Unidirectional Derivation Semantics for Synchronous Tree-Adjoining Grammars*

Matthias Büchse[1,**], Andreas Maletti[2,**], and Heiko Vogler[1]

[1] Department of Computer Science, Technische Universität Dresden
01062 Dresden, Germany
{matthias.buechse,heiko.vogler}@tu-dresden.de

[2] Institute for Natural Language Processing, Universität Stuttgart
Pfaffenwaldring 5b, 70569 Stuttgart, Germany
andreas.maletti@ims.uni-stuttgart.de

Abstract. Synchronous tree-adjoining grammars have been given two types of semantics: one based on bimorphisms and one based on synchronous derivations, in both of which the input and output trees are constructed synchronously. We introduce a third type of semantics that is based on unidirectional derivations. It derives output trees based on a given input tree and thus marks a first step towards conditional probability distributions. We prove that the unidirectional semantics coincides with the bimorphism-based semantics with the help of a strong correspondence to linear and nondeleting extended top-down tree transducers with explicit substitution. In addition, we show that stateful synchronous tree-adjoining grammars admit a normal form in which only adjunction is used. This contrasts the situation encountered in the stateless case.

1 Introduction

A major task in natural-language processing is machine translation [17]; i.e., the automatic translation from one language into another. For this task engineers use a multitude of formal translation models such as *synchronous context-free grammars* [1,4] and *extended top-down tree transducers* [22,29,16,14]. Shieber claims [27] that these formalisms are too weak to capture naturally occurring translation. Instead he suggests *synchronous tree-adjoining grammars* [28,27], which are already used in some machine translation systems [30,21,6].

Here we consider synchronous tree-adjoining grammars with states (STAGs) as defined by Büchse et al. [2], who added states to traditional synchronous tree-adjoining grammars in the spirit of [11]. Product constructions and normal forms require states to maintain appropriate pieces of information along a derivation.

* This is an extended and revised version of: [A. Maletti: *A tree transducer model for synchronous tree-adjoining grammars.* In Proc. 48th Annual Meeting Association for Computational Linguistics, pages 1067–1076, 2010].

** The first and second author were supported by the German Research Foundation (DFG) grants VO 1011/6-1 and MA 4959/1-1, respectively.

H.-C. Yen and O.H. Ibarra (Eds.): DLT 2012, LNCS 7410, pp. 368–379, 2012.

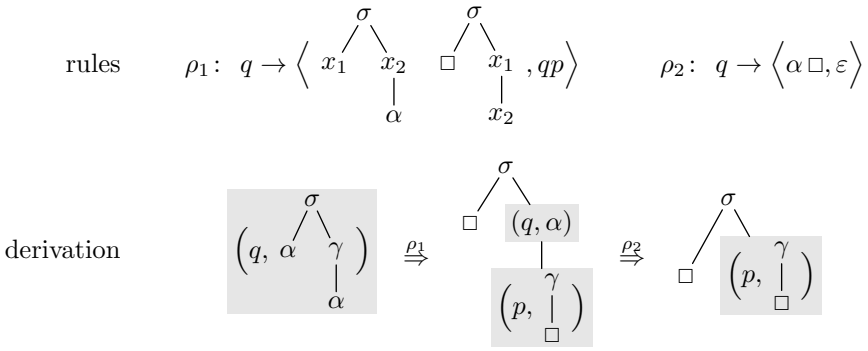

Fig. 1. State q has input rank 0 and output rank 1. In the first step of the derivation, the variables x_1 and x_2 are bound to α and $\gamma(\square)$, respectively.

In addition, states permit equivalence results without resorting to relabelings, which appear frequently in the literature on stateless synchronous tree-adjoining grammars [25,26].

Roughly speaking, a STAG is a linear nondeleting extended top-down tree transducer that additionally can use adjunction [15] (monadic second-order substitution). Two example rules of a STAG are illustrated in Fig. 1. In general, a STAG rule is of the form $q \to \langle \zeta\zeta', q_1 \cdots q_m \rangle$, in which the input tree ζ and the output tree ζ' are trees over the terminal symbols, the nullary substitution symbol $\square$, and the variables $x_1, \ldots, x_m$. Each variable establishes a (one-to-one) synchronization link between ζ and ζ', and the states $q_1, \ldots, q_m$ govern the links $x_1, \ldots, x_m$. Every state has an input rank i and an output rank j where $i, j \in \{0, 1\}$. If state q has input rank i, then the input tree of every rule with left-hand side q contains $\square$ exactly i times. The same interpretation applies to the output rank and the output tree. For example, the state q in Fig. 1 has rank $(0, 1)$. In this sense, we allow heterogeneous states with input rank 0 and output rank 1 (or vice versa).

We introduce a STAG semantics based on unidirectional derivations (see Fig. 1). This semantics processes the input tree and produces output trees in the same manner as the classical derivation semantics for tree transducers [22,29,8,10] and thus marks a first step towards conditional probability distributions; i.e., the probability of an output tree given an input tree. More precisely, given an input tree, the unidirectional derivation semantics precisely determines the decision tree containing all leftmost derivations, which corresponds to a Markov process. It remains a challenge to find a proper probability assignment for the rules because the input trees of different rules may overlap.

As usual in a derivation-based semantics, the application of a rule consists of a matching and a replacement phase. In the matching phase, we select a rule with left-hand side q and a pair consisting of state q and an input tree fragment in the sentential form (shaded in Fig. 1). Then we match the input tree fragment to the input tree of the rule by matching variables governed by states with input

rank 0 using first-order substitution, and second-order substitution otherwise. This yields a binding of the variables in the rule. In the replacement phase, the pair is replaced in the sentential form by the output tree of the rule, in which the variables are substituted appropriately and paired with the governing state. The derivation process is started with the pair consisting of the initial state and the input tree, and we apply derivation steps as long as possible. In the end, we obtain a sentential form of exclusively terminal symbols, which represents an output tree. In this way, every STAG computes a tree transformation, which is a binary relation on unranked trees.

To relate our semantics to the literature, we adapt the conventional bimorphism-based semantics for STAG [2], which develops the input and output tree synchronously. It coincides with the bimorphism semantics of [26], which in turn coincides with the conventional synchronous derivation semantics [7]. Our goal is to show that the unidirectional semantics coincides with the bimorphism semantics. We achieve this in three steps.

First we define (linear and nondeleting) extended top-down tree transducers (XTOPs) as particular STAGs, and we establish the equivalence of STAGs and certain XTOPs using explicit substitution under the unidirectional semantics. Second, it is known that every XTOP computes the same tree transformation using the bimorphism and the unidirectional derivation semantics [18]. This remains true for our particular unidirectional derivation semantics, which uses a leftmost derivation strategy. Third, we establish the equivalence corresponding to the first step under the bimorphism semantics. This last result was already announced in [19]. In contrast to [19], we present a full proof of it, and we make the restrictions on the XTOP obvious by avoiding partial evaluations of explicit substitutions.

All our results are contained in Corollaries 15 and 16. In particular, they show that stateful STAGs allow a normal form that uses only adjunction. Consequently, our STAGs with potentially heterogeneous states have the same expressive power as the STAGs of [2], which only have homogeneous states.

2 Preliminaries

The set of all nonnegative integers is $\mathbb{N}$. An *alphabet* is any finite nonempty set Σ of symbols. A *(monadic) doubly ranked alphabet* (Q, rk) is a finite set Q together with a rank mapping $\mathrm{rk}\colon Q \to \{0,1\}^2$. We also write $Q^{(m,n)} = \mathrm{rk}^{-1}(m,n)$ for all $m, n \in \{0,1\}$. We just write Q for the doubly ranked alphabet, if the ranking is obvious. We set $Q^{(m,*)} = Q^{(m,0)} \cup Q^{(m,1)}$ and $Q^{(*,n)} = Q^{(0,n)} \cup Q^{(1,n)}$.

The set U_Σ of all *(unranked) trees over* Σ is inductively defined to be the smallest set U such that $\sigma(t_1, \ldots, t_k) \in U$ for every $k \in \mathbb{N}$, $\sigma \in \Sigma$, and $t_1, \ldots, t_k \in U$. To avoid excessive quantification, we often drop expressions like "for all $k \in \mathbb{N}$" if they are obvious from the context. Sometimes we assign a *rank* $k \in \mathbb{N}$ to a symbol $\sigma \in \Sigma$ and then require that every occurrence of σ in a tree has exactly k successors. The set C_Σ contains all trees $t \in U_{\Sigma \cup \{\Box\}}$ in which the nullary symbol $\Box$ occurs exactly once.

Fig. 2. Illustration of the two forms of substitution.

For a tree $t \in U_\Sigma$, we denote the set of its *positions* by $\mathrm{pos}(t)$, where each position is encoded in the usual way as a sequence of positive integers; i.e., $\mathrm{pos}(t) \subseteq \mathbb{N}^*$ (Gorn's notation). The label of t at $v \in \mathrm{pos}(t)$ is $t(v)$. Given $\Delta \subseteq \Sigma$, we let $\mathrm{pos}_\Delta(t) = \{v \in \mathrm{pos}(t) \mid t(v) \in \Delta\}$. If $v \in \mathrm{pos}(t)$ has no successors, then $t\langle\!|u|\!\rangle_v^0$ denotes the tree that is obtained from t by replacing the leaf at v by the tree $u \in U_\Sigma$ (*first-order substitution*). If v has exactly one successor namely the subtree t', then $t\langle\!|u|\!\rangle_v^1$ denotes the tree that is obtained from t by replacing the subtree at v by $u\langle\!|t'|\!\rangle_{v'}^0$ where $u \in C_\Sigma$ and $v' = \mathrm{pos}_{\{\Box\}}(u)$ (*monadic second-order substitution*). If the symbol $t(v)$ occurs exactly once in t, then we also write $t\langle\!|t(v)/u|\!\rangle^i$ instead of $t\langle\!|u|\!\rangle_v^i$. Figure 2 illustrates the two forms of substitution.

A *regular tree grammar (in normal form)* [12,13] is a tuple $H = (P, \Sigma, p_0, R)$ where P is a finite set (*states*), Σ is an alphabet with $P \cap \Sigma = \emptyset$, $p_0 \in P$ (*initial state*), and R is a finite set of *rules*; every rule has the form $p \to \sigma(p_1, \ldots, p_k)$ where $p, p_1, \ldots, p_k \in P$ and $\sigma \in \Sigma$ (note that $\sigma(p_1, \ldots, p_k)$ is a tree over $\Sigma \cup P$). The *derivation relation induced by* H is the binary relation $\Rightarrow_H$ over $U_{\Sigma \cup P}$ such that $s \Rightarrow_H t$ if and only if there are a rule $p \to \sigma(p_1, \ldots, p_k)$ in R and a position $v \in \mathrm{pos}_{\{p\}}(s)$ such that $t = s\langle\!|\sigma(p_1, \ldots, p_k)|\!\rangle_v^0$. The *tree language generated by* H is $L(H) = \{t \in U_\Sigma \mid p_0 \Rightarrow_H^* t\}$.

3 Synchronous Tree-Adjoining Grammars

We extend the STAG syntax of [2] to allow heterogeneous states. Recall that occurrences of x_j can have different rank in different trees. We denote states by variants of q.

Definition 1. A *synchronous tree-adjoining grammar with states (STAG)* is a tuple $G = (Q, \Sigma, q_0, R)$ where

- Q is a monadic doubly-ranked alphabet (of *states*),
- Σ is an alphabet (*terminal alphabet*),
- $q_0 \in Q^{(0,0)}$ (*initial state*),
- R is a finite set of *rules* of the form $q \to \langle \zeta\zeta', q_1 \cdots q_m \rangle$ where the following holds for ζ (and the same holds for ζ' with $Q^{(*,i)}$ instead of $Q^{(i,*)}$):
  - ζ is a tree over $\Sigma \cup \{x_1, \ldots, x_m\} \cup \{\Box\}$,
  - $\Box$ occurs exactly i times in ζ if $q \in Q^{(i,*)}$,
  - every x_j occurs exactly once in ζ, and it has rank i in ζ if $q_j \in Q^{(i,*)}$. $\diamond$

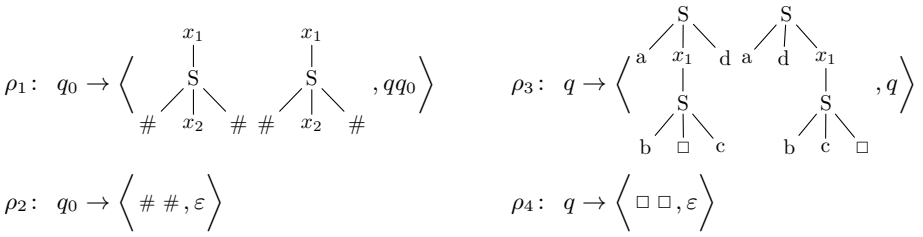

Fig. 3. Rules of the STAG G_{ex} with $q_0 \in Q^{(0,0)}$ and $q \in Q^{(1,1)}$.

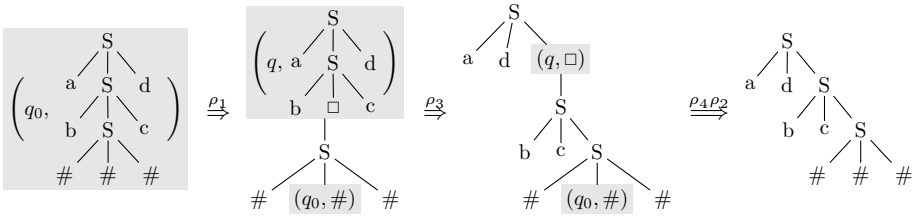

Fig. 4. Example derivation of the STAG G_{ex} of Fig. 3.

The STAG $G = (Q, \Sigma, q_0, R)$ is a (linear and nondeleting) *extended top-down tree transducer (XTOP)* if $Q = Q^{(0,0)}$. Figure 3 shows the rules of the example STAG G_{ex}, which is taken from [3, Fig. 2(a)]. It is not an XTOP. Next, we introduce the unidirectional derivation semantics for STAG. To this end, let $G = (Q, \Sigma, q_0, R)$ be a STAG and $\Delta = Q \times U_{\Sigma \cup \{\square\}}$.

Definition 2. Let ρ be the rule $q \to \langle \zeta\zeta', q_1 \cdots q_m \rangle$ in R. We define the binary relation $\overset{\rho}{\Rightarrow}$ over $U_{\Sigma \cup \{\square\} \cup \Delta}$ as follows: $\xi_1 \overset{\rho}{\Rightarrow} \xi_2$ (or: $\xi_1 \Rightarrow \xi_2$ via ρ) if and only if there is a minimal element $v \in \mathrm{pos}_{\Delta}(\xi_1)$ with respect to the lexicographic ordering and there are $t_1, \ldots, t_m \in U_\Sigma \cup C_\Sigma$ such that

1. $\square$ occurs exactly i times in t_j if $q_j \in Q^{(i,*)}$,
2. $\xi_1(v) = (q, \zeta\theta_1 \cdots \theta_m)$ with $\theta_j = (\!|x_j/t_j|\!)^i$ for $q_j \in Q^{(i,*)}$, and
3. $\xi_2 = \xi_1(\!|\zeta'\theta'_1 \cdots \theta'_m|\!)^i_v$ for $q \in Q^{(*,i)}$, where $\theta'_j = (\!|x_j/(q_j, t_j)|\!)^0$ if $q_j \in Q^{(*,0)}$ and $\theta'_j = (\!|x_j/(q_j, t_j)(\square)|\!)^1$ otherwise.

For every $\rho_1, \ldots, \rho_n \in R$, we let $\overset{d}{\Rightarrow} = \overset{\rho_1}{\Rightarrow}; \cdots; \overset{\rho_n}{\Rightarrow}$ where $d = \rho_1 \cdots \rho_n$ and semicolon denotes the composition of binary relations. For every $p \in Q^{(0,0)}$ we define the tree transformation

$$\kappa^p_G = \{(s,t) \in U_\Sigma \times U_\Sigma \mid \exists d \in R^* \colon (p,s) \overset{d}{\Rightarrow} t\} \ .$$

The STAG G *derivation-induces* the tree transformation $\kappa_G = \kappa^{q_0}_G$. ◇

An example derivation is demonstrated in Fig. 4. In the second step, we have $\rho = \rho_3$ with input tree ζ and output tree ζ', $m = 1$, ξ_1 and ξ_2 as in the figure, and $t_1 = \square$. Consequently, $\theta_1 = (\!|x_1/\square|\!)^1$ and $\theta'_1 = (\!|x_1/(q, \square)(\square)|\!)^1$.

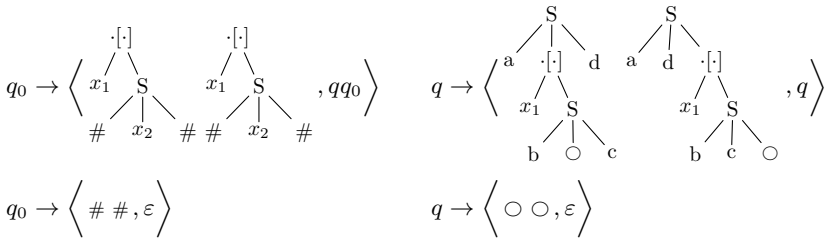

Fig. 5. XTOP M_{ex} using explicit substitution.

4 Relating STAG and XTOP

In this section, we show that STAGs are essentially as powerful as XTOPs using explicit substitution (both with respect to the unidirectional derivation semantics). Our construction builds on the ideas of [9, Thm. 3] and [5, Prop. 4.19].

We begin by defining explicit substitution [19]. Let $\cdot[\cdot]$ be a special binary symbol, which indicates the substitution replacing all $\bigcirc$ in its first child by its second child. For every alphabet Σ with $\cdot[\cdot], \bigcirc \notin \Sigma$, let $\underline{\Sigma} = \Sigma \cup \{\cdot[\cdot], \bigcirc\}$ where $\bigcirc$ is nullary. The evaluation $\cdot^{\mathrm{E}} \colon U_{\underline{\Sigma}} \to U_{\Sigma \cup \{\square\}}$ is inductively defined by $\bigcirc^{\mathrm{E}} = \square$, $\sigma(t_1, \ldots, t_k)^{\mathrm{E}} = \sigma(t_1^{\mathrm{E}}, \ldots, t_k^{\mathrm{E}})$ for every $\sigma \in \Sigma$ and $t_1, \ldots, t_k \in U_{\underline{\Sigma}}$, and for every $t, u \in U_{\underline{\Sigma}}$ the tree $\cdot[\cdot](t, u)^{\mathrm{E}}$ is obtained by replacing all occurrences of $\bigcirc$ in t^{E} by u^{E}. We lift $\cdot^{\mathrm{E}}$ to a tree transformation τ by $\tau^{\mathrm{E}} = \{(s^{\mathrm{E}}, t^{\mathrm{E}}) \mid (s, t) \in \tau\}$.

Figure 5 shows the rules of the XTOP M_{ex} using explicit substitution. We claim that $(\kappa_{M_{\mathrm{ex}}})^{\mathrm{E}} = \kappa_{G_{\mathrm{ex}}}$, and we will provide a proof of this claim in this section. In the following, let $M = (P, \underline{\Sigma}, p_0, R')$ be an XTOP using explicit substitution. The next definition essentially captures the appropriate use of the substitution symbol.

Definition 3. A tree $t \in U_{\underline{\Sigma}}$ is *well-behaved* (under $\cdot^{\mathrm{E}}$) if $t^{\mathrm{E}} \in U_{\Sigma}$ and $t_1^{\mathrm{E}} \in C_{\Sigma}$ for every subtree of the form $\cdot[\cdot](t_1, t_2)$ in t. A tree transformation $\tau \subseteq U_{\underline{\Sigma}} \times U_{\underline{\Sigma}}$ is *well-behaved* if it only contains pairs of well-behaved trees. Finally, M is well-behaved if κ_M is well-behaved. $\diamond$

We observe that $t^{\mathrm{E}} \in U_{\Sigma}$ if and only if each occurrence of $\bigcirc$ in t is inside the first subtree of some occurrence of $\cdot[\cdot]$, which is clearly a regular (or, equivalently, recognizable) property. Similarly, $t^{\mathrm{E}} \in C_{\Sigma}$ if and only if all but exactly one occurrence of $\bigcirc$ fulfill the previous condition. This is again a regular property.

Next, we distinguish four types of states to establish a normal form for well-behaved XTOPs.

Definition 4. A state $q \in Q$ is an *input i-state* with $i \in \{0, 1\}$ if t^{E} contains $\square$ exactly i times for every $(t, u) \in \kappa_M^q$. The same notions are defined for the output side. $\diamond$

Recall that both the domain and the range of κ_M are effectively regular (by a combination of [8, Cor. 3.11] and [18, Thm. 4]). By the remarks below Def. 3 and the decidability of inclusion for regular tree languages [12, Thm. II.10.3], we can decide whether a state is an input 0-state or an input 1-state (or neither). Thus, we can effectively compute the following subsets of P:

$$P_{i,j} = \{p \in P \mid p \text{ is an input } i\text{-state and an output } j\text{-state}\} \ .$$

Definition 5. The XTOP M is *substitution normalized* if $p_0 \in P_{0,0}$, the sets $P_{i,j}$ form a partition of P, and for every rule $p \to \langle \zeta\zeta', p_1 \cdots p_m\rangle$ in R' and position $v \in \mathrm{pos}(\zeta)$:

- if $\zeta(v) = \cdot[\cdot]$, then $\zeta(v1) = x_j$ with input 1-state p_j, and
- if $\zeta(v) = x_j$ with input 1-state p_j, then $v = v'1$ for some v' and $\zeta(v') = \cdot[\cdot]$.

The same conditions are required for the output side. ◇

Lemma 6. *For every well-behaved XTOP M there is a substitution normalized XTOP M' with $\kappa_M = \kappa_{M'}$ and vice versa.*

Proof. Here we only show how to rearrange the input trees in the rules to obtain the form required in a substitution normalized XTOP. In essence, we push each occurrence of the substitution symbol $\cdot[\cdot]$ down towards a $\bigcirc$ or a variable corresponding to an input 1-state. This is achieved by replacing

- $\cdot[\cdot](\cdot[\cdot](t_1, t_i), t_2)$ by $\cdot[\cdot](t_1, \cdot[\cdot](t_i, t_2))$
- $\cdot[\cdot](\sigma(t_1, \ldots, t_k), t')$ by $\sigma(t_1, \ldots, t_{i-1}, \cdot[\cdot](t_i, t'), t_{i+1}, \ldots, t_k)$
- $\cdot[\cdot](\bigcirc, t')$ by t'

if $\bigcirc$ or a variable corresponding to an input 1-state occurs in t_i. These replacements are iterated. Finally, if x_j with an input 1-state q_j occurs outside the first subtrees of all occurrences of $\cdot[\cdot]$ (which may happen in rules for input 1-states), then we replace x_j by $\cdot[\cdot](x_j, \bigcirc)$. Clearly, these transformations preserve the semantics. □

Now we can make our claim more precise. We want to show that for every STAG G there is a well-behaved XTOP M such that $\kappa_G = (\kappa_M)^{\mathrm{E}}$ and vice versa. To this end, we first relate the STAG $G = (Q, \Sigma, q_0, R)$ and the substitution normalized XTOP $M = (P, \underline{\Sigma}, p_0, R')$.

Definition 7. The STAG G and the substitution normalized XTOP M are *related* if $Q^{(i,j)} = P_{i,j}$, $q_0 = p_0$, and

$$R = \{p \to \langle \mathrm{tr}(\zeta)\, \mathrm{tr}(\zeta'), p_1 \cdots p_m\rangle \mid p \to \langle \zeta\zeta', p_1 \cdots p_m\rangle \in R'\} \ ,$$

where the partial mapping $\mathrm{tr}\colon U_{\underline{\Sigma} \cup X_m} \to U_{\Sigma \cup X_m \cup \{\square\}}$, with $X_m = \{x_1, \ldots, x_m\}$, is given by (note that variables may occur with different rank in $\mathrm{tr}(t)$ and t)

$$\mathrm{tr}(\bigcirc) = \square \qquad \mathrm{tr}(\sigma(t_1, \ldots, t_k)) = \sigma(\mathrm{tr}(t_1), \ldots, \mathrm{tr}(t_k))$$
$$\mathrm{tr}(x_j) = x_j \qquad \mathrm{tr}(\cdot[\cdot](x_j, t)) = x_j(\mathrm{tr}(t)) \ . \qquad ◇$$

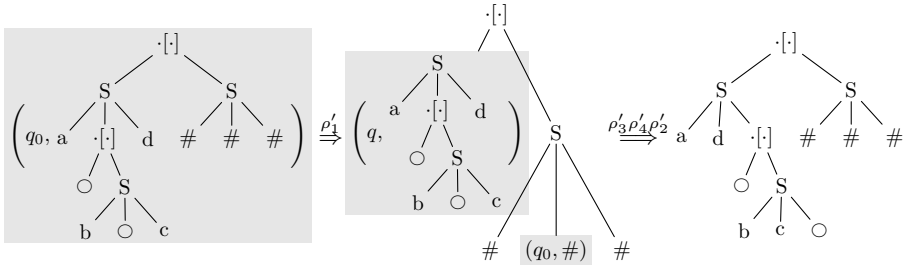

Fig. 6. Derivation of the XTOP M_{ex} of Fig. 5, where $\text{tr}(\rho'_j) = \rho_j$. It corresponds to the derivation of Fig. 4 via 'eval'.

It is simple to check that Def. 7 is constructive. The STAG G_{ex} of Fig. 3 and the XTOP M_{ex} of Fig. 5 are related. We note that the mapping 'tr' is similar to the mappings 'ateb' in [5, Prop. 4.10] and YIELD_f in [10, Lm. 5.8]. Next, we show that the second-order substitution (adjunction) of a STAG can be delayed in the same manner as for macro tree transducers [10, Lm. 5.5].

Lemma 8. *If the STAG G and the XTOP M are related, then $\kappa_G = (\kappa_M)^{\text{E}}$.*

Proof. First we lift the evaluation $\cdot^{\text{E}}$ to sentential forms. Let $\underline{\Delta} = Q \times U_{\underline{\Sigma} \cup \{\square\}}$. We define the mapping $\text{eval}\colon U_{\underline{\Sigma} \cup \underline{\Delta}} \to U_{\Sigma \cup \{\square\} \cup \Delta}$ as follows:

$$\text{eval}(○) = \square \qquad \text{eval}(\sigma(t_1, \ldots, t_k)) = \sigma(\text{eval}(t_1), \ldots, \text{eval}(t_k))$$

$$\text{eval}((q,t)) = (q, \text{eval}(t)) \qquad \text{eval}(\cdot[\cdot](t,t')) = \begin{cases} \text{eval}(t)(\text{eval}(t')) & \text{if } t \in \underline{\Delta} \\ \text{eval}(t)(\!|\square / \text{eval}(t')|\!)^0 & \text{if } t \notin \underline{\Delta} \end{cases}$$

The derivations of Figs. 4 and 6 are related via 'eval'.

Second we prove the equation $\kappa_G = (\kappa_M)^{\text{E}}$ in five steps (using $\Rightarrow_G$ and $\Rightarrow_M$ to denote the derivation relations of G and M, respectively).

1. We can uniquely reconstruct from any (successful) derivation of G or M the pair of input and output tree.
2. Since the state behavior is preserved, any derivation d is successful for M if and only if $\text{tr}(d)$ is successful for G, where we lift 'tr' from trees to rule sequences in the obvious manner.
3. $\xi_1 \Rightarrow_M \xi_2$ via ρ implies $\text{eval}(\xi_1) \Rightarrow_G \text{eval}(\xi_2)$ via $\text{tr}(\rho)$ (by construction).
4. We prove $\kappa_G \subseteq (\kappa_M)^{\text{E}}$. For this let $(s,t) \in \kappa_G$. By definition, there is a sequence $d \in (R')^*$ of rules such that $(q_0, s) \Rightarrow_G t$ via $\text{tr}(d)$. By Statement 2, there are s' and t' such that $(p_0, s') \Rightarrow_M t'$ via d. The inductive extension of Statement 3 yields that $(q_0, \text{eval}(s')) \Rightarrow_G \text{eval}(t')$ via $\text{tr}(d)$. Finally, we obtain $\text{eval}(s') = s$ and $\text{eval}(t') = t$ from Statement 1.
5. The statement $(\kappa_M)^{\text{E}} \subseteq \kappa_G$ follows directly from the inductive extension of Statement 3. □

Theorem 9. *For every STAG G there is a well-behaved XTOP M such that $\kappa_G = (\kappa_M)^{\mathrm{E}}$, and vice versa.*

Proof. The statement follows directly from Def. 7 and Lemmas 6 and 8. □

5 Bimorphism Semantics

Now we define a semantics for STAGs in terms of bimorphisms [26], which we adapt from [2]. As before, let $G = (Q, \Sigma, q_0, R)$ be a STAG. First, we define the regular tree grammar D_G, which generates the derivation trees of G. Second, we define two mappings $h_1^{(0)}$ and $h_2^{(0)}$, which retrieve from a derivation tree the derived input tree and output tree, respectively.

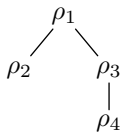

Fig. 7. Derivation tree.

Definition 10. For every $p \in Q$ the *p-derivation grammar of G* is the regular tree grammar $D_G^p = (Q, R, p, R'')$ where

$$R'' = \{q \to \rho(q_1, \dots, q_m) \mid \rho = q \to \langle \zeta\zeta', q_1 \cdots q_m \rangle \in R\} \ .$$

For the input side, we define the *embedded tree homomorphisms*

$$h_1^{(0)} \colon \textstyle\bigcup_{q \in Q^{(0,*)}} L(D_G^q) \to U_\Sigma \qquad \text{and} \qquad h_1^{(1)} \colon \textstyle\bigcup_{q \in Q^{(1,*)}} L(D_G^q) \to C_\Sigma \ ,$$

simultaneously as follows. Let ρ be a rule $q \to \langle \zeta\zeta', q_1 \cdots q_m \rangle$ in R with $q \in Q^{(i,*)}$. Then $h_1^{(i)}(\rho(d_1, \dots, d_m)) = \zeta\theta_1 \cdots \theta_m$ with $\theta_j = (\!|x_j / h_1^{(k)}(d_j)|\!)^k$ for $q_j \in Q^{(k,*)}$.

For the output side, the embedded tree homomorphisms $h_2^{(0)}$ and $h_2^{(1)}$ are defined in the same way, but using the output tree ζ' of ρ and $Q^{(*,i)}$ instead of $Q^{(i,*)}$. For every $p \in Q^{(0,0)}$ we define the tree transformation

$$\tau_G^p = \{(h_1^{(0)}(d), h_2^{(0)}(d)) \in U_\Sigma \times U_\Sigma \mid d \in L(D_G^p)\} \ .$$

The STAG G *bimorphism-induces* the tree transformation $\tau_G = \tau_G^{q_0}$. ◇

Figure 7 shows the derivation tree corresponding to the derivation of Fig. 4. We note that if G is an XTOP, then $h_1^{(0)}$ and $h_2^{(0)}$ are linear nondeleting tree homomorphisms in the sense of [12]. For XTOPs we recall the following theorem.

Theorem 11. *We have that $\tau_M = \kappa_M$ for every XTOP M.*

Proof. The proof of [18, Thm. 4] also applies to our (leftmost) unidirectional derivation semantics. □

6 Relating STAG and XTOP, again

In this section, we compare STAG and XTOP with explicit substitution, this time with respect to the bimorphism semantics.

Lemma 12. *If the STAG G and the XTOP M are related, then $\tau_G = (\tau_M)^{\mathrm{E}}$.*

Proof. Since 'tr' is bijective between R and R', it is also bijective between $L(D_G)$ and $\mathrm{tr}(L(D_M))$, where 'tr' is extended in the natural fashion to a deterministic relabeling. Hence, we can restrict our attention to the embedded tree homomorphisms. For reasons of symmetry we only consider the input side. To avoid confusion, we augment the subscript of the embedded tree homomorphisms by the respective grammar.

We prove the following statement by structural induction on d: for every $d \in \bigcup_{p \in P} L(D_M^p)$ we have that $h_{G,1}(\mathrm{tr}(d)) = (h_{M,1}^{(0)}(d))^{\mathrm{E}}$ with $h_{G,1} = h_{G,1}^{(0)} \cup h_{G,1}^{(1)}$. To this end, let $d = \rho(d_1, \ldots, d_m)$ with $\rho = p \to \langle \zeta\zeta', p_1 \cdots p_m \rangle$ in R'. Then

$$h_{G,1}(\mathrm{tr}(d)) = \mathrm{tr}(\zeta)\theta'_1 \cdots \theta'_m \overset{(\star)}{=} (\zeta\theta_1 \cdots \theta_m)^{\mathrm{E}} = (h_{M,1}^{(0)}(d))^{\mathrm{E}} \ ,$$

where $\theta'_j = (\!|x_j / h_{G,1}^{(k)}(\mathrm{tr}(d_j))|\!)^k$ for $q_j \in Q^{(k,*)}$. For $(\star)$, we prove the following statement: for every $\zeta \in U_{\underline{\Sigma} \cup X_m}$ such that each element of $X_m = \{x_1, \ldots, x_m\}$ occurs at most once, we have that $\mathrm{tr}(\zeta)\langle \theta'_j \rangle_{x_j \in \mathrm{var}(\zeta)} = (\zeta \langle \theta_j \rangle_{x_j \in \mathrm{var}(\zeta)})^{\mathrm{E}}$ where $\mathrm{var}(\zeta)$ is the set of all variables that occur in ζ. □

Theorem 13. *For every STAG G there is a well-behaved XTOP M such that $\tau_G = (\tau_M)^{\mathrm{E}}$, and vice versa.*

Proof. The statement follows from Def. 7, Lm. 12, Lm. 6, and Thm. 11. □

7 Results

In this section, we summarize our results. First we prove a normal form theorem. Its construction is inspired by the lexicalization of tree substitution grammars via tree-adjoining grammars [24,15,20]. As before, let $G = (Q, \Sigma, q_0, R)$ be a STAG. It is *uniform* if $Q = Q^{(1,1)} \cup \{q_0\}$ and the initial state does not occur in the right-hand side of any rule.

Theorem 14. *For every STAG G there is a uniform STAG G' with $\tau_G = \tau_{G'}$.*

Proof. Without loss of generality, let G be such that q_0 does not occur in the right-hand side of any rule of R. As an intermediate step, we construct an input-uniform STAG G' with states $P = P^{(1,*)} \cup \{q_0\}$ such that $\tau_G = \tau_{G'}$.

We set $G' = (P, \Sigma, q_0, R')$ with $P^{(0,0)} = \{q_0\}$, $P^{(1,i)} = Q^{(1,i)} \cup \big(\Sigma \times Q^{(0,i)}\big)$, and the rules are defined as follows. Let ρ be a rule $q \to \langle \zeta\zeta', q_1 \cdots q_m \rangle$ in R, and $\alpha_1, \ldots, \alpha_m \in \Sigma$. We distinguish three cases.

- Case 1 (initial): Let $q \in Q^{(1,*)}$ or $q = q_0$. Then $q \to \langle \bar{\zeta}\zeta', \bar{q}_1 \cdots \bar{q}_m \rangle$ is in R' where for every j we set $\bar{q}_j = (\alpha_j, q_j)$ if $q_j \in Q^{(0,*)}$ and $\bar{q}_j = q_j$ if $q_j \in Q^{(1,*)}$, and $\bar{\zeta}$ is obtained from ζ by replacing every nullary occurrence of x_j by $x_j(\alpha_j)$.

- Case 2 (transport): Let $q \in Q^{(0,*)}$, $q \neq q_0$, and i such that x_i occurs nullary in ζ. Then $(\alpha_i, q) \to \langle \bar{\zeta}\zeta', \bar{q}_1 \cdots \bar{q}_m \rangle$ is in R' where $\bar{q}_j$ is as in Case 1 and $\bar{\zeta}$ is obtained from ζ by replacing x_i by $x_i(\Box)$ and replacing every nullary occurrence of x_j by $x_j(\alpha_j)$ for $j \neq i$.
- Case 3 (check): Let $q \in Q^{(0,*)}$, $q \neq q_0$, and x_j occurs unary in ζ for every j. For every leaf $v \in \mathrm{pos}_\Sigma(\zeta)$, the rule $(\zeta(v), q) \to \langle \zeta(\!|\Box|\!)^0_v \zeta', q_1 \cdots q_m \rangle$ is in R'.
- No further rules are in R'.

We omit the proof that $\tau_{G'} = \tau_G$. For reasons of symmetry, a version of the construction can be defined that produces an output-uniform STAG. Note that both constructions preserve input- and output-uniformity. We obtain the desired uniform STAG by applying both constructions in sequence. □

Corollary 15. *For every STAG G we have $\tau_G = \kappa_G$.*

Proof. The construction in Def. 7 is bijective between substitution normalized XTOPs and STAGs. We obtain the result by Lemmas 8 and 12 and Thm. 11. □

Corollary 16. *Let $\tau \subseteq U_\Sigma \times U_\Sigma$. The following are equivalent:*

1. *There is a STAG G with $\tau = \kappa_G$.*
2. *There is a well-behaved XTOP M with $\tau = (\kappa_M)^{\mathrm{E}}$.*
3. *There is a well-behaved XTOP M with $\tau = (\tau_M)^{\mathrm{E}}$.*
4. *There is a STAG G with $\tau = \tau_G$.*
5. *There is a uniform STAG G with $\tau = \tau_G$.*

Proof. The equivalences $(1 \Leftrightarrow 2)$, $(2 \Leftrightarrow 3)$, $(3 \Leftrightarrow 4)$, and $(4 \Leftrightarrow 5)$ are Theorems 9, 11, 13, and 14, respectively. □

References

1. Aho, A.V., Ullman, J.D.: The Theory of Parsing, Translation, and Compiling. Prentice Hall (1972)
2. Büchse, M., Nederhof, M.J., Vogler, H.: Tree parsing with synchronous tree-adjoining grammars. In: Proc. Parsing Technologies. pp. 14–25. ACL (2011)
3. Büchse, M., Nederhof, M.J., Vogler, H.: Tree parsing for tree-adjoining machine translation (2012), submitted; `www.inf.tu-dresden.de/index.php?node_id=1571`
4. Chiang, D.: Hierarchical phrase-based translation. Comput. Linguist. 33(2), 201–228 (2007)
5. Courcelle, B., Franchi-Zannettacci, P.: Attribute grammars and recursive program schemes I. Theoret. Comput. Sci. 17(2), 163–191 (1982)
6. DeNeefe, S.: Tree-Adjoining Machine Translation. Ph.D. thesis, University of Southern California (2011)
7. DeNeefe, S., Knight, K., Vogler, H.: A decoder for probabilistic synchronous tree insertion grammars. In: Proc. Applications of Tree Automata in Natural Language Processing. pp. 10–18. ACL (2010)
8. Engelfriet, J.: Bottom-up and top-down tree transformations—a comparison. Math. Systems Theory 9(3), 198–231 (1975)

9. Engelfriet, J.: Some open questions and recent results on tree transducers and tree languages. In: Book, R.V. (ed.) Formal Language Theory—Perspectives and Open Problems, pp. 241–286. Academic Press (1980)
10. Engelfriet, J., Vogler, H.: Macro tree transducers. J. Comput. System Sci. 31(1), 71–146 (1985)
11. Fülöp, Z., Maletti, A., Vogler, H.: Preservation of recognizability for synchronous tree substitution grammars. In: Proc. Applications of Tree Automata in Natural Language Processing. pp. 1–9. ACL (2010)
12. Gécseg, F., Steinby, M.: Tree Automata. Akadémiai Kiadó, Budapest (1984)
13. Gécseg, F., Steinby, M.: Tree languages. In: Rozenberg and Salomaa [23], chap. 1, pp. 1–68
14. Graehl, J., Knight, K., May, J.: Training tree transducers. Comput. Linguist. 34(3), 391–427 (2008)
15. Joshi, A., Schabes, Y.: Tree-adjoining grammars. In: Rozenberg and Salomaa [23], chap. 2, pp. 69–123
16. Knight, K., Graehl, J.: An overview of probabilistic tree transducers for natural language processing. In: Proc. Computational Linguistics and Intelligent Text Processing. LNCS, vol. 3406, pp. 1–24. Springer (2005)
17. Koehn, P.: Statistical Machine Translation. Cambridge University Press (2010)
18. Maletti, A.: Compositions of extended top-down tree transducers. Inform. and Comput. 206(9–10), 1187–1196 (2008)
19. Maletti, A.: A tree transducer model for synchronous tree-adjoining grammars. In: Proc. Association for Computational Linguistics. pp. 1067–1076. ACL (2010)
20. Maletti, A., Engelfriet, J.: Strong lexicalization of tree adjoining grammars. In: Proc. Association for Computational Linguistics. ACL (2012), to appear
21. Nesson, R., Shieber, S.M., Rush, A.: Induction of probabilistic synchronous tree-insertion grammars for machine translation. In: Proc. Association for Machine Translation in the Americas (2006)
22. Rounds, W.C.: Mappings and grammars on trees. Math. Systems Theory 4(3), 257–287 (1970)
23. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages, vol. 3. Springer (1997)
24. Schabes, Y.: Mathematical and Computational Aspects of Lexicalized Grammars. Ph.D. thesis, University of Pennsylvania, Philadelphia (1990)
25. Shieber, S.M.: Synchronous grammars as tree transducers. In: Proc. Tree Adjoining Grammar and Related Formalisms. pp. 88–95 (2004)
26. Shieber, S.M.: Unifying synchronous tree adjoining grammars and tree transducers via bimorphisms. In: Proc. European Chapter of the Association for Computational Linguistics. pp. 377–384. ACL (2006)
27. Shieber, S.M.: Probabilistic synchronous tree-adjoining grammars for machine translation: The argument from bilingual dictionaries. In: Proc. Syntax and Structure in Statistical Translation. pp. 88–95. ACL (2007)
28. Shieber, S.M., Schabes, Y.: Synchronous tree-adjoining grammars. In: Proc. Computational Linguistics. pp. 253–258. University of Helsinki (1990)
29. Thatcher, J.W.: Generalized^2 sequential machine maps. J. Comput. System Sci. 4(4), 339–367 (1970)
30. The XTAG Project, `www.cis.upenn.edu/~xtag/`