Generalized Derivations with Synchronized Context-Free Grammars* **

Markus Holzer[1], Sebastian Jakobi[1], and Ian McQuillan[2]

[1] Institut für Informatik, Universität Giessen,
Arndtstr. 2, 35392 Giessen, Germany
email: {holzer,jakobi}@informatik.uni-giessen.de
[2] Department of Computer Science, University of Saskatchewan
Saskatoon, SK S7N 5A9, Canada
email: mcquillan@cs.usask.ca

Abstract. Synchronized context-free grammars are special context-free grammars together with a relation which must be satisfied between every pair of paths from root to leaf in a derivation tree, in order to contribute towards the generated language. In the past, only the equality relation and the prefix relation have been studied, with both methods generating exactly the ET0L languages. In this paper, we study arbitrary relations, and in particular, those defined by a transducer. We show that if we use arbitrary a-transducers, we can generate all recursively enumerable languages, and moreover, there exists a single fixed transducer, even over a two letter alphabet, which allows to generate all recursively enumerable languages. We also study the problem over unary transducers. Although it is left open whether or not we can generate all recursively enumerable languages with unary transducers, we are able to demonstrate that we can generate all ET0L languages as well as a language that is not an indexed language. Only by varying the transducer used to define the relation, this generalization is natural, and can give each of the following language families: context-free languages, a family between the E0L and ET0L languages, ET0L languages, and recursively enumerable languages.

1 Introduction

The study of *synchronization* in formal language theory was originally initiated by Hromkovič in order to study communication between parallel computations of Turing machines and alternating Turing machines [5,6]. Then, in [12], Salomaa introduced synchronized tree automata, which allowed for limited communication between different paths of the trees. Synchronized context-free (SCF) grammars were created [7] to study the yields of synchronized tree automata, as string languages. This model consists of context-free grammars, where the

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

** Published in Proceedings of DLT 2012. The final authenticated version is available online at https://doi.org/10.1007/978-3-642-31653-1_11.

nonterminals are ordered pairs, and the second component is an optional synchronization symbol used to communicate with other branches of the tree. For every pair of paths between root and leaf, the sequences of synchronization symbols from the top towards the bottom of the tree must be related to each other, according to some given relation, in order to contribute towards the generated language. The language generated by an SCF grammar depends on the relation used. In the past, only two relations have been studied. The first is the equality relation, where the sequence of synchronization symbols must be equal for every pair of paths from root to leaves. The second is the prefix relation, where between every two paths, one must be a prefix of the other. It has been shown that both language families are identical [7], and equal to the family of ET0L languages [9]. Moreover, if only one synchronization symbol is allowed, in a model called *counter synchronized context-free grammars*, a language family strictly between E0L and ET0L is obtained [2]. No other relations have been studied, although the definition of SCF grammars is general enough that any binary relation on words can be used.

In this paper, we use a common computational model, a transducer, to vary the relation. We can obtain standard equality and prefix synchronization with specific transducers, as well as the counter SCF languages. We show that one can in fact generate all recursively enumerable languages by varying the transducer. Moreover, there is a single fixed transducer, over a two letter alphabet, which gives all recursively enumerable languages. We also examine transducers over a one letter alphabet. In contrast to normal SCF languages with equality synchronization over a one letter alphabet, we are able to generate all ET0L languages. Therefore, we can simulate equality synchronization over any alphabet with a transducer over a unary alphabet. Further, we are able to generate a non-ET0L language, and indeed, non-indexed language, with a one letter transducer. However, the exact capacity with unary transducers is left open.

The paper is organized as follows. In the next section we introduce the necessary notations on a-transducers and SCF grammars, leading to the notion of M-synchronized context-free grammars and languages. Then in Section 3 we first give some examples on M-SCF grammars to become familiar with this new concept and its basic mechanisms. The main result of this section is the characterization of the family of recursively enumerable languages by M-SCF grammars, even by a single fixed transducer. Moreover, this single fixed transducer can be chosen to be over a two letter alphabet. Finally, in Section 4 we study the generative capacity of M-SCF for unary a-transducers. Finally, we state some open problems related to SCF and M-SCF grammars and languages.

2 Preliminaries

In this section, we will define the necessary preliminaries, as well as define synchronized context-free grammars and languages as they have been defined previously in the literature.

Let $\mathbb{N}$ and $\mathbb{N}^+$ be the set of non-negative and positive integers, respectively. An alphabet A is a finite, non-empty set of symbols. The set of all words over A is denoted by A^*, which contains the empty word λ. 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 $x \leq_{\mathrm{p}} y$, if $y = xu$ for some word $u \in A^*$. Also, $w_1 \simeq_{\mathrm{p}} w_2$ if and only if either $w_1 \leq_{\mathrm{p}} w_2$ or $w_2 \leq_{\mathrm{p}} w_1$. We also say $w_1 \simeq_{\mathrm{e}} w_2$ if and only if $w_1 = w_2$.

We will next define an a-transducer. Intuitively, it is a nondeterministic gsm that allows output on a λ input. They are also referred to as rational transducers. An *a-transducer* is a 6-tuple $M = (Q, A_1, A_2, \delta, q_0, F)$, where Q is the finite state set, A_1 is the input alphabet, A_2 is the output alphabet, δ is a finite subset of $Q \times A_1^* \times A_2^* \times Q$, $q_0 \in Q$ is the initial state and $F \subseteq Q$ is the set of final states. Let $\vdash$ be the relation on $Q \times A_1^* \times A_2^*$ defined by letting $(q, xw, z_1) \vdash (p, w, z_2)$ for each $w \in A_1^*$ if $(q, x, y, p) \in \delta$ and $z_2 = z_1 y$. An intermediate stage of a computation of M, or a configuration of M, is represented by a triple (q, w, z) where M is in state q, with w the input still to be read, and z the accumulated output. Let $\vdash^*$ be the reflexive, transitive closure of $\vdash$. For such an a-transducer, and for each word $w \in A_1^*$, let

$$M(w) = \{\, z \mid (q_0, w, \lambda) \vdash^* (q, \lambda, z) \text{ for some } q \in F \,\}.$$

For every set $L \subseteq \Sigma_1^*$, let $M(L) = \bigcup_{w \in L} M(w)$. The mapping M is called an a-transducer mapping or a-transduction.

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 starting 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. A *tree domain* D is a nonempty finite subset of $\mathbb{N}_+^*$ such that

1. if $\mu \in D$, then every prefix of μ belongs to D and
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, $\mathrm{dom}(t)$. A node $\mu \in \mathrm{dom}(t)$ is labelled by $t(\mu)$. A node $\lambda \in \mathrm{dom}(t)$, denoted by $\mathrm{root}(t)$, is called the root of t. The set of leaves of t is denoted $\mathrm{leaf}(t)$. The subtree of t at node μ is t/μ. 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. For element $\mu \in \mathrm{dom}(t)$, let $\mathrm{path}_t(\mu)$ be the sequence of symbols of A occurring on the path from the root of t to the node μ.

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 $\mu 1, \mu 2, \ldots, \mu k$, for $k \geq 1$. Then $t(\mu) \to t(\mu 1)t(\mu 1)\cdots t(\mu k) \in P$.

The set of derivation trees of G is denoted $T(G)$. The *yield* of a derivation 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 $\mathbb{N}_+^*$. The derivation trees of G are in one-to-one correspondence with the equivalence classes of derivations of G producing terminal words, and thus

$$L(G) = \{\, \mathrm{yd}(t) \mid t \in T(G) \,\}.$$

In the following, for simplicity we also write $\mathrm{yd}(T(G))$ instead.

The family of context-free languages [4,11] is denoted as usual by $\mathcal{L}(\mathrm{CF})$, and with $\mathcal{L}(\mathrm{E0L})$ and $\mathcal{L}(\mathrm{ET0L})$ we refer to the family of E0L (extended Lindenmayer systems without interaction) and ET0L (extended tabled Lindenmayer systems without interaction) languages, respectively—see [10].

Definition 1. *A* synchronized context-free grammar (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 the condition* $h_G((v, x)) = x$ *for all* $v \in V$ *and* $x \in S \cup \{\lambda\}$.

This morphism follows the definition of an important concept, namely the synchronizing sequence of a path on the derivation tree.

Definition 2. *Let* G *be a* SCF *grammar. For a derivation tree* t *of* G, $t_1 = \mathrm{inner}(t)$ *and a node* $\mu \in \mathrm{leaf}(t_1)$, *the* synchronizing sequence *(sync-sequence) corresponding to* μ *is* $\mathrm{seq}_{t_1}(\mu) = h_G(\mathrm{path}_{t_1}(\mu))$.

Next, we will restrict the trees that will be used to generate SCF languages.

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

The z-acceptable SCF language families are defined as follows:

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

It was proven in [7] that p- and e-synchronization generate the same family of languages, i.e., $\mathcal{L}_{\mathrm{e}}(\mathrm{SCF}) = \mathcal{L}_{\mathrm{p}}(\mathrm{SCF}) = \mathcal{L}(\mathrm{SCF})$. In [9] it was shown that SCF grammars generate the family of ET0L languages, i.e., $\mathcal{L}(\mathrm{SCF}) = \mathcal{L}(\mathrm{ET0L})$, and given an SCF grammar and a derivation mode one can effectively construct an equivalent ET0L system and *vice versa*. Furthermore, equality synchronization with only a single situation symbol or prefix synchronization with a single situation symbol plus an endmarker generates the counter synchronized-context free languages, a language family strictly between the $\mathcal{L}(\mathrm{E0L})$ and $\mathcal{L}(\mathrm{ET0L})$ language families [2,8].

3 Transducer synchronization

Next, we generalize the equality and prefix relation to arbitrary relations defined by a transducer.

Definition 5. *Let $G = (V, S, T, P, I)$ be an SCF grammar together with an a-transducer $M = (Q, S, S, \delta, q_0, F)$. Then, $w_1 \simeq_M w_2$ if and only if at least one of $w_1 \in M(w_2)$ or $w_2 \in M(w_1)$ is true.*

Now we can define the derivation tree we are interested in, as follows:

Definition 6. *A derivation tree t of G is said to be M-acceptable if*

$$\mathrm{seq}_{\mathrm{inner}(t)}(\mu) \simeq_M \mathrm{seq}_{\mathrm{inner}(t)}(\nu),$$

for each $\mu, \nu \in \mathrm{leaf}(\mathrm{inner}(t))$. The set of M-acceptable derivation trees of G is denoted by $T_M(G)$.

Finally, we define the accepted language and the notation for the language family in question.

Definition 7. *The* M-synchronized language *of G is $L_M(G) = \mathrm{yd}(T_M(G))$. The families of* M-SCF *languages are denoted $\mathcal{L}_M(\mathrm{SCF})$. Moreover, the set of languages generated by all such transducers is denoted by $\mathcal{L}_*(\mathrm{SCF})$.*

As an example, consider the fixed transducer M_{e} on a two letter alphabet S that reads a word w and outputs w. This produces the same relation as the equality relation on two letters. And because it is known [8] that two situation symbols are sufficient to generate all languages in $\mathcal{L}_{\mathrm{e}}(\mathrm{SCF})$, we can conclude that $\mathcal{L}_{M_{\mathrm{e}}}(\mathrm{SCF}) = \mathcal{L}_{\mathrm{e}}(\mathrm{SCF}) = \mathcal{L}(\mathrm{ET0L})$. We can also build a transducer which gives the prefix relation over a binary alphabet. If we consider M_{c} that is the same as M_{e} except over a single letter alphabet, then we get the counter synchronized context-free languages [2], a family of languages strictly between the $\mathcal{L}(\mathrm{E0L})$ and $\mathcal{L}(\mathrm{ET0L})$ languages. And, the transducer that outputs λ for all inputs generates the context-free languages. However, next we give a fixed transducer whereby we can generate a non-ET0L language.

Example 8. Consider the language

$$L = \{\, x\#\phi(x) \mid x \in \{0,1\}^* \text{ with } |x| = 2^n, \text{ for } n \geq 0 \,\},$$

where ϕ is a homomorphism defined by mapping 0 to a and 1 to b. In order to show that L is a non-ET0L language we argue as follows: the family of ET0L languages is a full-AFL and therefore closed under arbitrary homomorphisms, and the language $h(L)$, where $h : \{a, b, 0, 1, \#\}^* \to \{a, b, 0, 1\}^*$ is the erasing homomorphism defined by $h(x) = x$, for $x \in \{a, b, 0, 1\}$, and $h(\#) = \lambda$, is not an ET0L language [10, page 252, Corollary 2.11]. Hence L is not an ET0L language either.

The grammar is defined as $G = (V, S, T, P, I)$, where the productions are as follows:

$I \to (X, l)\#(Y, l')$

$(X, l) \to (X, l)(X, r) \mid (X, 0) \mid (X, 1)$	$(Y, l') \to (Y, l')(Y, r') \mid (Y, a) \mid (Y, b)$
$(X, r) \to (X, l)(X, r) \mid (X, 0) \mid (X, 1)$	$(Y, r') \to (Y, l')(Y, r') \mid (Y, a) \mid (Y, b)$
$(X, 0) \to 0$	$(Y, a) \to a$
$(X, 1) \to 1$	$(Y, b) \to b$

Let g be a homomorphism which maps 0 to a, 1 to b, l to l' and r to r'. The transducer M is defined, by mapping strings in S^* as follows:

1) maps strings of the form $x\alpha$ to $y\beta$, where $x, y \in \{l, r\}^+, |x| = |y|$, and $\alpha, \beta \in \{0, 1\}$,
2) maps strings of the form $x\alpha$ to $y\beta$, where $x, y \in \{l', r'\}^+, |x| = |y|$, and $\alpha, \beta \in \{a, b\}$,
3) maps strings of the form $x\alpha$ to $g(x)g(\alpha)$, where $x \in \{l, r\}^+$, and $\alpha \in \{0, 1\}$,
4) maps strings of the form $x\alpha$ to $y\beta$, where $x \in \{l, r\}^+, y \in \{l', r'\}^+, \alpha, \beta \in \{0, 1\}, y \neq g(x)$, and $|x| = |y|$, and
5) maps strings of the form $x\alpha$ and $y\beta$, where $x \in \{l, r\}^+, y \in \{l', r'\}^+, \alpha \in \{0, 1\}$, and $\beta \in \{a, b\}$ to the empty word.

Every tree produced that is M-acceptable consists of two subtrees as children of I, rooted at (X, l), and (Y, l') as well as a third branch producing only the marker #, as seen in Figure 1. Derivations of M of the form in 1), are used to synchronize between every two paths of the left subtree. Indeed, every such path must be of the same length, and the number of nonterminals in the left subtree doubles at each height, and each path can generate either 0 or 1. Therefore, the yield of every such left subtree is of the form $x \in \{0, 1\}^*$ with $|x| = 2^n$, for $n \geq 0$. Similarly, rules of type 2) are used to synchronize the right subtree, and produce words of the form $y \in \{a, b\}^*$ with $|y| = 2^n$, for $n \geq 0$. Moreover, each path from root to leaf of the left subtree has some unique situation sequence $x\alpha$, where $x \in \{l, r\}^+$ and $\alpha \in \{0, 1\}$. For the one unique path in the right subtree with the situation sequence of $g(x)g(\alpha)$, the leaf of the first subtree must be α, while it must be $g(\alpha)$ in the second subtree, using rules of type 3). However, for every other path from root to leaf whereby $x\alpha$ is the situation sequence of the first tree, and $y\beta$ is the situation sequence of the second tree, then $g(x) \neq y$, and no

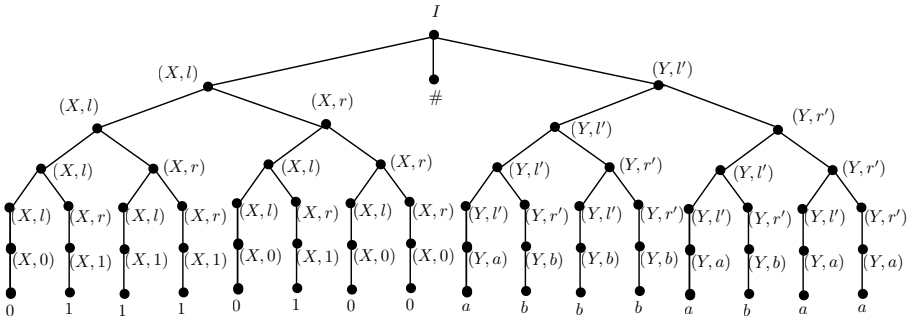

Fig. 1. An acceptable tree generating the word 01110100#*abbbabaa*.

conditions are placed on α and β allowing non-"matching" paths to synchronize arbitrarily. Rules of type 5) allow to synchronize with the marker #, which has an empty synchronization sequence. □

Example 9. We will give another example of a non-standard simulation of the linear languages. Linear languages are context-free languages, where there is at most one nonterminal on the right hand side of every production (as seen on the left diagram of Figure 2). It is known that all linear languages can be accepted by linear grammars where each production is of the form $A \to bB$, $A \to Bb$, or $A \to b$, where A, B are nonterminals and b is a terminal; instead of having rules of the form $A \to b$ in the grammar, one can require to have rules $A \to \lambda$ instead. The derivation trees for all such linear grammars consist of a single "path of nonterminals" with terminals to the right or left at each height. It is obvious that all such grammars can be generated without any synchronization at all. However, in the simulation presented here, the terminals generated to the left of the main "path of nonterminals" in the linear grammar are now generated on a completely separate branch of the tree, and the synchronization communicates the information about their proper placement. The intuition behind the simulation appears in Figure 2. We create labels in bijective correspondence with the productions of the linear grammar. Then we create three branches. The first nondeterministically generates a sequence of production symbols as situation sequence. The second generates all terminals to the left of the main branch of the linear grammar. The third generates all terminals to the right of the main branch of the linear grammar. The situation symbols, and the transducer, communicate the production symbols between branches. This example is important in understanding the simulation of arbitrary recursively enumerable languages by synchronized context-free grammars by varying transducers. □

Next we study some basic closure properties of M-SCF languages, for arbitrary, but fixed, a-transducers M.

Proposition 10. *For every a-transducer M, $\mathcal{L}_M(\mathrm{SCF})$ is a full semi-AFL.*

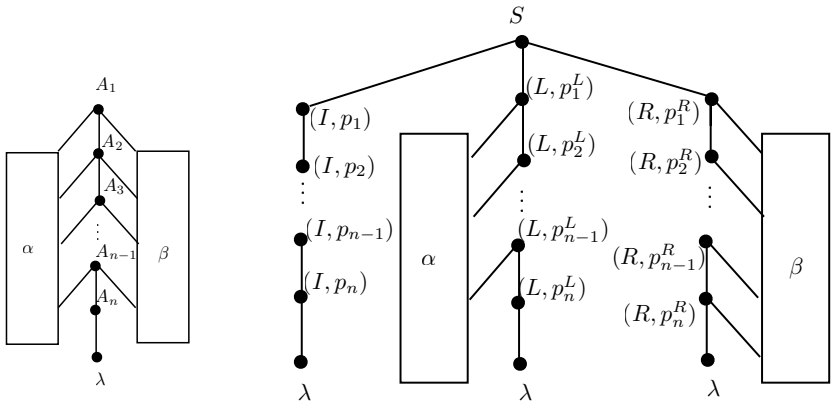

Fig. 2. The tree on the left is a derivation tree of a linear grammar, using a sequence of productions $p_1, p_2, \ldots, p_n$. The tree on the right is a simulation with a synchronized context-free grammar, where the terminals derived to the left of the main branch are now on a completely separate branch from those derived on the right.

Proof. Languages families that are full semi-AFLs are closed under homomorphism, inverse homomorphism, intersection with regular languages, and union. The results for homomorphism, inverse homomorphism and intersection with regular languages follows using exactly the same proofs as those for ET0L [10]. The proof for union can be seen by also using the standard proof for context-free languages, where we create a new grammar with a new start symbol, that goes to either of the two original start symbols (using non-synchronizing nonterminals). □

It is an open question whether or not there exists a transducer which generates a language family not closed under concatenation, Kleene star, or both. It is clear however, that there are some transducers that give language families that are also closed under concatenation and Kleene star, as the ET0L and counter synchronized-context-free languages are closed under these as well.

This immediately implies that $\mathcal{L}_*(\text{SCF})$ is also a full semi-AFL. In particular, closure under homomorphism is important, as we prove next that this family can generate all recursively enumerable languages, by using the well-known characterization that every recursively enumerable language is equal to $h(L_1 \cap L_2)$, for a homomorphism h, and two linear context-free languages. Therefore, we will now show that we can accept the intersection of two linear context-free languages. The proof is omitted for reasons of space.

Proposition 11. *Let $G_1 = (N_1, T, P, I)$ and $G_2 = (N_2, T, Q, J)$ be two linear context-free grammars. Then $L(G_1) \cap L(G_2)$ is an M-*SCF *language, for some a-transducer M (that depends on both G_1 and G_2).* □

The simulation is similar in nature to the simulation of linear grammars in Example 9. In that example, we created three branches to simulate the single branch of the linear grammar. The first branch generated the sequence of productions used in the linear grammar. The second branch generated all terminals to the left and the third generated all to the right. Here, we adopt a similar technique for both linear grammars to be simulated. In this case however, we require four branches for each grammar, giving a total of eight branches.

The first four generate a word from the first linear grammar, and the second four simulate the second linear grammar, however the second set of four branches will only verify the same word is generated as the first grammar before outputting the empty word across the entire second set of four branches.

Both grammars G_1 and G_2 can generate the same word, but the first grammar for example could generate the word with more letters to the right of the main branch than the second grammar (and therefore the second would generate more to the left than the first grammar). Therefore, it becomes non-trivial to test equality as we cannot simply test for equality using synchronization passed from the top towards the bottom in the tree. Then, we use the first and fifth branches to use sequences of production labels in correspondence with the two grammars. We use the second and sixth branches to generate those terminals that occur to the left of *both* linear grammar trees. We use the fourth and eighth branches to generate those terminals that occur to the right of *both* linear grammar trees. And lastly, we use the third and seventh branch to generate those remaining terminals which occur to the left of one grammar, and to the right of the other. This final synchronization, between the third and seventh branch, is more complicated, as they are generated in the opposite order (the terminals for one will be generated from the top to the bottom in the tree, while the other will be generated bottom-up). Then, we use a synchronization argument that uses a length argument on the remaining sequence of one grammar in comparison to what has already been generated in the other grammar. This combined with the fact that $\mathcal{L}_*(\mathrm{SCF})$ is closed under homomorphism, and also with the fact that every such language can be accepted by a Turing machine gives us:

Theorem 12. *$\mathcal{L}_*(\mathrm{SCF})$ is equal to the family of recursively enumerable languages.* □

Next, we show that there exists a fixed transducer generating all recursively enumerable languages.

Proposition 13. *There exists a fixed transducer M such that $\mathcal{L}_M(\mathrm{SCF})$ is equal to the recursively enumerable languages.*

Proof. Because every recursively enumerable language can be generated by a synchronized context-free grammar with some transducer, we can start with some universal Turing machine, which accepts the language

$$L = \{ \langle A, w \rangle \mid w \in L(A) \},$$

where A is a Turing machine encoded over some alphabet disjoint from $w \in \Sigma^*$, where $\Sigma = \{0, 1\}$. This language can be accepted with some synchronized context-free grammar G using some transducer M for synchronization, by Theorem 12. But then, given any Turing machine A over Σ, we can construct $L(A) = h(L \cap \langle A, \Sigma^* \rangle)$, where h is a homomorphism which erases everything not in Σ, but maps every letter of Σ to itself. Moreover, by Proposition 10, every fixed transducer gives a full semi-AFL and is therefore closed under intersection with regular languages and homomorphism. Therefore, this language is in $\mathcal{L}_M(\text{SCF})$. Therefore, we can generate all recursively enumerable languages over Σ.

Let $\Gamma = \{a_1, a_2, \ldots, a_k\}$ be an arbitrary alphabet. Then every recursively enumerable language L over Γ is equal to $g^{-1}(L')$, for some recursively enumerable $L' \subseteq \Sigma^*$, where g maps each letter a_i to 0^i1, for $1 \leq i \leq k$. And since every full semi-AFL is closed under inverse homomorphism, we can generate every recursively enumerable language using M for synchronization. Hence, $\mathcal{L}_M(\text{SCF})$ is equal to the family of recursively enumerable languages. □

To finish off this section, we see that we can use standard encoding techniques in order to use only a two letter transducer alphabet. The proof is omitted.

Lemma 14. *Let M be a transducer on S^* with $|S| > 2$. There exists M' over the alphabet $\{s, r\}$ such that $\mathcal{L}_M(\text{SCF}) \subseteq \mathcal{L}_{M'}(\text{SCF})$.* □

Then, starting with the fixed transducer from Proposition 13, we obtain:

Corollary 15. *There exists a fixed transducer M over a two letter alphabet such that $\mathcal{L}_M(\text{SCF})$ is equal to the recursively enumerable languages.* □

4 Unary Transducers

We know that a fixed transducer over a two letter alphabet is enough to generate all recursively enumerable languages. The question remains as to what languages we can generate over unary transducers. We mentioned that if we examine the fixed unary transducer that outputs exactly the input, we generate the counter synchronized context-free languages, which gives a language family strictly between $\mathcal{L}(\text{E0L})$ and $\mathcal{L}(\text{ET0L})$. It remains to be seen whether or not we can generate languages that are not in this language family. We show next that we can.

We demonstrate an example of a more complex language that we can generate with a unary transducer. Consider the transducer M that on input s^n, nondeterministically outputs either s^n or s^{2n}. Consider the synchronized context-free grammar G defined by the following productions:

$$(A, s) \to (A, s)(A, s) \mid a(B, s)$$
$$(B, s) \to (B, s)(B, s) \mid b$$

where we use (A, s) as the initial nonterminal. Then, we can see that every tree in $L_M(G)$ is of the form of Figure 3, and this generates the language

$$L = \{\, (ab^{2^n})^{2^n} \mid n \geq 0 \,\}.$$

The key to this example is that for every M-acceptable tree, there exists $n > 0$, such that the situation sequence must be either s^n or s^{2n} for every path from root to leaf. But, when using productions of the form $(A, s) \to a(B, s)$, the terminal a is a leaf, but (B, s) is not. Therefore every leaf on the subtree of (B, s) must have a situation sequence that is strictly longer than the situation sequence of the leaf a. Therefore, for every tree, every time the letter a is used, the length of the situation sequence must be n above it, and every time b is used, the situation sequence must be of length $2n$.

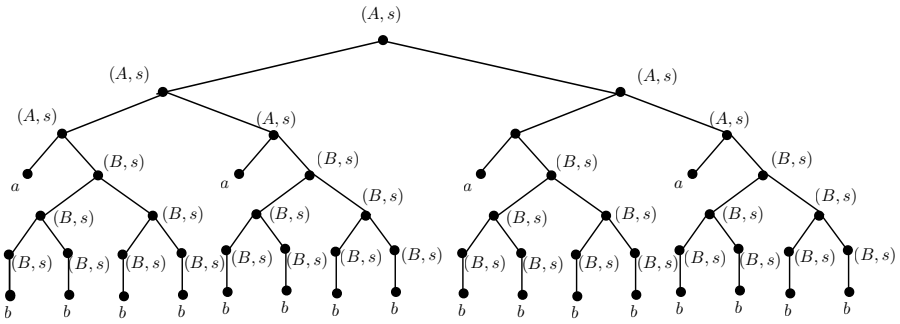

Fig. 3. A tree generating $(ab^4)^4$, where the grammar accepts $\{\, (ab^{2^n})^{2^n} \mid n \geq 0 \,\}$.

The following hierarchy is known

$$\mathcal{L}(\text{E0L}) \subsetneq \mathcal{L}(\text{cSCF}) \subsetneq \mathcal{L}(\text{ET0L}) \subsetneq \mathcal{L}(\text{INDEX}).$$

Here $\mathcal{L}(\text{INDEX})$ is the well-known family of indexed languages [1]. However, we see next that the language L above with M, is not even an indexed language, and therefore cannot be accepted with the simple unary or non-unary transducer that outputs exactly the input. The proof is omitted for reasons of space. It uses the shrinking lemma for indexed grammars from [3].

Proposition 16. *The language* $L = \{\, (ab^{2^n})^{2^n} \mid n \geq 0 \,\} \notin \mathcal{L}(\text{INDEX})$. □

Then, in particular, language L is not counter synchronized context-free and could not be generated by using a transducer that outputs exactly the input. In the next proposition we prove the following lower bound for any transducer, which utilizes a similar idea as to how to generate the above non-indexed language by a SCF grammar and a transducer. Again, the proof is omitted.

Proposition 17. *Let $G = (V, S, T, P, I)$ be an SCF grammar. Then $L_e(G)$ is an M-*SCF *language, for some fixed unary a-transducer M.* □

Intuitively, every time the original grammar uses a situation symbol, the new grammar uses four copies of the only situation symbol, and splits into two branches at either the second or third position of the four copies depending upon which situation symbol the original grammar is using. In this way, it is able to simulate two symbols of the original grammar with a single symbol and a more complex transducer. Then, since synchronized context-free grammars with equality synchronization is equal to the family of ET0L languages, we obtain:

Theorem 18. *There is a fixed unary transducer M such that $\mathcal{L}_M$(*SCF*) contains all* ET0L *languages.* □

The exact generative capacity when using fixed unary transducers, or all unary transducers is left open. A related question, is to determine the power of unit-productions $A \to B$ and λ-productions $A \to \lambda$, for nonterminals A and B in SCF and M-SCF grammars. For ordinary SCF grammars unit-productions seem vital to prove the equivalence between the equality and prefix synchronization. On the other hand, the equivalence of the family of SCF languages to the family of ET0L languages [9] can be used to eliminate λ-productions in SCF grammars. Whether the situation for unit- and λ-productions is similar in the case of M-SCF grammars (general or unary) must be clarified by further research.

References

1. Aho, A.V.: Indexed grammars—an extension of context-free grammars. J. ACM 15(4), 647–671 (1968)
2. Bordihn, H., Holzer, M.: On the computational complexity of synchronized context-free languages. J. Univ. Comput. Sci. 8(2), 119–140 (2002)
3. Gilman, R.H.: A shrinking lemma for indexed languages. Theoret. Comput. Sci. 163(1–2), 277–281 (1996)
4. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979)
5. Hromkovič, J., Karhumäki, J., Rovan, B., Slobodová, A.: On the power of synchronization in parallel computations. Discrete Appl. Math. 32, 155–182 (1991)
6. Hromkovič, J., Rovan, B., Slobodová, A.: Deterministic versus nondeterministic space in terms of synchronized alternating machines. Theoret. Comput. Sci. 132, 319–336 (1994)
7. Jürgensen, H., Salomaa, K.: Block-synchronized context-free grammars. In: Du, D.Z., Ko, J.I. (eds.) Advances in Algorithms, Languages, and Complexity, pp. 111–137. Kluwer (1997)
8. McQuillan, I.: Descriptional complexity of block-synchronization context-free grammars. J. Autom., Lang. Comb. 9(2/3), 317–332 (2004)
9. McQuillan, I.: The generative capacity of block-synchronized context-free grammars. Theoret. Comput. Sci. 337(1–3), 119–133 (2005)
10. Rozenberg, G., Salomaa, A.: The Mathematical Theory of L Systems, Pure and Applied Mathematics, vol. 90. Academic Press (1980)
11. Salomaa, A.: Formal Languages. ACM Monograph Series, Academic Press (1973)
12. Salomaa, K.: Synchronized tree automata. Theoret. Comput. Sci. 127, 25–51 (1994)