StSyn_2, StInh_2, \Sigma_2, \Delta_2, a_2, \sharp_2, R)$ where $R \subset \mathcal{R}$ is defined in the same way as in Definition 4.2. The calculation of $\widehat{proj}_{M_1}(M_2)$ is called *projection*.

- $\widehat{se}_{M_1}(U)$ with $U = (Syn_2, Inh_2, StSyn_2, StInh_2, \Sigma_2, \Delta_2, a_2, \sharp_2, \bigcup_{\gamma \in \Sigma_2 \cup \{\sharp_2\}} R^\gamma)$ returns

$$(Syn_2, Inh_2, StSyn_2, StInh_2, \Sigma_2, \Delta_2, a_2, \sharp_2, \bigcup_{\gamma \in \Sigma_2 \cup \{\sharp_2\}} nf(\Rightarrow_{\widehat{SE}}, R^\gamma))$$

  where the binary relation $\Rightarrow_{\widehat{SE}}$ over $\mathcal{R}$ is defined as follows:

  1. $\varphi \Rightarrow_{\widehat{SE}} \psi$ if $\varphi \Rightarrow_{SE} \psi$ where $\Rightarrow_{SE}$ is the relation defined in Definition 4.2.
  2. $\{x \xrightarrow{\gamma} \mathcal{E}[a(Head(\zeta))]\} \uplus R \Rightarrow_{\widehat{SE}} \{x \xrightarrow{\gamma} \mathcal{E}[Head(a(\zeta))]\} \uplus R$.
  3. $\{x \xrightarrow{\gamma} \mathcal{E}[a(Tail(\zeta))]\} \uplus R \Rightarrow_{\widehat{SE}} \{x \xrightarrow{\gamma} \mathcal{E}[Tail(a(\zeta))]\} \uplus R$.
  4. $\{x \xrightarrow{\gamma} \mathcal{E}[a(Cons(\eta,\zeta))]\} \uplus R \Rightarrow_{\widehat{SE}} \{x \xrightarrow{\gamma} \mathcal{E}[Cons(a(\eta),a(\zeta))]\} \uplus R$.
  5. $\{x \xrightarrow{\gamma} \mathcal{E}[a(Empty)]\} \uplus R \Rightarrow_{\widehat{SE}} \{x \xrightarrow{\gamma} \mathcal{E}[Empty]\} \uplus R$.
  6. (i) $\left\{ \begin{array}{c} a(Head(\zeta)) \xrightarrow{\gamma} \eta' \\ a(Tail(\zeta)) \xrightarrow{\gamma} \zeta' \end{array} \right\} \uplus R \Rightarrow_{\widehat{SE}} \{a(\zeta) \xrightarrow{\gamma} Cons(\eta',\zeta')\} \uplus R$.

     (ii) $\{a(Head(\zeta)) \xrightarrow{\gamma} \eta'\} \uplus R \Rightarrow_{\widehat{SE}} \{a(\zeta) \xrightarrow{\gamma} Cons(\eta', Empty)\} \uplus R$,
     if R contains no rule whose left hand side is in the form of $a(Tail(\zeta))$.

     (iii) $\{a(Tail(\zeta)) \xrightarrow{\gamma} \zeta'\} \uplus R \Rightarrow_{\widehat{SE}} \{a(\zeta) \xrightarrow{\gamma} Cons(\bot,\zeta')\} \uplus R$,
     if R contains no rule whose left hand side is in the form of $a(Head(\zeta))$.
  7. $\{a(Cons(\eta,\zeta)) \xrightarrow{\gamma} \zeta'\} \uplus R \Rightarrow_{\widehat{SE}} \left\{ \begin{array}{c} a(\eta) \xrightarrow{\gamma} Head(\zeta') \\ a(\zeta) \xrightarrow{\sigma} Tail(\zeta') \end{array} \right\} \uplus R$.
  8. $\{a(Empty) \xrightarrow{\gamma} \zeta'\} \uplus R \Rightarrow_{\widehat{SE}} R$.

  where $a \in Syn_1 \cup Inh_1$, $\eta \in RHS_o(M_2)$, $\zeta \in RHS_s(M_2)$, $\eta' \in BHS_o(M_1,M_2)$, $\zeta' \in BHS_s(M_1, M_2)$ and $\mathcal{E}$ is a context, provided that the rewriting rule of 1 and 6 is applied only when no other rewriting rule can be applied.

The calculation of $\widehat{se}_{M_1}(U)$ is called *symbolic evaluation*. Note that both sides of any rule in $nf(\Rightarrow_{\widehat{SE}}, R^\gamma)$ do not include the form of $a(\sigma(\eta_1,\cdots,\eta_n))$, $a(Head(\zeta))$, $a(Tail(\zeta))$, $a(Cons(\eta,\zeta))$ or $a(Empty)$.

- $\widehat{ren}(U)$ with $U = (Syn_2, Inh_2, StSyn_2, StInh_2, \Sigma_2, \Delta_2, a_2, \sharp_2, R)$ returns

$$(Syn, Inh, StSyn, StInh, \Sigma_2, \Delta_1, \langle a_1, a_2\rangle, \sharp_2, \Theta(R) \cup R_{dmy})$$

where *Syn* and *Inh* is given is the same way as *ren* in Definition 4.2 and

$$\begin{aligned} StSyn &= \{\langle a, \boldsymbol{a}'\rangle \mid \langle a, \boldsymbol{a}'\rangle \in Syn_1 \times StSyn_2 \cup Inh_1 \times StInh_2\} \\ StInh &= \{\langle a, \boldsymbol{a}'\rangle \mid \langle a, \boldsymbol{a}'\rangle \in Syn_1 \times StInh_2 \cup Inh_1 \times StSyn_2\} \\ \Theta(R) &= \{x' \xrightarrow{\sigma} \eta' \mid x' = \theta(x), \eta' = \theta(\eta), x \xrightarrow{\sigma} \eta \in R\} \\ \theta &= [a(a'(\varphi)) := \langle a, a'\rangle(\varphi)]_{a \in Att_1, a' \in Att_2 \cup StAtt_2, \varphi \in \Pi} \end{aligned}$$

with $Att_1 = Syn_1 \cup Inh_1$, $Att_2 = Syn_2 \cup Inh_2$ and $StAtt_2 = StSyn_2 \cup StInh_2$. R_{dmy} is the set of dummy rules which gives

  - $\langle a, a'\rangle(\varphi) \xrightarrow{\sigma} \perp$ for any rule $a(a'(\varphi)) \xrightarrow{\sigma} \zeta \notin R$, $a \in Att_1$, $a' \in Att_2$, $\varphi \in \Pi$ and $\sigma \in \Sigma_1 \uplus \{\sharp_1\}$, and
  - $\langle a, \boldsymbol{a}'\rangle(\varphi) \xrightarrow{\sigma} Empty$ for any rule $a(\boldsymbol{a}'(\varphi)) \xrightarrow{\sigma} \zeta \notin R$, $a \in Att_1$, $\boldsymbol{a}' \in StAtt_2$, $\varphi \in \Pi$ and $\sigma \in \Sigma_1 \uplus \{\sharp_1\}$.

The calculation of $\widehat{ren}(U)$ is called *renaming*.

We give a partial example of our descriptional composition, where an ATT M_{itop} and an SATT M_{ptoi} are composed. First, $\widehat{proj}_{M_1}$ yields a set of the following attribute rules from an attribute rule (42) in M_2:

$$a_0(\boldsymbol{s}(\pi 1)) \overset{one}{\rightarrow} a_0(Cons(one, \boldsymbol{s}(\pi))) \tag{57}$$

$$a_1(Cons(one, \boldsymbol{s}(\pi))) \overset{one}{\rightarrow} a_1(\boldsymbol{s}(\pi 1)). \tag{58}$$

Next, $\widehat{se}_{M_1}$ yields a set of the following attribute rules from (58):

$$a_1(one) \overset{one}{\rightarrow} Head(a_1(\boldsymbol{s}(\pi 1))) \tag{59}$$

$$a_1(\boldsymbol{s}(\pi)) \overset{one}{\rightarrow} Tail(a_1(\boldsymbol{s}(\pi 1))), \tag{60}$$

where $a_1(one)$ is intended to be rewritten by the rule 1 of symbolic evaluation. Finally, $\widehat{ren}$ yields a following attribute rule from (60):

$$\langle a_1, \boldsymbol{s}\rangle(\pi) \overset{one}{\rightarrow} Tail(\langle a_1, \boldsymbol{s}\rangle(\pi 1)). \tag{61}$$

Figure 7 shows the final result M_{ptop} of the composition of M_{itop} and M_{ptoi}. Considering the roles of M_{itop} and M_{ptoi}, the role of M_{ptop} is expected to be that of the identical transformation mapping prefix representations onto prefix representations. In fact, M_{ptop} does not behave as an identical transformation for an input tree which is invalid as a prefix representation, such as $1, 2, \times, +$. That is because M_{ptoi} fails to return a tree representing an infix representation for such an invalid input.

We prove the correctness of the extended descriptional method, *i.e.,* $[\![M_1 \hat{\odot} M_2]\!] = [\![M_1]\!] \circ [\![M_2]\!]$ for an ATT M_1 and a SATT M_2. The proof is completed by simulating each step in Definition 4.9 with the corresponding step in Definition 4.2. First, we define the n-depth simulation for an intermediate result of projection or symbolic evaluation.

$M_{ptop} = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, \langle a_0, a_0\rangle, \sharp_2, R)$, where

- $Syn = \{\langle a_0, a_0\rangle\}$, $Inh = \{\langle a_1, a_0\rangle\}$,
- $StSyn = \{\langle a_1, \boldsymbol{s}\rangle\}$, $StInh = \{\langle a_0, \boldsymbol{s}\rangle\}$,
- $\Sigma = \Delta = \{one^{(1)}, two^{(1)}, plus^{(1)}, multi^{(1)}, end^{(0)}\}$,
- $R = \{$ $\langle a_0, a_0\rangle(\pi) \xrightarrow{\sharp_2} \langle a_0, a_0\rangle(\pi 1)$,

  $\langle a_1, a_0\rangle(\pi 1) \xrightarrow{\sharp_2} end$,

  $\langle a_0, \boldsymbol{s}\rangle(\pi 1) \xrightarrow{\sharp_2} Empty$,

  $\langle a_0, a_0\rangle(\pi) \xrightarrow{one} \langle a_0, a_0\rangle(\pi 1)$,

  $\langle a_1, a_0\rangle(\pi 1) \xrightarrow{one} \langle a_1, a_0\rangle(\pi)$,

  $\langle a_1, \boldsymbol{s}\rangle(\pi) \xrightarrow{one} Tail(\langle a_1, \boldsymbol{s}\rangle(\pi 1))$,

  $\langle a_0, \boldsymbol{s}\rangle(\pi 1) \xrightarrow{one} Cons(one(Head(\langle a_1, \boldsymbol{s}\rangle(\pi 1))), \langle a_0, \boldsymbol{s}\rangle(\pi))$

  $\langle a_0, a_0\rangle(\pi) \xrightarrow{two} \langle a_0, a_0\rangle(\pi 1)$,

  $\langle a_1, a_0\rangle(\pi 1) \xrightarrow{two} \langle a_1, a_0\rangle(\pi)$,

  $\langle a_1, \boldsymbol{s}\rangle(\pi) \xrightarrow{two} Tail(\langle a_1, \boldsymbol{s}\rangle(\pi 1))$,

  $\langle a_0, \boldsymbol{s}\rangle(\pi 1) \xrightarrow{two} Cons(two(Head(\langle a_1, \boldsymbol{s}\rangle(\pi 1))), \langle a_0, \boldsymbol{s}\rangle(\pi))$

  $\langle a_0, a_0\rangle(\pi) \xrightarrow{plus} \langle a_0, a_0\rangle(\pi 1)$,

  $\langle a_1, a_0\rangle(\pi 1) \xrightarrow{plus} \langle a_1, a_0\rangle(\pi)$,

  $\langle a_1, \boldsymbol{s}\rangle(\pi) \xrightarrow{plus} Cons(plus(Head(\langle a_1, \boldsymbol{s}\rangle(\pi 1))),$
  $Cons(Head(\langle a_0, \boldsymbol{s}\rangle(\pi)), Tail(\langle a_1, \boldsymbol{s}\rangle(\pi 1))))$,

  $\langle a_0, \boldsymbol{s}\rangle(\pi 1) \xrightarrow{plus} Cons(Head(Tail(\langle a_0, \boldsymbol{s}\rangle(\pi))), Tail(Tail(\langle a_0, \boldsymbol{s}\rangle(\pi))))$,

  $\langle a_0, a_0\rangle(\pi) \xrightarrow{multi} \langle a_0, a_0\rangle(\pi 1)$,

  $\langle a_1, a_0\rangle(\pi 1) \xrightarrow{multi} \langle a_1, a_0\rangle(\pi)$,

  $\langle a_1, \boldsymbol{s}\rangle(\pi) \xrightarrow{multi} Cons(multi(Head(\langle a_1, \boldsymbol{s}\rangle(\pi 1))),$
  $Cons(Head(\langle a_0, \boldsymbol{s}\rangle(\pi)), Tail(\langle a_1, \boldsymbol{s}\rangle(\pi 1))))$,

  $\langle a_0, \boldsymbol{s}\rangle(\pi 1) \xrightarrow{multi} Cons(Head(Tail(\langle a_0, \boldsymbol{s}\rangle(\pi))), Tail(Tail(\langle a_0, \boldsymbol{s}\rangle(\pi))))$,

  $\langle a_0, a_0\rangle(\pi) \xrightarrow{end} Head(\langle a_0, \boldsymbol{s}\rangle(\pi))$,

  $\langle a_1, \boldsymbol{s}\rangle(\pi) \xrightarrow{end} Cons(\langle a_1, a_0\rangle(\pi), Empty)$ $\}$

Figure 7: A SATT M_{ptop} obtained by composing an ATT M_{itop} and a SATT M_{ptoi}

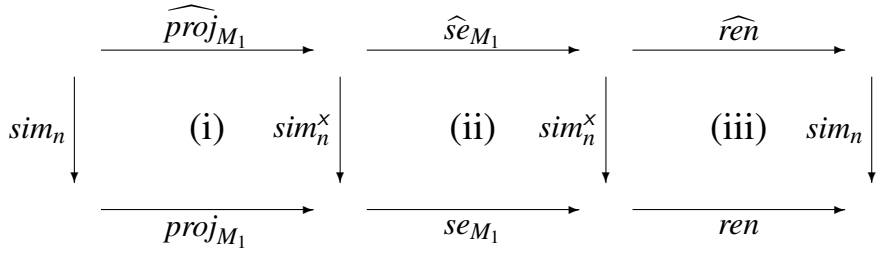

Figure 8: The extended descriptional composition and its simulation

Definition 4.10 Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, a, \sharp, R)$ be one of the intermediate results at a step of descriptional composition for an ATT $M_1 = (Syn_1, Inh_1, \Sigma_1, \Delta_1, a_1, \sharp_1, R_1)$ and a SATT $M_2 = (Syn_2, Inh_2, StSyn_2, StInh_2, \Sigma_2, \Delta_2, a_2, \sharp_2, R_2)$, $\Pi = \{\pi, \pi 1, \pi 2, \cdots\}$, $\mathcal{S}$ be a stack system (X_o, X_s, Δ_1) with

$$\begin{aligned} X_o &= \{a(\eta) \mid a \in Syn_1 \cup Inh_1, \eta \in RHS_o(M_2)\} \\ &\quad \cup \{a(\sharp_1(\eta)) \mid a \in Syn_1 \cup Inh_1, \eta \in RHS_o(M_2)\} \\ X_s &= \{a(\zeta) \mid a \in Syn_1 \cup Inh_1, \zeta \in RHS_s(M_2)\} \end{aligned}$$

and Γ_o, Γ_s be functions given by

$$\begin{aligned} \Gamma_o(a(\eta)) &= a(\alpha_o^{\mathcal{S}', \Gamma'_o, \Gamma'_s, n}(\eta)) \\ \Gamma_s(a(\zeta), i) &= \begin{cases} \bot & (\text{if } \alpha_s^{\mathcal{S}', \Gamma'_o, \Gamma'_s, n}(\zeta, i) = \bot) \\ a(\alpha_s^{\mathcal{S}', \Gamma'_o, \Gamma'_s, n}(\zeta, i)) & (\text{otherwise}) \end{cases} \end{aligned}$$

where $\mathcal{S}'$ is a stack system (X'_o, X'_s, Δ_2) with $X'_o = \{a(\varphi) \mid a \in Syn_2 \cup Inh_2, \varphi \in \Pi\}$, $X'_s = \{a(\varphi) \mid a \in StSyn_2 \cup StInh_2, \varphi \in \Pi\}$, and two functions Γ'_o and Γ'_s are given by $\Gamma'_o(a(\varphi)) = a(\varphi)$ and $\Gamma'_s(\boldsymbol{a}(\varphi), i) = \langle \boldsymbol{a}, i \rangle(\varphi)$ with $\varphi \in \Pi$. The n-depth simulation $sim_n^{\mathcal{X}}(M)$ is defined by the following septuple:

$$sim_n^{\mathcal{X}}(M) \stackrel{def}{=} (Syn', Inh', \Sigma, \Delta \cup \{\bot\}, a, \sharp, \alpha_{\mathcal{R}}^{\mathcal{S}, \Gamma_o, \Gamma_s, n}(R))$$

where $Syn' = Syn \cup \{\langle \boldsymbol{a}, i \rangle \mid \boldsymbol{a} \in StSyn, 1 \leq i \leq n\}$ and $Inh' = Inh \cup \{\langle \boldsymbol{a}, i \rangle \mid \boldsymbol{a} \in StInh, 1 \leq i \leq n\}$.

We prove that every diagram in Figure 8 commutes *i.e.,* each step in the extended descriptional composition is simulated by the corresponding step in the original descriptional composition.

Lemma 4.11 *Let M_1 be an ATT. The three functions $\widehat{proj}_{M_1}$, $\widehat{se}_{M_1}$ and $\widehat{ren}$ which are defined in* Definition 4.9 *satisfy the following equations:*

(i) $sim_n^{\mathcal{X}} \circ \widehat{proj}_{M_1} = proj_{M_1} \circ sim_n$

(ii) $sim_n^{\mathcal{X}} \circ \widehat{se}_{M_1} = se_{M_1} \circ sim_n^{\mathcal{X}}$

(iii) $sim_n \circ \widehat{ren} = ren \circ sim_n^{\mathcal{X}}$

Proof. Let $M_1 = (Syn_1, Inh_1, \Sigma_1, \Delta_1, a_1, \sharp_1, R_1)$ and $M_2 = (Syn_2, Inh_2, StSyn_2, StInh_2, \Sigma_2, \Delta_2, a_2, \sharp_2, R_2)$ be an ATT and a SATT with $\Delta_2 \subset \Sigma_1$. We use $\mathcal{S}$, Γ_o, Γ_s, $\mathcal{S}'$, Γ'_o and Γ'_s, each of which is as given in Definition 4.10, and functions α_o, α_s, $\alpha_{\mathcal{R}}$, α'_o and α'_s be respectively defined by $\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}$, $\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}$, $\alpha_{\mathcal{R}}^{\mathcal{S},\Gamma_o,\Gamma_s,n}$, $\alpha_o^{\mathcal{S}',\Gamma'_o,\Gamma'_s,n}$ and $\alpha_s^{\mathcal{S}',\Gamma'_o,\Gamma'_s,n}$.

(i) Neither $proj_{M_1}$ nor $\widehat{proj}_{M_1}$ change the sets of attributes. Each of the functions sim_n and $sim_n^{\times}$ changes the sets of attributes but they coincide. Hence $sim_n^{\times} \circ \widehat{proj}_{M_1}(M_2)$ and $proj_{M_1} \circ sim_n(M_2)$ have the same sets of attributes. It is enough to show that $sim_n^{\times} \circ \widehat{proj}_{M_1}(M_2)$ and $proj_{M_1} \circ sim_n(M_2)$ yield the same set of rules for every rule $x \xrightarrow{\gamma} \eta \in R_2$. If $\eta \in RHS_o(M_2) \cup \{\sharp_2(\eta') \mid \eta' \in RHS_o(M_2)\}$ and $\zeta \in RHS_s(M_2)$, from Definition 4.10, the following equations hold:

$$\begin{aligned} \alpha_o(a(\eta)) &= \Gamma_o(a(\eta)) \\ &= a(\alpha'_o(\eta)) \\ \alpha_s(a(\zeta), i) &= \Gamma_s(a(\zeta), i) \\ &= a(\alpha'_s(\zeta, i)). \end{aligned}$$

This implies that $sim_n^{\times} \circ \widehat{proj}_{M_1}(M_2)$ and $proj_{M_1} \circ sim_n(M_2)$ have the same set of rules.

(ii) Let $U = (Syn_2, Inh_2, StSyn_2, StInh_2, \Sigma_2, \Delta_2, a_2, \sharp_2, \bigcup_{\gamma \in \Sigma_2 \cup \{\sharp_2\}} R^{\gamma})$ be an intermediate result. It is enough to show that

$$\alpha_{\mathcal{R}}(\Phi) \Rightarrow_{SE}^{?} \alpha_{\mathcal{R}}(\Psi) \quad \text{if} \quad \Phi \Rightarrow_{\widehat{SE}} \Psi. \tag{62}$$

If the above clause holds, we have $\alpha_{\mathcal{R}}(R) \Rightarrow_{SE}^{*} \alpha_{\mathcal{R}}(nf(\Rightarrow_{\widehat{SE}}, R))$. Since $nf(\Rightarrow_{\widehat{SE}}, R)$ has no reducible form for $\Rightarrow_{SE}$ by the definition of $\Rightarrow_{\widehat{SE}}$ and $\alpha_{\mathcal{R}}$ yields no reducible form for $\Rightarrow_{SE}$, $\alpha_{\mathcal{R}}(nf(\Rightarrow_{\widehat{SE}}, R))$ is irreducible. Therefore, $nf(\Rightarrow_{SE}, \alpha_{\mathcal{R}}(R)) = \alpha_{\mathcal{R}}(nf(\Rightarrow_{\widehat{SE}}, R))$ holds. This means that every R^{γ} with $\gamma \in \Sigma_2 \cup \{\sharp_2\}$ yields the same set of rules in the computation of $sim_n^{\times} \circ \widehat{se}_{M_1}(U)$ and $se_{M_1} \circ sim_n^{\times}(U)$.

We prove (62) by case analysis on rewriting rules for $\Rightarrow_{\widehat{SE}}$ in Definition 4.9.

(CASE 1) Let R be the following set of rules:

$$\{b_i(\sigma(e_1, \cdots, e_n)) \xrightarrow{\gamma} \zeta_i \mid 1 \leq i \leq m\} \uplus R^{\gamma}_{misc}.$$

We have $R \Rightarrow_{\widehat{SE}} P \uplus Q$ where

$$\begin{aligned} P &= \left\{ b_i(e_j) \xrightarrow{\gamma} \theta(\eta) \;\middle|\; \begin{array}{l} b_i(\pi j) \xrightarrow{\sigma} \eta \in R_1^{\sigma}, \\ 1 \leq i \leq m, 1 \leq j \leq n \end{array} \right\} \\ Q &= \left\{ x \xrightarrow{\gamma} \rho^*(\eta) \;\middle|\; \begin{array}{l} x \xrightarrow{\gamma} \eta \in R^{\gamma}_{misc}, \\ \rho = [a_k(\sigma(e_1, \cdots, e_n)) := \theta(\psi_k)]_{1 \leq k \leq l}, \\ a_k(\pi) \xrightarrow{\sigma} \psi_k \in R_1^{\sigma}, 1 \leq k \leq l \end{array} \right\} \\ \theta &= [b_i(\pi) := \zeta_i]_{1 \leq i \leq m} [\pi j := e_j]_{1 \leq j \leq n} \end{aligned}$$

and we also have $\alpha_{\mathcal{R}}(R) \Rightarrow_{SE} P' \uplus Q'$ where

$$\begin{array}{rcl} P' & = & \left\{ b_i(\alpha_o(e_j)) \xrightarrow{\gamma} \theta'(\eta) \;\middle|\; \begin{array}{l} b_i(\pi j) \xrightarrow{\sigma} \eta \in R_1^{\sigma}, \\ 1 \le i \le m, 1 \le j \le n \end{array} \right\} \\ Q' & = & \left\{ x \xrightarrow{\gamma} \rho^*(\eta) \;\middle|\; \begin{array}{l} x \xrightarrow{\gamma} \eta \in \alpha_{\mathcal{R}}(R_{misc}^{\gamma}), \\ \rho = [a_k(\sigma(\alpha_o(e_1), \cdots, \alpha_o(e_n))) := \theta'(\psi_k)]_{1 \le k \le l}, \\ a_k(\pi) \xrightarrow{\sigma} \psi_k \in R_1^{\sigma}, 1 \le k \le l \end{array} \right\} \\ \theta' & = & [b_i(\pi) := \alpha_o(\zeta_i)]_{1 \le i \le m} [\pi j := \alpha_o(e_j)]_{1 \le j \le n}. \end{array}$$

We can show that $P' = \alpha_{\mathcal{R}}(P)$ and $Q' = \alpha_{\mathcal{R}}(Q)$. Hence $\alpha_{\mathcal{R}}(R) \Rightarrow_{SE} \alpha_{\mathcal{R}}(P \uplus Q)$.

(CASE 2) Let P be a set of attribute rules $\{x \xrightarrow{\gamma} \mathcal{E}[a(Head(\zeta))]\}$ with $\zeta \in RHS_s(M_2)$. Then we have $P \uplus R \Rightarrow_{\widehat{SE}} Q \uplus R$ where $Q = \{x \xrightarrow{\gamma} \mathcal{E}[Head(a(\zeta))]\}$. The following equations hold:

$$\begin{array}{rcl} \alpha_o(a(Head(\zeta))) & = & \Gamma_o(a(Head(\zeta))) \\ & = & a(\alpha'_s(\zeta, 1)) \\ \alpha_o(Head(a(\zeta))) & = & \alpha_s(a(\zeta), 1) \\ & = & \Gamma_s(a(\zeta), 1) \\ & = & a(\alpha'_s(\zeta, 1)). \end{array}$$

From Lemma 3.17, we obtain $\mathcal{E}[a(Head(\zeta))] =_{\alpha} \mathcal{E}[Head(a(\zeta))]$ where $=_{\alpha}$ is given in Lemma 3.17. This implies $\alpha_{\mathcal{R}}(P) = \alpha_{\mathcal{R}}(Q)$. Hence, $\alpha_{\mathcal{R}}(P \uplus R) = \alpha_{\mathcal{R}}(Q \uplus R)$ holds.

(CASE 3) Let P be a set of attribute rules $\{x \xrightarrow{\gamma} \mathcal{E}[a(Tail(\zeta))]\}$ with $\zeta \in RHS_s(M_2)$. Then we have $P \uplus R \Rightarrow_{\widehat{SE}} Q \uplus R$ where $Q = \{x \xrightarrow{\gamma} \mathcal{E}[Tail(a(\zeta))]\}$. The following equations hold:

$$\begin{array}{rcl} \alpha_s(a(Tail(\zeta)), i_0) & = & \left\{ \begin{array}{ll} \bot & (\text{if } \alpha'_s(Tail(\zeta), i_0) = \bot) \\ a(\alpha'_s(Tail(\zeta), i_0)) & (\text{otherwise}) \end{array} \right. \\ & = & \left\{ \begin{array}{ll} \bot & (\text{if } i_0 = n \text{ or } \alpha'_s(\zeta, i_0 + 1) = \bot) \\ a(\alpha'_s(\zeta, i_0 + 1)) & (\text{otherwise}) \end{array} \right. \\ \alpha_s(Tail(a(\zeta)), i_0) & = & \left\{ \begin{array}{ll} \bot & (\text{if } i_0 = n) \\ \alpha'_s(a(\zeta), i_0 + 1) & (\text{otherwise}) \end{array} \right. \\ & = & \left\{ \begin{array}{ll} \bot & (\text{if } i_0 = n \text{ or } \alpha'_s(\zeta, i_0 + 1) = \bot) \\ a(\alpha'_s(\zeta, i_0 + 1)) & (\text{otherwise}) \end{array} \right. \end{array}$$

for any $1 \le i_0 \le n$. From Lemma 3.17, $\mathcal{E}[a(Tail(\zeta))] =_{\alpha} \mathcal{E}[Tail(a(\zeta))]$ with $=_{\alpha}$ given in Lemma 3.17 holds. This implies $\alpha_{\mathcal{R}}(P) = \alpha_{\mathcal{R}}(Q)$. Hence, $\alpha_{\mathcal{R}}(P \uplus R) = \alpha_{\mathcal{R}}(Q \uplus R)$ holds.

(CASE 4) Similar to CASE 3.

(CASE 5) Similar to CASE 3.

(CASE 6) Consider the case 6 (i) in Definition 4.9. Let P be a set of attribute rules $\{a(\mathit{Head}(\zeta)) \xrightarrow{\gamma} \eta', a(\mathit{Tail}(\zeta)) \xrightarrow{\gamma} \zeta'\}$ with $\zeta \in \mathit{RHS}_s(M_2)$, $\eta' \in \mathit{BHS}_o(M_1,M_2)$ and $\zeta' \in \mathit{BHS}_s(M_1,M_2)$. We have $P \uplus R \Rightarrow_{\widehat{SE}} Q \uplus R$ where $Q = \{a(\zeta) \xrightarrow{\gamma} \mathit{Cons}(\eta',\zeta')\}$. Since $a(\mathit{Head}(\zeta)) \in \mathit{BHS}_o(M_1,M_2)$ and $a(\mathit{Tail}(\zeta)) \in \mathit{BHS}_s(M_1, M_2)$, we have

$$
\begin{array}{rcl}
\alpha_{\mathcal{R}}(P) & = & \{x \xrightarrow{\gamma} e \mid x = \alpha_o(a(\mathit{Head}(\zeta))) \neq \bot,\ e = \alpha_o(\eta')\} \\
& & \cup \{x_i \xrightarrow{\gamma} e_i \mid x_i = \alpha_s(a(\mathit{Tail}(\zeta)),i) \neq \bot,\ e_i = \alpha_s(\zeta',i),\ 1 \leq i \leq n\} \\
& = & \{x \xrightarrow{\gamma} e \mid x = a(\alpha_s(\zeta,1)),\ \alpha_s(\zeta,1) \neq \bot,\ e = \alpha_o(\eta')\} \\
& & \cup \left\{ x_i \xrightarrow{\gamma} e_i \;\middle|\; \begin{array}{l} x_i = a(\alpha_s(\zeta,i+1)),\ \alpha_s(\zeta,i+1) \neq \bot,\ e_i = \alpha_s(\zeta',i), \\ 1 \leq i \leq n-1 \end{array} \right\} \\
& & \qquad (\text{ from } \alpha_s(\mathit{Tail}(\zeta),n) = \bot.\ ) \\
\alpha_{\mathcal{R}}(Q) & = & \{x_i \xrightarrow{\gamma} e_i \mid x_i = \alpha_s(a(\zeta),i),\ e_i = \alpha_s(\mathit{Cons}(\eta',\zeta'),i),\ 1 \leq i \leq n\} \\
& = & \{x_1 \xrightarrow{\gamma} e_1 \mid x_1 = \alpha_s(a(\zeta),1) \neq \bot,\ e_1 = \alpha_o(\eta')\} \\
& & \cup \{x_i \xrightarrow{\gamma} e_i \mid x_i = \alpha_s(a(\zeta),i) \neq \bot,\ e_i = \alpha_s(\zeta',i-1),\ 2 \leq i \leq n\} \\
& = & \{x_1 \xrightarrow{\gamma} e_1 \mid x_1 = a(\alpha'_s(\zeta,1)),\ \alpha'_s(\zeta,1) \neq \bot,\ e_1 = \alpha'_o(\eta')\} \\
& & \cup \{x_i \xrightarrow{\gamma} e_i \mid x_i = a(\alpha'_s(\zeta,i)),\ \alpha'_s(\zeta,i) \neq \bot,\ e_i = \alpha'_s(\zeta',i-1),\ 2 \leq i \leq n\}
\end{array}
$$

Then $\alpha_{\mathcal{R}}(P) = \alpha_{\mathcal{R}}(Q)$ holds, hence $\alpha_{\mathcal{R}}(P \uplus R) = \alpha_{\mathcal{R}}(Q \uplus R)$. We can similarly show the statement in the cases 6 (ii), (iii).

(CASE 7) Similar to CASE 6.

(CASE 8) Let P be a set of attribute rules $\{a(\mathit{Empty}) \xrightarrow{\gamma} \zeta'\}$ with $\zeta' \in \mathit{BHS}_s(M_1,M_2)$. We have $P \uplus R \Rightarrow_{\widehat{SE}} R$. $\alpha_s(a(\mathit{Empty}),i) = \bot$ since $\alpha'_s(\mathit{Empty},i) = \bot$. This implies $\alpha_{\mathcal{R}}(P) = \emptyset$, hence $\alpha_{\mathcal{R}}(P \uplus R) = \alpha_{\mathcal{R}}(R)$.

(iii) Let $U = (\mathit{Syn}_2, \mathit{Inh}_2, \mathit{StSyn}_2, \mathit{StInh}_2, \Sigma_2, \Delta_2, a_2, \sharp_2, R)$ be an intermediate result after the symbolic evaluation, let Π be a set $\{\pi, \pi 1, \cdots, \pi m\}$ with $m = \max\{k \mid \Sigma_1^{(k)} \neq \emptyset\}$. and θ, θ' be replacements represented by

$$
\begin{array}{rcl}
\theta & = & [a(a'(\varphi)) := \langle a,a' \rangle(\varphi)]_{a \in \mathit{Att}_1, a' \in \mathit{Att}_2, \varphi \in \Pi} \\
\theta' & = & [a(a'(\varphi)) := \langle a,a' \rangle(\varphi)]_{a \in \mathit{Att}_1, a' \in \mathit{Att}_2 \cup \mathit{StAtt}_2, \varphi \in \Pi},
\end{array}
$$

respectively, where $\mathit{Att}_1 = \mathit{Syn}_1 \cup \mathit{Inh}_1$, $\mathit{Att}_2 = \mathit{Syn}_2 \cup \mathit{Inh}_2$ and $\mathit{StAtt}_2 = \mathit{StSyn}_2 \cup \mathit{StInh}_2$.

We prove the statement (iii) by showing that

$$
\begin{array}{rcll}
\alpha_o(\theta'(\eta)) & = & \theta(\alpha'_o(\eta)) & (63) \\
\alpha_s(\theta'(\zeta),i) & = & \theta(\alpha'_s(\zeta,i)) & (64)
\end{array}
$$

for any $\eta \in \mathit{BHS}_o$, $\zeta \in \mathit{BHS}_s$ and $1 \leq i \leq n$ where BHS_o and BHS_s are subsets of $\mathit{BHS}_o(M_1,M_2)$ and $\mathit{BHS}_s(M_1,M_2)$ such that their elements have no occurrence of the expression of the form $a(\sigma(\eta_1,\cdots,\eta_n))$, $a(\mathit{Head}(\zeta))$, $a(\mathit{Tail}(\zeta))$, $a(\mathit{Cons}(\eta,\zeta))$ or $a(\mathit{Empty})$. The equations (63) and (64) can be proved by induction on the structure of η or ζ, if (63) and (64) hold where $\eta = a(a'(\varphi))$ and $\zeta = a(\boldsymbol{a}'(\varphi))$ for $a \in \mathit{Att}_1$, $a' \in \mathit{Att}_2$, $\boldsymbol{a}' \in \mathit{StAtt}_2$ and $\varphi \in \Pi$. If $a \in \mathit{Att}_1$, $a' \in \mathit{Att}_2$

and $\varphi \in \Pi$, then $\alpha_o(a(a'(\varphi))) = a(a'(\varphi))$. Hence (63) holds. If $a \in Att_1$, $\boldsymbol{a}' \in StAtt_2$, $\varphi \in \Pi$ and i_0, then

$$\begin{aligned}\theta(\alpha'_s(a(\boldsymbol{a}'(\varphi)), i_0)) &= \theta(a(\alpha'_s(\boldsymbol{a}'(\varphi), i_0))) \\ &= \theta(a(\langle \boldsymbol{a}', i_0\rangle(\varphi))) \\ &= \langle a, \langle \boldsymbol{a}', i_0\rangle\rangle(\varphi) \\ &= \langle a, \boldsymbol{a}', i_0\rangle(\varphi)\end{aligned}$$

because $\alpha'_s(\boldsymbol{a}'(\varphi), i_0) = \langle \boldsymbol{a}', i_0\rangle(\varphi) \neq \perp$. From the definition of α_s,

$$\begin{aligned}\alpha_s(\theta'(a(\boldsymbol{a}'(\varphi))), i_0) &= \alpha_s(\langle a, \boldsymbol{a}'\rangle(\varphi), i_0) \\ &= \langle \langle a, \boldsymbol{a}'\rangle, i_0\rangle(\varphi) \\ &= \langle a, \boldsymbol{a}', i_0\rangle(\varphi).\end{aligned}$$

Hence (64) holds. (63) and (64) imply that $sim_n(\widehat{ren}(U))$ and $ren \circ sim_n^{\times}(U)$ have the same set of attribute rules because it is trivial that both sets of dummy rules produced by $sim_n \circ \widehat{ren}$ and $ren \circ sim_n^{\times}$ coincide. Therefore $sim_n \circ \widehat{ren}(U) = ren \circ sim_n^{\times}(U)$. □

It follows from this lemma that the n-depth simulation of the composition of an ATT M_1 and a SATT M_2 equals to the composition of an ATT M_1 and n-depth simulation of a SATT M_2.

Proposition 4.12 *Let M_1 and M_2 be a sur-ATT a sur-SATT, respectively. Then*

$$[\![M_1 \odot sim_n(M_2)]\!] = [\![sim_n(M_1 \hat{\odot} M_2)]\!]$$

holds for any n.

Proof.

$$\begin{aligned}M_1 \odot sim_n(M_2) &= ren \circ se_{M_1} \circ proj_{M_1} \circ sim_n(M_2) && \text{(by Definition 4.2)} \\ &= sim_n \circ \widehat{ren} \circ \widehat{se}_{M_1} \circ \widehat{proj}_{M_1}(M_2) && \text{(by Lemma 4.11)} \\ &= sim_n(M_1 \hat{\odot} M_2) && \text{(by Definition 4.9)}\end{aligned}$$

□

The correctness of the extended descriptional composition is an immediate consequence of this proposition.

Theorem 4.13 ($\hat{\odot}$-Correctness, I) *If M_1 is a well-defined sur-ATT and M_2 is and a well-defined sur-SATT, then $M_1 \hat{\odot} M_2$ is a well-defined sur-SATT such that $[\![M_1]\!] \circ [\![M_2]\!] = [\![M_1 \hat{\odot} M_2]\!]$, i.e., $[\![M_1]\!] \circ [\![M_2]\!](t) = [\![M_1 \hat{\odot} M_2]\!](t)$ for any input t.*

Proof. Assume that t is an arbitrary input tree in $dom([\![M_2]\!])$, M_1 is a well-defined sur-ATT and M_2 is a well-defined sur-SATT. Then $sim_n(M_2)$ is a sur-ATT for any n by Definition 4.7.

Letting $n = \max\{lub(M_2,t), lub(M_1 \hat{\odot} M_2,t)\}$, we have

$$\begin{array}{rcll}
[\![M_1 \hat{\odot} M_2]\!](t) & = & [\![sim_n(M_1 \hat{\odot} M_2)]\!](t) & \text{(by Theorem 3.22)} \\
& = & [\![M_1 \odot sim_n(M_2)]\!](t) & \text{(by Proposition 4.12)} \\
& = & [\![M_1]\!] \circ [\![sim_n(M_2)]\!](t) & \text{(by Theorem 4.3)} \\
& = & [\![M_1]\!]([\![sim_n(M_2)]\!](t)) & \text{(by the definition of } \circ) \\
& = & [\![M_1]\!]([\![M_2]\!](t)) & \text{(by Theorem 3.22)} \\
& = & [\![M_1]\!] \circ [\![M_2]\!](t) & \text{(by the definition of } \circ)
\end{array}$$

It follows that $M_1 \odot sim_n(M_2)$ is a well-defined sur-ATT for any n from Theorem 4.3. Hence, $sim_n(M_1 \hat{\odot} M_2)$ is a well-defined sur-ATT for any n. Therefore $M_1 \hat{\odot} M_2$ is a well-defined sur-SATT from Definition 4.7 and Corollary 3.23. □

The transition by equations in the above proof does not depend on the sur-condition of M_1 and M_2. Therefore we obtain the following theorem by using Theorem 4.5 instead of Theorem 4.3.

Theorem 4.14 ($\hat{\odot}$-Correctness, II) *If M_1 is a TDTT and M_2 is a well-defined SATT, then $M_1 \hat{\odot} M_2$ is a well-defined SATT such that $[\![M_1]\!] \circ [\![M_2]\!] = [\![M_1 \hat{\odot} M_2]\!]$.*

The closure properties are corollaries to these theorems.

Corollary 4.15

(i) $\mathit{ATT}_{su} \circ \mathit{SATT}_{su} = \mathit{SATT}_{su}$.

(ii) $\mathit{TDTT} \circ \mathit{SATT} = \mathit{SATT}$.

Proof. The first statement follows immediately from the fact that ATT_{su} contains the identical tree transformation and that $\mathit{ATT}_{su} \circ \mathit{SATT}_{su} \subseteq \mathit{SATT}_{su}$ holds from Theorem 4.13. Similarly, the second statement follows from Theorem 4.14. □

5 Conclusion

We have presented a composition method for stack-attributed tree transducers (satt) and proved the correctness of it. Stack-attributed tree transducers are more powerful than attributed tree transducers (att) due to a stack mechanism. Our composition method is based upon the fact that stack-attributed tree transducers are approximated by attributed tree transducers once an input tree is fixed.

We proved also that the composition method enjoys a closure property under the restriction called single-use restriction (sur). This indicates that the composition of a sur-att and a sur-satt results in a single sur-satt, which can be subject to composition with another sur-att.

This paper has only dealt with the composition of an att and an satt. We believe that the result of the composition of two satt's cannot be obtained by a single satt. Instead, we would obtain an attributed tree transducers whose attribute values have nested stack structures, *i.e.,* stack of stacks. It would be interesting to compare the result with the composition of other tree transducers: macro tree transducer [EV85], macro attributed tree transducer [KV94], n-iterated pushdown tree transducer [EV88], et al.

Acknowledgment

The author wishes to express his gratitude to Assoc. Prof. Susumu Nishimura for his generous and patient guidance. Further, he is grateful to Dr. Joost Engelfriet and anonymous reviewers for quite constructive comments. He is also grateful to Shin-Cheng Mu for his kind help and advice on the manuscript. He also thanks Prof. Masato Takeichi, Assoc. Prof. Zhenjiang Hu and Assoc.Prof. Masahito Hasegawa for their encouragement and support.

References

[ASU86] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers — Principles, Techniques, and Tools*. Addison-Wesley, 1986.

[AU73] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation and Compiling*. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[CDPR99] Loïc Correnson, Etienne Duris, Didier Parigot, and Gilles Roussel. Declarative program transformation: A deforestation case-study. In *Principles and Practice of Declarative Programming*, volume 1702 of *LNCS*, pages 360–377. Springer Verlag, 1999.

[Eng80] Joost Engelfriet. Some open questions and recent results on tree transducers and tree languages. In *Ronald V. Book, Formal Language Theory: Perspectives and Open Problems, Academic Press*. Academic Press, 1980.

[EV85] Joost Engelfriet and Heiko Vogler. Macro tree transducers. *Journal of Computer and System Sciences*, 31(1):71–146, August 1985.

[EV88] Joost Engelfriet and Heiko Vogler. High level tree transducers and iterated pushdown tree transducers. *Acta Informatica*, 26(1–2):131–192, October 1988.

[Far84] Rodney Farrow. Generating a production compiler from an attribute grammar. *IEEE Software*, 1(4):77–93, October 1984.

[Fül81] Z. Fülöp. On attributed tree transducers. *Acta Cybernetica*, 5:261–280, 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] Harald Ganzinger. Increasing modularity and language-independency in automatically generated compilers. *Science of Computer Programming*, 3(3):223–278, 1983.

[GG84] Harald Ganzinger and Robert Giegerich. Attribute coupled grammars. In *Proceedings of the ACM SIGPLAN '84 Symposium on Compiler Construction*, volume 19(6) of *SIGPLAN Notices*, pages 157–170, June 1984.

[Gie88] Robert Giegerich. Composition and evaluation of attribute coupled grammars. *Acta Informatica*, 25(4):355–423, May 1988.

[HT85] Susan Horwitz and Tim Teitelbaum. Relations and attributes: a symbiotic basis for editing environments. In *ACM SIGPLAN '85 Symp. on Language Issues in Programming Environments*, pages 93–106. ACM press, Seattle, WA, June 1985.

[Joh87] Thomas Johnsson. Attribute grammars as a functional programming paradigm. In Gilles Kahn, editor, *Proceedings of the Conference on Functional Programming Languages and Computer Architecture*, volume 274 of *LNCS*, pages 154–173. Springer Verlag, 1987.

[KHZ82] U. Kastens, B. Hutt, and E. Zimmermann. *GAG: A Practical Compiler Generator*. Number 141 in Lecture Notes in Computer Science. Springer Verlag, 1982.

[Knu68] Donald E. Knuth. Semantics of context-free languages. *Mathematical Systems Theory*, 2(2):127–145, 1968.

[Knu71] Donald E. Knuth. Correction: Semantics of context-free languages. *Mathematical Systems Theory*, 5(1):95–96, 1971.

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

[Küh98] A. Kühnemann. Benefits of tree transducers for optimizing functional programs. *Lecture Notes in Computer Science*, 1530:146–157, 1998.

[KV94] Armin Kühnemann and Heiko Vogler. Synthesized and inherited functions. A new computational model for syntax-directed semantics. *Acta Informatica*, 31(5):431–477, 1994.

[Nak04] Keisuke Nakano. XTiSP: XML transformation language intended for stream processing, 2004. Unpublished manuscript.

[NN01] Keisuke Nakano and Susumu Nishimura. Deriving event-based document transformers from tree-based specifications. In *LDTA'2001 Workshop on Language Descriptions, Tools and Applications*, volume 44(2) of *Electronic Notes in Theoretical Computer Science*. Elsevier Science, 2001. available on-line: `http://www.elsevier.nl/gej-ng/31/29/23/73/27/show/Products/notes/index.htt`.

[Rep84] T. Reps. *Generating Language-based Environments*. MIT Press, Cambridge, Ma, 1984.

[RM89] P. Rechenberg and H. Moessenboeck. *A compiler generator for microcomputers*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[Rou70] William C. Rounds. Mappings and grammars on trees. *Mathematical Systems Theory*, 4(3):257–287, 1970.

[VK04] Janis Voigtländer and Armin Kühnemann. Composition of functions with accumulating parameters. *Journal of Functional Programming*, 14:317–363, 2004.

[W3C] Extensible markup language (XML). `http://www.w3c.org/XML/`.