Compositions of Top-down Tree Transducers with ε-rules

Andreas Maletti[1,⋆] and Heiko Vogler[2]

[1] Universitat Rovira i Virgili, Departament de Filologies Romàniques
Avinguda de Catalunya 35, 43002 Tarragona, Spain
andreas.maletti@urv.cat
[2] Technische Universität Dresden, Faculty of Computer Science
01062 Dresden, Germany
heiko.vogler@tu-dresden.de

Abstract. Top-down tree transducers with ε-rules (εtdtts) are a restricted version of extended top-down tree transducers. They are implemented in the framework Tiburon and fulfill some criteria desirable in a machine translation model. However, they compute a class of transformations that is not closed under composition (not even for linear and nondeleting εtdtts). A composition construction that composes two εtdtts M and N is presented, and it is shown that the construction is correct, whenever (i) N is linear, (ii) M is total or N is nondeleting, and (iii) M has at most one output symbol in each rule.

1 Introduction

Many aspects of machine translation (MT) of natural languages can be formalized by employing weighted finite-state (string) transducers [1,2]. Successful implementations based on this word- or phrase-based approach are, for example, the At&t Fsm toolkit [3], Xerox's finite-state calculus [4], the Rwth toolkit [5], Carmel [6], and OpenFst [7]. However, the phrase-based approach is not expressive enough, for example, to easily handle the rotation needed in the translation of the English structure NP-V-NP (subject-verb-noun phrase) to the Arabic structure V-NP-NP. A finite-state transducer can only implement this rotation by storing the subject, which might be very long, in its finite memory.

Syntax-based (or tree-based) formalisms can remedy this shortage. Examples of such formalisms are the top-down tree transducer [8,9], the extended top-down tree transducer [10,11,12], the synchronous tree substitution grammar [13], the synchronous tree-adjoining grammar [14], the multi bottom-up tree transducer [15,16,17,18], and the extended multi bottom-up tree transducer [19]. Some of these models are formally compared in [20,21,19] and an overview on their usability in syntax-based MT is presented in [22,23]. For example, the toolkit Tiburon [24] implements weighted extended top-down tree transducers with some standard operations.

⋆ This author was financially supported by the *Ministerio de Educación y Ciencia* (MEC) grants JDCI-2007-760 and MTM-2007-63422.

A. Yli-Jyrä et al. (Eds.): FSMNLP 2009, LNAI 6062, pp. 69–80, 2010.

In this paper, we consider top-down tree transducers with ε-rules (εtdtts), which are a (syntactically) restricted version of extended top-down tree transducers [25,21] and as such implemented in TIBURON [24]. In fact, εtdtts properly generalize finite-state transducers, and thus existing models for the phrase-based approach can be straightforwardly translated into εtdtts. Moreover, ε-rules sometimes allow the designer to more compactly express himself. The addition of ε-rules is also a step towards symmetry of the model; extended top-down tree transducers have full symmetry in the linear and nondeleting case.

It is often beneficial in the development process to train "small" task-specific transducers [26]. Then we would like to compose those "small" transducers to obtain a single transducer, to which further operations can be applied. The success of the finite-state transducer toolkits (like CARMEL [6] and OPENFSM [7]) is to a large extent due to this compositional approach, which allows us to avoid cascades of transducers.

Here, we study εtdtts in order to better understand compositions of extended top-down tree transducers. In fact, several phenomena (ε-rules and nonshallow left-hand sides) contribute to the problems faced in such compositions. To investigate the effect of ε-rules, we prove a composition result inspired by BAKER [27,28]. Namely, the composition of two εtdtts M and N can be computed by one εtdtt if (i) N is linear, (ii) M is total or N is nondeleting, and (iii) M has at most one output symbol in each rule (cf. Theorem 17). Compared to BAKER's original result [27] for nondeterministic top-down tree transducers, we have the additional condition that each rule of M contains at most one output symbol. Our result generalizes the composition closure for transformations computed by finite-state transducers, for which Condition (iii) can always be achieved [29, Cor. III.6.2]. However, this is not true for linear and nondeleting εtdtts, and we investigate procedures that reduce the number of output symbols per rule.

2 Notation

Let $X = \{x_1, x_2, \ldots\}$ be a fixed set of variables and $X_k = \{x_i \mid 1 \leq i \leq k\}$ for every $k \geq 0$. The composition of the relations $\tau_1 \subseteq A \times B$ and $\tau_2 \subseteq B \times C$ is denoted by $\tau_1 ; \tau_2$. This notation is extended to classes of relations in the obvious way. Ranked alphabets are defined as usual. We use Σ_k to denote the set of all symbols of rank k in the ranked alphabet Σ. To indicate that $\sigma \in \Sigma$ has rank k we write $\sigma^{(k)}$. Two ranked alphabets Σ and Δ are called compatible if every symbol of $\Sigma \cap \Delta$ is assigned the same rank in Σ as in Δ. Then, $\Sigma \cup \Delta$ is again a ranked alphabet. The set of Σ-trees indexed by a set V is denoted by $T_\Sigma(V)$. We denote by $C_\Sigma(V) \subseteq T_\Sigma(\{x_1\} \cup V)$ the set of all contexts over Σ indexed by V, which are Σ-trees indexed by $\{x_1\} \cup V$ such that the variable x_1 occurs exactly once. We abbreviate $T_\Sigma(\emptyset)$ and $C_\Sigma(\emptyset)$ by T_Σ and C_Σ, respectively.

Given $L \subseteq T_\Sigma(V)$ we denote the set $\{\sigma(t_1, \ldots, t_k) \mid \sigma \in \Sigma_k, t_1, \ldots, t_k \in L\}$ by $\Sigma(L)$. Now let $t \in T_\Sigma(V)$. We denote the set of all variables $x \in X$ that occur in t by $\mathrm{var}(t)$. Finally, for any finite set $Y \subseteq X$ we call a mapping $\theta \colon Y \to T_\Sigma(V)$

a substitution. Such a substitution θ applied to t yields the tree $t\theta$ that is obtained from t by replacing each occurrence of every $x \in Y$ by $\theta(x)$. If $Y = \{x_1\}$ and $t \in C_\Sigma(V)$, then we also write $t[\theta(x_1)]$ instead of $t\theta$.

3 Top-down Tree Transducers with ε-rules

In this section, we recall top-down tree transducers [9,8,30,31]. We slightly change their definition to allow rules that do not consume an input symbol. Such rules are called *(input) ε-rules*.

Definition 1. *A* top-down tree transducer with ε-rules *(for short: εtdtt) is a system $(Q, \Sigma, \Delta, I, R)$ where:*

- *Q is a finite set of* states *(disjoint to Σ and Δ). Each state is considered to be of rank* 1.
- *Σ and Δ are ranked alphabets of* input *and* output symbols*, respectively.*
- *$I \subseteq Q$ is a set of* initial states.
- *R is a finite set of* rules *of the form*
  (i) $q(\sigma(x_1, \ldots, x_k)) \to r$ with $q \in Q$, $\sigma \in \Sigma_k$, and $r \in T_\Delta(Q(X_k))$ or
  (ii) $q(x_1) \to r$ with $q \in Q$ and $r \in T_\Delta(Q(X_1))$.

Rules of the form (i) are input-consuming*, whereas rules of the form (ii) are* ε-rules*. We denote the set of input-consuming and ε-rules of R by R^Σ and R^ε, respectively. An εtdtt without ε-rules is called a top-down tree transducer (tdtt).*

For simplicity, we generally assume that input and output ranked alphabets are compatible (i.e., each encountered symbol has only one rank). For the rest of this section, let $M = (Q, \Sigma, \Delta, I, R)$ be an εtdtt. A rule $l \to r \in R$ is called *linear* (respectively, *nondeleting*) if every variable $x \in \mathrm{var}(l)$ occurs at most once (respectively, at least once) in r. The εtdtt M is *linear* (respectively, *nondeleting*) if all of its rules are so.

Example 2. Let $N = (\{p\}, \Sigma, \Sigma, \{p\}, R)$ be the εtdtt with $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$ and the following rules:

$$p(x_1) \to \gamma(p(x_1)) \qquad p(\gamma(x_1)) \to \gamma(p(x_1))$$
$$p(\sigma(x_1, x_2)) \to \sigma(p(x_1), p(x_2)) \qquad p(\alpha) \to \alpha \ .$$

Clearly, N is linear and nondeleting. Intuitively, the ε-rule allows the transducer to insert (at any place in the output tree) any number of γ-symbols. The remaining rules just output the input symbol and process the subtrees recursively.

The semantics of the εtdtt M is given by rewriting (cf. [21, Definition 2]). Since in the composition of two εtdtts M and N (cf. Definition 13), the εtdtt N will process right-hand sides of rules of M and such right-hand sides involve symbols of the form $q(x_i)$ that are not in the input alphabet of N, we define the rewrite relation also for trees containing symbols not present in $Q \cup \Sigma \cup \Delta$.

So let Σ' and Δ' be two compatible ranked alphabets such that $\Sigma \subseteq \Sigma'$ and $\Delta \subseteq \Delta'$. Moreover, let $l \to r \in R$ be a rule, $C \in C_{\Delta'}(Q(T_{\Sigma'}))$ be a context, and $\theta\colon \mathrm{var}(l) \to T_{\Sigma'}$ be a substitution. Then we say that $C[l\theta]$ *rewrites to* $C[r\theta]$ *using* $l \to r$, denoted by $C[l\theta] \Rightarrow_M^{l\to r} C[r\theta]$. For every $\zeta, \xi \in T_{\Delta'}(Q(T_{\Sigma'}))$ we write $\zeta \Rightarrow_M \xi$ if there exists $\rho \in R$ such that $\zeta \Rightarrow_M^{\rho} \xi$. The *tree transformation computed by* M, denoted by τ_M, is the relation

$$\tau_M = \{(t, u) \in T_\Sigma \times T_\Delta \mid \exists q \in I\colon q(t) \Rightarrow_M^* u\}$$

where $\Rightarrow_M^*$ denotes the reflexive, transitive closure of $\Rightarrow_M$. Two εtdtt are *equivalent* if their computed tree transformations coincide.

Example 3. Consider the εtdtt N of Example 2. To illustrate the use of symbols that are not in $Q \cup \Sigma$, let v be such a new symbol. Then

$$p(\gamma(v)) \Rightarrow_N \gamma(p(\gamma(v))) \Rightarrow_N \gamma(\gamma(p(v)))\ .$$

In a similar way, we have $p(\gamma(\alpha)) \Rightarrow_N^* \gamma(\gamma(p(\alpha))) \Rightarrow_N \gamma(\gamma(\alpha))$, and consequently, $(\gamma(\alpha), \gamma(\gamma(\alpha))) \in \tau_N$.

We say that M is *total* if for every $q \in Q$ and $t \in T_\Sigma$ there exists $u \in T_\Delta$ such that $q(t) \Rightarrow_M^* u$. This property is clearly decidable, which can be seen as follows: We first modify M such that $I = \{q\}$ and call the resulting εtdtt M_q. It is known [25] that the domain of τ_{M_q} is a regular tree language, and thus we can decide whether it is T_Σ [30,31]. The domain of τ_{M_q} is T_Σ for all states $q \in Q$ if and only if M is total. The εtdtt of Example 2 is total, but let us illustrate the notion of totality on another example.

Example 4. We consider the ranked alphabet $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$ and the εtdtt $M = (\{q, q_1\}, \Sigma, \Sigma, \{q\}, R)$ with the rules

$$\begin{aligned} q(x_1) &\to \gamma(q_1(x_1)) & q(\gamma(x_1)) &\to \gamma(q(x_1)) \\ q_1(\sigma(x_1, x_2)) &\to \sigma(q(x_1), \sigma(\alpha, q(x_2))) & q(\alpha) &\to \alpha\ . \end{aligned}$$

Clearly, there is no tree $u \in T_\Sigma$ such that $q_1(\alpha) \Rightarrow_M^* u$. Thus M is not total.

Finally, the class of tree transformations computed by εtdtts is denoted by εTOP. We use 'l' and 'n' to restrict to the transformations computed by linear and nondeleting εtdtts, respectively. For example, ln-εTOP denotes the class of transformations computed by linear, nondeleting εtdtts.

We conclude this section by showing that ln-εTOP is not closed under composition. This shows that linear, nondeleting εtdtts are strictly less expressive than linear, nondeleting recognizable tree transducers of [32] because the latter compute a class of transformations that is closed under composition [32, Theorem 2.4]. Note that the latter have regular extended right-hand sides.

Theorem 5. ln-εTOP ; ln-εTOP $\not\subseteq$ l-εTOP.

Proof. Consider the two transformations

$$\tau_1 = \{(\alpha, \sigma(\alpha, \alpha))\} \qquad \text{and} \qquad \tau_2 = \{(\sigma(\alpha, \alpha), \sigma(\gamma^m(\alpha), \gamma^n(\alpha))) \mid m, n \geq 0\}$$

where $\gamma^m(\alpha)$ abbreviates $\gamma(\gamma(\cdots\gamma(\alpha)\cdots))$ containing the symbol γ exactly m times. It is easy to show that τ_1 and τ_2 are in ln-εTOP. However, the composition $\tau_1 ; \tau_2$ clearly cannot be computed by any linear εtdtt. □

4 Separating Output Symbols

Next we define two normal forms of εtdtts. One will be the normal form we already mentioned, in which each rule contains at most one output symbol. For our composition results in the next section, we will require the first transducer to be in this normal form. This normal form can always be achieved, but in general linearity is not preserved (see Theorem 11). The second normal form is less restricted, viz. we only demand that no (nontrivial) context can be separated from any right-hand side of a rule. This normal form can always be achieved (see Theorem 8) while preserving linearity or nondeletion. The underlying algorithm is valuable because on some input transducers it achieves 1-symbol normal form.

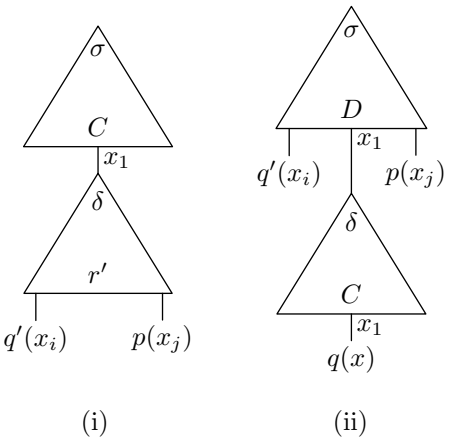

Fig. 1. The two forms of right-hand sides that are forbidden in a maximally output-separated εtdtt.

Definition 6 (cf. [19, Definition 4]). *Let $M = (Q, \Sigma, \Delta, I, R)$ be an εtdtt.*

- *It is in* 1-symbol normal form *if $r \in Q(X) \cup \Delta(Q(X))$ for every $l \to r \in R$.*
- *It is* maximally output-separated *if for every $l \to r \in R$:*
  - *(i) $r \neq C[r']$ for every context $C \in C_\Delta \setminus \{x_1\}$ and $r' \in \Delta(T_\Delta(Q(X)))$, and*
  - *(ii) $r \neq D[C[q(x)]]$ for every context $D \in C_\Delta(Q(X)) \setminus \{x_1\}$ potentially containing states, context $C \in C_\Delta \setminus \{x_1\}$, state $q \in Q$, and variable $x \in X$.*

By definition, every εtdtt in 1-symbol normal form is maximally output-separated. Let us explain the normal forms in more detail. The right-hand side of a rule of an εtdtt in 1-symbol normal form is either a state followed by a subtree variable (like $q(x_1)$) or a single output symbol at the root with states as children (like $\delta(p(x_1), q(x_2))$ or α). The weaker property of maximal output-separation only demands that no nontrivial context C without states can be split from the right-hand side r of a rule. More precisely, Condition (i) states that C cannot be taken off from r such that at least one output symbol remains, and Condition (ii) states that $C[q(x)]$ may not form a proper subtree of r (see Figure 1).

Example 7. Let $M = (\{q\}, \Sigma, \Sigma, \{q\}, R)$ be the linear, nondeleting, and total εtdtt with $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$ and the following rules

$$q(\sigma(x_1, x_2)) \to \gamma(\sigma(q(x_1), \sigma(\alpha, q(x_2)))) \tag{ρ_1}$$
$$q(\gamma(x_1)) \to \gamma(q(x_1)) \tag{ρ_2}$$
$$q(\alpha) \to \alpha \ . \tag{ρ_3}$$

This εtdtt is not maximally output-separated because the right-hand side of ρ_1 can be decomposed into the context $\gamma(x_1)$ and the subtree $\sigma(q(x_1), \sigma(\alpha, q(x_2)))$, which violates Condition (i) of Definition 6. Further, the subtree $\sigma(\alpha, q(x_2))$ can be decomposed into the context $\sigma(\alpha, x_1)$ and the subtree $q(x_2)$, which violates Condition (ii) of Definition 6. Note that the rules ρ_2 and ρ_3 satisfy Conditions (i) and (ii) because they have only one output symbol.

It was formally verified in [33] that for every linear and nondeleting εtdtt there exists an equivalent linear and nondeleting εtdtt that is maximally output-separated. Here we extend this result and show that, in general, every εtdtt can be transformed into an equivalent maximally output-separated εtdtt.

Theorem 8. *For every εtdtt M we can effectively construct an equivalent maximally output-separated εtdtt N. Moreover, if M is linear (respectively, nondeleting), then so is N.*

Proof. Let $M = (Q, \Sigma, \Delta, I, R)$ be an εtdtt. We will present an iterative procedure that transforms M into an equivalent maximally output-separated εtdtt. Suppose that M is not maximally output-separated; otherwise the procedure terminates. Thus, there exist a rule $l \to r \in R$ and

(i) a context $C \in C_\Delta \setminus \{x_1\}$ and $r' \in \Delta(T_\Delta(Q(X)))$ such that $r = C[r']$, or
(ii) a context $D \in C_\Delta(Q(X)) \setminus \{x_1\}$ potentially containing states, a context $C \in C_\Delta \setminus \{x_1\}$, a state $q \in Q$, and a variable $x \in X$ such that $r = D[C[q(x)]]$.

Let $p \notin Q$ be a new state and $l = p'(l')$ for some $p' \in Q$ and $l' \in T_\Sigma(X)$. We construct the εtdtt $M' = (Q \cup \{p\}, \Sigma, \Delta, I, R')$ with $R' = (R \setminus \{l \to r\}) \cup R''$ where:

- In case (i), R'' contains the two rules: $p'(x_1) \to C[p(x_1)]$ and $p(l') \to r'$.
- Otherwise R'' contains the two rules: $l \to D[p(x)]$ and $p(x_1) \to C[q(x_1)]$.

Note that M' is linear (respectively, nondeleting), if M is. Moreover, we observe that each new right-hand side (i.e., $C[p(x_1)]$, r', $D[p(x)]$, and $C[q(x_1)]$) contains strictly fewer output symbols than r. Hence, this procedure can only be finitely iterated and eventually must yield a maximally output-separated εtdtt N. The proof that M and M' are equivalent (i.e., $\tau_M = \tau_{M'}$) is straightforward and dropped here. □

Example 9. Recall the linear, nondeleting, and total εtdtt M of Example 7, which is not maximally output-separated. After having applied the first item of

the construction to rule ρ_1 we obtain the εtdtt which is shown in Example 4. Since this is still not maximally output-separated, we apply the second item of the construction to the rule $q_1(\sigma(x_1, x_2)) \to \sigma(q(x_1), \sigma(\alpha, q(x_2)))$ and obtain the εtdtt $M' = (\{q, q_1, q_2\}, \Sigma, \Sigma, \{q\}, R')$ with rules

$$\begin{aligned} q(x_1) &\to \gamma(q_1(x_1)) & q_1(\sigma(x_1, x_2)) &\to \sigma(q(x_1), q_2(x_2)) \\ q(\gamma(x_1)) &\to \gamma(q(x_1)) & q_2(x_1) &\to \sigma(\alpha, q(x_1)) \\ q(\alpha) &\to \alpha \ . \end{aligned}$$

The linear and nondeleting εtdtt M' is not in 1-symbol normal form but maximally output-separated. Note that M' is not total.

Whereas the normalization of an εtdtt to one that is maximally output-separated preserves linearity and nondeletion, there exist εtdtts (in fact, even linear, nondeleting tdtts) that admit no equivalent linear, nondeleting εtdtt in 1-symbol normal form.

Example 10. Let $M = (\{q\}, \Sigma, \Delta, \{q\}, R)$ be the linear, nondeleting, and total tdtt such that $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$, $\Delta = \{\delta^{(3)}, \alpha^{(0)}\}$, and R contains the two rules

$$q(\sigma(x_1, x_2)) \to \delta(q(x_1), q(x_2), \alpha) \qquad q(\alpha) \to \alpha \ .$$

Clearly, M is maximally output-separated, but more interestingly, there exists no equivalent linear or nondeleting εtdtt N in 1-symbol normal form. To illustrate this, let N be an equivalent εtdtt in 1-symbol normal form. Obviously, it must have a rule $\rho = l \to r$ that generates the δ in the output. This rule ρ must contain three variables in r, which proves that N is not linear because l is either $p(x_1)$, $p(\sigma(x_1, x_2))$, or $p(\alpha)$ for some state p. Now suppose that there are states q_1, q_2, q_3 and $z_1, z_2, z_3 \in X_2$ such that $r = \delta(q_1(z_1), q_2(z_2), q_3(z_3))$. Then $q_3(t) \Rightarrow_N^* \alpha$ for every $t \in T_\Sigma$. Taking $t = \sigma(\alpha, \alpha)$, it is obvious that this cannot be achieved by a nondeleting εtdtt.

The previous example shows that linearity and nondeletion must be sacrificed to obtain 1-symbol normal form. Indeed we will show in the next theorem that for every εtdtt there exists an equivalent εtdtt in 1-symbol normal form. Naturally, the obtained εtdtt is, in general, nonlinear and deleting (and typically also non-total). Unfortunately, this limits the application in practice because some important operations cannot be applied to nonlinear εtdtts (e.g., computation of the range, image of a regular tree language, etc.).[3]

Theorem 11 (cf. [19, Theorem 5]). *For every εtdtt we can effectively construct an equivalent εtdtt in 1-symbol normal form.*

[3] We thus suggest the following normalization procedure: First transform into maximally output-separated normal form and only then apply the transformation into 1-symbol normal form, if still necessary.

Proof. We present a procedure that iteratively yields the desired εtdtt. Let $M = (Q, \Sigma, \Delta, I, R)$ be an εtdtt that is not in 1-symbol normal form. Thus there exists a rule $l \to r \in R$ such that r contains at least two output symbols (i.e., $r \notin Q(X) \cup \Delta(Q(X))$). Consequently, let $l = p(l')$ for some $p \in Q$ and $l' \in T_\Sigma(X)$, and let $r = \delta(r_1, \dots, r_k)$ for some $k \geq 1$, $\delta \in \Delta_k$, and $r_1, \dots, r_k \in T_\Delta(Q(X))$. Let $q_1, \dots, q_k$ be k new distinct states. We construct the εtdtt $M' = (Q', \Sigma, \Delta, I, R')$ where $Q' = Q \cup \{q_1, \dots, q_k\}$, $R' = (R \setminus \{l \to r\}) \cup \{\rho, \rho_1, \dots, \rho_k\}$, and

$$\begin{aligned} \rho &= p(x_1) \to \delta(q_1(x_1), \dots, q_k(x_1)) \\ \rho_1 &= q_1(l') \to r_1, \quad \dots, \quad \rho_k = q_k(l') \to r_k \ . \end{aligned}$$

Clearly, the right-hand sides of the new rules all contain fewer output symbols than r. Thus, the iteration of the above procedure must eventually terminate with an εtdtt N in 1-symbol normal form. The proof that M and M' are equivalent (i.e., $\tau_M = \tau_{M'}$) is similar to the corresponding proof of Theorem 8. □

Example 12. Recall the linear and nondeleting εtdtt M' from Example 9. The interesting rule is $q_2(x_1) \to \sigma(\alpha, q(x_1))$. According to the construction in the proof of Theorem 11, we replace this rule by the rules

$$q_2(x_1) \to \sigma(q_3(x_1), q_4(x_1)) \quad q_3(x_1) \to \alpha \quad q_4(x_1) \to q(x_1) \ .$$

Thus we obtain the εtdtt $M'' = (\{q, q_1, q_2, q_3, q_4\}, \Sigma, \Sigma, \{q\}, R'')$ with rules

$$\begin{aligned} q(x_1) &\to \gamma(q_1(x_1)) & q_1(\sigma(x_1, x_2)) &\to \sigma(q(x_1), q_2(x_2)) \\ q(\gamma(x_1)) &\to \gamma(q(x_1)) & q_2(x_1) &\to \sigma(q_3(x_1), q_4(x_1)) \\ q(\alpha) &\to \alpha & q_3(x_1) &\to \alpha \\ & & q_4(x_1) &\to q(x_1) \ . \end{aligned}$$

Note that M'' is in 1-symbol normal form, but nonlinear, deleting, and non-total.

5 Composition Construction

This section is devoted to the compositions of tree transformations computed by εtdtts. We start by recalling the classical composition result for tdtts [28,27]. Let M and N be tdtts. If (i) M is deterministic or N is linear, and (ii) M is total or N is nondeleting, then $\tau_M ; \tau_N$ can be computed by a tdtt.

Here we focus on εtdtts, so let M and N be εtdtts. Since deterministic transducers have only few applications in natural language processing, we modify Condition (i) to: (i) N is linear. We will prove that if Conditions (i) and (ii) are fulfilled and, additionally, (iii) M is in 1-symbol normal form, then there exists an εtdtt, denoted by $M ; N$, that computes $\tau_M ; \tau_N$.

Now we define the composition $M ; N$ of M and N. Our principal approach coincides with the one of [27,19]. In fact, if M and N are tdtts, then $M ; N$ is equal to the composition of M and N as defined in [27, p. 195].

To simplify the construction of $M\,;N$, we make the following conventions valid. Let $M = (Q, \Sigma, \Gamma, I_M, R_M)$ and $N = (P, \Gamma, \Delta, I_N, R_N)$ be εtdtts such that all involved ranked alphabets are compatible. Moreover, we do not distinguish between the trees $p(q(t))$ and $(p,q)(t)$ where $p \in P$ and $q \in Q$. Recall that we have defined the rewrite relation of an εtdtt on trees that may contain symbols that are not present in that transducer.

Definition 13 (cf. [19, Definition 9]). *Let M be in 1-symbol normal form. The* composition *of M and N is the εtdtt*

$$M\,;N = (P \times Q, \Sigma, \Delta, I_N \times I_M, R'_M \cup R'_N \cup R'_{M;N})$$

where:

$$\begin{aligned} R'_M &= \{p(l) \to p(r) \mid p \in P, l \to r \in R_M, r \in Q(X)\} \\ R'_N &= \{l[q(x_1)] \to r[q(x_1)] \mid q \in Q, l \to r \in R_N^{\varepsilon}\} \\ R'_{M;N} &= \{p(l) \to r' \mid p \in P, l \to r \in R_M, \exists \rho \in R_N^{\Gamma} \colon p(r) \Rightarrow_N^{\rho} r'\} \ . \end{aligned}$$

Let us discuss the three sets R'_M, R'_N, and $R'_{M;N}$ in detail. The set R'_M contains variants of all rules of R_M that do not contain any output symbol. For each state $p \in P$, such a variant is obtained by annotating the two states (in the left- and right-hand side) by p (i.e., replacing q by (p,q)). The set R'_N contains variants of the ε-rules of R_N; this time annotated with a state $q \in Q$. This is illustrated in Example 14. Finally, the set $R'_{M;N}$ contains rules that are obtained by processing the right-hand side of a rule of R_M that contains an output symbol by an input-consuming rule of R_N. Since M is in 1-symbol normal form, each rule of R_M has at most one output symbol and the rule of R_N will consume it.

The construction trivially preserves linearity and nondeletion. Let us illustrate the construction by composing our running example with the transducer of Example 2.

Example 14. We consider the εtdtt $M'' = (Q, \Sigma, \Sigma, \{q\}, R'')$ of Example 12 (with $Q = \{q, q_1, q_2, q_3, q_4\}$) and the εtdtt $N = (\{p\}, \Sigma, \Sigma, \{p\}, R)$ of Example 2. Note that M'' is in 1-symbol normal form and N is linear and nondeleting. Here we only discuss some of the rules of $M''\,;N$ that are needed to process the input $t = \sigma(\gamma(\alpha), \alpha)$. Since $p(x_1) \to \gamma(p(x_1))$ is in R_N, we have that

$$\{(p,q')(x_1) \to \gamma((p,q')(x_1)) \mid q' \in Q\}$$

is a subset of R'_N. Moreover, by $p(q(x_1)) \Rightarrow_{M''} p(\gamma(q_1(x_1))) \Rightarrow_N \gamma(p(q_1(x_1)))$ we obtain the rule $(p,q)(x_1) \to \gamma((p,q_1)(x_1))$ in $R'_{M'';N}$. Finally, we have

$$p(q_1(\sigma(x_1, x_2))) \Rightarrow_{M''} p(\sigma(q(x_1), q_2(x_2))) \Rightarrow_N \sigma(p(q(x_1)), p(q_2(x_2)))$$

and thus the rule $(p,q_1)(\sigma(x_1,x_2)) \to \sigma((p,q)(x_1),(p,q_2)(x_2))$ in $R'_{M'';N}$. Altogether, we obtain this potential derivation where we write $\Rightarrow$ for $\Rightarrow_{M'';N}$:

$$(p,q)(t) \Rightarrow \gamma((p,q)(t)) \Rightarrow \gamma(\gamma((p,q_1)(t))) \Rightarrow \gamma(\gamma(\sigma((p,q)(\gamma(\alpha)),(p,q_2)(\alpha)))) \ .$$

Continuing also with rules we did not explicitly show, we obtain

$$\begin{aligned} &\Rightarrow \gamma(\gamma(\sigma(\gamma((p,q)(\alpha)),(p,q_2)(\alpha)))) \Rightarrow \gamma(\gamma(\sigma(\gamma(\alpha),(p,q_2)(\alpha)))) \\ &\Rightarrow \gamma(\gamma(\sigma(\gamma(\alpha),\gamma((p,q_2)(\alpha))))) \Rightarrow \gamma(\gamma(\sigma(\gamma(\alpha),\gamma(\alpha)))) \ . \end{aligned}$$

Next, we will consider the correctness of our composition construction of Definition 13. To this end, we prove that $\tau_M ; \tau_N = \tau_{M;N}$ provided that (i) N is linear, (ii) M is total or N is nondeleting, and (iii) M is in 1-symbol normal form. Henceforth we assume the notation of Definition 13.[4]

We start with the easy direction and show that every derivation that first uses exclusively derivation steps of M and then only steps of N can be simulated by a derivation of $M ; N$.

Lemma 15 (cf. [19, Lemma 12]). *Let $\zeta \in P(Q(T_\Sigma))$, $\xi \in P(T_\Gamma)$, and $u \in T_\Delta$ be such that $\zeta \Rightarrow_M^* \xi \Rightarrow_N^* u$. If M is in 1-symbol normal form, then $\zeta \Rightarrow_{M;N}^* u$. In particular, $\tau_M ; \tau_N \subseteq \tau_{M;N}$.*

Proof. The proof can be done by induction on n in the derivation $\zeta \Rightarrow_M^n \xi \Rightarrow_N^* u$ where $\Rightarrow_M^n$ is the n-fold composition of $\Rightarrow_M$. In the induction step, we have to consider two situations, namely whether the first applied rule (of R_M) creates an output symbol or not. If it does not, then we can immediately use the induction hypothesis to conclude $\zeta \Rightarrow_{M;N}^* u$. Otherwise, we need to identify the rules (potentially many because N might copy with the help of ε-rules) in $\xi \Rightarrow_N^* u$ that process the created output symbol. We then apply those rules immediately and use the induction hypothesis. □

Lemma 16 (cf. [19, Lemma 11]). *Let $\zeta \in P(Q(T_\Sigma))$ and $u \in T_\Delta$ be such that $\zeta \Rightarrow_{M;N}^* u$. If (i) N is linear and (ii) M is total or N is nondeleting, then $\zeta \, (\Rightarrow_M^* ; \Rightarrow_N^*) \, u$. In particular, $\tau_{M;N} \subseteq \tau_M ; \tau_N$.*

Proof. We prove the statement by induction on n in the derivation $\zeta \Rightarrow_{M;N}^n u$. Clearly, there are three types (R'_M, R'_N, and $R'_{M;N}$) of rules to distinguish. The case R'_M is trivial. In the other two cases, we defer the derivation step using a rule of R_N. This can only be done, if (i) N is linear (because otherwise N might copy redeces of M) and (ii) M is total or N is nondeleting (because otherwise M can get stuck on a redex that originally was deleted by N). □

Theorem 17. *Let M and N be εtdtts. If (i) N is linear, (ii) M is total or N is nondeleting, and (iii) M is in 1-symbol normal form, then $M ; N$ computes $\tau_M ; \tau_N$.*

Proof. The theorem follows directly from Lemmas 15 and 16. □

The reader might wonder why we keep requirement (iii), although it can always be achieved using Theorem 11. First, totality is not preserved by Theorem 11, and second, if we consider linear εtdtts M and N, then $M ; N$ is also linear. The latter fact can, for example, be used to show that ln-εTOP is closed under left-composition with relabelings [30,31], which are computed by linear, nondeleting top-down tree transducers that are always in 1-symbol normal form.

[4] The following lemmas exhibit an interesting symmetry to the bottom-up case described in [19]. Roughly speaking, the preconditions of Lemmas 15 and 16 are exchanged in the bottom-up case.

In addition, the composition closure of finite-state transductions follows from the linear variant because finite-state transducers can be brought into 1-symbol normal form using the procedure for maximal output-separation.[5]

We note that Theorem 17 generalizes the main theorem of [33], which states that a linear and nondeleting εtdtt in 1-symbol normal form can be composed with a linear and nondeleting εtdtt.

Corollary 18. $\mathrm{ln}{-}\varepsilon\mathrm{TOP}\,;\cdots;\,\mathrm{ln}{-}\varepsilon\mathrm{TOP} \subset \varepsilon\mathrm{TOP}$.

Proof. Inclusion is immediate by Theorems 11 and 17. Strictness follows from the fact that not all transformations of εTOP preserve regularity whereas all transformations of $\mathrm{ln}{-}\varepsilon\mathrm{TOP}$ do [30,25]. □

6 Conclusions and Open Problems

In this paper we have considered εtdtts, which straightforwardly generalize finite-state (string) transducers, and thus, technology developed for the latter model can be embedded into the former one. We have proved that, for two εtdtts M and N, the composition of τ_M and τ_N can be computed by one εtdtt provided that (i) N is linear, (ii) M is total or N is nondeleting, and (iii) M is in 1-symbol normal form. This generalizes Baker's composition result for nondeterministic top-down tree transducers. Moreover, our investigation on composition of εtdtt might give some insight in how to solve the open problem stated in [34,35]: find a tree transducer model that is expressive, modular, inclusive, and trainable.

Another open problem (stated by one of the referees) is the following: given a tree transducer type (such as linear, nondeleting εtdtts) whose class of transformations is not closed under composition (such as ln-εTOP) but ideally has the other three properties (of the list mentioned above), develop an algorithm that accepts two transducers of that type and determines whether their composition can be captured by the same type of transducer (and if it can, return such a transducer).

Acknowledgments The authors gratefully acknowledge the support of the research group around Kevin Knight (ISI/USC). In addition, the authors would like to thank Carmen Heger, who researched compositions of linear and nondeleting εtdtts in [33]. Also, we thank the referees for helpful suggestions, which improved the readability of the paper.

References

1. Hopcroft, J.E., Ullman, J.D.: Introduction to automata theory, languages, and computation. Addison-Wesley (1979)

[5] In principle, we could also use the procedure of Theorem 11, but we would need to prove that, in this special case, linearity and nondeletion are actually preserved.

2. Mohri, M.: Weighted Automata Algorithms. In: Handbook of Weighted Automata. Springer (2009) 209–252
3. Mohri, M., Pereira, F.C.N., Riley, M.: The design principles of a weighted finite-state transducer library. Theoret. Comput. Sci. **231**(1) (2000) 17–32
4. Kaplan, R.M., May, M.: Regular models of phonological rule systems. Computational Linguistics **20**(3) (1994) 331–378
5. Kanthak, S., Ney, H.: Fsa: an efficient and flexible C^{++} toolkit for finite state automata using on-demand computation. In: Proc. ACL. (2004) 510–517
6. Graehl, J.: Carmel: finite-state toolkit. ISI/USC. (1997) available at: `http://www.isi.edu/licensed-sw/carmel/`.
7. Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., Mohri, M.: OpenFst — a general and efficient weighted finite-state transducer library. In: Proc. CIAA. Volume 4783 of LNCS., Springer (2007) 11–23
8. Rounds, W.C.: Mappings and grammars on trees. Math. Syst. Theory **4**(3) (1970) 257–287
9. Thatcher, J.W.: Generalized^2 sequential machine maps. J. Comput. Syst. Sci. **4**(4) (1970) 339–367
10. Arnold, A., Dauchet, M.: Transductions inversibles de forêts. Thèse 3ème cycle M. Dauchet, Université de Lille (1975)
11. Arnold, A., Dauchet, M.: Bi-transductions de forêts. In: Proc. ICALP, Cambridge University Press (1976) 74–86
12. Graehl, J., Knight, K., May, J.: Training tree transducers. Computational Linguistics **34**(3) (2008) 391–427
13. Shabes, Y.: Mathematical and computational aspects of lexicalized grammars. PhD thesis, University of Pennsylvania (1990)
14. Shieber, S.M., Schabes, Y.: Synchronous tree-adjoining grammars. In: Proc. ACL. (1990) 253–258
15. Lilin, E.: Une généralisation des transducteurs d'états finis d'arbres: les S-transducteurs. Thèse 3ème cycle, Université de Lille (1978)
16. Lilin, E.: Propriétés de clôture d'une extension de transducteurs d'arbres déterministes. In: Proc. CAAP. Volume 112 of LNCS., Springer (1981) 280–289
17. Fülöp, Z., Kühnemann, A., Vogler, H.: A bottom-up characterization of deterministic top-down tree transducers with regular look-ahead. Inf. Process. Lett. **91**(2) (2004) 57–67
18. Fülöp, Z., Kühnemann, A., Vogler, H.: Linear deterministic multi bottom-up tree transducers. Theoret. Comput. Sci. **347**(1–2) (2005) 276–287
19. Engelfriet, J., Lilin, E., Maletti, A.: Extended multi bottom-up tree transducers. In: Proc. DLT. Volume 5257 of LNCS., Springer (2008) 289–300
20. Shieber, S.M.: Synchronous grammars as tree transducers. In: Proc. TAG+7. (2004) 88–95
21. Maletti, A.: Compositions of extended top-down tree transducers. Inf. Comput. **206**(9–10) (2008) 1187–1196
22. Knight, K., Graehl, J.: An overview of probabilistic tree transducers for natural language processing. In: Proc. CICLing. Volume 3406 of LNCS., Springer (2005) 1–24
23. Knight, K., May, J.: Applications of Weighted Automata in Natural Language Processing. In: Handbook of Weighted Automata. Springer (2009) 555–580
24. May, J., Knight, K.: Tiburon: A weighted tree automata toolkit. In: Proc. CIAA. Volume 4094 of LNCS., Springer (2006) 102–113
25. Maletti, A., Graehl, J., Hopkins, M., Knight, K.: The power of extended top-down tree transducers. SIAM J. Comput. **39**(2) (2009) 410–430

26. Yamada, K., Knight, K.: A decoder for syntax-based statistical MT. In: ACL. (2002) 303–310
27. Baker, B.S.: Composition of top-down and bottom-up tree transformations. Inform. Control **41**(2) (1979) 186–213
28. Engelfriet, J.: Bottom-up and top-down tree transformations—a comparison. Math. Systems Theory **9**(3) (1975) 198–231
29. Berstel, J.: Transductions and context-free languages. Teubner, Stuttgart (1979)
30. Gécseg, F., Steinby, M.: Tree Automata. Akadémiai Kiadó, Budapest (1984)
31. Gécseg, F., Steinby, M.: Tree languages. In: Handbook of Formal Languages. Volume 3. Springer (1997) 1–68
32. Kuich, W.: Full abstract families of tree series I. In: Jewels Are Forever. Springer (1999) 145–156
33. Heger, C.: Composition of linear and nondeleting top-down tree transducers with ε-rules. Master's thesis, TU Dresden (2008)
34. Knight, K.: Capturing practical natural language transformations. Machine Translation **21**(2) (2007) 121–133
35. Knight, K.: Requirements on an MT system. Personal communication (2008)