SURVEY: TREE TRANSDUCERS IN MACHINE TRANSLATION

Andreas Maletti*
Universitat Rovira i Virgili, Departament de Filologies Romàniques
Av. Catalunya 35, 43002 Tarragona, Spain.
Email: `andreas.maletti@urv.cat`

Abstract
In this survey, several model of tree transducers are investigated with respect to properties that are relevant in machine translation. These properties include: suitable expressiveness, symmetry, preservation of regularity, and closure under composition. For three tree transducer models, top-down tree transducers, extended top-down tree transducers, and extended multi bottom-up tree transducers, the relevant results are presented in an informal and illustrative manner. The aim of the survey is provide a synopsis that illustrates how theory (tree transducers) and practice (machine translation) interact on this particular example. Additional details can be found in the original results that are referenced throughout the text.

1. Introduction

The two research areas of *tree automata and tree transducers* [22, 23, 20] and *computational linguistics* [33, 27] started on a common basis. The first tree-transducer model, the top-down tree transducer, was developed by Thatcher [41] and Rounds [38] to formalize the notion of transformational grammars of natural language, which was proposed by Chomsky [7]. Unfortunately, the two areas diverged rather soon. The research in automata theory focused on new and generalized models, which overcome certain expressiveness shortcomings and algorithmic problems. For example, the following tree-transducer or related models were introduced and investigated:

- bottom-up tree transducers [42] and attributed tree transducers [17],
- macro tree transducers [8, 9, 14] and modular tree transducers [15],
- tree bimorphisms [2] and various models with synchronization [37].

However, computational linguists had turned to simpler transducer models based on strings such as finite-state transducers [26]. Those transducers are easy to illustrate, to implement, to generalize to a probabilistic setting, and to train on large quantities of linguistic data. With the introduction of the IBM-models [5], the statistical approach to machine translation and with it finite-state transducers became the de-facto standard in machine translation. With an

*Author was financially supported by the *Ministerio de Educación y Ciencia* (MEC) grant JDCI-2007-760.

ever growing amount of available linguistic data and engineering improvements that enabled computational linguists to utilize those large amounts of data, the translation quality of machine translation systems improved steadily during the 90s. However, finite-state transducers are, in general, not expressive enough for many applications in natural language processing because they cannot perform certain translations in a systematic manner. A finite-state transducer can learn any particular translation, but it might not be able to generalize properly (linguistically) due to its limited expressiveness.

Recently, computational linguists have turned back to tree-transducer models for problems such as machine translation [28], question answering, and summarization. In these applications, it is important to directly manipulate the phrase structure (the parse tree), which is imposed on sentences of natural language. For example, tree transducers, which operate on this phrase structure, have the ability to move complete syntactical structures (subtrees) or process different phrase structures (subtrees) differently. In particular, this led to a reconnection of the two research areas [39, 40]. In this survey, we will focus on unweighted models of tree transducers in machine translation. It should be noted that in most application areas weighted versions are used, but to keep the presentation simple, we present only unweighted devices.

Shieber [39] and others have argued that top-down tree transducers [41, 38] are generally inadequate for linguistic tasks because they cannot straightforwardly model local rotation, which is the reordering of subtrees at different tree depths. Top-down tree transducers need to use their copying power to realize this kind of transformation. However, copying top-down tree transducers do not have nice algorithmic properties, which led to the conclusion that they are generally inadequate. This drawback was addressed with the generalization of top-down tree transducers to extended top-down tree transducers, which were mentioned in [38] and have been investigated in [1, 2, 24, 29, 32]. This new model does not suffer from the outlined representational inadequacies, but none of the natural classes of translations computed by it is closed under composition [2, 32]. This new problem was addressed in [30, 13, 31], where the extended multi bottom-up tree transducer is proposed as an alternative. Its expressiveness is generally higher than that of the corresponding extended top-down tree transducers and the standard composition results hold true, which means that linear nondeleting and linear extended multi bottom-up tree transducers compute a class of translations that is closed under composition. However, the additional power yields that other important operations fail, in general. For example, even translations computed by linear nondeleting extended multi bottom-up tree transducers do not preserve the regularity of tree languages.

In this survey, we recall the 3 mentioned models and compare them with respect to a number of linguistically and theoretically motivated properties:

- Expr: The model is suitably expressive for applications in machine translation, which for the models under consideration boils down to the ability to handle local rotations.
- Sym: The model is symmetric, which means that for every translation that can be computed with the model also its inverse translation can be computed. This property allows us consider a tree transducer that translates English to German as a translator from

German to English simply by inverting it.

- Pres, Pres^{-1}: The translations computed by the model preserve regularity [22, 23] (Pres) and their inverses preserve regularity (Pres^{-1}). Regular tree languages are finitely representable using regular tree grammars [22, 23] and the property demands that the image (and preimage) of such a language under the translation remains regular (i.e., representable by a regular tree grammar).
- Comp: The class of translations computed by the model is closed under composition; i.e., for every two translations in the class also their composition is in the class.

We demonstrate for each mentioned model which properties it fulfills and which it fails. We generally try to illustrate the necessary constructions and counterexamples. The interested reader may then consult the original literature, which we indicate, to find the formal constructions and proofs. The aim of this survey is to provide an overview of the properties that the various tree-transducer models have. In addition, the sketched constructions shall serve as a starting point for the understanding of the general constructions. Finally, the survey shall raise the awareness on both sides (automata theory and computational linguistics) and provide an easy, intuitive access to the tree-transducer literature.

Overall, the survey is structured as follows: Section 2 recalls basic notions and notation. In Section 3 we present the first model, which is the classical top-down tree transducer. In this section, we also explain the linguistic and algorithmic motivation for some of the properties in our list. Section 4 continues the investigation with extended top-down tree transducers. The last model, the extended multi bottom-up tree transducer, is discussed in Section 5. In the last section (Section 6) we will present a table summarizing our findings. To increase the usefulness of the summary table, we add some additional models that we do not discuss in detail in this survey to the table.

2. Notation

The set of nonnegative integers is $\mathbb{N}$, and we let $[n] = \{1, 2, \dots, n\}$ for every $n \in \mathbb{N}$. We use the fixed set $\mathrm{X} = \{\mathrm{x}_1, \mathrm{x}_2, \dots\}$ of (formal) variables and its subsets $\mathrm{X}_n = \{\mathrm{x}_i \mid i \in [n]\}$ for every $n \in \mathbb{N}$. Now, let A, B, and C be sets. Any subset of $A \times B$ is a relation (or translation) from A to B. Let $\tau \subseteq A \times B$ and $\tau' \subseteq B \times C$ be relations. The inverse relation τ^{-1} of τ is $\tau^{-1} = \{(b, a) \mid (a, b) \in \tau\}$. Moreover, the composition $\tau;\tau'$ of τ and τ' is the relation $\tau;\tau' \subseteq A \times C$ given by

$$\tau \,;\, \tau' = \{(a, c) \mid \exists b \in B\colon (a, b) \in \tau, (b, c) \in \tau'\} \enspace .$$

Finally, the image $\tau(S)$ of $S \subseteq A$ under τ is $\tau(S) = \{b \in B \mid \exists a \in S\colon (a, b) \in \tau\}$, and the preimage of $T \subseteq B$ under τ is $\tau^{-1}(T)$. These notions for relations extend to classes of relations in the standard manner. A relation on B is simply a subset of $B \times B$. For every $L \subseteq B$, we let $\mathrm{id}_L = \{(b, b) \mid b \in L\}$. The reflexive transitive closure of a relation $\tau \subseteq B \times B$ is denoted by τ^*.

An alphabet Σ is a nonempty, finite set of symbols. For every $\mathrm{rk}\colon \Sigma \to \mathbb{N}$, the pair (Σ, rk) is a ranked alphabet, and a symbol $\sigma \in \Sigma$ has rank $\mathrm{rk}(\sigma)$. We simply write Σ instead of (Σ, rk) whenever the mapping rk is clear from the context or irrelevant. Moreover, we let $\Sigma_k = \{\sigma \in \Sigma \mid \mathrm{rk}(\sigma) = k\}$ for every $k \in \mathbb{N}$. If we want to make the rank $k = \mathrm{rk}(\sigma)$ of a symbol $\sigma \in \Sigma$ explicit, then we write σ^k.

Let Σ be a ranked alphabet and S a set. For every ranked subalphabet $\Delta \subseteq \Sigma$ and set T, we let

$$\Delta(T) = \{\delta(t_1, \ldots, t_k) \mid k \in \mathbb{N}, \delta \in \Delta_k, t_1, \ldots, t_k \in T\} \ .$$

The set $T_\Sigma(S)$ of all Σ-trees indexed by S is the smallest set T such that $S \cup \Sigma(T) \subseteq T$. We generally assume that $\Sigma \cap S = \emptyset$, and thus may write $\alpha()$ simply as α for every $\alpha \in \Sigma_0$. We let $T_\Sigma = T_\Sigma(\emptyset)$. By $\mathrm{var}(t)$ we denote the set $\{x \in \mathrm{X} \mid x \text{ occurs in } t\}$. For every $V \subseteq \mathrm{X}$, a tree $t \in T_\Sigma(\mathrm{X})$ is V-linear (respectively, V-nondeleting) if every $x \in V$ occurs at most (respectively, at least) once in t. A tree $t \in T_\Sigma(V)$ is a V-context if it is both V-linear and V-nondeleting. The set of all such V-contexts is $C_\Sigma(V)$. Any mapping $\theta \colon V \to T_\Sigma(S)$ is a substitution. The application $t\theta$ of the substitution θ to a tree $t \in T_\Sigma(S)$ is defined by $x\theta = \theta(x)$ for every $x \in V$ and $\sigma(t_1, \ldots, t_k)\theta = \sigma(t_1\theta, \ldots, t_k\theta)$ for every $\sigma \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma(S)$. If $V = \mathrm{X}_k$, then we also write $t[\theta(\mathrm{x}_1), \ldots, \theta(\mathrm{x}_k)]$ instead of $t\theta$.

Next, we recall particular term-rewrite systems [3]. Let $R \subseteq \bigcup_{k \in \mathbb{N}} C_\Sigma(\mathrm{X}_k) \times T_\Sigma(\mathrm{X}_k)$ be a finite relation. In the following, such sets R are called rule sets, in which Σ is often the union of an input and output ranked alphabet. To avoid technical difficulty, we simply assume that such unions can be formed (by renaming conflicting symbols). Note that each rule set is a ranked alphabet, where the rank k of a rule $(l, r) \in R$ is such that $l \in C_\Sigma(\mathrm{X}_k)$. We typically write $(l, r) \in R$ as $l \to r$. A rule set R is linear (respectively, nondeleting) if r is X_k-linear (respectively, X_k-nondeleting) for every $l \to r \in R$ of rank k. Any rule set R induces a rewrite relation $\Rightarrow_R$ on T_Σ as follows. For every $t, u \in T_\Sigma$, let $t \Rightarrow_R u$ if there exists a rule $l \to r \in R$ of rank k, a context $c \in C_\Sigma(\mathrm{X}_1)$, and a substitution $\theta \colon \mathrm{X}_k \to T_\Sigma$ such that $t = c[l\theta]$ and $u = c[r\theta]$.

Any subset of $T_\Sigma(V)$ is a tree language. The regular tree languages are those generated by regular tree grammars. A regular tree grammar [22, 23] is a tuple $G = (N, \Sigma, S, P)$ such that N is a finite set of nonterminals, Σ is a ranked alphabet of input symbols, $S \subseteq N$ is a set of start nonterminals, and $P \subseteq N \times T_\Sigma(N)$ is a finite set of productions. Clearly, P is a rule set (without variables) and its induced rewrite relation is $\Rightarrow_P$. The tree language generated by G is $L(G) = \{t \in T_\Sigma \mid \exists n \in S \colon n \Rightarrow_P^* t\}$. A tree language L is regular if it can be generated by some regular tree grammar.

A relation $\tau \subseteq T_\Sigma \times T_\Delta$ is also called tree transformation or tree translation. A mapping $f \colon T_\Sigma \to T_\Delta$ is a tree homomorphism, if for every $\sigma \in \Sigma_k$ there exists $f(\sigma) \in T_\Delta(\mathrm{X}_k)$ such that $f(\sigma(t_1, \ldots, t_k)) = f(\sigma)[f(t_1), \ldots, f(t_k)]$ for every $t_1, \ldots, t_k \in T_\Sigma$. A tree homomorphism $f \colon T_\Sigma \to T_\Delta$ is a particularly simple tree translations. It is linear (respectively, complete) if $f(\sigma)$ is X_k-linear (respectively, X_k-nondeleting) for every $\sigma \in \Sigma_k$.

3. Top-down Tree Transducer

The first tree transducer model that we recall is the *top-down tree transducer*. It was introduced by [38, 41] to formalize the model of transformational grammars [6], which was originally proposed by CHOMSKY and his followers. The phrase-structure models of natural language required formal devices that manipulate those phrase structures, which were and are typically represented as trees. To simplify the following presentation, we assume that those phrase-structure trees (or parse trees) are ranked trees, which means that each symbol has a fixed rank. This is uncommon in NLP, so whenever it is ambiguous, then we indicate that a symbol S has rank n by writing S^n (see Figure 2).

A *top-down tree transducer* (tdtt) is a tuple $M = (Q, \Sigma, \Delta, I, R)$ where

- Q is a finite set of *states* (considered as unary symbols; i.e., symbols of rank 1),
- Σ and Δ are ranked alphabets of *input* and *output symbols* with $Q \cap (\Sigma \cup \Delta) = \emptyset$,
- $I \subseteq Q$ is a set of *initial states*, and
- $R \subseteq Q(\Sigma(\mathrm{X})) \times T_\Delta(Q(\mathrm{X}))$ is a finite *rule* set.

The tdtt M is *linear* (respectively, *nondeleting*) if the rules R are. Clearly, the rules R of the tdtt M give rise to a term-rewrite relation $\Rightarrow_R^*$. The semantics of the tdtt M is $\tau_M = \{(t, u) \in T_\Sigma \times T_\Delta \mid \exists q \in I \colon q(t) \Rightarrow_R^* u\}$, which is a relation $\tau_M \subseteq T_\Sigma \times T_\Delta$. In general, we can assume that each rule has the shape $q(\sigma(\mathrm{x}_1, \ldots, \mathrm{x}_k)) \to u$ where $q \in Q$, $\sigma \in \Sigma_k$, and $u \in T_\Delta(Q(\mathrm{X}_k))$.

As an illustration, let us consider the following tdtt $M = (Q, \Sigma, \Delta, \{q\}, R)$ where

- $Q = \{q, \underline{1}, \underline{2}^+, \underline{3}^+\}$,
- $\Sigma = \{\mathit{zero}, \mathit{one}, \ldots, \mathit{twelve}, \mathit{twen\text{-}}, \mathit{thir\text{-}}, \mathit{fif\text{-}}, \mathit{eigh\text{-}}, \mathit{\text{-}teen}^1, \mathit{\text{-}ty}^2, \mathit{\text{-}ty}^1, \mathit{hundred}^2, \mathit{hundred}^1\}$,
- $\Delta = \{0, \ldots, 9, 100, 10\}$ where 100 and 10 are binary and all other symbols are nullary, and
- R contains the rules

$$\begin{array}{rrr}
q(\mathit{hundred}^2(\mathrm{x}_1, \mathrm{x}_2)) \to 100(q(\mathrm{x}_1), q(\mathrm{x}_2)) & q(\mathit{hundred}^1(\mathrm{x}_1)) \to 100(q(\mathrm{x}_1), 10(0, 0)) & \\
q(\mathit{\text{-}ty}^2(\mathrm{x}_1, \mathrm{x}_2)) \to 10(\underline{2}^+(\mathrm{x}_1), \underline{1}(\mathrm{x}_2)) & q(\mathit{\text{-}ty}^1(\mathrm{x}_1)) \to 10(\underline{2}^+(\mathrm{x}_1), 0) & \\
q(\mathit{\text{-}teen}^1(\mathrm{x}_1)) \to 10(1, \underline{3}^+(\mathrm{x}_1)) & & \\
q(\mathit{zero}) \to 10(0, 0) & \ldots & q(\mathit{nine}) \to 10(0, 9) \\
q(\mathit{ten}) \to 10(1, 0) & q(\mathit{eleven}) \to 10(1, 1) & q(\mathit{twelve}) \to 10(1, 2) \\
\underline{1}(\mathit{zero}) \to 0 & \ldots & \underline{1}(\mathit{nine}) \to 9 \\
\underline{2}^+(\mathit{twen\text{-}}) \to 2 & \underline{2}^+(\mathit{thir\text{-}}) \to 3 & \underline{2}^+(\mathit{four}) \to 4 \\
\underline{2}^+(\mathit{fif\text{-}}) \to 5 & \underline{2}^+(\mathit{six}) \to 6 & \underline{2}^+(\mathit{seven}) \to 7 \\
\underline{2}^+(\mathit{eigh\text{-}}) \to 8 & \underline{2}^+(\mathit{nine}) \to 9 & \\
\underline{3}^+(\mathit{thir\text{-}}) \to 3 & \underline{3}^+(\mathit{four}) \to 4 & \underline{3}^+(\mathit{fif\text{-}}) \to 5 \\
\underline{3}^+(\mathit{six}) \to 6 & \underline{3}^+(\mathit{seven}) \to 7 & \underline{3}^+(\mathit{eigh\text{-}}) \to 8 \\
\underline{3}^+(\mathit{nine}) \to 9\ . & &
\end{array}$$

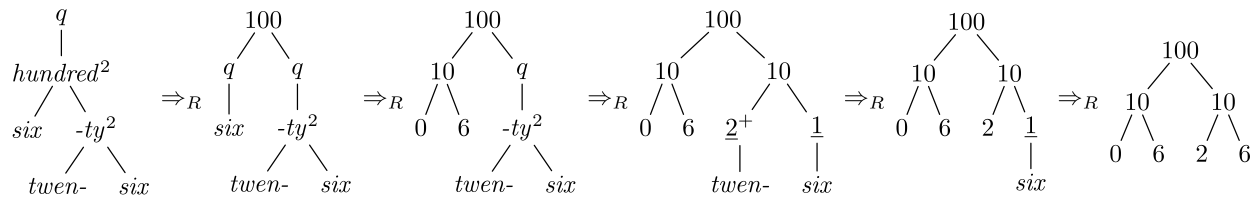

Figure 1: Example derivation using a linear nondeleting tdtt.

This tdtt M is linear and nondeleting. It can translate English representations of a limited set of numbers into a decimal-like notation. For example, it can perform the translation displayed in Figure 1. It also checks whether the number is properly represented in English. For example, it fails to translate any of the following:

$$\textit{-ty}^1(\textit{one}) \qquad \textit{hundred}^2(\textit{fif-}, \textit{-teen}^1(\textit{twelve})) \qquad \textit{-ty}(\textit{five}, \textit{zero})$$

because it considers them ill-shaped. In general, linear nondeleting tdtts are able to check whether the input tree is an element of a given regular tree language.

Let us see how this model fares in the machine-translation domain. Shieber [39] argues that linear tdtts are not expressive enough and that only general tdtts have sufficient expressive power to be considered for the machine-translation task. Some problematic translations are presented in Figure 2. Let us discuss them in some detail. The translation displayed left in Figure 2 is typical in the translation from English to Arabic [28]. The English determiner '*the*' in front of the noun '*boy*' is dropped in the Arabic translation. If we want to model this with a tdtt, then we need a rule $q(\mathrm{NP}^2(\mathrm{x}_1, \mathrm{x}_2)) \to u$ for some state q and output tree u. Clearly the output tree u must match (up to the state) the target tree, which yields that at most one call $p(\mathrm{x}_i)$ with state p and $i \in [2]$ can be contained in u. This immediately yields that one subtree (either the determiner in the first subtree or the noun in the second subtree) is not considered by the tdtt, which thus must be deleting. However, to perform the translation the noun '*boy*' should be considered because the tdtt is required to produce its translation '*atefl*'. Thus, assume that the rule is $q(\mathrm{NP}^2(\mathrm{x}_1, \mathrm{x}_2)) \to \mathrm{NP}^1(p(\mathrm{x}_2))$. In that scenario, the tdtt cannot determine whether there is only a determiner in the first subtree. In fact, the first subtree could as well contain material that has to be translated into Arabic. Even if we add special rules for those situations, the permissive rule $q(\mathrm{NP}^2(\mathrm{x}_1, \mathrm{x}_2)) \to \mathrm{NP}^1(p(\mathrm{x}_2))$ allows the tdtt to just ignore the first subtree, which is clearly not desirable. It can be argued that this is purely a problem of annotation and that the parse tree for the English side should be adjusted. Indeed, this particular problem can be resolved in this manner, but the phrase-structure annotation is standardized and the linguistic resources are annotated this way. In addition, an adaption of the parse trees to each particular language pair that is used in machine translation seems unfeasible.

The problem that a tdtt cannot check deleted subtrees was also identified in the tree-transducer literature [10] because it is a shortcoming of linear tdtts when compared to linear bottom-up tree transducers [10, 42] (linear bots). Since this missing feature, called *regular look-ahead*, also impacts on closure under composition, it was introduced to tdtts [11], which essentially

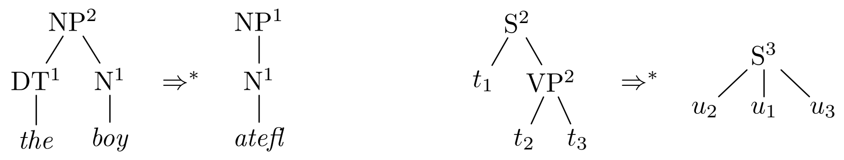

Figure 2: Problematic translations for linear tdtts.

allows them to check whether the current input tree can be generated by a specific regular tree grammar before a rule is applied. In this manner, a tdtt with regular look-ahead can first check the regular property whether the first subtree is indeed only a determiner before it executes the rule $q(\mathrm{NP}^2(\mathrm{x}_1, \mathrm{x}_2)) \to \mathrm{NP}^1(p(\mathrm{x}_2))$. We will not formally recall tdtts with regular look-ahead here, but we will include them in our summary tables.

The right translation of Figure 2 also represents a common translation rule from English to Arabic. It represents the fact that English is predominantly 'S-V-O' (subject-verb-object), which is represented by the phrase structure $\mathrm{S}^2(t_1, \mathrm{VP}^2(t_2, t_3))$ where t_1, t_2, t_3 are (in order) the parse tree of the subject, the verb, and the object. In contrast, the dominating sentence structure in Arabic is 'V-S-O', which is represented by the flat phrase structure $\mathrm{S}^3(u_2, u_1, u_3)$, where u_i is the translation of t_i (i.e., u_1, u_2, u_3 are in order the translated subject, the translated verb, and the translated object). It can easily be seen that such a translation cannot be achieved by a linear tdtt, which is also formally demonstrated in [32]. However, general tdtts can induce this translation using, for example, the following rules:

$$q(\mathrm{S}^2(\mathrm{x}_1, \mathrm{x}_2)) \to \mathrm{S}^3(q_1(\mathrm{x}_2), p(\mathrm{x}_1), q_2(\mathrm{x}_2)) \qquad q_1(\mathrm{VP}^2(\mathrm{x}_1, \mathrm{x}_2)) \to p(\mathrm{x}_1) \qquad q_2(\mathrm{VP}^2(\mathrm{x}_1, \mathrm{x}_2)) \to p(\mathrm{x}_2)\ .$$

Note that any tdtt with such rules is both copying (i.e., nonlinear) and deleting. This leads to our first conclusion:

Expr: *Top-down tree transducers with regular look-ahead are expressive enough for machine translation. Linear tdtts or tdtts without regular look-ahead are not expressive enough.*

In general, tdtts are also not symmetric, which means that for a given tdtt M, we might be unable to find a tdtt with the same properties (linear, nondeleting, with or without regular look-ahead) as M that computes τ_M^{-1}. This can easily be seen on the translations sketched in Figure 2. Their inverses can easily be realized by linear nondeleting tdtts. Moreover, copying tdtts can easily produce two equal copies of an input subtree [10]. Any tdtt that is supposed to compute the inverse of the translation of such a tdtt necessarily must check the equality of two output subtrees, which are part of its input tree. However, the domain $\tau^{-1}(T_\Delta)$, which is the language of all input trees that can be translated, of a tdtt M is regular [10], which proves that such an equality check is not possible.

Sym: *Top-down tree transducers are not symmetric.*

Given a compact representation of a tree language of input or output trees, it is often desirable to compute a compact representation of the corresponding translations. To keep the presentation simple, we restrict ourselves to regular tree languages here, which can be finitely represented by regular tree grammars as recalled in Section 2. Thus, it would be desirable if the image and preimage of a regular tree language under the translation τ_M of a tdtt M would again be regular. Since tdtts are not symmetric, we have to consider both the image and the preimage separately. As already argued, the ability of a copying tdtt M to generate identical copies of input subtrees clearly means that the image of a regular tree language under its translation τ_M need not be regular anymore. However, [10, 11] proves that the preimage of a regular tree language under τ_M for every tdtt M is again regular and that the image of a regular tree language under τ_M for every a linear tdtt M is regular. We will show how to prove this result in the next section, where we deal with more general top-down tree transducers.

> PRES, PRES^{-1}: *Linear tdtts preserve regularity and the preimage of a regular tree language under τ_M is regular for each tdtt M.*

Finally, we consider whether the class of translations computed by various tdtts is closed under composition. Composition is an important operation because it supports a modular development of the final translation system. Special components (like number translation, proper-name translation, etc.) can be hand-crafted or trained on special resources and then composed with the other modules to obtain the final transducer. For example, the translation system described in [43] consists of 3 modules: reordering, insertion, and word-translation. Of course, such a chain (or cascade) of compositions can also be evaluated every time an input is provided [35]. However, this can prove to be very inefficient, especially when the chain is evaluated for many inputs (like a production machine-translation system). For tdtts this bucket-brigade approach is feasible because for every input tree a tdtt can produce only finitely many translations (and thus a regular tree language). However, if we start with a regular tree language L as input, then it depends on regularity-preservation whether the image of L under τ_M for a tdtt M is still regular.

Classical composition, which composes two tdtts into a single one, can be used to reduce a chain of tdtts to a single tdtt that computes the same translation as the chain of tdtts. This is a preprocessing step. At evaluation time this avoids the computation of the intermediate results that occur in the bucket-brigade approach. Note that offline composition can succeed even if regularity is not preserved. An example of this phenomenon is presented in Section 5. The standard composition strategy is to process the right-hand sides of the rules with the next transducer in the chain. Let us illustrate this on a rule of the example tdtt presented earlier. Suppose that we have another tdtt that translates the decimal-like representation into a German representation. Such a tdtt could have a rule set R' that contains (among others) the following rules

$$p(100(\mathrm{x}_1, \mathrm{x}_2)) \to \mathit{hundert}(p(\mathrm{x}_1), p(\mathrm{x}_2)) \qquad p(10(\mathrm{x}_1, x_2)) \to \mathit{zehn}(\underline{1}(\mathrm{x}_2), \mathrm{zig}(\mathrm{x}_2)) \qquad \underline{1}(0) \to \varepsilon \qquad \mathrm{zig}(0) \to \varepsilon \enspace .$$

Figure 3 shows how to derive a new rule from a rule of our example tdtt using the rules just presented. The obtained rule of the composed tdtt is

$$\langle p, q\rangle(\mathit{hundred}^1(\mathrm{x}_1)) \to \mathit{hundert}(\langle p, q\rangle(\mathrm{x}_1), \mathit{zehn}(\varepsilon, \varepsilon)) \enspace ,$$

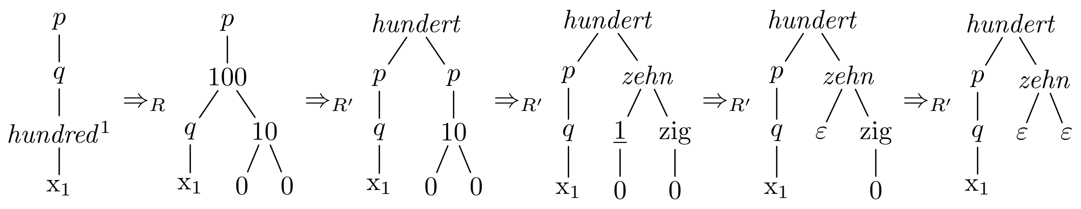

Figure 3: Illustration of composition for linear nondeleting tdtts.

Model \ Criterion	EXPR	SYM	PRES	PRES^{-1}	COMP
Linear nondeleting tdtt	✗	✗	✓	✓	✓
Linear tdtt	✗	✗	✓	✓	✗
Linear tdtt with regular look-ahead	✗	✗	✓	✓	✓
General tdtt	✗	✗	✗	✓	✗
General tdtt with regular look-ahead	✓	✗	✗	✓	✗

Table 1: Overview of formal properties of top-down tree transducers.

where we paired the states to obtain a state of the composed tdtt. This simple composition strategy works for linear nondeleting tdtts and for linear tdtts with regular look-ahead. In the latter case the regular look-ahead needs to be handled, but we will not detail it here.

In general, nonlinear tdtts cannot be composed, which is demonstrated in [10]. Similarly, linear tdtts (without regular look-ahead) do not compute a class of translations that is closed under composition. This is best illustrated by the problematic translation displayed left in Figure 2, which can be achieved by a composition of two linear tdtts. The first linear tdtt would just inspect the input tree and mark the node 'NP' should its first subtree indeed contain only a determiner (as in the example). The second linear tdtt can then safely delete the subtree containing the determiner provided that the mark is present. Since we already demonstrated that no single linear tdtt can achieve this, a chain of two linear tdtts must be more powerful than a single one. In fact, chains of linear tdtts are as powerful as a single linear tdtt with regular look-ahead, which is as powerful as a chain of two linear tdtts.

COMP: *Only linear tdtts with regular look-ahead and linear nondeleting tdtts compute a class of translations that is closed under composition.*

The properties of the individual models discussed in this section are summarized in Table 1, where we use

- EXPR: for suitable expressiveness in the machine-translation task,
- SYM: for symmetry,
- PRES: for preservation of regularity,
- PRES^{-1}: for preservation of regularity of the inverse translation, and
- COMP: for closure under composition.

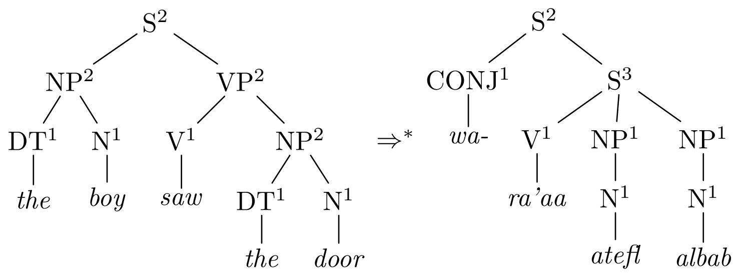

Figure 4: English-to-Arabic translation that can be computed by an extended top-down tree transducer.

Mind that we often identify the syntactic model with the class of translations it computes.

4. Extended Top-down Tree Transducer

Next we recall *extended top-down tree transducers*. Since linear tdtts are not expressive enough and nonlinear tdtts do not have good algorithmic properties, this alternative model was proposed for applications in machine translation by [24, 25]. In fact, an equivalent model, the *linear complete bimorphism* was already investigated in [1, 2]. We will mention this connection later when we consider preservation of regularity. A systematic study of extended top-down tree transducers can be found in [32].

An *extended top-down tree transducer* (xtt) is a tuple $M = (Q, \Sigma, \Delta, I, R)$ where

- Q is a finite set of *states* (considered as unary),
- Σ and Δ are ranked alphabets of *input* and *output symbols* with $Q \cap (\Sigma \cup \Delta) = \emptyset$,
- $I \subseteq Q$ is a set of *initial states*, and
- $R \subseteq Q(T_\Sigma(\mathrm{X})) \times T_\Delta(Q(\mathrm{X}))$ is a finite *rule* set.

In other words, the rules of an xtt can have an arbitrary context in the left-hand side instead of contexts that contain exactly one input symbol as in tdtts. As before M is *linear* (respectively, *nondeleting*) if the rules R are. In the natural language processing literature a slightly weaker version of linear nondeleting xtt (without states) is known as *synchronous tree-substitution grammar* [39]. Clearly, also the rules R of an xtt M give rise to a term-rewrite relation $\Rightarrow_R^*$. The semantics of the xtt M is $\tau_M = \{(t, u) \in T_\Sigma \times T_\Delta \mid \exists q \in I \colon q(t) \Rightarrow_R^* u\}$.

Figure 4 shows a translation from English to Arabic. It can be achieved with the following xtt $(Q, \Sigma, \Delta, \{q\}, R)$ where

- $Q = \{q, p, r\}$,
- $\Sigma = \{\mathrm{S}^2, \mathrm{NP}^2, \mathrm{VP}^2, \mathrm{DT}^1, \mathrm{N}^1, \mathrm{V}^1, \textit{the}, \textit{boy}, \textit{saw}, \textit{door}\}$,
- $\Delta = \{\mathrm{S}^3, \mathrm{S}^2, \mathrm{CONJ}^1, \mathrm{V}^1, \mathrm{N}^1, \mathrm{NP}^1, \textit{wa-}, \textit{ra'aa}, \textit{atefl}, \textit{albab}\}$, and
- R contains the following rules:

$$q(\mathrm{x}_1) \to \mathrm{S}^2(\mathrm{CONJ}^1(\textit{wa-}), q(\mathrm{x}_1)) \qquad p(\mathrm{NP}^2(\mathrm{DT}^1(\textit{the}), \mathrm{N}^1(\textit{boy}))) \to \mathrm{NP}^1(\mathrm{N}^1(\textit{atefl}))$$

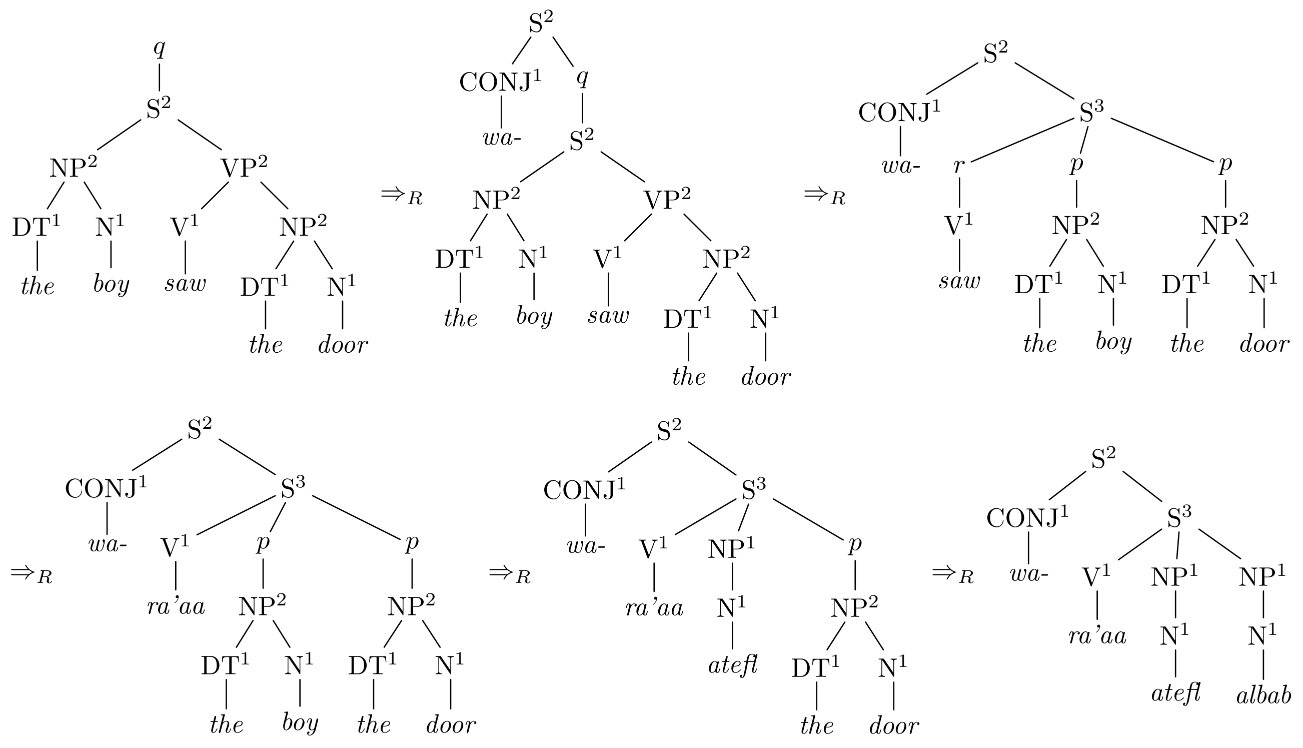
Figure 5: Derivation using the example rules that realizes the translation of Figure 4.

$$q(\mathrm{S}^2(\mathrm{x}_1, \mathrm{VP}^2(\mathrm{x}_2, \mathrm{x}_3))) \to \mathrm{S}^3(r(\mathrm{x}_2), p(\mathrm{x}_1), p(\mathrm{x}_3)) \qquad p(\mathrm{NP}^2(\mathrm{DT}^1(\textit{the}), \mathrm{N}^1(\textit{door}))) \to \mathrm{NP}^1(\mathrm{N}^1(\textit{albab}))$$
$$r(\mathrm{V}^1(\textit{saw})) \to \mathrm{V}^1(\textit{ra'aa}) \ .$$

This xtt is linear and nondeleting. An example derivation realizing the translation of Figure 4 is displayed in Figure 5.

The following plain-language properties of xtt distinguish the model from tdtts [32]:

- *Finite look-ahead*: Every xtt can inspect a finite part of subtrees it deletes.
- *Deep attachment of variables*, which enables local rotations like the one displayed right in Figure 2.
- *Infinitely many outputs for one input* with the help of ε-rules (i.e., rules whose left-hand side is $q(\mathrm{x}_1)$.

Overall, even linear nondeleting xtts have sufficient power for the application in machine translation, which is the main reason why the toolkit Tiburon [34] implements xtts. In fact, linear nondeleting xtts can be used to learn any finite corpus of bilingual data using the methods of [21, 24, 25].

Expr: *Even linear nondeleting xtts are expressive enough for applications in machine translation.*

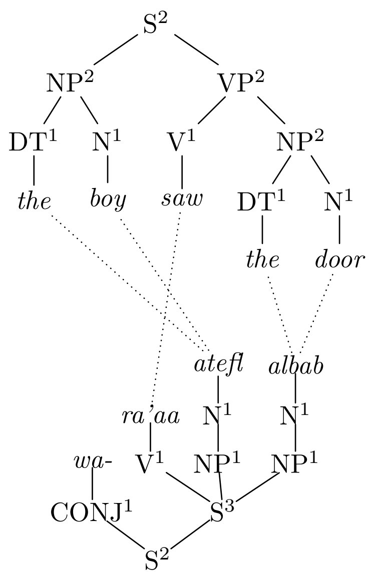

Figure 6: Training data for a linear and nondeleting xtt.

Let us quickly recall how we can obtain a linear and nondeleting xtt from bilingual training data. A bilingual corpus typically contains sentences in two languages like:

the boy saw the door
wa- ra'aa atefl albab

These sentences are parsed (automatically or by hand) to create, for example, the tree pair of Figure 4. Adding a state on top of the input tree, we could obtain a rule for a linear and nondeleting xtt. However, such rules are clearly too specific and overfit the data because we cannot translate any unseen data. To remedy this situation, a word aligner like GIZA^{++} [36] is typically used to relate the words in the sentences (i.e., which source words are or contribute to the translation of certain target words). Thus, the training material is a set of annotated tree pairs that look like the one in Figure 6.

Given such aligned tree pairs, we can now extract rules in a more clever fashion [21]. For example, the pair of subtrees $\mathrm{NP}^2(\mathrm{DT}^1(\textit{the}), \mathrm{N}^1(\textit{boy}))$ and $\mathrm{NP}^1(\mathrm{N}^1(\textit{atefl}))$ can be extracted because it is consistent with the alignments (no alignment between words outside to words inside the subtrees). In this way, the rules of the example xtt of this section can be extracted. Finally, as already mentioned in the Introduction, applications typically use weighted versions of the devices to distinguish more likely translations from less likely ones. For example, a rule that encodes the standard 'S-V-O' word order in English should be prefered (receive a "better" weight) to a rule that encodes a nonstandard word order. Essentially, the weight training procedure [24, 25] counts how often a particular rule is used in a certain corpus and assigns better weights to rules that are used more often.

Next, we present an alternative characterization that can be used prove certain results such as symmetry and preservation of regularity for certain xtt. A *linear and complete bimorphism* [2] is a tuple $B = (h_{\mathrm{i}}, L, h_{\mathrm{o}})$ such that $h_{\mathrm{i}} \colon T_\Gamma \to T_\Sigma$ and $h_{\mathrm{o}} \colon T_\Gamma \to T_\Delta$ are linear complete tree homomorphisms (called the input and the output homomorphism) and $L \subseteq T_\Gamma$ is a regular tree language (called the center language). The tree transformation computed by B is

$$\tau_B = \{(h_{\mathrm{i}}(s), h_{\mathrm{o}}(s)) \mid s \in L\} \enspace .$$

In other words, $\tau_B = h_{\mathrm{i}}^{-1} \,;\, \mathrm{id}_L \,;\, h_{\mathrm{o}}$.

We already remarked in Section 2 that each rule set is a ranked alphabet. Given a linear and nondeleting xtt $M = (Q, \Sigma, \Delta, I, R)$, we can construct tree homomorphisms $h_{\mathrm{i}} \colon T_R \to T_\Sigma$ and $h_{\mathrm{o}} \colon T_R \to T_\Delta$ and a weighted tree grammar $G = (Q, R, I, P)$ such that

- $h_{\mathrm{i}}(\rho) = l$,
- $h_{\mathrm{o}}(\rho) = r$, and
- $q \to \rho(q_1, \ldots, q_k) \in P$

for every rule $\rho = q(l) \to r\theta$ with $l \in C_\Sigma(\mathrm{X}_k)$, $r \in C_\Delta(\mathrm{X}_k)$, and $q, q_1, \ldots, q_k \in Q$ where θ is the substitution such that $\mathrm{x}_i\theta = q_i(\mathrm{x}_i)$ for every $i \in [k]$. Clearly, the constructed tree homomorphisms are linear and complete. The proof that the constructed bimorphism $(h_{\mathrm{i}}, L(G), h_{\mathrm{o}})$ is equivalent to the original xtt M is omitted here, but it can be found in [30]. Moreover, a similar characterization, in which the output tree homomorphism is only linear can be provided for linear xtt with regular look-ahead [30]. Finally, we remark that linear complete bimorphisms are equally powerful as linear nondeleting xtts. This immediately yields that linear nondeleting xtts are symmetric (because linear complete bimorphisms certainly are), whereas it can easily be proved that all stronger models of xtts do not have this property.

Sym: *Only linear nondeleting xtts are symmetric.*

With the help of the bimorphism characterization [30], we can thus represent each linear xtt as a chain of the inverse of a linear and complete tree homomorphism, an intersection with a regular tree language, and a linear tree homomorphism. Since inverses of tree homomorphisms and linear tree homomorphisms preserve regularity [22, 23] and regular tree languages are closed under intersection [22, 23], we can easily prove a number of results concerning preservation of recognizability for linear xtts using the bimorphisms constructed for them. The remaining results (displayed in Table 2) can easily be obtained and are detailed in [32].

Pres, Pres^{-1}: *Linear xtts with regular look-ahead preserve regularity and the preimage of every regular tree language is regular under τ_M for any xtt M.*

Finally, let us consider composition. Unfortunately, it is known [2] that even linear nondeleting xtts compute a class of translations that is not closed under composition. The counterexample of [2] is presented in Figure 7. Both individual translations can easily be implemented using linear nondeleting xtts, but the full translation cannot be implemented by a single linear and

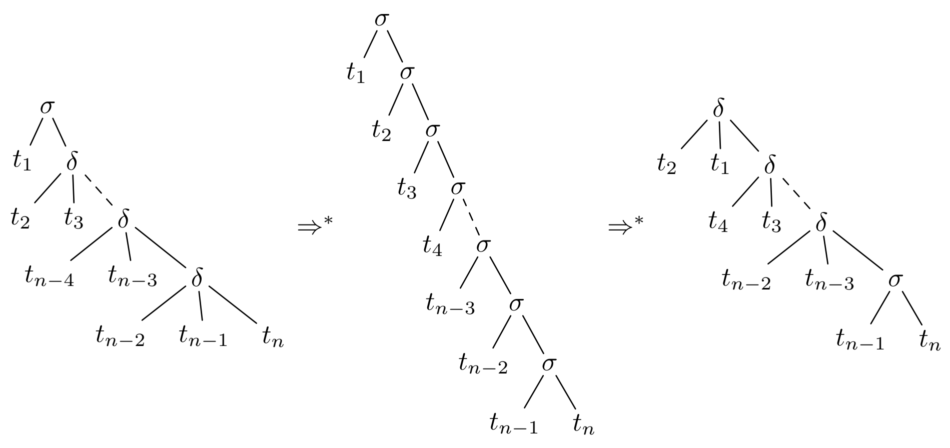

Figure 7: Counterexample of [2] demonstrating that linear nondeleting xtts cannot be composed.

nondeleting xtt, which is due to the "misalignment" of subtrees [2]. In fact, [32] shows that every class $\mathcal{L}$ of tree translations that

- contains all translations computed by linear nondeleting tdtts,
- is suitably expressive for machine translation, and
- contains only translations computable by linear xtts with regular look-ahead

is not closed under composition. This shows that inside the class of translations computed by linear xtts, there does not seem to be a suitable class for machine translation that is also closed under composition. Another result of [32] shows that also every class $\mathcal{L}$ of tree transformations that

- contains all translations computed by tdtts and
- contains only translations computable by xtts with regular look-ahead

is also not closed under composition. Overall, this yields that closure under composition and sufficient expressivity for machine translation exclude each other for most reasonable classes of xtts.

> COMP: *None of the discussed classes of xtts compute a class of translations that is closed under composition.*

The properties of the individual models of xtt are summarized in Table 2. We continue to use the abbreviations (for properties) that were introduced at the end of Section 3.

Model \ Criterion	Expr	Sym	Pres	Pres^{-1}	Comp
Linear nondeleting xtt	✓	✓	✓	✓	✗
Linear xtt	✓	✗	✓	✓	✗
Linear xtt with regular look-ahead	✓	✗	✓	✓	✗
General xtt	✓	✗	✗	✓	✗
General xtt with regular look-ahead	✓	✗	✗	✓	✗

Table 2: Overview of formal properties of extended top-down tree transducers.

5. Extended Multi Bottom-up Tree Transducer

Finally, we recall extended multi bottom-up tree transducers [18, 19, 30, 13]. Compared to [18, 19] we slightly adapt the model by omitting the special root symbol. This symbol is required in their model so that the bottom-up device may deterministically identify the root of the input tree. However, in natural language processing deterministic tree transducers have only very limited applications [29], so we will not deal with deterministic devices and can thus omit the special root symbol. In addition, we extend the multi bottom-up tree transducers [18, 19, 30] in the same manner as we extended tdtts to xtts.

An *extended multi bottom-up tree transducer* (xmbot) is a tuple $M = (Q, \Sigma, \Delta, F, R)$ such that

- Q is a ranked alphabet (of *states*),
- Σ and Δ are ranked alphabets (of *input* and *output symbols*) with $Q \cap (\Sigma \cup \Delta) = \emptyset$,
- $F \subseteq Q_1$ is a set of *final states*, and
- $R \subseteq T_\Sigma(Q(\mathrm{X})) \times Q(T_\Delta(\mathrm{X}))$ is a finite rule set.

As usual, we say that M is *linear* (respectively, *nondeleting*) if R is such. The semantics of the xmbot M is also given by the rewrite relation $\Rightarrow_R$. The tree transformation τ_M computed by M is

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

Let us present an example xmbot $M = (Q, \Sigma, \Sigma, \{f\}, R)$ where

- $Q = \{f^1, q^2\}$,
- $\Sigma = \{\sigma^2, a^1, b^1, e^0\}$, and
- R contains the following rules, which are also displayed in Figure 8:

$$a(q(\mathrm{x}_1, \mathrm{x}_2)) \to q(a(\mathrm{x}_1), a(\mathrm{x}_2)) \qquad q(\mathrm{x}_1, x_2) \to f(\sigma(\mathrm{x}_1, \mathrm{x}_2))$$
$$b(q(\mathrm{x}_1, \mathrm{x}_2)) \to q(b(\mathrm{x}_1), b(\mathrm{x}_2)) \qquad e \to q(e, e) \ .$$

This xmbot M is linear and nondeleting, and it computes the tree translation

$$\tau_M = \{(t, \sigma(t, t)) \mid t \in T_\Gamma\}$$

where $\Gamma = \{a^1, b^1, e^0\}$. An example derivation is presented in Figure 9. Overall, this shows

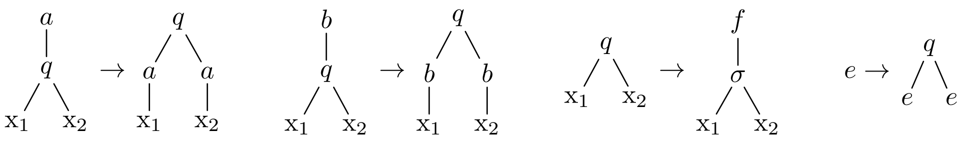

Figure 8: Example rules of an xmbot.

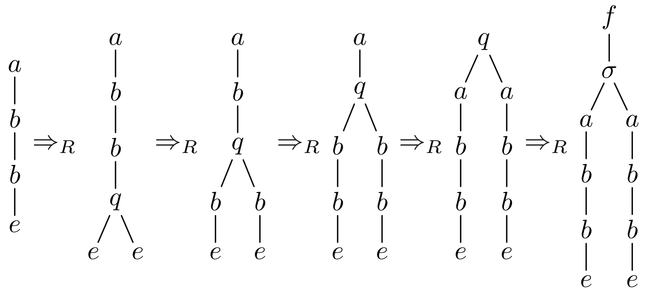

Figure 9: Example derivation using the example rules of Figure 8.

that even linear xmbots have some copying power. Indeed [13] shows that linear xmbots are *finitely copying* [12].

Next, let us consider whether xmbots are sufficiently powerful for machine translation. Since linear nondeleting xtts are powerful enough as demonstrated in Section 4, we simply show that each such xtt can be simulated by a linear and nondeleting xmbot. The simple transformation from an xtt to an xmbot is illustrated in Figure 10 on an example rule. Since the correspondence is very direct, we do not present a formal construction or proof. In general, the suitability and advantages of xmbots over xtts in the area of machine translation are discussed in detail in [31]. Overall, we showed that

> Expr: *Linear and nondeleting xmbots are expressive enough for applications in machine translation.*

We have already seen that even linear (linear nondeleting) xmbots have a certain copying power (see Figures 8 and 9). Thus, xmbots can produce identical copies of an input subtree in the output, however, an xmbot cannot check the equality of two input subtrees. Roughly speaking, a state can keep several (identical) output trees because its rank can be different from 1. On those output trees it can enforce equality by generating them synchronously. However, each

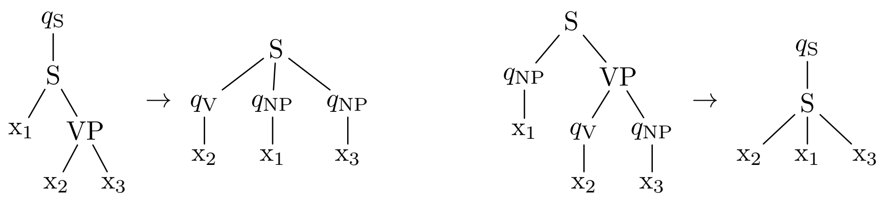

Figure 10: Transformation of an xtt rule (left) into an xmbot rule (right).

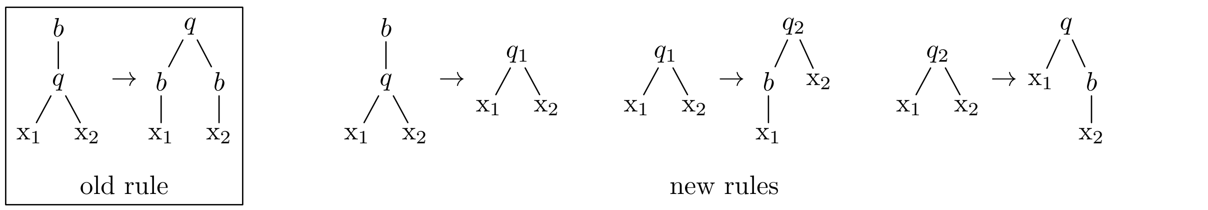

Figure 11: Transformation of an xmbot rule into 1-symbol normal-form.

state processes only one input subtree at a time, so it cannot enforce a similar condition on the input subtrees. This can also be seen on the bimorphism characterization of [2, 13], which shows that linear nondeleting xmbots are as powerful as linear complete bimorphisms, in which the output homomorphism is an m-morphism [2] instead of a tree homomorphism (which is the same as a 1-morphism).

> Sym: *Extended multi bottom-up tree transducers are not symmetric.*

The same argumentation can be used to prove that even linear nondeleting xmbots do not preserve regularity. Given our example xmbot M, the input tree language T_Γ is clearly regular, but its image $\tau_M(T_\Gamma)$ is $\{\sigma(t,t) \mid t \in T_\Gamma\}$, which is obviously not regular. Using the mentioned bimorphism characterization [2, 13], which can easily be extended to general xmbots [13], we can prove that the preimage of every regular tree language under τ_M for every xmbot M is again regular [13].

> Pres, Pres^{-1}: *Extended multi bottom-up tree transducers do not preserve regularity, but the preimage of any regular tree language is again regular under τ_M for every xmbot M.*

In the transformation of xtt rules to xmbot rules (see Figure 10) we can see that all states of the constructed xmbot are unary (as in the xtt). However, xmbots also allow states whose rank is larger than 1, which yields some interesting properties. The most important property with respect to closure under composition is that we can reduce each xmbot to an equivalent xmbot whose rules have at most 1 input or output symbol. Such rules and xmbots of exclusively such rules are in *1-symbol normal-form*. We present an example in Figure 11. The formal construction is detailed in [13], where it is also shown that the properties ‘linear’ and ‘nondeleting’ are preserved by the construction.

Thus, instead of composing arbitrary xmbots, it is sufficient to consider compositions of xmbots in 1-symbol normal-form. The problem (misalignment) described in Section 4 can thus be avoided because each rule contains only one symbol. Consequently, it is sufficient to distinguish three simple cases in the composition construction of two xmbots M and N with rule sets P and R, respectively:

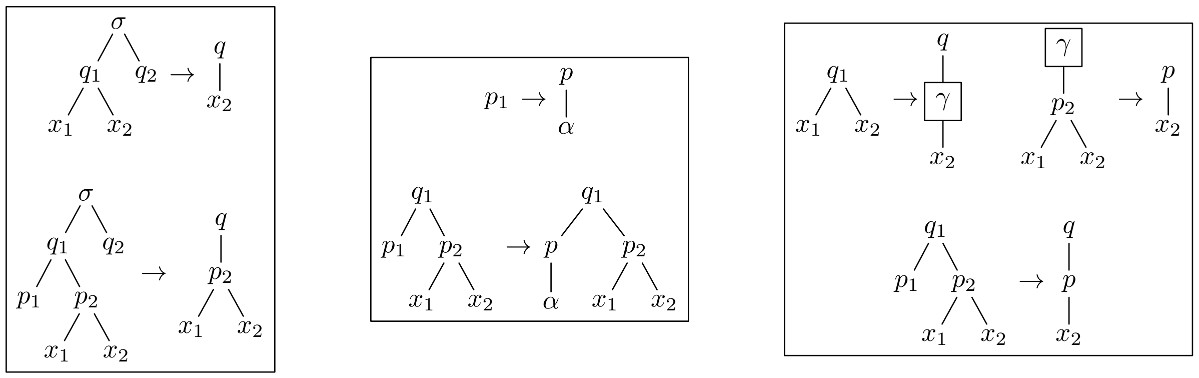
Figure 12: Composition of xmbot rules in 1-symbol normal-form (top: original rule(s); bottom: composed rule).

Model \ Criterion	Expr	Sym	Pres	Pres^{-1}	Comp
Linear nondeleting xmbot	✓	✗	✗	✓	✓
Linear xmbot	✓	✗	✗	✓	✓
General xmbot	✓	✗	✗	✓	✗

Table 3: Overview of formal properties of multi bottom-up tree transducers.

- a rule of P that contains an input symbol,
- a rule of R that contains an output symbol, and
- a rule of P that contains an output symbol together with a rule of R that consumes this symbol (i.e., that contains the same symbol in its left-hand side).

In this manner, we can easily construct the rules of a composed xmbot that computes the composition τ_M ; τ_N. Each mentioned case is illustrated in Figure 12. Again, the formal construction (and its correctness) can be found in [13]. Overall, this composition construction allows us to obtain the "classical" composition results that are known for bottom-up tree transducers [10, 4]. In particular, linear nondeleting xmbots and linear xmbots compute a class of translations that is closed under composition. Mind that xmbots are bottom-up devices and thus automatically have regular look-ahead, so those results correspond exactly to the composition results for tdtts (see Section 3). Overall, this closure is slightly surprising because even linear nondeleting xmbots are finitely copying, which yields that the simple bucket-brigade approach to composition would fail (because regularity is not preserved).

Comp: *Linear nondeleting xmbots and linear xmbots compute a class of translations that is closed under composition. General xmbots compute a class that is not closed under composition.*

As usual, we summarize the properties of the individual models of xmbots in Table 3.

Model \ Criterion	Expr	Sym	Pres	Pres^{-1}	Comp
Linear nondeleting tdtt	✗	✗	✓	✓	✓
Linear tdtt	✗	✗	✓	✓	✗
Linear tdtt with regular look-ahead	✗	✗	✓	✓	✓
General tdtt	✗	✗	✗	✓	✗
General tdtt with regular look-ahead	✓	✗	✗	✓	✗
Linear nondeleting bot	✗	✗	✓	✓	✓
Linear bot	✗	✗	✓	✓	✓
General bot	✓	✗	✗	✓	✗
Linear nondeleting xtt	✓	✓	✓	✓	✗
Linear xtt	✓	✗	✓	✓	✗
Linear xtt with regular look-ahead	✓	✗	✓	✓	✗
General xtt	✓	✗	✗	✓	✗
General xtt with regular look-ahead	✓	✗	✗	✓	✗
Linear nondeleting xbot	✓	✓	✓	✓	✗
Linear xbot	✓	✗	✓	✓	✗
General xbot	✓	✗	✗	✓	✗
Linear nondeleting xmbot	✓	✗	✗	✓	✓
Linear xmbot	✓	✗	✗	✓	✓
General xmbot	✓	✗	✗	✓	✗
Regularity-preserving linear nondeleting xmbot	✓	✗	✓	✓	✓
Regularity-preserving linear xmbot	✓	✗	✓	✓	✓
Rational tree relations	✓	✓	✓	✓	✓

Table 4: Overview of formal properties of various tree transducers.

6. Conclusion

We have presented several tree-transducer models that can easily be used in machine translation. We introduced them informally and presented their main properties as they relate to the machine-translation task at hand. Table 4 summarizes the findings. In addition to the models that we discussed in this survey, we added 'bottom-up tree transducers' (bot) and 'extended bottom-up tree transducers' (xbot) to the table for illustration. An *extended bottom-up tree transducer* (xbot) is an xmbot in which all states are unary, and a *bottom-up tree transducer* (bot) is an extended bottom-up tree transducer, in which each rule has a left-hand side in $\Sigma(Q(\mathrm{X}))$ where Σ is the ranked alphabet of input symbols and Q is the set of states. It can easily be seen that no *simple* tree transducer model is optimal in the sense that it enjoys all discussed properties.

Moreover, we added three "non-standard" models that enjoy even better properties. First, we can consider only regularity-preserving linear (linear nondeleting) xmbots. Using a decomposition of [13] and a decidability result of [16], we can decide whether a given xmbot is regularity-preserving. Since the composition of two regularity-preserving translations clearly is again regularity-preserving (using the bucket-brigade approach) and linear (linear nondeleting) xmbots can be composed, we obtain the results stated in the two penultimate lines in Table 4. The final line refers to the *rational tree relations* of [37], which are computed by "invertable regularity-preserving linear nondeleting xmbot". It is again easily checked that the composition of two invertable translations is again invertable, which yields the result shown in the last line of Table 4.

References

[1] André Arnold and Max Dauchet. Bi-transductions de forêts. In *Proc. ICALP*, pages 74–86. Edinburgh University Press, 1976.

[2] André Arnold and Max Dauchet. Morphismes et bimorphismes d'arbres. *Theoret. Comput. Sci.*, 20(4):33–93, 1982.

[3] Franz Baader and Tobias Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[4] Brenda S. Baker. Composition of top-down and bottom-up tree transductions. *Inform. and Control*, 41(2):186–213, 1979.

[5] Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

[6] Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956.

[7] Noam Chomsky. *Aspects of the Theory of Syntax.* MIT Press, 1965.

[8] Bruno Courcelle and Paul Franchi–Zannettacci. Attribute grammars and recursive program schemes. *Theoret. Comput. Sci.*, 17(1):163–191, 1982.

[9] Bruno Courcelle and Paul Franchi–Zannettacci. Attribute grammars and recursive program schemes. *Theoret. Comput. Sci.*, 17(1):235–257, 1982.

[10] Joost Engelfriet. Bottom-up and top-down tree transformations: A comparison. *Math. Systems Theory*, 9(3):198–231, 1975.

[11] Joost Engelfriet. Top-down tree transducers with regular look-ahead. *Math. Systems Theory*, 10(1):289–303, 1976.

[12] Joost Engelfriet. Three hierarchies of transducers. *Math. Systems Theory*, 15(2):95–125, 1982.

[13] Joost Engelfriet, Eric Lilin, and Andreas Maletti. Composition and decomposition of extended multi bottom-up tree transducers. *Acta Inform.*, 46(8):561–590, 2009.

[14] Joost Engelfriet and Heiko Vogler. Macro tree transducers. *J. Comput. System Sci.*, 31(1):71–146, 1985.

[15] Joost Engelfriet and Heiko Vogler. Modular tree transducers. *Theor. Comput. Sci.*, 78(2):267–303, 1991.

[16] Zoltán Ésik. Decidability results concerning tree transducers II. *Acta Cybernet.*, 6(3):303–314, 1983.

[17] Zoltán Fülöp. On attributed tree transducers. *Acta Cybernet.*, 5(1):261–279, 1981.

[18] Zoltán Fülöp, Armin Kühnemann, and Heiko Vogler. A bottom-up characterization of deterministic top-down tree transducers with regular look-ahead. *Inf. Process. Lett.*, 91(2):57–67, 2004.

[19] Zoltán Fülöp, Armin Kühnemann, and Heiko Vogler. Linear deterministic multi bottom-up tree transducers. *Theoret. Comput. Sci.*, 347(1–2):276–287, 2005.

[20] Zoltán Fülöp and Heiko Vogler. Weighted tree automata and tree transducers. In Manfred Droste, Werner Kuich, and Heiko Vogler, editors, *Handbook of Weighted Automata*, chapter 9, pages 313–403. Springer, 2009.

[21] Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. What's in a translation rule? In *Proc. HLT-NAACL*, pages 273–280. Association for Computational Linguistics, 2004.

[22] Ferenc Gécseg and Magnus Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.

[23] Ferenc Gécseg and Magnus Steinby. Tree languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 1, pages 1–68. Springer, 1997.

[24] Jonathan Graehl and Kevin Knight. Training tree transducers. In *Proc. HLT-NAACL*, pages 105–112. Association for Computational Linguistics, 2004. See also [25].

[25] Jonathan Graehl, Kevin Knight, and Jonathan May. Training tree transducers. *Computational Linguistics*, 34(3):391–427, 2008.

[26] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[27] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Processing*. Prentice-Hall, 2000.

[28] Kevin Knight. Capturing practical natural language transformations. *Machine Translation*, 21(2):121–133, 2007.

[29] Kevin Knight and Jonathan Graehl. An overview of probabilistic tree transducers for natural language processing. In *Proc. CICLing*, volume 3406 of LNCS, pages 1–24. Springer, 2005.

[30] Andreas Maletti. Compositions of extended top-down tree transducers. *Inform. and Comput.*, 206(9–10):1187–1196, 2008.

[31] Andreas Maletti. Why synchronous tree substitution grammars? In *Proc. HLT-NAACL*, pages 876–884. Association for Computational Linguistics, 2010.

[32] Andreas Maletti, Jonathan Graehl, Mark Hopkins, and Kevin Knight. The power of extended top-down tree transducers. *SIAM J. Comput.*, 39(2):410–430, 2009.

[33] Chris Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[34] Jonathan May and Kevin Knight. TIBURON: a weighted tree automata toolkit. In *Proc. CIAA*, volume 4094 of LNCS, pages 102–113. Springer, 2006.

[35] Jonathan May, Kevin Knight, and Heiko Vogler. Efficient inference through cascades of weighted tree transducers. In *Proc. ACL*, pages 1058–1066. Association for Computational Linguistics, 2010.

[36] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

[37] Jean-Claude Raoult. Rational tree relations. *Bull. Belg. Math. Soc.*, 4(1):149–176, 1997.

[38] William C. Rounds. Mappings and grammars on trees. *Math. Syst. Theory*, 4(3):257–287, 1970.

[39] Stuart M. Shieber. Synchronous grammars as tree transducers. In *Proc. TAG+7*, pages 88–95, 2004.

[40] Stuart M. Shieber. Unifying synchronous tree adjoining grammars and tree transducers via bimorphisms. In *Proc. EACL*, pages 377–384. Association for Computational Linguistics, 2006.

[41] James W. Thatcher. Generalized^2 sequential machine maps. *J. Comput. System Sci.*, 4(4):339–367, 1970.

[42] James W. Thatcher. Tree automata: An informal survey. In Alfred V. Aho, editor, *Currents in the Theory of Computing*, chapter 4, pages 143–172. Prentice Hall, 1973.

[43] Kenji Yamada and Kevin Knight. A decoder for syntax-based statistical MT. In *Proc. ACL*, pages 303–310. Association for Computational Linguistics, 2002.