Operational characterization of scattered MCFLs
Technical Report

Zoltán Ésik and Szabolcs Iván
University of Szeged, Hungary

Abstract. We give a Kleene-type operational characterization of Muller context-free languages (MCFLs) of well-ordered and scattered words.

1 Introduction

A word, called ‘arrangement’ in [12], is an isomorphism type of a countable labeled linear order. They form a generalization of the classic notions of finite and \( \omega \)-words.

Finite automata on \( \omega \)-words have by now a vast literature, see [20] for a comprehensive treatment. Finite automata acting on well-ordered words longer than \( \omega \) have been investigated in [2,9,10,22,23], to mention a few references. In the last decade, the theory of automata on well-ordered words has been extended to automata on all countable words, including scattered and dense words. In [3,5,8], both operational and logical characterizations of the class of languages of countable words recognized by finite automata were obtained.

Context-free grammars generating \( \omega \)-words were introduced in [11] and subsequently studied in [19]. Context-free grammars generating arbitrary countable words were defined in [13,14]. 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 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 [13] 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 \( \omega^n \).

A Kleene-type characterization of BCFLs of well-ordered and scattered words was given in [16]. Here we provide a Kleene-type characterization of MCFLs of well-ordered and scattered words. Before presenting the necessary preliminaries in detail, we give a formulation of our main result, at least in the well-ordered case.
Suppose that $\Sigma$ is an alphabet, and let $\Sigma^\#$ denote the set of all (countable) words over $\Sigma$. Let $P(\Sigma^\#)$ be the set of all subsets of $\Sigma^\#$. The set of $\mu\omega T_w$-expressions over $\Sigma$ is defined by the following grammar:

$$T ::= a \mid \varepsilon \mid x \mid T + T \mid T \cdot T \mid \mu x.T \mid T^\omega$$

Here, each letter $a \in \Sigma$ denotes the language containing $a$ as its unique word, while $\varepsilon$ denotes the language containing only the empty word. The symbols $+$ and $\cdot$ are interpreted as set union and concatenation over $P(\Sigma^\#)$, and the variables $x$ range over languages in $\Sigma^\#$. The $\mu$-operator corresponds to taking least fixed points. Finally, $^\omega$ is interpreted as the $\omega$-power operation over $P(\Sigma^\#)$: $L \mapsto L \cdot L \cdot \cdots$. An expression is closed if each variable occurs in the scope of a least fixed-point operator. Each closed expression denotes a language in $P(\Sigma^\#)$.

Our main result in the well-ordered case, which is a corollary of Theorem 2, is:

**Theorem 1.** A language $L \subseteq \Sigma^\#$ is an MCFL of well-ordered words iff it is denoted by some closed $\mu\omega T_w$-expression.

**Example 1.** The expression $\mu x.(x^\omega + a + b + \varepsilon)$ denotes the set of all well-ordered words over the alphabet $\{a, b\}$.

It was shown in [16] that the syntactic fragment of the above expressions, with the $\omega$-power operation restricted to closed expressions, characterizes the BCFLs of well-ordered words. A similar, but more involved result holds for MCFLs of scattered words, cf. Theorem [2]. Both theorems were conjectured by the authors of [16].

## 2 Notation

### 2.1 Linear orderings

A *linear ordering* is a pair $(I, <)$, where $I$ is a set and $<$ is an irreflexive transitive trichotomous relation (i.e. a strict total ordering) on $I$. If $I$ is finite or countable, we say that the ordering is finite or countable as well. In this paper, all orderings are assumed to be countable. A good reference for linear orderings is [21].

An *embedding* of the linear ordering $(I, <)$ into $(J, \prec)$ is an order preserving function $f : I \to J$, i.e. $x < y$ implies $f(x) \prec f(y)$ for each $x, y \in I$. If $f$ is surjective, we call it an *isomorphism*. Two linear orderings are said to be *isomorphic* if there exists an isomorphism between them. Isomorphism between linear orderings is an equivalence relation; classes of this equivalence relation are called *order types*. If $I \subseteq J$ and $<$ is the restriction of $\prec$ onto $I$, then we say that $(I, <)$ is a *sub-ordering* of $(J, \prec)$.

Examples of linear orderings are the ordering $(\mathbb{N}, <)$ of the positive integers, the ordering $(\mathbb{N}_-, <)$ of the negative integers, the ordering $(\mathbb{Z}, <)$ of the integers
and the ordering \((\mathbb{Q}, <)\) of the rationals. The respective order types are denoted \(\omega, -\omega, \zeta\) and \(\eta\). In order to ease notation, we write simply \(I\) for \((I, <)\) if the ordering \(<\) is standard or known from the context.

An ordering is \textit{scattered} if it does not have a sub-ordering of order type \(\eta\), otherwise it is \textit{quasi-dense}. An ordering is a \textit{well-ordering} if it does not have a sub-ordering of order type \(-\omega\). Order types of well-orderings are called \textit{ordinals}.

When \((I, <)\) is an ordering and for each \(i \in I\), \((J_i, <_i)\) is an ordering, then the \textit{generalized sum} \(\sum_{i \in I} (J_i, <_i)\) is the disjoint union \(\{(i, j) : i \in I, j \in J_i\}\) equipped with the lexicographic ordering \((i, j) < (i', j')\) iff \(i < i'\), or \(i = i'\) and \(j <_i j'\). It is known that if \((I, <)\) and the \((J_i, <_i)\) are scattered or well-ordered, then so is the generalized sum. The operation of generalized sum can be extended to order types since it preserves isomorphisms. For example, \(\zeta = -\omega + \omega\). Ordinals are also equipped with an exponentiation operator.

Hausdorff classified linear orderings into an infinite hierarchy. Following [17], we present a variant of this hierarchy. Let \(VD_0\) be the collection of all finite linear orderings, and when \(\alpha\) is some ordinal, let \(VD_\alpha\) be the collection of all finite sums of linear orderings of the form \(\sum_{i \in \mathbb{Z}} (I_i, <_i)\), where for each integer \(i \in \mathbb{Z}\), \((I_i, <_i)\) is a member of \(VD_\alpha\), for some ordinal \(\alpha_i < \alpha\). According to a theorem of Hausdorff (see e.g. [21], Thm. 5.24), a (countable) linear ordering \((I, <)\) is scattered if and only if it belongs to \(VD_\alpha\) for some (countable) ordinal \(\alpha\); the least such \(\alpha\) is called the \textit{rank} of \((I, <)\), denoted \(\text{rank}(I, <)\).

### 2.2 Words, tree domains, trees

An \textit{alphabet} is a finite nonempty set \(\Sigma\) of symbols, usually called \textit{letters}. A \textit{word} over \(\Sigma\) is a linear ordering \((I, <)\) equipped with a \textit{labeling function} \(\lambda : I \rightarrow \Sigma\). An \textit{embedding of words} is a mapping preserving the order and the labeling; a surjective embedding is an \textit{isomorphism}. Order theoretic properties of the underlying linear ordering of a word are transferred to the word. A word is finite if its underlying linear order is finite, and an \(\omega\)-word, if its underlying linear order is a well-order of order type \(\omega\). We usually identify isomorphic words and denote by \(\Sigma^2\) the set of all words over \(\Sigma\). As usual, we denote the collection of finite and \(\omega\)-words over \(\Sigma\) by \(\Sigma^*\) and \(\Sigma^\omega\), respectively. The length of a word \(u \in \Sigma^*\) is denoted \(|u|\). A language over \(\Sigma\) is a subset of \(\Sigma^2\). As in the introduction, we let \(P(\Sigma^2)\) denote the collection of all languages over \(\Sigma\).

When \((I, <)\) is a linear ordering and \(w_i = (J_i, <_i, \lambda_i)\) for \(i \in I\) are words, then we define their \textit{concatenation} \(\prod_{i \in I} w_i\) as the word with underlying linear order \(\sum_{i \in I} (J_i, <_i)\) and labeling \(\lambda(i, j) = \lambda_i(j)\). When \(I\) has two elements, we obtain the usual notion of concatenation, denoted \(u \cdot v\), or just \(uv\). The operation of concatenation is extended to languages in \(P(\Sigma^2)\): \(\prod_{i \in I} L_i = \{\prod_{i \in I} w_i : w_i \in L_i\}\)
Let \( L_1 \}. When \( L, L_1, L_2 \subseteq \Sigma^\omega \), then we define \( L_1 + L_2 \) to be the set union and \( L_1 L_2 = \{ uv : u \in L_1, v \in L_2 \} \). Moreover, we define \( L^n = \prod_{i \in \mathbb{N}} L \).

The set \( P(\Sigma^\omega) \) of languages over \( \Sigma \), equipped with the inclusion order, is a complete lattice. When \( A \) is a set, a function \( f : P(A)^n \to P(A) \) is monotone if \( A_i \subseteq A_i' \) for each \( i \in [n] \) implies \( f(A_1, \ldots, A_n) \subseteq f(A_1', \ldots, A_n') \). The following fact is clear.

**Lemma 1.** The functions \(+, \cdot : P(\Sigma^\omega)^2 \to P(\Sigma^\omega)\) and \(\omega : P(\Sigma^\omega) \to P(\Sigma^\omega)\) are monotone.

We will also consider pairs of words over an alphabet \( \Sigma \), equipped with a finite concatenation and an \( \omega \)-product operation. For pairs \((u, v), (u', v') \) in \( \Sigma^\omega \times \Sigma^\omega \), we define the product \((u, v) \cdot (u', v')\) to be the pair \((uv, v'v)\), and when for each \( i \in \mathbb{N} \), \((u_i, v_i)\) is in \( \Sigma^\omega \times \Sigma^\omega \), then we let \( \prod_{i \in \mathbb{N}} (u_i, v_i) \) be the word \( (\prod_{i \in \mathbb{N}} u_i) (\prod_{i \in \mathbb{N}} v_i) \).

Let \( P(\Sigma^\omega \times \Sigma^\omega) \) denote the set of all subsets of \( \Sigma^\omega \times \Sigma^\omega \). Then \( P(\Sigma^\omega \times \Sigma^\omega) \) is naturally equipped with the operations of set union \( L + L' \), concatenation \( L \cdot L' = \{(u, v) \cdot (u', v') : (u, v) \in L, (u', v') \in L'\} \) and Kleene star \( L^* = \{\varepsilon\} \cup L \cup L^2 \cup \cdots \). We also define an \( \omega \)-power operation \( P(\Sigma^\omega \times \Sigma^\omega) \to P(\Sigma^\omega) \) by \( L^\omega = \{\prod_{i \in \mathbb{N}} (u_i, v_i) : (u_i, v_i) \in L\} \). When \( L_1, L_2 \subseteq \Sigma^\omega \), let \( L_1 \times L_2 = \{(u, v) : u \in L_1, v \in L_2 \} \subseteq \Sigma^\omega \times \Sigma^\omega \).

**Lemma 2.** The functions

\[
\begin{align*}
\times & : P(\Sigma^\omega)^2 \to P(\Sigma^\omega \times \Sigma^\omega) \\
+ & : P(\Sigma^\omega \times \Sigma^\omega)^2 \to P(\Sigma^\omega \times \Sigma^\omega) \\
* & : P(\Sigma^\omega \times \Sigma^\omega)^2 \to P(\Sigma^\omega \times \Sigma^\omega) \\
\omega & : P(\Sigma^\omega \times \Sigma^\omega) \to P(\Sigma^\omega)
\end{align*}
\]

are monotone.

We will use Lemma 1 and Lemma 2 in the following context. Suppose that for each \( i \in [n] = \{1, \ldots, n\} \), \( f_i : P(\Sigma^\omega)^{n+p} \to P(\Sigma^\omega) \) is a function that can be constructed by function composition from the above functions, the projection functions and constant functions. Let \( f = (f_1, \ldots, f_n) : P(\Sigma^\omega)^{n+p} \to P(\Sigma^\omega)^n \) be the target tupling of the \( f_i \). Then \( f \) is a monotone function, and by Tarski’s fixed point theorem, for each \( y \in P(\Sigma^\omega)^p \) there is a least solution of the fixed point equation \( x = f(x, y) \) in the variable \( x \) ranging over \( P(\Sigma^\omega)^n \). This least fixed point, denoted \( \mu x. f(x, y) \), gives rise to a function \( P(\Sigma^\omega)^p \to P(\Sigma^\omega)^p \) in the parameter \( y \). It is known that this function is also monotone, see e.g. [6].

A tree domain is a prefix closed nonempty (but possibly infinite) subset of \( \mathbb{N}^* \). Elements of a tree domain \( T \) are also called nodes of \( T \). When \( x \) and \( x \cdot i \) are nodes of \( T \) for \( x \in \mathbb{N}^* \) and \( i \in \mathbb{N} \), then \( x \cdot i \) is a child of \( x \). A descendant of a node
2.3 Muller context-free languages of scattered words

A Muller context-free grammar, or MCFG for short, is a system \( G = (V, \Sigma, R, S, \mathcal{F}) \), where \( V \) is the alphabet of nonterminals, \( \Sigma \) is the alphabet of terminals, \( \Sigma \cap V = \emptyset \), \( R \) is the finite set of productions of the form \( A \rightarrow \alpha \) with \( A \in V \) and \( \alpha \in (\Sigma \cup V)^* \), \( S \in V \) is the start symbol and \( \mathcal{F} \subseteq P(V) \) is the set of nonempty accepting sets.

A derivation tree of the above grammar \( G \) is a tree \( t : \text{dom}(t) \rightarrow V \cup \Sigma \cup \{\varepsilon\} \) satisfying the following conditions:

1. For each inner node \( x \) of \( t \) there exists a rule \( X \rightarrow X_1 \ldots X_n \) in \( R \) such that \( t(x) = X \), the children of \( x \) are exactly \( x \cdot 1, \ldots, x \cdot n \), and for each \( i \in [n] \), \( t(x \cdot i) = X_i \) so that when \( n = 0 \), \( x \) has a single child \( x \cdot 1 \) labeled \( \varepsilon \);

2. For each infinite path \( \pi \) of \( t \), infLabels(\( \pi \)) is an accepting set of \( G \).

A derivation tree is complete if its leaves are all labeled in \( \Sigma \cup \{\varepsilon\} \). If \( t \) is a derivation tree having root symbol \( t(\varepsilon) = A \), then we say that \( t \) is an \( A \)-tree.
The language \( L(G, A) \subseteq \Sigma^\dagger \) generated from \( A \in V \) is the set of frontier words of complete \( A \)-trees. The language \( L(G) \) generated by \( G \) is \( L(G, S) \). An MCFL is a language generated by some MCFG.

**Example 2.** If \( G = (\{S, I\}, \{a, b\}, R, S, \{\{I\}\}) \), with

\[
R = \{ S \to a, S \to b, S \to \varepsilon, S \to I, I \to SI \},
\]

then \( L(G) \) consists of all the well-ordered words over \( \{a, b\} \).

**Example 3.** If \( G = (\{S, I\}, \{a, b\}, R, S, \{\{I\}\}) \), with

\[
R = \{ S \to a, S \to b, S \to \varepsilon, S \to I, I \to SIS \},
\]

then \( L(G) \) consists of all the scattered words over \( \{a, b\} \).

Let \( L \subseteq \Sigma^\dagger \) be an MCFL consisting of scattered words only and \( G = (V, \Sigma, R, S, F) \) an MCFG with \( L(G) = L \). We may assume that \( G \) is in normal form \([14]\) – among the properties of this normal form we will use the following ones (see \([14]\), Prop. 14) frequently:

- For every derivation tree there is a locally finite derivation tree with the same root symbol and same labeled frontier.
- The frontier of each derivation tree is scattered.

In the rest of the paper, we fix an MCFG \( G = (V, \Sigma, R, S, F) \) in normal form generating only scattered words.

When \( t \) is a derivation tree, then we define \( \text{rank}(t) = \text{rank} (\text{fr}(t)) \). For a derivation tree \( t \), let \( \text{maxNodes}(t) \) be the prefix of \( \text{dom}(t) \) consisting of the nodes having maximal rank, i.e. \( \text{maxNodes}(t) = \{ x \in \text{dom}(t) : \text{rank}(t|_x) = \text{rank}(t) \} \). Suppose that \( t \) is locally finite. It is known, (see e.g. \([15]\), proof of Proposition 1, paragraph 4) that in this case \( \text{maxNodes}(t) \) is the union of finitely many maximal paths. Clearly, the set \( \{ \pi_1, \ldots, \pi_n \} \) of these paths is unique. Let \( \text{level}(t) \) stand for the above \( n \), the number of maximal paths covering \( \text{maxNodes}(t) \). Also, let \( \text{branch}(t) \) stand for the longest common prefix of the paths \( \pi_1, \ldots, \pi_n \) (which is a finite word if \( \text{level}(t) > 1 \) and is \( \pi_1 \) if \( \text{level}(t) = 1 \)).

We say that a (not necessarily locally finite) derivation tree \( t \) is simple if \( \text{maxNodes}(t) \) contains a single infinite path \( \pi \) and if \( \text{infLabels}(\pi) = \text{labels}(\pi) \), i.e. \( \text{head}(\pi) = \varepsilon \). (When \( t \) is additionally locally finite, then this path \( \pi \) contains all nodes of \( \text{maxNodes}(t) \).) Such a path is called the central path of \( t \). If \( t \) is a simple \( A \)-tree and \( F \) is the set of labels of its central path, then we call \( t \) an \( F \)-simple \( A \)-tree.

### 3 The main result

For locally finite complete derivation trees \( t' \) and \( t \), let \( t' \prec t \) if one of the following conditions holds:

6
1. rank(t′) < rank(t);
2. rank(t′) = rank(t) and level(t′) < level(t);
3. rank(t′) = rank(t), level(t′) = level(t) > 1 and \(|branch(t′)| < |branch(t)|\).
4. rank(t′) = rank(t), level(t′) = level(t) = 1, that is, the set of nodes of maximal rank is a path \( \pi \) in \( t \) and a path \( \pi' \) in \( t' \). Then let \( t' \succ t \) iff 
\[ |head(\pi')| < |head(\pi)|. \]

**Lemma 3.** The relation \( \prec \) is a well-partial order (wpo) of locally finite complete derivation trees. The minimal elements of this wpo are the one-node trees corresponding to the elements of \( \Sigma \cup \{\varepsilon\} \). Suppose that \( t \) is a locally finite complete derivation tree and \( t' = t_x \) is a proper subtree of \( t \), so that \( x \neq \varepsilon \). If \( t \) is not simple, or if \( t \) is simple but \( x \) does not belong to the central path of \( t \), then \( t' \prec t \).

**Proof.** It is clear that \( \prec \) is irreflexive. To prove that it is transitive, suppose that \( t'' \prec t' \) and \( t' \prec t \). If rank(\( t'' \)) < rank(\( t' \)), then clearly \( t'' \prec t \). Suppose that rank(\( t'' \)) = rank(\( t' \)). Then also rank(\( t'' \)) = rank(\( t' \)) = rank(\( t \)). If level(\( t'' \)) < level(\( t \)) then \( t'' \prec t \) again. Thus, we may suppose that level(\( t'' \)) = level(\( t \)), so that level(\( t'' \)) = level(\( t' \)) = level(\( t \)) = \( n \). Now there are two cases. If \( n > 1 \), then, since \( t'' \prec t' \) and \( t' \prec t \), we know that \(|branch(t'')| < |branch(t')| < |branch(t)|\) and thus \( t'' \prec t \). If \( n = 1 \), then the maximal nodes form a single maximal path in each of the trees \( t'', t' \) and \( t \). Let us denote these paths by \( \pi'', \pi' \) and \( \pi \), respectively. As \( t'' \prec t' \) and \( t' \prec t \), we have that \(|head(\pi'')| < |head(\pi')| < |head(\pi)|\), so that \( t'' \prec t \) again.

The fact that there is no infinite decreasing sequence of locally finite complete derivation trees with respect to the relation \( \prec \) is clear, since every set of ordinals is well-ordered.

Suppose now that \( t \) is a locally finite complete derivation tree which has at least two nodes. By assumption, \( t \) has a leaf node \( x \). Let \( t' = t|_x \). If rank(\( t' \)) < rank(\( t \)) then \( t' \prec t \). Otherwise, rank(\( t' \)) = rank(\( t \)) = 0 and \( t \) is necessarily finite (since the frontier of an infinite complete derivation tree is infinite). Clearly, maxNodes(\( t \)) is the set of all nodes of \( t \), and either level(\( t' \)) = 1 < level(\( t \)), or level(\( t' \)) = level(\( t \)) = 1. In the latter case, \( t \) has a single maximal path \( \pi \), and \(|head(\pi)| = 0 < |head(\pi)|\) for the single maximal path \( \pi' \) of \( t' \). In either case, \( t' \prec t \). Thus, no locally finite complete derivation tree having more than one node is minimal. On the other hand, all one-node complete derivation trees corresponding to the elements of \( \Sigma \cup \{\varepsilon\} \) are clearly minimal (and locally finite).

To prove the last claim, suppose that \( t \) is a locally finite complete derivation tree and \( t' = t|_x \). If rank(\( t' \)) < rank(\( t \)), we are done. Otherwise, rank(\( t' \)) = rank(\( t \)) and \( x \) is a member of maxNodes(\( t \)). Thus, if \( \pi \) is a maximal path of maxNodes(\( t \)), then \( x\pi \) is a maximal path of maxNodes(\( t \)). Hence level(\( t' \)) \leq level(\( t \)). If level(\( t' \)) < level(\( t \)), we are done. Otherwise, level(\( t' \)) = level(\( t \)) and maxNodes(\( t \)) = xmaxNodes(\( t' \)).

Now there are two cases.
1. If \( \text{level}(t) > 1 \), then \( \text{branch}(t) = x \text{branch}(t') \), thus \( |\text{branch}(t')| < |\text{branch}(t)| \) and \( t' < t \).

2. Suppose that \( \text{level}(t) = 1 \), and let \( \pi \) denote the unique maximal path of \( t \) whose nodes form the set \( \text{maxNodes}(t) \). Since \( \text{rank}(t') = \text{rank}(t) \), we have that \( x \) belongs to \( \pi \) and, by assumption, \( t \) is not simple. Since \( t \) is not simple and has at least two nodes, \( \text{head}(\pi) \neq \varepsilon \) and \( |\text{head}(\pi')| < |\text{head}(\pi)| \), where \( \pi' \) is the unique maximal path of \( t' \) whose nodes form the set \( \text{maxNodes}(t') \).

(Actually \( \pi' \) is determined by the proper suffix \( \pi|_{x} \) of \( \pi \).) \( \Box \)

Now we define certain ordinary \( \omega \)-regular languages \cite{IS20} corresponding to central paths of simple derivation trees. Let \( \Gamma \) stand for the (finite) set consisting of those triplets

\[
(\alpha, B, \beta) \in (V \cup \Sigma)^* \times V \times (V \cup \Sigma)^*
\]

for which \( \alpha B \beta \) occurs as the right-hand side of a production of \( G \). For any nonterminal \( A \in V \) and accepting set \( F \in \mathcal{F} \), let \( R_{A,F} \subseteq \Gamma^\omega \) stand for the set of \( \omega \)-words over \( \Gamma \) accepted by the deterministic (partial) Muller (word) automaton \((F, \Gamma, \delta, A, \{F\})\), with \( B = \delta(C, (\alpha, D, \beta)) \) if and only if \( D = B \) and \( C \rightarrow \alpha B \beta \) is a production of \( G \). By definition, each \( R_{A,F} \) is an \( \omega \)-regular set which can be built from singleton sets corresponding to the elements of \( \Gamma \) by the usual regular operations and the \( \omega \)-power operation (actually, since every state has to be visited infinitely many times, \( R_{A,F} \) can be written as the \( \omega \)-power of a regular language of finite words over \( \Gamma \)).

Members of \( R_{A,F} \) correspond to central paths of \( F \)-simple \( A \)-trees in the following sense. Given \( w = (\alpha_1, A_1, \beta_1)(\alpha_2, A_2, \beta_2)\ldots \in R_{A,F} \), we define an \( F \)-simple \( A \)-tree \( t_w \) of \( G \) as follows. The nodes \( x_0, x_1, \ldots \) of the central path of \( t_w \) are \( x_0 = \varepsilon \), and \( x_i = x_{i-1} \cdot (|\alpha_i| + 1) \), for \( i > 0 \). Each \( x_i \) has \( |\alpha_{i+1} A_{i+1} \beta_{i+1}| \) children, respectively labeled by the letters of the word \( \alpha_{i+1} A_{i+1} \beta_{i+1} \). Nodes not on the central path of \( t_w \) are leaf nodes.

It is straightforward to see the following claims:

1. For each \( w \in R_{A,F} \), \( t_w \) is an \( F \)-simple \( A \)-tree.
2. Every \( F \)-simple \( A \)-tree has a prefix of the form \( t_w \), for some \( w \in R_{A,F} \). Thus, every such tree can be constructed by choosing an appropriate \( w \in R_{A,F} \), and substituting a derivation tree \( t_x \) with root symbol \( t_w(x) \) for each leaf \( x \) of \( t_w \).

Moreover, it is clear that when \( w = (\alpha_1, A_1, \beta_1)(\alpha_2, A_2, \beta_2)\ldots \), then \( \text{lfr}(t_w) \) is \((\prod_{i \in \mathbb{N}} \alpha_i) \cdot (\prod_{i \in \mathbb{N}} \beta_i)\).

Let us assign a variable \( X_A \) to each \( A \in V \), and let \( \mathcal{X} \) be the set of all variables.

For each ordinary regular expression \( r \) over \( \Gamma \), we define an expression (term) \( \overline{\gamma} \) over \( \Sigma \cup \mathcal{X} \) involving the function symbols \( \times, +, \cdot \). To this end, when \( \alpha \) is a word in \((\Sigma \cup V)^*\), let \( \overline{\alpha} \) be the word in \((\mathcal{X} \cup \Sigma)^* \) obtained by replacing each occurrence of a nonterminal \( A \) by the variable \( X_A \). Then, for a letter \( \gamma = (\alpha, A, \beta) \in \Gamma \), define \( \overline{\gamma} = \overline{\alpha} \times \overline{\beta} \). To obtain \( \overline{\gamma} \), we replace each occurrence of a letter \( \gamma \) in \( r \) by \( \overline{\gamma} \).
When $A$ is a nonterminal and $A \in F$ for some $F \in \mathcal{F}$, consider an ordinary regular expression $r_{A,F}$ over $\Gamma$ such that $r_{A,F}^\omega$ denotes the set $R_{A,F}$ (defined above) of all $\omega$-words corresponding to central paths of $F$-simple $A$-trees. Then consider the following system of equations $E_G$ associated with $G$ in the variables $\mathcal{X}$:

$$X_A = \sum_{A \rightarrow w \in R} \tau + \sum_{A \in F \in \mathcal{F}} (r_{A,F})^\omega.$$  

**Example 4.** The system of equations $E_G$ associated with the grammar in Example 3 is:

$$X_S = a + b + \varepsilon + X_I$$  

$$X_I = (X_S \times X_S)^\omega$$

As usual, we can associate a function $f_G : P(\Sigma^\ast)^X \rightarrow P(\Sigma^\ast)^X$ with $E_G$. By Lemmas 1 and 2 and using the facts that the projections are monotone and that monotone functions are closed under function composition, we have that $f_G$ is monotone. Thus, $f_G$ has a least fixed point.

**Proposition 1.** For each $A \in V$, the corresponding component of the least fixed point solution of the system $E_G$ is the language $L(G, A)$ of all words derivable from $A$.

**Proof.** The fact that the languages $L(G, A)$, $A \in V$, form a solution is clear from the definition of $E_G$. Let us also define $L(G, a) = \{a\}$, for each $a \in \Sigma \cup \{\varepsilon\}$. Suppose that the family of languages $L_A$, $A \in V$ is another solution, and let $L_a = \{a\}$ for $a \in \Sigma \cup \{\varepsilon\}$. We want to show that if $t$ is a locally finite complete $A$-tree with $lfr(t) = u$, then $u \in L_A$, for each $A \in \Sigma \cup \{\varepsilon\} \cup V$. We apply well-founded induction with respect to the wpo $\prec$.

For the base case, if $t$ consists of a single node, then $A = a \in \Sigma \cup \{\varepsilon\}$, $u = a$, and our claim is clear. Otherwise, there are two cases: either $t$ is a simple tree, or not.

If $t = A(t_1, \ldots, t_n)$ is not simple, then we have $t_i \prec t$ for each $i \in [n]$ by Lemma 3. Let $A_i$ be the root symbol of $t_i$ and $u_i$ the labeled frontier word of $t_i$ for each $i$. By the induction hypothesis, each $u_i$ is a member of $L_{A_i}$. Since $t$ is a derivation tree, $A \rightarrow A_1 \ldots A_n$ is a production of $G$. Thus, by the construction of $E_G$, $u_i = u_1 \ldots u_n \in L_A$.

Otherwise, if $t$ is an $F$-simple $A$-tree for some $F \in \mathcal{F}$ and $A \in V$, then $t$ can be constructed from a tree $t_w$ with $w \in R_{A,F}$ by replacing each leaf node $x$ of $t_w$ by some complete derivation tree $t_x$ with root symbol $t_w(x)$. Since such leaves are not on the central path of $t$, we have $t_x \prec t$ for each $x$, again by Lemma 3. Applying the induction hypothesis, we get that the labeled frontier word $u_x$ of
each $t_x$ is a member of $L_{t_u(x)}$. Thus, by the construction of $E_G$, $u$ is a member of $L_A$. \hfill \Box

It is well-known, cf. [41] or [3], Chapter 8, Theorem 2.15 and Chapter 6, Section 8.1, Equation (3.2), that when $L, L', L''$ are complete lattices and $f : L \times L' \times L'' \rightarrow L$ and $g : L \times L' \times L'' \rightarrow L'$ are monotone functions, then the least solution (in the parameter $z$) of the system of equations

\[
\begin{align*}
  x &= f(x, y, z) \\
y &= g(x, y, z)
\end{align*}
\]

can be obtained by Gaussian elimination as

\[
\begin{align*}
x &= \mu x.f(x, \mu y.g(x, y, z), z) \\
y &= \mu y.g(\mu x.f(x, \mu y.g(x, y, z), y), z, y, z)
\end{align*}
\]

Using this fact and Proposition [11] we obtain our final result.

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

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

Here, $a \in \Sigma$ and $x \in X$ for an infinite countable set of variables. An occurrence of a variable is free if it is not in the scope of a $\mu$-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 $P(\Sigma^\flat)$ and $P(\Sigma^\flat \times \Sigma^\flat)$ introduced earlier. When the free variables of an expression form the set $Y$, then an expression denotes a language in $P((\Sigma \cup Y)^\sharp)$.

**Remark 1.** Actually, $\varepsilon$ is redundant, as it is expressible by $((\mu x.x \times \mu x.x)^\omega)$. We do not need a constant 0 denoting the empty set of pairs since it is expressible by $(\mu x.x) \times (\mu x.x)$.

**Theorem 2.** A language $L \subseteq \Sigma^\sharp$ is an MCFL of scattered words if and only if it can be denoted by a closed $\mu \omega T_s$-expression.

**Proof.** It is easy to show that each expression denotes an MCFL of scattered words. One uses the following facts, where $\Delta$ denotes an alphabet and $x, \# \not\in \Delta$.

1. The set of MCFLs (of scattered words) over $\Delta$ is closed under $+$ and $\cdot$.
2. If $L, L' \subseteq \Delta^\sharp$ are MCFLs (of scattered words), then $L \# L' \subseteq (\Delta \cup \{\#\})^\sharp$ is an MCFL (of scattered words).
3. Suppose that $L, L' \subseteq \Delta^\sharp \# \Delta^\sharp$ are MCFLs (of scattered words). Then

\[
\{uv\#v'u' : u\#u' \in L, v\#v' \in L'\} \subseteq \Delta^\sharp \# \Delta^\sharp
\]

is an MCFL (of scattered words).
4. Suppose that $L \subseteq \Delta^\#\Delta^\#$ is an MCFL (of scattered words). Then

$$\{u_1 \ldots u_n \# v_n \ldots v_1 : n \geq 0, \ u_i \# v_i \in L\} \subseteq \Delta^\#\Delta^\#$$

is an MCFL (of scattered words).

5. Suppose that $L \subseteq \Delta^\#\Delta^\#$ is an MCFL (of scattered words). Then

$$\{(u_1 u_2 \ldots) (\ldots v_2 v_1) : u_i \# v_i \in L\} \subseteq \Delta^\#$$

is an MCFL (of scattered words).

6. Suppose that $L \subseteq (\Delta \cup \{x\})^\#$ is an MCFL (of scattered words). Then, with respect to set inclusion, there is a least language $L' \subseteq \Delta^\#\Delta^\#$ such that $L[x \mapsto L'] = L'$, and this language $L'$ is an MCFL (of scattered words). (Here, $L[x \mapsto L']$ is the language obtained from $L$ by ‘substituting’ $L'$ for $x$.)

It is known (see [14]) that the class of MCFLs is (effectively) closed under substitution and that every context-free language of finite words (in particular, $\{a, b\}$, $\{ab\}$ or $\{a\#b\}$) is an MCFL, showing Items 1–3 above.

For Items 4 and 5, let $G = (V, \Delta \cup \{\#\}, R, S, F)$ be an MCFG generating the MCFL $L \subseteq \Delta^\#\Delta^\#$. Then

$$G_1 = (V \cup \{\#\}, \Delta \cup \{\#'\}, R \cup \{\# \mapsto \#', \# \mapsto S\}, \#, F)$$

generates the MCFL $L_1 = \{u_1 \ldots u_n \#' v_n \ldots v_1 : n \geq 0, u_i \# v_i \in L\}$, showing Item 4 (applying the substitution $\#' \mapsto \{\#\}$) and

$$G_2 = (V \cup \{\#\}, \Delta, R \cup \{\# \mapsto S\}, \#, F \cup \{H \cup \{\#\} : H \subseteq V\})$$

generates the MCFL defined in Item 5.

Finally, let $G = (V, \Delta \cup \{x\}, R, S, F)$ be an MCFG generating $L \subseteq (\Delta \cup \{x\})^\#$. Then

$$G_3 = (V \cup \{x\}, \Delta, R \cup \{x \mapsto S\}, x, F)$$

generates the language $L'$ of Item 6.

The other direction follows from Proposition [1] \[ \square \]

Example 5. The expression $\mu x.((x \times x)^\omega + a + b + \epsilon)$ denotes the set of all scattered words over the alphabet $\{a, b\}$.

Example 6. Let $L \subseteq \{a, b\}^\#$ be the language of all words $w$ such that the word obtained from $w$ by removing all occurrences of letter $b$ is well-ordered, as is the ‘mirror image’ of the word obtained by removing all occurrences of letter $a$. It is not difficult to show that each word in $L$ contains only a finite number of ‘alternations’ between $a$ and $b$. Using this fact, an MCFG generating $L$ is:
\( G = (\{S, A, B, I, J\}, \Sigma, R, S, \{{\{I\}}, \{{J\}}\}) \) with \( R \) consisting of the productions

\[
\begin{align*}
S & \rightarrow AS \mid BS \mid \varepsilon \\
A & \rightarrow a \mid \varepsilon \mid I \\
I & \rightarrow AI \\
B & \rightarrow b \mid \varepsilon \mid J \\
J & \rightarrow JB
\end{align*}
\]

Using the algorithm described above (with some simplification), an expression for \( L \) is:

\[
\begin{align*}
t_S &= \mu x_S.((t_A + t_B)x_S + \varepsilon) \\
\end{align*}
\]

with

\[
\begin{align*}
t_A &= \mu x_A.(a + \varepsilon + (x_A \times \varepsilon)^\omega) \\
t_B &= \mu x_B.(b + \varepsilon + (\varepsilon \times x_B)^\omega).
\end{align*}
\]

We restate Theorem 1 and show that it is a corollary of Theorem 2.

**Theorem.** A language \( L \subseteq \Sigma^* \) is an MCFL of well-ordered words iff it is denoted by some closed \( \mu \omega T_w \)-expression.

**Proof.** Recall that the set of \( \mu \omega T_w \)-expressions over an alphabet \( \Sigma \) is defined by the grammar

\[
T ::= a \mid \varepsilon \mid x \mid T + T \mid T \cdot T \mid \mu x.T \mid T^\omega
\]

where \( a \in \Sigma \) and \( x \) ranges over the set \( X \) of variables, moreover, an expression \( t \) is closed if each occurrence of a variable \( x \) in \( t \) is within the scope of some prefix \( \mu x \). Below we will sometimes view the construct \( t^\omega \) as a shorthand for \( (t \times \varepsilon)^\omega \).

For one direction, we show by structural induction that for a \( \mu \omega T_w \)-expression \( t \) with free variables in \( X \), the language \(|t| \subseteq (\Sigma \cup X)^2\) denoted by \( t \) consists of well-ordered words. For the base cases, i.e. when \( t = a, t = \varepsilon \) or \( t = x \), the claim clearly holds. If \( t = t_1 + t_2 \) or \( t = t_1 \cdot t_2 \), or \( t = t_1^\omega \), for some expressions \( t_1, t_2 \), our claim is again clear (using the fact that every well-ordered product of well-ordered words is well-ordered in the last two cases). Finally, if \( t = \mu x.t_1 \), where \( t_1 \) denotes an MCFL \( L \subseteq (\Sigma \cup X)^2 \), \(|t| \) is the language \( \bigcup L_\alpha \), where \( L_0 = \emptyset \) and for each \( \alpha > 0 \),

\[
L_\alpha = L_{<\alpha} \cup L[x \mapsto L_{<\alpha}]
\]

where \( L_{<\alpha} = \left( \bigcup_{\beta < \alpha} L_\beta \right) \). Thus, if \( L \) contains only well-ordered words then so does each \( L_\alpha \), since languages of well-ordered words are closed under substitution.

12
For the other direction, we may restrict ourselves to expressions (of type $T$ or $P$) which do not have any subexpression denoting the empty set, nor any subexpression other than $\varepsilon$ denoting $\{\varepsilon\}$.

Suppose that $t$ and $p$ are such expressions of type $T$ and $P$, respectively. It is not difficult to prove the following claim by (simultaneous) structural induction:

**Claim A.** If $t$ has a subexpression (belonging to the syntactic category $P$) of the form $t_1 \times t_2$ with $t_2 \neq \varepsilon$, then $|t|$ contains a word which is not well-ordered. If $p$ has a subexpression $p' = t_1 \times t_2$ with $t_2 \neq \varepsilon$, then $|p|$ contains a pair $(u, v)$ such that either $v \neq \varepsilon$ or one of $u, v$ is not well-ordered.

To prove this, first note that $t$ cannot have the form $a, \varepsilon$ or $x$. When $p = t_1 \times t_2$, for some $t_1, t_2$, $t_2 \neq \varepsilon$, then our claim clearly holds for $p$, since either one of $|t_1|$ and $|t_2|$ contains a word which is not well-ordered, or $|t_2|$ contains a nonempty word. The induction step is clear when $p = p_1 + p_2$, $p = p_1 \cdot p_2$, $p = p_1^*$, or when $t = t_1 + t_2$, $t = t_1 \cdot t_2$, or $t = p_1^*$. When $t = \mu x. t_1$, then $t_1$ contains a word $u$ which is not well-ordered. Since by assumption $|t|$ contains a nonempty word $v$, $t$ contains $u[x \mapsto v]$, which is not well-ordered.

To complete the proof, note that if each subexpression of $t$ of the form $t_1 \times t_2$ satisfies $t_2 = \varepsilon$, then we can transform $t$ into an equivalent $\mu \omega T_w$ expression by repeatedly replacing subexpressions of the form $t_1 \times \varepsilon$ with $t_1$ and subexpressions of the form $t_1^*$ with $\mu x.(t_1 x + \varepsilon)$. $\square$

Using Claim A, we may develop a low-degree polynomial-time algorithm for the following decision problem: given a closed $\mu \omega T_x$-expression $t$ of syntactic category $T$, does the language denoted by $t$ consist of well-ordered words only? The expression $t$ may be assumed to be given as an expression tree.

In the following, $t_1, t_2$ denote expressions belonging to the syntactic category $T$ and $p_1, p_2$ denote expressions of syntactic category $P$. Expressions $e, e_1, e_2$ are arbitrary. We also allow the symbol $\emptyset$ to appear in expressions, which denotes the empty language.

In the first step of the algorithm, we transform $t$ into an equivalent expression $t_0$ which is either the symbol $\emptyset$, or contains no subexpression denoting the empty set. This can be done by a straightforward algorithm in linear time using the fact that an expression of the form $\mu x. t_1$ denotes the empty language iff $t_1[x/\emptyset]$, the expression obtained from $t_1$ by replacing each free occurrence of $x$ in $t_1$ by $\emptyset$ denotes the empty language.

Suppose now that $t_0$ is not the symbol $\emptyset$, so that $t_0$ is not empty. We construct another equivalent expression in which each subexpression of syntactic category $T$ denoting $\{\varepsilon\}$ is $\varepsilon$ itself. To achieve this, we determine for each subexpression $e$ of $t_0$ the set $\text{SYMBOLS}(e) \subseteq \Sigma \cup \mathcal{X}$ containing all the symbols that occur in some word of $|e|$ (or in a word in a pair of $|e|$, if $e$ is of type $P$). The recursion
rules for this are:

\[
\begin{align*}
\text{Symbols}(\varepsilon) &= \emptyset, \\
\text{Symbols}(x) &= \{x\}, \\
\text{Symbols}(a) &= \{a\}, \\
\text{Symbols}(e_1 \cdot e_2) &= \text{Symbols}(e_1 + e_2) = \text{Symbols}(e_1) \cup \text{Symbols}(e_2), \\
\text{Symbols}(p_1^\omega) &= \text{Symbols}(p_2^\omega) = \text{Symbols}(p_1), \\
\text{Symbols}(t_1 \times t_2) &= \text{Symbols}(t_1) \cup \text{Symbols}(t_2), \\
\text{Symbols}(\mu x.t) &= \text{Symbols}(t_1) - \{x\}.
\end{align*}
\]

Note that the correctness of these rules (e.g. the one for concatenation) depends on the assumption that no subexpression of \( t_\emptyset \) denotes the empty set.

Having computed \( \text{Symbols}(e) \) for each subexpression \( e \), observe that \( |e| = \{\varepsilon\} \) for a subexpression \( e \) of syntactic category \( T \) if and only if \( \text{Symbols}(e) = \emptyset \). Hence, during the computation of \( \text{Symbols}(\cdot) \), we can flag each subexpression of \( t_\emptyset \) of type \( T \) by a bit indicating whether it denotes the language \( \{\varepsilon\} \). Using this information, we can then replace each maximal subexpression denoting \( \{\varepsilon\} \) by \( \varepsilon \), yielding an equivalent expression \( t_{\varepsilon_\emptyset} \) containing no occurrence of the symbol \( \emptyset \) such that each subexpression of type \( T \) different from \( \varepsilon \) denotes a language containing at least one nonempty word. Applying now Claim A to \( t_{\emptyset \varepsilon} \), we get the desired decision procedure answering the question whether the given closed expression \( t \) denotes a language of well-ordered words.

All steps can be performed in (deterministic) linear time in the usual RAM model of computation, say, except for the computation of the function \( \text{Symbols}(\cdot) \) whose time complexity depends on the data structure chosen for representing sets of symbols. If this data structure is a self-balancing binary tree, which supports the construction of \( \emptyset \) and the singleton sets in constant time, the removal of one element from an \( n \)-element set in \( O(\log n) \) time and the construction of the union of two sets with \( n \) and \( k \) elements in \( O(\min\{n, k\} \cdot \log(n + k)) \) time (destroying the two sets, which is not a problem since only their emptiness flag is needed later, which is already stored), respectively, then we get an overall time complexity of \( O(n \cdot \log^2 n) \). Thus we have shown the following:

**Corollary 1.** The problem whether an arbitrary closed \( \mu \omega T_x \)-expression of syntactic category \( T \) denotes a language which consists of well-ordered words only, can be decided in \( O(n \cdot \log^2 n) \) time (in the usual RAM model of computation).

**References**

1. J.W. de Bakker and D. Scott, A theory of programs. *IBM Seminar Vienna*, August 1969.
2. N. Bedon. Finite automata and ordinals. *Theoretical Computer Science*, 156(1996), 119–144.
3. N. Bedon, A. Bès, O. Carton and C. Rispal. Logic and rational languages of words indexed by linear orderings. In: proc. *CSR 2008*, LNCS 5010, Springer, 2008, 76–85.
4. H. Bekić. Definable operations in general algebras, and the theory of automata and flowcharts. *IBM Seminar Vienna*, December 1969.

5. A. Bès and O. Carton. A Kleene theorem for languages of words indexed by linear orderings. In: proc. *DLT’2005*, LNCS 3572, Springer, 2005, 158–167.

6. S. L. Bloom and Z. Ésik. *Iteration Theories*. EATCS Monograph Series in Theoretical Computer Science, Springer, 1993.

7. L. Boasson. Context-free sets of infinite words. *Theoretical Computer Science (Fourth GI Conf., Aachen, 1979)*, LNCS 67, Springer, 1979, 1–9.

8. V. Bruyère and O. Carton. Automata on linear orderings. *J. Computer and System Sciences, 73*(2007), 1–24.

9. J. R. Büchi. The monadic second order theory of $\omega_1$. In: *Decidable theories, II*, Lecture Notes in Math., Vol. 328, Springer, 1973, 1–127.

10. Y. Choueka. Finite automata, definable sets, and regular expressions over $\omega^n$-tapes. *J. Computer and System Sciences, 17*(1978), no. 1, 81–97.

11. R. S. Cohen and A. Y. Gold. Theory of $\omega$-languages, parts one and two. *J. Computer and System Sciences, 15*(1977), 169–208.

12. B. Courcelle. Frontiers of infinite trees. *Theoretical Informatics and Applications, 12*(1978), 319–337.

13. Z. Ésik and S. Iván. Büchi context-free languages. *Theoretical Computer Science, 412*(2011), 805–821.

14. Z. Ésik and S. Iván. On Muller context-free grammars. *Theoretical Computer Science, 416*(2012), 17–32.

15. Z. Ésik and S. Iván. Hausdorff rank of scattered context-free linear orders, In: *LATIN 2012*, LNCS 7256, Springer, 2012, 291–302.

16. Z. Ésik and S. Okawa. On context-free languages of scattered words. *Developments in Language Theory 2012*, LNCS 7410, Springer, 2012, 142–153.

17. B. Khoussainov, S. Rubin and F. Stephan. Automatic linear orders and trees, *ACM Transactions on Computational Logic (TOCL), 6*(2005), 675–700.

18. R. Muller, Infinite sequences and finite machines. In: *4th Annual Symposium on Switching Circuit Theory and Logical Design*, IEEE Computer Society, 1963, 3–16.

19. M. Nivat. Sur les ensembles de mots infinis engendrés par une grammaire algébrique. (French) *Theoretical Informatics and Applications, 12*(1978), 259–278.

20. D. Perrin and J.-E. Pin. *Infinite Words*. Elsevier, 2004.

21. J. G. Rosenstein. *Linear Orderings*. Academic Press, 1982.

22. J. Wojciechowski. Classes of transfinite sequences accepted by finite automata, *Fundamenta Informaticae, 7*(1984), 191–223.

23. J. Wojciechowski. Finite automata on transfinite sequences and regular expressions, *Fundamenta Informaticae, 8*(1985), 379–396.