On Tree Substitution Grammars

Andreas Maletti and Kevin Stier(✉)

Faculty of Mathematics and Computer Science, Universität Leipzig,
PO Box 100 920, 04009 Leipzig, Germany
{maletti,stier}@informatik.uni-leipzig.de

Abstract. Tree substitution grammars are formal models that are used extensively in natural language processing. It is demonstrated that their expressive power is located strictly between the local tree grammars and the regular tree grammars. A decision procedure for the problem of determining whether a tree substitution grammar generates a local tree language is provided. Unfortunately, the class of tree substitution languages is neither closed under union, nor intersection, nor complements. Indeed unions of tree substitution languages even generate an infinite hierarchy. However, all finite and all co-finite tree languages are tree substitution languages.

1 Introduction

Trees are a fundamental data structure in computer science and are used in many application areas like natural language processing [12], database theory [1], and compiler construction [17]. All the mentioned applications as well as others [6,7] require effective representations of sets of trees, also called tree languages. These requirements triggered detailed investigations of various classes of tree languages since the 1960s and by now there exists an abundance of models [5].

The most robust of those classes of tree languages are the regular tree languages [6,7], which are generated by finite-state tree automata, which are a natural extension of the finite-state string automata that generate the regular string languages [18]. Most standard problems are decidable for the regular tree languages and they generally enjoy the same nice algorithmic properties as the regular string languages. The main feature of those automata are their finitely many states, which enable most of the positive properties. However, these states are not exhibited directly in the trees generated. In application areas like natural language processing, in which representations of tree languages have to be inferred from finite sets of trees, practitioners often resorted to simpler models, in which the representation can more readily be induced from the sample.

Tree substitution grammars were originally introduced as a special case of tree-adjoining grammars [9,11], in which no adjunction is allowed. This restriction proved useful in the lexicalization of context-free grammars [10]. However, tree substitution grammar soon became popular in the parsing community [15]

K. Stier—Supported by DFG Research Training Group 1763 (QuantLA).

N. Jonoska and D. Savchuk (Eds.): DLT 2020, LNCS 12086, pp. 237–250, 2020.
https://doi.org/10.1007/978-3-030-48516-0_18

under the approach called data-oriented parsing [3] and were the formal model of many state-of-the-art parsers [16]. Similarly, synchronous tree substitution grammars, which are the same as the syntax-directed translation schemes of [2], are used in many statistical machine translation models [4,8,13,14]. Despite the multitude of applications, a fundamental study of their expressive power is missing. Rather they are attributed properties like "extended domain of locality", which provides some intuition, but has no formal definition.

A tree substitution grammar G is essentially a finite set F of tree fragments together with a set R of permissible root labels. Those tree fragments can be arbitrarily tall or large, which distinguishes tree substitution grammars from local tree grammars [6,7]. In addition, the fragments can contain leaves that are labeled by internal symbols. Leaves with such labels are called open and can be expanded further by fragments of F that have the same symbol as root label. Indeed G generates trees from a permissible root label of R by successively expanding open leaves with fragments of F until no open leaves remain. The set of all trees derivable in this manner is called the tree language generated by G. The tree languages that can be generated by some tree substitution grammar are called the tree substitution languages.

In this contribution we start a fundamental study of the expressive power of tree substitution grammars. We show that tree substitution grammars are strictly more expressive than local tree grammars [6,7], but strictly less expressive than finite-state tree automata (see Corollary 10). This, in particular, yields that most standard decision problems are also decidable for tree substitution languages because they are regular. In addition, it is decidable to determine whether a given tree substitution language is local (see Theorem 8). The decidability status of the related question whether a given regular tree language is a tree substitution language remains open. It is interesting to note that all finite and co-finite tree languages are tree substitution languages (see Theorem 6), which makes them much more useful for the approximation of finite samples of trees than the local tree languages, which do not contain all finite tree languages.

We also investigate the closure properties of the tree substitution languages. Unfortunately, they are neither closed under union (see Theorem 9), nor under intersection (see Theorem 13), nor under complement (see Theorem 14). In fact, unions of tree substitution languages even form a strict hierarchy (see Theorem 11), so unions of k tree substitution languages are strictly less expressive than unions of $k + 1$ tree substitution languages. A similar hierarchy is significantly more difficult to prove for intersections and remains an open problem because intersections break the "extended domain of locality" (as shown in the proof of Theorem 13) and can manage a non-explicit information transport over unbounded distances in the trees. Indeed the trivial union construction, which just takes the union of the fragments of the individual tree substitution grammars $G_1, \ldots, G_n$, does yield a tree substitution grammar G that can generate each tree that can be generated by some G_i. However, G might overgeneralize in the sense that it may also generate trees that cannot be generated by any $G_1, \ldots, G_n$. This property is utilized in grammar induction to generalize

beyond the seen data. Overall, the expressive power of tree substitution grammars is interesting and offers new challenging problems because they are used extensively in real-world applications despite their brittle expressive power. It is exactly this absence of good closure properties, which requires separate arguments for each individual problem and thus makes several problems challenging as outlined in the open problems section.

2 Preliminaries

We denote the set of nonnegative integers (including 0) by $\mathbb{N}$. For every $k \in \mathbb{N}$, we use the subset $[k] = \{i \in \mathbb{N} \mid 1 \leq i \leq k\}$. An alphabet A is simply a finite set and $A^* = \bigcup_{k \in \mathbb{N}} A^k$ is the set of all finite words over A, where $A^k = A \times \cdots \times A$ containing k factors A and $A^0 = \{\varepsilon\}$, of which ε is called the empty word. The length $|w|$ of a word $w = a_1 \cdots a_k \in A^*$ with $a_1, \ldots, a_k \in A$ is $|w| = k$; i.e. the number of symbols making up w. Given words $v, w \in A^*$, their concatenation is written $v.w$ or simply vw. We write $v \preceq w$ provided that there exists $u \in A^*$ such that $vu = w$. The relation $\preceq$ is actually a partial order, called the prefix order.

Let S be a set and $R \subseteq S \times S$ be a relation. The identity on S is the relation $\mathrm{id}_S = \{(s, s) \mid s \in S\}$. Given another relation $R' \subseteq S \times S$, the composition $R\,;R'$ is given by $R\,;R' = \{(s_1, s_3) \mid \exists s_2 \in S\colon (s_1, s_2) \in R, (s_2, s_3) \in R'\}$. The relation R is reflexive if $\mathrm{id}_S \subseteq R$, and it is transitive if $R\,;R \subseteq R$. The reflexive, transitive closure of R is $R^* = \bigcup_{k \in \mathbb{N}} R^k$ and the transitive closure of R is $R^+ = \bigcup_{k \geq 1} R^k$, where $R^0 = \mathrm{id}_S$ and $R^k = R\,;\cdots;R$ containing k times the relation R.

A ranked alphabet (Σ, rk) is a pair consisting of an alphabet Σ and a mapping $\mathrm{rk}\colon \Sigma \to \mathbb{N}$ that assigns a rank to each symbol of Σ. We usually denote a ranked alphabet (Σ, rk) by just Σ alone when the ranks are clear. We also write $\sigma^{(k)}$ to indicate that $\mathrm{rk}(\sigma) = k$. Moreover, for every $k \in \mathbb{N}$, we let $\Sigma_k = \{\sigma \in \Sigma \mid \mathrm{rk}(\sigma) = k\}$. Given a ranked alphabet Σ and a set Z, the set $T_\Sigma(Z)$ of Σ" trees indexed by Z is the smallest set T such that $Z \subseteq T$ and $\sigma(t_1, \ldots, t_k) \in T$ for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, and $t_1, \ldots, t_k \in T$. We abbreviate $T_\Sigma(\emptyset)$ simply to T_Σ, and any subset $L \subseteq T_\Sigma$ is called tree language. It is co-finite if $T_\Sigma \setminus L$ is finite.

Next, we recall some common notions and notations for trees. In the following, let $t \in T_\Sigma(Z)$ be a tree for a ranked alphabet Σ and a set Z. The set $\mathrm{pos}(t)$ of positions of t is inductively defined by $\mathrm{pos}(z) = \{\varepsilon\}$ for all $z \in Z$, and $\mathrm{pos}(\sigma(t_1, \ldots, t_k)) = \{\varepsilon\} \cup \{i.p \mid i \in [k], p \in \mathrm{pos}(t_i)\}$ for every $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, and $t_1, \ldots, t_k \in T_\Sigma(Z)$. The height of t is defined by $\mathrm{ht}(t) = \max_{p \in \mathrm{pos}(t)} |p|$, and the size of t is defined by $|t| = |\mathrm{pos}(t)|$. A leaf is a position $p \in \mathrm{pos}(t)$ such that $p.1 \notin \mathrm{pos}(t)$. We denote the subset of leaves of $\mathrm{pos}(t)$ by $\mathrm{leaf}(t)$. Given a position $p \in \mathrm{pos}(t)$, the label $t(p)$ of t at p and the subtree $t|_p$ of t at p are defined by $z(\varepsilon) = z|_\varepsilon = z$ for all $z \in Z$, and

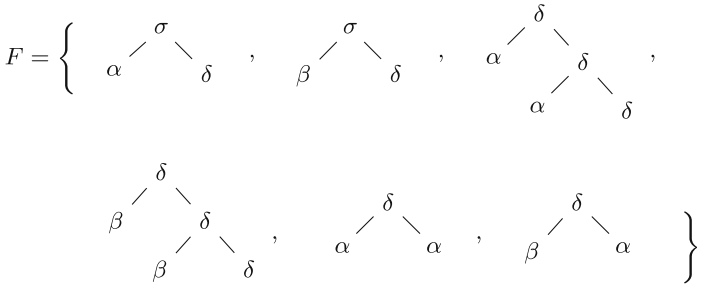

Fig. 1. Fragments of the TSG of Example 2.

$$\big(\sigma(t_1,\dots,t_k)\big)(p) = \begin{cases} \sigma & \text{if } p = \varepsilon \\ t_i(p') & \text{if } p = i.p' \text{ with } i \in \mathbb{N} \text{ and } p' \in \operatorname{pos}(t_i) \end{cases}$$

$$\sigma(t_1,\dots,t_k)|_p = \begin{cases} \sigma(t_1,\dots,t_k) & \text{if } p = \varepsilon \\ t_i|_{p'} & \text{if } p = i.p' \text{ with } i \in \mathbb{N} \text{ and } p' \in \operatorname{pos}(t_i) \end{cases}$$

for all $k \in \mathbb{N}$, $\sigma \in \Sigma_k$, and $t_1,\dots,t_k \in T_\Sigma(Z)$. Finally, the replacement $t[u]_p$ of the leaf $p \in \operatorname{leaf}(t)$ by another tree $u \in T_\Sigma(Z)$ is given by $\alpha[u]_\varepsilon = u$ for every $\alpha \in Z \cup \Sigma_0$, and $\sigma(t_1,\dots,t_k)[u]_{i.p'} = \sigma(t_1,\dots,t_{i-1},t_i[u]_{p'},t_{i+1},\dots,t_k)$ for every $k \in \mathbb{N}$, $i \in [k]$, $\sigma \in \Sigma_k$, $t_1,\dots,t_k \in T_\Sigma(Z)$, and $p' \in \operatorname{pos}(t_i)$.

We reserve the use of the special symbol $\Box$. A tree $t \in T_\Sigma(Z \cup \{\Box\})$ is a context, if there exists exactly one $p \in \operatorname{pos}(t)$ with $t(p) = \Box$; i.e., there is exactly one occurrence of $\Box$ in t. The set of all such contexts is denoted by $C_\Sigma(Z)$. Given a context $c \in C_\Sigma(Z)$ and a tree $t \in T_\Sigma(Z \cup \{\Box\})$, the substitution $c[t]$ of t into c yields the tree $c[t]_p$, where p is the unique position $p \in \operatorname{pos}(c)$ with $c(p) = \Box$. Note that given $c, c' \in C_\Sigma(Z)$, also $c[c'] \in C_\Sigma(Z)$. Similarly, we write $c^k[t]$ for $c[c[\cdots c[t]\cdots]]$ containing the context c a total of k times.

Finally, let us recall regular tree grammars (RTGs) [6,7]. An RTG is a tuple $G = (Q, \Sigma, Q_0, P)$, where Q is a finite set of states such that $Q \cap \Sigma = \emptyset$, Σ is a ranked alphabet of input symbols, $Q_0 \subseteq Q$ is a set of initial states, and $P \subseteq Q \times T_\Sigma(Q)$ is a finite set of productions. We also write productions (q,t) as $q \to t$. The derivation relation for $\xi, \zeta \in T_\Sigma(Q)$ is defined for every $\xi, \zeta \in T_\Sigma(Q)$ by $\xi \Rightarrow_G \zeta$ if and only if there exists a production $q \to t \in P$ and a context $c \in C_\Sigma(Q)$ such that $\xi = c[q]$ and $\zeta = c[t]$. The tree language generated by G is $L(G) = \bigcup_{q \in Q_0} \{t \in T_\Sigma \mid q \Rightarrow_G^+ t\}$. A tree language L is regular if there exists an RTG G such that $L(G) = L$. The class of regular tree languages is denoted by RTL. We note that RTL coincides with the class of tree languages generated by tree automata [6,7].

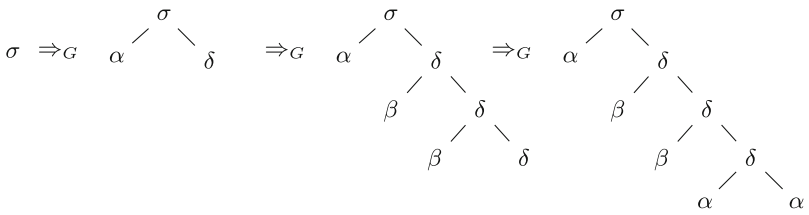

Fig. 2. Example derivation steps using the TSG G of Example 2.

3 Tree Substitution Grammars

Let us start with the formal definition of tree substitution grammars (TSGs) taken essentially from the natural language processing community [10,11]. TSGs have been applied to various tasks including parsing [16] and machine translation [19]. Consequently, the definitions of TSGs vary, but our definition captures the essence of the notion, while still being convenient to work with.

Definition 1. *A* tree substitution grammar *(TSG) is a tuple* $G = (\Sigma, R, F)$*, in which* Σ *is a ranked alphabet of input symbols,* $R \subseteq \Sigma$ *is a set of* root labels*, and* $F \subseteq T_\Sigma(\Sigma) \setminus \Sigma$ *is a finite set of* fragments*. The TSG* G *is a* local tree grammar *(LTG) if* $\mathrm{ht}(f) \leq 1$ *for all* $f \in F$*.*

Example 2. Consider the ranked alphabet $\Sigma = \{\sigma^{(2)}, \delta^{(2)}, \alpha^{(0)}, \beta^{(0)}\}$ and the TSG $G = (\Sigma, \{\sigma\}, F)$ with the fragments displayed in Fig. 1. Clearly, this TSG is not an LTG due to the third and fourth fragment.

Next we present the derivation semantics for a TSG $G = (\Sigma, R, F)$. Essentially we start the derivation process with a tree consisting solely of a root label of R and then iteratively replace a leaf by a fragment of F with the same root label. This process can be repeated until no replacements are possible anymore. If the such obtained tree t contains only leaves that are labeled by nullary symbols, then t is part of the tree language generated by G.

Definition 3. *Let* $G = (\Sigma, R, F)$ *be a TSG. For any two trees* $\xi, \zeta \in T_\Sigma(\Sigma)$*, we write* $\xi \Rightarrow_G \zeta$ *if there exists a fragment* $f \in F$ *and a context* $c \in C_\Sigma(\Sigma)$ *such that* $\xi = c[f(\varepsilon)]$ *and* $\zeta = c[f]$*. The TSG* G *generates the tree language* $L(G) = \{t \in T_\Sigma \mid \exists \sigma \in R\colon \sigma \Rightarrow_G^* t\}$*.*

Example 4. Let $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}\}$ and consider the TSG $G = (\Sigma, \{\sigma\}, F)$ with the fragments displayed in Fig. 3. The derivation presented in Fig. 3 illustrates that a derived tree can contain several leaves that still need to be independently replaced. More precisely, both occurrences of γ in the tree $\sigma(\gamma, \gamma)$ are independently replaced in the displayed derivation.

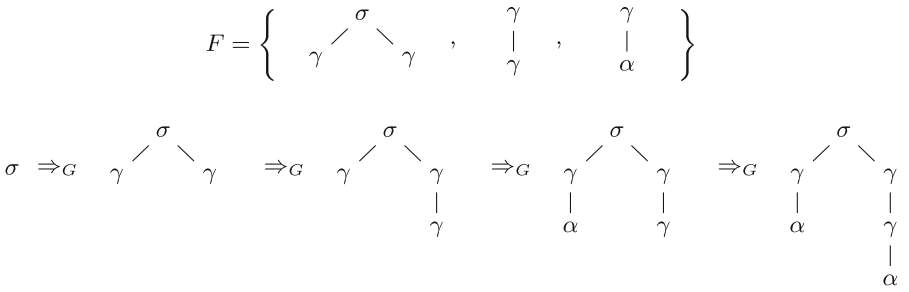

Fig. 3. Fragments of the TSG G of Example 4 and example derivation steps.

Example 5. Consider the TSG G from Example 2. A few derivation steps are displayed in Fig. 2. Let $c_\alpha = \delta(\alpha, \delta(\alpha, \Box))$ and $c_\beta = \delta(\beta, \delta(\beta, \Box))$. Overall, this TSG generates the tree language

$$\big\{\sigma(x, c_1[\cdots c_n[\delta(y, \alpha)]\cdots]) \mid x, y \in \{\alpha, \beta\},\, n \in \mathbb{N},\, \forall i \in [n] \colon c_i \in \{c_\alpha, c_\beta\}\big\}.$$

Two TSGs G and G' are *equivalent* if $L(G) = L(G')$. A tree language L is a *tree substitution language* if there exists a TSG G such that $L = L(G)$, and it is *local* [6,7] if there exists a local tree grammar G such that $L = L(G)$. The classes of all tree substitution languages and all local tree languages are denoted by TSL and LTL, respectively.

4 Expressive Power

In this section, we investigate the expressive power of tree substitution grammars and start with some simple tree languages that are contained in TSL. To this end, let FIN and co-FIN be the classes of all finite and all co-finite tree languages, respectively.

Theorem 6. FIN $\cup$ co-FIN $\subseteq$ TSL.

Proof. Every finite tree language $L \subseteq T_\Sigma$ is trivially a tree substitution language via the TSG (Σ, R, L) with $R = \{t(\varepsilon) \mid t \in L\}$.

Now, let $L \subseteq T_\Sigma$ be a co-finite tree language and $T_\Sigma \setminus L = \{t_1, \ldots, t_k\}$ be the finitely many trees outside L. Moreover, let $n > \max_{i \in [k]} \mathrm{ht}(t_i)$ be larger than the height of the tallest tree from $\{t_1, \ldots, t_k\}$. We construct the TSG (Σ, R, F) with

- $R = \{t(\varepsilon) \mid t \in L\}$ and
- $F = \{t \in L \mid \mathrm{ht}(t) \leq 2n\} \cup \{t \in T_\Sigma(\Sigma) \mid n \leq \mathrm{ht}(t) \leq 2n\}$.

Clearly, F is finite. Now we prove $L(G) = L$. For $L(G) \subseteq L$ it is sufficient to show that $t_i \notin L(G)$ for every $i \in [k]$. Obviously, the fragments of F are either in L or have height at least n, which proves $L(G) \subseteq L$. We prove the converse

$L \subseteq L(G)$ by contradiction, so suppose that there exists $t \in L$ with $t \notin L(G)$. Then there also exists a smallest $t' \in L$ with $t' \notin L(G)$. Since all trees $t' \in L$ with $\mathrm{ht}(t') \leq 2n$ can be generated directly using a single fragment from F, we must have $\mathrm{ht}(t') > 2n$. Let

$$P = \{p \in \mathrm{pos}(t') \mid |p| \leq n, \exists p' \in \mathrm{pos}(t') \colon p \preceq p', |p'| > 2n\}$$

be the short positions that are prefixes to long positions, and let $C = \max_{\preceq} P$ be the maximal (with respect to $\preceq$) elements of P. We construct the unique tree $f \in T_\Sigma(\Sigma)$ with positions

$$\mathrm{pos}(f) = \{p \in \mathrm{pos}(t') \mid |p| \leq 2n\} \setminus \{p \in \mathrm{pos}(t') \mid \exists c \in C \colon c \prec p\}$$

and labels $f(p) = t'(p)$ for all $p \in \mathrm{pos}(f)$. In other words, we obtain f by cutting all paths in t' that have length more than $2n$ at length n. Obviously, $f \in F$. In addition, we observe that $\mathrm{ht}(t'|_p) > n$ for all $p \in C$. For every $p \in C$, we thus obtain $t'|_p \in L$ and $t'|_p \in L(G)$ since $|t'|_p| < |t'|$ and t' is the smallest counterexample. However, this yields that $t'(\varepsilon) \Rightarrow_G f$ as well as $f(p) \Rightarrow_G^* t'|_p$ for all $p \in C$. Altogether $t'(\varepsilon) \Rightarrow_G^* t'$, which proves that $t' \in L(G)$ contradicting the assumption. □

Next we relate the class of tree substitution languages to the well-known classes of local and regular tree languages, respectively. Unsurprisingly, they are situated strictly between them, but the second strictness will be established later (see Corollary 10).

Theorem 7. $\mathrm{LTL} \subsetneq \mathrm{TSL} \subseteq \mathrm{RTL}$.

Proof. The first inclusion holds by definition. For the latter, let $G = (\Sigma, R, F)$ be a TSG and $S \notin \Sigma$ a new symbol. We construct an RTG $G' = (\overline{\Sigma} \cup \{S\}, \Sigma, S, P)$ such that $L(G') = L(G)$. To this end, we use copies $\overline{\Sigma} = \{\overline{\sigma} \mid \sigma \in \Sigma\}$ of the input symbols of Σ as states. The productions are given by $P = P_S \cup P'$ with

$$P_S = \{S \to \mathrm{rel}(\sigma) \mid \sigma \in R\}$$
$$P' = \{\overline{f(\varepsilon)} \to \mathrm{rel}(f) \mid f \in F\},$$

where $\mathrm{rel} \colon T_\Sigma(\Sigma) \to T_\Sigma(\overline{\Sigma})$ is inductively defined by

$$\mathrm{rel}(\sigma) = \begin{cases} \sigma & \text{if } \sigma \in \Sigma_0 \\ \overline{\sigma} & \text{otherwise} \end{cases}$$

for every $\sigma \in \Sigma$ and $\mathrm{rel}(\sigma(t_1, \ldots, t_k)) = \sigma(\mathrm{rel}(t_1), \ldots, \mathrm{rel}(t_k))$ for all $k \in \mathbb{N} \setminus \{0\}$, $\sigma \in \Sigma_k$, and $t_1, \ldots, t_k \in T_\Sigma(\Sigma)$. Clearly any derivation $\xi_0 \Rightarrow_G \xi_1 \Rightarrow_G \cdots \Rightarrow_G \xi_n$ of G yields a corresponding derivation $\mathrm{rel}(\xi_0) \Rightarrow_{G'} \mathrm{rel}(\xi_1) \Rightarrow_{G'} \cdots \Rightarrow_{G'} \mathrm{rel}(\xi_n)$ of G'. Together with $\mathrm{rel}(t) = t$ for all $t \in T_\Sigma$ and the new initial states, we obtain $L(G) \subseteq L(G')$. The converse is proved similarly.

The first inclusion is strict because $\mathrm{FIN} \subseteq \mathrm{TSL}$ by Theorem 6, but it is well-known [6,7] that $\mathrm{FIN} \not\subseteq \mathrm{LTL}$. □

Fig. 4. The tree languages $L(G_1)$ and $L(G_2)$ used in the proof of Theorem 9.

The inclusion TSL $\subseteq$ RTL immediately yields that most interesting problems are decidable for tree substitution languages. For example, the emptiness, finiteness, inclusion, and equivalence problems are all decidable because they are decidable for regular tree languages [6,7]. We proceed with a subclass definability problem: Is it decidable whether an effectively presented tree substitution language is local? Whenever we speak about an effectively presented tree substitution language L, we assume that we are actually given a tree substitution grammar G such that $L(G) = L$. Let $G = (\Sigma, R, F)$ be a TSG. A fragment $f \in F$ is *useless* if G and $(\Sigma, R, F \setminus \{f\})$ are equivalent. The TSG G is *reduced* if no fragment $f \in F$ is useless. Clearly, for every TSG we can construct an equivalent reduced TSG.

Theorem 8. *For every effectively presented* $L \in$ TSL*, it is decidable whether* $L \in$ LTL*.*

Proof. Let $G = (\Sigma, R, F)$ be a reduced tree substitution grammar such that $L(G) = L$. We construct the local tree grammar $G' = (\Sigma, R, F')$ with

$$F' = \{f(p)(f(p.1), \ldots, f(p.k)) \mid f \in F,\ k \in \mathbb{N},\ p \in \operatorname{pos}(f) \setminus \operatorname{leaf}(f),\ f(p) \in \Sigma_k\}.$$

Obviously, $L = L(G) \subseteq L(G')$ and all fragments of F' are essential for this property. Consequently, L is local if and only if $L(G') \subseteq L$. Since both $L(G')$ and L are regular by Theorem 7 and inclusion is decidable for regular tree languages [6,7], we obtain the desired statement. □

5 Closure Properties

In this section, we investigate the closure properties of the class of tree substitution languages. More specifically, we investigate the Boolean operations and the hierarchy for union. Unfortunately, the results are all negative, but they and, in particular, their proofs shed additional light on the expressive power of tree substitution languages. Let us start with union.

Theorem 9. TSL *is not closed under union.*

Proof. Consider the ranked alphabet $\Sigma = \{\sigma^{(2)}, \gamma^{(1)}, \alpha^{(0)}, \beta^{(0)}\}$ and the LTGs

$$G_1 = \big(\Sigma, \{\sigma\}, \{\sigma(\gamma, \alpha),\ \gamma(\gamma),\ \gamma(\alpha)\}\big)$$
$$G_2 = \big(\Sigma, \{\sigma\}, \{\sigma(\gamma, \beta),\ \gamma(\gamma),\ \gamma(\beta)\}\big),$$

which generate the local tree languages (see Fig. 4)

$$L(G_1) = \big\{\sigma\big(c^n[\alpha], \alpha\big) \mid n \in \mathbb{N}\big\} \qquad \text{and} \qquad L(G_2) = \big\{\sigma\big(c^n[\beta], \beta\big) \mid n \in \mathbb{N}\big\}$$

with $c = \gamma(\square)$. Now suppose that their union $L = L(G_1) \cup L(G_2)$ is a tree substitution language; i.e., $L \in \mathrm{TSL}$. Hence there exists a TSG $G = (\Sigma, R, F)$ such that $L(G) = L$. Let $n \in \mathbb{N}$ be such that $n > \max_{f \in F} \mathrm{ht}(f)$. Since $t = \sigma(c^n[\alpha], \alpha) \in L$, there must exist a derivation $\sigma \Rightarrow_G^* t$ and $\sigma \in R$. Since $\mathrm{ht}(t) > n$ at least two derivation steps are required, so $\sigma \Rightarrow_G \sigma(c^k[\gamma], \alpha) \Rightarrow_G^+ t$ for some $0 \leq k < n$, which yields the subderivation $\gamma \Rightarrow_G^+ c^{n-k}[\alpha]$. In the same manner we consider the tree $t' = \sigma(c^n[\beta], \beta) \in L$, for which the derivation $\sigma \Rightarrow_G \sigma(c^\ell[\gamma], \beta) \Rightarrow_G^+ t'$ for some $0 \leq \ell < n$ and the subderivation $\gamma \Rightarrow_G^+ c^{n-\ell}[\beta]$ must exist. However, exchanging the subderivations yields the derivation

$$\sigma \Rightarrow_G \sigma(c^k[\gamma], \alpha) \Rightarrow_G^+ \sigma(c^k[c^{n-\ell}[\beta]], \alpha),$$

which shows $\sigma(c^{n-\ell+k}[\beta], \alpha) \in L(G) = L$ contradicting $L = L(G_1) \cup L(G_2)$. □

Since the class of regular tree languages is closed under union [6,7], we obtain the following corollary from Theorems 7 and 9.

Corollary 10. $\mathrm{LTL} \subsetneq \mathrm{TSL} \subsetneq \mathrm{RTL}$.

We demonstrated that the union of two tree substitution languages need not be a tree substitution language. Next, we ask ourselves whether additional unions increase the expressive power even further. For every $k \in \mathbb{N}$ let

$$\cup_k\text{-TSL} = \{L_1 \cup \cdots \cup L_k \mid L_1, \ldots, L_k \in \mathrm{TSL}\}$$

be the class of those tree languages that can be presented as unions of k tree substitution languages. Since $\emptyset \in \mathrm{TSL}$ (see Theorem 6), we obtain $\cup_0\text{-TSL} = \emptyset$, $\cup_1\text{-TSL} = \mathrm{TSL}$, and $\cup_k\text{-TSL} \subseteq \cup_{k+1}\text{-TSL}$ for every $k \in \mathbb{N}$. Next, we show that the mentioned inclusion is actually strict, so that we obtain an infinite hierarchy.

Theorem 11. $\cup_k\text{-TSL} \subsetneq \cup_{k+1}\text{-TSL}$ *for all* $k \in \mathbb{N}$.

Proof. The statement is clear for $k = 0$, so let $k \geq 1$. Consider the ranked alphabet $\Sigma = \{\sigma^{(2)}, \delta^{(2)}, \alpha^{(0)}\}$ and the TSG $G_i = (\Sigma, \{\sigma\}, F_i)$ for every $i \in [k+1]$, where

$$F_i = \{\sigma(\delta, s_i),\ s_i,\ \delta(\delta, \alpha),\ \delta(s_i, \alpha)\}$$

and $s_i = c_r^i[\alpha]$ with $c_r = \delta(\alpha, \square)$. Clearly, $L(G_i) = \{\sigma(c_\ell^n[s_i], s_i) \mid n \in \mathbb{N}\}$ with $c_\ell = \delta(\square, \alpha)$. The tree substitution language $L(G_i)$ and the tree s_i are illustrated in Fig. 5.

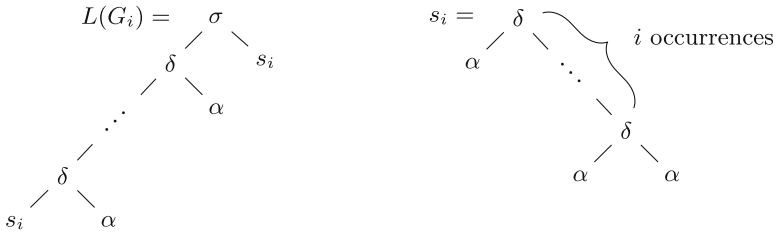

Fig. 5. Illustration of the tree substitution languages used in the proof of Theorem 11.

Obviously, $L = L(G_1) \cup \cdots \cup L(G_{k+1}) \in \cup_{k+1}$-TSL and those individual tree languages are infinite and pairwise disjoint. For the sake of a contradiction, assume that $L \in \cup_k$-TSL; i. e. there exist $L'_1, \ldots, L'_k \in$ TSL such that $L = L'_1 \cup \cdots \cup L'_k$. The pigeonhole principle establishes that there exist $i \in [k]$ and $m, n \in [k+1]$ with $m \neq n$ such that $L_m \cap L'_i$ and $L_n \cap L'_i$ are infinite. Let $G = (\Sigma, R, F)$ be a TSG such that $L(G) = L'_i$. Let $z > \max_{f \in F} \operatorname{ht}(f)$. Since $L_m \cap L(G)$ is infinite, there exists $x > z$ such that $\sigma(c_\ell^x[s_m], s_m) \in L(G)$. Similarly, there exists $y > z$ such that $\sigma(c_\ell^y[s_n], s_n) \in L(G)$ because $L_n \cap L(G)$ is infinite. Inspecting the derivations for those trees there exist $x', y' \in \mathbb{N}$ such that

$$\sigma \Rightarrow_G \sigma(c_\ell^{x'}[\delta], s_m) \Rightarrow_G^* \sigma(c_\ell^x[s_m], s_m) \quad \text{with subderivation} \quad \delta \Rightarrow_G^+ c_\ell^{x-x'}[s_m]$$
$$\sigma \Rightarrow_G \sigma(c_\ell^{y'}[\delta], s_n) \Rightarrow_G^* \sigma(c_\ell^y[s_n], s_n) \quad \text{with subderivation} \quad \delta \Rightarrow_G^+ c_\ell^{y-y'}[s_n]$$

Exchanging the subderivations we obtain

$$\sigma \Rightarrow_G \sigma(c_\ell^{x'}[\delta], s_m) \Rightarrow_G^* \sigma(c_\ell^{x'+y-y'}[s_n], s_m)$$

and thus $\sigma(c_\ell^{x'+y-y'}[s_n], s_m) \in L(G) \subseteq L$, which is a contradiction because $m \neq n$. □

Corollary 12 (of Theorem 11).

$$\cup_0\text{-TSL} \subsetneq \cup_1\text{-TSL} \subsetneq \cup_2\text{-TSL} \subsetneq \cup_3\text{-TSL} \subsetneq \cup_4\text{-TSL} \subsetneq \cdots$$

Let us move on to intersection. Unfortunately, TSL is not closed under intersection, but intersections of TSL become quite powerful. In particular, they allow information to be transported over unbounded distances, which can be observed from the proof.

Theorem 13. TSL *is not closed under intersection.*

Proof. Recall the ranked alphabet $\Sigma = \{\sigma^{(2)}, \delta^{(2)}, \alpha^{(0)}, \beta^{(0)}\}$ and the TSG G of Example 2 as well as the contexts $c_\alpha = \delta(\alpha, \delta(\alpha, \square))$ and $c_\beta = \delta(\beta, \delta(\beta, \square))$ from Example 5. Additionally, let $G' = (\Sigma, \{\sigma\}, F')$ with F' displayed in Fig. 6. The generated tree substitution languages $L(G)$ and $L(G')$ are

$$\{\sigma(x, c_1[\cdots c_n[\delta(y, \alpha)] \cdots]) \mid x, y \in \{\alpha, \beta\},\ n \in \mathbb{N},\ \forall i \in [n] \colon c_i \in \{c_\alpha, c_\beta\}\}$$
$$\{\sigma(x, \delta(x, c_1[\cdots c_n[\alpha] \cdots]) \mid x \in \{\alpha, \beta\},\ n \in \mathbb{N},\ \forall i \in [n] \colon c_i \in \{c_\alpha, c_\beta\}\}$$

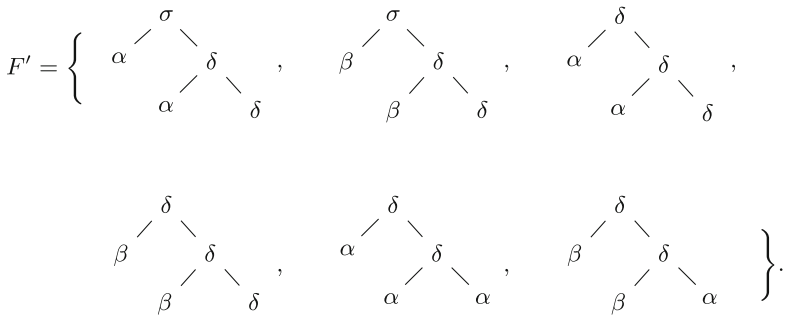

Fig. 6. Fragments of the TSG G' used in the proof of Theorem 13.

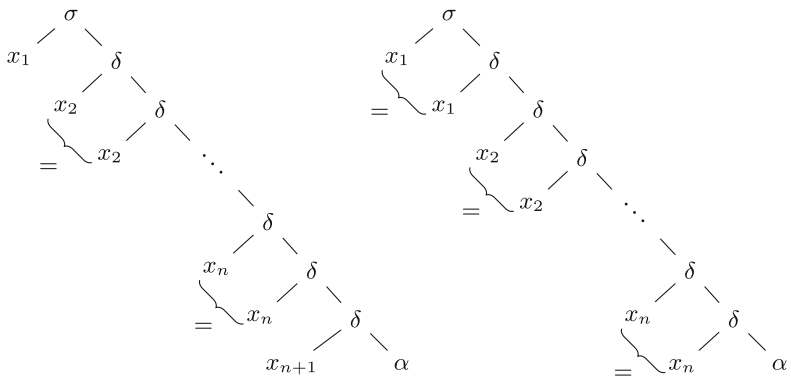

Fig. 7. Tree substitution languages $L(G)$ and $L(G')$ used in the proof of Theorem 13.

respectively, which are also illustrated in Fig. 7. Their intersection

$$L(G) \cap L(G') = \{\sigma(\alpha, \delta(\alpha, c_\alpha[\cdots c_\alpha[\alpha]\cdots])) \mid n \in \mathbb{N}\} \cup \{\sigma(\beta, \delta(\beta, c_\beta[\cdots c_\beta[\alpha]\cdots])) \mid n \in \mathbb{N}\}$$

contains only trees, in which all left children along the spine carry the same label. This tree language is not a tree substitution language, which can be proved using the subderivation exchange technique used in the proof of Theorem 9. □

Note how the intersection achieves a global synchronization in the proof of Theorem 13. This power makes the investigation of the intersection hierarchy difficult. We leave the strictness of the intersection hierarchy as an open problem and conclude by considering the complement.

Theorem 14. TSL *is not closed under complements.*

Proof. Consider the ranked alphabet $\Sigma = \{\gamma^{(1)}, A^{(1)}, B^{(1)}, \alpha^{(0)}, \beta^{(0)}\}$ and the LTG $G = (\Sigma, \{\gamma\}, F)$ with fragments

$$F = \{\gamma(A),\ A(A),\ A(\alpha)\} \cup \{\gamma(B),\ B(B),\ B(\beta)\}.$$

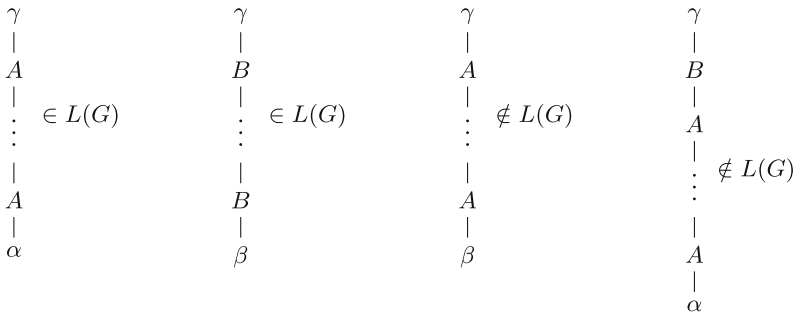

Fig. 8. Trees used in the proof of Theorem 14.

The generated tree language is illustrated in Fig. 8. Now suppose that its complement $L = T_\Sigma(\Sigma) \setminus L(G)$ is a tree substitution language; i.e., $L \in TSL$. Hence there exists a TSG $G' = (\Sigma, R, F)$ such that $L = L(G')$. Let $n \in \mathbb{N}$ be such that $n > \max_{f \in F} \operatorname{ht}(f)$. Since $t = \gamma(A^n(\beta)) \in L$ (see Fig. 8) there must exist a derivation $\gamma \Rightarrow_G^* t$ and $\gamma \in R$. Since $\operatorname{ht}(t) > n$ at least two derivation steps are required, so $\gamma \Rightarrow_G \gamma(A^k) \Rightarrow_G^+ t$ for some $0 \leq k < n$, which yields the subderivation $A \Rightarrow_G^+ A^{n-k}(\beta)$. Similarly, we consider the tree $t' = \gamma(B(A^n(\alpha))) \in L$ (see Fig. 8), for which the derivation $\gamma \Rightarrow_G^+ \gamma(B(A^\ell)) \Rightarrow_G^+ t'$ for some $0 \leq \ell < n$ and the subderivation $A \Rightarrow_G^+ A^{n-\ell}(\alpha)$ must exist. However, exchanging the subderivations yields the derivation

$$\gamma \Rightarrow_G \gamma(A^k) \Rightarrow_G^+ \gamma(A^k(A^{n-\ell}(\alpha)),$$

which shows $\gamma(A^k(A^{n-\ell}(\alpha)) \in L(G') = L$ contradicting $L = T_\Sigma(\Sigma) \setminus L(G)$. □

6 Open Problems

We showed that it is decidable whether a given tree substitution language is local. It remains open if we can also decide whether a given regular tree language is a tree substitution language. Progress on this problem will probably provide additional fine-grained insight into the expressive power of tree substitution grammars in comparison to the regular tree grammars.

Another open problem concerns the intersection hierarchy. We showed that unions of tree substitution languages can progressively express more and more tree languages. A similar hierarchy also exists for intersections of tree substitution languages and we showed that the intersection of two tree substitution languages is not necessarily a tree substitution languages. However, it remains open whether there is an infinite intersection hierarchy or whether it collapses at some level.

Acknowledgements. The authors gratefully acknowledge the financial support of the Research Training Group 1763 (QuantLA: Quantitative Logics and Automata), which is funded by the German Research Foundation (DFG). In addition, the authors would like to thank the anonymous reviewers for the careful reading of the manuscript and their valuable feedback.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley, Boston (1994)
2. Aho, A.V., Ullman, J.D.: The Theory of Parsing, Translation, and Compiling. Prentice Hall, Upper Saddle River (1972)
3. Bod, R.: The data-oriented parsing approach: theory and application. In: Fulcher, J., Jain, L.C. (eds.) Computational Intelligence: A Compendium. SCI, vol. 115, pp. 307–348. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78293-3_7
4. Chiang, D., Knight, K.: An introduction to synchronous grammars (2006). Tutorial at 44th ACL. https://www3.nd.edu/~dchiang/papers/synchtut.pdf
5. Comon, H., et al.: Tree automata techniques and applications (2007). http://tata.gforge.inria.fr
6. Gécseg, F., Steinby, M.: Tree languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 3, pp. 1–68. Springer, Heidelberg (1997). https://doi.org/10.1007/978-3-642-59126-6_1
7. Gécseg, F., Steinby, M.: Tree Automata. arXiv, 2nd edn. (2015). https://arxiv.org/abs/1509.06233
8. Howcroft, D.M., Klakow, D., Demberg, V.: Toward Bayesian synchronous tree substitution grammars for sentence planning. In: Proceedings of the 11th NLG, pp. 391–396. ACL (2018)
9. Joshi, A.K., Levy, L.S., Takahashi, M.: Tree adjunct grammars. J. Comput. Syst. Sci. **10**(1), 136–163 (1975)
10. Joshi, A.K., Schabes, Y.: Tree-adjoining grammars and lexicalized grammars. Technical report, MS-CIS-91-22, University of Pennsylvania (1991)
11. Joshi, A.K., Schabes, Y.: Tree-adjoining grammars. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 3, pp. 69–123. Springer, Heidelberg (1997). https://doi.org/10.1007/978-3-642-59126-6_2
12. Jurafsky, D., Martin, J.H.: Speech and Language Processing, 2nd edn. Prentice Hall, Upper Saddle River (2008)
13. Maletti, A.: Why synchronous tree substitution grammars? In: Proceedings of the 2010 NAACL, pp. 876–884. ACL (2010)
14. Maletti, A.: An alternative to synchronous tree substitution grammars. J. Nat. Lang. Eng. **17**(2), 221–242 (2011)
15. Sangati, F., Keller, F.: Incremental tree substitution grammar for parsing and sentence prediction. Trans. ACL **1**, 111–124 (2013)
16. Shindo, H., Miyao, Y., Fujino, A., Nagata, M.: Bayesian symbol-refined tree substitution grammars for syntactic parsing. In: Proceedings of the 50th ACL, pp. 440–448. ACL (2012)
17. Wilhelm, R., Seidl, H., Hack, S.: Compiler Design. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-17540-4

18. Yu, S.: Regular languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 1, pp. 41–110. Springer, Heidelberg (1997). https://doi.org/10.1007/978-3-642-59136-5_2
19. Zhang, J., Zhai, F., Zong, C.: Syntax-based translation with bilingually lexicalized synchronous tree substitution grammars. IEEE Trans. Audio Speech Lang. Proc. **21**(8), 1586–1597 (2013)