Some Results on Top-context-free Tree Languages

Dieter Hofbauer[1] * Maria Huber[2] ** Gregory Kucherov[2]

[1] Technische Universität Berlin, Franklinstraße 28/29, FR 6-2
D - 10587 Berlin, Germany, e-mail: dieter@cs.tu-berlin.de
[2] CRIN & INRIA-Lorraine, 615, rue du Jardin Botanique
54602 Villers-lès-Nancy, France, e-mail: {huber,kucherov}@loria.fr

Abstract. Top-context-free tree languages (called *corégulier* by Arnold and Dauchet [1, 2]) constitute a natural subclass of context-free tree languages. In this paper, we give further evidence for the importance of this class by exhibiting certain closure properties. We systematically treat closure under the operations *replacement* and *substitution* as well as under the corresponding *iteration* operations. Several other well-known language classes are considered as well. Furthermore, various characterizations of the regular top-context-free languages are given, among others by means of restricted regular expressions.

1 Introduction

This paper is motivated by our previous work on tree languages related to term-rewriting systems [11, 7, 10]. It is well-known that for a left-linear term-rewriting system R, the set $Red(R)$ of ground terms reducible by R is a regular tree language. Conversely, if $Red(R)$ is regular, then R can effectively be "linearized", i.e., a finite language can be substituted for its non-linear variables without changing the set of reducible ground terms [7, 10, 14]. However, little is known about conditions on which $Red(R)$ is context-free. Here again, non-linear variables play a crucial role.

This motivation led us to study the class of *top-context-free* languages, which turned out to be of special importance. Top-context-free tree languages are languages generated by context-free tree grammars in which right-hand sides of production rules contain non-terminal symbols, if at all, only at the top position. This class has been studied by Arnold and Dauchet [1, 2] who proved in particular the following result.

Theorem 1 *The language $L = \{f(t,t) \mid t \in L'\}$ is context-free iff L' is top-context-free iff L itself is top-context-free.*

The theorem remains true if "context-free" and "top-context-free" are replaced by "regular" and "finite" respectively. We conjecture that this analogy

* Partially supported by a post-doc grant of the M.E.N.
** Supported by the M.E.S.R.

can also be drawn for the language $Red(R)$ mentioned above: It is context-free iff a top-context-free language can be substituted for the non-linear variables in R without changing the set of reducible ground terms. This paper is devoted to the study of some related topics that could serve as a basis for further investigations.

The class of top-context-free languages is in a sense orthogonal to that of regular languages. Though these two classes intersect, there are very simple languages that are top-context-free but not regular and vice versa. For example, the language $\{f(g^i(a), g^i(a)) \mid i \geq 0\}$ is top-context-free but not regular. Conversely, Arnold and Dauchet showed that the language of all terms over a signature of one binary and one constant symbol is not top-context-free. In order to obtain a criterion that a language is not top-context-free we, more generally, prove that every top-context-free language is *slim*, a simple syntactic property.

We further study the class of languages that are both top-context-free and regular. We propose a number of equivalent characterizations of this class in terms of grammars (linear top-context-free, non-branching regular), linear regular expressions, and more syntactic properties (slim, passable, polynomially size-bounded).

Closure properties play an important role in all branches of formal language theory. A detailed analysis of closure properties for the class of regular tree languages can be found in [6]. In this paper we continue this topic, analyzing closure of different classes under the operations *replacement*, *substitution*, and their iterations. Replacement and substitution, considered also in [5], are two possible extentions of the string product to the tree case. In contrast to replacement, substitution replaces equal symbols by equal terms. We make an exhaustive study of closure properties under these operations for six classes of tree languages: finite, regular, linear top-context-free, top-context-free, linear context-free, and context-free.

2 Notations

We assume the reader to be familiar with basic definitions in term rewriting [4] and formal language theory [9, 6]. A signature Σ is a finite set of function symbols of fixed arity; for $n \geq 0$, Σ_n denotes the set of symbols in Σ of arity n. $\mathcal{T}_\Sigma(X)$ is the set of (finite) terms over Σ and a set of variables X. The set of ground terms, i.e., terms without variables, over Σ is denoted by $\mathcal{T}_\Sigma$. For $t \in \mathcal{T}_\Sigma(X)$, $\mathcal{P}os(t)$ is the set of positions in t, defined in the usual way as sequences of natural numbers. We write $p \leq q$ if p is a prefix of q. The subterm of t at position p is $t|_p$. For $p \in \mathcal{P}os(t)$, $t|_p$ is a principal subterm of t if $|p| = 1$, and it is a proper subterm of t if $|p| \geq 1$. Here, $|p|$ denotes the length of p. The depth of t is $|t| = max\{|p| \mid p \in \mathcal{P}os(t)\}$, its size is $size(t) = |\mathcal{P}os(t)|$. If t' is a subterm of t, then t can be written as $c[t']$ where $c[\,]$ is a context. A context is called Σ-context in case it contains only symbols from Σ.

A variable x is said to be linear in t if there is only one position $p \in \mathcal{P}os(t)$ such that $t|_p = x$, and is said to be non-linear in t otherwise. A term is linear if all its variables are linear and is non-linear otherwise.

If unambiguous, we sometimes prefer vector notation to "three dots notation". For example, $f(\vec{t})$ abbreviates $f(t_1, \ldots, t_n)$ (n and t_i will be clear from the context), and $f(\vec{x})$ stands for $f(x_1, \ldots, x_n)$. For $f \in \Sigma_n$ and $L_1, \ldots, L_n \subseteq \mathcal{T}_\Sigma$ let $f(L_1, \ldots, L_n) = \{f(t_1, \ldots, t_n) \mid t_1 \in L_1, \ldots, t_n \in L_n\}$. The cardinality of a finite (multi-)set S is denoted by $|S|$.

3 Context-free languages, Replacement, and Substitution

Definition 2 *A* context-free tree grammar $G = (N, \Sigma, P, S)$ *consists of disjoint signatures* N (nonterminals) *and* Σ (terminals), *a finite rewrite system* P *over* $N \cup \Sigma$, *and a distinct constant symbol* $S \in N_0$ (initial symbol); *all rules in* P *are of the form*

$$A(x_1, \ldots, x_n) \to t$$

where $A \in N_n$, $n \geq 0$, $x_1, \ldots, x_n$ *are pairwise different variables, and* $t \in \mathcal{T}_{N \cup \Sigma}(\{x_1, \ldots, x_n\})$.

- G *is said to be* regular *if* N *contains only constant symbols.*
- G *is said to be* top-context-free *if all proper subterms of right-hand sides of rules are in* $\mathcal{T}_\Sigma(\{x_1, \ldots, x_n\})$.
- G *is said to be* linear *if all right-hand sides of rules in* P *are linear.*

The language generated by a grammar $G = (N, \Sigma, P, S)$ is

$$\mathcal{L}(G) = \{t \in \mathcal{T}_\Sigma \mid S \to_P^* t\}.$$

A language $L \subseteq \mathcal{T}_\Sigma$ is called (linear) context-free (regular, ...) if there is a (linear) context-free (regular, ...) grammar generating L. For $A \in N_n$ we also use the more general notation

$$\mathcal{L}(G, A(x_1, \ldots, x_n)) = \{t \in \mathcal{T}_\Sigma(X) \mid A(x_1, \ldots, x_n) \to_P^* t\}.$$

Thus $\mathcal{L}(G) = \mathcal{L}(G, S)$.

FIN (REG, LINTOPCF, TOPCF, LINCF, CF) will denote the class of finite (regular, linear top-context-free, top-context-free, linear context-free, context-free) languages. By definition we have the inclusions FIN $\subseteq$ REG $\subseteq$ LINCF $\subseteq$ CF, FIN $\subseteq$ LINTOPCF $\subseteq$ TOPCF $\subseteq$ CF and LINTOPCF $\subseteq$ LINCF. All inclusions are proper and both LINTOPCF and TOPCF are incomparable with REG.

Top-context-free languages were studied by Arnold and Dauchet [1, 2] under the name of *langages coréguliers*. In [1] they showed that TOPCF coincides with the class of languages obtained by deterministic top-down tree transformations on monadic regular languages. It can even be shown that it is sufficient to consider a single monadic language, e.g., $\mathcal{T}_{\{0,1,\lambda\}}$ where 0 and 1 are symbols of arity 1 and λ is a constant symbol.

Regular tree languages have been extensively studied, e.g., in [6]. For context-free tree languages see, e.g., [12]. Several normal forms have been defined for regular and context-free grammars. In this paper we will use the fact that for each grammar there is a *reduced* grammar of the same type, generating the same language. A grammar is said to be reduced if all nonterminals A in N_n, $n \geq 0$, are

- *reachable*, i.e., $S \rightarrow_P^* c[A(t_1, \ldots, t_n)]$ for some term $c[A(t_1, \ldots, t_n)] \in \mathcal{T}_{\Sigma \cup N}$, and
- *productive*, i.e., $\mathcal{L}(G, A(x_1, \ldots, x_n)) \neq \emptyset$.

It is well-known that such a normal form always exists. For top-context free grammars we will use another normal form. We call a context-free grammar *slow* if all right-hand sides of its rules contain exactly one Σ-symbol. Thus, slow top-context-free grammars contain only rules of the form

$$A(x_1, \ldots, x_n) \rightarrow f(y_1, \ldots, y_m)$$
$$A(x_1, \ldots, x_n) \rightarrow B(z_1, \ldots, f(y_1, \ldots, y_m), \ldots, z_k),$$

where $B \in N_k$, $k \geq 1$, $f \in \Sigma_m$, $m \geq 0$, $\{y_1, \ldots, y_m, z_1, \ldots, z_k\} \subseteq \{x_1, \ldots, x_n\}$. Whereas not all context-free languages can be generated by slow grammars (for an example see [3], exercise 17), for each (linear) top-context-free language there is a slow reduced (linear) top-context-free grammar generating this language. A proof can be found in [8].

The operations c-replacement and c-substitution constitute two different ways of "replacing" a constant c in all terms of a language L by terms of a language L'. The c-substitution operation substitutes all occurrences of c in a term of L by the same term of L'. When applying the c-replacement operation, all c's in terms of L are replaced independently by possibly different terms of L'. Thus the c-substitution corresponds to the usual substitution, if c is treated as a variable. The c-replacement corresponds to a substitution, where all c's are considered as different (linear) variables. In [5] c-replacement and c-substitution are called OI- and IO-substitution respectively.

Definition 3 *For languages L, L' and a constant symbol c, the* c-replacement of L' into L *is $L \cdot_c L' = \bigcup_{t \in L} t \cdot_c L'$, where $t \cdot_c L'$ is defined by (for $n \geq 0$)*

$$f(t_1, \ldots, t_n) \cdot_c L' = \begin{cases} L' & \text{if } f = c, \\ f(t_1 \cdot_c L', \ldots, t_n \cdot_c L') & \text{otherwise.} \end{cases}$$

The c-substitution of L' into L *is $L \diamond_c L' = \bigcup_{t \in L} \bigcup_{t' \in L'} \{t \diamond_c t'\}$, where $t \diamond_c t'$ is defined by (for $n \geq 0$)*

$$f(t_1, \ldots, t_n) \diamond_c t' = \begin{cases} t' & \text{if } f = c, \\ f(t_1 \diamond_c t', \ldots, t_n \diamond_c t') & \text{otherwise.} \end{cases}$$

When c is clear from the context or arbitrary, we will not mention it. Given classes of languages $\mathcal{C}$ and $\mathcal{C}'$, $\mathcal{C}$ is said to be *closed under replacement (substitution)* by $\mathcal{C}'$, if $L \cdot_c L' \in \mathcal{C}$ ($L \diamond_c L' \in \mathcal{C}$ respectively) holds for all $L \in \mathcal{C}$, $L' \in \mathcal{C}'$ and all constant symbols c. We will also write $\mathcal{C} \cdot \mathcal{C}' \subseteq \mathcal{C}$ ($\mathcal{C} \diamond \mathcal{C}' \subseteq \mathcal{C}$). $\mathcal{C}$ is said to be closed under replacement (substitution) if it is closed under replacement (substitution) by $\mathcal{C}$ (cf. Gécseg and Steinby [6], III.3.6 and II.4.3).

The replacement and the substitution operations for tree languages give rise to two types of star operations, the replacement iteration and the substitution

iteration. For a language L and a constant c, the *c-replacement iteration* is defined by $L^{*_c} = \bigcup_{n\geq 0} L_n$, where

$$L_0 = \{c\} \text{ and } L_{n+1} = L_{n \cdot_c}(L \cup \{c\}).$$

The *c-substitution iteration* is defined by $L^{\diamond_c} = \bigcup_{n\geq 0} L^{\diamond_{n,c}}$, where

$$L_0 = \{c\} \text{ and } L_{n+1} = L_n{}^{\diamond_c} L$$

The c-replacement iteration (called *c-iteration* in [6]) generalizes the star operation for word languages to trees. Given a class $\mathcal{C}$ of languages, $\mathcal{C}$ is said to be closed under replacement iteration (substitution iteration), if $L^{*_c} \in \mathcal{C}$ ($L^{\diamond_c} \in \mathcal{C}$ respectively) holds for all $L \in \mathcal{C}$ and all constant symbols c. We will also write $\mathcal{C}^* \subseteq \mathcal{C}$ ($\mathcal{C}^\diamond \subseteq \mathcal{C}$).

4 Top-context-free Languages Are Slim

In this section we give a a criterion for showing that certain languages are not top-context-free. It can be seen as a generalization of a proof method used by Arnold and Dauchet in [1]. We prove that every top-context-free language is *slim*. Intuitively, a term is slim if it can be "decomposed" in a top-down way such that at each intermediate step only a bounded number of different subterms occur. If there is such a bound, uniform for all terms in the language, then the language is said to be slim. Formally:

Definition 4 (slim) *A* decomposition *of $t \in \mathcal{T}_\Sigma$ is a finite sequence $D_0, \ldots, D_m$ of subsets of $\mathcal{T}_\Sigma$ where $D_0 = \{t\}$, $D_m = \emptyset$, and for all i, $0 \leq i < m$, there is some term $f(t_1, \ldots, t_n) \in D_i$, $n \geq 0$, such that*

$$D_{i+1} = (D_i \setminus \{f(t_1, \ldots, t_n)\}) \cup \{t_1, \ldots, t_n\}.$$

t is k-slim *for $k \in I\!N$ if t has a decomposition $D_0, \ldots, D_m$ where $|D_i| \leq k$ for all i, $0 \leq i \leq m$. A language is k-slim if it contains only k-slim terms; it is said to be* slim *if it is k-slim for some k.*

EXAMPLE 5. All terms $f(a, f(a, \ldots f(a, a) \ldots))$ are 2-slim. The same is true, more generally, for all terms $f(g^{i_1}(a), f(g^{i_2}(a), \ldots f(g^{i_m}(a), a) \ldots)))$, $i_j \in I\!N$.

EXAMPLE 6. Define $t_0 = a, t_{i+1} = f(t_i, t_i)$. The sequence $\{t_i\}, \{t_{i-1}\}, \ldots, \{t_0\}, \emptyset$ is a decomposition of t_i, therefore t_i is 1-slim.

Note that every subterm of a k-slim term is also k-slim. Note also that all languages over a signature containing only symbols of arity at most one are 1-slim. Hence there are slim languages which are not top-context-free – not even recursively enumerable.

Lemma 7 *Top-context-free languages are slim.*

Proof. Let $G = (N, \Sigma, P, S)$ be a slow top-context-free grammar and let k be the maximal arity of symbols in $N \cup \Sigma$. We will show that $\mathcal{L}(G)$ is k-slim. Consider a derivation $S = t_0 \to_P \ldots \to_P t_m = t$ of a term $t \in \mathcal{T}_\Sigma$. Define sets D_i, $1 \leq i \leq m$, by $D_m = \{t_m\}$, $D_{m-1} = \{s \mid s \text{ a principal subterm of } t_m\}$, and, for $0 < i < m$,

$$D_{i-1} = \begin{cases} (D_i \cap (\{y_1\gamma, \ldots, y_p\gamma\} \setminus \{f(z_1, \ldots, z_q)\gamma\})) \cup \{z_1\gamma, \ldots, z_q\gamma\} \\ \qquad \text{if } f(z_1, \ldots, z_q)\gamma \in D_i, \\ D_i \quad \text{else.} \end{cases}$$

where $t_{i-1} = l\gamma$ and $t_i = r\gamma$ for a rule

$$l = A(x_1, \ldots, x_n) \to B(y_1, \ldots, f(z_1, \ldots, z_q), \ldots, y_p) = r$$

from P. If now in the sequence $D_m, \ldots, D_0$ repetitions of sets are eliminated we obtain a decomposition of t. Clearly $|D_i| \leq k$, thus t is k-slim. □

The next – somewhat technical – lemma gives sufficient conditions for a term to be *not* k-slim. Define $B_k = \mathcal{P}os(t_k)$, t_k from example 6, and $b_k = \{p \in B_k \mid |p| = k\}$. An injective function $h : B_k \to D$ is a *(homeomorphic) embedding* from B_k into a tree domain D, if different edges in B_k – seen as a directed graph – map to disjoint paths in D. Note that $h(B_k)$ is uniquely determined by $h(b_k) \subseteq D$.

Lemma 8 *Let $t \in \mathcal{T}_\Sigma$ and $h : B_k \to \mathcal{P}os(t)$ be an embedding. Suppose $t|_q \neq t|_{\bar{q}}$ for all $p, \bar{p} \in h(b_k)$ and all $q, \bar{q}$ with $q \leq p$, $\bar{q} \leq \bar{p}$, $q \neq \bar{q}$. Then t is not k-slim.*

We conclude this section with some immediate corollaries to lemma 8. Let us mention, however, the limitations of this simple criterion. The term $h(h(a,b,c), h(b,c,a), h(c,a,b))$, e.g., is not 4-slim, but this is not provable using lemma 8.

EXAMPLE 9. The language $\mathcal{T}_\Sigma$ of all ground terms over signature Σ is not slim if, and only if, $\Sigma_n \neq \emptyset$ for some $n \geq 2$, and $\Sigma_0 \neq \emptyset$. Clearly terms built up using only unary symbols and constants are 1-slim, also the empty set is slim. Conversely, given a symbol $h \in \Sigma_n$, $n \geq 2$, and $a \in \Sigma_0$, for each k there are terms which are not k-slim. Define terms $t[i,j]$ inductively by

$$\begin{aligned} t[0,0] &= h(a, a, \ldots), \\ t[0, j+1] &= h(a, t[0,j], \ldots), \\ t[i+1, j] &= h(t[i, 2j], t[i, 2j+1], \ldots) \end{aligned}$$

where $\ldots$ is filled with a's. Now $t[k,0]$ is not k-slim by lemma 8 using an embedding where $h(b_k) = \{1,2\}^k$.

Corollary 10 *$\mathcal{T}_\Sigma$ is top-context-free iff Σ contains only symbols of arity at most one or no constant symbol.*

This is obvious from example 9 and the fact that, in case Σ has only symbols of arity at most one, $\mathcal{T}_\Sigma$ is generated by the top-context-free grammar

$$\begin{aligned} S &\to A(c) \text{ for all constant symbols } c \in \Sigma, \\ A(x) &\to A(f(x)) \text{ for all unary symbols } f \in \Sigma, \\ A(x) &\to x. \end{aligned}$$

By a simple generalization of example 9 we can prove the following lemma which will be used in the next section.

Definition 11 (branching) *A regular grammar $G = (N, \Sigma, P, S)$ is said to be* branching *if $A \to_P^* c[A, A]$ for some $A \in N$ and some term $c[A, A] \in \mathcal{T}_{N \cup \Sigma}$ containing A at more than one occurrence. Otherwise G is* non-branching.

Lemma 12 *If G is a branching reduced regular grammar, then $\mathcal{L}(G)$ is not slim, hence not top-context-free.*

5 Regular Top-context-free Languages

In this section we study the class of languages that are both regular and top-context-free.

5.1 Linear Top-context-free Languages

For strings, a right-regular grammar (rules of the form $A \to wB$) can always be transformed into an equivalent left-regular one (rules of the form $A \to Bw$), and vice versa. When regarding string grammars as tree grammars – map letters to unary function symbols – this means transforming a regular tree grammar into an equivalent top-context-free tree grammar and vice versa. This is not possible in general. The following lemma, however, allows to go from regular grammars to (linear) top-context-free ones and vice versa under additional assumptions.

Lemma 13 *For a language L the following statements are equivalent:*

(1) L is generated by a regular grammar where no right-hand side of a rule contains more than one nonterminal.
(2) L is generated by a linear top-context-free grammar where each nonterminal has arity at most one.
(3) L is generated by a top-context-free grammar where the right-hand side of every rule contains at most one variable position.

Using lemma 13 we prove now the converse to lemma 12.

Lemma 14 *A language generated by a non-branching regular grammar is linear top-context-free.*

Proof. Let $G = (N, \Sigma, P, S)$ be a non-branching regular grammar that we suppose to be reduced. We will show that $\mathcal{L}(G)$ is linear top-context-free by induction on $|N|$. Clearly for $N = \{S\}$, $\mathcal{L}(G)$ is linear top-context-free by lemma 13. For $|N| > 1$ we define

$$N' = \{A \in N \mid A \to_P^* c[S] \text{ for some context } c[\,]\},$$

$N'' = N \setminus N'$, $P' = P \cap \{A \to t \mid A \in N'\}$, and $G' = (N', \Sigma \cup N'', P', S)$.

First observe that $\mathcal{L}(G')$ is linear top-context-free by lemma 13. Indeed, suppose $A \to c[B, B']$ is a rule in P' where $B, B' \in N'$ occur at different positions in the term $c[B, B']$. Then by definition of N' and since G is reduced, we have

$$S \to_P^* a[A] \to_P a[c[B, B']] \to_P^* a[c[b[S], b'[S]]]$$

for some contexts $a[\,]$, $b[\,]$, $b'[\,]$, contradicting the assumption that G is non-branching.

For $A \in N''$ define $G_A = (N_A, \Sigma, P_A, A)$ where $N_A = \{B \in N \mid A \to_P^* c[B]$ for some context $c[\,]\}$ and $P_A = P \cap \{B \to t \mid B \in N_A\}$. Note that $N_A \subseteq N''$ and that G_A is reduced and non-branching. From $S \notin N''$ we get $|N_A| \leq |N''| < |N|$, thus by induction hypothesis, $\mathcal{L}(G_A)$ is linear top-context-free.

Let $N'' = \{A_1, \ldots, A_n\}$. As is easily seen, $\mathcal{L}(G_{A_i}) = \mathcal{L}(G, A_i)$ for $1 \leq i \leq n$ and

$$\mathcal{L}(G) = \mathcal{L}(G') \cdot_{A_1} \mathcal{L}(G_{A_1}) \cdots \cdot_{A_n} \mathcal{L}(G_{A_n}).$$

Since $\mathcal{L}(G')$ as well as $\mathcal{L}(G_{A_1}), \ldots, \mathcal{L}(G_{A_n})$ are linear top-context-free, $\mathcal{L}(G)$ is linear top-context-free, too, by lemma 23. □

We conclude this section with a criterion allowing to prove that certain languages are not linear top-context-free. In close analogy to the result that every top-context-free language is slim, we will show that every linear top-context-free language is *passable*, a notion defined by K. Salomaa [13]. The definition of "k-passable" is just the definition of "k-slim" if sets are treated as *multisets*. Salomaa also showed that passability of a language implies a uniform polynomial size-bound, i.e., the size of terms in the language is polynomially bounded by their depth.

Definition 15 (passable) *A* multi-decomposition *of $t \in \mathcal{T}_\Sigma$ is a finite sequence $D_0, \ldots, D_m$ of multisets over $\mathcal{T}_\Sigma$ where $D_0 = \{t\}$, $D_m = \emptyset$, and for all i, $0 \leq i < m$, there is some term $f(t_1, \ldots, t_n) \in D_i$, $n \geq 0$, such that*

$$D_{i+1} = (D_i \setminus \{f(t_1, \ldots, t_n)\}) \cup \{t_1, \ldots, t_n\}$$

where all sets and operations are interpreted as multisets and multiset operations. t is k-passable *for $k \in N$, if t has a multi-decomposition $D_0, \ldots, D_m$ where $|D_i| \leq k$ for all i, $0 \leq i \leq m$. A language is k-passable if it contains only k-passable terms; it is said to be* passable *if it is k-passable for some k.*

Salomaa [13] defines k-passability in a slightly different way – using tree automata – which, however, is easily seen to be equivalent to the definition given above. Note that only the empty set is 0-passable.

Let us call a language L *polynomially size-bounded* if there is a polynomial p over $\mathbb{N}$ with one argument such that $size(t) \leq p(|t|)$ for all $t \in L$. Lemma 3.2 in [13] states that $size(t) \leq k \cdot |t|^k + 1$ for all $t \in L$, provided that L is k-passable. Together with the following result this can be used to prove that a language is not linear top-context-free.

Lemma 16 *Linear top-context-free languages are passable.*

Proof. Let G be a linear top-context-free grammar where nonterminals have arity at most k; without loss of generality we assume that G is slow. A straightforward induction on the length of a derivation then shows that $\mathcal{L}(G)$ is $\max\{1,k\}$-passable. □

EXAMPLE 17. The top-context-free language $L = \{t_k \mid k \geq 0\}$ from example 6 is not linear top-context-free, since it is not polynomially size-bounded.

5.2 Linear Regular Expressions

The class REG of regular languages is closed under replacement and replacement iteration. Moreover, according to Kleene's theorem for tree languages, REG is the smallest class containing all finite sets and closed under union, replacement and replacement iteration (cf [6]). In other words, every regular language can be represented by a *regular expression*, constructed from finite sets by using union, replacement and replacement iteration. On the other hand, REG is not closed under substitution and substitution iteration (cf section 6). The situation is inverse for top-context-free languages. We will show in section 6 that this class is closed under substitution and substitution iteration and is not closed under replacement and replacement iteration.

Nevertheless, the subclass of top-context-free languages which are also regular can be represented by regular expressions with restricted use of replacement and replacement iteration.

Definition 18 *Let $\alpha \notin \Sigma$ be a new constant symbol. We inductively define a set of* linear regular expressions *$\mathcal{E}$, an auxiliary set of* contextual linear regular expressions *$\mathcal{CE}$, and the language $\mathcal{L}(e) \subseteq \mathcal{T}_\Sigma$ ($\mathcal{L}(e) \subseteq \mathcal{T}_{\Sigma\cup\{\alpha\}}$ resp.) represented by $e \in \mathcal{E}$ ($e \in \mathcal{CE}$ resp.) as follows:*

- *for every $a \in \Sigma_0$: $a \in \mathcal{E}$; $\mathcal{L}(a) = \{a\}$; $\alpha \in \mathcal{CE}$; $\mathcal{L}(\alpha) = \{\alpha\}$.*
- *for every $f \in \Sigma_n$: if $e_1, \ldots, e_n \in \mathcal{E}$, then $f(e_1, \ldots, e_n) \in \mathcal{E}$;*
  if $\bar{e} \in \mathcal{CE}$, and $1 \leq i \leq n$, then $f(e_1, \ldots, e_{i-1}, \bar{e}, e_{i+1}, \ldots, e_n) \in \mathcal{CE}$;
  in either case $\mathcal{L}(f(e_1, \ldots, e_n)) = f(\mathcal{L}(e_1), \ldots, \mathcal{L}(e_n))$.
- *if $e_1, e_2 \in \mathcal{E}$, then $e_1 \cup e_2 \in \mathcal{E}$; if $e_1, e_2 \in \mathcal{CE}$, then $e_1 \cup e_2 \in \mathcal{CE}$;*
  in either case $\mathcal{L}(e_1 \cup e_2) = \mathcal{L}(e_1) \cup \mathcal{L}(e_2)$.

- *if* $e_1, e_2 \in \mathcal{CE}$, *then* $e_1 \cdot_\alpha e_2 \in \mathcal{CE}$, *and* ${e_1}^{*_\alpha} \in \mathcal{CE}$;
  $\mathcal{L}(e_1 \cdot_\alpha e_2) = \mathcal{L}(e_1) \cdot_\alpha \mathcal{L}(e_2)$; $\mathcal{L}({e_1}^{*_\alpha}) = \mathcal{L}(e_1)^{*_\alpha}$.
- *if* $\bar{e} \in \mathcal{CE}$, $e \in \mathcal{E}$, *then* $\bar{e} \cdot_\alpha e \in \mathcal{E}$; $\mathcal{L}(\bar{e} \cdot_\alpha e) = \mathcal{L}(\bar{e}) \cdot_\alpha \mathcal{L}(e)$.

Obviously, for every $e \in \mathcal{E} \cup \mathcal{CE}$, $\mathcal{L}(e)$ is a regular language, since e is a regular expression. For $e \in \mathcal{CE}$, it is easy to see by induction that α occurs exactly once in each term of $\mathcal{L}(e)$. Therefore, if $e \in \mathcal{CE}$, then $\mathcal{L}(e)^{*_\alpha} = \mathcal{L}(e)^{\diamond_\alpha}$ and $\mathcal{L}(e) \cdot_\alpha L = \mathcal{L}(e) \diamond_\alpha L$ for every $L \subseteq \mathcal{T}_\Sigma$.

Proposition 19 *A language represented by a linear regular expression is top-context-free.*

Proof. According to the remark above, every replacement (replacement iteration resp.) in e can be interpreted as a substitution (substitution iteration resp.) without changing $\mathcal{L}(e)$. TopCf is closed under union, substitution and substitution iteration (see section 6). Closure under substitution implies that $f(L_1, \ldots, L_n)$ is top-context-free if $L_1, \ldots, L_n$ are top-context-free. Thus every operation in linear regular expressions preserves top-context-freeness. □

The proof of the following lemma (see [8]) uses a standard technique for constructing regular expressions from automata (or grammar) representations. It is somewhat a mixture of that for words (cf [9]) and that for trees (cf [6]).

Lemma 20 *A language generated by a non-branching regular grammar can be represented by a linear regular expression.*

5.3 Characterizations of Regular Top-context-free Languages

Collecting together all results on regular top-context-free languages obtained so far, we can state the following

Theorem 21 *For a regular language L the following statements are equivalent:*

(1) L is linear top-context-free.
(2) L is top-context-free.
(3) L is slim.
(4) L can be generated by a non-branching regular grammar. Moreover, every reduced regular grammar generating L is non-branching.
(5) L can be represented by a linear regular expression.
(6) L is passable.
(7) L is polynomially size-bounded.

Proof. Trivially (1) ⇒ (2), and (2) ⇒ (3) by lemma 7. (3) ⇒ (4) is lemma 12 and (4) ⇒ (1) by lemma 14. (4) ⇒ (5) by lemma 20 and (5) ⇒ (2) by proposition 19. (1) ⇒ (6) by lemma 16 and (6) ⇒ (7) by lemma 3.2 in [13]. Finally, (7) ⇒ (4) is easy to show. Indeed, if L is generated by a reduced branching regular grammar, a sequence of terms of L can be constructed in an obvious way such that their size grows exponentially with respect to their depth. □

Conditions (4) and (5) can be seen as syntactic characterizations of Reg ∩ TopCf = Reg ∩ LinTopCf.

6 Closure Properties

In this chapter we study some closure properties of classes of tree languages. First we just mention a few well-known properties, then concentrate on studying closure of the classes FIN, REG, LINTOPCF, TOPCF, LINCF, CF under replacement, substitution and their iterations. We summarize positive and negative results in tables.

We define (linear) homomorphisms on $\mathcal{T}_\Sigma$ as (linear) *tree homomorphisms* in the usual way [6].

Lemma 22 FIN, REG, LINTOPCF, TOPCF, LINCF, *and* CF *are closed under union, under intersection with regular languages, and under linear homomorphisms.*

TOPCF is even closed under arbitrary homomorphisms, and a proof of its closure properties can be found in [1].

Lemma 23 (Closure under Replacement) *(1)* FIN, REG, LINTOPCF, TOPCF, LINCF, *and* CF *are closed under replacement by* FIN.
(2) LINTOPCF·LINTOPCF = LINTOPCF *and* LINTOPCF·TOPCF = TOPCF.
(3) REG, LINCF, *and* CF *are closed under replacement.*
(4) TOPCF·LINTOPCF $\not\subseteq$ TOPCF.

Proof. (1) Let $G = (\Sigma, N, P, S)$ be a context-free grammar and let L' over Σ' be finite. Then the grammar $(\Sigma \cup \Sigma',\ N,\ \{A(\vec{x}) \to t' \mid A(\vec{x}) \to t \in P,\ t' \in t \cdot_c L'\},\ S)$ generates $\mathcal{L}(G) \cdot_c L'$. This grammar is of the same type as G.

(2) Let $G = (N, \Sigma, P, S)$ be linear top-context-free and $G' = (N', \Sigma', P', S')$ be top-context-free. Without loss of generality let G be slow; this is crucial for the proof, since it guarantees that, if $t \in \mathcal{T}_{N \cup \Sigma}(X)$ is the right-hand side of a rule in a slow linear grammar and y is a variable not occuring in t, then $t \cdot_c y$ is linear.

A top-context-free grammar $\bar{G}$ generating $\mathcal{L}(G) \cdot_c \mathcal{L}(G')$ is constructed as follows: The set of nonterminals in $\bar{G}$ is $N \times (N' \cup \{\text{done}\})$ where "done" is a new unary nonterminal; if A has arity n and A' has arity m, then (A, A') has arity $n + m$. The initial symbol of $\bar{G}$ is (S, S') and $\bar{G}$ contains the following rules:

$$
\begin{array}{ll}
(A, A')(\vec{x}, \vec{y}) \to (A, B')(\vec{x}, \vec{t}) & \text{if } A'(\vec{y}) \to B'(\vec{t}) \text{ is in } P', A \in N, \\
(A, A')(\vec{x}, \vec{y}) \to (A, \text{done})(\vec{x}, t) & \text{if } A'(\vec{y}) \to t \text{ is in } P', t \in \mathcal{T}_{\Sigma'}, A \in N, \\
(A, \text{done})(\vec{x}, y) \to (B, \text{done})(\vec{t}, y) & \text{if } A(\vec{x}) \to B(\vec{t}) \text{ is in } P, \vec{t} \text{ does not contain } c, \\
(A, \text{done})(\vec{x}, y) \to (B, S')(\vec{t} \cdot_c \{y\}) & \text{if } A(\vec{x}) \to B(\vec{t}) \text{ is in } P, \vec{t} \text{ contains } c, \\
(A, \text{done})(\vec{x}, y) \to t \cdot_c \{y\} & \text{if } A(\vec{x}) \to t \text{ is in } P, t \in \mathcal{T}_\Sigma.
\end{array}
$$

In order to proof that $\bar{G}$ generates $\mathcal{L}(G) \cdot_c \mathcal{L}(G')$ use the fact that $(A, S')(\vec{t}) \to^*_{\bar{G}} (A, \text{done})(\vec{t}, t')$ implies $t' \in \mathcal{L}(G')$. Moreover, as G is linear, there are no copies of subterms, which could cause c's at different positions to be always instantiated by the same terms. Note also that $\bar{G}$ is linear if G' is linear.

(3) Let $G = (\Sigma, N, P, S)$ and $G' = (\Sigma', N', P', S')$ be context-free grammars with $N \cap N' = \emptyset$. Then the context-free grammar

$$(\Sigma \cup \Sigma',\ N \cup N',\ \{A(\vec{x}) \to t \cdot_c \{S'\} \mid A(\vec{x}) \to t \in P\} \cup P',\ S)$$

generates $\mathcal{L}(G) \cdot_c \mathcal{L}(G')$. It is regular (linear), if both G and G' are regular (linear).

(4) Let L be generated by $\{S \to A(c),\ A(x) \to A(f(x,x)),\ A(x) \to x\}$. Let $L' = \{g^i(a) \mid i \geq 0\}$ be generated by $\{S \to A(c),\ A(x) \to A(g(x)),\ A(x) \to x\}$. Then $L \cdot_c L'$ is not slim, thus not top-context-free, as is easily shown using results of section 4. □

Lemma 24 (Closure under Substitution) *(1)* Fin, Reg, LinTopCf, TopCf, LinCf, *and* Cf *are closed under substitution by* Fin.
(2) TopCf$\diamond$TopCf = TopCf *and* Cf$\diamond$TopCf = Cf.
(3) Fin$\diamond$LinTopCf = LinTopCf.
(4) Fin$\diamond$Reg $\not\subseteq$ Cf.
(5) LinTopCf$\diamond$LinTopCf $\not\subseteq$ LinTopCf.
(6) Reg$\diamond$LinTopCf $\not\subseteq$ LinCf *and* Reg$\diamond$LinTopCf $\not\subseteq$ TopCf.

Proof. (1) First observe that for a ground term t, $\diamond_c\{t\}$ is a linear homomorphism. Hence by lemma 22, $L \in \mathcal{C}$ implies $L\diamond_c\{t\} \in \mathcal{C}$. Now (1) follows from $L\diamond_c L' = \bigcup_{t \in L'}(L\diamond_c\{t\})$ and from the closure under union (lemma 22).

(2) From a context-free grammar G and a top-context-free grammar G' with disjoint sets of nonterminals, a context-free grammar $\bar{G}$ generating $\mathcal{L}(G)\diamond_c\mathcal{L}(G')$ can be constructed as follows: Nonterminals in $\bar{G}$ are the nonterminals of G' together with a nonterminal $\bar{A}$ of arity $n+1$ for each nonterminal A in G of arity n; hence $\bar{S}$ is unary for S, the initial symbol of G. The initial symbol of $\bar{G}$ is S', the initial symbol of G'. The rules of $\bar{G}$ are:

$$\begin{array}{rl} A'(\vec{x}) \to t & \text{if } A'(\vec{x}) \to t \text{ is a rule in } G',\ \text{t contains a nonterminal,} \\ A'(\vec{x}) \to \bar{S}(t) & \text{if } A'(\vec{x}) \to t \text{ is a rule in } G',\ \text{t contains no nonterminal,} \\ \bar{A}(\vec{x}, y) \to h(t)\diamond_c y & \text{if } A(\vec{x}) \to t \text{ is a rule in } G, \end{array}$$

where h is the (linear) homomorphism defined by

$$h(A(t_1, \ldots, t_n)) = \bar{A}(h(t_1), \ldots, h(t_n), c) \text{ for all nonterminals } A \text{ and}$$
$$h(f(t_1, \ldots, t_n)) = f(h(t_1), \ldots, h(t_n)) \text{ for all terminals } f.$$

In the special case where G is top-context-free this yields:

$$\begin{array}{ll} \bar{A}(\vec{x}, y) \to \bar{B}(\vec{t}\diamond_c y, y) & \text{if } A(\vec{x}) \to B(\vec{t}) \text{ is a rule in } G, \\ \bar{A}(\vec{x}, y) \to t\diamond_c y & \text{if } A(\vec{x}) \to t \text{ is a rule in } G,\ \text{t contains no nonterminal.} \end{array}$$

Thus $\bar{G}$ is top-context-free in case G is top-context-free.

(3) Let G be a linear top-context-free grammar, and s a term. Since LinTopCf is closed under union, it is sufficient to show that $s\diamond_c\mathcal{L}(G)$ is generated by a

linear top-context-free grammar $\bar{G}$. $\bar{G}$ is constructed as follows: Let s contain m occurrences of the symbol c. For each nonterminal A in G of arity n we have a nonterminal $\bar{A}$ of arity $m \cdot n$ in $\bar{G}$. If $\vec{x}$ is $x_1, \ldots, x_n$ then let $\pi_1, \ldots, \pi_m$ be variable renamings such that all variables $x_i\pi_j$ are pairwise distinct. Let $\vec{x}\pi_i$ denote $x_1\pi_i, \ldots, x_n\pi_i$, for $\vec{t}$ similarly.

Now, if $A(\vec{x}) \to B(\vec{t})$ is a rule in G, then $\bar{A}(\vec{x}\pi_1, \ldots, \vec{x}\pi_m) \to \bar{B}(\vec{t}\pi_1, \ldots, \vec{t}\pi_m)$ is a rule in $\bar{G}$.

If $A(\vec{x}) \to t$ is a rule in G where t contains no nonterminal, then $\bar{A}(\vec{x}\pi_1, \ldots, \vec{x}\pi_m) \to s'$ is a rule in $\bar{G}$, where s' is obtained from s by replacing the m symbols c successively by $t\pi_1, \ldots, t\pi_m$.

(4) Consider $\mathcal{T}_{\{f,a\}}$. This language is not top-context-free by corollary 10, hence by theorem 1 the language $\{f(c,c)\}\diamond_c\mathcal{T}_{\{f,a\}}$ is not context-free.

(5) Let L be generated by $\{S \to A(c),\ A(x) \to A(f(c,x)),\ A(x) \to x\}$. Let $L' = \{g^i(a) \mid i \geq 0\}$ be generated by $\{S \to A(c),\ A(x) \to A(g(x)),\ A(x) \to x\}$. Then $L\diamond_c L'$ is not linear top-context-free. To show this, an appropriate pumping lemma for linear top-context-free languages can be used, see [8].

(6) Consider $\mathcal{T}_{\{f,c\}}\diamond_c\{g^i(a) \mid i \geq 0\}$. It is neither linear context-free (by a pumping lemma for linear context-free languages given in [8]) nor top-context-free (using results from section 4). □

Lemma 25 (Closure under Replacement Iteration) *(1)* REG, LINCF, *and* CF *are closed under replacement iteration.*
(2) $\text{FIN}^* \not\subseteq$ TOPCF.

Proof. (1) Let G be a context-free grammar. A context-free grammar $\bar{G}$ generating $\mathcal{L}(G)^{*_c}$ is constructed as follows: $\bar{G}$ has the same nonterminals and the same initial symbol S as G, and the rules of $\bar{G}$ are:

$$\begin{aligned} A(\vec{x}) &\to t\cdot_c\{S\} \quad \text{if } A(\vec{x}) \to t \text{ is a rule in G,} \\ S &\to c. \end{aligned}$$

If G is regular or linear then $\bar{G}$ is so.

(2) Consider $\{f(c,c)\}^{*_c} = \mathcal{T}_{\{c,f\}}$; it is not top-context-free by corollary 10. □

Lemma 26 (Closure under Substitution Iteration) *(1)* TOPCF *is closed under substitution iteration.*
(2) $\text{REG}^\diamond \not\subseteq$ CF.
(3) $\text{FIN}^\diamond \not\subseteq$ LINTOPCF.

Proof. (1) Given a top-context-free grammar $G = (N, \Sigma, P, S)$, in order to get a top-context-free grammar $\bar{G}$ generating $\mathcal{L}(G)^{\diamond_c}$ we can use a construction analogous to that given in the proof of lemma 24(2). Just the following rules

have to be added. If $A(\vec{x}) \to t$ is a rule in G, $t \in \mathcal{T}_\Sigma(X)$, we have in $\bar{G}$ not only the rule $\bar{A}(\vec{x}, y) \to t \diamond_c y$, but also the rule

$$\bar{A}(\vec{x}, y) \to \bar{S}(t \diamond_c y).$$

(2) Consider the regular language $L = \{f(c,c)\} \cup \mathcal{T}_\Sigma$ over $\Sigma = \{a, h\}$, where a is a constant and h is binary. We have $L^{\diamond_c} = \{f(c,c)\}^{\diamond_c} \cup (\{f(c,c)\}^{\diamond_c} \diamond_c \mathcal{T}_\Sigma)$. Thus $L^{\diamond_c} \cap \{f(t,t') \,|\, t, t' \in \mathcal{T}_\Sigma\} = \{f(t,t) \,|\, t \in \mathcal{T}_\Sigma\}$, which is not context-free by theorem 1 and corollary 10. As $\{f(t,t') \,|\, t, t' \in \mathcal{T}_\Sigma\}$ is regular and CF is closed under intersection with regular languages, $L^{\diamond_c}$ is not context-free.

(3) For $L = \{f(c,c)\}$, $L^{\diamond_c}$ is the set of complete binary trees over $\{c, f\}$. Thus, by example 17, $L^{\diamond_c} \not\subseteq$ LINTOPCF. □

CLOSURE BY REPLACEMENT: $\mathcal{C} \cdot \mathcal{C}'$

$\mathcal{C}$ \ $\mathcal{C}'$	FIN	REG	LINTOPCF	TOPCF	LINCF	CF
FIN	= FIN 23(1)	= REG 23(3)	= LINTOPCF 23(2)	= TOPCF 23(2)	= LINCF 23(3)	= CF 23(3)
REG	= REG 23(1)	= REG 23(3)	⊆ LINCF 23(3)	⊆ CF 23(3)	= LINCF 23(3)	= CF 23(3)
LINTOPCF	= LINTOPCF 23(1)	⊆ LINCF 23(3)	= LINTOPCF 23(2)	= TOPCF 23(2)	= LINCF 23(3)	= CF 23(3)
TOPCF	= TOPCF 23(1)	⊆ CF 23(3)	⊆ CF 23(3), 23(4)	⊆ CF 23(3), 23(4)	⊆ CF 23(3)	= CF 23(3)
LINCF	= LINCF 23(1)	= LINCF 23(3)	= LINCF 23(3)	⊆ CF 23(3)	= LINCF 23(3)	= CF 23(3)
CF	= CF 23(1), 23(3)	= CF 23(3)	= CF 23(3)	= CF 23(3)	= CF 23(3)	= CF 23(3)

CLOSURE BY SUBSTITUTION: $\mathcal{C} \diamond \mathcal{C}'$

$\mathcal{C}$ \ $\mathcal{C}'$	FIN	REG	LINTOPCF	TOPCF	LINCF	CF
FIN	= FIN 24(1)	⊈ CF 24(4)	= LINTOPCF 24(3)	= TOPCF 24(2)	⊈ CF 24(4)	⊈ CF 24(4)
REG	= REG 24(1)	⊈ CF 24(4)	⊆ CF 24(2), 24(6)	⊆ CF 24(2), 24(6)	⊈ CF 24(4)	⊈ CF 24(4)
LINTOPCF	= LINTOPCF 24(1)	⊈ CF 24(4)	⊆ TOPCF 24(2), 24(5)	= TOPCF 24(2)	⊈ CF 24(4)	⊈ CF 24(4)
TOPCF	= TOPCF 24(1)	⊈ CF 24(4)	= TOPCF 24(2)	= TOPCF 24(2)	⊈ CF 24(4)	⊈ CF 24(4)
LINCF	= LINCF 24(1)	⊈ CF 24(4)	⊆ CF 24(2), 24(6)	⊆ CF 24(2), 24(6)	⊈ CF 24(4)	⊈ CF 24(4)
CF	= CF 24(1)	⊈ CF 24(4)	= CF 24(2)	= CF 24(2)	⊈ CF 24(4)	⊈ CF 24(4)

$\mathcal{C}$	Closure by Replacement Iteration: $\mathcal{C}^*$	Substitution Iteration: $\mathcal{C}^\diamond$
Fin	$\subseteq$ Reg 25(1), 25(2)	$\subseteq$ TopCf 26(1), 26(3)
Reg	$=$ Reg 25(1)	$\not\subseteq$ Cf 26(2)
LinTopCf	$\subseteq$ LinCf 25(1), 25(2)	$\subseteq$ TopCf 26(1), 26(3)
TopCf	$\subseteq$ Cf 25(1), 25(2)	$=$ TopCf 26(1)
LinCf	$=$ LinCf 25(1)	$\not\subseteq$ Cf 26(2)
Cf	$=$ Cf 25(1)	$\not\subseteq$ Cf 26(2)

References

1. A. Arnold and M. Dauchet. Transductions de Forêts Reconnaissables Monadiques. Forêts Corégulières. *RAIRO Informatique Théorique et applications*, 10(3):5–28, 1976.
2. A. Arnold and M. Dauchet. Un Théorème de Duplication pour les Forêts Algébriques. *JCSS*, 13:223–244, 1976.
3. M. Dauchet and S. Tison. Structural Complexity of Classes of Tree Languages. In *Tree Automata and Languages*, pp. 327–353. Elsevier (North-Holland), 1992.
4. N. Dershowitz and J.-P. Jouannaud. Rewrite Systems. In *Handbook of Theoretical Computer Science*, vol. B, pp. 243–320. Elsevier, 1990.
5. J. Engelfriet and E. Schmidt. IO and OI. I. *JCSS*, 15:329–353, 1977.
6. F. Gécseg and M. Steinby. *Tree automata*. Akadémiai Kiadó, Budapest, 1984.
7. D. Hofbauer and M. Huber. Computing linearizations using test sets. In *3rd International Workshop on Conditional Term Rewriting Systems*, LNCS 656, pp. 287–301. Springer-Verlag, 1992.
8. D. Hofbauer, M. Huber, and G. Kucherov. Some Results on Top-context-free Tree Languages. Centre de Recherche en Informatique de Nancy, Technical Report, to appear, 1993.
9. J. Hopcroft and J. Ullman. *Introduction to Automata Theory. Languages and Computation*. Addison-Wesley, 1979.
10. G. Kucherov and M. Tajine. Decidability of regularity and related properties of ground normal form languages. In *3rd International Workshop on Conditional Term Rewriting Systems*, LNCS 656, pp. 272–286. Springer-Verlag, 1992.
11. G. A. Kucherov. On relationship between term rewriting systems and regular tree languages. In *4th Conference on Rewriting Techniques and Applications*, LNCS 488, pp. 299–311. Springer-Verlag, 1991.
12. T. S. E. Maibaum. Pumping lemmas for term languages. *JCSS*, 17:319–330, 1978.
13. K. Salomaa. Deterministic Tree Pushdown Automata and Monadic Tree Rewriting Systems. *JCSS*, 37:367–394, 1988.
14. S. Vágvölgyi and R. Gilleron. For a rewriting system it is decidable whether the set of irreducible ground terms is recognizable. *Bulletin of the European Association for Theoretical Computer Science*, 48:197–209, 1992.

This article was processed using the LaTeX macro package with LLNCS style