Syntax-Directed Translations and Quasi-alphabetic Tree Bimorphisms — Revisited

Andreas Maletti* and Cătălin Ionuţ Tîrnăucă**

Universitat Rovira i Virgili
Departament de Filologies Romàniques
Av. Catalunya 35, 43002 Tarragona, Spain
andreas.maletti@urv.cat
catalinionut.tirnauca@estudiants.urv.cat

Abstract. Quasi-alphabetic tree bimorphisms by [Steinby, Tîrnăucă: Defining syntax-directed translations by tree bimorphisms. *Theor. Comput. Sci.*, to appear. http://dx.doi.org/10.1016/j.tcs.2009.03.009, 2009] are reconsidered. It is known that the class of (string) translations defined by such bimorphisms coincides with the class of syntax-directed translations. This result is extended to a smaller class of tree bimorphisms namely (linear and complete) symbol-to-symbol tree bimorphisms. Moreover, it is shown that the class of simple syntax-directed translations coincides with the class of translations defined by alphabetic tree bimorphisms (also known as finite-state relabelings). This proves that alphabetic tree bimorphisms are not sufficiently powerful to model all syntax-directed translations. Finally, it is shown that the class of tree transformations defined by quasi-alphabetic tree bimorphisms is closed under composition. The corresponding result is known in the variable-free case. Overall, the main results of [Steinby, Tîrnăucă] are strengthened.

Keywords: syntax-directed translation, regular tree language, tree bimorphism, natural language processing

1 Introduction

The field of syntax-based machine translation was established by the demanding need of systems used in practical translations between natural languages (for example, Arabic to English). Modern systems should be able to perform local rotations and capture syntax-sensitive transformations (i.e, tree transformations). Another important property that such a system should possess is composability. This property allows us to split the system into subsystems, which are easier to handle, train, and study. Those subsystems can then be assembled into a large system by an automatic composition construction [1,2].

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

** This author is indebted to the MEC project MTM-2007-63422 which made this work possible.

S. Bozapalidis and G. Rahonis (Eds.): CAI 2009, LNCS 5725, pp. 305–317, 2009.

Two powerful tools that define tree transformations have been proposed during the past decades in the formal language community: tree transducers and tree bimorphisms (see [3,4] for surveys). The former devices are operational and easy to implement but closure under composition only holds for few classes of tree transformations [5,3,2]. This closure is easier to establish using the latter devices by imposing suitable restrictions on their constituents [6,7,8,9], but tree bimorphisms are more difficult to implement. More precisely, a tree bimorphism is formed by two tree homomorphisms and a center tree language. The tree transformation is obtained by applying both homomorphisms to elements of the center tree language. One homomorphism yields the input tree and the other homomorphism yields the corresponding output tree. If we take the yield of the input and output tree, then we obtain a (string) translation.

Synchronous grammars [10,11,12] are another way to define tree transformations. They easily capture even difficult local rotations that are required by pairs of natural languages with very different syntax-structures (e.g., Chinese and English). A synchronous grammar basically consists of two grammars, in which the productions have associated nonterminals. The derivations are then obtained by applying two suitable rules, one of each grammar, to associated nonterminals. Again one side produces the input tree and the other side produces the output tree in this fashion. Unfortunately, few closure under composition results were known about such grammars until [13] related synchronous grammars and tree bimorphisms.

One synchronous grammar device is the syntax-directed translation schema (SDTS), which appeared first as a simple model of a compiler [10] (see [14] for a survey). In the spirit of [13], quasi-alphabetic tree bimorphisms [15] were shown to be as powerful as SDTSs for string translations. Moreover, for quasi-alphabetic tree bimorphisms, in which the center tree language does not permit variables, the class of tree transformations (and thus also the class of string translations) defined by them is shown to be closed under composition [15].

Here we sharpen the connection between SDTSs and tree bimorphisms. The class of all translations defined by SDTSs coincides with the class of all translations defined by (linear and complete) symbol-to-symbol tree bimorphisms (see Section 3). The latter devices define a strictly smaller class of tree transformations than quasi-alphabetic tree bimorphisms. In addition, simple SDTSs [16,17] are equally powerful as alphabetic tree bimorphisms [3] (finite-state relabelings [5]). Finally, we strengthen the closure under composition result of quasi-alphabetic tree bimorphisms by showing that the class of tree transformations defined by them remains closed under composition even if we allow variables in the center tree language (see Section 4).

2 Preliminaries

The nonnegative integers are $\mathbb{N}$. For every $k \in \mathbb{N}$, the set $\{i \in \mathbb{N} \mid 1 \leqslant i \leqslant k\}$ is denoted by $[k]$. Let R, S, and T be sets and $\rho \subseteq R \times S$ a relation. We occasionally write $r\ \rho\ s$ instead of $(r, s) \in \rho$. The *inverse* of ρ is $\rho^{-1} = \{(s, r) \mid r\ \rho\ s\}$ and the

reflexive and transitive closure of ϱ is denoted by ϱ^*. The *composition* of ρ with $\tau \subseteq S \times T$ is $\rho\,;\tau = \{(r,t) \mid \exists s \in S\colon r\ \rho\ s\ \tau\ t\}$. Finally, $|S|$ is the cardinality of the (finite) set S.

For a set V, we denote by V^* the set of all *strings* over V and by ε the *empty string*. An *alphabet* is a finite set (of symbols). A *ranked alphabet* (Σ, rk) is an alphabet Σ together with a mapping $\mathrm{rk}\colon \Sigma \to \mathbb{N}$. Often we leave rk implicit. For every $k \in \mathbb{N}$, let $\Sigma_k = \{f \in \Sigma \mid \mathrm{rk}(f) = k\}$.

Let Σ be a ranked alphabet and T a set. Then

$$\Sigma(T) = \{f(t_1, \ldots, t_k) \mid f \in \Sigma_k, t_1, \ldots, t_k \in T\}\ .$$

The set $T_\Sigma(V)$ of all *Σ-trees indexed by* variables V is the smallest set $\mathcal{T}$ such that $V \subseteq \mathcal{T}$ and $\Sigma(\mathcal{T}) \subseteq \mathcal{T}$. Subsets of $T_\Sigma(V)$ are *tree languages*. Such a tree language L is *variable-free* (respectively, *almost variable-free*) if $L \subseteq T_\Sigma$ (respectively, $L \subseteq T_\Sigma \cup V$). Generally, for all considered trees $t \in T_\Sigma(V)$ we assume that $\Sigma \cap V = \emptyset$, so that we can safely write c instead of $c()$ for every $c \in \Sigma_0$. For every tree $t \in T_\Sigma(V)$, the set $\mathrm{pos}(t) \subseteq \mathbb{N}^*$ of *positions* of t is inductively defined by $\mathrm{pos}(v) = \{\varepsilon\}$ for every $v \in V$, and

$$\mathrm{pos}(f(t_1, \ldots, t_k)) = \{\varepsilon\} \cup \{iw \mid i \in [k], w \in \mathrm{pos}(t_i)\}$$

for every $f \in \Sigma_k$ and $t_1, \ldots, t_k \in T_\Sigma(V)$. Let $w \in \mathrm{pos}(t)$. The *label* of t at w, the *subtree* of t at w, and the *replacement* of that subtree by $s \in T_\Sigma(V)$ are denoted by $t(w)$, by $t|_w$, and by $t[s]_w$, respectively.

A tree $t \in T_\Sigma(V)$ is *linear* (respectively, *nondeleting*) in $Y \subseteq V$ if every $y \in Y$ occurs at most (respectively, at least) once in t. Let $D \subseteq V \cup \Sigma_0$. The *$D$-yield* of t is defined inductively by $\mathrm{yd}_D(d) = d$ for every $d \in D$, $\mathrm{yd}_D(v) = \varepsilon$ for every $v \in V \setminus D$, and

$$\mathrm{yd}_D(f(t_1, \ldots, t_k)) = \mathrm{yd}_D(t_1) \cdots \mathrm{yd}_D(t_k)$$

for every $f \in \Sigma_k \setminus D$ and $t_1, \ldots, t_k \in T_\Sigma(V)$.

We fix a set $X = \{x_i \mid i \geqslant 1\}$ of *formal variables* (disjoint to all other ranked alphabets and variables considered). For every $n \in \mathbb{N}$, we let $X_n = \{x_i \mid i \in [n]\}$. For all $t, t_1, \ldots, t_n \in T_\Sigma(V \cup X_n)$, we denote by $t[t_1, \ldots, t_n]$ the result obtained by replacing, for every $i \in [n]$, every occurrence of x_i in t by t_i. For every $v \in V$, we denote by $t[v \leftarrow (t_1, \ldots, t_n)]$ the result of replacing, for every $i \in [n]$, the i-th (with respect to the lexicographic order on the positions) occurrence of v by t_i.

A *regular tree grammar* is a tuple $G = (N, \Sigma, V, P, S)$ consisting of

- an alphabet N of *nonterminal symbols* such that $N \cap (\Sigma \cup V) = \emptyset$,
- a finite set P of *productions* of the form $A \to r$, in which $A \in N$ and $r \in T_\Sigma(N \cup V)$, and
- a *start symbol* $S \in N$.

The size of G, denoted by $|G|$, is $|G| = |P|$. For any $s, t \in T_\Sigma(N \cup V)$, we write $s \Rightarrow_G t$ if there exists $A \to r \in P$ such that t can be obtained from s by replacing one occurrence of A by r. The tree language *generated* by G is

$L(G) = \{t \in T_\Sigma(V) \mid S \Rightarrow_G^* t\}$. A tree language L is *recognizable* if there exists a regular tree grammar G such that $L = L(G)$. The family of all recognizable (respectively, recognizable variable-free and recognizable almost variable-free) tree languages is denoted by Rec (respectively, $\mathrm{Rec}_{\mathrm{vf}}$ and $\mathrm{Rec}_{\mathrm{avf}}$).

A *tree homomorphism* $\varphi\colon T_\Sigma(V) \to T_\Delta(Y)$ can be presented by a mapping $\varphi_V\colon V \to T_\Delta(Y)$ and mappings $\varphi_k\colon \Sigma_k \to T_\Delta(Y \cup X_k)$ for every $k \in \mathbb{N}$ as follows:

- $v\varphi = \varphi_V(v)$ for every $v \in V$, and
- $f(t_1, \ldots, t_k)\varphi = \varphi_k(f)[t_1\varphi, \ldots, t_k\varphi]$ for every $t_1, \ldots, t_k \in T_\Sigma(V)$ and $f \in \Sigma_k$.

We say that it is *normalized* if for every $f \in \Sigma_k$ there exists $n \in \mathbb{N}$ such that $\mathrm{yd}_X(\varphi_k(f)) = x_1 \cdots x_n$. Moreover, such a homomorphism φ is

- *linear* [3,7,18] (respectively, *complete* [18]) if $\varphi_k(f)$ is linear (respectively, nondeleting) in X_k for every $f \in \Sigma_k$,
- *quasi-alphabetic* [15] if it is linear and complete, $\varphi_V(v) \in Y$ for every $v \in V$, and $\varphi_k(f) \in \Delta(Y \cup X_k)$ for every $f \in \Sigma_k$,
- *symbol-to-symbol* [18] if it is quasi-alphabetic and $\varphi_k(f) \in \Delta(X_k)$ for every $f \in \Sigma_k$, and
- *alphabetic* [3,18] if it is symbol-to-symbol and normalized.

Note that our 'symbol-to-symbol' corresponds to "linear, complete, and symbol-to-symbol" of [18], and 'alphabetic' homomorphisms are sometimes called relabelings [5]. We denote by qaH, ssH, and aH the classes of all quasi-alphabetic, symbol-to-symbol, and alphabetic tree homomorphisms, respectively.

A *tree bimorphism* is a triple $B = (\varphi, L, \psi)$ where $L \subseteq T_\Gamma(Z)$ is a tree language, $\varphi\colon T_\Gamma(Z) \to T_\Sigma(V)$ and $\psi\colon T_\Gamma(Z) \to T_\Delta(Y)$ are tree homomorphisms, called input and output homomorphism, respectively. The size of B, denoted by $|B|$, is defined to be the size of a representation (e.g., by a regular tree grammar) of L. The *tree transformation defined by* B is $\tau_B = \{(t\varphi, t\psi) \mid t \in L\}$. We reserve the special variable e. The *translation defined by* B is

$$\mathrm{yd}(\tau_B) = \{(\mathrm{yd}_{V\setminus\{e\}}(s), \mathrm{yd}_{Y\setminus\{e\}}(t)) \mid (s,t) \in \tau_B\} \ .$$

Note the special treatment of e. It is never output but acts as the empty string. For all classes $\mathcal{H}_1$ and $\mathcal{H}_2$ of tree homomorphisms and every class $\mathcal{L}$ of tree languages, we denote by $\mathcal{B}(\mathcal{H}_1, \mathcal{L}, \mathcal{H}_2)$ the class of tree transformations τ_B where $B = (\varphi, L, \psi)$ with $\varphi \in \mathcal{H}_1$, $L \in \mathcal{L}$, and $\psi \in \mathcal{H}_2$. In particular, we say that a tree bimorphism (φ, L, ψ) is *quasi-alphabetic* (respectively, *symbol-to-symbol*, *alphabetic*, and *normalized*) if both φ and ψ have this property and $L \in \mathrm{Rec}$. Moreover, a bimorphism (φ, L, ψ) is *variable-free* (respectively, *almost variable-free*) if L is so.

A system $M = (Q, \Sigma, \Delta, F, R)$ is a *bottom-up tree transducer* [19,5] if

- $Q = Q_1$ is a unary ranked alphabet of *states*,
- Σ and Δ are an *input* and an *output alphabet*, respectively,
- $F \subseteq Q$ is a set of *final states*, and

– R is a finite set of *rules* of the form $f(q_1(x_1), \ldots, q_k(x_k)) \to r$ where $f \in \Sigma_k$, $q_1, \ldots, q_k \in Q$, and $r \in Q(T_\Delta(X_k))$.

The bottom-up tree transducer $M = (Q, \Sigma, \Delta, F, R)$ is *linear* (respectively, *nondeleting*) if the right-hand side r is linear (respectively, nondeleting) in X_k for every $f(q_1(x_1), \ldots, q_k(x_k)) \to r \in R$. The one-step *derivation relation* $\Rightarrow_M$ is defined as follows. For every $\zeta, \xi \in T_\Sigma(Q(T_\Delta))$ we have $\zeta \Rightarrow_M \xi$ if and only if there exists a rule $f(q_1(x_1), \ldots, q_k(x_k)) \to r \in R$, a position $w \in \mathrm{pos}(\zeta)$, and $s_1, \ldots, s_k \in T_\Delta$ such that $\zeta|_w = f(q_1(s_1), \ldots, q_k(s_k))$ and $\xi = \zeta[s]_w$ with $s = r[s_1, \ldots, s_n]$. The *tree transformation computed by* M is

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

3 Syntax-directed Translation Schema

In this section, we explore the connection between quasi-alphabetic tree bimorphisms and syntax-directed translation schemata (SDTSs) [10,16,17]. It was shown in [15] that quasi-alphabetic tree bimorphisms and SDTSs are equally powerful when we consider them as translation devices for strings. This close connection is the main motivation for quasi-alphabetic tree bimorphisms [15]. Here we show that the mentioned connection already holds between SDTSs and symbol-to-symbol tree bimorphisms, a class that is smaller and well-known. Moreover, we show that simple SDTS correspond to alphabetic tree bimorphisms (also called finite-state relabelings [5]). The latter result proves that alphabetic tree bimorphisms are strictly less powerful than SDTSs.

Roughly speaking, a syntax-directed translation schema consists of 2 context-free grammars (CFGs) over a common set of nonterminals. A production of an SDTS is of the form $A \to u;w$ such that $A \to u$ and $A \to w$ are CFG productions, and additionally, the same nonterminals occur in u and w. Formally, a *syntax-directed translation schema* (SDTS) is a system $T = (N, V, Y, P, S)$ where

– N is an alphabet of *nonterminals* disjoint with $V \cup Y$,
– V and Y are an *input* and *output alphabet*, respectively,
– P is a finite set of *productions* of the form $A \to u \,;\, w$ where $A \in N$, $u \in (N \cup V)^*$, $w \in (N \cup Y)^*$, and the nonterminals in w are a permutation of the nonterminals in u, and
– $S \in N$ is a *start symbol*.

An SDTS is called *simple* if the nonterminals occur in same order in u and w for each production $A \to u \,;\, w$ in P. Finally, the size of T, denoted by $|T|$, is defined as the numbers of its productions (i.e., $|T| = |P|$).

To present the semantics of SDTS, we use the slightly informal notion of associated nonterminals. Whenever we apply a production in a derivation, we have to apply it to two "associated" nonterminals. This notion can easily be formalized, but we avoid this here to present the matter without excessive detail. The *translation forms* of T, which are elements of $(N \cup V)^* \times (N \cup Y)^*$, are defined inductively as follows:

- (S, S) is a translation form and the two nonterminals S are *associated*.
- If $(u_1 A u_2, w_1 A w_2)$ is a translation form in which the two explicit instances of A are associated and $A \to u\,;\, w$ is a production in P, then we have $(u_1 A u_2, w_1 A w_2) \Rightarrow_T (u_1 u u_2, w_1 w w_2)$ and the latter is a translation form. The nonterminals of u and w are associated exactly as they are associated in the production and the nonterminals of u_1 and u_2 are associated with those of w_1 and w_2 in the new translation form exactly as in the original one.

The *translation defined* by T is the relation

$$\tau_T = \{(u, w) \in V^* \times Y^* \mid (S, S) \Rightarrow_T^* (u, w)\} \ .$$

A major normal form for CFGs is the Chomsky normal form, but unfortunately its analogue cannot be achieved for SDTSs. However, [17] shows that we can obtain the following normal form. Note that we changed the definition slightly and demand that at most one terminal symbol occurs in each part of a production.

Definition 1. *An SDTS (N, V, Y, P, S) is in* normal form *if*

- $u, w \in N^*$ *or*
- $u \in V \cup \{\varepsilon\}$ *and* $w \in Y \cup \{\varepsilon\}$ *for every production* $A \to u\,;\, w$ *in* P.

Proposition 2 (cf. [17, Lemma 3.1]). *For every SDTS T there exists an SDTS T' in normal form such that $\tau_T = \tau_{T'}$. If T is simple, then T' can be chosen to be simple as well.*

Proof. Let $T = (N, V, Y, P, S)$ be an SDTS. We first construct the new SDTS $T' = (N', V, Y, P', S)$ where

- $N' = N \cup \{\underline{v} \mid v \in V\} \cup \{\overline{y} \mid y \in Y\}$ with $\underline{v}$ and $\overline{y}$ being new nonterminals,
- for every $v \in V$ and $y \in Y$ the following two rules are in P'

$$\underline{v} \to v\,;\, \varepsilon \qquad \text{and} \qquad \overline{y} \to \varepsilon\,;\, y \ ,$$

- and for every production of P with associated nonterminal permutation $\sigma\colon [n] \to [n]$

$$A \to u_0 A_1 u_1 \cdots A_n u_n\,;\, w_0 A_{\sigma(1)} w_1 \cdots A_{\sigma(n)} w_n$$

  where $u_0, \dots, u_n \in V^*$, $w_0, \dots, w_n \in Y^*$, and $A, A_1, \dots, A_n \in N$, the following production is in P'

$$A \to \underline{u_0}\,\overline{w_0} A_1 \underline{u_1}\,\overline{w_1} \cdots A_n \underline{u_n}\,\overline{w_n}\,;\, \underline{u_0}\,\overline{w_0} A_{\sigma(1)} \underline{u_1}\,\overline{w_1} \cdots A_{\sigma(n)} \underline{u_n}\,\overline{w_n}$$

  where for every $v_1, \dots, v_k \in V$ and $y_1, \dots, y_m \in Y$ we define

$$\underline{v_1 \cdots v_k} = \underline{v_1} \cdots \underline{v_k} \qquad \text{and} \qquad \overline{y_1 \cdots y_m} = \overline{y_1} \cdots \overline{y_m} \ .$$

- The set P' does not contain any further productions.

Obviously, T' is in normal form. Moreover, it is simple if T is so. Finally, it is easy to see that $\tau_{T'} = \tau_T$. □

Let us consider the complexity of the construction in the proof of Proposition 2. Clearly, the number of productions of T' is $|P| + |V| + |Y|$. Thus, the size of T' is $|T| + |V| + |Y|$. It is a reasonable assumption that for every $v \in V$ (respectively, $y \in Y$) there is at least one production in P in which v (respectively, y) occurs (otherwise we can simply drop the offending v or y). Consequently, $|V| + |Y| \leqslant 2|T|$ and $|T'| \in O(|T|)$, which proves that the size of T' is linear in the size of T.

Before we proceed with the mentioned connection between SDTSs and quasi-alphabetic tree bimorphisms, let us recall the well-known link between SDTSs and simple SDTSs.

Theorem 3 (see [16, Theorem 2]). *The class of all translations defined by simple SDTSs is properly contained in the class of all translations defined by SDTSs.*

In [15, Theorem 5.7] it was shown that SDTSs and quasi-alphabetic tree bimorphisms define the same (string) translations. The correspondence is very close since the derivations of an SDTS can be obtained from the tree transformation of the corresponding bimorphism. However, we will show that this correspondence already exists between SDTSs and symbol-to-symbol tree bimorphisms. Moreover, we will show that simple SDTSs and alphabetic tree bimorphisms define the same class of translations. Let us first consider the direction in which we construct a tree bimorphism for an SDTS. Since the only difference between quasi-alphabetic and symbol-to-symbol tree bimorphisms is in their homomorphisms, let us reconsider the construction of those homomorphisms from [15, Sect. 5]. We only change the behavior on productions that only have terminal symbols on the right-hand sides.

Definition 4. *Let $T = (N, V, Y, P, S)$ be an SDTS in normal form. For every production $p = (A \to u\,; w) \in P$ let $\mathrm{rk}(p) = n$ be such that $u \in N^n V^*$. This turns the set P into a ranked alphabet. Moreover, let $P' = \bigcup_{k \geqslant 1} P_k$. We construct the homomorphisms*

$$\varphi\colon T_{P'}(P_0) \to T_{P'}(V') \qquad \text{and} \qquad \psi\colon T_{P'}(P_0) \to T_{P'}(Y')$$

where $V' = V \cup \{e\}$ and $Y' = Y \cup \{e\}$ as follows: Let $p = (A \to u\,; w) \in P$.

- *If $p \in P_0$, then*

$$\varphi_{P_0}(p) = \begin{cases} e & \text{if } u = \varepsilon \\ u & \text{if } u \in V \end{cases} \qquad \text{and} \qquad \psi_{P_0}(p) = \begin{cases} e & \text{if } w = \varepsilon \\ w & \text{if } w \in V. \end{cases}$$

- *If $p \in P'_k$, then $\varphi_k(p) = p(x_1, \ldots, x_k)$ and $\psi_k(p) = p(x_{\sigma(1)}, \ldots, x_{\sigma(k)})$ where $\sigma\colon [k] \to [k]$ is the nonterminal permutation of p.*

By Proposition 2 we can assume normal form without loss of generality. Our construction is very similar to the construction of [15] if we restrict ourselves to SDTSs in normal form. The constructed homomorphisms φ and ψ are symbol-to-symbol, and if T is simple, then they are alphabetic. Thus, a minor modification of a principal result of [15, Sect. 5] yields our first result.

Lemma 5 (cf. [15, Prop. 5.5]). *For every SDTS T, there exists a symbol-to-symbol tree bimorphism B such that* $\mathrm{yd}(\tau_B) = \tau_T$. *If T is simple, then B can be chosen to be alphabetic.*

Let us consider the size of the resulting bimorphism. The construction in the proof of [15, Prop. 5.5] yields a local center tree language $L \subseteq T_P$ (the symbols are productions and their rank is determined by the number of nonterminals as in Definition 4). Roughly speaking, the language L contains all legal derivations (i.e., the nonterminals in productions match). For this tree language L we can construct the following regular tree grammar $G = (N, P, \emptyset, P', S)$, where for every production $p \in P_n$ (note that we assume that T is in normal form) with associated nonterminal permutation $\sigma\colon [n] \to [n]$ such that

$$p = A \to A_1 \cdots A_n v \,;\, A_{\sigma(1)} \cdots A_{\sigma(n)} y$$

for some $A, A_1, \ldots, A_n \in N$, $v \in V \cup \{\varepsilon\}$, and $y \in Y \cup \{\varepsilon\}$, the set P' contains the production $A \to p(A_1, \ldots, A_n)$. All productions of P' are constructed in this manner. Obviously, the size of G is the same as the size of T. Thus, the size of the bimorphism B constructed in Lemma 5 is linear in the size of the input SDTS T.

For the converse, we can again reconsider [15]. In [15, Prop. 5.6] it is proved that for every quasi-alphabetic tree bimorphism B there exists an SDTS T such that $\tau_T = \mathrm{yd}(\tau_B)$. Clearly, every symbol-to-symbol bimorphism is quasi-alphabetic, and moreover, it is an easy exercise to confirm that the SDTS constructed in [15, Prop. 5.6] is simple if B is alphabetic. Our minor modification of the definition of the translation defined by a tree bimorphism (the special treatment of the symbol e) requires only a minor change in the proof of [15, Prop. 5.6].

Lemma 6. *For every symbol-to-symbol tree bimorphism B, there is an SDTS T such that* $\tau_T = \mathrm{yd}(\tau_B)$. *If B is alphabetic, then T can be chosen to be simple.*

In the construction of [15, Prop. 5.6] the center tree language is represented as a local tree language, but in the same spirit the construction can be done if the center tree language is represented by a regular tree grammar. Every production of the tree grammar yields a production of the constructed SDTS. Thus, the size of the constructed SDTS is linear in the size of the input bimorphism (see, for example, [20, Theorem 4] on how to handle the regular tree grammar).

This yields the following relations between SDTSs and symbol-to-symbol tree bimorphisms. It was shown in [15] that the class of all translations defined by SDTSs coincides with the class of all translations defined by quasi-alphabetic tree bimorphisms. Here we sharpen this result.

Theorem 7. *The class of translations defined by arbitrary (respectively, simple) SDTSs coincides with the class of translations defined by symbol-to-symbol (respectively, alphabetic) tree bimorphisms.*

If we consider Theorems 3 and 7 together, we obtain that the class of all translations defined by alphabetic tree bimorphisms is properly contained in the class of all translations defined by symbol-to-symbol tree bimorphisms.

4 Closure under Composition

In this section we reconsider the problem of closure under composition for the class of tree transformations defined by quasi-alphabetic tree bimorphisms. It was shown in [15] that if we restrict ourselves to quasi-alphabetic tree bimorphisms with a variable-free center tree language, then the resulting class of tree transformations is closed under composition. Here we want to extend this result to include variables. The following proposition is trivial, but indicates why closure under composition is possible whereas closure under intersection fails [21].

Proposition 8. *For every quasi-alphabetic bimorphism B, there exist a quasi-alphabetic bimorphism B_1 with a normalized input homomorphism and a quasi-alphabetic bimorphism B_2 with a normalized output homomorphism such that $\tau_B = \tau_{B_1} = \tau_{B_2}$. If B is variable-free (almost variable-free, respectively), then B_1 and B_2 can be chosen such that they are variable-free (almost variable-free, respectively).*

So we showed that one homomorphism of a quasi-alphabetic bimorphism can always be normalized. As a final step we try to get rid of the variables as much as possible.

Lemma 9. $\mathcal{B}(\mathrm{qaH}, \mathrm{Rec}_{\mathrm{avf}}, \mathrm{qaH}) = \mathcal{B}(\mathrm{qaH}, \mathrm{Rec}, \mathrm{qaH})$

Proof. Let $B = (\varphi, L, \psi)$ be a quasi-alphabetic tree bimorphism with $L \subseteq T_\Gamma(Z)$. Moreover, let Y be a set, $h\colon Z \to Y$ be a bijection, and $M = (Q, \Gamma', \Omega', Q, R)$ be the linear bottom-up tree transducer with

- $Q = Y \cup \{\star\}$,
- $\Gamma'_k = \Gamma_k$ for every $k \geqslant 1$ and $\Gamma'_0 = \Gamma_0 \cup Z$,
- $\Omega'_k = \{t \in \Gamma(Q) \mid k = |t|_\star\}$ for every $k \geqslant 1$ and $\Omega'_0 = \Gamma_0 \cup Z$.
- The set R of rules is given as follows:
  - For every $z \in Z$, let $z \to q(z)$ be a rule of R where $q = h(z)$.
  - For every $f \in \Gamma_k$ and $q_1, \ldots, q_k \in Q$, the following is a rule of R

$$f(q_1(x_1), \ldots, q_k(x_k)) \to \star((f(q_1, \ldots, q_k))(x_{i_1}, \ldots, x_{i_n}))\ ,$$

    where $i_1 < \cdots < i_n$ and $\{i_1, \ldots, i_n\} = \{i \in [k] \mid q_i = \star\}$.

Let $L' = \tau_M(L)$ be the image of L under τ_M. Clearly, L' is almost variable-free, and by [3, Lemma IV.6.5], the tree language L' is recognizable. We construct the bimorphism $B' = (\varphi', L', \psi')$ such that $\varphi'_Z = \varphi_Z$, $\psi'_Z = \psi_Z$, and

$$\varphi'_k(t) = \varphi(t[\star \leftarrow (x_1, \dots, x_k)]) \quad \text{and} \quad \psi'_k(t) = \psi(t[\star \leftarrow (x_1, \dots, x_k)])$$

for every $t \in \Omega_k$. Clearly, φ' and ψ' are quasi-alphabetic. Thus, B' is an almost variable-free quasi-alphabetic bimorphism. Note that M is deterministic and total and thus τ_M is a mapping [5]. Finally, let us prove that $\tau_{B'} = \tau_B$. For this, we prove that $t\varphi = \tau_M(t)\varphi'$ and $t\psi = \tau_M(t)\psi'$ for every $t \in T_\Gamma(Z)$. Clearly, it is sufficient to prove the former statement since the argument is totally symmetric. First, let $t \in Z$. Then

$$t\varphi = \tau_M(t)\varphi = \tau_M(t)\varphi' \ .$$

Now, let $t = f(t_1, \dots, t_k)$ for some $f \in \Sigma_k$ and $t_1, \dots, t_k \in T_\Gamma(Z)$. Moreover, for every $i \in [k]$, let $q_i = h(t_i)$ if $t_i \in Z$ and $q_i = \star$ otherwise. Then

$$\begin{aligned}
\tau_M(f(t_1, \dots, t_k))\varphi' &= \omega(\tau_M(t_{i_1}), \dots, \tau_M(t_{i_n}))\varphi' \\
&= \varphi'_n(\omega)[\tau_M(t_{i_1})\varphi', \dots, \tau_M(t_{i_n})\varphi'] \\
&= \varphi(f(q_1, \dots, q_k)[\star \leftarrow (x_1, \dots, x_n)])[t_{i_1}\varphi, \dots, t_{i_n}\varphi] \\
&= f(q_1, \dots, q_k)[\star \leftarrow (t_{i_1}, \dots, t_{i_n})]\varphi \\
&= f(t_1, \dots, t_k)\varphi
\end{aligned}$$

with $\omega = f(q_1, \dots, q_k)$, $i_1 < \cdots < i_n$, and $\{i_1, \dots, i_n\} = \{i \in [k] \mid q_i = \star\}$. This completes the proof. □

It is proved in [15, Theorem 7.4] that $\mathcal{B}(\mathrm{qaH}, \mathrm{Rec}_{\mathrm{vf}}, \mathrm{qaH})$ is closed under composition. Let us take another look at composition closure results. First, we point out why it is far easier to prove the closure only for tree transformations defined by variable-free or almost variable-free quasi-alphabetic bimorphisms.

Lemma 10. *Let $\varphi\colon T_\Sigma(V) \to T_\Gamma(Z)$ and $\psi\colon T_\Delta(Y) \to T_\Gamma(Z)$ be normalized quasi-alphabetic tree homomorphisms, and let $s \in T_\Sigma \cup V$ and $t \in T_\Delta \cup Y$. If $s\varphi = t\psi$, then $\mathrm{pos}(s) = \mathrm{pos}(t)$.*

Proof. First, let $s \in V$. Then $s\varphi \in Z$. Since $s\varphi = t\psi$, it follows that $t \in Y$ and hence $\mathrm{pos}(s) = \mathrm{pos}(t)$. Second, let $s = f(s_1, \dots, s_k)$ for some $f \in \Sigma_k$ and $s_1, \dots, s_k \in T_\Sigma$. Then $s\varphi = \varphi_k(f)[s_1\varphi, \dots, s_k\varphi] = t\psi$. Since φ and ψ are quasi-alphabetic, we have $s_i\varphi \notin Z$ for every $i \in [k]$. If we additionally take into account that $s\varphi = t\psi$, then we can conclude that $t = g(t_1, \dots, t_k)$ for some $g \in \Delta_k$ and $t_1, \dots, t_k \in T_\Delta$. Moreover, since φ and ψ are normalized, it also follows that $\varphi_k(f) = \psi_k(g)$. Using the induction hypothesis, we thus obtain $\mathrm{pos}(s) = \mathrm{pos}(t)$. □

The previous proposition essentially states that all almost variable-free trees with the same image under two normalized quasi-alphabetic tree homomorphisms can be paired up in a product data structure $T_{\Sigma\times\Delta}(V \times Y)$. Let us plug the statements together and establish the relation to closure under composition.

Lemma 11. $\mathcal{B}(\mathrm{qaH}, \mathrm{Rec}, \mathrm{qaH})$ *is closed under composition if*

$$\{t \in T_\Omega \cup V \mid t\varphi = t\psi\} \tag{$\dagger$}$$

is a recognizable tree language for every ranked alphabet Ω, set V of variables, and pair (φ, ψ) of normalized quasi-alphabetic tree homomorphisms.

Proof. Let $B_1 = (\varphi_1, L_1, \psi_1)$ and $B_2 = (\varphi_2, L_2, \psi_2)$ be quasi-alphabetic bimorphisms. Without loss of generality, let B_1 and B_2 be almost variable-free by Lemma 9. Moreover, suppose that ψ_1 and φ_2 are normalized by Proposition 8. Let

$$\begin{aligned}\tau &= \tau_{B_1} ; \tau_{B_2} \\ &= \{(s,r) \mid \exists t\colon (s,t) \in \tau_{B_1}, (t,r) \in \tau_{B_2}\} \\ &= \{(t\varphi_1, r\psi_2) \mid t \in L_1, r \in L_2, t\psi_1 = r\varphi_2\} \ .\end{aligned}$$

Since $t\psi_1 = r\varphi_2$, it follows by Lemma 10 that $\mathrm{pos}(t) = \mathrm{pos}(r)$. Hence the quantified t and r in the last displayed equation can be stored in a single tree $s \in T_{\Sigma\times\Delta} \cup (V \times Y)$ such that $s\pi_1 = t$ and $s\pi_2 = r$ where π_1 and π_2 are the usual projections to the first and second component.

Let $T = T_{\Sigma\times\Delta} \cup (V \times Y)$. We can continue the displayed equations by

$$\begin{aligned}\tau &= \{(t\pi_1\varphi_1, t\pi_2\psi_2) \mid t \in T, t\pi_1 \in L_1, t\pi_2 \in L_2, t\pi_1\psi_1 = t\pi_2\varphi_2\} \\ &= \{(t\pi_1\varphi_1, t\pi_2\psi_2) \mid t \in \pi_1^{-1}(L_1) \cap \pi_2^{-1}(L_2) \cap L\}\end{aligned}$$

where $L = \{t \in T \mid t\pi_1\psi_1 = t\pi_2\varphi_2\}$. It is easily seen that the tree homomorphisms $\pi_1\varphi_1$, $\pi_2\psi_2$, $\pi_1\psi_1$, and $\pi_2\varphi_2$ are quasi-alphabetic. Moreover, $\pi_1\psi_1$, and $\pi_2\varphi_2$ are normalized. Consequently, L, $\pi_1^{-1}(L_1)$, and $\pi_2^{-1}(L_2)$ are recognizable by [3, Theorem II.4.18]. Thus, $\pi_1^{-1}(L_1) \cap \pi_2^{-1}(L_2) \cap L$ is recognizable by [3, Theorem II.4.2], and hence, $\tau \in \mathcal{B}(\mathrm{qaH}, \mathrm{Rec}, \mathrm{qaH})$, which completes the proof. □

So whenever the equality sets [the sets ($\dagger$) in Lemma 11] are recognizable, we can construct a quasi-alphabetic bimorphism that computes the composition of two given quasi-alphabetic bimorphisms. It remains to prove that the equality sets are recognizable (the premise of Lemma 11).

Lemma 12. *Let $\varphi\colon T_\Omega(Z) \to T_\Sigma(V)$ and $\psi\colon T_\Omega(Z) \to T_\Sigma(V)$ be normalized quasi-alphabetic tree homorphisms. Then $L = \{t \in T_\Omega \cup Z \mid t\varphi = t\psi\}$ is recognizable.*

Proof. We construct the regular tree grammar $G = (\{S\}, \Omega', P, S)$ where

- $\Omega'_k = \Omega_k$ for every $k \geqslant 1$ and $\Omega'_0 = \Omega_0 \cup Z$, and
- $P = P_1 \cup P_2$ with

$$\begin{aligned}P_1 &= \{S \to z \mid z \in Z, z\varphi = z\psi\} \\ P_2 &= \{S \to f(S, \ldots, S) \mid f \in \Omega_k, \varphi_k(f) = \psi_k(f)\} \ .\end{aligned}$$

Then $L = L(G) \cap (T_\Omega \cup Z)$, which is recognizable [3, Theorem II.4.2]. □

We are now ready to state our main result. Let Loc (respectively, $\mathrm{Loc}_{\mathrm{vf}}$) be the class of all local (respectively, local variable-free) tree languages [3]. Note that [15, Theorem 7.4] proves that $\mathcal{B}(\mathrm{qaH}, \mathrm{Loc}_{\mathrm{vf}}, \mathrm{qaH})$ is closed under composition. Since every recognizable tree language is the image of a local tree language under an alphabetic tree homomorphism [3, Theorem II.9.5], we immediately obtain

$$\mathcal{B}(\mathrm{qaH}, \mathrm{Loc}_{\mathrm{vf}}, \mathrm{qaH}) = \mathcal{B}(\mathrm{qaH}, \mathrm{Rec}_{\mathrm{vf}}, \mathrm{qaH})$$
$$\mathcal{B}(\mathrm{qaH}, \mathrm{Loc}, \mathrm{qaH}) = \mathcal{B}(\mathrm{qaH}, \mathrm{Rec}, \mathrm{qaH}) \ .$$

The closure of $\mathcal{B}(\mathrm{qaH}, \mathrm{Rec}_{\mathrm{vf}}, \mathrm{qaH})$ is thus proved in [15] and here we prove it for $\mathcal{B}(\mathrm{qaH}, \mathrm{Rec}, \mathrm{qaH})$. Note that our approach is slightly different.

Theorem 13 (cf. [15, Theorem 7.4]). $\mathcal{B}(\mathrm{qaH}, \mathrm{Rec}, \mathrm{qaH})$ *is closed under composition.*

Proof. Follows directly from Lemmata 11 and 12. □

Acknowledgements

The authors are grateful to Magnus Steinby for fruitful discussions regarding Section 3. In addition, the authors appreciate the effort of the reviewers, who pointed out several inconsistencies and mistakes.

References

1. Knight, K., Graehl, J.: An overview of probabilistic tree transducers for natural language processing. In: Proc. CICLing. Volume 3406 of LNCS., Springer (2005) 1–24
2. Knight, K.: Capturing practical natural language transformations. Machine Translation **21**(2) (2007) 121–133
3. Gécseg, F., Steinby, M.: Tree Automata. Akadémiai Kiadó, Budapest (1984)
4. Nivat, M., Podelski, A., eds.: Tree Automata and Languages. North-Holland (1992)
5. Engelfriet, J.: Bottom-up and top-down tree transformations: A comparison. Math. Syst. Theory **9**(3) (1975) 198–231
6. Arnold, A., Dauchet, M.: Morphismes et bimorphismes d'arbres. Theor. Comput. Sci. **20**(1) (1982) 33–93
7. Bozapalidis, S.: Alphabetic tree relations. Theor. Comput. Sci. **99**(2) (1992) 177–211
8. Takahashi, M.: Primitive transformations of regular sets and recognizable sets. In: Proc. ICALP, North-Holland (1972) 475–480
9. Steinby, M.: On certain algebraically defined tree transformations. In: Proc. Algebra, Combinatorics and Logic in Computer Science. Volume 42 of Colloquia Mathematica Societatis János Bolyai., North-Holland (1986) 745–764
10. Irons, E.T.: A syntax directed compiler for ALGOL 60. Comm. ACM **4**(1) (1961) 51–55

11. Shieber, S.M., Schabes, Y.: Synchronous tree-adjoining grammars. In: Proc. COLING, ACL (1990) 253–258
12. Satta, G., Peserico, E.: Some computational complexity results for synchronous context-free grammars. In: Proc. HLT/EMNLP, ACL (2005) 803–810
13. Shieber, S.M.: Synchronous grammars as tree transducers. In: Proc. TAG+7. (2004) 88–95
14. Aho, A.V., Ullman, J.D.: Parsing. Volume 1 of The Theory of Parsing, Translation, and Compiling. Prentice Hall (1972)
15. Steinby, M., Tîrnăucă, C.I.: Defining syntax-directed translations by tree bimorphisms. Theor. Comput. Sci. (2009) to appear. http://dx.doi.org/10.1016/j.tcs.2009.03.009.
16. Aho, A.V., Ullman, J.D.: Properties of syntax directed translations. J. Comput. Syst. Sci. **3**(3) (1969) 319–334
17. Aho, A.V., Ullman, J.D.: Syntax directed translations and the pushdown assembler. J. Comput. Syst. Sci. **3**(1) (1969) 37–56
18. Comon-Lundh, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Löding, C., Tison, S., Tommasi, M.: Tree automata—techniques and applications. available at: http://tata.gforge.inria.fr/ (2007)
19. Thatcher, J.W.: Tree automata: An informal survey. In: Currents in the Theory of Computing. Prentice Hall (1973) 143–172
20. Maletti, A.: Compositions of extended top-down tree transducers. Inf. Comput. **206**(9–10) (2008) 1187–1196
21. Maletti, A., Tîrnăucă, C.I.: Properties of quasi-alphabetic tree bimorphisms. unpublished, available at: http://arxiv.org/abs/0906.2369v1 (2009)