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@ipl.t.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] as a way of describing semantics of context-free languages. They specify a 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 compiler construction[ASU86, Far84, RM89, KHZ82], editing environments[Rep84, HT85] and program transformation[Joh87, CDPR99].

Composition of attribute grammars is one of the well-studied issue 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 a so-called *single-use condition*. Whereas it has been developed for composing compiler components, the composition method makes a general contribution. For example, Kühnemann[Küh98] and Correnson et al.[CDPR99] independently applied the descriptional composition to functional program transformation by utilizing the fact that attribute grammars have a close connection 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 a transformation from the postfix representation of numerical formulae, *e.g.* $2,1,2,+,\times$ in Figure 1(a), to the infix representation, *e.g.* $2\times(1+2)$ in Figure 1(b). This transformation requires a pushdown stack device in which subtrees of the output tree are stored. Figure 2 gives a definition for postfix-to-infix transformation in attribute grammar. The attribute $\boldsymbol{s}$ receives stack values. We write $[\,]$ for an empty stack,

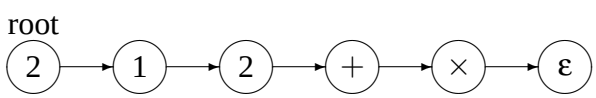

Figure 1(a): Postfix representation

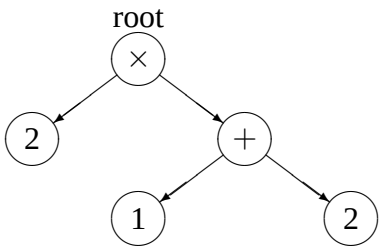

Figure 1(b): Infix representation

Figure 1: Two kinds of representations of numerical formulae

$e_1 :: e_2$ for a stack obtained by adding a value e_1 to a stack value e_2, $hd(e)$ for the top element of a stack e and $tl(e)$ for a stack e without the top element. The existing composition method cannot deal with such attribute grammars that require a stack mechanism.

We present a new composition method for attribute grammars with a pushdown stack device. The method is formalized in a framework of attributed tree transducers[Fül81], which is one of the formal computational models of attribute grammars. In this paper, we introduce *stack-attributed tree transducers* by extending attributed tree transducers with a stack device and propose a composition method for stack-attributed tree transducers that extends the descriptional composition method. Stack-attributed tree transducers are more powerful than attributed tree transducers because of the pushdown storages. This relation is similar to that between pushdown automata and finite state automata: A class of languages accepted by the formers, context-free languages, is larger than a class of languages accepted by the latters, regular languages[AU73].

This paper contains two main results with respect to our composition method for stack-attributed tree transducers. First, we prove the correctness of our composition method by utilizing the fact that a stack-attributed tree transducer can be simulated by an attributed tree transducer once an input tree is fixed. The correctness of a composition method for stack-attributed tree transducers is reduced to that for attributed tree transducers which has been proved by [Gie88]. Second, we prove the *leftward closure property* of the composition under a certain condition. A set F of tree transformations satisfies the leftward closure property for G if $G \circ F = F$ where $G \circ F = \{h \mid h(x) = g(f(x)), g \in G, f \in F\}$ (in contrast, F satisfies the full closure property if $F \circ F = F$). This paper shows that the set of stack-attributed tree transducers satisfies the leftward closure property for the set of attributed tree transducers. A stack-attributed tree transducer can be composed with an attributed tree transducer by our composition method. Then we obtain a single stack-attributed tree transducer, which is ready to be composed with another

$S \to T$	$S.a_0 = T.a_0$	$T \to +, T_1$	$T.a_0 = T_1.a_0$
	$T.\boldsymbol{s} = [\,]$		$T_1.\boldsymbol{s} = (hd(tl(T.\boldsymbol{s})) + hd(T.\boldsymbol{s})) :: (tl(tl(T.\boldsymbol{s})))$
$T \to 1, T_1$	$T.a_0 = T_1.a_0$	$T \to \times, T_1$	$T.a_0 = T_1.a_0$
	$T_1.\boldsymbol{s} = 1 :: T.\boldsymbol{s}$		$T_1.\boldsymbol{s} = (hd(tl(T.\boldsymbol{s})) \times hd(T.\boldsymbol{s})) :: (tl(tl(T.\boldsymbol{s})))$
$T \to 2, T_1$	$T.a_0 = T_1.a_0$	$T \to \varepsilon$	$T.a_0 = hd(T.\boldsymbol{s})$
	$T_1.\boldsymbol{s} = 2 :: T.\boldsymbol{s}$		

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

$$TDTT \subsetneq ATT \quad (1)$$
$$ATT \subsetneq MTT \quad (2)$$
$$MTT_{wsu} \subsetneq ATT \quad (3)$$
$$ATT \subsetneq SATT \quad (4)$$

$$TDTT \circ TDTT = TDTT \quad (5)$$
$$ATT_{su} \circ ATT_{su} = ATT_{su} \quad (6)$$
$$TDTT \circ ATT = ATT \quad (7)$$
$$TDTT \circ MTT = MTT \quad (8)$$
$$MTT_{wsu} \circ MTT_{nc} = MTT \quad (9)$$
$$ATT_{su} \circ SATT_{su} = SATT_{su} \quad (10)$$
$$TDTT \circ SATT = SATT \quad (11)$$

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

attributed tree transducer again.

Our composition method is proposed not only for theoretical interest. The composition method for stack-attributed tree transducers is useful in practice for transformations over structured documents such as XML[W3C]. Nakano and Nishimura[NN01] utilized the descriptional composition method to generate stream-to-stream document transformations from tree-to-tree document transformations. Given a tree-to-tree document transformation T, a stream-to-stream document transformation can be obtained by a composition of three transformations *Unparse*, T and *Parse*: *Unparse* is an unparsing(tree-to-stream) transformation; *Parse* is a parsing(stream-to-tree) transformation. They defined each transformation by an attribute grammar and applied the descriptional composition method. In their framework, they faced the problem that parsing transformation requires a stack mechanism, which cannot be dealt with by the existing descriptional composition. They circumvent the problem by restricting the maximum nesting depth of input documents to a fixed number and by simulating the stack device with a finite set of attributes. Our composition method provides a general solution of the above problem without the work around. In this application, the leftward closure property of our composition method plays an important role. Assume that a tree-to-tree transformation T is given by an attributed-tree transducer. Since *Parse* is defined by a stack-attributed tree transducer, the composition of T and *Parse* is obtained as a stack-attributed tree transducer from the leftward closure property. Since *Unparse* is defined by an attributed tree transducer, the composition of *Unparse* and the previous result is a stack-attributed tree transducer. Consequently, we obtain a stream-to-stream transformation as a stack-attributed tree transducer.

There are a number of researches on tree transducer composition methods. Figure 3 shows the inclusion relations of various classes of tree transformations and their compositions. We write $TDTT$, ATT, MTT and $SATT$ to represent top-down tree transducers[Rou70], attributed tree transducers[Fül81], macro tree transducers[EV85] and stack-attributed tree transducers, respectively. The subscript *su*, *wsu* and *nc* indicate subsets of tree transducers restricted by the conditions *single-use*[Gie88], *weakly single-use* and *non-copying*[Küh98], respectively. The relations (1) and (7) are shown in [Fül81]; (2) and (8), in [EV85]; (3), in [Küh98]; (5), in [Rou70]; (6), in [GG84, Gie88]; (9), in [Voi01]. The relations (4), (10) and (11), involving $SATT$, are shown in this paper.

Our results (10) and (11) make an important contribution in the sense that $SATT$ is one of the largest sets that enjoys a closure property in some sense. The equations (5) and (6) indicate the full closure property. However, $TDTT$ and ATT_{su} are not expressive enough. The equations (7)

and (8) indicate the leftward closure property, as well as our result (10). However, they provide only the composition with $TDTT$. As for the composition with $TDTT$, we can show (11) indicating that $SATT$ also satisfies the leftward closure property. Although the composition in (9) may possibly provide more powerful composition than (10), this composition is not closed for neither MTT_{su} nor MTT_{wsu}.

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 under a certain restriction. Section 4 introduces a composition method for attributed tree transducers, presents a 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 $\mathbb{N}$ and the set of positive integers by $\mathbb{N}_+$. 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 a set of finite strings over a set P of symbols by P^*. A null string is denoted by ε.

A designated symbol $\perp$ means the undefined value. A function f from a set P to a set Q is denoted by $f : P \to Q$. Then $dom(f)$ and $range(f)$ denote the *domain* of f and the *range* of f, respectively, such that $dom(f) = \{x \in P \mid f(x) \neq \perp\}$ and $range(f) = \{f(x) \in Q \mid x \in dom(f)\}$. A function $f \circ g$ represents a *composition* of two functions f and g where $f \circ g(x) = f(g(x))$ for any $x \in dom(g)$. Let F and G be sets of functions. 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 $a_i \Rightarrow a_{i+1} (1 \leq i \leq n)$ for some $a_1, \cdots, a_{n+1} \in A$. 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 $m \geq 0$, $m \geq 1$ and $m \in \{0,1\}$, respectively. $a \in A$ 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 b is a *normal form* of a and write $nf(\Rightarrow, a)$ for b.

A *ranked set* Σ is a set in which every symbol is associated with a non-negative integer called *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 $\sigma \in \Sigma^{(n)}$. The designated symbol $\perp$ is of rank 0. Let Σ be a ranked alphabet and A be a set of variables disjoint with Σ. The set of Σ-*labeled trees indexed by* A, denoted by $T_\Sigma(A)$ (or T_Σ, if A is empty), is the smallest set T satisfying

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

We denote by $t[x/s]$ the *substitution* of occurrences of a variable x by s. Let $t, s_1, \cdots, s_n, u_1, \cdots, u_n$ be trees in $T_\Sigma(X)$ such that every $u_i (1 \leq i \leq n)$ is a subtree of t, provided that u_i is not a subtree of u_j for any i and j with $i \neq j$. The tree $t[u_1, \cdots, u_n := s_1, \cdots, s_n]$ or $t[u_i := s_i]_{1 \leq i \leq n}$

is obtained from t by simultaneously *replacing* every subtree at the points of occurrences of $u_1, \cdots, u_n$ by the trees $s_1, \cdots, s_n \in T_\Sigma(X)$, respectively. If $\rho = [u_1, \cdots, u_n := s_1, \cdots, s_n]$, then we may write $\rho(t)$ for $t[u_1, \cdots, u_n := s_1, \cdots, s_n]$. A $T_\Sigma(X)$-*context* $\mathcal{E}$ (or *context*, simply) is an element of $T_\Sigma(X \uplus \{\bullet\})$ where a symbol $\bullet$, called *hole*, occurs exactly once in $\mathcal{E}$. A tree $\mathcal{E}[t]$ with $t \in T_\Sigma(X)$ stands for $\mathcal{E}[\bullet/t] \in T_\Sigma(X)$. We also say U-context for a context $\mathcal{E}$ such that $\mathcal{E}[u] \in U$ for some $u \in U$. Let ρ be a replacement $[u_i := s_i]_{1 \leq i \leq n}$. Then the replacement ρ^* is defined by $nf(\Rightarrow_\rho, \rho)$ where $\Rightarrow_\rho$ is the binary relation such that $[u_i := t_i]_{1 \leq i \leq n} \Rightarrow_\rho [u_i := \rho(t_i)]_{1 \leq i \leq n}$ only if $t_i \neq \rho(t_i)$ for some $1 \leq i \leq n$.

The prefix-closed set of all *paths* of t, denoted by $path(t) (\subseteq \mathbb{N}^*)$, is defined by $path(\sigma(t_1, \cdots, t_k)) = \{\varepsilon\} \cup \{iw \mid 1 \leq i \leq k, w \in path(t_i)\}$ if $\sigma \in \Sigma^{(k)}$. Note that $path(\sigma^{(0)}) = \{\varepsilon\}$. Every path $w \in path(t)$ refers to a corresponding label of t, denoted by $label_t(w)$, which is defined by $label_{\sigma(t_1, \cdots, t_n)}(\varepsilon) = \sigma$ and $label_{\sigma(t_1, \cdots, t_n)}(iw) = label_{t_i}(w)$ for every $1 \leq i \leq n$ and $w \in path(t_i)$.

3 Stack-Attributed Tree Transducers

In this section, we give the definition of attributed tree transducers(ATT) and stack-attributed tree transducers(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 [FV98].

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 unary ranked alphabets with $Syn \cap Inh = \emptyset$ and $k \in \mathbb{N}$. The set $RHS(Syn, Inh, \Delta, k)$ of *right-hand sides* over *Syn*, *Inh* and Δ is given by the smallest subset *RHS* of $T_{\Delta \cup Syn \cup Inh}(\{\pi, \pi 1, \cdots, \pi k\})$ which 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 $\eta_1, \cdots, \eta_l \in RHS$, the term $\delta(\eta_1, \cdots, \eta_l)$ is in *RHS*,

where $\pi, \pi 1, \cdots, \pi k$ are called *path variables*. □

Definition 3.2 An *attributed tree transducer*(ATT) is a septuple $M = (Syn, Inh, \Sigma, \Delta, a_0, \sharp, R)$ where

- *Syn* and *Inh* are unary ranked alphabets satisfying $Syn \cap Inh = \emptyset$, of which the elements are called *synthesized attributes* and *inherited attributes*, respectively,
- Σ and Δ are ranked alphabets with $(Syn \cup Inh) \cap (\Sigma \cup \Delta) = \emptyset$, called the *input alphabet* and the *output alphabet*, respectively,
- $a_0 \in Syn$ is a designated attribute, called the *initial attribute*,
- $\sharp \notin \Sigma$ is an unary ranked symbol, called the *initial symbol*,

- R is a set of *attribute rules* such that $R = \bigcup_{\sigma \in \Sigma \uplus \{\sharp\}} R^{\sigma}$ 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 $1 \leq i \leq k$ with $k = rank(\sigma)$, 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 any term such that $\eta_{\sigma} \in RHS(Syn, Inh, \Delta, rank(\sigma)) \cup \{\bot\}$ and $\eta_{\sharp}$ is any term such that $\eta_{\sharp} \in RHS(Syn, \emptyset, \Delta, 1) \cup \{\bot\}$.

□

Our definition is slightly different from [FV98] in the way of giving the specification of an ATT. e use no environment describing values of inherited attributes of the root node of an input tree. We define values of inherited attributes in attribute rules for the initial symbol instead of the environment.

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 $P \to \sigma\ P_1\ \cdots\ P_k$, in classical representation of attribute grammars where P is a nonterminal symbol. 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 $P \to \sigma\ P_1\ P_2$. We give an example of an ATT and an attribute grammar counterpart of the ATT.

Example 3.3 We give an 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 integers 1 and 2 by using two binary operators $+$ and $\times$. The infix representation is specified by a binary tree in which nodes and leaves are numerical operators and numbers, respectively. The postfix representation is specified by a monadic tree whose element is either a numerical operator or a number. An 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 a set of the following attribute rules:

$$a_0(\pi) \xrightarrow{\sharp} a_0(\pi 1) \tag{12}$$
$$a_1(\pi 1) \xrightarrow{\sharp} \mathit{end} \tag{13}$$
$$a_0(\pi) \xrightarrow{\mathit{one}} \mathit{one}(a_1(\pi)) \tag{14}$$
$$a_0(\pi) \xrightarrow{\mathit{two}} \mathit{two}(a_1(\pi)) \tag{15}$$
$$a_0(\pi) \xrightarrow{\mathit{plus}} a_0(\pi 1) \tag{16}$$
$$a_1(\pi 1) \xrightarrow{\mathit{plus}} a_0(\pi 2) \tag{17}$$
$$a_1(\pi 2) \xrightarrow{\mathit{plus}} \mathit{plus}(a_1(\pi)) \tag{18}$$
$$a_0(\pi) \xrightarrow{\mathit{multi}} a_0(\pi 1) \tag{19}$$
$$a_1(\pi 1) \xrightarrow{\mathit{multi}} a_0(\pi 2) \tag{20}$$
$$a_1(\pi 2) \xrightarrow{\mathit{multi}} \mathit{multi}(a_1(\pi)) \tag{21}$$

Figure 4 shows an attribute grammar which is counterpart of the ATT M_{itop}. The rules (12) and (13), namely $R^{\sharp}$, correspond to a 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 attribute rules for the production rules $T \to 1$, $T \to 2$, $T \to T_1 + T_2$ and $T \to T_1 \times T_2$, 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 $M = (\mathit{Syn}, \mathit{Inh}, \Sigma, \Delta, a_0, \sharp, R)$ be an ATT, $t \in T_\Sigma$ and η_w stand for a tree obtained by replacing every occurrence of π with a path $w \in \mathit{path}(\sharp(t))$ in η. The *derivation relation induced by M on t* is a binary relation $\Rightarrow_{M,t}$ over $T_\Delta(\{a(w) \mid a \in \mathit{Syn} \cup \mathit{Inh}, w \in \mathit{path}(\sharp(t))\})$ defined by:

- $a(w) \Rightarrow_{M,t} \eta_w$ where $a \in \mathit{Syn}$, $a(\pi) \xrightarrow{\sigma} \eta \in R$ and $\sigma = \mathit{label}_{\sharp(t)}(w)$.
- $b(wi) \Rightarrow_{M,t} \eta_w$ where $b \in \mathit{Inh}$, $b(\pi i) \xrightarrow{\sigma} \eta \in R$ and $\sigma = \mathit{label}_{\sharp(t)}(w)$.
- $\delta(\eta_1, \cdots, \eta_i, \cdots, \eta_n) \Rightarrow_{M,t} \delta(\eta_1, \cdots, \eta_i', \cdots, \eta_n)$ where $\eta_1, \cdots, \eta_{i-1}$ are irreducible and $nf(\Rightarrow_{M,t}, \eta_i) = \eta_i' (\neq \eta_i, \bot)$.
- $\delta(\eta_1, \cdots, \eta_i, \cdots, \eta_n) \Rightarrow_{M,t} \bot$ where $\eta_1, \cdots, \eta_{i-1}$ are irreducible and $nf(\Rightarrow_{M,t}, \eta_i) = \bot$.

						T	$\to$	$T_1 + T_2$	$T.a_0$	$=$	$T_1.a_0$
S	$\to$	T	$S.a_0$	$=$	$T.a_0$				$T_1.a_1$	$=$	$T_2.a_0$
			$T.a_1$	$=$	ε				$T_2.a_1$	$=$	$+, T.a_1$
T	$\to$	1	$T.a_0$	$=$	$1, T.a_1$	T	$\to$	$T_1 \times T_2$	$T.a_0$	$=$	$T_1.a_0$
T	$\to$	2	$T.a_0$	$=$	$2, T.a_1$				$T_1.a_1$	$=$	$T_2.a_0$
									$T_2.a_1$	$=$	$\times, T.a_1$

Figure 4: An attribute grammar counterpart of Example 3.3

It is obvious that this reduction system is determinant. Note that we need to compute $nf(\Rightarrow_{M,t}, \eta_i)$ in the above definition. In this paper, we do not consider derivation relations in the case where $nf(\Rightarrow_{M,t}, \eta_i)$ cannot be computed.

An *attribute value* of $a(w)$ for an ATT M and an input tree t is defined by $nf(\Rightarrow_{M,t}, a(w))$, if any, where $w \in path(\sharp(t))$. We may omit subscripts M and t of $\Rightarrow$ when it is clear from the context. Note that $nf(\Rightarrow_{M,t}, a(w))$ is a tree over the output alphabet Δ of M. This is because any term having occurrences of terms of the form $b(v)$ for $b \in Inh \cup Syn$ and a path v is always reducible. □

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

Definition 3.5 Let $M = (Syn, Inh, \Sigma, \Delta, a_0, \sharp, R)$ be an ATT. We define the *semantics* of an ATT M by a function $[\![M]\!] : T_\Sigma \to T_\Delta$ such that $[\![M]\!](t) \stackrel{def}{=} nf(\Rightarrow_{M,t}, a_0(\varepsilon))$ if there exists $nf(\Rightarrow_{M,t}, a_0(\varepsilon))$. The ATT M is *well-defined* if $[\![M]\!](t)$ can be defined for every $t \in T_\Sigma$. □

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

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

by the following reduction steps:

$$\begin{aligned}
a_0(\varepsilon) &\Rightarrow^{\sharp,\varepsilon} a_0(1) && (22)\\
&\Rightarrow^{multi,1} a_0(11) && (23)\\
&\Rightarrow^{two,11} two(a_1(11)) && (24)\\
&\Rightarrow^{multi,1} two(a_0(12)) && (25)\\
&\Rightarrow^{plus,12} two(a_0(121)) && (26)\\
&\Rightarrow^{one,121} two(one(a_1(121))) && (27)\\
&\Rightarrow^{plus,12} two(one(a_0(122))) && (28)\\
&\Rightarrow^{two,122} two(one(two(a_1(122)))) && (29)\\
&\Rightarrow^{plus,12} two(one(two(plus(a_1(12))))) && (30)\\
&\Rightarrow^{multi,1} two(one(two(plus(multi(a_1(1)))))) && (31)\\
&\Rightarrow^{\sharp,\varepsilon} two(one(two(plus(multi(end))))) && (32)
\end{aligned}$$

where the superscripts of $\Rightarrow^{\sigma,w}$ indicate that the derivation relation is based on a σ-rule at the path w. For example, a reduction step $\Rightarrow^{plus,12}$ at (30) is derived by replacing π with a path 12 in $\sharp(t)$ in a *plus*-rule at (18). □

Let M be an ATT. M is called a *top-down tree transducer*(TDTT) if there is no inherited attribute in M. We write ATT and $TDTT$ for sets of tree transformers such that $ATT = \{[\![M]\!] \mid M \text{ is a well-defined ATT.}\}$ and $TDTT = \{[\![M]\!] \mid M \text{ is a TDTT.}\}$. It is well-known that $TDTT \subsetneq ATT$ holds[Fül81].

3.2 Stack-Attributed Tree Transducers

Following the definition of ATTs, we give the definition of SATTs. SATTs extend ATTs with a new class of attributes, called *stack attributes*. There are two types of attributes in a SATT, output attributes and stack attributes. Output attributes receive trees over the output alphabet as well as attributes in ATTs. Stack attributes receive pushdown stacks of trees over the output alphabet. The following definition introduces a 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 and Δ is a ranked alphabet. For a stack system $\mathcal{S}$, we define a set $EXP_o(\mathcal{S})$ of *output expressions*, a set $EXP_s(\mathcal{S})$ of *stack expressions* as the smallest subset EXP_o and EXP_s, which are disjoint, satisfying that:

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

□

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

We specify the set of right-hand sides in attribute rules in SATTs before giving the definition of SATTs. We use bold-faced symbols, *e.g.* $\boldsymbol{a}, \boldsymbol{a_1}, \cdots$, 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 $\mathcal{S}$ be a stack system (X_o, X_s, Δ) with

$$X_o = \{a(\pi i) \mid a \in Syn, 1 \leq i \leq k\} \cup \{b(\pi) \mid b \in Inh\} \quad (33)$$

$$X_s = \{\boldsymbol{a}(\pi i) \mid \boldsymbol{a} \in StSyn, 1 \leq i \leq k\} \cup \{\boldsymbol{b}(\pi) \mid \boldsymbol{b} \in StInh\}. \quad (34)$$

We define the sets of *output right-hand sides* and *stack right-hand sides* as follows.

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

□

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

$$M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, a_0, \sharp, R)$$

where

- *Syn*, *Inh*, Σ, Δ, a_0 and $\sharp$ are the same as in Definition 3.2. 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,
- R is a set of *attribute rules* such that $R = \bigcup_{\sigma \in \Sigma \uplus \{\sharp\}} R^\sigma$ with finite sets R^σ of σ-rules satisfying the following conditions. For every $\sigma \in \Sigma \uplus \{\sharp\}$ 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 $1 \leq i \leq k$, 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 $1 \leq i \leq k$, 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 any terms such that $\eta_\sigma \in RHS_o(Syn, Inh, StSyn, StInh, \Delta, rank(\sigma)) \cup \{\bot\}$ and $\zeta_\sigma \in RHS_s(Syn, Inh, StSyn, StInh, \Delta, rank(\sigma)) \cup \{\bot\}$ and where $\eta_\sharp$ and $\zeta_\sharp$ are any terms such that $\eta_\sharp \in RHS_o(Syn, \emptyset, StSyn, \emptyset, \Delta, 1)$ and $\zeta_\sharp \in RHS_s(Syn, \emptyset, StSyn, \emptyset, \Delta, 1)$.

Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, a_0, \sharp, R)$ be a SATT and $\mathcal{K}$ be a set $\{k \mid \Sigma^{(k)} \neq \emptyset\}$. We define $RHS_o(M)$ and $RHS_s(M)$ as follows:

$$RHS_o(M) \stackrel{def}{=} \bigcup_{k \in \mathcal{K}} RHS_o(Syn, Inh, StSyn, StInh, \Delta, k) \tag{35}$$

$$RHS_s(M) \stackrel{def}{=} \bigcup_{k \in \mathcal{K}} RHS_s(Syn, Inh, StSyn, StInh, \Delta, k) \tag{36}$$

□

Example 3.10 The postfix-to-infix conversion in Section 1 can be expressed by a SATT M_{ptoi}, which corresponds to an attribute grammar in Figure 2. A 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 a set of the following attribute rules:

$$a_0(\pi) \xrightarrow{\sharp} a_0(\pi 1) \tag{37}$$
$$\boldsymbol{s}(\pi 1) \xrightarrow{\sharp} Empty \tag{38}$$
$$a_0(\pi) \xrightarrow{one} a_0(\pi 1) \tag{39}$$
$$\boldsymbol{s}(\pi 1) \xrightarrow{one} Cons(one, \boldsymbol{s}(\pi)) \tag{40}$$
$$a_0(\pi) \xrightarrow{two} a_0(\pi 1) \tag{41}$$
$$\boldsymbol{s}(\pi 1) \xrightarrow{two} Cons(two, \boldsymbol{s}(\pi)) \tag{42}$$
$$a_0(\pi) \xrightarrow{plus} a_0(\pi 1) \tag{43}$$
$$\boldsymbol{s}(\pi 1) \xrightarrow{plus} Cons(plus(Head(Tail(\boldsymbol{s}(\pi))), Head(\boldsymbol{s}(\pi))), Tail(Tail(\boldsymbol{s}(\pi)))) \tag{44}$$
$$a_0(\pi) \xrightarrow{multi} a_0(\pi 1) \tag{45}$$
$$\boldsymbol{s}(\pi 1) \xrightarrow{multi} Cons(multi(Head(Tail(\boldsymbol{s}(\pi))), Head(\boldsymbol{s}(\pi))), Tail(Tail(\boldsymbol{s}(\pi)))) \tag{46}$$
$$a_0(\pi) \xrightarrow{end} Head(\boldsymbol{s}(\pi)) \tag{47}$$

□

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

Definition 3.11 Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, a_0, \sharp, R)$ be a SATT, $t \in T_\Sigma$, $\mathcal{S} = (X_o \cup \{\bot\}, X_s, \Delta)$ be a stack system with

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

and ζ_w stand for an expression obtained by replacing every occurrence of π with a path $w \in path(\sharp(t))$ in ζ. The *derivation relation induced by M on t* is defined by a binary relation $\Rightarrow_{M,t} \subseteq (EXP_o(\mathcal{S}) \times (EXP_o(\mathcal{S}) \cup \{\bot\})) \cup (EXP_s(\mathcal{S}) \times EXP_s(\mathcal{S}))$ such that:

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

- $Tail(\zeta) \Rightarrow_{M,t} \begin{cases} \zeta' & (\text{if } \zeta = Cons(\eta', \zeta')) \\ Empty & (\text{if } \zeta = Empty) \\ Tail(\zeta') & (\text{otherwise, } \zeta \Rightarrow_{M,t} \zeta') \end{cases}$ where $\eta' \in EXP_o(\mathcal{S}) \cup \{\bot\}$ and $\zeta, \zeta' \in EXP_s(\mathcal{S})$.
- $\delta(\eta_1, \cdots, \eta_i, \cdots, \eta_n) \Rightarrow_{M,t} \delta(\eta_1, \cdots, \eta_i', \cdots, \eta_n)$ where $\eta_1, \cdots, \eta_{i-1}$ are irreducible and $nf(\Rightarrow_{M,t}, \eta_i) = \eta_i' (\neq \eta_i, \bot)$.
- $\delta(\eta_1, \cdots, \eta_i, \cdots, \eta_n) \Rightarrow_{M,t} \bot$ where $\eta_1, \cdots, \eta_{i-1}$ are irreducible and $nf(\Rightarrow_{M,t}, \eta_i) = \bot$.

This reduction system is determinant. We define an *attribute value* of $a(w)$ by $nf(\Rightarrow_{M,t}, a(w))$, if any, where w is a path in $\sharp(t)$. We may omit subscripts M and t of $\Rightarrow$ when it is clear from the context. □

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

Proposition 3.12 *Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, a_0, \sharp, R)$ be a SATT, $t \in T_\Sigma$, $a \in Syn \cup Inh$, and $w \in path(\sharp(t))$. If there exists $nf(\Rightarrow_{M,t}, a(w))$, then we always have*

$$nf(\Rightarrow_{M,t}, a(w)) \in T_\Delta.$$

□

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

Suppose that there exists $t' = nf(\Rightarrow_{M,t}, a(w))$. By the definition of $\Rightarrow_{M,t}$, we have $t' \in EXP_o$. We prove the proposition by analyzing the structure of t'. First, we show that there is no occurrence of the form of $a(w)$ with $a \in Syn \cup Inh \cup StSyn \cup StInh$ in t'. Second, we show that there is no occurrence of the form of $Cons(\eta, \zeta)$ and $Empty$ in t'. Finally, we show that there is no occurrence of the form of $Head(\zeta)$ and $Tail(\zeta)$ in t'. These facts indicate $t' \in T_\Delta$.

We can find that there is no occurrence of the form of $a(w)$ with $a \in Syn \cup Inh \cup StSyn \cup StInh$ in t' since all terms having occurrences of $a(w)$ are reducible by the reduction rules based on the attribute rules.

We can find that there is no occurrence of the form of $Cons(\eta, \zeta)$ in t' as follows. 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 $\mathcal{E} = \mathcal{E}'[Head(\bullet)]$ or $\mathcal{E} = \mathcal{E}'[Tail(\bullet)]$. This contradicts the fact that t' is irreducible. Thus there is no occurrence of the form of $Cons(\eta, \zeta)$ in t'. Similarly, we can show that there is no occurrence of the form of $Empty$ in t'.

We can find also that there is no occurrence of the form of $Head(\zeta)$ and $Tail(\zeta)$ in t'. The facts we have shown above imply that there is no occurrence of stack expressions in t'. Hence this indicates there is no occurrence of $Head$ and $Tail$ in t'. □

The semantics of SATTs is defined by the attribute value of the initial attribute at the root node in the same way as that of ATTs.

Definition 3.13 Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, a_0, \sharp, R)$ be a SATT. We define the *semantics* of a SATT M by a function $[\![M]\!] : T_\Sigma \to T_\Delta$ such that $[\![M]\!](t) \stackrel{def}{=} nf(\Rightarrow_{M,t}, a_0(\varepsilon))$ if there exists $nf(\Rightarrow_{M,t}, a_0(\varepsilon))$. The SATT M is *well-defined* if $[\![M]\!](t)$ can be defined for every $t \in T_\Sigma$. □

Example 3.14 Let M be an SATT given in Example 3.10. Figure 5 shows the reduction step for deriving

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

where $t = two(one(two(plus(multi(end)))))$ 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*, *TC*, *HE* and *TE* indicate the applied rule in Definition 3.11, *i.e. HC* for $Head(Cons(\eta,\zeta))$, *TC* for $Tail(Cons(\eta,\zeta))$, *HE* for $Head(Empty)$ and *TE* for $Tail(Empty)$. □

We write $SATT$ for a set of tree transformers such that $SATT = \{[\![M]\!] \mid M$ is a well-defined SATT.$\}$. We can easily show that $ATT \subsetneq SATT$.

Theorem 3.15 $ATT \subsetneq SATT$.

Proof. Since $ATT \subseteq SATT$ by the definitions of ATT and SATT, it is enough to show that this inclusion is proper, *i.e.* there is a SATT M which cannot be represented by any ATTs. Let $\Sigma = \{parenL^{(1)}, parenR^{(1)}, end^{(0)}\}$ and $\Delta = \{true^{(0)}\}$. For any ATT from T_Σ to T_Δ, we construct an equivalent finite state automaton by taking the set $\{\langle v_{a_1}, \cdots, v_{a_n}\rangle \mid v_{a_i} = true$ or $\perp\}$ as the state set, where each v_{a_i} represents the value of the attribute a_i in the ATT. The transition rules of the finite state automaton follow from attribute rules of the ATT. Consider a tree transformation from T_Σ to T_Δ which represents a parenthesis balancing validator, *i.e.* it returns *true* iff every right parenthesis *parenR* has its left counterpart *parenL* and vice versa in the input. Since this transformation cannot be achieved by any finite state automaton, the transformation cannot be represented by any ATT. On the other hand, the transformation can be defined by a SATT. Therefore we have $ATT \subsetneq SATT$. □

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

For a given input tree, the depth of stack values involved in a 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 stack values in every derivation is finite. This indicates that, once the input is fixed, a SATT can be simulated by an ATT obtained by replacing stack-attributes with a finite number of attributes. For example, if the number of trees to be stored in a stack is less than n, then an attribute rule $\boldsymbol{a}(\pi) \xrightarrow{\sigma} Cons(\eta, \boldsymbol{a}(\pi 1))$ with a stack synthesized attribute $\boldsymbol{a}$ in a SATT can be replaced with the following attribute rules:

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

where each $\langle \boldsymbol{a}, i\rangle (1 \leq i \leq n)$ is a synthesized attribute of an ATT that indicates the i-th stack element.

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

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

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

Definition 3.16 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 where A is a set of variables, and $n \in \mathbb{N}_+$. We define two *simulating functions* $\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n} : EXP_o(\mathcal{S}) \to T_{\Delta\cup\{\bot\}}(A)$ and $\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n} : EXP_s(\mathcal{S}) \times \mathbb{N}_+ \to T_{\Delta\cup\{\bot\}}(A)$ as follows:

$$\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e) = \Gamma_o(e) \quad \text{if } e \in X_o. \tag{48}$$

$$\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e,i) = \Gamma_s(e,i) \quad \text{if } e \in X_s. \tag{49}$$

$$\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\delta(e_1,\cdots,e_m)) = \begin{cases} \bot & (\text{if } \alpha_o(e_k) = \bot \text{ for some } 1 \le k \le m) \\ \delta(\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e_1),\cdots,\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e_m)) & (\text{otherwise}) \end{cases} \tag{50}$$

$$\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\mathit{Cons}(e_1,e_2),i) = \begin{cases} \alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e_1) & (\text{if } i = 1) \\ \alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e_2,i-1) & (\text{otherwise}) \end{cases} \tag{51}$$

$$\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\mathit{Empty},i) = \bot \tag{52}$$

$$\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\mathit{Head}(e)) = \alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e,1) \tag{53}$$

$$\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\mathit{Tail}(e),i) = \begin{cases} \bot & (\text{if } i \ge n) \\ \alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e,i+1) & (\text{otherwise}) \end{cases} \tag{54}$$

We also define the simulating function $\alpha_{\mathcal{R}}^{\mathcal{S},\Gamma_o,\Gamma_s,n}$ for attribute rules as follows. Let R be a set of attribute rules of the form $x \xrightarrow{\sigma} \eta$ where $\langle x,\eta\rangle \in (EXP_o(\mathcal{S}) \times EXP_o(\mathcal{S})) \cup (EXP_s(\mathcal{S}) \times EXP_s(\mathcal{S}))$. $\alpha_{\mathcal{R}}^{\mathcal{S},\Gamma_o,\Gamma_s,n}(R)$ is defined by

$$\{\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(x) \xrightarrow{\gamma} \alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\eta) \mid \alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(x) \neq \bot, x \xrightarrow{\gamma} \eta \in R\}$$
$$\cup \{\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(x,i) \xrightarrow{\gamma} \alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\zeta,i) \mid \alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(x,i) \neq \bot, 1 \le i \le n, x \xrightarrow{\gamma} \zeta \in R\}$$

□

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 functions, 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 restricted. The simulation is defined as follows.

Definition 3.18 Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, a_0, \sharp, R)$ be a SATT, and $\mathcal{S} = (X_o, X_s, \Delta)$ be a 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 $\Pi = \{\pi, \pi 1, \pi 2, \cdots\}$, Γ_o be a function such that $\Gamma_o(a(\varphi)) = a(\varphi)$ for every $a(\varphi) \in X_o$ and Γ_s be a 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 defined by the following ATT:

$$sim_n(M) \stackrel{def}{=} (Syn', Inh', \Sigma, \Delta \cup \{\bot\}, a_0, \sharp, \alpha_{\mathcal{R}}^{\mathcal{S},\Gamma_o,\Gamma_s,n}(R))$$

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

Let M be a SATT and t be a fixed input tree. If we take n large enough, then the derivation chains 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 the derivation chain by M on t.

Definition 3.19 Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, a_0, \sharp, R)$ be a SATT, $n \in \mathbb{N}_+$, $\mathcal{S} = (X_o, X_s, \Delta)$ be as given in Definition 3.11 and $\varphi \in EXP_o(\mathcal{S})$ satisfy $a_0(\varepsilon) \Rightarrow_{M,t}^* \varphi$. We define the n-depth simulation of φ by $\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\varphi)$ where Γ_o be a function such that $\Gamma_o(a(w)) = a(w)$ for every $a(w) \in X_o$, Γ_s be a function such that $\Gamma_s(\boldsymbol{a}(w), i) = \langle \boldsymbol{a}, i\rangle(w)$ for every $\boldsymbol{a}(w) \in X_s$ and $i \in \mathbb{N}_+$. □

Choosing the first branch of (54) in the computation of the n-depth simulation indicates that the stack depth is shorter than it is 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 a shortage of the stack depth is observed.

Lemma 3.20 *Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, a_0, \sharp, R)$ be a SATT, t be an input tree for M such that $[\![M]\!](t)$ can be defined, $\mathcal{S}, \Gamma_o, \Gamma_s$ be as given in* Definition 3.19, *and φ be a term such that $a_0(\varepsilon) \Rightarrow_{M,t}^* \varphi$. Suppose that $n > \max\{D_o(\varphi)\}$ where $D_o(e)$ stands for the set of the second arguments of α_s occurring in the computation of $\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(e)$. If $\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\eta)$ and $\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\eta, j)$ occur in the computation of $\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\varphi)$, then we have the following clauses:*

(i) If $\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\eta)$ is reducible by $\Rightarrow_{sim_n(M),t}$, then η is reducible by $\Rightarrow_{M,t}$.

(ii) If $\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\eta, j)$ is reducible by $\Rightarrow_{sim_n(M),t}$, then $Head(Tail^{j-1}(\eta))$ is reducible by $\Rightarrow_{M,t}$.

(iii) If $\alpha_o^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\eta) \Rightarrow_{sim_n(M),t} \bot$, then $\eta \Rightarrow_{M,t}^* \bot$.

(iv) If $\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}(\eta, j) \Rightarrow_{sim_n(M),t} \bot$, then $Head(Tail^{j-1}(\eta)) \Rightarrow_{M,t}^* \bot$.

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.
(CASE $\eta = a(w)$ FOR (i) AND (iii)) It is clear that (i) holds because $a(w)$ is always reducible. Suppose $\alpha_o(a(w)) \Rightarrow_{sim_n(M),t} \bot$. Note that $\alpha_o(a(w)) = a(w)$. There is an attribute rule of the form $a(\pi) \xrightarrow{\sigma} \bot$ with $\sigma = label_{\sharp(t)}(w)$ or $a(\pi i) \xrightarrow{\sigma} \bot$ with $\sigma = label_{\sharp(t)}(v)$ and $w = vi$ in

$sim_n(M)$. From the definition of $sim_n(M)$, M has an attribute rule of the same form. Then we have $a(w) \Rightarrow_{M,t} \bot$. Therefore (iii) holds.
(CASE $\eta = \boldsymbol{a}(w)$ FOR (ii) AND (iv)) It is clear that (ii) holds because $\boldsymbol{a}(w)$ is always reducible. Suppose that $\alpha_s(\boldsymbol{a}(w), j) \Rightarrow_{sim_n(M),t} \bot$ with $\boldsymbol{a} \in StSyn$. Let $\boldsymbol{a}(\pi) \xrightarrow{\sigma} \zeta$ be an attribute rule in M where $\sigma = label_{\sharp(t)}(w)$. Then $sim_n(M)$ has an attribute rule $\langle \boldsymbol{a}, j \rangle(\pi) \xrightarrow{\sigma} \alpha'_s(\zeta, j)$ where α'_s is $\alpha_s^{\mathcal{S},\Gamma_o,\Gamma_s,n}$ as given in Definition 3.16. If $\alpha'_s(\zeta, j) = \bot$ holds, we have $\alpha_s(\zeta_w, j) = \bot$ with ζ_w as given in Definition 3.11. Then $Head(Tail^{j-1}(\boldsymbol{a}(w))) \Rightarrow^+_{M,t} \alpha_s(\zeta_w, j) = \bot$. Therefore (iv) holds. We similarly show the statement in the case of $\boldsymbol{a} \in StInh$.
(CASE $\eta = \delta(\eta_1, \cdots, \eta_m)$ FOR (i) AND (iii)) Suppose $\alpha_o(\eta)$ is reducible. Then $\alpha_o(\eta_k)$ is reducible for some k. η_k is reducible from the induction hypothesis. Then η is reducible. Hence (i) holds. Suppose $\alpha_o(\eta) \Rightarrow_{sim_n(M),t} \bot$. We have $\alpha_o(\eta_k) \Rightarrow_{sim_n(M),t} \bot$ for some k. Then $\eta_k \Rightarrow_{M,t} \bot$ from the induction hypothesis. Hence (iii) holds.
(CASE $\eta = Cons(\eta', \zeta')$ FOR (ii) AND (iv)) Since $Head(Tail^{j-1}(\eta))$ is reducible to $\alpha_s(\eta, j)$, (ii) holds. Suppose $\alpha_s(\eta, j) \Rightarrow_{sim_n(M),t} \bot$. If $j = 1$, then we have $\alpha_o(\eta') \Rightarrow_{sim_n(M),t} \bot$. Then $\eta' \Rightarrow^*_{M,t} \bot$ from the induction hypothesis. Hence (iv) holds since we have $Head(\eta) \Rightarrow^*_{M,t} \eta'$. If $j > 1$, we have $\alpha_s(\zeta', j-1) \Rightarrow^*_{sim_n(M),t} \bot$. Then $Head(Tail^{j-2}(\zeta')) \Rightarrow^*_{M,t} \bot$ follows from the induction hypothesis. Hence (iv) holds since we have $Head(Tail^{j-1}(\eta)) \Rightarrow_{M,t} Head(Tail^{j-2}(\zeta'))$.
(CASE $\eta = Empty$ FOR (ii) AND (iv)) Since $Head(Tail^{j-1}(Empty))$ is reducible to $\bot$, (ii) and (iv) hold.
(CASE $\eta = Head(\eta')$ FOR (i) AND (iii)) We have $\alpha_o(\eta) = \alpha_s(\eta', 1)$. $Head(\eta')$ is reducible from the induction hypothesis. Therefore (i) holds. If $\alpha_o(\eta) \Rightarrow_{sim_n(M),t} \bot$, we have $\alpha_s(\eta', 1) \Rightarrow^*_{sim_n(M),t} \bot$. Then $Head(\eta') \Rightarrow^*_{M,t} \bot$ follows from the induction hypothesis. Hence (iii) holds.
(CASE $\eta = Tail(\eta')$ FOR (ii) AND (iv)) We have $\alpha_s(\eta, j) = \alpha_s(\eta', j+1)$ from $j < n$ which is the assumption of j in this lemma. $Head(Tail^j(\eta'))$ is reducible from the induction hypothesis. Since $Head(Tail^j(\eta')) = Head(Tail^{j-1}(\eta))$, (ii) holds. If $\alpha_s(\eta, j) \Rightarrow_{sim_n(M),t} \bot$, then $\alpha_s(\eta', j+1) \Rightarrow_{sim_n(M),t} \bot$ follows from the induction hypothesis. Hence $Head(Tail^j(\eta')) \Rightarrow^*_{M,t} \bot$ from the induction hypothesis. Since $Head(Tail^{j-1}(\eta)) = Head(Tail^j(\eta'))$, (iv) holds. □

The next theorem shows that a SATT is mimicked by an n-depth simulation of the SATT for enough large n for every fixed input.

Theorem 3.21 *Let M be a SATT and t be an input tree for M. There exists $n_0 \in \mathbb{N}$ such that $[\![sim_n(M)]\!](t) = [\![M]\!](t)$ for any $n \geq n_0$.*

Proof. Let $M = (Syn, Inh, StSyn, StInh, \Sigma, \Delta, a_0, \sharp, R)$ be a SATT, $t \in T_\Sigma$ be an input tree for M such that the output tree $t' = [\![M]\!](t)$ can be defined, and $n \in \mathbb{N}_+$ and $a_0(\varepsilon) \Rightarrow_{M,t} \varphi_1 \Rightarrow_{M,t} \cdots \Rightarrow_{M,t} \varphi_k (= t')$ be a derivation chain induced by M on t. Let $\mathcal{S}$, Γ_o and Γ_s be as given in Definition 3.19. 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, α'_s and $\alpha'_{\mathcal{R}}$ for $\alpha_o^{\mathcal{S}',\Gamma'_o,\Gamma'_s,n}$, $\alpha_s^{\mathcal{S}',\Gamma'_o,\Gamma'_s,n}$ and $\alpha_{\mathcal{R}}^{\mathcal{S}',\Gamma'_o,\Gamma'_s,n}$, respectively.

Suppose that $n > \max\{D_o(\phi) \mid a_0(\varepsilon) \Rightarrow^*_{M,t} \phi\}$ where D_o is as defined in Lemma 3.20. We show that

$$\alpha_o(\varphi) \Rightarrow^?_{sim_n(M),t} \alpha_o(\psi) \quad \text{if} \quad \varphi \in EXP_o(\mathcal{S}) \text{ and } \varphi \Rightarrow_{M,t} \psi \tag{55}$$

$$\alpha_s(\varphi, j) \Rightarrow^?_{sim_n(M),t} \alpha_s(\psi, j) \quad \text{if} \quad \varphi \in EXP_s(\mathcal{S}), \varphi \Rightarrow_{M,t} \psi \tag{56}$$

where $\alpha_o(\varphi)$ and $\alpha_s(\varphi, j)$ occurs in the computation of $\alpha_o(\phi)$ such that $a_0(\varepsilon) \Rightarrow^*_{M,t} \phi$. This implies $nf(\Rightarrow_{sim_n(M),t}, \alpha_o(a_0(\varepsilon))) = nf(\Rightarrow_{M,t}, a_0(\varepsilon))$, *i.e.* $[\![sim_n(M)]\!](t) = [\![M]\!](t)$ since $\alpha_o(a_0(\varepsilon)) = a_0(\varepsilon)$.

We prove the statements (55) and (56) by induction on the structure of φ.

(CASE $\varphi = a(w)$ WITH $a \in Syn$) We have $\varphi \Rightarrow_{M,t} \eta_w$ where $a(\pi) \xrightarrow{\sigma} \eta \in R$ and η_w is obtained by replacing π with w in η. Since $a(\pi) \xrightarrow{\sigma} \alpha'_o(\eta) \in \alpha'_{\mathcal{R}}(R)$, we have $a(w) \Rightarrow_{sim_n(M),t} \eta'_w$ where η'_w is obtained by replacing π with w in $\alpha'_o(\eta)$. Now we can show $\eta'_w = \alpha_o(\eta_w)$ by the definitions of α_o and α'_o and the fact that $\Gamma_o(a(w))(= a(w))$ is obtained by replacing π with w in $\Gamma'_o(a(\pi))(= a(\pi))$. Then (55) holds because $\alpha_o(a(w)) = a(w)$.

(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 $\varphi \Rightarrow_{M,t} \zeta_w$ where $\boldsymbol{a}(\pi) \xrightarrow{\sigma} \zeta \in R$ and ζ_w is obtained by replacing π with w in ζ. Since $\langle \boldsymbol{a}, j\rangle(\pi) \xrightarrow{\sigma} \alpha'_s(\zeta, j) \in \alpha'_{\mathcal{R}}(R)$, we have $\langle \boldsymbol{a}, j\rangle \Rightarrow_{sim_n(M),t} \zeta'_{w,j}$ where $\zeta'_{w,j}$ is obtained by replacing π with w in $\alpha'_s(\zeta, j)$. Similar to the first case, we can show $\zeta'_{w,j} = \alpha_s(\zeta_w, j)$. Then (56) holds.

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

(CASE $\varphi = Head(Cons(\eta, \zeta))$) We have $\varphi \Rightarrow_{M,t} \eta$. (55) holds because $\alpha_o(Head(Cons(\eta, \zeta))) = \alpha_o(\eta)$.

(CASE $\varphi = Head(Empty)$) We have $\varphi \Rightarrow_{M,t} \bot$. (55) holds because $\alpha_o(Head(Empty)) = \alpha_o(\bot)$.

(CASE $\varphi = Head(\zeta)$ WITH $\zeta \neq Cons(\eta_1, \zeta_1)$ AND $\zeta \neq Empty$) We have $\varphi \Rightarrow_{M,t} Head(\zeta')$ where $\zeta \Rightarrow^+_{M,t} \zeta'$. We find $\alpha_s(\zeta, 1) \Rightarrow_{sim_n(M),t} \alpha_s(\zeta', 1)$ from the induction hypothesis. Then (55) holds.

(CASE $\varphi = Tail(Cons(\eta, \zeta))$) We have $\varphi \Rightarrow_{M,t} \zeta$. (56) holds because $\alpha_s(Tail(Cons(\eta, \zeta)), j) = \alpha_s(\zeta, j)$ for $1 \leq j < n$ from the assumption on n.

(CASE $\varphi = Tail(Empty)$) We have $\varphi \Rightarrow_{M,t} Empty$. (56) holds because $\alpha_s(Tail(Empty), j) = \alpha_s(Empty, j+1)(= \bot)$ for any j.

(CASE $\varphi = Tail(\zeta)$ WITH $\zeta \neq Cons(\eta_1, \zeta_1)$ AND $\zeta \neq Empty$) We have $\varphi \Rightarrow_{M,t} Tail(\zeta')$ where $\zeta \Rightarrow^+_{M,t} \zeta'$. We find $\alpha_s(\zeta, j) \Rightarrow_{sim_n(M),t} \alpha_s(\zeta', j)$ from the induction hypothesis. Then (55) holds.

(CASE $\varphi = \delta(\eta_1, \cdots, \eta_m)$ WHERE $\eta_1, \cdots, \eta_{k-1}$ ARE IRREDUCIBLE AND $nf(\Rightarrow_{M,t}, \eta_k) = \eta'_k (\neq \eta_k, \bot)$.) We have $\delta(\eta_1, \cdots, \eta_k, \cdots, \eta_m) \Rightarrow_{M,t} \delta(\eta_1, \cdots, \eta'_k, \cdots, \eta_m)$ by the definition of $\Rightarrow_{M,t}$. $\alpha_o(\eta_1), \cdots, \alpha_o(\eta_{k-1})$ are irreducible from Lemma 3.20 (i). From $\eta'_k \neq \bot$, Lemma 3.20 (i), (iii) and the induction hypothesis, we have $\alpha_o(\eta'_k) = nf(\Rightarrow_{sim_n(M),t}, \alpha_o(\eta_k))(\neq \bot)$. Then $\alpha_o(\varphi) = \delta(\alpha_o(\eta_1), \cdots, \alpha_o(\eta_k), \cdots, \alpha_o(\eta_m)) \Rightarrow^?_{sim_n(M),t} \delta(\alpha_o(\eta_1), \cdots, \alpha_o(\eta'_k), \cdots, \alpha_o(\eta_m))$. Therefore (55) holds.

(CASE $\varphi = \delta(\eta_1, \cdots, \eta_m)$ WHERE $\eta_1, \cdots, \eta_{k-1}$ ARE IRREDUCIBLE AND $nf(\Rightarrow_{M,t}, \eta_k) = \bot$.) We have $\varphi \Rightarrow_{M,t} \bot$ by the definition of $\Rightarrow_{M,t}$. $\alpha_o(\eta_1), \cdots, \alpha_o(\eta_{k-1})$ are irreducible from Lemma 3.20 (i). From Lemma 3.20 (i), (iii) and the induction hypothesis, $nf(\Rightarrow_{sim_n(M),t}, \alpha_o(\eta_k)) = \alpha_o(\bot)(= \bot)$ holds. Then $\alpha_o(\varphi) = \delta(\alpha_o(\eta_1), \cdots, \alpha_o(\eta_k) \cdots, \alpha_o(\eta_m)) \Rightarrow_{sim_n(M),t} \bot$. Therefore (55) holds. □

For a SATT M and an input tree t, we write $mxd(M,t)$ to denote the minimum possible number n_0 in Theorem 3.21.

Corollary 3.22 *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 $(Syn, Inh, StSyn, StInh, \Sigma, \Delta, a_0, \sharp, R)$. From Definition 3.13, it suffices to show that there exists $nf(\Rightarrow_{M,t}, a_0(\varepsilon))$ 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 $n_0 \geq mxd(M,t)$. Suppose that $sim_n(M)$ is a well-defined ATT for any $n \in \mathbb{N}_+$. Then there exists $nf(\Rightarrow_{sim_{n_0}(M),t}, a_0(\varepsilon))$ which is $[\![sim_{n_0}(M)]\!](t)$. Since $[\![M]\!](t) = [\![sim_{n_0}(M)]\!](t)$ holds from Theorem 3.21, there exists $nf(\Rightarrow_{M,t}, a_0(\varepsilon))$ 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 $\varphi \in \{\pi, \pi1, \pi2 \cdots\}$ 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 $M_1 © M_2$ to denote a single ATT which is the result of descriptional composition of M_1 and M_2. The ATT $M_1 © M_2$ computes a transformation equivalent to $[\![M_1]\!] \circ [\![M_2]\!]$, thus $M_1 © M_2$ 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 $M_1 © M_2$ 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 (©-Correctness, Ganzinger[Gan83] and Giegerich[Gie88]) *If M_1 and M_2 are well-defined sur-ATTs, then $M_1 © M_2$ is a well-defined sur-ATT such that $[\![M_1]\!] \circ [\![M_2]\!] = [\![M_1 © M_2]\!]$.*

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

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 $\boldsymbol{s}(\pi)$ is referred three times in a *plus*-rule (44), these three references do not overlap each other: $Head(\boldsymbol{s}(\pi))$ represents a reference to the first element of the stack; $Head(Tail(\boldsymbol{s}(\pi)))$ represents a reference to the second element of the stack; $Tail(Tail(\boldsymbol{s}(\pi)))$ 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 $\Pi=\{\pi,\pi1,\pi2,\cdots\}$. The SATT $M_1\widehat{\copyright}M_2$ 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 (40) in M_2:

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

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

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

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

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

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):

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

Figure 6 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.* $[\![M_1 \widehat{\copyright} M_2]\!] = [\![M_1]\!] \circ [\![M_2]\!]$ 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 6: A SATT M_{ptop} obtained by composing an ATT M_{itop} and a SATT M_{ptoi}

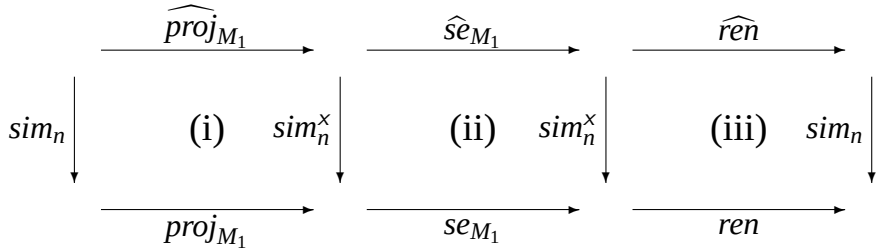

Figure 7: The extended descriptional composition and its simulation

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 $sim_n^x(M)$ is defined by the following septuple:

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

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 7 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) $sim_n^x \circ \widehat{proj}_{M_1} = proj_{M_1} \circ sim_n$

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

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

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:

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

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

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

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

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

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:

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

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:

$$\begin{aligned}
\alpha_s(a(Tail(\zeta)), i_0) &= \begin{cases} \bot & (\text{if } \alpha'_s(Tail(\zeta), i_0) = \bot) \\ a(\alpha'_s(Tail(\zeta), i_0)) & (\text{otherwise}) \end{cases} \\
&= \begin{cases} \bot & (\text{if } i_0 = n \text{ or } \alpha'_s(\zeta, i_0 + 1) = \bot) \\ a(\alpha'_s(\zeta, i_0 + 1)) & (\text{otherwise}) \end{cases} \\
\alpha_s(Tail(a(\zeta)), i_0) &= \begin{cases} \bot & (\text{if } i_0 = n) \\ \alpha'_s(a(\zeta), i_0 + 1) & (\text{otherwise}) \end{cases} \\
&= \begin{cases} \bot & (\text{if } i_0 = n \text{ or } \alpha'_s(\zeta, i_0 + 1) = \bot) \\ a(\alpha'_s(\zeta, i_0 + 1)) & (\text{otherwise}) \end{cases}
\end{aligned}$$

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 $a \in \mathit{Att}_1$,

$a' \in Att_2$ 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 $ren \circ sim_n^x(U)$ 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 $ren \circ sim_n^x$ coincide. Therefore $sim_n \circ \widehat{ren}(U) = ren \circ sim_n^x(U)$. □

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*

$$[\![M_1 \copyright sim_n(M_2)]\!] = [\![sim_n(M_1 \widehat{\copyright} M_2)]\!]$$

holds for any n.

Proof.

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

□

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

Theorem 4.13 ($\widehat{\copyright}$-Correctness, I) *If M_1 is a well-defined sur-ATT and M_2 is and a well-defined sur-SATT, then $M_1 \widehat{\copyright} M_2$ is a well-defined sur-SATT such that $[\![M_1]\!] \circ [\![M_2]\!] = [\![M_1 \widehat{\copyright} M_2]\!]$, i.e. $[\![M_1]\!] \circ [\![M_2]\!](t) = [\![M_1 \widehat{\copyright} M_2]\!](t)$ 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 $n = \max\{mxd(M_2,t), mxd(M_1 \widehat{\copyright} M_2,t)\}$, we have

$$\begin{array}{rcll}
[\![M_1 \widehat{\copyright} M_2]\!](t) & = & [\![sim_n(M_1 \widehat{\copyright} M_2)]\!](t) & \text{(by Theorem 3.21)} \\
& = & [\![M_1 \copyright sim_n(M_2)]\!](t) & \text{(by Proposition 4.12)} \\
& = & [\![M_1]\!] \circ [\![sim_n(M_2)]\!](t) & \text{(by Theorem 4.3)} \\
& = & [\![M_1]\!]([\![sim_n(M_2)]\!](t)) & \text{(by the definition of } \circ) \\
& = & [\![M_1]\!]([\![M_2]\!](t)) & \text{(by Theorem 3.21)} \\
& = & [\![M_1]\!] \circ [\![M_2]\!](t) & \text{(by the definition of } \circ)
\end{array}$$

It follows that $M_1 \copyright sim_n(M_2)$ is a well-defined sur-ATT for any n from Theorem 4.3. Hence, $sim_n(M_1 \widehat{\copyright} M_2)$ is a well-defined sur-ATT for any n. Therefore $M_1 \widehat{\copyright} M_2$ is a well-defined sur-SATT from Definition 4.7 and Corollary 3.22. □

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

The closure properties are corollaries to these theorems.

Corollary 4.15

(i) $ATT_{su} \circ SATT_{su} = SATT_{su}$.

(ii) $TDTT \circ SATT = SATT$.

Proof. The first statement follows immediately from the fact that ATT_{su} contains the identical tree transformation and that $ATT_{su} \circ SATT_{su} \subseteq SATT_{su}$ 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 SATTs 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.

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. I and II.

[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.

[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 push-down 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 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. Published as ACM SIGPLAN Notices, volume 20, number 7.

[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.

[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.

[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 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science, 2001. available online: `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.

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

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