Theoretical Computer Science 410 (2009) 3198–3208

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

On the synchronized derivation depth of context-free grammars☆

Franziska Biegler[a,*,1], Kai Salomaa[b,1]

[a] Department of Computer Science, University of Western Ontario, London, ON, N6A 5B7, Canada
[b] School of Computing, Queen's University, Kingston, ON K7L 3N6, Canada

ARTICLE INFO

Keywords:
Synchronization
Context-free languages
Regulated rewriting
Derivation complexity

ABSTRACT

We consider depth of derivations as a complexity measure for synchronized and ordinary context-free grammars. This measure differs from the earlier considered synchronization depth in that it counts the depth of the entire derivation tree. We consider (non-)existence of trade-offs when using synchronized grammars as opposed to non-synchronized grammars and establish lower bounds for certain classes of linear context-free languages.

1. Introduction

Context-free languages are among the best studied and understood families of formal languages. Unfortunately, their generative power is insufficient to model many important phenomena of formal and natural languages, see e.g. [1]. Context-sensitive languages, the next level in the Chomsky-hierarchy, are so powerful that they become difficult to handle. For this reason different extensions of context-free grammars have been proposed, see e.g. [1–3], in order to increase the generative capacity while maintaining as many of the desired properties of context-free languages as possible.

In [4], H. Jürgensen and K. Salomaa introduced a new extension of context-free grammars, synchronized context-free (SCF) grammars as well as block-synchronized context-free (BSCF) grammars, in which independent paths in a context-free derivation can communicate in order to synchronize. Different aspects of SCF and BSCF grammars were studied in [4–7]. Measuring the amount of synchronization in SCF grammars and languages by functions was done in [8,9], where the number of situation symbols and the lengths of the situation sequences were used as measures. The idea of synchronization as a method of communication was proposed in a similar way for automata in [10].

In this paper we now count the total depth of the derivation trees, i.e. the non-synchronized nonterminals are counted as well. For each context-free language this yields two different depth functions, one if the usage of situation symbols is allowed, and another (potentially different) one if the grammar has to be context-free. Please recall, that the depth and count functions for context-free languages as discussed in [9,8] are always constant, due to the fact that only synchronized nonterminals are being counted.

We investigate for which classes of languages the added synchronization feature introduces significant savings with respect to the depth of derivation trees. We show that non-bounded context-free languages that are structured in a certain way always require linear depth when generated by a context-free grammar. We also answer an open problem from [9] by providing an SCF grammar with a depth synchronization function outside of the known hierarchy.

Similar measures of derivation complexity were also discussed in [11–13], and it was conjectured in [12] that all context-free non-regular languages need linear derivation depth. In [14] this conjecture was disproven and a hierarchy of context-free languages that require sublinear depth was given.

☆ A preliminary version of this paper appeared in DCFS 2008 [F. Biegler, K. Salomaa, On the synchronized derivation depth of context-free grammars, in: C. Câmpeanu, G. Pighizzini (Eds.), 10th International Workshop on Descriptional Complexity of Formal Systems, Charlottetown, Canada, 2008, pp. 73–84].

* Corresponding author. Tel.: +1 5196612111.
E-mail addresses: fbiegler@csd.uwo.ca (F. Biegler), ksalomaa@cs.queensu.ca (K. Salomaa).

1 Research supported, in part, by the Natural Sciences and Engineering Research Council of Canada.

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.03.031

2. Preliminaries

Let $\mathbb{N}$ and $\mathbb{N}^+$ be the set of non-negative and positive integers, respectively, and let $\mathbb{R}^+$ be the set of non-negative real numbers. An alphabet A is a finite, non-empty set of symbols. The set of all words over A is denoted by A^*, and this set contains the empty word, λ. The set of all words over A of length at most m for some $m \geq 0$ is denoted by $A^{\leq m}$. A language L over A is any subset of A^*. For a word $x \in A^*$ let $|x|$ denote the length of x. We say x is a prefix of y, denoted by $x \leq_p y$ if $y = xu$ for some word $u \in A^*$. For a word $|w| = n$ and $1 \leq k \leq n$ we denote by $w(k)$ the kth letter of w. We use $\subseteq$, $\subset$ and $\setminus$ to denote subset, proper subset and set difference.

For alphabets A and B, a morphism (anti-morphism) from A to B is a mapping $\varphi : A^* \to B^*$ ($\theta : A^* \to B^*$) with $\varphi(\lambda) = \lambda$ and $\varphi(uv) = \varphi(u)\varphi(v)$ ($\theta(\lambda) = \lambda$ and $\theta(uv) = \theta(v)\theta(u)$) for all $u, v \in A^*$. If $A \subseteq B$ we denote by $\pi_A : B^* \to A^*$ the projection onto A. A morphism $\varphi : A^* \to B^*$ is called *linearly erasing* if and only if there exists a constant $c < 1$, such that for each word $w \in A^*$, φ erases at most $c \cdot |w|$ symbols.

A *tree domain* D is a non-empty finite subset of $\mathbb{N}^*$ such that

(1) If $\mu \in D$, then every prefix of μ belongs to D.
(2) For every $\mu \in D$ there exists $i \geq 0$ such that $\mu j \in D$ if and only if $1 \leq j \leq i$.

Let A be a set. An *A-labelled tree* is a mapping $t : D \to A$, where D is a tree domain. Elements of D are called nodes of t and D is said to be the domain of t, $\mathrm{dom}(t)$. The node $\mu \in \mathrm{dom}(t)$ is labelled by $t(\mu)$. The node $\lambda \in \mathrm{dom}(t)$, denoted by $\mathrm{root}(t)$, is called the root of t. A node $\mu \in \mathrm{dom}(t)$ is called a *leaf* of t if and only if $\mu j \notin \mathrm{dom}(t)$ for all $j \geq 1$. The set of leaves of t is denoted by $\mathrm{leaf}(t)$. The subtree of t at node μ is t/μ. The set of subtrees of t is $\mathrm{sub}(t) = \{t/\mu \mid \mu \text{ is a node of } t\}$ which we extend to sets of trees T by $\mathrm{sub}(T) = \bigcup_{t \in T} \mathrm{sub}(t)$. Note that there could be several isomorphic subtrees of t. When there is no confusion, we refer to a node simply by its label.

Nodes of a tree t that are not leaves are called *inner nodes* of t. The *inner tree of t*, $\mathrm{inner}(t)$ is the tree obtained from t by cutting off all the leaves. The *yield* of an A-labelled tree t, $\mathrm{yd}(t)$, is the word obtained by concatenating the labels of the leaves of t from left to right; the leaves are ordered by the lexicographic ordering of $\mathbb{N}^*$. For $\mu \in \mathrm{dom}(t)$, $\mathrm{path}_t(\mu)$ is the sequence of tree nodes occurring on the path from the root of t to the node μ. By abuse of notation and when there is no potential for confusion we also use $\mathrm{path}_t(\mu)$ to refer to the sequence of symbols from A occurring on the path from the root of t to the node μ. The size of a tree t, $\mathrm{size}(t)$, is the number of nodes in the tree, and the depth of t is the length of the longest path in t, i.e. $\mathrm{depth}(t) = \max\{|\mathrm{path}_t(\mu)| \mid \mu \in t\}$.

A *context-free grammar* is denoted by $G = (N, T, P, I)$, where N and T are disjoint alphabets of nonterminals and terminals respectively, $I \in N$ is the initial nonterminal, and P is a finite set of productions of the form $X \to w$ where $X \in N$ and $w \in (N \cup T)^*$. Derivations of context-free grammars can be represented as trees.

Let $G = (N, T, P, I)$ be a CF grammar. A $(N \cup T \cup \{\lambda\})$-labelled tree t is a *derivation tree of G* if it satisfies the following conditions.

(1) The root of t is labelled by the initial nonterminal, that is, $t(\lambda) = I$.
(2) The leaves of t are labelled by terminals or by the symbol λ.
(3) Let $\mu \in \mathrm{dom}(t)$ have k immediate successors $\mu_1, \ldots, \mu_k$, $k \geq 1$. Then $t(\mu) \to t(\mu_1) \cdots t(\mu_k) \in P$. If one of the successors is labelled by λ, then $k = 1$ and $t(\mu) \to \lambda \in P$.

The set of derivation trees of G is denoted by $T(G)$. The derivation trees of G are in one-to-one correspondence with the equivalence classes of derivations of G producing terminal words, and thus $L(G) = \{\mathrm{yd}(t) \mid t \in T(G)\}$. Above, in the word $\mathrm{yd}(t)$, we identify occurrences of the symbol λ with the empty word.

We use a regular expression and the language generated by it synonymously throughout this paper.

The family of regular and context-free languages are denoted by $\mathcal{L}(\mathrm{REG})$ and $\mathcal{L}(\mathrm{CF})$, respectively [15–17]. The family of ET0L (extended tabled Lindenmayer systems without interaction) languages is denoted by $\mathcal{L}(\mathrm{ET0L})$ (see [18,19]).

We use asymptotic representations of functions as defined in [20]. As the definition of Ω given in other publications might differ we explicitly define only Ω, please see [20] for the definitions of $\mathcal{O}$ and Θ. Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+$ be functions, then we say $g \in \Omega(f)$ if and only if there exist constants $c > 0$ and $n_0 \in \mathbb{N}$, such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$. The function f is called an upper, lower or tight bound of g if $g \in \mathcal{O}(f)$, $g \in \Omega(f)$ or $g \in \Theta(f)$, respectively.

3. Synchronized context-free grammars and languages

In this section the definitions of synchronized context-free grammars as originally given in [4] are stated and the measures of derivation complexity that are investigated in this paper are defined.

Definition 1. A synchronized context-free grammar (shortly SCF) is a five-tuple

$$G = (V, S, T, P, I)$$

such that $G' = (V \times (S \cup \{\lambda\}), T, P, I)$ is a context-free grammar and V, S and T are the alphabets of base nonterminals, situation symbols and terminals, respectively. The alphabet of nonterminals is $V \times (S \cup \{\lambda\})$, where elements of $V \times S$ are called *synchronized nonterminals* and elements of $V \times \{\lambda\}$ are called *non-synchronized nonterminals* which are usually denoted by their base nonterminals only. We define the morphism $h_G : (V \times (S \cup \{\lambda\}))^* \longrightarrow S^*$ by $h_G((v, x)) = x$ for all $v \in V$ and $x \in S \cup \{\lambda\}$.

Essentially an SCF grammar is a context-free grammar where the nonterminals are pairs of base nonterminals and situation symbols. The sequences of situation symbols are used as a method of limited communication. A tree t is a derivation tree of an SCF grammar if it is a derivation tree of the underlying context-free grammar.

Definition 2. Let G be an SCF grammar. For a derivation tree t of G, $t_1 = \text{inner}(t)$ and a node $\mu \in \text{leaf}(t_1)$, the synchronizing sequence (sync-sequence) corresponding to μ is $\text{seq}_{t_1}(\mu) = h_G(\text{path}_{t_1}(\mu))$. Also, define $\text{seq}_t = \{s \mid \text{seq}_{t_1}(\mu) = s, \mu \in \text{leaf}(t_1)$ and $s' \in \text{seq}_{t_1}(\mu')$, $\mu' \in \text{leaf}(t_1)$ implies $|s'| \leq |s|\}$. If this set is a singleton, we use seq_t to refer to the element in the set.

Now we define which derivation trees of the underlying context-free grammar are considered valid derivations of an SCF grammar. There are two derivation modes for SCF grammars, namely equality and prefix mode. In the former the situation sequences along all paths of the inner tree have to be equal, while the situation sequences of the inner tree have to be pairwise in prefix relation in the latter mode. We say, $w_1 \simeq_\text{p} w_2$ ($w_1 \simeq_\text{e} w_2$) if and only if w_1 and w_2 are in prefix relation (are equal).

Definition 3. Let $G = (V, S, T, P, I)$ be an SCF grammar and $z \in \{\text{p}, \text{e}\}$. A derivation tree t of G is said to be z-acceptable if, for each $\mu, \nu \in \text{leaf}(\text{inner}(t))$, $\text{seq}_{\text{inner}(t)}(\mu) \simeq_z \text{seq}_{\text{inner}(t)}(\nu)$. The set of z-acceptable derivation trees of G is denoted by $T_z(G)$.

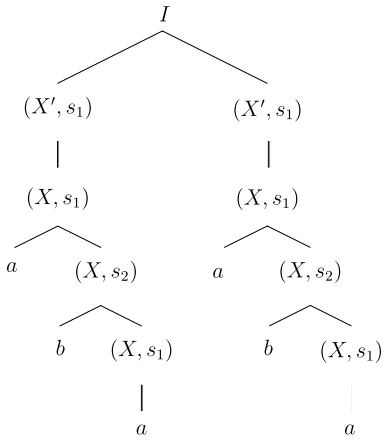
Definition 4. For $z \in \{\text{p}, \text{e}\}$, the z-synchronized language of G is $L_z(G) = \text{yd}(T_z(G))$. The families of z-SCF languages, for $z \in \{\text{p}, \text{e}\}$, and SCF languages are denoted by $\mathcal{L}_z(\text{SCF})$ and $\mathcal{L}(\text{SCF}) = \mathcal{L}_\text{e}(\text{SCF}) \cup \mathcal{L}_\text{p}(\text{SCF})$.

Notice that if t is an e- or p-acceptable derivation, then seq_t is a singleton.
The following example should help us to clarify the definitions.

Example 5. Let $G = (\{I, X, X'\}, \{s_1, s_2\}, \{a, b\}, P, I)$ be an SCF grammar where P contains, for $i, j \in \{1, 2\}$, the following productions:

$$I \to (X', s_i)(X', s_i), \qquad (X', s_i) \to (X, s_i) \mid \lambda$$
$$(X, s_1) \to a(X, s_i) \mid a, \qquad (X, s_2) \to b(X, s_i) \mid b.$$

The following tree is a derivation tree of the underlying context-free grammar:



This tree is e- and p-synchronized with respect to G, as all the situation sequences are equal. In e-mode G generates the non-context-free language $L_\text{e}(G) = \{ww \mid w \in \{a, b\}^*\}$ and in p-mode G generates the regular language $L_\text{p}(G) = \{a, b\}^*$.

It was proven in [4] that SCF grammars in p- and e-synchronization mode generate the same family of languages, i.e. $\mathcal{L}_\text{e}(\text{SCF}) = \mathcal{L}_\text{p}(\text{SCF}) = \mathcal{L}(\text{SCF})$. In [6] it was proven that SCF grammars generate the family of ET0L languages, i.e. $\mathcal{L}(\text{SCF}) = \mathcal{L}(\text{ET0L})$, and that given an SCF grammar and a derivation mode one can effectively construct an equivalent ET0L system and vice versa. The length synchronization context-free grammars of [21] have the same generative capacity.

4. Derivation complexity

We define four new measures for the descriptional complexity of context-free and synchronized context-free grammars and languages, the count and the depth in either synchronized or non-synchronized mode.

Definition 6. Let t be a derivation tree. Then $||t||_\text{c}$ is the number of nodes in $\text{inner}(t)$ and $||t||_\text{d}$ is the length of the longest path in $\text{inner}(t)$, i.e. the depth of the tree.

Definition 7. Let G be a context-free grammar. The non-synchronized derivation count (respectively depth), indicated by $y \in \{c, d\}$, of a word $w \in L(G)$ is

$$(\not{s}, y)\text{-der}_G(w) = \begin{cases} \min\{||t||_y \mid t \in T(G), \text{yd}(t) = w\}, & w \in L(G), \\ 0, & w \in T^* \setminus L(G). \end{cases}$$

Definition 8. Let G be a synchronized context-free grammar and let $z \in \{e, p\}$. The synchronized derivation count (respectively depth), indicated by $y \in \{c, d\}$, of a word $w \in L_z(G)$ is

$$(s, y)_z\text{-der}_G(w) = \begin{cases} \min\{||t||_y \mid t \in T_z(G), \text{yd}(t) = w\}, & w \in L_z(G), \\ 0, & w \in T^* \setminus L_z(G). \end{cases}$$

The above definitions are extended to grammars and languages similar to [9].

Definition 9. Let G be a context-free grammar. The non-synchronized count (respectively depth) derivation function $(\not{s}, y)\text{-der}_G : \mathbb{N} \to \mathbb{N}$, $y \in \{c, d\}$, of G, is defined as

$$(\not{s}, y)\text{-der}_G(n) = \max\{(\not{s}, y)\text{-der}_G(w) \mid |w| \leq n\}.$$

Definition 10. Let G be an SCF grammar. The synchronized count (respectively depth) derivation function $(s, y)_z\text{-der}_G : \mathbb{N} \to \mathbb{N}$, of G in z-mode, $z \in \{p, e\}$, $y \in \{c, d\}$, is defined as

$$(s, y)_z\text{-der}_G(n) = \max\{(y, z)\text{-der}_G(w) \mid |w| \leq n\}.$$

Note that, for an SCF grammar G, $z \in \{p, e\}$ and $y \in \{c, d\}$, both functions $(s, y)_z\text{-der}_G(n)$ and $(\not{s}, y)\text{-der}_G(n)$ are always monotonically increasing.

In order to make the following definitions more compact we sometimes add the subscript z to the non-synchronized functions without changing the meaning.

Definition 11. Let $L \in \mathcal{L}(\text{SCF})$ and $z \in \{e, p\}$, $y \in \{c, d\}$, $x \in \{s, \not{s}\}$. Let $f : \mathbb{N} \to \mathbb{N}$ be a function. We say that the $(x, y)_z$-synchronization measure of L has

- upper bound $f(n)$, denoted by $(x, y)_z\text{-der}_L \in \mathcal{O}(f(n))$, if there exists a grammar G, such that $L_z(G) = L$ and $(x, y)_z\text{-der}_G \in \mathcal{O}(f(n))$;
- lower bound $f(n)$, denoted by $(x, y)_z\text{-der}_L \in \Omega(f(n))$, if for all grammars G with $L_z(G) = L$ we have $(x, y)_z\text{-der}_G \in \Omega(f(n))$;

We furthermore say that f is an $(x, y)_z$-derivation representative of L, denoted by $(x, y)_z\text{-der}_L \in \Theta(f(n))$, if $(x, y)_z\text{-der}_L \in \mathcal{O}(f(n))$ and $(x, y)_z\text{-der}_L \in \Omega(f(n))$.

Note that the existence of a representative grammar for each language does not follow from the definition.

4.1. Overview of synchronization functions

The synchronization measures defined in [8,9] measure the derivation complexity of SCF grammars by counting the synchronized nonterminals in the longest path (for depth-synchronization functions) or by counting the synchronized nonterminals in the entire tree (for count-synchronization functions). Note that this differs from Definitions 9 and 10 both of which measure the size or depth of the entire derivation tree of a synchronized grammar.

Analogously with Definitions 10 and 11, the synchronization measures of [8,9] have four variants for any SCF language L, namely,

$$(y, z)\text{-synch}_L$$

for $y \in \{c, d\}$, representing count and depth as above, and $z \in \{e, p\}$, representing prefix and equality synchronization modes as above. For the formal definitions see [9].

The known upper and lower bounds for synchronization measures are summed up in the following theorem.

Theorem 12. *Let $L \in \mathcal{L}(\text{SCF})$. For $y \in \{c, d\}$, $z \in \{e, p\}$, (y, z)-synch$_L$ is bounded by a constant if and only if L is context-free. If L is not context-free, then, for $z \in \{e, p\}$, we have*

$$\begin{array}{ll} (c, z)\text{-synch}_L \in \Omega(n); & (c, z)\text{-synch}_L \in \mathcal{O}(n^2); \\ (d, z)\text{-synch}_L \in \Omega(\log n); & (d, z)\text{-synch}_L \in \mathcal{O}(n). \end{array}$$

For depth synchronization measures a strict hierarchy was shown to exist in [9].

Theorem 13. *Let $L_0 = \{w\$w \mid w \in \{0, 1, \#\}^*\}$. Then $(d, z)\text{-synch}_{L_0} \in \Theta(n)$. There exists a language $L \in \mathcal{L}(\text{SCF})$, such that, for $z \in \{e, p\}$, $(d, z)\text{-synch}_L \in \Theta(\log n)$. Furthermore for $k \geq 1$, there exists a language $L_k \in \mathcal{L}(\text{SCF})$ such that for $z \in \{e, p\}$, $(d, z)\text{-synch}_{L_k} \in \Theta(n^{\frac{1}{k}})$.*

In [9] it was left as an open problem whether there even exists an SCF *grammar* with a synchronization function different from the functions stated in the previous theorem. In Section 5 we show that there exists such a grammar.

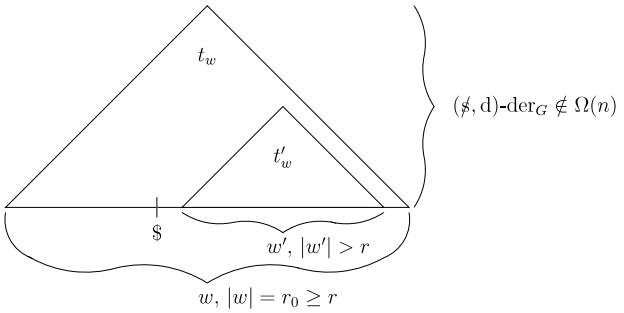


Fig. 1. A derivation tree illustrating Lemma 18.

4.2. Partial characterization results

The following result can be proven easily by modifying the proofs of some results in [8,9] for our current definitions of depth and count.

Theorem 14. *Let $x \in \{\mathrm{s}, \not\$\}$, $y \in \{\mathrm{c}, \mathrm{d}\}$, $z \in \{\mathrm{e}, \mathrm{p}\}$ and let L be an* SCF *language. Then*

(1) *L is finite if and only if $(x, y)_z\text{-der}_L \in \mathcal{O}(1)$;*
(2) *for each infinite $L \in \mathcal{L}(\mathrm{CF})$ we have, $(\not\$, \mathrm{d})\text{-der}_L \in \Omega(\log n)$, $(\not\$, \mathrm{d})\text{-der}_L \in \mathcal{O}(n)$;*
(3) *for each infinite $L \in \mathcal{L}(\mathrm{SCF})$ and $z \in \{\mathrm{e}, \mathrm{p}\}$ we have, $(\mathrm{s}, \mathrm{d})_z\text{-der}_L \in \Omega(\log n)$, $(\mathrm{s}, \mathrm{d})_z\text{-der}_L \in \mathcal{O}(n)$ and $(\mathrm{s}, \mathrm{c})_z\text{-der}_L \in \Omega(n)$, $(\mathrm{s}, \mathrm{c})_z\text{-der}_L \in \mathcal{O}(n^2)$.*

In the case of non-synchronized count-derivation functions we obtain stronger results than those known for count-synchronization functions [8].

Theorem 15. *For each infinite $L \in \mathcal{L}(\mathrm{CF})$ we have $(\not\$, \mathrm{c})\text{-der}_L \in \Theta(n)$.*

Proof. It is obvious that $(\not\$, \mathrm{c})\text{-der}_L \in \Omega(n)$. Furthermore, every context-free grammar G in Chomsky normal form has $(\not\$, \mathrm{c})\text{-der}_G \in \Theta(n)$. □

We now establish a connection between the measures used in this paper and the ones used in [8,9].

Theorem 16. *Let $L \in \mathcal{L}(\mathrm{SCF})$ and let $f : \mathbb{N} \to \mathbb{N}$ be a function. Then for $y \in \{\mathrm{c}, \mathrm{d}\}$, $z \in \{\mathrm{e}, \mathrm{p}\}$, $(\mathrm{s}, y)_z\text{-der}_G \in \mathcal{O}(f)$ implies $(y, z)\text{-synch}_L \in \mathcal{O}(f)$.*

Proof. There exists an SCF grammar G for L with $(\mathrm{s}, y)_z\text{-der}_G \in \mathcal{O}(f)$. Then obviously $(y, z)\text{-synch}_G \in \mathcal{O}(f)$, which implies the statement. □

It is obvious from Lemma 14 that the converse implication does not hold for any context-free language. We also provide a counterexample for the converse implication for non-context-free languages at the end of Section 4.3.

The following facts are known from [12].

Theorem 17 (*[12]*). (1) *Let $L \in \mathcal{L}(\mathrm{REG})$ be infinite. Then, for $x \in \{\mathrm{s}, \not\$\}$, $z \in \{\mathrm{e}, \mathrm{p}\}$, we have $(x, \mathrm{d})_z\text{-der}_L \in \Theta(\log n)$.*
(2) *There exists a context-free language L such that $(\mathrm{s}, \mathrm{d})\text{-der}_L \in \Theta(n)$.*

In [12,13] it is conjectured that all context-free non-regular languages require linear derivation depth when generated with a context-free grammar.

We proceed to extending the results from [13,14] by showing that a certain subclass of context-free languages always need context-free grammars with linear depth to generate them. For each regular language Q, we define $L_Q = \{ww^R \mid w \in Q\}$ and $L_Q^{\$} = \{w\$w^R \mid w \in Q\}$. It is obvious that L_Q and $L_Q^{\$}$ are linear context-free languages and that $L_Q^{\$}$ is non-regular whenever Q is infinite.

Lemma 18. *Let $L = \{\varphi_1(w)\$\varphi_2(w^R) \mid w \in Q\}$ for some infinite regular language Q and morphisms φ_1, φ_2, and let G be a context-free grammar with $L(G) = L$.*

If $(\not\$, \mathrm{d})\text{-der}_G \notin \Omega(n)$, then for all $r \in \mathbb{N}$ there exists $r_0 \geq r$ and there exists $w \in L$ with $|w| = r_0$, such that the optimal derivation tree t_w for w has a subtree t'_w with $r < |\mathrm{yd}(t'_w)|$ and $\$ \notin \mathrm{yd}(t'_w)$ (Fig. 1).

Proof. Without loss of generality assume G to be in Chomsky normal form (which only increases the depth by a linear factor). We have $(\not\$, \mathrm{d})\text{-der}_G \notin \Omega(n)$, i.e.

$$\forall c > 0, \forall k_0 \in \mathbb{N}, \exists k \geq k_0 \text{ such that } (\not\$, \mathrm{d})\text{-der}_G(k_0) < c \cdot k.$$

Assume that the lemma statement is not true. Then there exists an $r \in \mathbb{N}$, such that for all $r_0 \geq r$ and all w with $|w| = r_0$, the optimal derivation tree t_w for w does not have any subtree t'_w with $r < |\mathrm{yd}(t'_w)|$ and $\$ \notin \mathrm{yd}(t'_w)$.

In other words, if t'_w is a subtree of t_w, then t'_w does not satisfy both $r < |\mathrm{yd}(t'_w)|$ and $\$ \notin \mathrm{yd}(t'_w)$. Consider the path from the root of t_w to the node μ that generates the leaf labelled by \$. Then, because of Chomsky normal form, there are exactly $p = |\mathrm{path}_{t_w}(\mu)|$ nodes in the tree that are direct children of nodes in $\mathrm{path}_{t_w}(\mu)$ and all these nodes are labelled with nonterminals. Let T' be the set of subtrees of t_w having one of these nodes as the root. For all trees $t' \in T'$ we have $|\mathrm{yd}(t')|_\$ = 0$, thus we must have $r > |\mathrm{yd}(t')|$. But then the longest path through the derivation tree t_w has to have at least length $p > \frac{|w|}{r}$. Thus, for all words $w \in L$ with $|w| \geq r$, we have $(\not\$, \mathrm{d})\text{-der}_G(w) \geq \frac{|w|}{r}$, which implies $(\not\$, \mathrm{d})\text{-der}_G \in \Omega(n)$, a contradiction. □

Theorem 19. *Let Q be an infinite regular language. Then $(\not\$, \mathrm{d})\text{-der}_{L_Q^\$} \in \Omega(n)$.*

Proof. Assume that $(\not\$, \mathrm{d})\text{-der}_{L_Q^\$} \notin \Omega(n)$. Then, by Lemma 18, there exists an infinite number of subtrees of derivation trees that do not contain the \$ sign in their yield. As the set of nonterminal symbols is finite, this implies that there exists a nonterminal A, such that there are derivation trees $t_1, t_2 \in T(G)$, with subtrees t'_1, t'_2, respectively, such that the root of both t'_1 and t'_2 is labelled A and $\mathrm{yd}(t'_1) \neq \mathrm{yd}(t'_2)$ and neither $\mathrm{yd}(t'_1)$ nor $\mathrm{yd}(t'_2)$ contain the \$ sign.

Then we can substitute t'_2 for t'_1 in t_1, thus creating another derivation tree $t_{12} \in T(G)$. But $\mathrm{yd}(t_{12}) \notin L_Q^\$$, a contradiction. □

By the linear upper bound of Theorem 14 we obtain the following corollaries.

Corollary 20. *For every linear language L' that can be written as $L' = L_Q^\$$ for some infinite regular language Q, we have $(\not\$, \mathrm{d})\text{-der}_{L'} \in \Theta(n)$.*

We also obtain the following.

Corollary 21. *Let L' be a linear language such that $L' = \{\varphi_1(w)\$\varphi_2(w^R)\}$ for some infinite regular language Q and linearly-erasing morphisms φ_1 and φ_2. Then we have $(\not\$, \mathrm{d})\text{-der}_{L'} \in \Theta(n)$.*

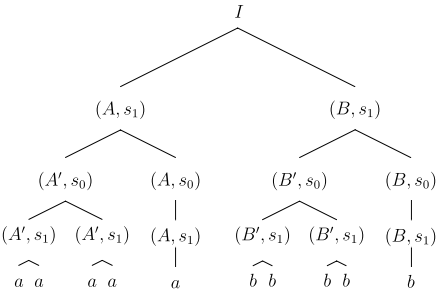
4.3. SCF grammars for CF languages

We now look at the total depth in SCF grammars which generate context-free languages. This can tell us how much a context-free language can be compressed by using an SCF grammar to generate it. It was shown in [12] that the amount of compression (with respect to the depth) varies immensely depending on the language. Namely $L_1 = \{a^n b^n \mid n \in \mathbb{N}\}$ can be compressed to having logarithmic depth when using an SCF grammar (they show this for an E0L system), while $L_2 = \{w\$w^R \mid w \in \{0,1\}^*\}$ always requires linear depth to be generated even if an SCF grammar is used (they show this for context-sensitive grammars). The following example gives an SCF grammar which generates L_1 with logarithmic depth.

Example 22. Let $G = (V, S, T, P, I)$ by $V = \{I, A, B, A', B'\}$, $S = \{s_0, s_1\}$, $T = \{a, b\}$ and the following productions are, for $i, j \in \{0,1\}$ and $X \in \{A, B\}$, in P.

$$I \to (A, s_i)(B, s_i) \qquad (X, s_0) \to (X, s_i) \mid \lambda$$
$$(X, s_1) \to (X', s_i)(X, s_i) \mid x \qquad (X', s_i) \to (X', s_j)(X', s_j) \mid xx.$$

The following tree t is an e-synchronized derivation tree of G.



We have $\mathrm{yd}(t) = a^5 b^5$ and t has situation sequence $s_1 s_0 s_1$. In e-mode, G generates $L_e(G) = \{a^n b^n \mid n \in \mathbb{N}\}$ and, as n is encoded in binary, we have $(\mathrm{s}, \mathrm{d})_e\text{-der}_G \in \Theta(\log n)$.

It is obvious that we can also generate all non-context-free languages of the form $a_1^n a_2^n \cdots a_k^n$ with logarithmic synchronized derivation depth in a similar way.

The following is an alternative construction to establish $(\mathrm{s}, \mathrm{d})_z\text{-der}_{L_2} \in \Theta(n)$ by using the result $(\mathrm{d}, z)\text{-synch}_{L'_2} \in \Theta(n)$ from [9], where $L'_2 = \{w\$w \mid w \in \{0,1\}^*\}$.

Lemma 23. $(\mathrm{s}, \mathrm{d})_z\text{-der}_{L_2} \in \Theta(n)$, *for $z \in \{\mathrm{e}, \mathrm{p}\}$.*

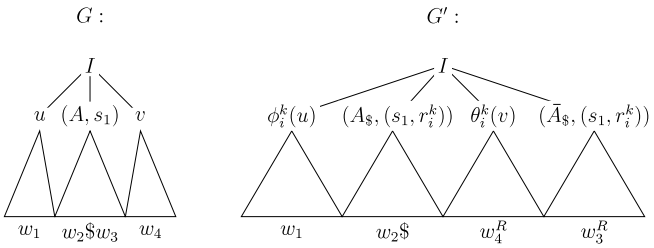


Fig. 2. A derivation tree of G is converted into a derivation tree of G'.

Proof. Let $L_2' = \{w\$w \mid w \in \{0,1\}^*\}$. We show that $(s, d)_z\text{-der}_{L_2} \notin \Omega(n)$ implies $(d, z)\text{-synch}_{L_2'} \notin \Omega(n)$, which contradicts $(d, z)\text{-synch}_{L_2'} \in \Omega(n)$, as shown in [9].[2]

Assume $G = (V, S, T, P, I)$ is an SCF grammar and $z \in \{p, e\}$ such that $L_z(G) = L_2$ and $(s, d)_z\text{-der}_{L_2} \notin \Omega(n)$. By [8] we can assume G to be λ-free. In the following we construct an SCF grammar $G' = (V', S', T, P', I)$ with $L_z(G') = L_2'$. As usual we denote the sets of all nonterminals of G and G' by $N = V \times (S \cup \{\lambda\})$ and $N' = V' \times (S' \cup \{\lambda\}$, respectively. The idea of the construction of G' is to reverse the second half of the words in L_2. To do that, the path leading to the \$ middle-marker has to be "split" into two paths, one in the middle of the word, which only generates the symbols to the left of the marker in the original grammar and eventually the marker itself and another path at the end of the word that generates the mirror images of the symbols to the right of the marker. These two paths have to be synchronized to ensure that they always generate symbols according to the same rule. To do that the situation symbols of the new grammar are pairs of the old situation symbols and production labels which are used only to synchronize the two parts of the \$-path. Furthermore we have to guess during the application of a production to the \$-path, which of the generated nonterminals will eventually derive the \$ marker. This has to be the same in both paths. Fig. 2 illustrates how a derivation tree of G is converted into a derivation tree of G'.

We assign unique production labels $R = \{r_1, r_2, \ldots, r_{|P|}\}$ to the productions in P and we define m to be the maximal length of the right-hand sides of productions in P.

We have $V' = \{A, \bar{A}, A_\$, \bar{A}_\$ \mid A \in V\}$ and $S' = (S \cup \{\lambda\}) \times \{r_i^k \mid 1 \le i \le |P|, 1 \le k \le m\}$. The second coordinate of the new situation symbols is used to synchronize the two halves of the path ending in the \$ sign in the original grammar. At each derivation step the two nonterminals representing the left and right half of the \$-path have to communicate which production they are using (indicated by the subscript i) and which nonterminal is guessed to be the one that will derive \$ (indicated by k).

The productions of G' are defined so that for nonterminals on the two \$-paths (where the first component of the nonterminal has subscript \$) a production in P' simulates the production of G that is determined by the second coordinate of the situation symbols. For nonterminals where the first component is not of the form $A_\$$ or $\bar{A}_\$$, the productions of P' are independent of the second components of the situation symbols

We define morphisms $\varphi_i^k : (N \cup T)^* \to (N' \cup T)^*$ and anti-morphisms $\theta_i^k : (N \cup T)^* \to (N' \cup T)^*$ for all i with $1 \le i \le |P|$, $1 \le k \le m$, where $\varphi_i^k(a) = \theta_i^k(a) = a$ for all $a \in T$, $\varphi_i^k((A, f)) = (A, (f, r_i^k))$ and $\theta_i^k((A, f)) = (\bar{A}, (f, r_i^k))$ for all $A \in V, f \in S \cup \{\lambda\}$. The morphism φ only adds the second coordinate of the situation symbol, while the anti-morphism θ also labels the nonterminals as being in the second half of the word (i.e. after the nonterminal that eventually derives the \$-sign).

At the beginning of a derivation we guess which nonterminal will eventually derive the \$-sign and split the derivation tree accordingly, as shown in Fig. 2. Thus, for all productions $I \to u(A, f)v \in P$ with $u, v \in (N \cup T)^*, A \in V, f \in S \cup \{\lambda\}$ and for all i and k with $1 \le i \le |P|$, $1 \le k \le m$ we have

$$I \to \varphi_i^k(u)(A_\$, (f, r_i^k))\theta_i^k(v)(\bar{A}_\$, (f, r_i^k)) \in P'.$$

For nonterminals that are not labelled by \$, we just copy the productions of G, change the situation symbols so that they are now pairs as discussed above, and if the nonterminal appears in the second half of the derivation tree (i.e. after the \$-sign we also reverse the right-hand side of the production. Thus, for all productions $(A, f) \to u \in P$ with $A \in V, f \in S \cup \{\lambda\}, u \in (N \cup T)^*, |u|_\$ = 0$ we have for all i, j, k, l with $1 \le i, j \le |P|$ and $1 \le k, l \le m$

$$(A, (f, r_i^k)) \to \varphi_j^l(u) \in P'; \qquad (\bar{A}, (f, r_i^k)) \to \theta_j^l(u) \in P'.$$

If a nonterminal is labelled by the \$-sign, then the second coordinate of the situation symbol tells us (1) which production we have to use next (through the label i) and (2) which nonterminal on the right-hand side will carry the \$-sign label at the next derivation layer (through the label k). This way we are making sure that the two paths labelled by the

[2] Strictly speaking, Lemma 23 of [9] establishes a linear lower bound for synchronization depth of the language $\{w\$w \mid w \in \{0, 1, \#\}^*\}$, that uses one more terminal symbol. Corollary 26 of [9] then explains how the result can be made to apply for languages over a binary alphabet, and in a similar way the lower bound can be translated also for our language L_2' here.

\$-sign are actually behaving like the two halves of the path leading to the \$-sign in the original derivation. For all i and k, $1 \leq i \leq |P|$, $1 \leq k \leq m$ such that the production labelled r_i is of the form $(A, f) \to u(B, f')v \in P$ for some $A \in V, f, f' \in S \cup \{\lambda\}$, $u, v \in (N \cup T)^*$, $|uv|_\$ = 0$, $|u| = k$ we have for all j, l with $1 \leq j \leq |P|$ and $1 \leq l \leq m$

$$(A_\$, (f, r_i^k)) \to \varphi_j^l(u)(B_\$, (f', r_j^l)) \in P';$$
$$(\bar{A}_\$, (f, r_i^k)) \to \theta_j^l(v)(\bar{B}_\$, (f', r_j^l)) \in P'.$$

The paths labelled by \$ can only terminate if the current production label (indicated by i) (1) belongs to a terminating production and (2) derives the dollar sign at the precise position indicated by k. Thus, for all i and k, $1 \leq i \leq |P|$, $1 \leq k \leq m$ such that the production labelled r_i is of the form $(A, f) \to u\$v \in P$ for some $A \in V, f \in S \cup \{\lambda\}$, $u, v \in (N \cup T)^*$, $|uv|_\$ = 0$, $|u| = k$ we have for all j, l with $1 \leq j \leq |P|$ and $1 \leq l \leq m$

$$(A_\$, (f, r_i^k)) \to \varphi_j^l(u)\$ \in P'; \qquad (\bar{A}_\$, (f, r_i^k)) \to \theta_j^l(v) \in P'.$$

It is easy to see that G' generates L_2' and that $(\mathrm{s}, \mathrm{d})_z\text{-der}_G \notin \Omega(n)$ implies $(\mathrm{d}, z)\text{-synch}_{G'} \notin \Omega(n)$, but this is a contradiction as it was shown in [9] that $(\mathrm{d}, z)\text{-synch}_{L_2'} \in \Omega(n)$. □

Note that both languages L_1 and L_2 are deterministic linear languages and that both of them can be generated with a context-free grammar that only uses one nonterminal. Thus, derivation complexity seems to be a substantially different measure of descriptional complexity, than the measures usually considered in the literature.

We can now also give an example of a non-context-free language for which the derivation depth is greater than the synchronized depth.

Example 24. Let $L = \{w\$w^R a^n b^n c^n \mid w \in \{0, 1\}^*, n \geq 1\}$. Then, for $z \in \{\mathrm{e}, \mathrm{p}\}$, $(\mathrm{d}, z)\text{-synch}_L \in \Theta(\log n)$, as L is the concatenation of a context-free language and a non-context-free language for which has logarithmic synchronized depth by Example 22. On the other hand, Lemma 23 implies that $(\mathrm{s}, \mathrm{d})_z\text{-der}_L \in \Theta(n)$.

4.4. Context-free languages with logarithmic depth

We now look at subfamilies of context-free languages, the synchronized depth of which is logarithmic, even though they require linear depth when being generated by a context-free grammar. We show that $L_Q = \{\varphi_1(w)\varphi_2(w^R) \mid w \in Q\}$ for some regular language Q and morphism φ_1, φ_2 requires only logarithmic depth whenever $\varphi_1(Q)$ and $\varphi_2(Q)$ are finite concatenations of unary regular languages. Note that these languages require linear depth when being generated by a context-free grammar, as shown in [14].

We first define what we mean by a *simple regular expression*.

Definition 25. A regular expression E over an alphabet Σ is called *simple* if and only if there exist natural numbers $n, k_1, \ldots, k_n$ such that

$$E = a_{1,0}a_{1,1}^*a_{1,2}^* \cdots a_{1,k_1}^* + \cdots + a_{n,0}a_{n,1}^*a_{n,2}^* \cdots a_{n,k_n}^*$$

for some $a_{i,0} \in \Sigma \cup \{\lambda\}$ and $a_{i,j} \in \Sigma$ for $1 \leq i \leq n$ and $1 \leq j \leq k_i$. A regular language L is called simple if there exists a simple regular expression E with $L = L(E)$.

We now show that every regular language the morphic image of which is unary can be converted into a simple regular language with the same morphic image.

Lemma 26. *Let Σ be an alphabet and let E be a regular expression over Σ and let φ_1, φ_2 be morphisms such that $\varphi_1(E)$ and $\varphi_2(E)$ are unary. Then there exist a simple regular expression E' and morphisms φ_1', φ_2', such that $L = L'$, where $L = \{\varphi_1(w)\$\varphi_2(w^R) \mid w \in E\}$ and $L' = \{\varphi_1'(w)\$\varphi_2'(w^R) \mid w \in E'\}$.*

Proof. We define a set of rewriting rules for regular expressions over Σ and then show that the regular expression obtained by iteratively applying the rewriting rules is simple and satisfies $L = L'$ as defined in the lemma statement.

Let c and d be the letters, such that $\varphi_1(E) \subseteq \{c\}^*$, $\varphi_2(E) \subseteq \{d\}^*$. We fix an order on the alphabet Σ, which we denote by $<$.

Let $\tilde{E}$ be the regular expression obtained from E by removing all $*$-operators. Then $L(\tilde{E})$ is obviously finite. Let $m = \max\{|w| \mid w \in L(\tilde{E})\}$.

We now define a new alphabet $\Sigma' = \Sigma \cup \{x_w \mid w \in \Sigma^{\leq m}\}$, where $\Sigma \cap \{x_w \mid w \in \Sigma^{\leq m}\} = \emptyset$. Also we define the morphism $\varphi_1' : \Sigma'^* \to \{c\}^*$ and $\varphi_2' : \Sigma^* \to \{d\}^*$ by $\varphi_i'(a) = \varphi_i(a)$ for all $a \in \Sigma$, $i \in \{1, 2\}$ and $\varphi_i'(x_{a_1 \cdots a_k}) = \varphi_i(a_1 \cdots a_k)$ for all $k \leq m$ and $a_j \in \Sigma$ for $1 \leq j \leq k$ and $i \in \{1, 2\}$.

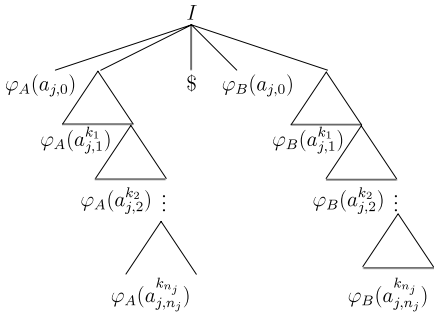


Fig. 3. Example derivation tree of G in the proof of Lemma 27.

Next we define a set of rewriting rules R that contains the following rules, where $a, b, b_1, \ldots, b_k \in \Sigma$, $u, v \in \Sigma^*$, $0 \leq k \leq |\Sigma|$, $n \in \mathbb{N}$ and $e_1, \ldots, e_n$ are regular expressions.

$$ab \to x_{ab}; x_u a \to x_{ua}; a x_u \to x_{au}; \text{ for all } a, b \in \Sigma, u, v, \in \Sigma^{\leq m} \tag{1}$$

$$x_{u'} x_{v'} \to x_{u'v'} \text{ for all } u'v' \in \Sigma^{\leq m} \tag{2}$$

$$(ab_1^* \cdots b_k^*)^* \to a^* b_1^* \cdots b_k^* \tag{3}$$

$$(e_1 + e_2 + \cdots + e_n)^* \to \sum_{l \leq n, i_j < i_{j+1}, 1 \leq j < l} e_{i_1}^* \cdots e_{i_l}^* \tag{4}$$

$$e_1 \cdot (e_2 + e_3 + \cdots + e_n) \to e_1 e_2 + e_1 e_3 + \cdots e_1 e_n \tag{5}$$

$$(e_2 + e_3 + \cdots + e_n) \cdot e_1 \to e_2 e_1 + e_3 e_1 + \cdots + e_n e_1 \tag{6}$$

$$a^* b \to b a^*; a^* a^* \to a^* \tag{7}$$

$$b^* a^* \to a^* b^* \text{ whenever } a < b. \tag{8}$$

First we prove that the rewriting rules defined above are well-founded, i.e. for each regular expression they will terminate after a finite number of rewriting steps.

First observe that there are at most $m \cdot |L(E')|$ applications of rules (1) and (2). Also, whenever e_1 and e_2 are simple, then $e_1 + e_2$ is simple as well. Rules (3) and (4) shorten the lengths of subexpressions surrounded by a Kleene-star and as there are no other rules that introduce expressions surrounded by a Kleene-star, the number of applications of these rules is also bounded. Rules (5) and (6) reduce the maximal length of non-simple subexpressions. The only other rules that can introduce non-simple subexpressions are rules (3) and (4), but as seen above they can only introduce a bounded number of such subexpressions. Once none of the other productions can be applied anymore, then rules (7) and (8) only order the symbols, the number of applications of these rules is obviously bounded.

It is easy to see that all the rules preserve the Parikh sets of the expressions, with the Parikh vectors of a symbol x_w begin defined as the Parikh vector of w.

Now all that we are left to show is that, once no more rule can be applied, we have a simple regular expression. There are no more Kleene-stars surrounding anything other than single symbols because of rules (3) and (4). There are also no more concatenations of anything other than single symbols because of rules (5) and (6). Furthermore rules (1), (2), (7) and (8) guarantee that there is at most one symbol not having a Kleene-star in each of the subexpressions connected by $+$. The above argument shows that E' is in fact simple and as it is easy to see that all the rules preserve the Parikh sets of the expressions (again with the Parikh vector of x_w being defined as the Parikh vector of w), $L = L'$ follows. □

We can now use the previous result to show that certain linear languages require at most logarithmic depth when they are generated by an SCF grammar.

Lemma 27. *Let Σ and Δ be finite alphabets, let L be a regular language over Σ and let $\varphi_A, \varphi_B : \Sigma^* \to \Delta^*$ be morphism, such that $\varphi_A(L) \subseteq \{a\}^*$ and $\varphi_B(L) \subseteq \{b\}^*$ for some $a, b \in \Delta$. Let $L' = \{\varphi_A(w)\varphi_B(w^R) \mid w \in L\}$ and $L'_\$ = \{\varphi_A(w)\$\varphi_B(w^R) \mid w \in L\}$. Then, for $z \in \{\text{e}, \text{p}\}$, $(\text{s}, \text{d})_z\text{-der}_{L'} \in \mathcal{O}(\log n)$ and $(\text{s}, \text{d})_z\text{-der}_{L'_\$} \in \mathcal{O}(\log n)$.*

Proof. Without loss of generality we can assume L to be simple by Lemma 26. Thus there exist $n, k_1, \ldots, k_n \in \mathbb{N}$ such that $L = a_{1,0} a_{1,1}^* a_{1,2}^* \cdots a_{1,k_1}^* + \cdots + a_{n,0} a_{n,1}^* a_{n,2}^* \cdots a_{n,k_n}^*$ for some $a_{i,j} \in \Sigma$ for $1 \leq i \leq n$ and $0 \leq j \leq k_i$.

To generate L' and $L'_\$$ by an SCF grammar with logarithmic synchronized derivation depth we use the binary encoding used in the proof of Theorem 30, which can be extended to accommodate finite unions of languages of the form $\{a_1^{n_1} a_2^{n_2} \cdots a_m^{n_m} \$ b_m^{n_m} \cdots b_2^{n_2} b_1^{n_1}\}$.

The derivation trees look like the tree in Fig. 3.

It is immediate that grammars constructed in this way have logarithmic depth in both modes and with or without the middle-marker. □

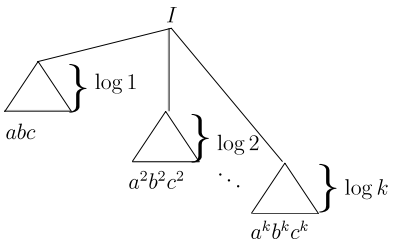


Fig. 4. Example derivation tree of G_p in the proof of Theorem 30.

In the previous lemma, we can replace $\mathcal{O}(\log n)$ by $\Theta(\log n)$ whenever L' or $L'_\$$ is infinite by Lemma 14.

We now extend Lemma 27 to more general morphic images.

Lemma 28. *Let L be a regular language over some alphabet Σ. Then the following two conditions are equivalent.*

(1) $L \subseteq a_1^* a_2^* \cdots a_k^*$ *for some* $a_1, a_2, \ldots a_k \in \Sigma$.
(2) *There exist natural numbers* $n, k_1, \ldots, k_n$ *such that there exist* unary *regular languages* L_{ij} *for all* $1 \le i \le n$ *and* $1 \le j \le k_i$, *such that*

$$L = \bigcup_{1 \le i \le n} L_{i_1} \cdot L_{i_2} \cdots L_{i_{k_i}}.$$

Proof. **(1) ⇒ (2):** Let $A = (Q, \Sigma, \delta, q_0, F)$ be a deterministic accessible and co-accessible finite automaton for L. Then the number of factors of the form ab, $a \neq b$ is bounded by k for each word in L. For any language $L \subseteq a_1^* a_2^* \cdots a_k^*$ we can write

$$L = \bigcup_{q \in Q} (R_q \cap a_1^*)(S_q \cap a_2^* \cdots a_k^*), \tag{9}$$

where R_q is the set of words that takes the initial state of A to q, and S_q is the set of words that takes q to a final state of A. Then the factorization (9) gives us the claim by induction (by using the distributivity of union and catenation).
(2) ⇒ (1): Let $L_i = L_{i_1} L_{i_2} \cdots L_{i_{k_i}}$ for all i, $1 \le i \le n$. Then the implication is obvious as $L \subseteq L_1 L_2 \cdots L_n$. □

The following theorem follows by Lemmas 27 and 28. It extends Lemma 27 to morphic images that are a subset of expressions of the form $a_1^* a_2^* \cdots a_n^*$.

Theorem 29. *Let Σ, Δ be finite alphabets, let L be a regular language and let $\varphi_1, \varphi_2 : \Sigma^* \to \Delta^*$ be morphism, such that $\varphi_i(L) \subseteq a_{i,1}^* a_{i,2}^* \cdots a_{i,k_i}^*$ for some $k \in \mathbb{N}$, $a_{i,1}, a_{i,2}, \ldots, a_{i,k_i} \in \Sigma$, $i \in \{1, 2\}$. Let $L' = \{\varphi_1(w)\varphi_2(w) \mid w \in L\}$ and $L'' = \{\varphi_1(w)\varphi_2(w^R) \mid w \in L\}$. Then, for $z \in \{\mathrm{e}, \mathrm{p}\}$, $(\mathrm{s}, \mathrm{d})_z$-$\mathrm{der}_{L''} \in \mathcal{O}(\log n)$ and (d, z)-$\mathrm{synch}_{L'} \in \mathcal{O}(\log n)$.*

There are many more context-free languages the depth of which is logarithmic when they are generated by an SCF grammar. For example, all finite concatenations (finite unions) of languages with logarithmic depth have logarithmic depth.

5. An extension of the depth synchronization hierarchy

Extending the idea of Example 22 we construct a grammar that has a synchronization function not belonging to any level of the hierarchy of Theorem 13.

Theorem 30. *For $z \in \{\mathrm{e}, \mathrm{p}\}$, there exists an* SCF *grammar G_z such that* (d, z)-$\mathrm{synch} \in \Theta(\sqrt{n} \log n)$.

Proof. We give the grammar for p-mode, the construction for e-mode only adds chain productions to extend the paths of the derivation trees. Let $G_p = (V, S, T, P, I)$ with $V = \{I, A, B, C, A', B', C', D_\lambda, D_A, D_B, D_C\}$, $S = \{q, e, s_0, s_1\}$, $T = \{a, b, c\}$ and the following productions are, for $i, j \in \{0, 1\}$, $f, f' \in S$, and $X \in \{A, B, C\}$, in P.

$$\begin{aligned}
I &\to I(J, f) \mid (J, f) & (J, f) &\to (J, f') \\
(J, q) &\to (A, s_i)(B, s_i)(C, s_i) & (X, s_0) &\to (X, s_i) \mid (D_\lambda, e) \\
(X, s_1) &\to (X', s_i)(X, s_i) \mid (D_x, e) & (X', s_i) &\to (X', s_j)(X', s_j) \mid (D_X, e)(D_X, e) \\
(D_\lambda, e) &\to \lambda & (D_X, e) &\to x.
\end{aligned}$$

Then G_p generates $L = \{a^n b^n c^n \mid n \ge 0\}^*$ in prefix mode. In the derivations, first an arbitrary number of nonterminals (J, f) for some $f \in S$ is generated. These can then generate an arbitrary sequence of situation symbols until they eventually generate a subtree that yields $a^l b^l c^l$ for some $l \in \mathbb{N}$. Note that in such a subtree the situation sequence always begins with a q begin-marker and ends with an e end-marker. We now look at the derivation trees of the words $w_k = abca^2b^2c^2 \cdots a^k b^k c^k$. Fig. 4 gives a derivation tree of such a word.

The words w_k are the worst possible words for the grammar with respect to the synchronization depth as no two instances of $a^{k_1} b^{k_1} c^{k_1}$ and $a^{k_2} b^{k_2} c^{k_2}$, $k_1 \neq k_2$ can be generated in parallel. We see that for any given k, the length of $w_k \in \Theta(k^2)$, while the depth of the (unique due to the additional situation symbols b and e) derivation tree of w_k is $\log k! \in \Theta(k \log k)$. This implies that (d, p)-$\mathrm{synch}_{G_p} \in \Theta(\sqrt{n} \log n)$. □

6. Open problems

More research is needed to properly understand for which subfamilies of context-free languages the introduction of a synchronization mechanism yields significant savings in derivation complexity.

Also it is an open problem to find more SCF grammars with synchronization functions outside of the hierarchy of Theorem 13, and to determine whether there exists an SCF language with a function outside of the hierarchy. We conjecture that this is the case for the language used in Theorem 30.

References

[1] J. Dassow, G. Păun, Regulated Rewriting in Formal Language Theory, in: EATCS Monographs in Theoretical Computer Science, vol. 18, Springer-Verlag, Berlin, 1989.
[2] E. Csuhaj-Varjú, J. Dassow, J. Kelemen, G. Păun, Grammar Systems: A Grammatical Approach to Distribution and Cooperation, in: Topics in Computer Mathematics, vol. 5, Gordon and Breach Science Publishers, Yverdon, 1994.
[3] A. Aho, Indexed grammars — an extension of context-free grammars, Journal of the ACM 15 (1968) 647–671.
[4] H. Jürgensen, K. Salomaa, Block-synchronization context-free grammars, in: Z. Du, I. Ko (Eds.), Advances in Algorithms, Languages, and Complexity, Kluwer Academic Publishers, The Netherlands, 1997, pp. 111–137.
[5] H. Bordihn, M. Holzer, On the computational complexity of synchronized context-free languages, Journal of Universal Computer Science 8 (2) (2002) 119–140.
[6] I. McQuillan, The generative capacity of block-synchronized context-free grammars, Theoretical Computer Science 337 (2005) 119–133.
[7] I. McQuillan, Descriptional complexity of block-synchronized context-free grammars, Journal of Automata, Languages and Combinatorics 9 (2004) 317–332.
[8] F. Biegler, Synchronization functions of synchronized context-free grammars and languages, Journal of Automata, Languages, and Combinatorics 12 (2007) 7–24.
[9] F. Biegler, I. McQuillan, K. Salomaa, An infinite hierarchy induced by depth synchronization, Theoretical Computer Science 387 (2007) 113–124.
[10] J. Hromkovič, J. Karhumäki, B. Rovan, A. Slobodová, On the power of synchronization in parallel computations, Discrete Applied Mathematics 32 (1991) 155–182.
[11] R. Book, Time-bounded grammars and their languages, Journal of Computer and System Sciences 5 (1971) 297–429.
[12] K. Culik II, H. Maurer, On the height of derivation trees, Tech. rep., Institut für Informationsverarbeitung Graz, 1978.
[13] F.-J. Brandenburg, On the height of syntactical graphs, in: P. Deussen (Ed.), Theoretical Computer Science, in: LNCS, vol. 104, Springer, Berlin Heidelberg, 1981, pp. 13–21.
[14] K. Reinhardt, A tree-height hierarchy of context-free languages, International Journal of Foundations of Computer Science 18 (2007) 1383–1394.
[15] J. Hopcroft, J. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, Reading, MA, 1979.
[16] A. Salomaa, Formal Languages, Academic Press, New York, 1973.
[17] S. Yu, Regular languages, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, vol. 1, Springer, Berlin Heidelberg, 1997, pp. 41–110.
[18] G. Rozenberg, A. Salomaa, The Mathematical Theory of L Systems, Academic Press, Inc., New York, 1980.
[19] L. Kari, G. Rozenberg, A. Salomaa, L Systems, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, vol. 1, Springer, Berlin Heidelberg, 1997, pp. 253–328.
[20] T. Cormen, C. Leiserson, R. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA, 1989.
[21] M. Madhu, K. Krithivasan, Length synchronization context-free grammars, Journal of Automata Languages and Combinatorics 9 (2004) 457–464.