THE POWER OF EXTENDED TOP-DOWN TREE TRANSDUCERS*

ANDREAS MALETTI†, JONATHAN GRAEHL‡, MARK HOPKINS§, AND KEVIN KNIGHT¶

Abstract. Extended top-down tree transducers (transducteurs généralisés descendants [Arnold, Dauchet: *Bi-transductions de forêts*. ICALP'76. Edinburgh University Press. 1976]) received renewed interest in the field of Natural Language Processing. Here those transducers are extensively and systematically studied. Their main properties are identified and their relation to classical top-down tree transducers is exactly characterized. The obtained properties completely explain the HASSE diagram of the induced classes of tree transformations. In addition, it is shown that most interesting classes of transformations computed by extended top-down tree transducers are not closed under composition.

Key words. tree transducer, natural language processing, hierarchy, composition closure

AMS subject classifications. 68Q45 (primary), 68T50 (secondary)

1. Introduction. The fields of tree automata (see [16, 17] for surveys) and computational linguistics (see [23, 20] for surveys) were once tightly integrated. For example, top-down tree transducers were devised by THATCHER [27] and ROUNDS [25], the latter of whom wanted to model the transformational grammars of natural language proposed by CHOMSKY [4] in the 1960s. For the most part, the fields subsequently went separate ways. By the 1990s, automata researchers had invented new models based on other concerns. Among these, we find:

- bottom-up tree transducers [28] and attributed tree transducers [15],
- macro tree transducers [5, 13] and modular tree transducers [14],
- tree bimorphisms [2], and various models with synchronization (e.g., [24]).

Computational linguists had returned to using simpler transducer models based on strings. Finite state transducers [19] are straightforward to write down, and it is easy to add probabilities to them and to train them on large quantities of linguistic data. The designer's goal is to capture a set of linguistic pairs of strings, e.g., a set of pairs (s_e, s_f) where s_e is an English sentence and s_f is its French translation. Finite state transducers are also closed under composition, which allows designers to model a complex translation as a cascade of several simpler translations.

However, finite state transducers are not expressive enough for many applications in natural language processing. Recently, computational linguists have turned back to tree transducer models for problems such as machine translation, question answering, and summarization. In these applications, it is important to directly manipulate the hierarchical structure imposed on sentences of natural language, e.g., moving complete syntactical structures or processing different structures in different manners. Here the designer's goal is to capture a set of linguistic pairs of *trees*. For example, we want a formalism to represent pairs (t_e, t_f), where t_e is a parse tree of an English sentence and t_f is a parse tree of its French translation.

In contrast to the string case, computational linguists are faced with a wealth of tree transducer formalisms to choose from. Expressiveness is a critical concern: e.g., is

*Corresponding author. Andreas Maletti. Universitat Rovira i Virgili, Departament de Filologies Romàniques, Av. Catalunya 35, 43002 Tarragona, Spain. E-mail: andreas.maletti@urv.cat

†International Computer Science Institute, Berkeley, USA. This author was supported by a fellowship within the Postdoc-Programme of the German Academic Exchange Service (DAAD).

‡Information Sciences Institute, University of Southern California, USA. E-mail: graehl@isi.edu

§Universität Potsdam, Germany. E-mail: mark.andrew.hopkins@gmail.com

¶Information Sciences Institute, University of Southern California, USA. E-mail: knight@isi.edu

FIG. 1.1. *Example of a local rotation.*

the model powerful enough to capture what is observed in empirical translation data? Other concerns include representational succinctness and computational complexity.

SHIEBER [26] and others have argued that top-down tree transducers [27, 25] are generally inadequate for linguistic tasks because they cannot straightforwardly model local rotation, i.e., the reordering of tree components at differing tree depths. An example of such a rotation is shown in Fig. 1.1. Top-down tree transducers are frowned upon because they must copy and delete to model such tree transformations, and general copying causes many operations to become intractable or impossible.

To address these drawbacks, a generalization of top-down tree transducers, called *extended top-down tree transducers*, was originally conceived by [25] and has been further pursued by [6, 1, 18, 21]. The extended transducer does not suffer from the representational inadequacies outlined above, and thus has emerged as a promising candidate for use in linguistic tasks. Rules in an extended transducer have left-hand sides with multiple symbols (not counting variables like $x_1, x_2, \dots$). For example, to model Fig. 1.1, we write an extended transducer rule like this:

$$q(\mathrm{S}(x_1, \mathrm{SINV}(x_2, x_3))) \to \mathrm{S}(q(x_3), \mathrm{VP}(q(x_2), q(x_1))) \ .$$

We contrast this with a top-down tree transducer (which has exactly one symbol on the left-hand sides), where we must employ copying and deleting:

$$\begin{aligned} q(\mathrm{S}(x_1, x_2)) &\to \mathrm{S}(r(x_2), \mathrm{VP}(s(x_2), q(x_1))) \\ r(\mathrm{SINV}(x_1, x_2)) &\to q(x_2) \\ s(\mathrm{SINV}(x_1, x_2)) &\to q(x_1) \ . \end{aligned}$$

Rules in an extended transducer may also have no symbol on the left-hand side, e.g.: $q(x_1) \to \mathrm{S}(r(x_1))$.

This paper provides a first thorough analysis of extended top-down tree transducers. The goal is an exact characterization of the power of extended tree transducers in terms of the power of top-down tree transducers. To this end, we identify three key properties that we label (X1) *finite look-ahead*, (X2) *deep attachment of variables*, and (X3) *infinitely many outputs for one input*. Actually, finite look-ahead is a weak form of the bottom-up property: *checking followed by deletion* [9]. Deep attachment of variables will allow us to specify local rotations easily. Finally, the last property makes it possible to have nondeterministic choice independent of input symbols.

It is known that linear (i.e., non-copying) top-down tree transducers do not have these features. They can only be simulated at the expense of either (a) adding them explicitly (e.g., top-down tree transducers with regular look-ahead [10]) or (b) dropping the linearity condition. We demonstrate under which circumstances these formal properties make extended transducers more expressive than their non-extended counterparts. In fact, we exactly characterize the inclusion relations between various classes of transformations computed by extended top-down tree transducers (Fig. 4.5),

thereby proving that the three mentioned properties are indeed the distinguishing features of extended top-down tree transducers.

Two simple classes of transformations, namely flattenings and shuffles, which are essential in the applications in Natural Language Processing, are formally introduced. These simple classes of transformations are among those computable by extended top-down tree transducers that do not employ copying or deleting ("linear and nondeleting"). It is shown that no class of transformations computable by linear extended top-down-tree transducers is closed under composition if it contains at least one of the two simple classes of transformations, and all transformations computed by linear and nondeleting top-down tree transducers. For flattenings this was essentially shown by Arnold and Dauchet [1, 2]. Unfortunately, this negative result implies that most interesting classes of transformations computed by extended top-down tree transducers are not closed under composition.

The paper is structured as follows. Section 2 recalls basic notions and the notion of extended top-down tree transducers with regular look-ahead. In Sect. 3, we present an example of an extended top-down tree transducer and highlight the short-comings of top-down tree transducers. We proceed in Sect. 4 with a detailed study of the expressive power of extended tree transducers. In this section, we derive the properties (X1), (X2), and (X3). Section 5 is devoted to the closure under composition. We end the paper with Conclusions in Sect. 6.

2. Preliminaries.

2.1. Sets, relations, and trees. We denote the set of nonnegative integers by $\mathbb{N}$. For every $n \in \mathbb{N}$, the subset $\{1, 2, \dots, n\}$ is denoted by $[n]$. We fix the set $X = \{x_1, x_2, \dots\}$ of (formal) variables and let $X_n = \{x_i \mid i \in [n]\}$ for every $n \in \mathbb{N}$. Now, let A, B, and C be sets. A *relation from A to B* is a subset of $A \times B$. Let $R \subseteq A \times B$ and $R' \subseteq B \times C$. The *inverse relation of R*, denoted by R^{-1}, is $\{(b, a) \mid (a, b) \in R\}$ and the *composition of R and R'*, denoted by $R\,;R'$, is $\{(a, c) \mid \exists b \in B\colon (a, b) \in R, (b, c) \in R'\}$. These notions extend to classes of relations in the standard manner. A *relation on A* is a subset of $A \times A$. For every $L \subseteq A$ we denote by id_L the relation $\{(a, a) \mid a \in L\}$. The reflexive and transitive closure of a relation $R \subseteq A \times A$ is denoted by R^*. Finally, by $\mathcal{P}(A)$ we denote the power set (i.e., the set of all subsets) of A.

An *alphabet* is a nonempty and finite set, of which the elements are called symbols. The set of all finite sequences (words) over a set Σ is denoted by Σ^* where ε denotes the empty sequence (the empty word). We denote concatenation of words u and w by $u.w$ or simply by the juxtaposition uw. The length of w is denoted by $|w|$.

Let Σ be an alphabet and $\mathrm{rk}\colon \Sigma \to \mathbb{N}$. Then (Σ, rk) is a *ranked alphabet*, and a symbol $\sigma \in \Sigma$ has *rank* $\mathrm{rk}(\sigma)$. In the sequel, we simply write Σ instead of (Σ, rk), whenever the mapping rk is clear from the context or arbitrary. Moreover, we denote the set $\{\sigma \in \Sigma \mid \mathrm{rk}(\sigma) = k\}$ by Σ_k for every $k \in \mathbb{N}$. Whenever we want to make the rank of a symbol $\sigma \in \Sigma$ explicit, we write $\sigma^{(k)}$ where $k = \mathrm{rk}(\sigma)$. Thus, $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$ defines a ranked alphabet with a binary (i.e., rank 2) symbol σ and a nullary (i.e., rank 0) symbol α.

Let Σ be a ranked alphabet and V a set. The set of *Σ-trees indexed by V*, denoted by $T_\Sigma(V)$, is the smallest set T such that (i) $V \subseteq T$ and (ii) for every $\sigma \in \Sigma_k$ and $t_1, \dots, t_k \in T$ also $\sigma(t_1, \dots, t_k) \in T$. We generally assume that $\Sigma \cap V = \emptyset$ and thus write $\alpha()$ simply as α for every $\alpha \in \Sigma_0$. Given $\gamma \in \Sigma_1$, we abbreviate $\gamma(\gamma(\cdots\gamma(t)\cdots))$ with n symbols γ on top of $t \in T_\Sigma(V)$ simply by $\gamma^n(t)$. Finally, we write T_Σ for $T_\Sigma(\emptyset)$.

The set of *positions* of $t \in T_\Sigma(V)$, denoted by $\mathrm{pos}(t) \subseteq \mathbb{N}^*$, is inductively defined by $\mathrm{pos}(v) = \{\varepsilon\}$ for every $v \in V$ and

$$\mathrm{pos}(\sigma(t_1, \dots, t_k)) = \{\varepsilon\} \cup \{i.w \mid i \in [k], w \in \mathrm{pos}(t_i)\}$$

for every $\sigma \in \Sigma_k$ and $t_1, \dots, t_k \in T_\Sigma(V)$. Let $t, t' \in T_\Sigma(V)$ and $w \in \mathrm{pos}(t)$. The label of t at position w is denoted by $\mathrm{lab}_t(w)$, and the subtree of t that is rooted at w is denoted by $\mathrm{sub}_t(w)$. These notions can be defined inductively by $\mathrm{lab}_v(\varepsilon) = v$ and $\mathrm{sub}_v(\varepsilon) = v$ for every $v \in V$ and

$$\mathrm{lab}_{\sigma(t_1,\dots,t_k)}(w) = \begin{cases} \sigma & \text{if } w = \varepsilon \\ \mathrm{lab}_{t_i}(w') & \text{if } w = i.w' \text{ for some } i \in [k] \end{cases}$$

$$\mathrm{sub}_{\sigma(t_1,\dots,t_k)}(w) = \begin{cases} \sigma(t_1, \dots, t_k) & \text{if } w = \varepsilon \\ \mathrm{sub}_{t_i}(w') & \text{if } w = i.w' \text{ for some } i \in [k] \end{cases}$$

for every $\sigma \in \Sigma_k$ and $t_1, \dots, t_k \in T_\Sigma(V)$. For every $Z \subseteq \Sigma \cup V$ we write $\mathrm{pos}_Z(t)$ for $\{w \in \mathrm{pos}(t) \mid \mathrm{lab}_t(w) \in Z\}$. If $Z = \{z\}$, then we simply write $\mathrm{pos}_z(t)$ instead of $\mathrm{pos}_Z(t)$. We say that $z \in Z$ *occurs* $\mathrm{card}(\mathrm{pos}_z(t))$ times in t. Finally, $t[t']_w$ denotes the tree that is obtained from t by replacing the subtree $\mathrm{sub}_t(w)$ at w by t'.

The *height* of t, which is denoted by $\mathrm{ht}(t)$, is $\max\{|w| + 1 \mid w \in \mathrm{pos}(t)\}$. By $\mathrm{var}(t)$ we denote the set $\{v \in V \mid \mathrm{pos}_v(t) \neq \emptyset\}$, and every $v \in \mathrm{var}(t)$ *occurs* in t. The tree t is *V-linear* (respectively, *V-nondeleting*), if every $v \in V$ occurs at most (respectively, at least) once in t. For every V-linear $t \in T_\Sigma(V)$ and $v \in \mathrm{var}(t)$, we identify the unique element of $\mathrm{pos}_v(t)$ with $\mathrm{pos}_v(t)$.

Next we recall substitution of trees. Let $\theta \colon V \to T_\Sigma(V)$. Then $t\theta$ is inductively defined for every $v \in V$ by $v\theta = \theta(v)$ and for every $\sigma \in \Sigma_k$ and $t_1, \dots, t_k \in T_\Sigma(V)$ by $\sigma(t_1, \dots, t_k)\theta = \sigma(t_1\theta, \dots, t_k\theta)$. Occasionally, we also write $t[v \leftarrow \theta(v) \mid v \in V]$ for $t\theta$ to avoid the explicit definition of θ. In the special case that $V = X_n$ for some $n \in \mathbb{N}$, we also write $t[\theta(x_1), \dots, \theta(x_n)]$ for $t\theta$. Finally, let us recall *OI-substitution* of sets of trees. Now, let $\theta \colon V \to \mathcal{P}(T_\Sigma(V))$. Then $v\theta = \theta(v)$ for every $v \in V$ and $\sigma(t_1, \dots, t_k)\theta = \{\sigma(u_1, \dots, u_k) \mid u_1 \in t_1\theta, \dots, u_k \in t_k\theta\}$ for every $\sigma \in \Sigma_k$ and $t_1, \dots, t_k \in T_\Sigma(V)$. For $L \subseteq T_\Sigma(V)$ we define $L\theta = \bigcup_{t \in L} t\theta$. In the special case that $V = X_n$ for some $n \in \mathbb{N}$, we also write $t[\theta(x_1), \dots, \theta(x_n)]$ and $L[\theta(x_1), \dots, \theta(x_n)]$ for $t\theta$ and $L\theta$, respectively.

Next, we introduce two essential concepts. Two trees $s, t \in T_\Sigma(X)$ *match* (or: *are unifiable*) if $\mathrm{lab}_s(w) = \mathrm{lab}_t(w)$ for every $w \in \mathrm{pos}_\Sigma(s) \cap \mathrm{pos}_\Sigma(t)$. For example, $\sigma(x_1, x_1)$ matches $\sigma(\alpha, \beta)$ with $\sigma \in \Sigma_2$ and $\alpha, \beta \in \Sigma_0$ even though $\alpha \neq \beta$. Now, suppose that s and t match, and let $W \subseteq \mathrm{pos}_X(s) \cap \mathrm{pos}(t)$ be a set of common positions that are labeled by a variable in s. Then the *W-combination* of s and t, denoted by $s \bowtie_W t$, is defined by $s \bowtie_W t = (\cdots((s[\mathrm{sub}_t(w_1)]_{w_1})[\mathrm{sub}_t(w_2)]_{w_2})\cdots)[\mathrm{sub}_t(w_n)]_{w_n}$ where $W = \{w_1, \dots, w_n\}$. Thus,

$$\sigma(x_1, x_1) \bowtie_{\{1\}} \sigma(\alpha, \beta) = \sigma(\alpha, x_1) \quad \text{and} \quad \sigma(x_1, x_1) \bowtie_{\{1,2\}} \sigma(\alpha, \beta) = \sigma(\alpha, \beta) \ .$$

Note that $\mathrm{ht}(s \bowtie_W t) \leq \max(\mathrm{ht}(s), \mathrm{ht}(t))$. If $W = \mathrm{pos}_X(s) \cap \mathrm{pos}(t)$, then we simply write $\bowtie$ instead of $\bowtie_W$ and call $s \bowtie t$, which in that case is equal to $t \bowtie s$ (up to a renaming of variables), a *(least) unifier* of s and t.

2.2. Tree languages and tree transformations. Any subset of $T_\Sigma(V)$ is a *tree language*. Next we identify two important classes of tree languages. A tree

TABLE 2.1
Classes of tree transformations.

Class of transformations computed by	$\mathrm{XTOP}^{\mathrm{R}}$ $\mathrm{xtt}^{\mathrm{R}}$	$\mathrm{XTOP}^{\mathrm{F}}$ $\mathrm{xtt}^{\mathrm{F}}$	XTOP xtt	$\mathrm{TOP}^{\mathrm{R}}$ $\mathrm{tdtt}^{\mathrm{R}}$	$\mathrm{TOP}^{\mathrm{F}}$ $\mathrm{tdtt}^{\mathrm{F}}$	TOP tdtt

language $L \subseteq T_\Sigma$ has *finite depth* if there exists a finite set $T \subseteq T_\Sigma(X_1)$ such that $L = T[T_\Sigma]$. For example, T_Σ has finite depth because $T_\Sigma = \{x_1\}[T_\Sigma]$.

The recognizable tree languages are defined via a class of automata. A *(deterministic) bottom-up tree automaton* [16] (dta, for short) is a tuple $M = (Q, \Sigma, F, \delta)$ such that Q is an alphabet of *states*, Σ is a ranked alphabet of *input symbols*, $F \subseteq Q$ is a set of *final states*, and $\delta = (\delta_\sigma)_{\sigma\in\Sigma}$ is a family of *transition mappings* with $\delta_\sigma \colon Q^k \to Q$ for every $\sigma \in \Sigma_k$. We extend δ to a mapping $\widehat{\delta} \colon T_\Sigma \to Q$ as follows:

$$\widehat{\delta}(\sigma(t_1, \ldots, t_k)) = \delta_\sigma(\widehat{\delta}(t_1), \ldots, \widehat{\delta}(t_k))$$

for every $\sigma \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma$. For every $q \in Q$ we denote $\{t \in T_\Sigma \mid \widehat{\delta}(t) = q\}$ by $L(M)_q$. The tree language *accepted by* M is $L(M) = \bigcup_{q\in F} L(M)_q$. A tree language L is *recognizable* (or: *regular*) if there exists a dta M such that $L = L(M)$. We note that every finite depth tree language is recognizable because every finite tree language is recognizable, T_Σ is recognizable, and the recognizable tree languages are closed under OI-substitution [16, Theorem II.4.6].

Next, we recall the central concept of an extended top-down tree transducer [6, 1, 18, 21] (we follow the definitions of [22]). We immediately add regular look-ahead [10] to the device in order to present a model that generalizes all required transducing devices. Let Q be a finite set, Σ and Δ be ranked alphabets, and $T \subseteq T_\Sigma(X)$. By $Q(T)$ we denote the set $\{q(t) \mid q \in Q, t \in T\}$. The variable $x \in X$ *occurs* in $u \in T_\Delta(Q(T))$ if there exist $q \in Q$ and $t \in T$ such that (i) x occurs in t and (ii) $q(t)$ occurs in u. An *extended top-down tree transducer with regular look-ahead* ($\mathrm{xtt}^{\mathrm{R}}$, for short) is a tuple $M = (Q, \Sigma, \Delta, I, R, c)$ where

(i) Q is an alphabet of *states*,
(ii) Σ and Δ are ranked alphabets of *input* and *output symbols*,
(iii) $I \subseteq Q$ is a set of *initial states*,
(iv) R is a finite set of *rules* of the form $q(t) \to u$ with $q \in Q$ and $t \in T_\Sigma(X)$ X-linear and $u \in T_\Delta(Q(X))$ such that every $x \in X$ that occurs in u also occurs in t, and
(v) $c \colon R \to \mathcal{P}(T_\Sigma)$ is a *look-ahead restriction* such that $c(r)$ is recognizable for every $r \in R$.

In the sequel, we often simply write $(q(t) \to u) \in R$ and understand that $q \in Q$, $t \in T_\Sigma(X)$, and $u \in T_\Delta(Q(X))$. For every $r = (q(t) \to u) \in R$ we denote by $\mathrm{del}(r)$ the set of positions $w \in \mathrm{pos}_X(t)$ such that $\mathrm{lab}_t(w)$ does not occur in u. Since every rule $(q(t) \to u) \in R$ has an X-linear left-hand side t, for every $x \in \mathrm{var}(t)$ we occasionally identify $\mathrm{pos}_x(t)$ with its unique element.

The semantics of $\mathrm{xtt}^{\mathrm{R}}$ is given by rewriting [10]. For the sake of simplicity, let us assume that Q is disjoint with both Σ and Δ. Now, let $\zeta, \xi \in T_\Delta(Q(T_\Sigma))$ and $r = (q(t) \to u) \in R$. We say that ζ *rewrites to* ξ *using* r, denoted by $\zeta \Rightarrow^r_M \xi$, if there exist a position $w \in \mathrm{pos}(\zeta)$ and a substitution $\theta \colon \mathrm{var}(t) \to T_\Sigma$ such that

(i) $\mathrm{sub}_\zeta(w) = q(t\theta)$,
(ii) $t\theta \in c(r)$, and
(iii) $\xi = \zeta[u']_w$ where $u' = u[q(x) \leftarrow q(x\theta) \mid q \in Q, x \in \mathrm{var}(t)]$.

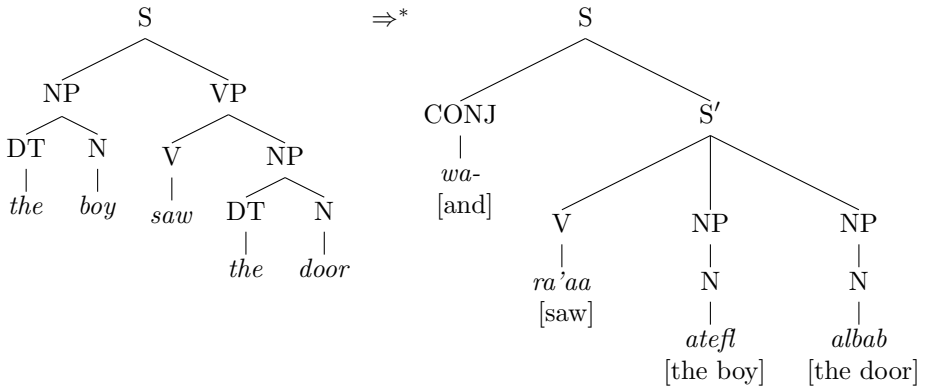

FIG. 3.1. *English-to-Arabic translation on syntax trees.*

Condition (ii) is called the *look-ahead check*. Let $\Rightarrow_M$ be $\bigcup_{r\in R} \Rightarrow_M^r$. The *tree transformation computed by* M is $\tau_M = \{(t,u) \in T_\Sigma \times T_\Delta \mid \exists q \in I\colon q(t) \Rightarrow_M^* u\}$. Two xtt^R M and M' are *equivalent* if $\tau_M = \tau_{M'}$. The class of tree transformations computed by xtt^R is denoted by XTOP^R. We defer examples of xtt^R to the next section.

Let $M = (Q, \Sigma, \Delta, I, R, c)$ be an xtt^R. We say that M has *finite look-ahead* if $c(r)$ has finite depth for every $r \in R$, and we say that M has *no look-ahead* if $c(r) = T_\Sigma$ for every $r \in R$. We use xtt^F as an abbreviation for xtt^R with finite look-ahead and xtt as an abbreviation for xtt^R with no look-ahead. For xtt we drop the look-ahead restriction c and just write $(Q, \Sigma, \Delta, I, R)$. Moreover, M is a *top-down tree transducer* [27, 25] (for short: tdtt) if all rules $r \in R$ have the form $r = (q(\sigma(x_1, \ldots, x_k)) \to u)$ for some $q \in Q$, $\sigma \in \Sigma_k$, and $u \in T_\Delta(Q(X))$. We use the abbreviations tdtt^R, tdtt^F, and tdtt with the obvious meaning.

Since we deal with finite state devices here, we generally assume that the look-ahead restriction of each xtt^R is given by dta (respectively, finite subsets of $T_\Sigma(X_1)$) for regular (respectively, finite) look-ahead. We use the denotations listed in Table 2.1 for classes of transformations.

Next we define some more properties of xtt^R. The xtt^R M is *linear* (respectively, *nondeleting*) if for every rule $(q(t) \to u) \in R$ and $x \in \mathrm{var}(t)$ there exist at most (respectively, at least) one $q \in Q$ and $w \in \mathrm{pos}(u)$ such that $\mathrm{lab}_u(w) = q(x)$. Finally, M is *input-ε-free* if no rule $r \in R$ is of the form $q(x) \to u$ for some $q \in Q$, $x \in X$, and $u \in T_\Delta(Q(X))$. We use the prefixes "l", "n", and "e" to restrict the introduced classes of transformations to those computed by linear, nondeleting, and input-ε-free devices, respectively. Combinations of prefixes are also allowed with the obvious effect. Thus, ln-TOP denotes the class of all transformations computed by linear and nondeleting tdtt.

Finally, we introduce two types of tree transformations. A tree transformation $\tau\colon T_\Sigma \to T_\Delta$ is a *flattening* (respectively, a *shuffle*) if there exists a linear and nondeleting input-ε-free xtt $M = (Q, \Sigma, \Delta, I, R)$ such that $\tau_M = \tau$ and $\mathrm{card}(\mathrm{pos}_\Delta(u)) = 1$ [respectively, $\mathrm{card}(\mathrm{pos}(t)) = \mathrm{card}(\mathrm{pos}(u))$] for every rule $(q(t) \to u) \in R$. We denote the classes of all flattenings and shuffles by FLAT and SHUF, respectively.

3. A complete example. Figure 3.1 shows a tree transformation from the realm of English-to-Arabic translation. We desire a collection of rules that allow us to transform the English input tree into the Arabic output tree, in a top-down fashion.

One such rule set R is:

$$q(x_1) \to q_{\mathrm{S}}(x_1) \tag{r_1}$$
$$q(x_1) \to \mathrm{S}(\mathrm{CONJ}(wa\text{-}), q_{\mathrm{S}}(x_1)) \tag{r_2}$$
$$q_{\mathrm{S}}(\mathrm{S}(x_1, \mathrm{VP}(x_2, x_3))) \to \mathrm{S}'(q_{\mathrm{V}}(x_2), q_{\mathrm{NP}}(x_1), q_{\mathrm{NP}}(x_3)) \tag{r_3}$$
$$q_{\mathrm{V}}(\mathrm{V}(saw)) \to \mathrm{V}(ra'aa) \tag{r_4}$$
$$q_{\mathrm{NP}}(\mathrm{NP}(\mathrm{DT}(the), \mathrm{N}(boy))) \to \mathrm{NP}(\mathrm{N}(ateft)) \tag{r_5}$$
$$q_{\mathrm{NP}}(\mathrm{NP}(\mathrm{DT}(the), \mathrm{N}(door))) \to \mathrm{NP}(\mathrm{N}(albab)) \tag{r_6}$$

These six rules make up part of a linear and nondeleting xtt $M = (Q, \Sigma, \Delta, I, R)$ with

- $Q = \{q, q_{\mathrm{S}}, q_{\mathrm{V}}, q_{\mathrm{NP}}\}$,
- $\Sigma = \{\mathrm{S}^{(2)}, \mathrm{NP}^{(2)}, \mathrm{VP}^{(2)}, \mathrm{DT}^{(1)}, \mathrm{N}^{(1)}, \mathrm{V}^{(1)}, the^{(0)}, boy^{(0)}, saw^{(0)}, door^{(0)}\}$,
- $\Delta = \{\mathrm{S}^{(2)}, \mathrm{S}'^{(2)}, \mathrm{CONJ}^{(1)}, \mathrm{V}^{(1)}, \mathrm{N}^{(1)}, \mathrm{NP}^{(1)}, wa\text{-}^{(0)}, ra'aa^{(0)}, ateft^{(0)}, albab^{(0)}\}$, and
- $I = \{q\}$.

When applied in proper sequence, they are sufficient to execute the desired transformation:

$$\begin{aligned}
& q(\mathrm{S}(\mathrm{NP}(\mathrm{DT}(the), \mathrm{N}(boy)), \mathrm{VP}(\mathrm{V}(saw), \mathrm{NP}(\mathrm{DT}(the), \mathrm{N}(door))))) \\
\Rightarrow_M^{r_2}\ & \mathrm{S}(\mathrm{CONJ}(wa\text{-}), q_{\mathrm{S}}(\mathrm{S}(\mathrm{NP}(\mathrm{DT}(the), \mathrm{N}(boy)), \mathrm{VP}(\mathrm{V}(saw), \\
& \mathrm{NP}(\mathrm{DT}(the), \mathrm{N}(door)))))) \\
\Rightarrow_M^{r_3}\ & \mathrm{S}(\mathrm{CONJ}(wa\text{-}), \mathrm{S}'(q_{\mathrm{V}}(\mathrm{V}(saw)), q_{\mathrm{NP}}(\mathrm{NP}(\mathrm{DT}(the), \mathrm{N}(boy))), \\
& q_{\mathrm{NP}}(\mathrm{NP}(\mathrm{DT}(the), \mathrm{N}(door))))) \\
\Rightarrow_M^{r_4}\ & \mathrm{S}(\mathrm{CONJ}(wa\text{-}), \mathrm{S}'(\mathrm{V}(ra'aa), q_{\mathrm{NP}}(\mathrm{NP}(\mathrm{DT}(the), \mathrm{N}(boy))), \\
& q_{\mathrm{NP}}(\mathrm{NP}(\mathrm{DT}(the), \mathrm{N}(door))))) \\
\Rightarrow_M^{r_5}\ & \mathrm{S}(\mathrm{CONJ}(wa\text{-}), \mathrm{S}'(\mathrm{V}(ra'aa), \mathrm{NP}(\mathrm{N}(ateft)), q_{\mathrm{NP}}(\mathrm{NP}(\mathrm{DT}(the), \mathrm{N}(door))))) \\
\Rightarrow_M^{r_6}\ & \mathrm{S}(\mathrm{CONJ}(wa\text{-}), \mathrm{S}'(\mathrm{V}(ra'aa), \mathrm{NP}(\mathrm{N}(ateft)), \mathrm{NP}(\mathrm{N}(albab))))
\end{aligned}$$

Since $q \in I$, the pair of trees displayed in Fig. 3.1 is in the tree transformation τ_M computed by M.

There are several linguistic facts to account for here. First, there is the word-order inversion from English (subject-verb-object) to Arabic (verb-subject-object), which happens regardless of the sizes of the input subject, verb, and object subtrees. This general inversion is captured in rule r_3. Such a local rotation is hard to capture with top-down tree transducers, requiring copying and deleting. Rule r_3 accomplishes the task through the *deep attachment of variables* in its left-hand side.

Second is the insertion of Arabic "*wa-*" (meaning "*and*"). Sentences in Arabic often begin with this particle, which does not appear in English. Rules r_1 and r_2 model the decision a translator faces, to either insert "*wa-*" or not. Input-epsilon rules (i.e., rules that have no input symbol on their left-hand side such as r_1 and r_2) are very convenient here, and while it may be possible to place a bound on the output and eliminate the input-epsilon rules, this would come at the cost of creating a more complex transducer. Moreover, a probabilistic version of this transducer requires that probabilities for r_1 and r_2 sum to one, and this would have to be maintained during epsilon removal.

Third is the translation of the two-word phrase "*the boy*" into one word "*ateft*". This is problematic for nondeleting top-down transducers. A rule with (English) left-hand side $q_{\mathrm{NP}}(\mathrm{NP}(x_1, x_2))$ can only create the monadic (Arabic) output $\mathrm{NP}(\mathrm{N}(\dots))$

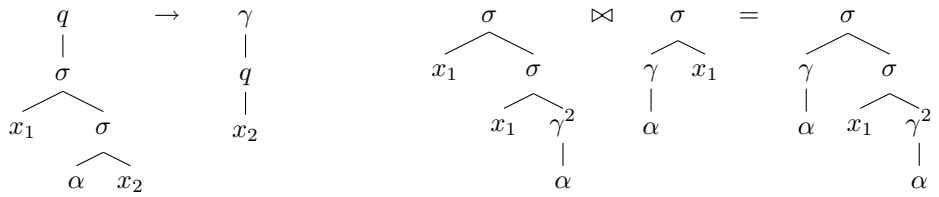

FIG. 4.1. *Example rule r, look-ahead obligation s, look-ahead restriction t_r, and unifier (see Theorem 4.1).*

at the expense of deletion. Moreso, *finite look-ahead* is needed to properly restrict the application of such a rule.

Of course, full-scale language translation involves many more complexities, but as we can see, this example already raises some theoretical issues related to the current paper.

4. Expressive Power. In this section, we explore the expressive power of extended top-down tree transducers. We provide a complete characterization of the inclusion relations (see Fig. 4.5) of the introduced classes of transformations. In particular, we relate $\mathrm{xtt}^{\mathrm{R}}$ and its restricted variants to the well-known $\mathrm{tdtt}^{\mathrm{R}}$ and its restricted variants. Except for Theorem 4.1 we only consider the nondeletion property in conjunction with the linearity property. In total, we investigate 18 classes of transformations computed by $\mathrm{xtt}^{\mathrm{R}}$ and 9 classes of transformations computed by $\mathrm{tdtt}^{\mathrm{R}}$.

In order to present concise results, we establish one more notation. We write inequalities (or equalities) with some of the prefixes in brackets (e.g., see Theorem 4.1). Such inequalities shall represent several inequalities in the following way. For each prefix π in brackets, we either opt to take the prefix and then consistently replace $[\pi]$ by π throughout or to omit it and remove $[\pi]$. Let us note that the consistent replacement is important, i.e., we either take all occurrences of the same prefix or omit all of them. For example, ln-XTOP = ln-$\mathrm{XTOP}^{\mathrm{F}}$ is an instance of the statement of Theorem 4.1 whereas ln-XTOP = l-$\mathrm{XTOP}^{\mathrm{F}}$ is not.

Our first result concerns finite look-ahead. Namely, we show that the expressive power of xtt and $\mathrm{xtt}^{\mathrm{F}}$ coincides. In essence, this shows that the non-shallow left-hand sides enable xtt to perform finite look-ahead. We call this intrinsic property of xtt: *xtt have finite look-ahead*. Now let us discuss how to prove the statement. The key idea is to store the "still-to-be-checked" look-ahead, called look-ahead obligation, into the state. The initial states have no look-ahead obligation, so their look-ahead obligation should be x_1, which matches every tree. Now suppose that we are in a state q with look-ahead obligation s. Further suppose we want to execute a rule $r \in R$ with look-ahead restriction $t_r \in T_\Sigma(X_1)$. If s and t_r do not match, then the rule r cannot be applied now. Otherwise, we first combine the look-ahead obligation s with the look-ahead restriction t_r. In fact, $s \bowtie t_r$ represents the combined look-ahead. Then we match the combined look-ahead with the left-hand side of the rule. If the match does not succeed, then the rule cannot be applied. The principal idea now is to reduce the combined look-ahead by the part matched to the left-hand side and distribute the rest as look-ahead obligation to the states in the right-hand side of the rule. The only problem are deletions because then we cannot delegate the look-ahead obligation. Thus, we extend the left-hand side, so that such look-ahead obligations are always met.

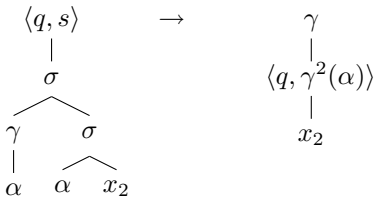

FIG. 4.2. *Resulting rule (see Theorem 4.1).*

Let us illustrate the above approach on a quick example. Suppose that

$$r = q(\sigma(x_1, \sigma(\alpha, x_2))) \to \gamma(q(x_2))$$

is a rule with look-ahead restriction $t_r = \sigma(\gamma(\alpha), x_1)$. Moreover, suppose that we want to construct the rule with look-ahead obligation $s = \sigma(x_1, \sigma(x_1, \gamma^2(\alpha)))$. Then we realize that the look-ahead obligation matches the look-ahead restriction and the unifier is $\sigma(\gamma(\alpha), \sigma(x_1, \gamma^2(\alpha)))$; we display those trees in Fig. 4.1. Clearly, this unifier matches the input tree in the left-hand side of the rule. However, the variable x_1 is deleted in the example rule, so we resolve the look-ahead and replace x_1 by $\gamma(\alpha)$. Moreover, we see that the look-ahead $\gamma^2(\alpha)$ needs to be forwarded as look-ahead obligation to the call of the state q in the right-hand side. Thus, we construct the rule $\langle q, s\rangle(\sigma(\gamma(\alpha), \sigma(\alpha, x_2))) \to \gamma(\langle q, \gamma^2(\alpha)\rangle(x_2))$, which is depicted in Fig. 4.2. Note that we use $\langle q, s\rangle$ instead of (q, s) for improved readability. In general, if the states of an $\mathrm{xtt}^{\mathrm{R}}$ are tuples, then we write $\langle a_1, \ldots, a_n\rangle$ instead of $(a_1, \ldots, a_n)$ to increase readability.

THEOREM 4.1 (Extended tree transducers have finite look-ahead).

$$[\mathrm{l}]\,[\mathrm{n}]\,[\mathrm{e}]\text{-}\mathrm{XTOP} = [\mathrm{l}]\,[\mathrm{n}]\,[\mathrm{e}]\text{-}\mathrm{XTOP}^{\mathrm{F}}$$

Proof. One inclusion is trivial. For the other direction, let $M = (Q, \Sigma, \Delta, I, R, c)$ be an $\mathrm{xtt}^{\mathrm{F}}$. Since M has finite look-ahead, $c(r)$ has finite depth for every rule $r \in R$. Moreover, for every $r \in R$ let $T_r \subseteq T_\Sigma(X_1)$ be a finite set such that $c(r) = T_r[T_\Sigma]$. Without loss of generality, suppose that T_r is a singleton for every $r \in R$, i.e., there exists a tree $t_r \in T_\Sigma(X_1)$ such that $T_r = \{t_r\}$. If this condition is not met, then we can split the rule $r \in R$ with $T_r = \{t_1, \ldots, t_n\}$ into n copies $r_1, \ldots, r_n$. Moreover, we assume, without loss of generality, that $x_1 \notin \mathrm{var}(t)$ for every $(q(t) \to u) \in R$.

Let $n = \max\{\mathrm{ht}(t_r) \mid r \in R\}$ and $S = \{s \in T_\Sigma(X_1) \mid \mathrm{ht}(s) \leq n\}$. Clearly, S is finite. We construct the xtt $M' = (Q \times S, \Sigma, \Delta, I \times \{x_1\}, R')$ as follows. For every rule $r = (q(t) \to u) \in R$ and $s \in S$ such that s matches t_r and $s' = s \bowtie t_r$ matches t, we construct one rule $(\langle q, s\rangle(t') \to u') \in R'$ where t' and u' are constructed as follows:

(i) Let $W = \mathrm{del}(r) \cap \mathrm{pos}(s')$ [see the paragraph below the definition of $\mathrm{xtt}^{\mathrm{R}}$ in Section 2.2 for the definition of $\mathrm{del}(r)$] and $t'' = t \bowtie_W s'$. Note that $x_1 \notin \mathrm{var}(t)$. We obtain t' from t'' by simply renaming each occurrence of x_1 in t'' to a variable $x \in X$ such that t' is X-linear.

(ii) For every $x \in \mathrm{var}(t)$ let

$$\mathrm{restr}_{s',t}(x) = \begin{cases} \mathrm{sub}_{s'}(w) & \text{if } w \in \mathrm{pos}(s') \text{ where } w = \mathrm{pos}_x(t) \\ x_1 & \text{otherwise.} \end{cases}$$

Then $u' = u[p(x) \leftarrow \langle p, \mathrm{restr}_{s',t}(x)\rangle(x) \mid p \in Q, x \in \mathrm{var}(t)]$.

Note that $W = \emptyset$ in item (i) provided that M is nondeleting. Thus, $t' = t$ in this case. Moreover, we observe that every $x \in X$ occurs as often in u' as it occurs in u. Consequently, M' is linear (respectively, nondeleting and input-ε-free) whenever M is so. Finally, let us sketch a proof of $\tau_{M'} = \tau_M$. It can be proved that

$$q(t) \Rightarrow_M^* u \quad \iff \quad \langle q, s\rangle(t) \Rightarrow_{M'}^* u$$

for every $q \in Q$, input tree $t \in T_\Sigma$, output tree $u \in T_\Delta$, and look-ahead obligation $s \in S$ such that s matches t. Both directions of this statement can be shown in a straightforward fashion by induction on the length of the derivations. Then $\{(t, u) \mid \exists q \in I\colon \langle q, x_1\rangle(t) \Rightarrow_{M'}^* u\}$ clearly equals $\{(t, u) \mid \exists q \in I\colon q(t) \Rightarrow_M^* u\}$ and hence $\tau_{M'} = \tau_M$. □

We showed that xtt have finite look-ahead. Consequently, the next question is whether they also have regular look-ahead. In Lemma 4.3 we prove that xtt do not necessarily have regular look-ahead. In essence, this allows us to conclude that finite look-ahead is indeed the maximal look-ahead power that xtt possess in general. We prepare the result by a simple normalization property. Let $M = (Q, \Sigma, \Delta, I, R, c)$ be an xtt$^{\mathrm{R}}$. The rule $r \in R$ is a *chain rule* if $r = (q(x) \to p(x))$ for some $p, q \in Q$ and $x \in X$.

Lemma 4.2 (Elimination of chain rules). *For every xtt^R M there exists an equivalent xtt^R without chain rules, which is linear (respectively, nondeleting) whenever M is so.*

Proof. The proof is straightforward and omitted. It uses the fact that recognizable tree languages are closed under intersection [16, Theorem II.4.2]. □

Now we prove that xtt do not necessarily have regular look-ahead. Obviously, for every recognizable language $L \subseteq T_\Sigma$ over the input ranked alphabet and $u \in T_\Delta$ over the output ranked alphabet, there is a linear tdtt$^{\mathrm{R}}$ M such that $\tau_M = \{(t, u) \mid t \in L\}$. However, we prove that for certain L and u there exists no xtt equivalent to M. Clearly, this deficiency is due to the lack of regular look-ahead.

Lemma 4.3 (Extended tree transducers do not have regular look-ahead).

$$\text{l-TOP}^{\mathrm{R}} \not\subseteq \text{XTOP}$$

Proof. Let $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$ and $\Delta = \{\beta^{(0)}\}$. Moreover, let

$$L = \{\sigma(t_1, t_2) \mid t_1, t_2 \in T_\Sigma, \exists (w_1, w_2) \in \mathrm{pos}_\alpha(t_1) \times \mathrm{pos}_\alpha(t_2)\colon |w_1| \text{ and } |w_2| \text{ are even}\} \ .$$

Intuitively, L contains all trees that have the symbol σ at the root and both direct subtrees have an occurrence of α at an odd level (where we suppose that the root is on level 1). For example, $t = \sigma(\alpha, \alpha)$ is in L but $\sigma(t, t)$ is not. Let $t_0 = \alpha$ and $t_{n+1} = \sigma(t_n, t_n)$ for every $n \in \mathbb{N}$. Then $t_n \in L$ iff n is odd. It is an easy exercise to prove that L is recognizable.

Let $\tau = \{(t, \beta) \mid t \in L\}$ and $(\{\star\}, \Sigma, \Delta, \{\star\}, \{r\}, c)$ be the linear tdtt$^{\mathrm{R}}$ with $r = (\star(\sigma(x_1, x_2)) \to \beta)$ and $c(r) = L$. Obviously, it computes τ, and consequently, $\tau \in \text{l-TOP}^{\mathrm{R}}$. It remains to prove that $\tau \notin \text{XTOP}$.

Suppose that there exists an xtt that computes τ. By Lemma 4.2, there also exists an xtt $M = (Q, \Sigma, \Delta, I, R)$ without chain rules that computes τ. Let

$$n = \max\{\mathrm{ht}(t) \mid (q(t) \to u) \in R\}$$

be the maximal height of a tree on the left-hand side of a rule. Since $(t_{2n+1}, \beta) \in \tau$ there must exist an initial state $q \in I$ such that $q(t_{2n+1}) \Rightarrow_M^* \beta$. Consider a derivation

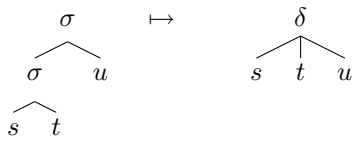

FIG. 4.3. *Example flattening where s, t, and u are arbitrary trees.*

$q(t_{2n+1}) \Rightarrow^r_M \xi \Rightarrow^*_M \beta$ with $r \in R$ and $\xi \in T_\Delta(Q(T_\Sigma))$. Let $r = (q(t) \to u)$ for some $t \in T_\Sigma(X)$ and $u \in T_\Delta(Q(X))$. It is immediately clear that u must be either (i) β or (ii) $p(x)$ for some $p \in Q$ and $x \in \mathrm{var}(t)$. Next we distinguish those two cases.

(i) Since $q(t_{2n+1}) \Rightarrow^r_M \beta$, we know that t matches t_{2n+1}. Clearly, $\mathrm{ht}(t) \leq n$, and consequently, t also matches t_{2n+2}. Then $q(t_{2n+2}) \Rightarrow^r_M \beta$ and $(t_{2n+2}, \beta) \in \tau_M$, which yields that M does not compute τ.

(ii) Clearly, $\mathrm{pos}_x(t) \neq \varepsilon$ since M has no chain rules. Let $\mathrm{pos}_x(t) = iw$ for some $i \in \{1, 2\}$ and $w \in \mathrm{pos}(\mathrm{sub}_t(i))$. By $t_{2n+1} = \sigma(t_{2n}, t_{2n})$ we have $\xi = p(\mathrm{sub}_{t_{2n}}(w))$. With $t' = \sigma(t_{2n+(i-1)}, t_{2n+(2-i)})$ we obtain $q(t') \Rightarrow^r_M \xi$ because t matches t' and $\mathrm{sub}_{t'}(iw) = \mathrm{sub}_{t_{2n}}(w)$. Consequently, $q(t') \Rightarrow^r_M \xi \Rightarrow^*_M \beta$ and $(t', \beta) \in \tau_M$, which yields that M does not compute τ.

We showed that both cases are contradictory, which proves that $\tau \notin \mathrm{XTOP}$. □

However, all nondeleting devices have regular look-ahead. This is well known for tdtt and can similarly be shown for input-ε-free xtt and xtt. For the result on tdtt we reconsider [10, Theorem 2.8] in the nondeleting case and then [9, Theorem 2.9]. The proof for xtt is left as an exercise.

THEOREM 4.4 (Nondeleting transducers have regular look-ahead).

$$\text{ln-TOP} = \text{ln-TOP}^{\mathrm{R}} \quad \text{and} \quad \text{ln [e]-XTOP} = \text{ln [e]-XTOP}^{\mathrm{R}}$$

To complete the picture concerning look-ahead, let us show that tdtt become stronger when adding finite look-ahead. It is known that tdtt^{R} are more powerful than tdtt [10, Corollary 2.3]. The power of tdtt^{F} is clearly between tdtt and tdtt^{R}. The next lemma shows that TOP is strictly included in TOP^{F}.

LEMMA 4.5 (Top-down tree transducers do not have finite look-ahead).

$$\text{l-TOP}^{\mathrm{F}} \not\subseteq \text{TOP} \quad \text{and} \quad \text{lne-XTOP} \not\subseteq \text{TOP}$$

Proof. It is obvious that $\tau = \{(\sigma(\alpha, \alpha), \alpha)\}$ is not in TOP. However, a linear tdtt^{F} with one state $\star$ can easily compute τ using the rule $r = (\star(\sigma(x_1, x_2)) \to \alpha)$ with $c(r) = \{\sigma(\alpha, \alpha)\}$. Moreover, a linear, nondeleting, input-ε-free xtt with one state $\star$ and the rule $\star(\sigma(\alpha, \alpha)) \to \alpha$ also computes τ. In fact, $\text{l-TOP}^{\mathrm{F}}$ and lne-XTOP contain every finite transformation. □

With these results we already characterized the effect of the look-ahead component on the power of transducers. But so far, we have not compared xtt with tdtt apart from the finite look-ahead component. In fact, xtt possess another property, here called *deep attachment of variables*, that separates it from tdtt. This feature can be used to implement local rotations of the kind demonstrated in Figs. 1.1 and 3.1. In particular, it allows us to implement flattenings (see Fig. 4.3 and rule r_3 in Section 3) and shuffles (as in Fig. 1.1). From the definition of flattenings and shuffles, the following lemma is immediate.

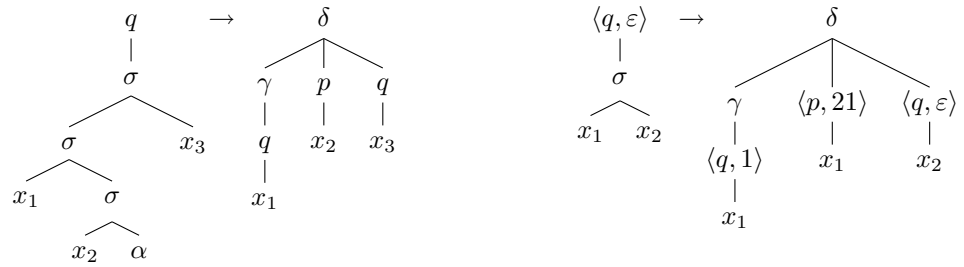

FIG. 4.4. *Example rule and resulting rule (see Theorem 4.8).*

LEMMA 4.6 (All xtt can flatten and shuffle).

$$\text{FLAT} \subseteq \text{lne-XTOP} \quad \text{and} \quad \text{SHUF} \subseteq \text{lne-XTOP}$$

Thus, we identified another feature of xtt. However, we will see in Theorem 4.8 that nonlinear tdtt^{R} can simulate *deep attachment of variables* with the help of copying and deleting. Thus, we now only show that flattening and shuffle cannot be implemented by linear tdtt^{R}.

LEMMA 4.7 (Linear tdtt^{R} can neither flatten nor shuffle).

$$\text{FLAT} \not\subseteq \text{l-TOP}^{\text{R}} \quad \text{and} \quad \text{SHUF} \not\subseteq \text{l-TOP}^{\text{R}}$$

Proof. Both statements are immediate by Theorem 5.2 because l-TOP^{R} is closed under composition by [10, Theorem 2.11]. ∎

We already remarked that the property *deep attachment of variables* can be simulated by nonlinear and deleting tdtt^{R}. In fact, we show how to simulate an input-ε-free xtt^{R} by a nonlinear tdtt^{R}. In case of an xtt we can simulate it with a tdtt^{F}. The finite look-ahead is needed because xtt in general have finite look-ahead (see Theorem 4.1 and Lemma 4.5).

Let us illustrate the idea of the construction on a small example. Suppose the input is an xtt and has the rule $q(\sigma(\sigma(x_1, \sigma(x_2, \alpha)), x_3)) \to \delta(\gamma(q(x_1)), p(x_2), q(x_3))$. In the constructed tdtt^{F}, we check the principal shape of the left-hand side by the finite look-ahead restriction. Moreover, we use auxiliary states that walk down the input tree to the point of the deep attachment. Thus, we construct the rule

$$\langle q, \varepsilon\rangle(\sigma(x_1, x_2)) \to \delta(\gamma(\langle q, 1\rangle(x_1)), \langle p, 21\rangle(x_1), \langle q, \varepsilon\rangle(x_2))$$

with look-ahead restriction $\sigma(\sigma(x_1, \sigma(x_1, \alpha)), x_1)$. Both mentioned rules are displayed in Fig. 4.4. The state p was originally called on x_2, and now we call $\langle p, w\rangle(x_i)$ where $w = 21$ and $i = 1$. Observe that $iw = \text{pos}_{x_2}(t)$ where t is the input tree in the left-hand side of the original rule. Moreover, we generate the support rules

$$\begin{aligned}
\langle q, 1\rangle(\sigma(x_1, x_2)) &\to \langle q, \varepsilon\rangle(x_1) \\
\langle p, 21\rangle(\sigma(x_1, x_2)) &\to \langle p, 1\rangle(x_2) \\
\langle p, 1\rangle(\sigma(x_1, x_2)) &\to \langle p, \varepsilon\rangle(x_1) \ .
\end{aligned}$$

These support rules allow us to walk down the input tree to the desired positions. We arrive at the required input subtree in a state $\langle q, \varepsilon\rangle$ for some $q \in Q$. As previously

discussed, we construct one rule using such a state for every rule in the original xtt. In essence, the application of a rule of the xtt is simulated by a corresponding rule of the tdtt followed by several applications of auxiliary rules.

THEOREM 4.8 (Copying and deletion yield *deep attachment of variables*).

$$\text{e-XTOP} = \text{TOP}^{\text{F}} \quad \text{and} \quad \text{e-XTOP}^{\text{R}} = \text{TOP}^{\text{R}}$$

Proof. One inclusion is trivial in both statements with the help of Theorem 4.1. Let $M = (Q, \Sigma, \Delta, I, R, c)$ be an input-ε-free xtt$^{\text{R}}$. Let $T = \{t \mid (q(t) \to u) \in R\}$, and let W be the smallest postfix-closed (i.e., if $vw \in W$, then also $w \in W$) set that contains $\bigcup_{t \in T} \text{pos}_X(t)$. We construct the tdtt$^{\text{R}}$ $M' = (Q \times W, \Sigma, \Delta, I \times \{\varepsilon\}, R', c')$ as follows:

(i) R' contains for every rule $r = (q(t) \to u) \in R$ with $t = \sigma(t_1, \ldots, t_k)$ for some $\sigma \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma(X)$ the following rule r'

$$\langle q, \varepsilon\rangle(\sigma(x_1, \ldots, x_k)) \to u[p(x) \leftarrow \langle p, w\rangle(x_i) \mid p \in Q, x \in \text{var}(t), i \in [k], \text{pos}_x(t) = iw]$$

and $c'(r') = c(r) \cap (t[x \leftarrow x_1 \mid x \in \text{var}(t)])[T_\Sigma]$.

(ii) R' contains for every $q \in Q$, $\sigma \in \Sigma_k$, and $iw \in W$ with $i \in [k]$ the following rule r'' with $c'(r'') = T_\Sigma$.

$$\langle q, iw\rangle(\sigma(x_1, \ldots, x_k)) \to \langle q, w\rangle(x_i)$$

The intersection in the look-ahead restriction in item (i) creates a recognizable tree language provided that $c(r)$ is recognizable [16, Theorem II.4.2]. If $c(r) = T_\Sigma$ for every $r \in R$ (i.e., M has no look-ahead), then M' has finite look-ahead. Thus, M' has regular (respectively, finite) look-ahead whenever M has regular (respectively, no) look-ahead. Moreover, we immediately observe that the rules in the second item allow $\langle q, w\rangle(t) \Rightarrow^*_{M'} \langle q, \varepsilon\rangle(\text{sub}_t(w))$ for every $q \in Q$, $t \in T_\Sigma$, and $w \in \text{pos}(t)$. In fact, there exists exactly one such derivation. Thus, the states $\langle q, w\rangle$ can be used to walk down the input tree t to position w.

It remains to show that $\tau_{M'} = \tau_M$. This is mostly straightforward; here we only show how M' simulates the application of a rule r given in the first item with the help of the rule r'. The main proof obligation is $q(s) \Rightarrow^*_M v$ exactly when $\langle q, \varepsilon\rangle(s) \Rightarrow^*_{M'} v$ for every $q \in Q$, $s \in T_\Sigma$, and $v \in T_\Delta$. We only prove one direction, so let $q(s) \Rightarrow^r_M \xi$ for some $q \in Q$, $s \in T_\Sigma$, $r = (q(t) \to u) \in R$, and $\xi \in T_\Delta(Q(T_\Sigma))$. There exists a substitution $\theta\colon X \to T_\Sigma$ such that $s = t\theta$ and $\xi = u[p(x) \leftarrow p(x\theta) \mid p \in Q, x \in \text{var}(t)]$. Moreover, $t\theta \in c(r)$. We first observe that $t\theta \in c'(r')$ and thus the rule r' is applicable to $\langle q, \varepsilon\rangle(s)$. We obtain $\langle q, \varepsilon\rangle(s) \Rightarrow^{r'}_{M'} u'$ with $u' = u[p(x) \leftarrow \langle p, w\rangle(t_i\theta) \mid p \in Q, x \in \text{var}(t), i \in [k], \text{pos}_x(t) = iw]$. Clearly, we have $\langle p, w\rangle(t_i\theta) \Rightarrow^*_{M'} \langle p, \varepsilon\rangle(\text{sub}_{t_i\theta}(w))$. The latter equals $\langle p, \varepsilon\rangle(\text{sub}_{t\theta}(iw))$. Since $\text{lab}_t(iw) = x$, we obtain $\text{sub}_{t\theta}(iw) = x\theta$. Consequently, $\langle q, \varepsilon\rangle(s) \Rightarrow^{r'}_{M'} u' \Rightarrow^*_{M'} u''$ where

$$u'' = u[p(x) \leftarrow \langle p, \varepsilon\rangle(x\theta) \mid p \in Q, x \in \text{var}(t)] = \xi[p(t) \leftarrow \langle p, \varepsilon\rangle(t) \mid p \in Q, t \in T_\Sigma] \ .$$

The application of the induction hypothesis then proves this direction. The argument for the converse direction uses that the derivation $\langle q, w\rangle(t) \Rightarrow^*_{M'} \langle q, \varepsilon\rangle(\text{sub}_t(w))$ is unique. ∎

The final feature of xtt that gives them additional expressive power when compared to tdtt is the ability to generate arbitrarily large outputs for some input tree

(which was the reason to restrict attention to input-ε-free xtt in Theorem 4.8). This can be achieved with the help of input-epsilon rules, e.g., $q(x) \to \gamma(q(x))$. In this case, the tree in the left-hand side simply shrinks to just a variable. Thus, we see that a large non-shallow tree in the left-hand side may yield the features of *finite look-ahead* and *deep attachment of variables*, whereas a left-hand side with only a variable may lead to infinitely many output trees for some particular input tree (even for linear and nondeleting xtt). However, for every input-ε-free xtt^{R} the set of translations (i.e., output trees) is finite for every input tree. By this argumentation, input-ε-free transducers are strictly less powerful than their unrestricted variants.

LEMMA 4.9 (Input-ε-free transducers generate bounded outputs).

$$\text{ln-XTOP} \not\subseteq \text{e-XTOP}^{\text{R}}$$

Let us recall the properties of xtt that lead to the separation of xtt and tdtt.

(X1) Finite look-ahead
(X2) Deep attachment of variables
(X3) Infinitely many outputs for one input

Finally, we need to separate linearity and copying as well as nondeletion and deletion (in the linear case). This is achieved in the same manner as for tdtt.

LEMMA 4.10.

$$\text{TOP} \not\subseteq \text{l-XTOP}^{\text{R}} \quad \text{and} \quad \text{l-TOP} \not\subseteq \text{ln-XTOP}$$

Proof. The first statement is easily proved since it is known that transformations of TOP need not preserve recognizability [16, Example II.4.15], whereas the ones of l-XTOP^{R} do [22, Theorem 4]. Moreover, we note that the transformations used to prove Theorem 5.2 are both in TOP but not in l-XTOP^{R} (as shown in the proofs).

For the second statement, consider the transformation $\tau = \{(t, \alpha) \mid t \in T_\Sigma\}$ where $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$. It is straightforward to prove that $\tau \in \text{l-TOP}$ and $\tau \notin \text{ln-XTOP}$. □

Now, we have all necessary results to completely characterize the subset relationships between all introduced classes of transformations computed by xtt. We collect our knowledge about the power of xtt^{R} in a HASSE diagram (see Fig. 4.5) with the additional convention that all edges are oriented upwards or to the right. As usual, an edge denotes strict inclusion. Unrelated classes in a HASSE diagram are incomparable with respect to set inclusion. We do not repeat the look-ahead results of Theorems 4.1 and 4.4, but do present the equalities of Theorem 4.8.

THEOREM 4.11. *The diagram in Fig. 4.5 is a* HASSE *diagram.*

Proof. The inclusions are trivial or by Theorems 4.1 and 4.8. The non-inclusions are proved in Lemmata 4.3, 4.5, 4.6, 4.7, 4.9, and 4.10 (note that Lemmata 4.6 and 4.7 together show that $\text{lne-XTOP} \not\subseteq \text{l-TOP}^{\text{R}}$). □

5. Closure under composition. In this section, we investigate which of the introduced classes of transformations are closed under composition. Composition is useful because it allows designers to break down a complex transformation into a cascade of simple transformations. These simple transformations can be represented using simple transducers, which can then be assembled automatically into a single large transducer that represents the complex transformation. Subsequent operations (e.g., application to an input tree or an input tree language, pruning, and probabilistic training) may then be applied to the complex transducer.

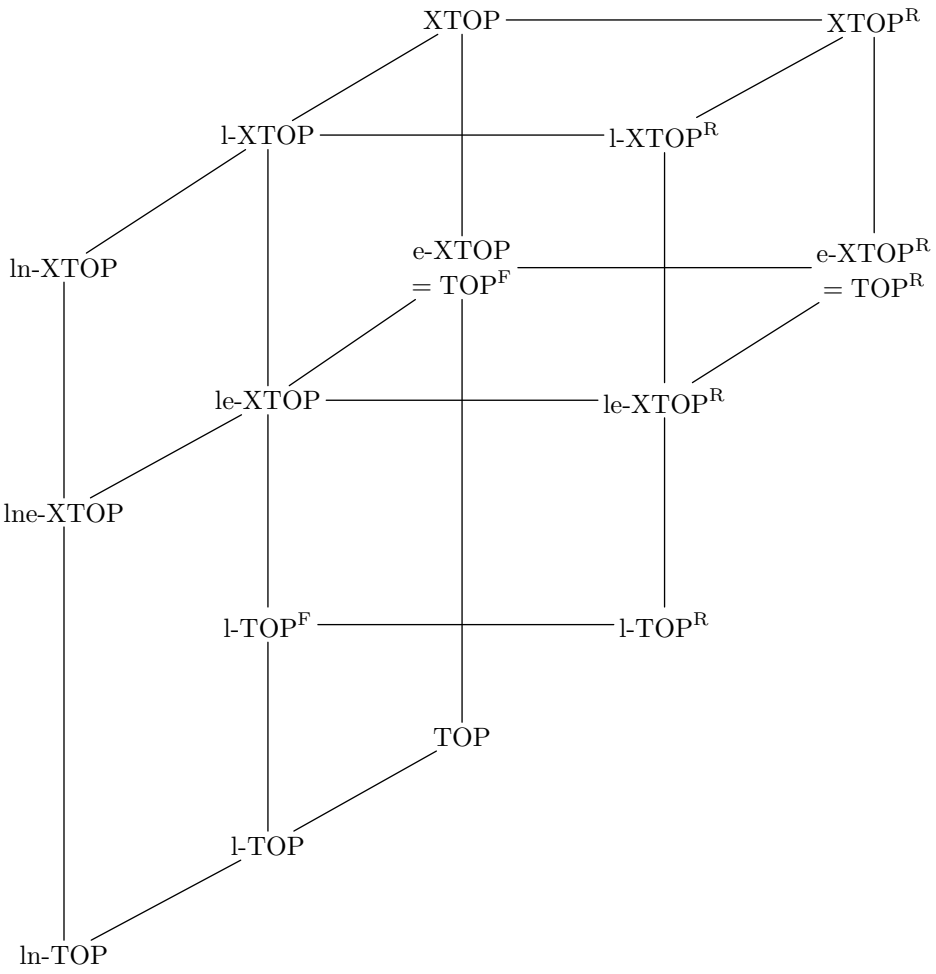

FIG. 4.5. HASSE *diagram of the classes of tree transformations computed by xtt^{R}. All edges are directed to the right or upward.*

Let us recall that ln-TOP and l-TOP^{R} are closed under composition. Both composition constructions can be found in [3], though the results trace back to [9]. Instead of linear tdtt^{R}, BAKER and ENGELFRIET investigated linear bottom-up tree transducers [28]. However, ENGELFRIET [10, Theorem 2.8] showed that linear bottom-up tree transducers are as powerful as linear tdtt^{R}.

Unfortunately, all of the following results are negative. First, any class that contains l-TOP should have the feature of regular look-ahead to be closed under composition. This is due to the fact that l-TOP^{R} is the composition closure of l-TOP by [10, Theorem 2.6] and [9, Lemma 3.2].

LEMMA 5.1 (Regular look-ahead is required for closure under composition). *If $\mathcal{L}$ is a class of tree transformations such that* l-TOP $\subseteq \mathcal{L} \subseteq$ XTOP*, then $\mathcal{L}$ is not closed under composition.*

Proof. It is known that l-TOP^{R} is the composition closure of l-TOP (see above). Hence, since l-TOP $\subseteq \mathcal{L}$ but l-TOP$^{R} \not\subseteq$ XTOP by Lemma 4.3, the class $\mathcal{L}$ cannot be closed under composition. ∎

FIG. 5.1. *Illustration of the tree transformations used in the proof of Theorem 5.2.*

We have just seen that the composition of transducers with finite look-ahead (or even no look-ahead) can only be implemented by a transducer with regular look-ahead. With respect to feature (X2), a similar problem arises. Intuitively speaking, we show that compositions of transducers with *deep* (but finite) *attachment of variables* require a more complicated form of attachment of variables, if they should be implemented on a single transducer. We will discuss one extension, called *regular attachment of variables*, briefly after the next lemma, but here we first show that linear $\mathrm{xtt}^{\mathrm{R}}$ do not have it.

In fact, it is proved in [1] and [2, Section 3.4] that no class of transformations computed by linear $\mathrm{xtt}^{\mathrm{R}}$ (see [22, Theorem 4] for the relation between linear $\mathrm{xtt}^{\mathrm{R}}$ and bimorphisms) containing ln-TOP and FLAT is closed under composition. Thus, by Lemma 4.6, no class between lne-XTOP and l-$\mathrm{XTOP}^{\mathrm{R}}$ is closed under composition (cf. [22, Corollaries 5 and 18]). We provide an alternative proof of this result and show that it also holds for shuffles instead of flattenings. It should, however, be noted that in [1, 2] the result is even proved for a subclass of ln-TOP and a superclass of l-$\mathrm{XTOP}^{\mathrm{R}}$.

THEOREM 5.2 (Linear $\mathrm{xtt}^{\mathrm{R}}$ do not have *regular attachment of variables*). *If $\mathcal{L}$ is a class of tree transformations such that* ln-TOP $\subseteq \mathcal{L} \subseteq$ l-$\mathrm{XTOP}^{\mathrm{R}}$ *and (i)* FLAT $\subseteq \mathcal{L}$ *or (ii)* SHUF $\subseteq \mathcal{L}$*, then $\mathcal{L}$ is not closed under composition.*

Proof. It suffices to show the following statements.

$$\text{ln-TOP} \,;\, \text{FLAT} \not\subseteq \text{l-XTOP}^{\mathrm{R}} \tag{5.1}$$

$$\text{ln-TOP} \,;\, \text{SHUF} \not\subseteq \text{l-XTOP}^{\mathrm{R}} \tag{5.2}$$

The former is shown in [2, Section 3.4]. Here we present a proof for both statements. To this end, let $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$ and $\Delta = \{\delta^{(3)}, \sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$. We consider the tree transformations

$$\tau_1 = \{(\sigma(\gamma^n(\sigma(s,t)),u), \delta(s,t,u)) \mid n \in \mathbb{N}, s,t,u \in T_\Sigma\}$$
$$\tau_2 = \{(\sigma(\gamma^n(\sigma(s,t)),u), \sigma(s,\sigma(t,u))) \mid n \in \mathbb{N}, s,t,u \in T_\Sigma\}$$

for (5.1) and (5.2), respectively. The transformations are depicted in Fig. 5.1. Let us first show that τ_1 can be computed by a composition of a linear and nondeleting tdtt and a flattening and τ_2 can be computed by a composition of a linear and nondeleting tdtt and a shuffle. To this end, let $M' = (\{\star, d, \mathrm{id}\}, \Sigma, \Sigma, \{\star\}, R')$ be the linear and nondeleting tdtt with the rules

$$\begin{aligned}
\star(\sigma(x_1,x_2)) &\to \sigma(d(x_1), \mathrm{id}(x_2)) & \mathrm{id}(\sigma(x_1,x_2)) &\to \sigma(\mathrm{id}(x_1), \mathrm{id}(x_2)) \\
d(\gamma(x_1)) &\to d(x_1) & \mathrm{id}(\gamma(x_1)) &\to \gamma(\mathrm{id}(x_1)) \\
d(\sigma(x_1,x_2)) &\to \sigma(\mathrm{id}(x_1), \mathrm{id}(x_2)) & \mathrm{id}(\alpha) &\to \alpha \ .
\end{aligned}$$

A routine proof shows that

$$\tau_{M'} = \{(\sigma(\gamma^n(\sigma(s,t)),u), \sigma(\sigma(s,t),u)) \mid n \in \mathbb{N}, s,t,u \in T_\Sigma\} \ .$$

The tree transformations

$$\tau_1' = \{(\sigma(\sigma(s,t),u), \delta(s,t,u)) \mid s,t,u \in T_\Sigma\}$$
$$\tau_2' = \{(\sigma(\sigma(s,t),u), \sigma(s,\sigma(t,u))) \mid s,t,u \in T_\Sigma\}$$

are a flattening and a shuffle, respectively, which is easily shown. Clearly, $\tau_{M'}; \tau_1' = \tau_1$ and $\tau_{M'}; \tau_2' = \tau_2$.

It remains to show that $\tau_1, \tau_2 \notin$ l-XTOP$^{\mathrm{R}}$. Suppose that $\tau_1 \in$ l-XTOP$^{\mathrm{R}}$. Then there exists a linear xtt$^{\mathrm{R}}$ $M = (Q, \Sigma, \Delta, I, R, c)$ such that $\tau_M = \tau_1$. Then also $\tau_2 \in$ l-XTOP$^{\mathrm{R}}$ since we can simply replace (keeping the look-ahead) all rules of the form $q(t) \to \delta(u_1,u_2,u_3)$ in R by $q(t) \to \sigma(u_1, \sigma(u_2,u_3))$ to obtain a linear xtt$^{\mathrm{R}}$ that computes τ_2.

Consequently, it suffices to prove that $\tau_2 \notin$ l-XTOP$^{\mathrm{R}}$. Suppose that there exists a linear xtt$^{\mathrm{R}}$ $M = (Q, \Sigma, \Delta, I, R, c)$ such that $\tau_M = \tau_2$. Without loss of generality, suppose that M has no chain rules (see Lemma 4.2) and let N be a dta such that for every $r \in R$ there exists a subset P of states with $c(r) = \bigcup_{p \in P} L(N)_p$. Moreover, let

$$n \geq \max\{\max(\mathrm{ht}(t), \mathrm{ht}(u)) \mid (q(t) \to u) \in R\}$$

be larger than the maximal height of input and output trees of R. Additionally, let n be larger than the number of states of N. For every $i \in \mathbb{N}$, we denote $\gamma^i(\alpha)$ simply by t_i. Let p be the state of N that recognizes t_n (i.e., $t_n \in L(N)_p$). Since N has at most n states and $\mathrm{ht}(t_n) = n+1$, there exists a tree $t_n' \in L(N)_p$ such that $t_n' \neq t_n$ by the pumping lemma for dta (see [16]).

Let $s = \sigma(\gamma^n(\sigma(\sigma(t_n,t_n),t_n)),t_{2n})$ [see Fig. 5.2 for illustration]. Clearly, there exists an initial state $q \in I$ such that $q(s) \Rightarrow_M^* u$ with $u = \sigma(\sigma(t_n,t_n), \sigma(t_n,t_{2n}))$. Let us consider a derivation $q(s) \Rightarrow_M^r \xi \Rightarrow_M^* u$ where $r \in R$ and $\xi \in T_\Delta(Q(T_\Sigma))$. We now distinguish three cases for r.

(i) Let $r = (q(t) \to q'(x))$ for some $t \in T_\Sigma(X)$, $q' \in Q$, and $x \in \mathrm{var}(t)$. Then $\xi = q'(\mathrm{sub}_s(\mathrm{pos}_x(t)))$. Clearly, $s \in c(r)$ and $\mathrm{pos}_x(t) \neq \varepsilon$ because M has no chain rules. Suppose that $\mathrm{pos}_x(t) = 1w$ for some $w \in \mathrm{pos}(\mathrm{sub}_t(1))$. Let

$$s' = \sigma(\gamma^n(\sigma(\sigma(t_n,t_n),t_n)), \gamma^n(t_n')) \ .$$

Clearly, $s' \in c(r)$ because $\mathrm{sub}_{s'}(1) = \mathrm{sub}_s(1)$ and $\mathrm{sub}_{s'}(2) = \gamma^n(t_n')$, which is recognized in the same state as t_{2n}. Obviously, $\mathrm{sub}_{s'}(\mathrm{pos}_x(t)) = \mathrm{sub}_s(\mathrm{pos}_x(t))$ and $\mathrm{lab}_{s'}(w') = \mathrm{lab}_s(w')$ for every $w' \in \mathrm{pos}(s)$ with $|w'| < \mathrm{ht}(t)$. Thus, t matches s' and $q(s') \Rightarrow_M^r \xi \Rightarrow_M^* u$. This yields $(s',u) \in \tau_M$, which is a contradiction. The case $\mathrm{pos}_x(t) = 2w$ can be handled in the same manner.

(ii) Now suppose that $r = (q(t) \to \sigma(u_1,u_2))$ for some $u_1, u_2 \in T_\Delta(Q(X))$. Since $\mathrm{ht}(u_i) < \mathrm{ht}(\mathrm{sub}_u(i))$ for every $i \in \{1,2\}$, the trees u_1 and u_2 each contain a variable. Since M is linear, those variables are distinct. Thus, $t = \sigma(\gamma^i(x_1), \gamma^j(x_2))$ for some $i,j \in \mathbb{N}$ smaller than n (note that, without loss of generality, the variables x_1 and x_2 can be used). Moreover, this yields that $u_1 = q_1(x)$ and $u_2 = q_2(y)$ for some $q_1, q_2 \in Q$ and $\{x,y\} = \{x_1,x_2\}$. We now distinguish two simple subcases. First, let us assume that $x_2 = y$. Then $q_2(t_{2n-j}) \Rightarrow_M^* \sigma(t_n,t_{2n})$. Let $r' = (q'(t') \to \sigma(u_1',u_2')) \in R$ be the rule that generates the σ in the output. Clearly, $\mathrm{card}(\mathrm{var}(t')) \leq 1$, which yields that

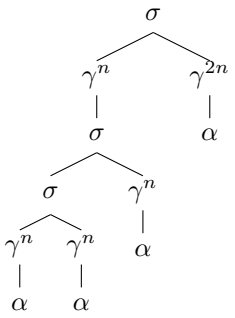

FIG. 5.2. *Input tree s used in the proof of Theorem 5.2.*

u'_1 or u'_2 does not contain a variable. However, $\mathrm{ht}(u'_1) < n$ and $\mathrm{ht}(u'_2) < n$, which is a contradiction. Finally, assume that $x_2 = x$. Then $q_1(t_{2n-j}) \Rightarrow^*_M \sigma(t_n, t_n)$. By a similar line of reasoning, we can also derive a contradiction in this case.
Both cases are contradictory, which allows us to conclude that $\tau_2 \notin \text{l-XTOP}^{\mathrm{R}}$. ∎

The transformations used to prove Theorem 5.2 cannot be computed using the deep but finite attachment of variables that linear xtt^{R} have. Intuitively speaking, the chain of γ-symbols is in the way and can be made suitably long. One can immediately conceive a model of extended top-down tree transducers with *regular attachment of variables* (i.e., the attachment of the variables is no longer restricted to finite depth, but rather it is given by a regular tree language). Formally, such a transducer is given by $(Q, \Sigma, \Delta, I, R, c)$ where Q, Σ, Δ, I, and c are as for xtt^{R} and R is a finite set of rules of the form $\langle q, L\rangle \to u$ where $q \in Q$, $L \subseteq T_\Sigma(X)$ is a nonempty recognizable tree language such that there exists $n \in \mathbb{N}$ such that t is X_n-linear and X_n-nondeleting for every $t \in L$, and $u \in T_\Delta(Q(X_n))$. The semantics is given as if it were the "extended top-down tree transducer" $(Q, \Sigma, \Delta, I, R', c')$ where for every rule $r = \langle q, L\rangle \to u$ of R and $t \in L$, the rule $r' = q(t) \to u$ is in R' and its look-ahead is $c'(r') = c(r)$. Note that this might yield an "extended top-down tree transducer" with infinitely many rules, but the definition of the semantics is meaningful also for such transducers. The properties (linear, nondeleting, etc.) of xtt^{R} can be defined in a straightforward fashion also for such transducers.

Let us show such a linear transducer with regular attachment of variables that computes the translation used in the proof of Theorem 5.2. Let Σ, Δ, and τ_2 be as they are in the proof of Theorem 5.2. Let $M = (\{\star, \mathrm{id}\}, \Sigma, \Delta, \{\star\}, \{r\} \cup R')$ be the tree transducer with $r = \langle \star, L\rangle \to \sigma(\mathrm{id}(x_1), \sigma(\mathrm{id}(x_2), \mathrm{id}(x_3)))$ where

$$L = \{\sigma(\gamma^n(\sigma(x_1, x_2)), x_3) \mid n \in \mathbb{N}\}$$

and R' containing the id-rules of M' in the proof of Theorem 5.2. Then M obviously computes τ_2. In the following, we do not consider such transducers. Finally, we note that there are interesting (i.e., non-universal) classes of tree transformations that contain l-XTOP and are closed under composition. An example of such a class is the class l-XMBOT of transformations computed by linear extended multi bottom-up tree transducers [12].

At this point, we already proved that most classes of Fig. 4.5 are not closed under composition. In addition, we know [10, Corollary 2.4] that TOP^{R} is not closed under composition. So, it only remains to show that XTOP^{R} is also not closed under composition. In essence, this is achieved in the same manner as for TOP^{R}. In fact,

the next lemma also proves that no class $\mathcal{L}$ such that $\mathrm{TOP} \subseteq \mathcal{L} \subseteq \mathrm{XTOP}^{\mathrm{R}}$ is closed under composition.

We need a new notion for the proof of the following lemma. Let Σ and Δ be ranked alphabets, and let $\rho\colon \Sigma \to \mathcal{P}(\Delta)$ be such that $\rho(\sigma) \subseteq \Delta_k$ for every $\sigma \in \Sigma_k$, i.e., ρ preserves the rank of symbols. The mapping ρ is then lifted to a *relabeling* $\rho\colon T_\Sigma \to \mathcal{P}(T_\Delta)$ by

$$\rho(\sigma(t_1, \ldots, t_k)) = \{\delta(u_1, \ldots, u_k) \mid \delta \in \rho(\sigma), \forall i \in [k]\colon u_i \in \rho(t_i)\}$$

for every $\sigma \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma$. By REL we denote the class of all relabelings. Note that a relabeling can be implemented by a linear and nondeleting tdtt with just a single state, i.e., $\mathrm{REL} \subseteq \text{ln-TOP}$.

Lemma 5.3 ($\mathrm{XTOP}^{\mathrm{R}}$ is not closed under composition). *If $\mathcal{L}$ is a class of tree transformations such that $\mathrm{TOP} \subseteq \mathcal{L} \subseteq \mathrm{XTOP}^{\mathrm{R}}$, then $\mathcal{L}$ is not closed under composition.*

Proof. Clearly, it is sufficient to prove that $\mathrm{REL}\,;\mathrm{TOP} \not\subseteq \mathrm{XTOP}^{\mathrm{R}}$. To this end, let $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$ and $\Delta = \{\delta^{(4)}, \sigma^{(2)}, \gamma_1^{(1)}, \gamma_2^{(1)}, \alpha^{(0)}\}$. Consider the mapping $\rho\colon \Sigma \to \mathcal{P}(\Delta)$ given by $\rho(\sigma) = \{\sigma\}$, $\rho(\gamma) = \{\gamma_1, \gamma_2\}$ and $\rho(\alpha) = \{\alpha\}$. Moreover, let $M' = (\{\star, \mathrm{id}\}, \Delta, \Delta, \{\star\}, R')$ be the tdtt with the following rules:

$$\begin{aligned}
\star(\sigma(x_1, x_2)) &\to \delta(\mathrm{id}(x_1), \mathrm{id}(x_1), \mathrm{id}(x_2), \mathrm{id}(x_2)) \\
\mathrm{id}(\gamma_1(x_1)) &\to \gamma_1(\mathrm{id}(x_1)) \\
\mathrm{id}(\gamma_2(x_1)) &\to \gamma_2(\mathrm{id}(x_1)) \\
\mathrm{id}(\alpha) &\to \alpha \ .
\end{aligned}$$

Clearly, M' computes $\{(\sigma(u_1, u_2), \delta(u_1, u_1, u_2, u_2)) \mid u_1, u_2 \in T_\Gamma\}$ where the ranked alphabet Γ is $\{\gamma_1^{(1)}, \gamma_2^{(1)}, \alpha^{(0)}\}$. Combined with the relabeling ρ we obtain the tree transformation

$$\tau = \{(\sigma(t_1, t_2), \delta(u_1, u_1, u_2, u_2)) \mid t_1, t_2 \in T_{\Sigma'}, u_1 \in \rho(t_1), u_2 \in \rho(t_2)\}$$

where $\Sigma' = \{\gamma^{(1)}, \alpha^{(0)}\}$.

It remains to prove that $\tau \notin \mathrm{XTOP}^{\mathrm{R}}$. Let $M = (Q, \Sigma, \Delta, I, R, c)$ be an $\mathrm{xtt}^{\mathrm{R}}$ such that $\tau_M = \tau$. Without loss of generality, suppose that M has no chain rules (see Lemma 4.2). Let n be an integer such that, for every $r \in R$, n is larger than the height of the trees t and u in $r = (q(t) \to u)$ and larger than the number of states of a dta recognizing $c(r)$. In addition, let n be larger than $\mathrm{card}(R)$. We denote the tree $\gamma^i(\alpha)$ simply by t_i for every $i \in \mathbb{N}$. Consider the input tree $s = \sigma(t_{2n}, t_{2n})$. Clearly, there must exist an initial state $q \in I$, a rule $r \in R$, and trees $u_1, u_1', u_2, u_2' \in T_\Delta$ such that $(u_1, u_2) \neq (u_1', u_2')$ and

$$\begin{aligned}
q(s) &\Rightarrow_M^r \xi \Rightarrow_M^* \delta(u_1, u_1, u_2, u_2) \\
q(s) &\Rightarrow_M^r \xi \Rightarrow_M^* \delta(u_1', u_1', u_2', u_2') \ .
\end{aligned}$$

The previous statement holds because there are less than n rules, but more potential outputs, i.e.,

$$\mathrm{card}(R) \leq \mathrm{card}(\{v \mid (s, v) \in \tau_M\}) \ .$$

So, at least two successful derivations have to start with the same rule. Moreover, we observe that $\{u_1, u_1', u_2, u_2'\} \subseteq \rho(t_{2n})$. Now we make a case distinction on the rule $r = (q(t) \to u)$.

(i) Suppose that $u = q'(x)$ for some $q' \in Q$ and $x \in \mathrm{var}(t)$. This case can easily be proved along the lines of item (i) in the proof of Theorem 5.2.

(ii) Suppose that $u = \delta(v_1, v_2, v_3, v_4)$ for some $v_1, v_2, v_3, v_4 \in T_\Delta(Q(X))$. Since $\mathrm{ht}(u_1) = \mathrm{ht}(u_2) = 2n+1$ but $\mathrm{ht}(v_i) < n$, it is clear that v_1, v_2, v_3, and v_4 each contain a variable. Moreover, it is apparent that they contain only one leaf. Let v_i contain $q_i(y_i)$ with $q_i \in Q$ and $y_i \in X$ for every $i \in [4]$. Clearly,

$$\begin{aligned}
v_1[q_1(y_1) \leftarrow q_1(\mathrm{sub}_s(\mathrm{pos}_{y_1}(t)))] &\Rightarrow_M^* u_1\\
v_2[q_2(y_2) \leftarrow q_2(\mathrm{sub}_s(\mathrm{pos}_{y_2}(t)))] &\Rightarrow_M^* u_1'\\
v_3[q_3(y_3) \leftarrow q_3(\mathrm{sub}_s(\mathrm{pos}_{y_3}(t)))] &\Rightarrow_M^* u_2\\
v_4[q_4(y_4) \leftarrow q_4(\mathrm{sub}_s(\mathrm{pos}_{y_4}(t)))] &\Rightarrow_M^* u_2' \ .
\end{aligned}$$

Consequently, $q(s) \Rightarrow_M^* \delta(u_1, u_1', u_2, u_2')$, which yields $(s, \delta(u_1, u_1', u_2, u_2')) \in \tau_M$. This is a contradiction because $(u_1, u_2) \neq (u_1', u_2')$.
Both cases are contradictory, thus $\tau \notin \mathrm{XTOP}^{\mathrm{R}}$. ∎

THEOREM 5.4 (Nonclosure under composition). *Except* ln-TOP *and* l-TOP$^{\mathrm{R}}$*, no class displayed in Fig. 4.5 is closed under composition.*

Proof. The nonclosure results are proved in Lemmata 5.1 and 5.3 and Theorem 5.2. ∎

6. Conclusion and open problems. We have provided a first in-depth analysis of extended top-down tree transducers designed for use in computational linguistics applications. We have demonstrated the circumstances under which extended transducers improve expressiveness over top-down tree transducers, and we have shown examples from machine translation that motivate this improved expressiveness. Our more general aim is to devise models that best explain the transformations we observe in empirical human language data, and to understand the formal properties of those models.

Several interesting problems remain open. In the light of composition hierarchy results for tree transformations computed by bimorphisms [2] and top-down tree transducers [11], it is interesting to study the composition closure of a class $\mathcal{L}$ of transformations computed by extended tree transducers. We showed that $\mathcal{L} \subset \mathcal{L}^2$ where $\mathcal{L}^2 = \mathcal{L}\,;\mathcal{L}$ (i.e., the hierarchy does not collapse at the first level), but it remains open whether an infinite hierarchy is formed (as it is the case for TOP [11]) or whether $\mathcal{L}^n = \mathcal{L}^{n+1}$ for some n (as it is the case, e.g., for (i) nl-TOP^1 [9] or (ii) l-TOP^2 [9] or (iii) B(lne-HOM, lne-HOM)^2 [2] or (iv) B(l-HOM, l-HOM)^4 [7, 8]).

In addition, it would be interesting to (syntactically) identify classes $\mathcal{L}$ such that ln-TOP $\subset \mathcal{L} \subset$ ln-XTOP and $\mathcal{L}$ is closed under composition. We showed (see Theorem 5.2) that no such class of transformations can handle flattenings or shuffles [cf. feature (X2)]. Since those features are important, it would also be interesting to identify classes $\mathcal{L} \subset$ ln-XTOP that can handle flattening or shuffle and are closed under composition (by Theorem 5.2 we necessarily have ln-TOP $\not\subseteq \mathcal{L}$).

REFERENCES

[1] ANDRÉ ARNOLD AND MAX DAUCHET, *Bi-transductions de forêts*, in Proc. 3rd Int. Coll. Automata, Languages and Programming, Edinburgh University Press, 1976, pp. 74–86.
[2] ———, *Morphismes et bimorphismes d'arbres.*, Theor. Comput. Sci., 20 (1982), pp. 33–93.
[3] BRENDA S. BAKER, *Composition of top-down and bottom-up tree transductions*, Inform. and Control, 41 (1979), pp. 186–213.
[4] NOAM CHOMSKY, *Aspects of the Theory of Syntax*, MIT Press, 1965.

[5] Bruno Courcelle and Paul Franchi–Zannettacci, *Attribute grammars and recursive program schemes*, Theor. Comput. Sci., 17 (1982), pp. 163–191 & 235–257.
[6] Max Dauchet, *Transductions inversibles de forêts*, Thèse 3ème cycle, Université de Lille, 1975.
[7] ———, *Transductions de forêts. Bimorphismes de magmoïdes*, Thèse d'État, Université de Lille, 1977.
[8] Max Dauchet and Sophie Tison, *Structural complexity of classes of tree languages*, in Tree Automata and Languages, North-Holland, 1992, pp. 327–354.
[9] Joost Engelfriet, *Bottom-up and top-down tree transformations—a comparison*, Math. Syst. Theory, 9 (1975), pp. 198–231.
[10] ———, *Top-down tree transducers with regular look-ahead*, Math. Syst. Theory, 10 (1976), pp. 289–303.
[11] ———, *Three hierarchies of transducers*, Math. Syst. Theory, 15 (1982), pp. 95–125.
[12] Joost Engelfriet, Eric Lilin, and Andreas Maletti, *Extended multi bottom-up tree transducers*, in Proc. 12th Int. Conf. Developments in Language Theory, vol. 5257 of LNCS, Springer-Verlag, 2008, pp. 289–300.
[13] Joost Engelfriet and Heiko Vogler, *Macro tree transducers*, J. Comput. System Sci., 31 (1985), pp. 71–146.
[14] ———, *Modular tree transducers*, Theor. Comput. Sci., 78 (1991), pp. 267–303.
[15] Zoltán Fülöp, *On attributed tree transducers*, Acta Cybernet., 5 (1981), pp. 261–279.
[16] Ferenc Gécseg and Magnus Steinby, *Tree Automata*, Akadémiai Kiadó, 1984.
[17] ———, *Tree languages*, in Handbook of Formal Languages, G. Rozenberg and A. Salomaa, eds., vol. 3, Springer-Verlag, 1997, ch. 1, pp. 1–68.
[18] Jonathan Graehl and Kevin Knight, *Training tree transducers*, in Proc. 2004 Human Language Technology Conf. NAACL, 2004, pp. 105–112.
[19] John E. Hopcroft and Jeffrey D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.
[20] Daniel Jurafsky and James H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Processing*, Prentice-Hall, 2000.
[21] Kevin Knight and Jonathan Graehl, *An overview of probabilistic tree transducers for natural language processing*, in Proc. 6th Int. Conf. Intelligent Text Processing and Computational Linguistics, 2005, pp. 1–24.
[22] Andreas Maletti, *Compositions of extended top-down tree transducers*, Inform. Comput., 206 (2008), pp. 1187–1196.
[23] Chris Manning and Hinrich Schütze, *Foundations of Statistical Natural Language Processing*, MIT Press, 1999.
[24] Jean-Claude Raoult, *Rational tree relations*, Bull. Belg. Math. Soc., 4 (1997), pp. 149–176.
[25] William C. Rounds, *Mappings and grammars on trees*, Math. Syst. Theory, 4 (1970), pp. 257–287.
[26] Stuart M. Shieber, *Synchronous grammars as tree transducers*, in Proc. 7th Int. Workshop Tree Adjoining Grammar and Related Formalisms, 2004, pp. 88–95.
[27] James W. Thatcher, *Generalized^2 sequential machine maps*, J. Comput. System Sci., 4 (1970), pp. 339–367.
[28] ———, *Tree automata: an informal survey*, in Currents in the Theory of Computing, Prentice Hall, 1973, pp. 143–172.