Tree transformations and dependencies

Andreas Maletti[⋆]

Universität Stuttgart
Institute for Natural Language Processing
Azenbergstraße 12, 70174 Stuttgart, Germany
andreas.maletti@ims.uni-stuttgart.de

Abstract. Several tree transformation devices that are relevant in natural language processing are presented with a focus on the dependencies that they are able to capture. In many cases, the consideration of the dependencies alone can be used to provide a high-level explanation of the short-comings of tree transformation devices and allows surprising insights into their structure.

1 Motivation

In the subfield of machine translation [31], which is concerned with the automatic translation of natural language texts, it was recently realized that string-based systems [50] cannot easily compute certain important translations [48, 1, 49, 9] and that the structural information provided by modern and reliable parsers [11, 28, 27, 6] actually helps the translation process [53]. This development created renewed interest in tree automata [5, 8] and tree transducers [32, 14], which are finite-state devices that compute tree languages and tree transformations, respectively. However, there does not exist a tree translation model that is universally accepted and used in the machine translation task. On the contrary, many different models with different expressive power are used.

The first formal tree transducer model was the top-down tree transducer investigated by Thatcher [46] and Rounds [41]. Several other models such as

- bottom-up tree transducers [47],
- attributed tree transducers [20, 30] and pebble tree transducers [38],
- macro tree transducers [12, 18] and modular tree transducers [19],
- monadic second-order logic tree transducers [16, 7], and
- tree bimorphisms [3] and various models with synchronization [40]

were introduced later and have been investigated in the theory of formal languages. In general, tree transducers process an input tree and nondeterministically generate an output tree. In the process, they can move complete subtrees or decide to process subtrees differently based on an internal state. Shieber [42] and others have argued that top-down tree transducers are generally inadequate for linguistic tasks. In this survey we will focus on three models that received attention from the machine translation community, which are:

[⋆] Supported by the German Research Foundation (DFG) grant MA/4959/1-1.

– extended top-down tree transducers [13, 2, 29, 26],
– extended multi bottom-up tree transducers [33, 3, 21, 22, 15, 35], and
– synchronous tree-sequence substitution grammars [40, 51, 52, 45].

We review these three models and investigate their expressive power from a very abstract viewpoint by looking only at the type of dependencies that they create. It turns out that the models, which indeed have successively more expressive power, can already be distinguished easily based on the types of dependencies that they can compute. Informally, a dependency records that a certain part of the output tree was created in accordance with a particular part of the input tree (or vice versa). This influence is informally called dependence, synchronization, or contribution [17]. Formally, we establish dependencies using the derivation mechanism, which is typically term rewriting [4]. However, for our purposes 'synchronous substitution' is much more suitable since the synchronization links are an explicit representation of our dependencies. Thus, we adjust the derivation process to use synchronous substitution keeping all synchronization links during the whole derivation.

We investigate the type of dependencies each of our three mentioned tree transformation models can compute. It shows that all of them enjoy a certain hierarchy property that can be used to quickly show that transformations that have crossing dependencies cannot be computed by any of our models. This allows us, for example, to set all three models apart from synchronous tree-adjoining grammars [44, 42, 43], which can compute crossing dependencies. Moreover, a stricter version of the hierarchy property also allows us to distinguish our three models, which we demonstrate with an example transformation in each case. These example transformations are taken from the literature, but instead of presenting the full proof from the literature, we simply "add" natural dependencies to the transformation and then show that no dependency computable by a certain model is compatible with these dependencies. This approach highlights the essential and illustrative part of the formal proof and avoids the technical part of the proof, which is still needed to justify the initial dependencies. The interested reader can find these technical parts in the cited literature or can prove them via a case analysis. Thus, the approach pursued here does not offer full proofs, but it will be obvious from the examples that those dependencies should be present.

The survey is structured as follows: Section 2 recalls basic notions and notation. In each of the next three sections (Sections 3–5) we recall one of our three tree transformation models in order of increasing expressive power (i.e., the order in which they are mentioned above). Section 3 also contains the definitions of the hierarchy properties of dependencies that we will investigate. We conclude with a short summary, which is presented as a table showing the identified properties of dependencies (including those for synchronous tree-adjoining grammars).

2 Notation

The set of all nonnegative integers is $\mathbb{N}$. A *relation ρ from a set S to a set T* is a subset $\rho \subseteq S \times T$. The set of all finite words over S is S^*, where ε is the empty word. The concatenation of the words $v, w \in S^*$ is $v.w$ or simply vw. The length of a word $w \in S^*$ is denoted by $|w|$. An *alphabet* Σ is a nonempty and finite set, of which the elements are called *symbols*. A *ranked alphabet* is a pair (Σ, rk) consisting of an alphabet Σ and a rank mapping $\mathrm{rk}\colon \Sigma \to \mathbb{N}$. For every $k \in \mathbb{N}$, let $\Sigma_k = \mathrm{rk}^{-1}(k)$. We typically write $\sigma^{(k)}$ to indicate that $\mathrm{rk}(\sigma) = k$. A *doubly ranked alphabet* simply has a rank mapping $\mathrm{rk}\colon \Sigma \to \mathbb{N}^2$. We use the same notations for ranked and doubly ranked alphabets. Moreover, we typically assume that the mapping rk is clear from the context.

The set $T_\Sigma(S)$ of *Σ-trees with leaf labels S* is the smallest set T such that $S \subseteq T$ and $\sigma(t_1, \dots, t_k) \in T$ for every $\sigma \in \Sigma_k$ and $t_1, \dots, t_k \in T$. We generally assume that $\Sigma \cap S = \emptyset$, and thus we write $\epsilon()$ simply as ϵ for every $\epsilon \in \Sigma_0$. Moreover, we write $\alpha^k(t)$ for $\alpha(\cdots \alpha(t) \cdots)$ containing k occurrences of α in the abbreviated list. We write T_Σ for $T_\Sigma(\emptyset)$. The set $\mathrm{pos}(t) \subseteq \mathbb{N}^*$ of *positions* of $t \in T_\Sigma(S)$ is inductively defined by $\mathrm{pos}(s) = \{\varepsilon\}$ for every $s \in S$ and

$$\mathrm{pos}(\sigma(t_1, \dots, t_k)) = \{\varepsilon\} \cup \bigcup_{i=1}^{k} \{iw \mid w \in \mathrm{pos}(t_i)\}$$

for every $\sigma \in \Sigma_k$ and $t_1, \dots, t_k \in T_\Sigma(S)$. The positions $\mathrm{pos}(t)$ are totally ordered by the lexicographic order $\sqsubseteq$ on $\mathbb{N}^*$ and the prefix order $\leq$ on $\mathbb{N}^*$. Let $t, t' \in T_\Sigma(S)$ and $w \in \mathrm{pos}(t)$. The *label* of t at w is $t(w)$, and the *w-rooted subtree* of t is $t|_w$. Formally, $s(\varepsilon) = s|_\varepsilon = s$ for every $s \in S$ and

$$t(w) = \begin{cases} \sigma & \text{if } w = \varepsilon \\ t_i(v) & \text{if } w = iv \text{ and } i \in \mathbb{N} \end{cases} \qquad \text{and} \qquad t|_w = \begin{cases} t & \text{if } w = \varepsilon \\ t_i|_v & \text{if } w = iv \text{ and } i \in \mathbb{N} \end{cases}$$

where $t = \sigma(t_1, \dots, t_k)$ for every $\sigma \in \Sigma_k$ and $t_1, \dots, t_k \in T_\Sigma(S)$. For every $L \subseteq S$, we let $\mathrm{pos}_L(t) = \{w \in \mathrm{pos}(t) \mid t(w) \in L\}$ and $\mathrm{pos}_s(t) = \mathrm{pos}_{\{s\}}(t)$ for every $s \in S$. The tree t is *linear* in L if $|\mathrm{pos}_l(t)| \leq 1$ for every $l \in L$. Moreover, $\mathrm{var}(t) = \{s \in S \mid \mathrm{pos}_s(t) \neq \emptyset\}$. The expression $t[u]_w$ denotes the tree that is obtained from $t \in T_\Sigma(S)$ by replacing the subtree $t|_w$ at w by $u \in T_\Sigma(S)$. We extend this notation to sequences $\boldsymbol{u} = u_1, \dots, u_n$ of trees and positions $\boldsymbol{w} = w_1, \dots, w_n$ of t that are pairwise incomparable with respect to the prefix order. Thus, $t[\boldsymbol{u}]_{\boldsymbol{w}}$ denotes the tree obtained from t by replacing the subtree $t|_{w_i}$ at w_i by u_i for all $1 \leq i \leq n$.

3 Extended top-down tree transducer

Our first model is the (linear and nondeleting) extended top-down tree transducer [13, 2, 29, 26] (XTOP), which is based on the classical top-down tree transducer [41, 46]. A top-down tree transducer is a special XTOP, in which all left-hand sides of rules contain exactly one input symbol. In general, the left-hand side of an XTOP can contain any number of input symbols [34, 37].

We present a syntactic version here that is closer to synchronized grammars [10], but equally expressive as the classical version [41, 46, 29, 26] with term rewrite rules. In general, equal states in the left and right-hand side of a rule are linked. In a derivation, they will be replaced at the same time. In a rule of an XTOP, these links are bijective.

Definition 1 (see [37, Sect. 2.2]). *A (linear and nondeleting)* extended tree transducer *(*XTOP*) is a tuple* $(Q, \Sigma, \Delta, I, R)$*, where*

- Q *is a finite set of* states*,*
- Σ *and* Δ *are ranked alphabets of* input *and* output symbols*,*
- $I \subseteq Q$ *is a set of* initial states*, and*
- $R \subseteq T_\Sigma(Q) \times Q \times T_\Delta(Q)$ *is a finite set of* rules *such that* l *and* r *are linear in* Q *and* $\mathrm{var}(l) = \mathrm{var}(r)$ *for every* $(l, q, r) \in R$*.*

In the following, let $M = (Q, \Sigma, \Delta, I, R)$ be an XTOP. As already mentioned, M is a *top-down tree transducer* if for every $(l, q, r) \in R$ there exist $\sigma \in \Sigma_k$ and $q_1, \dots, q_k \in Q$ such that $l = \sigma(q_1, \dots, q_k)$. To simplify the notation, we often write rules as $l \overset{q}{—} r$ instead of (l, q, r).

Example 2 (see [3, Sect. 3.4]). Let

$$M_{\mathrm{bin}} = (Q, \Sigma, \Sigma, \{\star\}, R) \qquad \text{and} \qquad M_{\mathrm{debin}} = (Q, \Sigma, \Sigma, \{\star\}, R')$$

be the XTOP with

- $Q = \{\star, p, q, r\}$,
- $\Sigma = \{\sigma^{(3)}, \delta^{(2)}, \alpha^{(1)}, \epsilon^{(0)}\}$,
- R, which contains the following rules for all $x \in \{q, p, r\}$:

$$\delta(p, \star) \overset{\star}{—} \delta(p, \star) \qquad \sigma(p, q, \star) \overset{\star}{—} \delta(p, \delta(q, \star)) \qquad \sigma(p, q, r) \overset{\star}{—} \delta(p, \delta(q, r))$$
$$\alpha(x) \overset{x}{—} \alpha(x) \qquad \epsilon \overset{x}{—} \epsilon \enspace ,$$

- and R', which contains the following rules for all $x \in \{q, p, r\}$:

$$\delta(p, \epsilon) \overset{\star}{—} \delta(p, \epsilon) \qquad \delta(p, \delta(q, \star)) \overset{\star}{—} \sigma(q, p, \star) \qquad \delta(p, \alpha(q)) \overset{\star}{—} \delta(p, \alpha(q))$$
$$\alpha(x) \overset{x}{—} \alpha(x) \qquad \epsilon \overset{x}{—} \epsilon \enspace .$$

Clearly, M_{bin} is even a top-down tree transducer, whereas M_{debin} is not a top-down tree transducer. The rules of M_{bin} are illustrated in Fig. 1.

Next, we move to the semantics of an XTOP M, which is given by synchronous substitution. While the links in an XTOP rule are implicit and established due to occurrences of equal states, we need an explicit linking structure for our sentential forms. In addition, these links will form the dependencies that we are interested in. To this end, we store a relation between positions of the input and output tree, which encodes the links. Let $\mathcal{L} = \mathcal{P}(\mathbb{N}^* \times \mathbb{N}^*) = \{S \mid S \subseteq \mathbb{N}^* \times \mathbb{N}^*\}$ be the set of all link structures. First, we define general sentential forms. Roughly speaking, we have an input tree and an output tree, in which positions are linked.

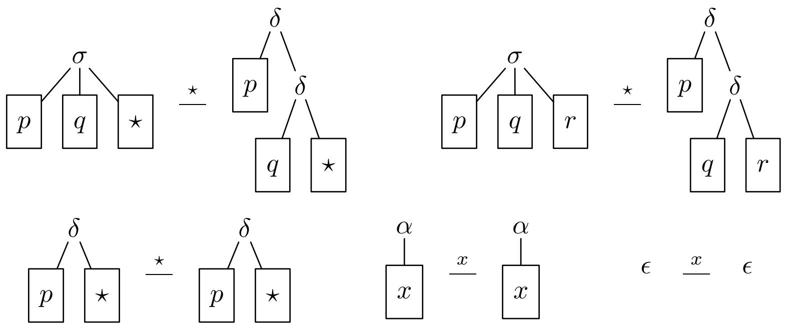

Fig. 1. Example rules of the XTOP M_{bin} of Ex. 2.

Definition 3 (see [23, Sect. 3]). *An element* $\langle \xi, D, \zeta \rangle \in T_\Sigma(Q) \times \mathcal{L} \times T_\Delta(Q)$ *is a* sentential form *if* $v \in \text{pos}(\xi)$ *and* $w \in \text{pos}(\zeta)$ *for every* $(v, w) \in D$.

Now we lift the implicit link structure in an XTOP rule into an explicit link relation. This link relation will then be used in the derivation process once the rule is applied to determine the next links in the obtained sentential form.

Definition 4. *Let* $l \overset{q}{—} r \in R$ *be a rule, and let* $v, w \in \mathbb{N}^*$*. The rule's link structure* $\text{links}_{v,w}(l \overset{q}{—} r) \in \mathcal{L}$ *is*

$$\text{links}_{v,w}(l \overset{q}{—} r) = \bigcup_{p \in Q} \{(vv', ww') \mid v' \in \text{pos}_p(l), w' \in \text{pos}_p(r)\} \enspace .$$

Note that $\text{links}_{v,w}(l \overset{q}{—} r)$ is a bijective relation on the state occurrences. The derivation process is started with a simple sentential form $\langle q, \{(\varepsilon, \varepsilon)\}, q\rangle$ consisting of the input tree q and the output tree q for some initial state $q \in I$ and the trivial link relating both states. This is clearly a link structure that is bijective between state occurrences. Next, we (nondeterministically) apply a rule $l \overset{q}{—} r$ to a pair of linked occurrences of the state q. Such an application replaces the linked occurrences of q by the left and right-hand side of the rule. The implicit links in the rule are added to the (explicit) link structure to obtain a new sentential form. This yields another link structure that is bijective between state occurrences. Since we are interested in the dependencies created during derivation, we preserve all links and never remove a link from the linking structure. Note that this preservation causes that the link structure need not be functional on all positions because we keep the links that were used in the replacement process. This replacement process is repeated until no linked occurrences of states remain.

Definition 5 (see [23, Sect. 3]). *Given two sentential forms* $\langle \xi, D, \zeta \rangle$ *and* $\langle \xi', D', \zeta' \rangle$ *such that* D *and* D' *are bijective on state occurrences, we write*

$$\langle \xi, D, \zeta \rangle \Rightarrow_M \langle \xi', D', \zeta' \rangle$$

if there exists a rule $l \overset{q}{—} r \in R$ *and an input position* $v \in \mathrm{pos}_q(\xi)$ *such that*

- $\xi' = \xi[l]_v$ *and* $\zeta' = \zeta[r]_w$*, where* $w \in \mathrm{pos}_q(\zeta)$ *is the unique* q*-labelled position such that* $(v, w) \in D$*, and*
- $D' = D \cup \mathrm{links}_{v,w}(l \overset{q}{—} r)$.

As usual $\Rightarrow_M^*$ *is the reflexive and transitive closure of* $\Rightarrow_M$*. The* XTOP M *computes the dependencies* $\mathrm{dep}(M) \subseteq T_\Sigma \times \mathcal{L} \times T_\Delta$*, which are given by*

$$\mathrm{dep}(M) = \{\langle t, D, u\rangle \in T_\Sigma \times \mathcal{L} \times T_\Delta \mid \exists q \in I \colon \langle q, \{(\varepsilon, \varepsilon)\}, q\rangle \Rightarrow_M^* \langle t, D, u\rangle\} \ .$$

Moreover, the XTOP M *computes the tree transformation* $M \subseteq T_\Sigma \times T_\Delta$*, which is given by* $M = \{(t, u) \mid (t, D, u) \in \mathrm{dep}(M)\}$.

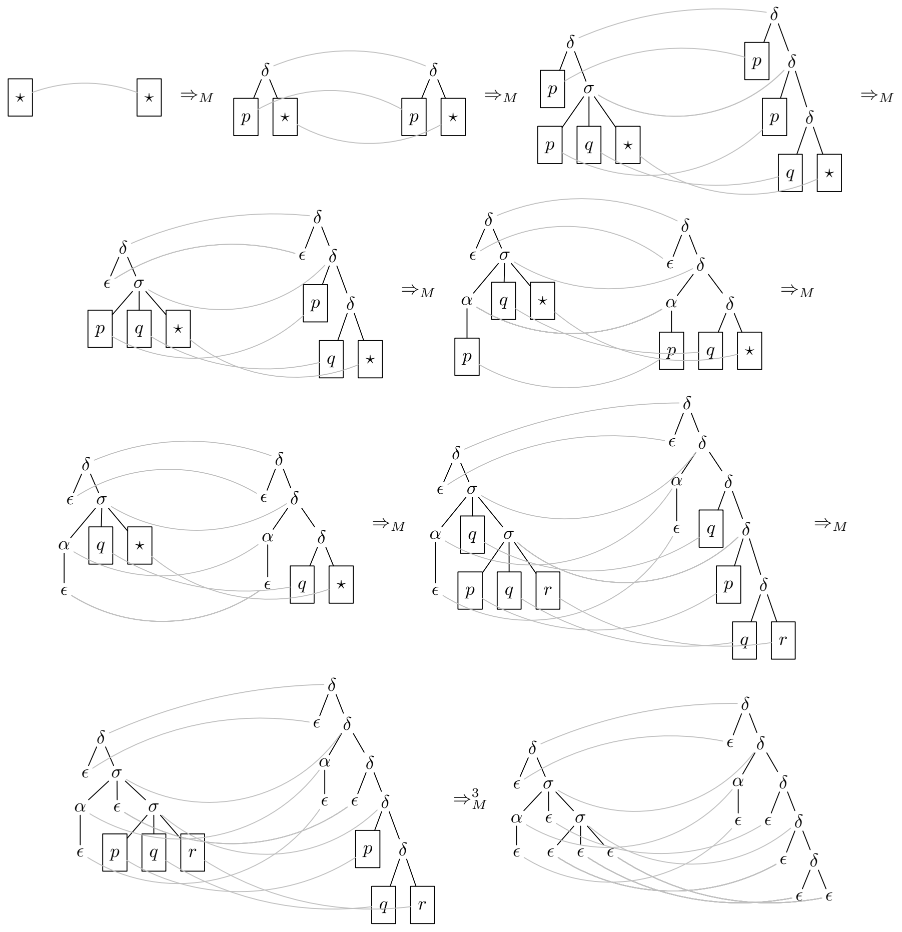

Fig. 2. Example derivation where $M = M_{\mathrm{bin}}$ (see Ex. 6).

Example 6. Let $M = M_{\text{bin}}$ be the XTOP of Ex. 2, and let $\delta(\epsilon, \sigma(\alpha(\epsilon), \epsilon, \sigma(\epsilon, \epsilon, \epsilon)))$ be the input tree. Selecting the only initial state $\star$, we can obtain the derivation that is displayed in Fig. 2. Overall $\delta(\epsilon, \delta(\alpha(\epsilon), \delta(\epsilon, \delta(\epsilon, \delta(\epsilon, \epsilon)))))$ is a translation of the input tree. The translations of M_{bin} and M_{debin} of Ex. 2 are illustrated in Fig. 3.

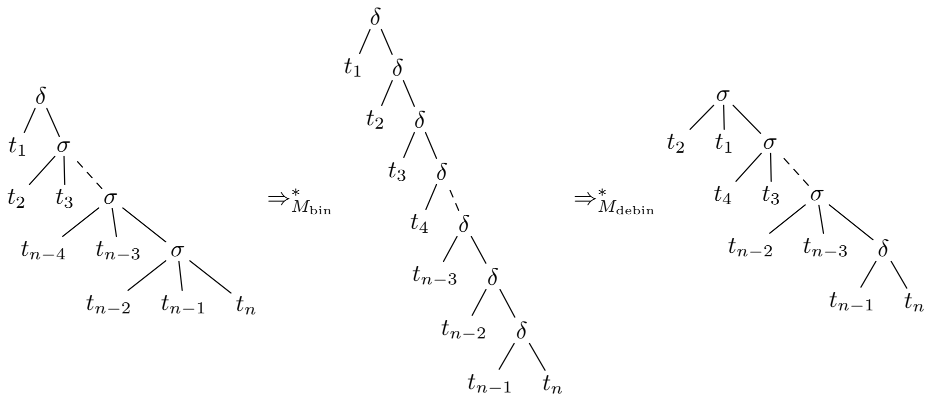

Fig. 3. Translations of [3] that are individually computed by the XTOP M_{bin} and M_{debin}.

Since every translation $(t, u) \in M$ is ultimately created by (at least) one successful derivation, we can inspect the links in the derivation process to exhibit the dependencies. Roughly speaking, the links establish which parts of the output tree were generated due to a particular part of the input tree. This correspondence is called *contribution* in [17].

Example 7. Recall the XTOP M_{bin} of Ex. 2 and the derivation in Ex. 6, which is displayed in Fig. 2. Looking at the last sentential form, which is displayed in Fig. 4 for easier reference, its linking structure is

$$\begin{aligned} D = \{&(\varepsilon, \varepsilon), (1, 1), (2, 2), (21, 21), (211, 211), (22, 221), (23, 222), \\ &(231, 2221), (232, 22221), (233, 22222)\} \enspace , \end{aligned}$$

which represents the dependencies introduced by the rule applications.

Next, let us observe some important properties of the computed dependencies. To this end, we disregard the actual input and output trees and say that a linking structure $D \in \mathcal{L}$ is computed by M if there exist an initial state $q \in I$ and trees $t \in T_\Sigma$ and $u \in T_\Delta$ such that $\langle q, \{(\varepsilon, \varepsilon)\}, q\rangle \Rightarrow^*_M \langle t, D, u\rangle$. The set of all linking structures computed by M is $\text{links}(M)$.

Definition 8. *A linking structure $D \in \mathcal{L}$ is* input hierarchical *if for every $(v_1, w_1), (v_2, w_2) \in D$ with $v_1 < v_2$ we have $w_2 \not< w_1$ and there exists $(v_1, w_1') \in D$ such that $w_1' \leq w_2$. It is* strictly input hierarchical *if additionally*

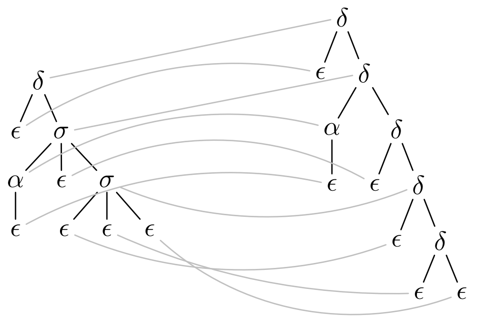

Fig. 4. Dependencies computed during the derivation of Fig. 2.

– *$w \leq w'$ or $w' \leq w$ for all $(v, w), (v, w') \in D$ and*
– *$v_1 \not< v_2$ for all $(v_1, w_1), (v_2, w_2) \in D$ with $w_1 \not\leq w_2$.*

Roughly speaking, input hierarchical linking structures have no crossing links (or dependencies). More formally, let $(v, w), (v', w') \in D$ be such that $v < v'$ and $w' < w$. Then (v, w) and (v', w') are *crossing links* (or *dependencies*). Clearly, such links cannot exist in an input hierarchical linking structure. The same notions can be defined for the output side by requiring the corresponding properties for the linking structure D^{-1}. For example, D is *strictly output hierarchical* if D^{-1} is strictly input hierarchical. Moreover, it is *strictly hierarchical* if it is both strictly input hierarchical and strictly output hierarchical. Finally, a set $\mathcal{D} \subseteq \mathcal{L}$ of linking structures has a certain hierarchical property if each element has it.

Example 9. The linking structure D of Ex. 7 is strictly hierarchical.

In addition, we also need a property that guarantees that there are enough links. Roughly speaking, there should be an integer that limits the distance between links.

Definition 10. *A set $\mathcal{D} \subseteq \mathcal{L}$ of link structures has* bounded distance *if there exists an integer $k \in \mathbb{N}$ such that for every $D \in \mathcal{D}$ we have that*

– *for all $(v, w), (vv', w') \in D$ with $|v'| > k$ there exist $v_1, v_2 \leq v'$ and w_1, w_2 such that $|v_1| \leq k \geq |v'| - |v_2|$ and $(vv_1, w_1), (vv_2, w_2) \in D$, and*
– *for all $(v, w), (v', ww') \in D$ with $|w'| > k$ there exist v_1, v_2 and $w_1, w_2 \leq w'$ such that $|w_1| \leq k \geq |w'| - |w_2|$ and $(v_1, ww_1), (v_2, ww_2) \in D$.*

In other words, between any two source- or target-nested links of large distance, there should exist links whose distance to the original links is small. This yields that the distance to the next nested link (if such a link does exist) can be at most k. Note however, that the above property does not require a link every k symbols. This property would also be true for all XTOP, but it would no longer be true for all MBOT, which are discussed in the next section. To keep the presentation simple, we only discuss 'bounded distance' as introduced.

Example 11. The set links(M_{bin}), where M_{bin} is the XTOP of Ex. 2, has bounded distance. For the input side, the distance is bounded by 1, and for the output side, it is bounded by 2.

Lemma 12. *The set* links(M) *computed by an* XTOP M *is strictly hierarchical with bounded distance.*

Proof. This lemma follows trivially from Definition 5.

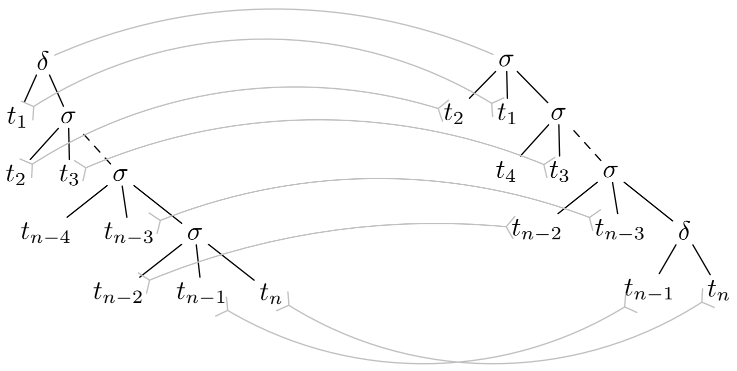

Fig. 5. Example translation of [3] with dependencies, where the inverse arrow heads indicate that the dependencies point to any node (not necessarily the root) inside the subtrees.

In addition, for a (linear and nondeleting) top-down tree transducer every input position has exactly one link (i.e., the linking structures encountered with top-down tree transducers are functional). Next, we define the notion of compatibility of linking structures (or dependencies). This notion will allow us to prescribe a semantic dependency and then analyze whether a certain class of tree transformation devices can handle such linking structures. Naturally, the implementation in a tree transformation device can add more dependencies, which are created by the particular choice of rules. Consequently, compatibility only requires that the given dependencies are a subset of the realized links in the linking structure. Moreover, given a set of dependencies for a given input and output tree, it is sufficient to be compatible to at least one dependency because already one compatible dependency would render the translation plausible.

Definition 13. *Let* $\langle \xi, D, \zeta \rangle$ *and* $\langle \xi, D', \zeta \rangle$ *be sentential forms with the same input and output trees. Then* $\langle \xi, D', \zeta \rangle$ *is* compatible *with* $\langle \xi, D, \zeta \rangle$ *if* $D \subseteq D'$*. Given sets* L *and* L' *of sentential forms,* L' *is* compatible *with* L *if for every* $\langle \xi, D, \zeta \rangle \in L$ *there exist* $\langle \xi, D', \zeta \rangle \in L'$ *and* $\langle \xi, D'', \zeta \rangle \in L$ *such that* $\langle \xi, D, \zeta \rangle$ *is compatible with* $\langle \xi, D'', \zeta \rangle$*.*

Figure 5 shows the composition of the tree transformations that are computed by the XTOP M_{bin} and M_{debin} of Ex. 2. This example was used in [3] to show that the class of transformations computed by XTOP is not closed under composition. In fact, assuming the dependencies indicated in Fig. 5, we can prove this statement by observing that this set of dependencies is not compatible to a strictly hierarchical dependence with bounded distance. Consequently, these dependencies cannot be computed by an XTOP.

Lemma 14. *The dependencies depicted in Fig. 5 are not compatible with the dependencies computed by any* XTOP*.*

Proof. Suppose that there is an XTOP that computes dependencies that are compatible with the dependencies depicted in Fig. 5. Then there exits a bound n such that all input and output tree pairs whose σ-spine is longer than n must have a link on this σ-spine. However, such a link together with the existing dependencies makes it incompatible to any strictly hierarchical dependency. □

The previous lemma also yields that the tree transformation of Fig. 5 cannot be computed by an XTOP. Actually, the difficult, but not very illustrative part of the full proof establishes that the dependencies depicted in Fig. 5 are really necessary. This part remains and is proved in [3].

Theorem 15 (see [3, Sect. 3.4]). *The tree transformation illustrated in Fig. 5 cannot be computed by any* XTOP*.*

4 Extended multi bottom-up tree transducer

In this section, we recall the (linear and nondeleting) extended multi bottom-up tree transducer (MBOT), which was introduced in [33, 3] in the shape of a particular bimorphism. The name "multi bottom-up tree transducer" seems to originate from [21, 22], where the deterministic variant of the model was rediscovered. A more detailed presentation of various multi bottom-up tree transducers can be found in [15], and [35] reports some results for the weighted model.

Definition 16 (see [35, Def. 2]). *A (linear and nondeleting)* extended multi bottom-up tree transducer *(*MBOT*) is a system $(Q, \Sigma, \Delta, I, R)$ where*

- *Q, Σ, and Δ are ranked alphabets of* states*,* input symbols*, and* output symbols*, respectively,*
- *$I \subseteq Q_1$ is a subset of initial states, all of which are unary, and*
- *$R \subseteq T_\Sigma(Q) \times Q \times T_\Delta(Q)^*$ is a finite set of rules such that l is linear in Q, $\text{rk}(q) = n$, and $\bigcup_{i=1}^{n} \text{var}(r_i) \subseteq \text{var}(l)$ for every $(l, q, r_1 \cdots r_n) \in R$.*

For all the remaining discussions, let $M = (Q, \Sigma, \Delta, I, R)$ be an MBOT. Clearly, any XTOP is an MBOT. In addition, two items deserve explicit mention. First, the set Q of states is a ranked alphabet in contrast to XTOP or traditional top-down or bottom-up tree transducers [46, 41, 47]. Roughly speaking, the rank $\text{rk}(q)$ of a state $q \in Q$ coincides with the number $|\boldsymbol{r}|$ of trees in

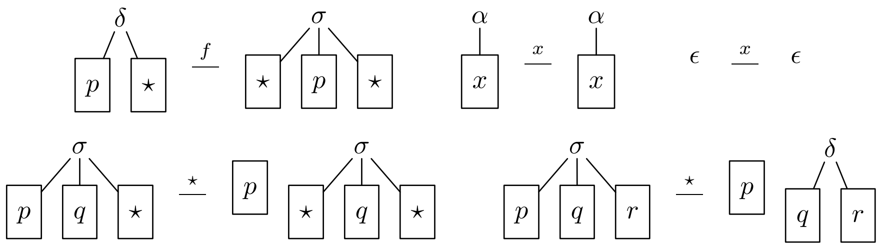

Fig. 6. MBOT rules of the MBOT M_{comp} of Ex. 17.

the right-hand side of all rules $(l, q, \boldsymbol{r}) \in R$. For example, a nullary state has no output trees at all and can be understood as a pure look-ahead [15] in the input tree. Second, all initial states are unary (i.e., have exactly one output tree). In this way, we obtain exactly one output tree and ultimately a relation between input and output trees. To simplify the discussion, we call l and $\boldsymbol{r}$ of a rule $(l, q, \boldsymbol{r}) \in R$ the left- and right-hand side, respectively. In accordance, we sometimes write $l \overset{q}{—} \boldsymbol{r}$ instead of $(l, q, \boldsymbol{r})$.

Example 17 (see [3, Sect. 3.4]). Let $M_{\text{comp}} = (Q, \Sigma, \Sigma, \{f\}, R)$ be the MBOT with

– $Q = \{\star^{(2)}, p^{(1)}, q^{(1)}, r^{(1)}, f^{(1)}\}$,
– $\Sigma = \{\sigma^{(3)}, \delta^{(2)}, \alpha^{(1)}, \epsilon^{(0)}\}$, and
– R, which contains the following rules for every $x \in \{p, q, r\}$:

$$\delta(p, \star) \overset{f}{—} \sigma(\star, p, \star) \qquad \sigma(p, q, \star) \overset{\star}{—} p \;.\; \sigma(\star, q, \star) \qquad \sigma(p, q, r) \overset{\star}{—} p \;.\; \delta(q, r)$$
$$\alpha(x) \overset{x}{—} \alpha(x) \qquad\qquad \epsilon \overset{x}{—} \epsilon \;\;,$$

where we separate trees in a sequence by full stops.

The rules of M_{comp} are illustrated in Fig. 6.

Since our rules now have a more general structure, we again need to lift the implicit link structure in an MBOT rule into an explicit link relation. This time we provide an input position and additionally as many output positions as required. The required number is the rank of the state q in a rule $l \overset{q}{—} \boldsymbol{r}$.

Definition 18. *Let $l \overset{q}{—} \boldsymbol{r} \in R$ be a rule. Moreover, let $v, w_1, \ldots, w_n \in \mathbb{N}^*$ and $\boldsymbol{w} = w_1 \cdots w_n$ where $n = \text{rk}(q)$. The rule's link structure* $\text{links}_{v,\boldsymbol{w}}(l \overset{q}{—} \boldsymbol{r}) \in \mathcal{L}$ *is*

$$\text{links}_{v,\boldsymbol{w}}(l \overset{q}{—} \boldsymbol{r}) = \bigcup_{p \in Q} \bigcup_{i=1}^{n} \{(vv', w_i w_i') \mid v' \in \text{pos}_p(l), w_i' \in \text{pos}_p(r_i)\} \;\;.$$

Note that the inverse relation $\text{links}_{v,\boldsymbol{w}}(l \overset{q}{—} \boldsymbol{r})^{-1}$ is functional on the state occurrences. However, in general, it is not bijective, and in particular, a state occurrence in the input might be without any link.

The semantics is again presented using synchronous substitution. However, this time several states in the output side of a sentential form can be linked to the state that is replaced in the input side of the sentential side. As before, the derivation process is started with a simple sentential form $\langle q, \{(\varepsilon, \varepsilon)\}, q\rangle$. Next, we (nondeterministically) apply a rule $l \overset{q}{\text{---}} \boldsymbol{r}$ to an occurrence of a state in the input side and all its linked occurrences on the output side. Those occurrences are replaced by the left and right-hand side of the rule, where the potentially several trees in the right-hand side replace the linked occurrences in lexicographic order. The final step adds the implicit links in the rule to the (explicit) link structure to obtain a new sentential form. Note that the functionality of the inverse implicit linking structure of a rule is lost in the sentential forms due to the preservation of old links.

Definition 19 (see [36, Sect. 3]). *Given two sentential forms $\langle \xi, D, \zeta\rangle$ and $\langle \xi', D', \zeta'\rangle$, we write $\langle \xi, D, \zeta\rangle \Rightarrow_M \langle \xi', D', \zeta'\rangle$ if there exists a rule $l \overset{q}{\text{---}} \boldsymbol{r} \in R$ and an input position $v \in \mathrm{pos}_q(\xi)$ such that*

- $\mathrm{rk}(q) = n$,
- $\xi' = \xi[l]_v$ *and* $\zeta' = \zeta[\boldsymbol{r}]_{\boldsymbol{w}}$, *where* $\boldsymbol{w} = w_1 \cdots w_n$ *with (i)* $w_1, \ldots, w_n \in \mathrm{pos}_q(\zeta)$, *(ii)* $w_1 \sqsubset \cdots \sqsubset w_n$, *and (iii)* $\{w_1, \ldots, w_n\} = \{w \mid (v, w) \in D\}$, *and*
- $D' = D \cup \mathrm{links}_{v,\boldsymbol{w}}(l \overset{q}{\text{---}} \boldsymbol{r})$.

As usual $\Rightarrow_M^*$ is the reflexive and transitive closure of $\Rightarrow_M$. The MBOT *M computes the dependencies* $\mathrm{dep}(M) \subseteq T_\Sigma \times \mathcal{L} \times T_\Delta$, *which are given by*

$$\mathrm{dep}(M) = \{\langle t, D, u\rangle \in T_\Sigma \times \mathcal{L} \times T_\Delta \mid \exists q \in I \colon \langle q, \{(\varepsilon, \varepsilon)\}, q\rangle \Rightarrow_M^* \langle t, D, u\rangle\} \ .$$

Moreover, the MBOT *M computes the relation $M \subseteq T_\Sigma \times T_\Delta$, which is given by $M = \{(t, u) \mid (t, D, u) \in \mathrm{dep}(M)\}$, and* $\mathrm{links}(M) = \{D \mid (t, D, u) \in \mathrm{dep}(M)\}$.

Example 20. It can easily be verified that M_{comp} of Ex. 17 computes the tree transformation depicted in Fig. 5. An example derivation using M_{comp} is shown in Fig. 7.

Consequently, the tree transformation used in the previous section (see Fig. 5) can be computed by an MBOT. Figure 9 roughly sketches the dependencies created during the computation of this transformation with the MBOT M_{comp} of Ex. 17. Next, let us look at the properties of the dependencies represented in Figs. 8 and 9.

Example 21. Figure 8 represents the sentential form $\langle t, D, u\rangle$, where

$$\begin{aligned}
t &= \delta(\epsilon, \sigma(\alpha(\epsilon), \epsilon, \sigma(\epsilon, \epsilon, \epsilon)))\\
u &= \sigma(\alpha(\epsilon), \epsilon, \sigma(\epsilon, \epsilon, \delta(\epsilon, \epsilon)))\\
D &= \{(\varepsilon, \varepsilon), (1, 2), (2, 1), (2, 3), (21, 1), (211, 11), (22, 32), (23, 31), (23, 33),\\
&\qquad (231, 31), (232, 331), (233, 332)\} \ .
\end{aligned}$$

The linking structure D is input hierarchical and strictly output hierarchical. The same properties also hold for the dependencies indicated in Fig. 9.

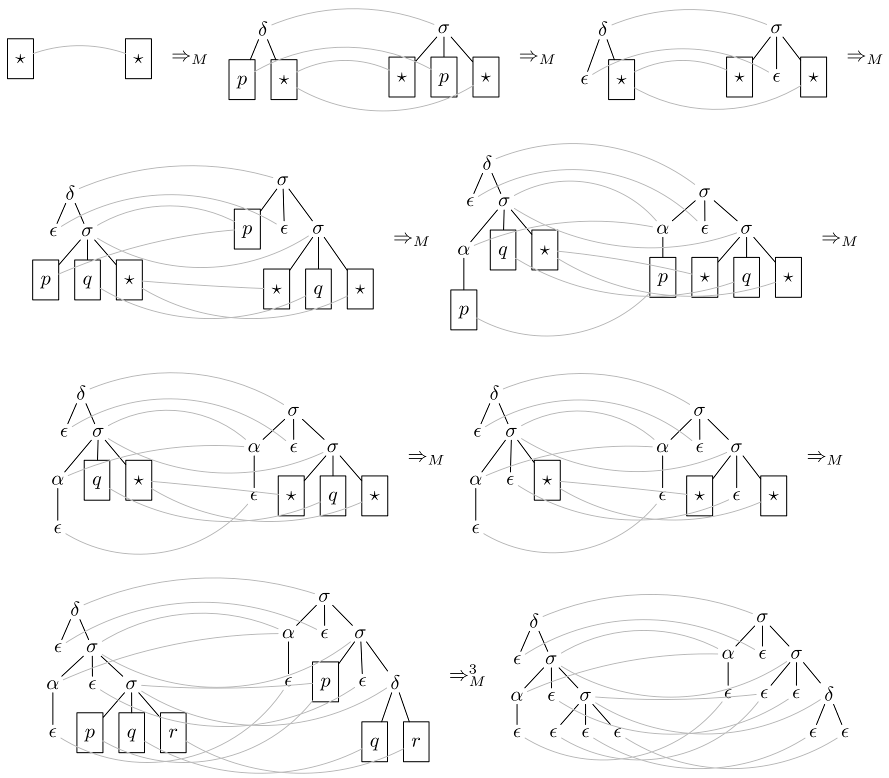
Fig. 7. Example derivation where $M = M_{\mathrm{comp}}$ (see Ex. 17).

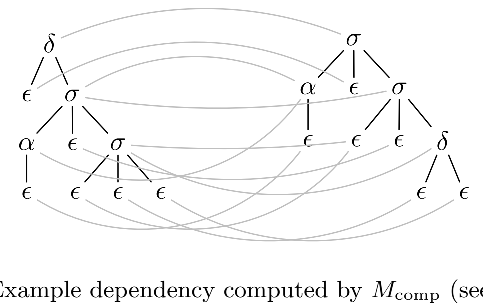
Fig. 8. Example dependency computed by M_{comp} (see Ex. 17).

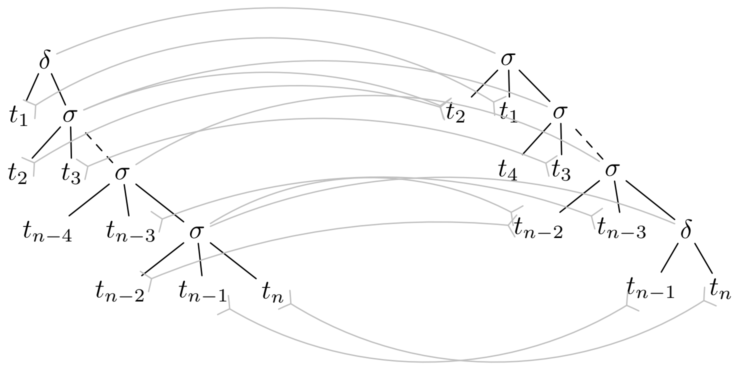

Fig. 9. Example translation of [3] with dependencies suitable for an MBOT.

The properties of the dependencies exhibited in Ex. 21 are indicative for all dependencies computed by MBOT. This is observed in the next lemma, which follows straightforwardly from Def. 19. As usual, the finite size of the rules yields bounded distance. Note that there can be unboundedly large parts of the input tree without any link. For example, input subtrees created by nullary states can have this property because the MBOT does only check a regular property [24, 25] and does not produce any corresponding output.

Lemma 22. *The set* links(M) *computed by an* MBOT M *is input hierarchical and strictly output hierarchical with bounded distance.*

Next, we again use our knowledge about the type of dependencies that are computable by an MBOT to illustrate a tree transformation that cannot be computed by any MBOT.

Example 23 (see [39, Ex. 4.5] and [40]). The MBOT $M_{\text{sort}} = (Q, \Sigma, \Delta, \{f\}, R)$ and $M_{\text{sort2}} = (Q, \Sigma, \Delta, \{f\}, R')$ are given by

- $Q = \{p^{(3)}, q^{(3)}, r^{(3)}, f^{(1)}\}$,
- $\Sigma = \{\epsilon^{(0)}, \alpha^{(1)}, \beta^{(1)}, \gamma^{(1)}\}$ and $\Delta = \Sigma \cup \{\sigma^{(3)}\}$,
- the following rules in R:

$$\alpha(p) \overset{p}{\text{---}} \alpha(p) \,.\, p \,.\, p \qquad q \overset{p}{\text{---}} q \,.\, q \,.\, q$$
$$\beta(q) \overset{q}{\text{---}} q \,.\, \beta(q) \,.\, q \qquad r \overset{q}{\text{---}} r \,.\, r \,.\, r \qquad p \overset{f}{\text{---}} \sigma(p,p,p)$$
$$\gamma(r) \overset{r}{\text{---}} r \,.\, r \,.\, \gamma(r) \qquad \epsilon \overset{r}{\text{---}} \epsilon \,.\, \epsilon \,.\, \epsilon \ ,$$

- and the following rules in R':

$$\alpha(p) \overset{p}{\text{---}} \alpha(p) \,.\, p \,.\, p \qquad \epsilon \overset{p}{\text{---}} \epsilon \,.\, \epsilon \,.\, \epsilon$$
$$\beta(q) \overset{q}{\text{---}} q \,.\, \beta(q) \,.\, q \qquad p \overset{q}{\text{---}} p \,.\, p \,.\, p \qquad r \overset{f}{\text{---}} \sigma(r,r,r)$$
$$\gamma(r) \overset{r}{\text{---}} r \,.\, r \,.\, \gamma(r) \qquad q \overset{r}{\text{---}} q \,.\, q \,.\, q \ .$$

Figure 10 displays the rules R of the example MBOT M_{sort}. Since the rules R' are very similar, we omitted a graphical representation. The MBOT M_{sort} and M_{sort2} compute the tree transformations

$$M_{\text{sort}} = \{(\alpha^\ell(\beta^m(\gamma^n(\epsilon))),\ \sigma(\alpha^\ell(\epsilon), \beta^m(\epsilon), \gamma^n(\epsilon))) \mid \ell, m, n \in \mathbb{N}\}$$
$$M_{\text{sort2}} = \{(\gamma^n(\beta^m(\alpha^\ell(\epsilon))),\ \sigma(\alpha^\ell(\epsilon), \beta^m(\epsilon), \gamma^n(\epsilon))) \mid \ell, m, n \in \mathbb{N}\}\ ,$$

respectively. In other words, M_{sort} sorts all α-symbols into the first output subtree (below σ), the β-symbols into the second subtree, and the γ-symbols into the third subtree.

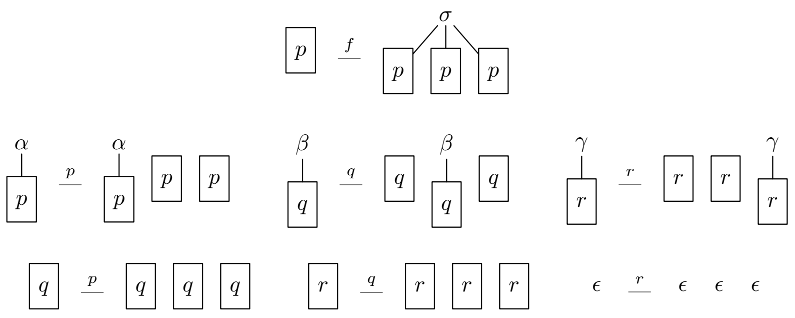

Fig. 10. Example rules of the MBOT M_{sort} of Ex. 23.

From the definition of the tree transformations of Ex. 23 we can evidently conclude some dependencies, which we depict in Fig. 11. Clearly, the shown dependencies are not strictly input hierarchical. However, they are input hierarchical and strictly output hierarchical. Consequently, the inverse dependencies are strictly input hierarchical and output hierarchical, but not strictly output hierarchical. In the same manner as for XTOP, we can conclude the following statement.

Lemma 24. *The inverse dependencies depicted in Fig. 11 are not compatible with the dependencies computed by any* MBOT.

Theorem 25 (see [39, Ex. 4.5]). *The inverse of the tree transformation illustrated in Fig. 11 cannot be computed by any* MBOT.

5 Synchronous tree-sequence substitution grammar

In this final section before the summary, we recall the synchronous tree-sequence substitution grammar (STSSG), which was introduced in [40, 51, 52, 45]. We keep the presentation terse because most mechanisms have been explained on the previous models.

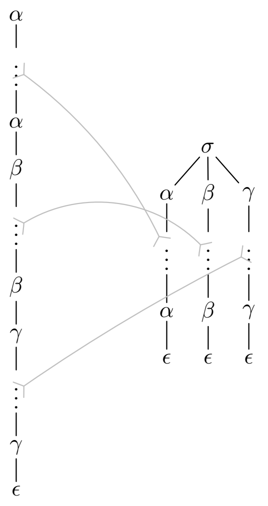

Fig. 11. Some dependencies of the tree transformation of Ex. 23.

Definition 26 (see [45, Sect. 2]). *A* synchronous tree-sequence substitution grammar *(*STSSG*) is a system* $(Q, \Sigma, \Delta, I, R)$ *where*

- Q *is a doubly ranked alphabet,*
- Σ *and* Δ *are ranked alphabets of* input *and* output symbols*, respectively,*
- $I \subseteq Q_{1,1}$ *is a subset of initial states, all of which are doubly unary, and*
- $R \subseteq T_\Sigma(Q)^* \times Q \times T_\Delta(Q)^*$ *is a finite set of rules such that* $\mathrm{rk}(q) = (m, n)$ *for every* $(l_1 \cdots l_m, q, r_1 \cdots r_n) \in R$.

For the rest of this section, let $M = (Q, \Sigma, \Delta, I, R)$ be an STSSG. Clearly, any MBOT is an STSSG, and moreover, any inverse transformation computed by an MBOT can be implemented by an STSSG. The ranks $\mathrm{rk}(q)$ of a state $q \in Q$ coincide with numbers $|\boldsymbol{l}|$ and $|\boldsymbol{r}|$ of trees in the left- and right-hand side of all rules $(\boldsymbol{l}, q, \boldsymbol{r}) \in R$. As before all initial states are doubly unary (i.e., have exactly one input and exactly one output tree). In this way, we again obtain a relation between input and output trees. As before, we call $\boldsymbol{l}$ and $\boldsymbol{r}$ of a rule $(\boldsymbol{l}, q, \boldsymbol{r}) \in R$ the left- and right-hand side, respectively. In accordance, we sometimes write $\boldsymbol{l} \overset{q}{—} \boldsymbol{r}$ instead of $(\boldsymbol{l}, q, \boldsymbol{r})$.

Definition 27. *Let* $\boldsymbol{l} \overset{q}{—} \boldsymbol{r} \in R$ *be a rule, and let* $v_1, \ldots, v_m, w_1, \ldots, w_n \in \mathbb{N}^*$, $\boldsymbol{v} = v_1 \cdots v_m$*, and* $\boldsymbol{w} = w_1 \cdots w_n$ *where* $\mathrm{rk}(q) = (m, n)$*. The rule's link structure* $\mathrm{links}_{\boldsymbol{v},\boldsymbol{w}}(\boldsymbol{l} \overset{q}{—} \boldsymbol{r}) \in \mathcal{L}$ *is*

$$\mathrm{links}_{\boldsymbol{v},\boldsymbol{w}}(\boldsymbol{l} \overset{q}{—} \boldsymbol{r}) = \bigcup_{p \in Q} \bigcup_{j=1}^{m} \bigcup_{i=1}^{n} \{(v_j v'_j, w_i w'_i) \mid v'_j \in \mathrm{pos}_p(l_j), w'_i \in \mathrm{pos}_p(r_i)\} \enspace .$$

As before, the semantics is presented using synchronous substitution. This time several states can be replaced in both the input and the output side of a sentential form.

Definition 28 (see [45, Sect. 2]). *Given two sentential forms $\langle \xi, D, \zeta \rangle$ and $\langle \xi', D', \zeta' \rangle$, we write $\langle \xi, D, \zeta \rangle \Rightarrow_M \langle \xi', D', \zeta' \rangle$ if there exists a rule $\boldsymbol{l} \overset{q}{—} \boldsymbol{r} \in R$, input positions $v_1, \ldots, v_m \in \mathrm{pos}_q(\xi)$, and output positions $w_1, \ldots, w_n \in \mathrm{pos}_q(\zeta)$ such that*

- $\mathrm{rk}(q) = (m, n)$,
- $\xi' = \xi[\boldsymbol{l}]_{\boldsymbol{v}}$ *and* $\zeta' = \zeta[\boldsymbol{r}]_{\boldsymbol{w}}$*, where* $\boldsymbol{v} = v_1 \cdots v_m$ *with* $v_1 \sqsubset \cdots \sqsubset v_m$ *and* $\boldsymbol{w} = w_1 \cdots w_n$ *with* $w_1 \sqsubset \cdots \sqsubset w_n$,
- *the positions are linked; i.e.,*

$$\{w_1, \ldots, w_n\} = \bigcup_{j=1}^{m} \{w \mid (v_j, w) \in D\}$$

$$\{v_1, \ldots, v_m\} = \bigcup_{i=1}^{n} \{v \mid (v, w_i) \in D\} \ ,$$

- $D' = D \cup \mathrm{links}_{\boldsymbol{v},\boldsymbol{w}}(\boldsymbol{l} \overset{q}{—} \boldsymbol{r})$.

As usual $\Rightarrow_M^*$ is the reflexive and transitive closure of $\Rightarrow_M$. The STSSG *M computes the dependencies* $\mathrm{dep}(M) \subseteq T_\Sigma \times \mathcal{L} \times T_\Delta$*, which are given by*

$$\mathrm{dep}(M) = \{\langle t, D, u \rangle \in T_\Sigma \times \mathcal{L} \times T_\Delta \mid \exists q \in I \colon \langle q, \{(\varepsilon, \varepsilon)\}, q \rangle \Rightarrow_M^* \langle t, D, u \rangle\} \ .$$

Moreover, the STSSG *M computes the relation $M \subseteq T_\Sigma \times T_\Delta$, which is given by $M = \{(t, u) \mid (t, D, u) \in \mathrm{dep}(M)\}$, and* $\mathrm{links}(M) = \{D \mid (t, D, u) \in \mathrm{dep}(M)\}$.

Example 29. It can easily be verified that M_{sort2}^{-1} (i.e., the inverse of M_{sort2} in which left- and right-hand side are exchanged) of Ex. 23 is an STSSG.

Lemma 30. *The set* $\mathrm{links}(M)$ *computed by an* STSSG *M is hierarchical with bounded distance.*

A final example will use this knowledge about the type of dependencies that are computable by an STSSG to show a tree transformation that cannot be computed by any STSSG.

Example 31 (see [39, Ex. 4.5] and [40]). The composition of the tree transformations M_{sort} and M_{sort2}^{-1} is

$$\{(\alpha^\ell(\beta^m(\gamma^n(\epsilon))),\ \gamma^n(\beta^m(\alpha^\ell(\epsilon)))) \mid \ell, m, n \in \mathbb{N}\} \ ,$$

which is shown in Fig. 12.

Figure 12 already shows some evident dependencies, and we easily notice that they are crossing. Consequently, no STSSG dependency is compatible with this dependence because they are all hierarchical by Lemma 30.

Lemma 32. *The dependencies depicted in Fig. 12 are not compatible with the dependencies computed by any* STSSG.

Theorem 33 (see [39, Ex. 4.5]). *The tree transformation illustrated in Fig. 12 cannot be computed by any* STSSG.

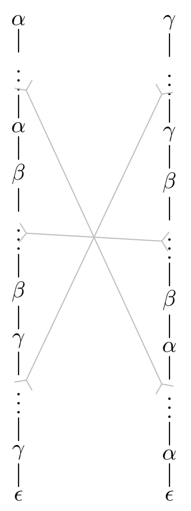

Fig. 12. Some dependencies of the tree transformation of Ex. 31.

6 Summary

We present the essential findings in the table below. It additionally contains the synchronous tree-adjoining grammar (STAG) [44, 42, 43], which has none of our hierarchy properties.

	input side	output side
XTOP	strictly hierarchical	strictly hierarchical
MBOT	hierarchical	strictly hierarchical
STSSG	hierarchical	hierarchical
STAG	—	—

References

1. Alshawi, H., Bangalore, S., Douglas, S.: Learning dependency translation models as collections of finite state head transducers. Comput. Linguist. 26(1), 45–60 (2000)
2. Arnold, A., Dauchet, M.: Bi-transductions de forêts. In: Michaelson, S., Milner, R. (eds.) Proc. ICALP. pp. 74–86. Edinburgh University Press (1976)
3. Arnold, A., Dauchet, M.: Morphismes et bimorphismes d'arbres. Theoret. Comput. Sci. 20(1), 33–93 (1982)
4. Baader, F., Nipkow, T.: Term rewriting and all that. Cambridge University Press (1998)
5. Berstel, J., Reutenauer, C.: Recognizable formal power series on trees. Theoret. Comput. Sci. 18(2), 115–148 (1982)
6. Bikel, D.M.: On the Parameter Space of Generative Lexicalized Statistical Parsing Models. Ph.D. thesis, University of Pennsylvania (2004)
7. Bloem, R., Engelfriet, J.: A comparison of tree transductions defined by monadic second order logic and by attribute grammars. J. Comput. System Sci. 61(1), 1–50 (2000)

8. Borchardt, B.: The Theory of Recognizable Tree Series. Ph.D. thesis, Technische Universität Dresden (2005)
9. Charniak, E., Knight, K., Yamada, K.: Syntax-based language models for statistical machine translation. In: Proc. MT Summit IX (2003)
10. Chiang, D.: An introduction to synchronous grammars. In: Proc. ACL. Association for Computational Linguistics (2006), part of a tutorial given with Kevin Knight
11. Collins, M.: Head-Driven Statistical Models for Natural Language Parsing. Ph.D. thesis, University of Pennsylvania (1999)
12. Courcelle, B., Franchi-Zannettacci, P.: Attribute grammars and recursive program schemes. Theoret. Comput. Sci. 17(2–3), 163–191, 235–257 (1982)
13. Dauchet, M.: Transductions inversibles de forêts. Thèse 3ème cycle, Université de Lille (1975)
14. Engelfriet, J., Fülöp, Z., Vogler, H.: Bottom-up and top-down tree series transformations. J. Autom. Lang. Combin. 7(1), 11–70 (2002)
15. Engelfriet, J., Lilin, E., Maletti, A.: Composition and decomposition of extended multi bottom-up tree transducers. Acta Inf. 46(8), 561–590 (2009)
16. Engelfriet, J., Maneth, S.: Macro tree transducers, attribute grammars, and MSO definable tree translations. Inform. and Comput. 154(1), 34–91 (1999)
17. Engelfriet, J., Maneth, S.: Macro tree translations of linear size increase are MSO definable. SIAM J. Comput. 32(4), 950–1006 (2003)
18. Engelfriet, J., Vogler, H.: Macro tree transducers. J. Comput. System Sci. 31(1), 71–146 (1985)
19. Engelfriet, J., Vogler, H.: Modular tree transducers. Theoret. Comput. Sci. 78(2), 267–303 (1991)
20. Fülöp, Z.: On attributed tree transducers. Acta Cybernet. 5(3), 261–279 (1981)
21. Fülöp, Z., Kühnemann, A., Vogler, H.: A bottom-up characterization of deterministic top-down tree transducers with regular look-ahead. Inf. Process. Lett. 91(2), 57–67 (2004)
22. Fülöp, Z., Kühnemann, A., Vogler, H.: Linear deterministic multi bottom-up tree transducers. Theoret. Comput. Sci. 347(1–2), 276–287 (2005)
23. Fülöp, Z., Maletti, A., Vogler, H.: Preservation of recognizability for synchronous tree substitution grammars. In: Drewes, F., Kuhlmann, M. (eds.) Proc. ATANLP. pp. 1–9. Association for Computational Linguistics (2010)
24. Gécseg, F., Steinby, M.: Tree Automata. Akadémiai Kiadó, Budapest (1984)
25. Gécseg, F., Steinby, M.: Tree languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 3, chap. 1, pp. 1–68. Springer (1997)
26. Graehl, J., Knight, K., May, J.: Training tree transducers. Comput. Linguist. 34(3), 391–427 (2008)
27. Klein, D., Manning, C.D.: Accurate unlexicalized parsing. In: Proc. ACL. pp. 423–430. Association for Computational Linguistics (2003)
28. Klein, D., Manning, C.D.: Fast exact inference with a factored model for natural language parsing. In: Proc. NIPS. pp. 3–10. MIT Press (2003)
29. Knight, K., Graehl, J.: An overview of probabilistic tree transducers for natural language processing. In: CICLing. LNCS, vol. 3406, pp. 1–24. Springer (2005)
30. Knuth, D.E.: Semantics of context-free languages. Math. Systems Theory 2(2), 127–145 (1968)
31. Koehn, P.: Statistical Machine Translation. Cambridge University Press (2010)
32. Kuich, W.: Tree transducers and formal tree series. Acta Cybernet. 14(1), 135–149 (1999)
33. Lilin, E.: Propriétés de clôture d'une extension de transducteurs d'arbres déterministes. In: Proc. CAAP. LNCS, vol. 112, pp. 280–289. Springer (1981)

34. Maletti, A.: Compositions of extended top-down tree transducers. Inform. and Comput. 206(9–10), 1187–1196 (2008)
35. Maletti, A.: An alternative to synchronous tree substitution grammars. J. Natur. Lang. Engrg. 17(2), 221–242 (2011)
36. Maletti, A.: How to train your multi bottom-up tree transducer. In: Proc. ACL. pp. 825–834. Association for Computational Linguistics (2011)
37. Maletti, A., Graehl, J., Hopkins, M., Knight, K.: The power of extended top-down tree transducers. SIAM J. Comput. 39(2), 410–430 (2009)
38. Milo, T., Suciu, D., Vianu, V.: Typechecking for XML transformers. J. Comput. System Sci. 66(1), 66–97 (2003)
39. Radmacher, F.G.: An automata theoretic approach to the theory of rational tree relations. Tech. Rep. AIB-2008-05, RWTH Aachen (2008)
40. Raoult, J.C.: Rational tree relations. Bull. Belg. Math. Soc. 4, 149–176 (1997)
41. Rounds, W.C.: Mappings and grammars on trees. Math. Systems Theory 4(3), 257–287 (1970)
42. Shieber, S.M.: Synchronous grammars as tree transducers. In: Proc. TAG+7. pp. 88–95 (2004)
43. Shieber, S.M.: Probabilistic synchronous tree-adjoining grammars for machine translation: The argument from bilingual dictionaries. In: Proc. SSST. pp. 88–95. Association for Computational Linguistics (2007)
44. Shieber, S.M., Schabes, Y.: Synchronous tree-adjoining grammars. In: Proc. CoLing. vol. 3, pp. 253–258 (1990)
45. Sun, J., Zhang, M., Tan, C.L.: A non-contiguous tree sequence alignment-based model for statistical machine translation. In: Proc. ACL. pp. 914–922. Association for Computational Linguistics (2009)
46. Thatcher, J.W.: Generalized^2 sequential machine maps. J. Comput. System Sci. 4(4), 339–367 (1970)
47. Thatcher, J.W.: Tree automata: An informal survey. In: Aho, A.V. (ed.) Currents in the Theory of Computing, pp. 143–172. Prentice Hall (1973)
48. Wu, D.: Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. Comput. Linguist. 23(3), 377–403 (1997)
49. Yamada, K., Knight, K.: A decoder for syntax-based statistical MT. In: Proc. ACL. pp. 303–310. Association for Computational Linguistics (2002)
50. Yu, S.: Regular languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 1, chap. 2, pp. 41–110. Springer (1997)
51. Zhang, M., Jiang, H., Aw, A., Li, H., Tan, C.L., Li, S.: A tree sequence alignment-based tree-to-tree translation model. In: Proc. ACL. pp. 559–567. Association for Computational Linguistics (2008)
52. Zhang, M., Jiang, H., Li, H., Aw, A., Li, S.: Grammar comparison study for translational equivalence modeling and statistical machine translation. In: Proc. CoLing. pp. 1097–1104. Association for Computational Linguistics (2008)
53. Zollmann, A., Venugopal, A., Och, F., Ponte, J.: A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In: Proc. CoLing. pp. 1145–1152. Association for Computational Linguistics (2008)