View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by University of Szeged

Acta Cybernetica **23** (2017) 349–369.

Regular Expressions for Muller Context-Free Languages*

Kitti Gelle[a] and Szabolcs Iván[a]

Abstract

Muller context-free languages (MCFLs) are languages of countable words, that is, labeled countable linear orders, generated by Muller context-free grammars. Equivalently, they are the frontier languages of (nondeterministic Muller-)regular languages of infinite trees.

In this article we survey the known results regarding MCFLs, and show that a language is an MCFL if and only if it can be generated by a so-called $\mu\eta$-regular expression.

Keywords: Muller context-free languages, well-ordered induction, regular expressions

1 Introduction

A word, also called "arrangement" in [9], is the isomorphism type of a labeled linear order. Thus, this notion is a generalization of finite and ω-words, permitting e.g. labelings of the integers of the rationals.

Finite automata on ω-words have by now a vast literature, see [21] for a comprehensive treatment. Finite automata acting on well-ordered words longer than ω have been investigated in [1, 6, 7, 24, 25], to mention a few references. Recently, the theory of automata on well-ordered words has been extended to automata on all countable words, including scattered and dense words. In [2, 5, 4], both operational and logical characterizations of the class of languages of countable words recognized by finite automata were obtained.

Context-free grammars generating ω-words were introduced in [8] and subsequently studied in [3, 19]. Context-free grammars generating arbitrary countable words were defined in [10, 11]. Actually, two types of grammars were defined, context-free grammars with Büchi acceptance condition (BCFG), and context-free grammars with Muller acceptance condition (MCFG). These grammars generate the Büchi and the Muller context-free languages of countable words, abbreviated as BCFLs and MCFLs. Every BCFL is clearly an MCFL, but there exists an

*This work was supported by NKFI grant no. 108448.

[a]University of Szeged, Hungary, E-mail: {kgelle,szabivan}@inf.u-szeged.hu

DOI: 10.14232/actacyb.23.1.2017.19

MCFL of well-ordered words that is not a BCFL, for example the set of all countable well-ordered words over some alphabet. In fact, it was shown in [10] that for every BCFL L of well-ordered words there is an integer n such that the order type of the underlying linear order of every word in L is bounded by ω^n.

In this paper we survey the results on Muller context-free languages, and we give an operational ("regular-expression-like") characterization of this class.

2 Notation

2.1 Linear orderings

A (strict) *linear ordering* is a pair $(I, <)$ where $<$ is a strict total ordering on I, that is, an irreflexive, transitive and trichotomous relation. Set-theoretical properties of the domain set I, such as finiteness, membership and cardinality are lifted to the ordering $(I, <)$ thus we can say e.g. that a linear ordering is countable, finite etc. In order to ease notation, we will omit the ordering $<$ from the pair and identify the ordering $(I, <)$ with its domain I, if the ordering relation is not important or is clear from the context. In this paper we will (unless stated otherwise) only deal with countable orderings. A good reference for linear orderings is [23].

An *embedding* of a linear ordering $(I, <)$ into a linear ordering $(J, \prec)$ is a (necessarily) injective mapping $h : I \to J$ preserving the ordering, i.e. $x < y$ implying $h(x) \prec h(y)$. We write $(I, <) \leq (J, \prec)$ if I can be embedded into J. Clearly, this $\leq$ relation is a preorder (a reflexive and transitive relation) between orderings. A surjective embedding is called an *isomorphism* between the orderings I and J; if there exists an isomorphism between I and J, then they are called *isomorphic*. Isomorphism of linear orderings is an equivalence relation, the classes of which are called *order types*. Well-known isomorphism types are the order type ω of the natural numbers $\mathbb{N}$, the type $-\omega$ of the negative integers, the type ζ of integers and the type η of rationals. Since isomorphism is compatible with embeddability, we can say that an order type α can be embedded into an order type β, written as $\alpha \leq \beta$. Note that this relation is not antisymmetric in general as the orderings $(0, 1)$ and $[0, 1]$ can be embedded into each other but they are not isomorphic as the latter one has a least element while the former one does not.

The ordering $(I, <)$ is a *sub-ordering* of $(J, \prec)$ if $I \subseteq J$ and $<$ is the restriction of $\prec$ onto I. An ordering is a *well-ordering* if it has no sub-ordering of type $-\omega$, is *scattered* if it has no sub-ordering of type η, *quasi-dense* if it is not scattered and *dense* if it has at least two elements and for any $x < y$ there exists some z with $x < z < y$. All dense countable orderings have the type η, possibly enriched with either a least or a greatest element (or both). Order types of well-orderings are called *ordinals*. Amongst the class of ordinals, the embeddability relation itself is a well-ordering, moreover, each ordinal α is either a *successor ordinal* $\alpha = \beta + 1$ for some ordinal β, in which case there is no other ordinal between α and β, or is a *limit ordinal* with $\alpha = \bigvee_{\beta<\alpha} \beta$, i.e. is the least upper bound of the set of ordinals strictly smaller than α with respect to the embeddability relation $<$. Note that although

the ordinals themselves form a proper class, whenever α is an ordinal, then the class of ordinals smaller than β is a set and whenever X is a set of ordinals, then their least upper bound $\bigvee X$ exists and is an ordinal. Moreover, the class of countable ordinals is the least class which contains the order types 0 and 1 and is closed under taking ω-sums, that is, taking suprema of ω-chains.

When $(I, \prec)$ is some linearly ordered index set and for each $i \in I$, $(J_i, <_i)$ is a linear order, then their *ordered sum* $(J, <) = \sum_{i \in I} (J_i, <_i)$ has the *disjoint union* $J = \{(i, j) : i \in I, j \in J_i\}$ as domain with $(i, j) < (i', j')$ if either $i \prec i'$ or $i = i'$ and $j <_i j'$. In particular, $(J_1, <_1) + (J_2, <_2)$ denotes the sum $\sum_{i \in \{1,2\}} (J_i, <_i)$. It is clear that a well-ordered sum of well-ordered orderings is well-ordered, and a scattered sum of scattered orderings is scattered. The sum operation is also compatible with the isomorphism, thus it can be extended to order types. For example, $-\omega + \omega = \zeta$ and $\eta + \eta = \eta + 1 + \eta = \eta$ where 1 is the order type of the singleton orderings; in general, n is the order type of the n-element orderings for $n \in \mathbb{N}_0 = \{0, 1, \ldots\}$. If α is the order type of I and β is the order type of each J_i, $i \in I$, then $\beta \times \alpha$ stands for the order type of the sum $\sum_{i \in I} J_i$. Hence, product of ordinals is an ordinal. Ordinals are also equipped with an exponentation operator but we will only use finite powers of the form α^n, with n being an integer, that is, $\alpha^0 = 1$ and $\alpha^{n+1} = \alpha^n \times \alpha$.

Hausdorff classified the scattered order types into an infinite hierarchy. We make use of the following variant [17] of this hierarchy: let VD_0 be the class of all finite order types, and when α is some ordinal, then let VD_α consist of the class of all order types that can be written as $\sum_{i \in \zeta} I_i$ where each I_i is a member of VD_{β_i} for some ordinal $\beta_i < \alpha$. Hausdorff's theorem states that a (countable) order type is scattered if and only if it is contained in VD_α for some (countable) ordinal α. The *Hausdorff-rank* $\mathrm{rank}(o)$ of a scattered order type o is the least ordinal α with $o \in VD_\alpha$.

2.2 Words, tree domains, trees

An *alphabet* is a finite nonempty set of symbols, or *letters*. Alphabets are usually denoted Σ, Γ in this paper, and letters are denoted by $a, b, c, \ldots$. A Σ-labeled linear ordering is a tuple $w = (\mathrm{dom}(w), \ell_w)$ where $\mathrm{dom}(w) = (I, <)$ is some linear ordering and $\ell_w : I \to \Sigma$ is the labeling function of w. We usually identify w with ℓ_w and write $w(i)$ for $\ell_w(i)$, $i \in \mathrm{dom}(w)$. Two words u and v are *isomorphic* if there is an isomorphism $h : \mathrm{dom}(u) \to \mathrm{dom}(v)$ which preserves also the labels, that is, $u(i) = v(h(i))$ for each $i \in \mathrm{dom}(u)$. A *word* over Σ is an isomorphism class of countable Σ-labeled linear orderings. For convenience, when u is a word, we take a representant of u and use the notation $\mathrm{dom}(u)$, ℓ_u referring the domain and labeling of the representant. Order theoretic properties are lifted to words. The set of all countable, ω- and finite words over Σ are respectively denoted $\Sigma^\#$, Σ^ω and Σ^*. In particular, ε denotes the empty word (having the empty set as domain). Note that as for any Σ, there is a unique η-word u such that for any $x < y \in \mathrm{dom}(u)$ and

letter $a \in \Sigma$ there is some $z \in \mathrm{dom}(u)$ with $x < z < y$ and $u(z) = a$, this u being called the *perfect shuffle of* Σ, and every countable Σ-word v is a *subword* of thus u (that is, $\mathrm{dom}(v)$ is a sub-ordering of $\mathrm{dom}(u)$ and ℓ_v is the restriction of ℓ_u to $\mathrm{dom}(v)$), thus $\Sigma^{\#}$ is indeed a set. A *language* over Σ is an arbitrary subset of $\Sigma^{\#}$. Languages will usually be denoted by $K, L, \ldots$.

When $(I, <)$ is a linear ordering and for each $i \in I$, w_i is a Σ-word, then their *product* or (con)catenation is the word $w = \prod_{i \in I} w_i$ with domain $\sum_{i \in I} \mathrm{dom}(w_i)$ and the labeling of ℓ_w being the *source tupling* of the labelings ℓ_{w_i}, that is, for (i, j), $i \in I$, $j \in \mathrm{dom}(w_i)$ let $w(i, j)$ be $w_i(j)$. Product is extended to languages in the expected way: when $(I, <)$ is the indexing ordering and to each $i \in I$, $L_i \subseteq \Sigma^{\#}$ is a language, then $\prod_{i \in I} L_i$ consists of those words $\prod_{i \in I} w_i$ where $w_i \in L_i$ for each $i \in I$. Binary products are simply written as $u \cdot v$ or $K \cdot L$, or simply uv and KL. When α is some order type, then L^α is the language $\prod_{i \in \alpha} L$, and L^* stands for the union of the languages L^n, n being a natural number. Also, L^+ stands for $\bigcup_{n>0} L^n$.

A *tree domain* is a prefix closed nonempty (but possibly infinite) subset T of $\mathbb{N}^*$. That is, whenever $x \cdot i$ is in T for $x \in \mathbb{N}^*$ and $i \in \mathbb{N}$, then so is x, in which case x is the *parent* of $x \cdot i$ and $x \cdot i$ is a *child* of x. Members of T are also called *nodes* of T. If $x \cdot y \in T$ for $x, y \in \mathbb{N}^*$, then x is called an *ancestor* of $x \cdot y$ and $x \cdot y$ is a *descendant* of x, denoted $x \preceq x \cdot y$. If y is nonempty, then we talk about *proper* ancestor / descendant, denoted $x \prec x \cdot y$. Nodes of T having no child are called *leaves* of T, the other nodes are called *inner nodes* of T. Clearly, ε is a member of every tree domain.

Subsets of a tree domain T which are tree domains themselves are called *prefixes* of T. A *path* π of T is a prefix in which every node has at most one child. Clearly, to every path π there exists a unique word u_π in $\mathbb{N}^{\leq \omega} = \mathbb{N}^* \cup \mathbb{N}^\omega$ such that $\pi = \{x \in \mathbb{N}^* : x \preceq u_\pi\}$. We will use π and u_π interchargably.

When T is a tree domain and $x \in T$, then the *sub-tree domain* of T is $T|_x = \{y \in \mathbb{N}^* : xy \in T\}$. It is clear that $T|_x$ is also a tree domain, and is a path if so is T.

A (Σ-)*tree* over some alphabet Σ is a labeled tree domain t, that is, a pair $(\mathrm{dom}(t), \ell_t)$ where $\mathrm{dom}(t)$ is a tree domain and $\ell_t : \mathrm{dom}(t) \to \Sigma$ is a labeling function. Similarly to the case of words, we often identify t with ℓ_t and write $t(x)$ in place of $\ell_t(x)$ for $x \in \mathrm{dom}(t)$. Notions of tree domains (nodes, paths, sub-tree domains etc.) are lifted to trees; the *subtree* of t rooted at some node $x \in \mathrm{dom}(t)$ has the domain $\mathrm{dom}(t)|_x$ and labeling $y \mapsto t(xy)$. When $X \subseteq \mathrm{dom}(t)$ is a set of nodes of t, then $\mathrm{labels}(t, X) = \{t(x) : x \in X\}$ is the set of labels occurring on the nodes belonging to X, and $\mathrm{infLabels}(t, X)$ is the set of labels occurring infinitely many times. In particular, if π is a path of t, then a letter $a \in \Sigma$ belongs to $\mathrm{infLabels}(t, \pi)$ if and only if for each $x \in \pi$ there exists some descendant $y \in \pi$ of x with $t(y) = a$. When t is clear from the context, we just write $\mathrm{labels}(X)$ and $\mathrm{infLabels}(X)$, respectively. For any infinite path π, there exists a $\preceq$-minimal node x of π with $\mathrm{infLabels}(\pi|_x) = \mathrm{labels}(\pi|_x)$, this node x is denoted $\mathrm{head}(\pi)$. Clearly,

$\mathrm{infLabels}(\pi) = \mathrm{infLabels}(\pi|_x) \subseteq \mathrm{labels}(\pi|_x) \subseteq \mathrm{labels}(\pi)$ for every path π and node $x \in \pi$.

2.3 Muller context-free grammars and languages

A *Muller context-free grammar* or MCFG for short, is a tuple $G = (N, \Sigma, P, S, \mathcal{F})$ where N and Σ are the disjoint alphabets of *nonterminals* (or variables) and *terminals*, respectively, $S \in N$ is the *start symbol*, P is the finite set of *productions* (or rules) of the form $A \to \alpha$ with $A \in N$ and $\alpha \in (N \cup \Sigma)^+$, and $\mathcal{F} \subseteq P(N)$ is the Muller *acceptance condition*. Here $P(N) = \{N' \subseteq N\}$ is the *power set* of N.

Observe that we explicitly disallow rules of the form $A \to \varepsilon$ here; this makes the treatment of leaf labels more uniform, and as it turns out, such rules can be mimicked by introducing a fresh nonterminal I, rules $A \to I$ and $I \to I$ and adding $\{I\}$ to the acceptance condition. Nevertheless, in our examples we will make use of rules $A \to \varepsilon$ in order to help readability.

An $(N \cup \Sigma)$-tree t is *locally consistent* (with G) if it satisfies the following condition: each inner node x of t is labeled by some nonterminal A and the set of children of x is $\{x{\cdot}1, \ldots, x{\cdot}n\}$ for some integer $n > 0$ with $A \to t(x{\cdot}1)t(x{\cdot}2)\ldots t(x{\cdot}n)$ being a production in P. A locally consistent tree is *complete* if its leaves are labeled in Σ. The leaves of any tree domain are linearly ordered by the *lexicographic ordering* $<_\ell$, that is, $u <_\ell v$ if and only if $u = u_1 \cdot i \cdot u_2$ and $v = u_1 \cdot j \cdot u_3$ for some words $u_1, u_2, u_3 \in \mathbb{N}^*$ and integers $i < j$. The *frontier word* of a tree t is the word $\mathrm{fr}(t)$ having the set of leaves as domain, ordered lexicographically, and labeling inherited from t. That is, $\mathrm{dom}(\mathrm{fr}(t)) = \{x \in \mathrm{dom}(t) : x \text{ is a leaf of } t\}$ and $\mathrm{fr}(t)(x) = t(x)$ for each leaf x.

A locally consistent tree t is a *derivation tree* of G if for each infinite path π of t the set $\mathrm{infLabels}(\pi)$ belongs to the acceptance condition $\mathcal{F}$. Given a symbol $X \in N \cup \Sigma$, we let $\Delta(G, X)$ denote the set of all complete derivation trees t of G whose *root symbol* $t(\varepsilon)$ is X. We write $A \Rightarrow_G^\infty \alpha$ for a symbol $A \in N \cup \Sigma$ and a word $\alpha \in (N \cup \Sigma)^\#$ if α is the frontier word of some derivation tree of G having root symbol A. The *language generated by* G is $L(G) = \{w \in \Sigma^\# : S \Rightarrow_G^\infty w\}$.

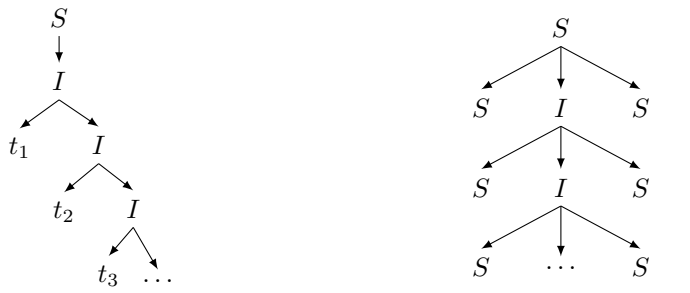
A language $L \subseteq \Sigma^\#$ of Σ-words is a *Muller context-free language*, or MCFL for short, if $L = L(G)$ for some MCFG G. In fact, Muller context-free languages are precisely the frontier languages of (nondeterministic Muller-)regular languages of infinite trees.

Example 1. If $G = (\{S, I\}, \Sigma, P, S, \{\{I\}\})$, with

$$P = \{S \to a : a \in \Sigma\} \cup \{S \to \varepsilon, S \to I, I \to SI\},$$

then $L(G)$ consists of all the well-ordered words over Σ.

Indeed, assume $t_1, t_2, \ldots$ are derivation trees. Then so is the tree t depicted in Figure 1a with frontier word $\mathrm{fr}(t_1)\mathrm{fr}(t_2)\ldots$. Thus, $L(G)$ contains the empty word (by $S \to \varepsilon$), the words of length 1 (by $S \to a$ and $S \to b$), and is closed under taking ω-products. Since the least class of order types which contains 0, 1 and which



(a) Tree for Example 1 (b) Tree for Example 2

Figure 1: Derivation trees corresponding to Examples 1 and 2

is closed under ω-sums is the class of all countable ordinals (see e.g. [23]), $L(G)$ contains all the well-ordered words over $\{a, b\}$. For the other direction, assume t is a derivation tree having a frontier word containing an infinite descending chain $\ldots <_\ell u_2 <_\ell u_1$. Then let us define the path $v_0, v_1, \ldots$ in t: $v_0 = \varepsilon$ and v_{i+1} is $v_i \cdot 1$ if this node is an ancestor of infinitely many u_j and $v_i \cdot 2$ otherwise (which happens if v_i corresponds to the production $I \to SI$ and the node $v_i \cdot 1$ (which is labeled S) has no descendant of the form u_j at all). Note that for each u_j there exists a unique v_{i_j} such that v_{i_j} is an ancestor of u_j and v_{i_j+1} is not, since the length of the words v_i grows without a bound. Now these nodes v_{i_j} correspond to the production $I \to SI$ and $v_{i_j+1} = v_{i_j} \cdot 1$, so that the successor of v_{i_j} along the path is labeled by S. Hence $v_0, v_1, \ldots$, is a path π in t such that $\mathrm{infLabels}(\pi)$ contains S, which is a contradiction since the only accepting set is $\{I\}$.

Example 2. If $G = (\{S, I\}, \Sigma, P, S, \{\{I\}\})$, with

$$P = \{S \to a : a \in \Sigma\} \cup \{S \to \varepsilon, S \to I, S \to SIS\},$$

then $L(G)$ consists of all the scattered words over Σ.

Indeed, the derivation tree depicted on Figure 1b shows that $L(G)$ is closed under $\omega + (-\omega)$-products of words. Since $\varepsilon \in L(G)$, the language is thus closed under ω-products and $-\omega$-products as well, thus closed under ζ-products. Since the one-letter words belong to $L(G)$, we get by Hausdorff's Theorem that $L(G)$ consists of all the scattered Σ-words.

We note that if instead of the Muller condition we define a Büchi-type acceptance condition, then we get a weaker device: the resulting class of "Büchi context-free languages" is strictly contained within the class of MCFLs, see [11, 10].

3 MSO-definable properties are decidable

The logic usually arising when dealing with "regular" structures is that of *monadic second-order* logic, or MSO. In [13] the following general decidability theorem was

proved:

Theorem 1. *The following problem is decidable: given an MCFG G generating Σ-words and an MSO formula φ evaluated Σ-words, does it hold that $w \models \varphi$ for every $w \in L(G)$?*

Thus in particular, it is decidable whether G generates scattered, or well-ordered, or dense words only.

We sketch the outline of the proof. First, to each MCFG $G = (N, \Sigma, P, S, \mathcal{F})$ we associate the grammar $G' = (N \cup P, \Sigma, P', S)$ where P' contains the following set of productions:

- For each nonterminal $A \in N$, there is a production $A \to (A \to \alpha_1)(A \to \alpha_2) \dots (A \to \alpha_n)$ in P' where $A \to \alpha_1, \dots, A \to \alpha_n$ are the productions in P having A on their left-hand side, in some fixed order.
- For each production $A \to X_1 \dots X_n$, there is a production $(A \to X_1 \dots X_n) \to X_1 \dots X_n$ in P'.

That is, we can rewrite a nonterminal to the sequence of its alternatives, and rewrite a production to its right-hand side. Thus, each nonterminal of G' has exactly one alternative (it is assumed that for each nonterminal A there is at least one production having left-hand side A), thus there is exactly one locally consistent tree of G' having root symbol S. We call this unique tree t_G the *grammar tree* of G.

Example 3. For the MCFG of Example 1, this tree t_G is depicted in Figure 2.

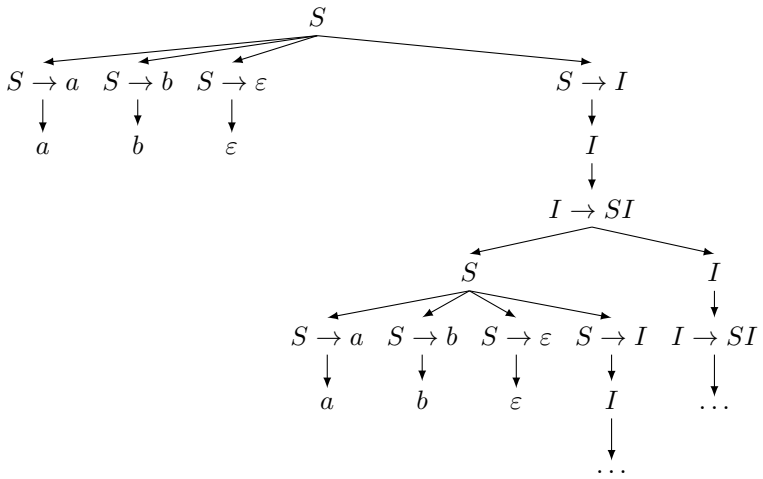


Figure 2: Grammar tree of the MCFG of Example 1

The cruical fact is that the grammar tree is a *regular* tree, having finitely many (more precisely, at most $|\Sigma| + |N| + |P|$) subtrees. By [22], the MSO theory of a regular tree is decidable, that is, given a regular tree t (which is t_G in our case) and an MSO formula ψ, it is decidable whether $t \vDash \psi$ holds. As t_G can be effectively constructed from G, it remains to construct a formula ψ from G and φ such that $t_G \vDash \psi$ holds if and only if for each $w \in L(G)$ we have $w \vDash \varphi$.

Informally, the formula ψ constructed in [13] has the semantics "whenever T is a derivation tree of G, its frontier word satisfies φ". Now derivation trees are encoded as subsets of $\mathrm{dom}(t_G)$ as follows: a subset $X \subseteq \mathrm{dom}(t_G)$ encodes a derivation tree of G if the following conditions all hold:

- X contains the root node.
- If some $x \in X$ is labeled by some nonterminal $A \in N$, then *exactly one* child of x belongs to X.
- If some $x \in X$ is labeled by some production $A \to \alpha$, then *all the children* of x belong to X.
- On each infinite path $\pi \subseteq X$, the set of symbols from N occurring infinitely many times belongs to $\mathcal{F}$.

These properties can be expressed in MSO and such a set encodes a derivation tree in the obvious way.

Example 4. Figure 3 shows a (part of a) derivation tree t of the grammar of Example 1 and the corresponding subset T of $\mathrm{dom}(t_G)$ (as nodes in boldface).

Then, given a set X of nodes of t_G, we can define the subset $Y \subseteq X$ of the leaves in X and the lexicographic ordering over this Y is also MSO-definable. Hence the formula "whenever X is a subset of $\mathrm{dom}(t_G)$ encoding a derivation tree, and Y is the set of leaves within X, then the word corresponding to Y satisfies φ" is expressible in MSO (moreover, is effectively computable from G and φ), thus proving Theorem 1.

4 A normal form

The general decidability theorem of the previous section does not give us an exact complexity result as model checking MSO is decidable, but nonelementary in general. In this section we give complexity results for several decision problems (and several undecidability results as well) regarding MCFLs, surveying the results of [11]. The decidable properties surveyed here are MSO-definable, thus their decidability is immediate from Theorem 1. For example, $L(G)$ is empty if and only if every member w of $L(G)$ satisfies the false formula $\downarrow$. (On the other hand, universality is not definable this way – and indeed, universality of MFCGs is already undecidable for singleton alphabets.)

(A slightly modified variant of) the normal form of MCFGs introduced in [11] is the following:

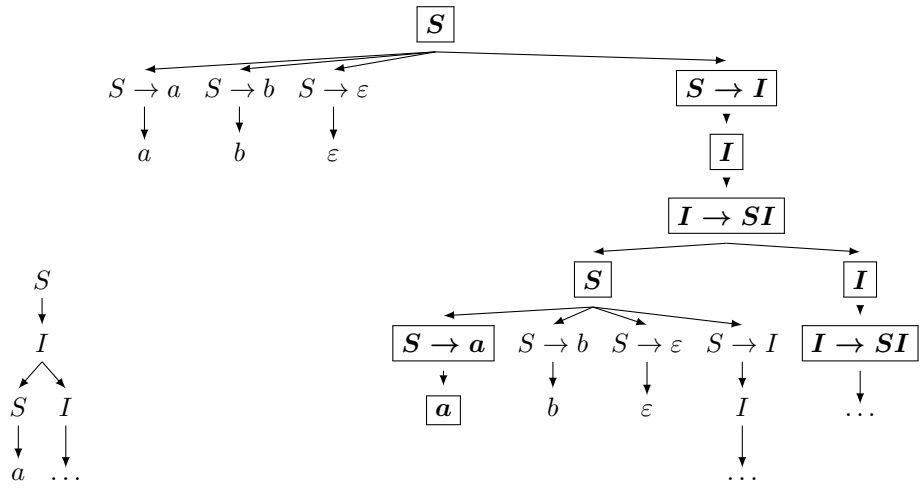


Figure 3: A part of a derivation tree t of Example 1 and the corresponding subset $\boldsymbol{T}$ of t_G

Definition 1. *An MCFG $G = (N, \Sigma, P, S, \mathcal{F})$ is in* normal form *if either P is empty, or P consists of the single production $S \to \varepsilon$, or for each $A \in N$ there exists a nonempty word w with $A \Rightarrow_G^\infty w$ and words u, v with $S \Rightarrow_G^\infty uAv$.*

Moreover, to each $F \in \mathcal{F}$ there exists a path π of some derivation tree t with $\mathrm{infLabels}(\pi) = F$.

In the terminology of [11], each nonterminal has to be +-productive and reachable, and each accepting set has to be viable.

Such a normal form of any MCFG is computable:

Theorem 2 ([11])**.** *For any MFCG G, an equivalent MCFG G' in normal form can be computed in* **PSPACE**. *The resulting grammar G' has a size polynomially bounded by the size of G.*

We sketch the algorithm here. The straightforward modification of the corresponding algorithms for ordinary context-free grammars works: first we check for each symbol X whether X is *productive* (is there a complete derivation tree with root symbol X at all). However, the complexity of this problem is **PSPACE**-complete due to the fact that the emptiness problem of Muller regular tree languages, given by a nondeterministic Muller tree automaton, is **PSPACE**-complete. Since there is a polynomial-time transformation from an MCFG to a corresponding Muller tree automaton and vice versa, we gain **PSPACE**-completeness for deciding productiveness of individual nonterminals, and emptiness of MCFLs as well:

Proposition 1. *[11] Deciding whether $L(G) = \emptyset$ is* **PSPACE**-*complete.*

Then, we can throw away all the non-productive nonterminals and get rid of all the productions that have a non-productive nonterminal on either of its sides.

After that, this set is further reduced to the set of *reachable* nonterminals, which can be done by the usual fixed-point construction for CFGs, as for each reachable nonterminal A there is a finite (not necessarily complete) derivation tree with root symbol S and A occurring as a leaf label. Then we can throw away all the non-reachable nonterminals. This can be done in polynomial time.

In the next step, a nonterminal generates A a nonempty word if and only if there is a finite (not necessarily complete) derivation tree rooted A that has some letter $a \in \Sigma$ occurring as a leaf label. This can also be decided in polynomial time by solving a reachability problem. Then, we can simply erase all the symbols from the right-hand sides from which only the empty word can be generated (and if the right-hand side of a rule becomes empty, then erase the rule itself as well), arriving to a grammar in the required normal form, apart from viability of each $F \in \mathcal{F}$. (This way we might lose the word ε from $L(G)$ – if we need the nonempty word, we can allow the production $S \to \varepsilon$ to be present, but in that case S should not appear on the right-hand side of any rule, as in the classical case.)

The normal form can be generated in **PSPACE**. Then, $L(G)$ contains a nonempty word if and only if there are still productions in G.

Proposition 2 ([11])**.** *It can be decided in* **PSPACE** *whether $L(G)$ contains a nonempty word.*

Now for retaining only the "viable" accepting sets, the following associated graph Γ_G is handy:

Definition 2. *Given an MCFG $G = (N, \Sigma, P, S, \mathcal{F})$, we define the following edge-labeled multigraph Γ_G: the vertices of Γ_G are the nonterminals, and there is an edge from A to B labeled (α, β) for $\alpha, \beta \in (N \cup \Sigma)^*$ if $A \to \alpha B \beta$ is a production of G.*

Now if G already contains only productive and reachable nonterminals, then a set $F \in \mathcal{F}$ is viable if and only if the subgraph of Γ_G induced by F is strongly connected, which is efficiently decidable, finishing the construction of the normal form.

However, language universality (and thus language inclusion and equivalence) is undecidable already for singleton alphabets (contrary to the case of context-free grammars, where undecidability holds only for alphabets of size at least two):

Proposition 3 ([10])**.** *It is undecidable whether $L(G) = \Sigma^\#$ for an MCFG G, even when Σ is a singleton alphabet.*

In fact, the problem is undecidable already for Büchi context-free grammars. The key for proving this is a reduction from the universality problem of context-free languages of finite words over the binary alphabet $\{a, b\}$: first we encode a by a^ω and b by $a^{-\omega}$. Then, the language L of those words in $a^\#$ *not* belonging to $\{a^\omega, a^{-\omega}\}^*$ is a MCFG and thus, as MCFLs are effectively closed under homomorphisms and finite unions, we get that $L(G) = \{a, b\}^*$ for the CFG G if and only if $L(G') = a^\#$ for some MCFG G' effectively constructed from G.

4.1 Languages of finite words

Given an MCFG $G = (N, \Sigma, P, S, \mathcal{F})$ in normal form, one can decide in (low-degree) polynomial time whether $L(G)$ consists of finite words only. (Note that decidability is clear as the property of being scattered is MSO-definable.)

The key observation here is that $L(G)$ contains an infinite word if and only if there is some $F \in \mathcal{F}$ and an edge $A \xrightarrow{\alpha,\beta} B$ in Γ_G with $\alpha\beta \neq \varepsilon$ and $A, B \in F$ which can be verified efficiently.

Also, it is quite straightforward to see that a language $L \subseteq \Sigma^*$ is context-free if and only if it is an MCFL: for one direction we only have to set $\mathcal{F} = \emptyset$. For the other direction somewhat more care is needed since the frontier word of an infinite tree can be empty. However, it can be decided in **PSPACE** whether $A \Rightarrow_G^\infty \varepsilon$ holds for a nonterminal A: we only have to remove the productions from G having some terminal symbol occurring on the right-hand side (that is, we retain the productions of the form $X \to \alpha$ with $\alpha \in N^*$), and apply an emptyness check for the generated language. Then, for each A generating ε we can include the production $A \to \varepsilon$ and the resulting (classical) context-free grammar will generate $L(G) \cap \Sigma^*$ if G is in normal form.

Thus,

Proposition 4 ([11])**.** *A language $L \subseteq \Sigma^*$ is context-free if and only if it is Muller context-free.*

Also, since emptiness of CFGs is efficiently decidable, we have:

Proposition 5 ([11])**.** *It is decidable in* **PSPACE** *whether an MCFG generates at least one finite word.*

4.2 Languages of well-ordered words

Given an MCFG $G = (N, \Sigma, P, S, \mathcal{F})$ in normal form, one can decide in (low-degree) polynomial time whether $L(G)$ consists of well-ordered words only. Again, decidability itself is already clear, since the property is MSO-definable.

Proposition 6 ([11])**.** *For an MFCG G in normal form, $L(G)$ contains a word which is not well-ordered if there is some set $F \in \mathcal{F}$, nonterminals $A, B \in F$ and an edge $A \xrightarrow{\alpha,\beta} B$ in Γ_G with $\beta \neq \varepsilon$.*

To see this, suppose there is a derivation tree t of G with a frontier word *not* having a well-ordered domain. Then there exists an infinite descending chain $\ldots < x_3 < x_2 < x_1 < x_0$ of leaves of t. Starting from the root, one can then build up an infinite path $\pi = y_0, y_1, \ldots$ such that for each node u of π, an infinite number of these leaves x_i are descendants of u. (This property holds for the root, and at each step we set y_{i+1} to be the first child of y_i which is an ancestor of some x_j.) Then, as each such leaf x_i has some finite depth, there exists an y_j for each x_i such that y_j is an ancestor of x_i but y_{j+1} is not; it is easy to see that in this case x_i is "on the right side" of π.

Hence, for this π it holds that $F = \mathrm{infLabels}(\pi)$ is contained within a nontrivial strongly connected component of Γ_G, moreover, there is an edge $A \xrightarrow{\alpha,\beta} B$ with $A, B \in F$ and $\beta \neq \varepsilon$, the other direction being also straightforward, simply by following a closed path in F visiting each edge at least once, iterating ω times and complete the resulting derivation tree (which already has an infinite descending chain among its leaves due to $\beta \neq \varepsilon$).

Now if $L(G)$ contains well-ordered words only, one can compute an interval of ordinals containing all the order types of $L(G)$:

Proposition 7 ([16])**.** *If the MCFG G generates well-ordered words only, then both the* minimum *and the* supremum *of the order types of the members of $L(G)$ are effectively computable ordinals.*

For the minimum, first we note that to each nonterminal A, if there is a finite word w with $A \Rightarrow_G^\infty w$, then the length n of the shortest such word is computable. Then we can replace each occurrence of these nonterminals by a^n: the minimum order types for the nonterminals in the resulting grammar will coincide with those of G.

Let us fix for each symbol X a complete derivation tree t_X with root symbol X, minimizing the order type of $\mathrm{fr}(t_X)$. It turns out that the members of N can be partially ordered by some properties of these trees t_X and that these trees t_X can be chosen in a way that each subtree of t_X is some t_Y for $Y \in N \cup \Sigma$.

The key construction to see this is the following. When a t is a complete derivation tree of such an MCFG G, then we call t *simple* if it is either finite or has some infinite path π with $\mathrm{infLabels}(\pi) = \mathrm{labels}(\pi)$ and moreover, each *production* corresponding to the nodes of π occur infinitely many times on π. As $A \to uBv \in P$ for some $A, B \in F \in \mathcal{F}$ implies $v = \varepsilon$, this path π has to be the *rightmost* path of t. Then, the order type of $\mathrm{fr}(t)$ is the ω-sum of the order types of the frontiers of the subtrees of t being adjacent to π (that is, rooted at some node x not on π whose parent is on π). As each left-hand side occurs infinitely many times, we get that each nonterminal adjacent to π has a strictly smaller minimum. Thus, if t_X is simple, then we can define t_X after being defined each t_Y where the minimum order type of Y is strictly smaller than that of X, and the order type of its frontier is computable.

Now if t_X is *not* simple, then the order type of its frontier is the sum of the order types corresponding to its direct children. Now all these order types but the last have to be smaller then the order type of $\mathrm{fr}(t_X)$ and the last child has to have one level "closer" for being simple (and this level is finite), establishing the inductive case. Thus, a standard iterative algorithm recomputing the minima from the current estimations eventually terminates and produces the minimum ordinals (in Cantor normal form, say).

For the supremum, the case analysis is slightly more involved. First, one seeks for *reproductive* nonterminals: A is called reproductive if $A \Rightarrow_G^\infty \alpha$ for some α in which A occurs infinite times. It turns out that A is reproductive if and only if there is a production $A \to X_1 \dots X_n B$ and an accepting set $F \in \mathcal{F}$ with $A, B \in F$ and $X_i \Rightarrow_G^\infty uAv$ for some $i \in [n]$ and $u, v \in \Sigma^\infty$, which is decidable.

Then, if A is reproductive, then it is easy to see that arbitrarily large countable ordinals can be generated from A, thus in that case, the supremum in question is ω_1, the smallest uncountable ordinal. Also, if $A \Rightarrow_G^\infty uBv$ for some reproductive nonterminal B, then the same holds for A as well.

Otherwse, to each production of the above form, the nonterminals X_i belong to a strongly connected component strictly below the component of A, again setting a straightforward induction argument: the reader is referred to [16] for the technical details.

As ω-languages are also well-ordered, we note that and a language of ω-words is context-free in the sense of Cohen and Gold [8] if and only if it is an MCFL [11], and moreover, it is decidable whether an MCFG generates only well-ordered words having order type at most ω.

4.3 Languages of scattered words

Analogously for the case of well-ordered words, it is decidable whether an MCFG G generates scattered words only, as this property is also MSO-definable.

Proposition 8 ([11])**.** *It is decidable in polynomial time whether an MCFG G in normal form generates scattered words only.*

The key observation is the following: $L(G)$ contains a quasi-dense word if and only if there exists a *finite* derivation tree t, two leaves x and y of t and a viable accepting set $F \in \mathcal{F}$ such that $\text{labels}(\pi_x) = \text{labels}(\pi_y) = F$ and $t(x) = t(y) = t(\varepsilon)$ where π_x and π_y respectively denote the paths from the root to x and y. As this property is further equivalent to the existence of some production $A \to \alpha B \beta C \gamma$ with $A, B, C \in F$ for a viable $F \in \mathcal{F}$, we have an efficient decision procedure.

For languages of scattered words, the main ingredient of many proofs is that of the *Hausdorff-rank*. Basically, given a derivation tree t, we can tag each node x of t by the rank of $\text{fr}(t|_x)$. Then, consider the subset D of the nodes tagged by the same ordinal as the root. As an infinite sum of scattered orderings of the same rank α has a rank strictly greater than α, this subset D cannot contain an infinite antichain, yielding that D is a finite union of paths. Then, one can partially order the set of derivation trees primarily by the rank of their frontier, secondary by the number of paths covering their respective sets D, and third, by the depth of the first node of D having at least two children in D. The defined ordering becomes then a well-ordering of the derivation trees, allowing us to apply well-founded induction.

The first such application is the following "gap theorem" of MCFLs of scattered words:

Proposition 9 ([11])**.** *The supremum of the Hausdorff-rank of the members of $L(G)$ is computable when G generates scattered words only.*

Moreover, this supremum is either ω_1 or some natural number.

The key observation for this result is again that reproductive nonterminals, and only those, can produce words of arbitrarily large (countable) rank, and for the others, a simple induction works over the strongly connected components of Γ_G.

Interestingly, it is also known [14] that an MCFL consisting only of scatterred words is a BCFL if and only if the second case applies, i.e. if it has a finite upper bound n on the Hausdorff-rank of its members.

In order to define the regular expression-like expressions capturing these MCFLs, we will also consider *pairs* of words over an alphabet Σ, equipped with a *finite* concatenation and an ω-product operation. For pairs (u, v), (u', v') in $\Sigma^{\#} \times \Sigma^{\#}$, we define the product $(u, v) \cdot (u', v')$ to be the pair $(uu', v'v)$, and when for each $i \in \omega$, (u_i, v_i) is in $\Sigma^{\#} \times \Sigma^{\#}$, then we let $\prod_{i \in \omega} (u_i, v_i)$ be the word $\big(\prod_{i \in \omega} u_i\big)\big(\prod_{i \in -\omega} v_i\big)$. Let $P(\Sigma^{\#} \times \Sigma^{\#})$ denote the set of all subsets of $\Sigma^{\#} \times \Sigma^{\#}$. Then $P(\Sigma^{\#} \times \Sigma^{\#})$ is equipped with the operations of set union, concatenation $L \cdot L' = \{(u, v) \cdot (u', v') : (u, v) \in L,\ (u', v') \in L'\}$ and Kleene star $L^* = \{\epsilon\} \cup L \cup L^2 \cup \cdots$. We also define an ω-power operation $P(\Sigma^{\#} \times \Sigma^{\#}) \to P(\Sigma^{\#})$ by L^{ω} consisting of the words of the form $\prod_{i \in \omega} (u_i, v_i)$ with $(u_i, v_i) \in L$ for each $i \in \omega$.

The motivation behind this notion is the following. If t is a derivation tree with a distinguished leaf x labeled by some nonterminal A, and frontier word $\mathrm{fr}(t) = uAv$, then this frontier word is represented by the pair (u, v). Now if we substitute a tree s with root symbol A in place of the distinguished leaf, having frontier word $\mathrm{fr}(s) = u'Bv'$, yielding the tree t', then we have $\mathrm{fr}(t') = uu'Bv'v$ which is $(u, v) \cdot (u', v')$ according to the product operation we defined on pairs. Similarly, if the root of t is also labeled A, then we can iterate substituting t in place of the distinguished leaf: if we iterate a finite number of times, then the Kleene star contains the "pair representant" of the resulting tree; if we iterate ω times, then the frontier is $u^{\omega}v^{-\omega} = (u, v)^{\omega}$.

Then, let the set of $\mu\omega T_s$-expressions over the alphabet Σ be defined by the following grammar (with T being the initial nonterminal):

$$\begin{aligned} T &::= a \mid \varepsilon \mid x \mid T + T \mid T \cdot T \mid \mu x.T \mid P^{\omega} \\ P &::= T \times T \mid P + P \mid P \cdot P \mid P^* \end{aligned}$$

Here, $a \in \Sigma$ and $x \in \mathcal{X}$ for an infinite countable set of variables. An occurrence of a variable is *free* if it is not in the scope of a μ-operation, and *bound*, if it is not free. A *closed expression* does not have free variable occurrences. The semantics of these expressions are defined as expected using the monotone functions over $P(\Sigma^{\#})$ and $P(\Sigma^{\#} \times \Sigma^{\#})$ introduced earlier.

The characterization theorem of [12] states that these notions correspond to each other:

Theorem 3 ([12])**.** *A language L is an MCFL of scattered words if and only if it can be denoted by some closed $\mu\omega T_s$-expression.*

The direction that such expressions always denote MCFLs is done by a straightforward construction, while the converse direction is again done via the well-ordering of the derivation trees we introduced earlier.

5 Operational characterization of general MCFLs

In this section we give an operational characterization of MCFLs in general, using the operational characterization of Muller regular languages of infinite trees given in [18, 20] which we reproved using slightly different methods in [15].

There, we introduced for each *ranked* alphabet Σ (that is, each symbol $a \in \Sigma$ has some arity $n \geq 0$; the set of n-ary symbols of Σ is denoted Σ_n) the set of *$\mu\eta$-regular tree expressions* (tree expressions for short) over Σ as the least set satisfying all the following conditions:

- If $a \in \Sigma_n$, $n \geq 0$ and $E_1, \ldots, E_n$ are tree expressions over Σ, then $a(E_1, \ldots, E_n)$ is a tree expression over Σ. When $n = 0$, we write a in place of $a()$.
- If E and F are tree expressions over Σ, then so is $(E + F)$.
- If E is a tree expression over $\Sigma \cup \{x\}$ for the nullary symbol x, and F is a tree expression over Δ, then $(E \cdot_x F)$ is a tree expression over $\Sigma \cup \Delta$.
- If E is a tree expression over $\Sigma \cup \{x\}$ for the nullary symbol x, then $(\mu x.E)$ and $(\eta x.E)$ are tree expressions over Σ.

In order to define the semantics of these expressions, we have to define the operations of x-product, μx and ηx on tree languages, for which we use *cuts* and *decompositions* of trees. So let Σ be an alphabet and let $\widehat{\Sigma}$ stand for the disjoint copy $\{\widehat{a} : a \in \Sigma\}$ of Σ. The hatted symbols are of arity zero. Given a tree t, and a subset $X \subseteq \mathrm{dom}(t)$ of its nodes, the *X-cut of t* is the following $\Sigma \cup \widehat{\Sigma}$-tree t/X: a node x belongs to $\mathrm{dom}(t/X)$ if and only if x is a node of t which is *not* a proper descendant of any non-root member of X, i.e. $\mathrm{dom}(t/X) = \mathrm{dom}(t) - \bigcup_{u \in X - \{\varepsilon\}} \{y \in \mathrm{dom}(t) : u \prec y\}$. The labeling of t/X is defined as follows: for a node $x \in \mathrm{dom}(t/X)$ let $(t/X)(x)$ be $t(x)$ if $x \notin X - \{\varepsilon\}$ and $\widehat{t(x)}$ if $x \in X - \{\varepsilon\}$. That is, we "cut" the tree at the nodes of X and add a hat to the symbols occurring at the cut-points.

Example 5. Figure 4 shows a tree t with frontier $a^\omega b^{-\omega}$. Choosing the set X to contain all the inner nodes of t, all the trees of the form $(t|_x)/(X|_x)$ are the same (shown on the right hand side of the Figure). Figure 5 shows a (finite) tree t which gets decomposed into a set of six trees.

Now when K is a language of $\Sigma \cup \widehat{\Sigma}$-trees and L is a language of Δ-trees for some alphabets Σ and Δ, then $K[\widehat{\Sigma}/L]$ is the following language of $\Sigma \cup \Delta$-trees: a tree t belongs to $K[\widehat{\Sigma}/L]$ if there exists some subset $X \subseteq \mathrm{dom}(t)$ such that t/X belongs to K and for each $x \in X$, $t|_x$ belongs to L. (Usually Δ is either Σ or $\Sigma \cup \widehat{\Sigma}$). That is, if we can cut t such that the "retained part" of the tree belongs to K and all the subtrees that are "cut down" belong to L. Observe that if there exists such an X, then the subset X' of $\preceq$-minimal elements of X is also fine (as $t/X = t/X'$ then, and the condition for the subtrees is also valid), thus in that case we can assume that t is cut by some antichain of its nodes.

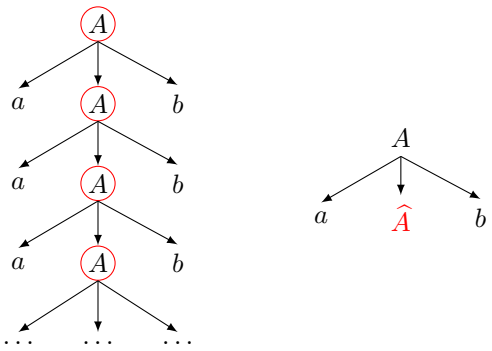


Figure 4: A tree and t one of its possible decompositions.

Given a tree language L over $\Sigma \cup \widehat{\Sigma}$, the function $K \mapsto L[\widehat{\Sigma}/K]$ is a monotone function (with respect to language inclusion), which maps the poset of $\Sigma \cup \widehat{\Sigma}$-tree languages to itself, thus it has a *least fixed point* that can be reached by the Kleene iteration $L_0 = \emptyset$, $L_\alpha = L[\widehat{\Sigma}/L_\beta]$ for successor ordinals $\alpha = \beta + 1$ and $L_\alpha = \bigcup_{\beta<\alpha} L_\beta$ for limit ordinals α, that is, there exists some (least) ordinal α such that L_α is the least fixed point of this function. This least fixed point is denoted L^μ and is a Σ-tree language. It is relatively easy to show that a Σ-tree t belongs to L^μ if and only if there is some subset $X \subseteq t$ of its nodes such that there is no infinite chain $x_1 \prec x_2 \prec \ldots$ in X and for each $x \in X$, the tree $(t|_x)/(X|_x)$ belongs to L.

The last operation on trees is the η-product. Given a $\Sigma \cup \widehat{\Sigma}$-tree language L, the language L^η is a language of Σ-trees: a tree t belongs to L^η if and only if there is some $X \subseteq \mathrm{dom}(t)$ of its nodes such that for each $x \in X$, the tree $(t|_x)/(X|_x)$ belongs to L. That is, now an arbitrary set of cut-points in the domain.

Now if L is a language of $\Sigma \cup \{x\}$-trees for a nullary symbol x, then $\mu x.L$ and $\eta x.L$ are the tree languages $\widehat{L}^\mu$ and $\widehat{L}^\eta$ where $\widehat{L}$ is the following language of $\Sigma \cup \widehat{\Sigma}$-trees: a tree t belongs to $\widehat{L}$ if and only if its projection defined by $a \mapsto a$, $\widehat{a} \mapsto x$ belongs to L. That is, the $\Sigma \cup \{x\}$-tree t' we get from t by relabeling each hatted symbol to x. Similarly, if K is a language of $\Sigma \cup \{x\}$-trees and L is some Δ-tree language, then let $K \cdot_x L$ be the language $\widehat{K}[\widehat{\Sigma}/L]$.

Example 6. When K consists of the tree single tree $A(x, \widehat{A})$, then K^η contains a single tree with root symbol A and frontier x^ω. Then, $K^\eta \cdot_x \{\widehat{A}\}$ contains a single tree with root symbol A and frontier $\widehat{A}^\omega$. Finally, the frontier language of $(K^\eta \cdot_x \{\widehat{A}\} \cup \{A(a), A(\widehat{A}, \widehat{A})\})^\mu$ contains all the nonempty well-ordered words over the singleton alphabet $\{a\}$.

After these definitions we are ready to define the semantics of tree expressions in the expected way. A tree expression E *denotes* a tree language $|E|$, a set of Σ-trees defined as follows:

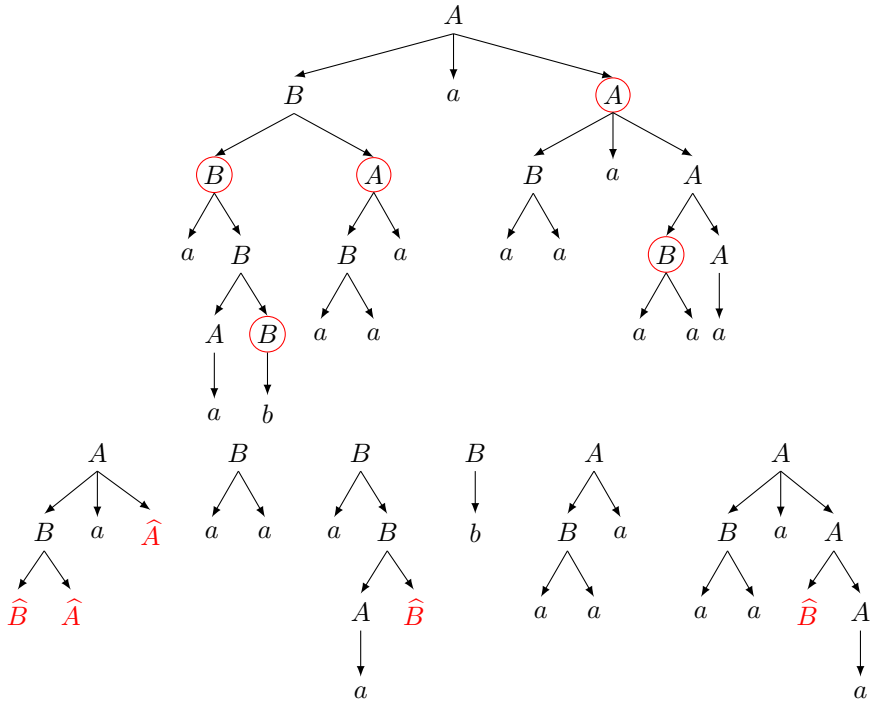


Figure 5: A decomposition of a finite tree.

- $|a(E_1, \ldots, E_n)|$ consists of those Σ-trees t whose root symbol is labeled a, the root have exactly the children $1, \ldots, n$ and for each $i \in [n]$, $t|_i \in |E_i|$.

- $|(E+F)| = |E| \cup |F|$, $|E \cdot_x F| = |E| \cdot_x |F|$, $|\mu x.E| = \mu x.|E|$ and $|\eta x.E| = \eta x.|E|$.

Now using the terms of [15], a tree language is Muller-regular if and only if it can be denoted by some $\mu\eta$-regular tree expression. Hence it is easy to derive $\mu\eta$-regular *word* expressions for MCFLs as MCFLs are exactly the frontier languages of Muller-regular tree languages. For this, we have to define operations corresponding to the $\cdot_x$, μx and ηx-operations above. Informally, for $\cdot x$ we can define the *substitution operation*: $K[x/L]$ contains those words we get from members of K in which we replace each occurrence of x by some word in L; then, $\mu x.L$ is the least fixed point of the monotone function $X \mapsto L[x/X]$; and, members of $\eta x.L$ are the Σ-words which we get by starting from the word x, then replacing each occurrence of x by some member of L, and repeat this process – the words occurring as "limits" of this (possibly infinite) replacement sequence are members of $\eta x.L$.

To treat the case of $\eta x.L$ formally, we introduce the class of *generalized* Σ*-trees* as follows. A *generalized tree domain* is a modified tree domain where we do not restrict the set of children of any node to be a finite linearly ordered set, but allow

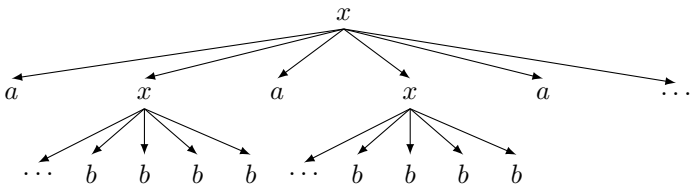


Figure 6: An L-x-substitution tree.

arbitrary countable linear orders. Nevertheless, each node has to have a finite depth.

More formally, given a partially ordered set $P = (P, <)$, a *P-tree domain* is a subset D of P^* satisfying the following conditions:

- D is nonempty and prefix-closed.
- For each node $d \in D$, the set $\{p \in P : d \cdot p \in D\}$ of the *children* of d is a linearly ordered subset of P.

When an alphabet Σ is also given, then a *P-Σ-tree* is a mapping $t : \mathrm{dom}(t) \to \Sigma$ from a P-tree domain to Σ, that is, a Σ-labeled P-tree domain and the *frontier word* of t is the Σ-word $\mathrm{fr}(t)$ whose domain is the set of the *leaves* of t (those nodes having no children) equipped with the *lexicographic* ordering: $p_1 \dots p_n <_\ell p'_1 \dots p'_m$ if and only if for some $i \leq m, n$ we have $p_i < p'_i$ and for each $j < i$, $p_j = p'_j$. Observe that this ordering is total on the leaves since for two different leaves $u = p_1 \dots p_n$ and $v = p'_1 \dots p'_m$ neither of them can be a prefix of the other, hence there exists a unique least index $i \leq m, n$ with $p_i \neq p'_i$; and as the set of the children of the node $p_1 \dots p_{i-1}$ is linearly ordered, it has to be either the case $p_i < p'_i$ or $p'_i < p_i$. Observe also that if each node has a countable children set, the $\mathrm{fr}(t)$ is a countable word.

Given a language $L \subseteq \big(\Sigma \cup \{x\}\big)^{\#}$, we define the languages $\mu x.L$ and $\eta x.L$ over Σ as follows. Let $P = \biguplus_{u \in L} \mathrm{dom}(u)$ be the disjoint union of the domains of all the words belonging to L. Then, an *L-x-substitition tree* is a P-$\big(\Sigma \cup \{x\}\big)$-tree satisfying the following conditions: the root is not a leaf node, each inner node is labeled by x, each leaf node is labeled in Σ and for each inner node u, the word formed by the labels of the children of u belongs to L.

Example 7. Figure 6 depicts an L-x-substitution tree where L is the language $\{b^{-\omega}, (ax)^\omega\}$.

Then, let $\eta x.L$ contain the frontier words of the L-x-substitution trees, and let $\mu x.L$ contain the frontier words of those L-x-substitution trees having no infinite paths.

Example 8. Figure 7 depicts an L-x-substitution tree for $L = \{axb\}$. This tree shows that $a^\omega b^{-\omega}$ is a member of $\eta x.L$ (but does not, in fact, belong to $\mu x.L$ as

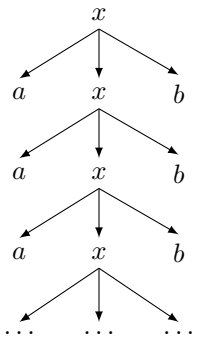


Figure 7: The word $a^\omega b^{-\omega}$ belongs to $\eta x.\{axb\}$.

it has an infinite path. For this language, $\mu x.L = \emptyset$. In contrast, $\mu x.\{axb, c\}$ is $\{a^n c b^n : n \geq 0\}$ and $\eta x.\{axb, c\}$ is $\mu x.\{axb, c\} \cup \{a^\omega b^{-\omega}\}$.)

These operations μ and η on languages over words correspond to the operations μ and η on tree languages in the sense $\mathrm{fr}(\mu x.L) = \mu x.\mathrm{fr}(L)$ and $\mathrm{fr}(\eta x.L) = \eta x.\mathrm{fr}(L)$ for each tree language L. We also make use of the $\cdot_x$ product operation: when $K \subseteq (\Sigma \cup \{x\})^\#$ and $L \subseteq \Delta^\#$ for the alphabets Σ and Δ, then $K \cdot_x L \subseteq (\Sigma \cup \Delta)^\#$ contains those words one can get from a word u in K by replacing each occurrence of x in u by some member of L. Or more technically, the frontier words of those $(K \cup L)$-x-substitution trees of depth at most two in which the word formed by the children of the root symbol belongs to K and each word formed by the children of the depth-one inner nodes belongs to L. In particular, the tree depicted in Figure 6 shows its frontier $(ab^{-\omega})^\omega$ belongs to $(ax)^\omega \cdot_x b^{-\omega}$.

Then also, $\mathrm{fr}(K \cdot_x L) = \mathrm{fr}(K) \cdot_x \mathrm{fr}(L)$ for arbitrary tree languages K and L.

By the characterization of Muller regular tree languages we get the following characterization:

Theorem 4. *A language $L \subseteq \Sigma^\#$ is an MCFL if and only if it can be generated from the singleton languages of one-letter words by a finite number of concatenation, binary union, $\cdot_x$-product, μx and ηx-operations.*

References

[1] Bedon, Nicolas. Finite automata and ordinals. *Theor. Comput. Sci.*, 156(1–2):119–144, 1996.

[2] Bedon, Nicolas, Bès, Alexis, Carton, Olivier, and Rispal, Chloé. *Logic and Rational Languages of Words Indexed by Linear Orderings*, pages 76–85. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[3] Boasson, Luc. Context-free sets of infinite words. In Weihrauch, Klaus, editor, *Theoretical Computer Science*, volume 67 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 1979.

[4] Bruyère, Véronique and Carton, Olivier. Automata on linear orderings. *J. Comput. Syst. Sci.*, 73(1):1–24, 2007.

[5] Bès, Alexis and Carton, Olivier. A Kleene theorem for languages of words indexed by linear orderings. In de Felice, Clelia and Restivo, Antonio, editors, *Developments in Language Theory*, volume 3572 of *Lecture Notes in Computer Science*, pages 158–167. Springer, 2005.

[6] Büchi, J. Richard. *The monadic second order theory of $\omega 1$*, pages 1–127. Springer Berlin Heidelberg, Berlin, Heidelberg, 1973.

[7] Choueka, Yaacov. Finite automata, definable sets, and regular expressions over ω^n-tapes. *Journal of Computer and System Sciences*, 17(1):81 – 97, 1978.

[8] Cohen, Rina S. and Gold, Arie Y. Theory of ω-languages. I. Characterizations of ω-context-free languages. *J. Comput. Syst. Sci.*, 15(2):169–184, 1977.

[9] Courcelle, Bruno. Frontiers of infinite trees. *ITA*, 12(4), 1978.

[10] Ésik, Zoltán and Iván, Szabolcs. Büchi context-free languages. *Theor. Comput. Sci.*, 412(8-10):805–821, 2011.

[11] Ésik, Zoltán and Iván, Szabolcs. On Müller context-free grammars. *Theor. Comput. Sci.*, 416:17–32, 2012.

[12] Ésik, Zoltán and Iván, Szabolcs. Operational characterization of scattered MCFLs. In Béal, Marie-Pierre and Carton, Olivier, editors, *Developments in Language Theory - 17th International Conference, DLT 2013, Marne-la-Vallée, France, June 18-21, 2013. Proceedings*, volume 7907 of *Lecture Notes in Computer Science*, pages 215–226. Springer, 2013.

[13] Ésik, Zoltán and Iván, Szabolcs. MSO-definable properties of Muller context-free languages are decidable. In Câmpeanu, Cezar, Manea, Florin, and Shallit, Jeffrey, editors, *Descriptional Complexity of Formal Systems - 18th IFIP WG 1.2 International Conference, DCFS 2016, Bucharest, Romania, July 5-8, 2016. Proceedings*, volume 9777 of *Lecture Notes in Computer Science*, pages 87–97. Springer, 2016.

[14] Ésik, Zoltán and Okawa, Satoshi. On context-free languages of scattered words. In Yen, Hsu-Chun and Ibarra, Oscar H., editors, *Developments in Language Theory - 16th International Conference, DLT 2012, Taipei, Taiwan, August 14-17, 2012. Proceedings*, volume 7410 of *Lecture Notes in Computer Science*, pages 142–153. Springer, 2012.

[15] Gelle, Kitti and Iván, Szabolcs. Expressions for regular languages of infinite trees. *Manuscript*, 2017.

[16] Iván, Szabolcs and Mészáros, Ágnes. Müller context-free grammars generating well-ordered words. In Dömösi, Pál and Iván, Szabolcs, editors, *Automata and Formal Languages, 13th International Conference, AFL 2011, Debrecen, Hungary, August 17-22, 2011, Proceedings.*, pages 225–240, 2011.

[17] Khoussainov, Bakhadyr, Rubin, Sasha, and Stephan, Frank. Automatic linear orders and trees. *ACM Trans. Comput. Logic*, 6(4):675–700, October 2005.

[18] Mostowski, Andrzej Wlodzimierz. Regular expressions for infinite trees and a standard form of automata. In *Symposium on Computation Theory*, pages 157–168, 1984.

[19] Nivat, Maurice. Sur les ensembles de mots infins engendrés par une grammaire algébrique. *ITA*, 12(3), 1978.

[20] Niwinski, Damian. Fixed points vs. infinite generation. In *Proceedings of the Third Annual IEEE Symposium on Logic in Computer Science (LICS 1988)*, pages 402–409. IEEE Computer Society Press, July 1988.

[21] Perrin, Dominique and Pin, Jean-Éric. Infinite words: automata, semigroups, logic and games, 2004.

[22] Rabin, Michael O. Decidability of second-order theories and automata on infinite trees. *Bull. Amer. Math. Soc.*, 74(5):1025–1029, 09 1968.

[23] Rosenstein, Joseph G. *Linear orderings.* Academic Press New York, 1981.

[24] Wojciechowski, Jerzy. Classes of transfinite sequences accepted by finite automata. *Fundamenta Informaticae*, 7:191–223, 1984.

[25] Wojciechowski, Jerzy. Finite automata on transfinite sequences and regular expressions. *Fundamenta Informaticae*, 8:379–396, 1985.