For submission to the Journal of Automata, Languages and Combinatorics
Created on May 12, 2023

HOW TO DEMONSTRATE CONTEXTFREENESS BY TREE-RESTRICTED GENERAL GRAMMARS

Alexander Meduna Zbyněk Křivka Martin Havel Ondřej Soukup

Brno University of Technology, Faculty of Information Technology, IT4Innovations Centre of Excellence, Božetěchova 1/2, 612 00 Brno, Czech Republic

krivka@fit.vut.cz (Z. Křivka) xhavel44@stud.fit.vutbr.cz (M. Havel)
meduna@fit.vut.cz (A. Meduna)

ABSTRACT

This paper introduces derivation trees for general grammars in Kuroda normal forms. Within these trees, it defines context-dependent pairs of nodes, which corresponds to rewriting two neighboring symbols by a non-context-free rule. It proves that the language generated by a general grammar in Kuroda normal form is context-free if there is a constant k such that every sentence w in the generated language is the frontier of a derivation tree in which any pair of neighboring paths contains k or fewer context-dependent pairs of nodes. The paper explains that this result represents a powerful tool for showing languages to be context-free. It sketches how to apply this tool in practice.

Keywords: general grammars, Kuroda normal forms, derivation trees, context-freeness

1. Introduction

Formal language theory has always intensively struggled to establish conditions under which general grammars generate a proper subfamily of the family of recursively enumerable languages because results like this often significantly simplify proofs that some languages are members of the subfamily. To illustrate, consider the well-known workspace theorem for general grammars, which plays a crucially important role in the grammatically oriented theory of formal languages as a whole (see Theorem III.10.1 in [14]). This theorem represents a powerful tool to demonstrate that if a general grammar H generates each of its sentences by a derivation satisfying a prescribed condition (specifically, this condition requires that there is a positive integer k such that H generates every sentence y in the generated language $L(H)$ by a derivation in which every sentential form x satisfies $|x| \leq k|y|$), then $L(H)$ is a member of the context-sensitive language family.

This work was supported by Brno University of Technology grant FIT-S-23-8209.

Concerning context-free languages, there have been achieved some results of this kind, too. First of all, [9] states that for a grammar, the set of terminal strings generated by left-to-right derivations is context-free. Second, [10] shows that the set of terminal strings generated by two-way derivations is context-free, which is further studied in [4]. Third, [3] demonstrates that a grammar generates a context-free language if the left-hand side of every rule contains only one nonterminal with terminal strings as the only context. Fourth, also [3] shows that if every rule of a general grammar has as its left context a string of terminal symbols at least as long as the right context, then the generated language is context-free. Fifth, [2] demonstrates that a grammar generates a context-free language if the right-hand side of every rule contains a string of terminals longer than any string of terminals between two nonterminals in the left-hand side.

Finally, the latest and closest result in [5] shows that counting the maximal number of non-context-free rules in a context-sensitive grammar used in a derivation leads to a context-free language. Compared to our result, we are able to work with a general grammar, not just context-sensitive, and we are able to work with an infinite number of non-context-free rules in a derivation to show the membership in the context-free language family if the conditions are fulfilled.

Continuing with this important investigation trend in the formal language theory, the present paper establishes another result of this kind based upon a restriction placed upon a graph-based representation of derivations in general grammars (noteworthy, none of the previously mentioned results on this subject has approached it in terms of the graph theory).

To give an insight into the new result achieved in the present paper, some terminology is first needed to be sketched. Recall that a general grammar G is in Kuroda normal form (see [8]) if any rule satisfies one of these forms:

$$AB \to CD,\ A \to BC,\ A \to B,\ A \to a, \text{ or } A \to \varepsilon,$$

where A, B, C, D are nonterminals, a is a terminal, and ε is the empty string. We define the notion of a derivation tree t graphically representing a derivation in G by analogy with this notion in terms of an ordinary context-free grammar (see Definition 6.8 on page 92 in [11]). In addition, however, we introduce context-dependent pairs of nodes in t as follows. In t, two paths are neighboring if no other path occurs between them. Let p and q be two neighboring paths in t. Let p contain a node k with a single child l, where k and l are labeled with A and C, respectively, and let q contain a node m with a single child n, where m and n are labeled with B and D, respectively. Let this four-node portion of t; consisting of k, l, m, and n; graphically represents an application of $AB \to CD$. Then, k and m are a context-dependent pair of nodes (see Fig. 1).

Making use of the terminology sketched above, we are now ready to explain the main result of this paper. It says that the language of G, $L(G)$, is context-free if and only if there is a constant k such that every $w \in L(G)$ is the frontier of a derivation tree d in which any pair of neighboring paths contains k or fewer context-dependent pairs of nodes. Apart from its obvious theoretical value, this result may be of some interest in practice, too. Specifically, some language processors, such as compiler

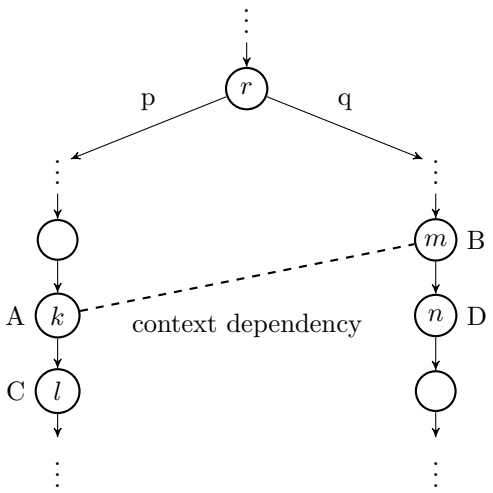

Figure 1: Illustration of context dependency in t

parsers, frequently require that the languages processed by them be context-free. As context dependency is obvious, the result stated above may play a useful role during the verification of this requirement.

The paper is organized as follows. First, Sections 2 and 3 give all the necessary terminology. Then, Section 4 establishes the main result of this paper. Finally, Section 5 closes this paper by showing an application perspective of the main result.

2. Preliminaries

We assume that the reader is familiar with discrete mathematics, including graph theory (see [1, 7, 6]) and formal language theory (see [11, 14, 13]).

A *directed graph* G is a pair $G = (V, E)$, where V is a finite *set of nodes* and $E \subseteq V \times V$ is a finite *set of edges*. For a node $v \in V$, the number of edges of the form $(x, v) \in E$, $x \in V$, is called an *in-degree* of v and is denoted by in-d(v). For a node $v \in V$, the number of edges of the form $(v, x) \in E$, $x \in V$, is called an *out-degree* of v and is denoted by out-d(v). Let $(v_0, v_1, \ldots, v_n)$ be an n-tuple of nodes, for some $n \geq 0$, where $v_i \in V$, for $0 \leq i \leq n$, and assume there exists an edge $(v_k, v_{k+1}) \in E$, for every pair of nodes v_k, v_{k+1}, where $0 \leq k \leq n-1$, then, we call it a *path of the length* n. Let $(v_0, v_1, \ldots, v_n)$ be a path in G, for some $n \geq 0$, and $v_0 = v_n$, then we call it *cycle*. A graph G is *acyclic* if it does not contain any cycle. Referring to a n-tuple of nodes, we sometimes omit brackets and commas if there is no risk of confusion.

For a set W, card(W) denotes its *cardinality*. An *alphabet* is a finite nonempty set —- the elements are called *symbols*. Let V be *alphabet*. V^* is the *set of all strings* over V. Algebraically, V^* represents the free monoid generated by V under the operation of concatenation. The identity of V^* is denoted by ε. Set $V^+ = V^* - \{\varepsilon\}$. Algebraically, V^+ is thus the free semigroup generated by V under the operation of concatenation.

For $w \in V^*$ and $a \in V$, $|w|$ denotes the *length of* w and $\#_a(w)$ denotes the *number of occurrences of the symbol* a *in* w. The *alphabet of* w, denoted by $\text{alph}(w)$, is the set of symbols appearing in w. For $i, j \geq 0$, define $\min(i, j) = i$ if $i < j$, otherwise $\min(i, j) = j$.

Let $\Rightarrow$ be a relation over V^*. Define the i-th power of $\Rightarrow$ as $\Rightarrow^i$, for $i \geq 0$. The transitive and transitive-reflexive closure of $\Rightarrow$ is denoted by $\Rightarrow^+$ and $\Rightarrow^*$, respectively. Unless explicitly stated otherwise, we write $x \Rightarrow y$ instead of $(x, y) \in \Rightarrow$ throughout.

The families of context-free, context-sensitive, and recursively enumerable languages are denoted by **CF**, **CS**, and **RE**, respectively.

Furthermore, we assume that the reader is familiar with Turing machines and their equivalence to general grammars (see [11]).

3. Definitions and Examples

Definition 1. An *(oriented) tree* is a directed acyclic graph $G = (V, E)$, with a specified node $\hat{r} \in V$ called *root* such that $\text{in-d}(\hat{r}) = 0$, and for all $x \in V - \{\hat{r}\}$, $\text{in-d}(x) = 1$ and there exists a path $(v_0, v_1, \ldots, v_n)$, where $v_0 = \hat{r}$, $v_n = x$, for some $n \geq 1$. For $v, u \in V$, where $(v, u) \in E$, v is called a *parent* of u, and u is called a child of v, respectively. For $v, u, z \in V$, where $(v, u), (v, z) \in E$, u is called a *sibling* of z. A node without children is called a *leaf*.

Let $G = (V, E)$ be a tree. Define the relation $<$ over V as follows. For a path $\alpha = (m_0, m_1, \ldots, m_k)$, where $m_0 = \hat{r}$, $m_i < m_k$, $0 \leq i \leq k - 1$, m_i is called a *predecessor* of m_k and m_k is called a *descendant* of m_i.

An *ordered tree* t is a tree, where for every set of siblings there exists a linear ordering. Assume o has the children $n_1, n_2, \ldots, n_r$ ordered in this way, where $r \geq 1$. Then, n_1 is the *leftmost child* of o, n_r is the *rightmost child* of o and n_i is the *direct left sibling* of n_{i+1}, n_{i+1} is the *direct right sibling* of n_i, $1 \leq i \leq r - 1$, and for $1 \leq j < k \leq r$, n_j is a *left sibling* of n_k and n_k is a *right sibling* of n_j. Let us extend the ordering according to the transitive closure of parent-children relation. Then, for a tree t we have a left-to-right ordered sequence of leafs $l_1, l_2, \ldots, l_k$, for some $k \geq 1$.

An ordered tree is called *labeled*, if there exists a set of labels $\mathcal{L}$ and a total mapping $l : V \to \mathcal{L}$. Let t be a labeled ordered tree, then the string of labels of all leaves written in the left-to-right order is called *frontier* of t and is denoted by $\text{frontier}(t)$. In what follows, we substitute a node of a tree by its label if there is no risk of confusion.

Let t be an ordered tree, and let t contain node o. Let $\alpha = (o, m_1, m_2, \ldots, m_r)$ and $\beta = (o, n_1, n_2, \ldots, n_s)$ be two paths in t, for some $r, s \geq 1$, such that o is the parent of m_1 and n_1, where

(I) m_1 is the direct left sibling of n_1;

(II) m_i is the rightmost child of m_{i-1}, and n_j is the leftmost child of n_{j-1}, $2 \leq i \leq r$, $2 \leq j \leq s$.

Then, α and β are two *neighboring paths* in t, α is a *left neighboring path* to β, and β is a *right neighboring path* to α.

Let us demonstrate the defined notions by the following example.

Example 2. The following graph (Fig. 2.) represents a labeled ordered tree t. Since any two distinct nodes have different labels, we will refer to their labels below. The root node $\hat{r}$ is a. It has no parent and two children b and c. Then, b is a sibling of c and c is a sibling of b. The leftmost child of b is d, while the rightmost is f. The node d is a left sibling of f, however, it is not the direct left sibling, which is e. The node f is the parent of k, but k has no child, so it is a leaf node. $horksmn = \text{frontier}(t)$. Consider the node e. The nodes a and b are predecessors of e, while i, j, o, p, and r are descendants of e'. The nodes c or d are not in predecessor relation with e, as they are neither predecessors of e, nor descendants of e. The sequence of nodes $bejpr$ is a path in t. The path bfk is neighboring to $bejpr$; unlike $abfk$, eio or bdh.

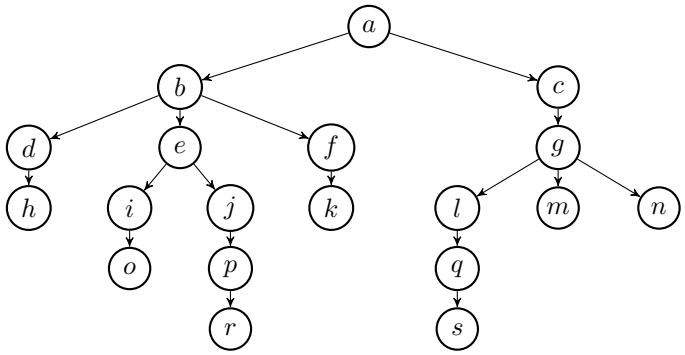

Figure 2: Labeled ordered tree t

Definition 3. A *general grammar* (GG for short) G is a quadruple $G = (V, T, P, S)$, where $V = T \cup N$ is a *total alphabet*, T is a *terminal alphabet*, N is a *nonterminal alphabet*, $P \subseteq (V - T)^* \times V^*$ is a finite *set of rules*. Instead of $p\colon (x, y) \in P$, where p is a unique *label*, we write $p\colon x \to y$. A rule and its label are interchangeable. $S \in N$ is a *start symbol*.

For every $u, v \in V^*$ and $p\colon x \to y \in P$, $\Rightarrow$ is the *direct derivation* relation over V^* and we write $uxv \Rightarrow uyv\,[p]$ or simply $uxv \Rightarrow uyv$. For $n \geq 0$, $\Rightarrow^n$ denotes the n-th power of $\Rightarrow$. Furthermore, $\Rightarrow^+$ and $\Rightarrow^*$ denote the transitive and transitive-reflexive closure of $\Rightarrow$, respectively. Let $\mathscr{F}(G) = \{w \in V^* \mid S \Rightarrow^* w\}$ denotes the *set of all sentential forms of* G. The *language* of G is $L(G) = \{w \in T^* \mid w \in \mathscr{F}(G)\}$.

G is *context-sensitive* if $x \to y \in P$ implies $|x| \leq |y|$. A rule $x \to y \in P$ is called *context-free* if its left-hand side consists of a single nonterminal; otherwise, it is a *non-context-free* rule. The grammar G is *context-free* if it has only context-free rules. In what follows, unless explicitly stated otherwise, we automatically assume that every general grammar and context-sensitive grammar is in Kuroda normal form.

GGs, context-sensitive GGs, and context-free GGs characterize **RE**, **CS**, and **CF**, respectively (see [13]).

Theorem 4. *A language L is recursively enumerable iff $L = L(G)$, where G is a general grammar in the Kuroda normal form.*

Proof. A language L is recursively enumerable iff L is generated by general grammar. Every GG can be converted to the Kuroda normal form (see Chapter 4 in [13]). □

Definition 5. Let t be a labeled ordered tree. The *left-bracketed representation of* t denoted by l-rep(t) can be obtained by applying the following recursive rules:

(I) If t has a root labeled $\hat{r}$ with subtrees $t_1, \ldots, t_k$ ordered in this way, then

$$\text{l-rep}(t) = \hat{r}\langle \text{l-rep}(t_1), \ldots, \text{l-rep}(t_k)\rangle.$$

(II) If t has a root labeled $\hat{r}$ with no direct descendants, then l-rep(t) $=$ $\hat{r}$.

Example 6. Consider labeled ordered tree t from Example 2. The left-bracketed representation of t is as follows.

$$a\langle b\langle d\langle h\rangle e\langle i\langle o\rangle j\langle p\langle r\rangle\rangle\rangle f\langle k\rangle\rangle c\langle g\langle l\langle q\langle s\rangle\rangle mn\rangle\rangle\rangle$$

Definition 7. Let $G = (V, T, P, S)$ be a GG in the Kuroda normal form.

(I) For $p\colon A \to x \in P$, $A\langle x\rangle$ is the *rule tree* that represents p.

(II) The *derivation trees* representing derivations in G are defined recursively as follows:

(A) One-node tree with a node labeled X is the derivation tree corresponding to $X \Rightarrow^0 X$ in G, where $X \in V$. If $X = \varepsilon$, we refer to the node labeled X as *ε-node* (*ε-leaf*); otherwise, we call it *non-ε-node* (*non-ε-leaf*).

(B) Let d be the derivation tree with frontier(d) $= uAv$ representing $X \Rightarrow^* uAv\,[\varrho]$ and let $p\colon A \to x \in P$. The derivation tree that represents

$$X \Rightarrow^* uAv\,[\varrho] \Rightarrow uxv\,[p]$$

is obtained by replacing the ith non-ε-leaf in d labeled A, with rule tree corresponding to p, $A\langle x\rangle$, where $i = |uA|$.

(C) Let d be the derivation tree with frontier(d) $= uABv$ representing $X \Rightarrow^* uABv\,[\varrho]$ and let $p\colon AB \to CD \in P$. The derivation tree that represents

$$X \Rightarrow^* uABv\,[\varrho] \Rightarrow uCDv\,[p]$$

is obtained by replacing the ith and $(i+1)$th non-ε-leaf in d labeled A and B with $A\langle C\rangle$ and $B\langle D\rangle$, respectively, where $i = |uA|$.

(III) A *derivation tree* in G is any tree t for which there is a derivation represented by t (see II in this definition).

Note, after replacement in II.C, the nodes A and B are the parents of the new leaves C and D, respectively, and we say that A and B are *context-dependent*, alternatively speaking, we say that there is a context dependency between A and B. In a derivation tree, two nodes are *context-independent* if they are not context-dependent.

Then, for any $p\colon A \to x \in P$, ${}_G\triangle(p)$ denotes rule tree corresponding to p. For any $A \Rightarrow^* x\,[\varrho]$ in G, where $A \in N$, $x \in V^*$, and $\varrho \in P^*$, ${}_G\triangle(A \Rightarrow^* x\,[\varrho])$ denotes the derivation tree corresponding to $A \Rightarrow^* x\,[\varrho]$. Just like we often write $A \Rightarrow^* x$ instead

of $A \Rightarrow^* x\,[\varrho]$, we sometimes simplify ${}_G\triangle(A \Rightarrow^* x\,[\varrho])$ to ${}_G\triangle(A \Rightarrow^* x)$ in what follows if there is no danger of confusion. Let ${}_G\blacktriangle$ denotes the set of all derivation trees in G. Finally, by ${}_G\triangle_x \in {}_G\blacktriangle$, we mean a derivation tree whose frontier is x, where $x \in \mathscr{F}(G)$.

If a node is labeled with a terminal, it is called a *terminal node*. If a node is labeled with a nonterminal, it is called a *nonterminal node*.

Let $\alpha = (o, m_1, m_2, \ldots, m_r)$ and $\beta = (o, n_1, n_2, \ldots, n_s)$ be two neighboring paths, where $r, s \geq 0$, α is the left neighboring path to β, and m_r and n_s are terminal nodes. Then, there is a t-tuple $\gamma = (g_1, g_2, \ldots, g_t)$ of nodes from α and t-tuple $\delta = (h_1, h_2, \ldots, h_t)$ of nodes from β, where $g_p < g_q$, for $1 \leq p < q \leq t$, $t < \min(r, s)$, and g_i and h_i are context-dependent, for $1 \leq i \leq t$. Let $\varrho = p_1 p_2 \ldots p_t$ be a string of non-context-free rules corresponding to context dependencies between γ and δ. We call ϱ the *right context of* α and the *left context of* β or the *context of* α *and* β. Consider a node m_i, where $1 \leq i \leq r$, and two $(t-k+1)$-tuples of nodes $\sigma = (g_k, g_{k+1}, \ldots, g_t)$ and $\varphi = (h_k, h_{k+1}, \ldots, h_t)$, where k is a minimal integer such that $m_i < g_k$. Then, a string of non-context-free rules $\tau = p_k p_{k+1} \ldots p_t$ corresponding to context dependencies between σ and φ is called the *right descendant context of* m_i, for some $1 \leq k \leq t$. Analogously, we define the notion of the *left descendant context* of a node n_j in β, for some $1 \leq j \leq s$.

Example 8. Let $G = (V,\ T,\ P,\ S)$ be a general grammar, where $V = \{S,\ S_a,\ S_b,\ X,\ X_a,\ X_b,\ Z_a,\ Z_b,\ A,\ 1,\ 2,\ 3,\ A_x,\ \overline{a},\ a,\ B,\ B_x,\ \overline{b},\ b\}$, $T = \{a,\ b\}$, and P contains the following rules:

(1) $S \to S_a B_x$
(2) $S \to S_b A_x$
(3) $S_a \to Z_a X$
(4) $S_b \to Z_b X$
(5) $X \to XX$
(6) $X \to AB$
(7) $X \to BA$
(8) $X \to AX_b$
(9) $X \to BX_a$
(10) $X_a \to XA$
(11) $X_b \to XB$
(12) $Z_a A \to AZ_a$
(13) $Z_a B \to BZ_a$
(14) $Z_b A \to AZ_b$
(15) $Z_b B \to BZ_b$
(16) $Z_a \to A$
(17) $Z_b \to B$
(18) $AB \to A_x B$
(19) $BA \to B_x A$
(20) $BA_x \to B_x A_x$
(21) $AA \to \overline{a}1$
(22) $1A \to \overline{a}2$
(23) $2A \to \overline{a}3$
(24) $3A_x \to \overline{a}\overline{a}$
(25) $A_x \to \overline{a}$
(26) $AA_x \to \overline{a}\overline{a}$
(27) $1A_x \to \overline{a}\overline{a}$
(28) $2A_x \to \overline{a}\overline{a}$
(29) $BB \to \overline{b}B_x$
(30) $B_x B \to \overline{b}B_x$
(31) $B_x B_x \to \overline{b}\overline{b}$
(32) $\overline{a} \to a$
(33) $\overline{b} \to b$

At this point, let us make only an informal observation that $L(G)$ is the language of all nonempty strings above T consisted of an equal number of as and bs, where every sequence of as is of a length between 1 and 5 and every sequence of bs is longer or equal 3. A rigorous proof comes later.

The string $aabbba$ can be obtained by the following derivation:

$$
\begin{array}{lll}
S \Rightarrow S_bA_x & [2] & \Rightarrow Z_bXA_x \quad [4] \\
\Rightarrow Z_bAX_bA_x & [8] & \Rightarrow Z_bAXBA_x \quad [11] \\
\Rightarrow Z_bAABBA_x & [6] & \Rightarrow AZ_bABBA_x \quad [14] \\
\Rightarrow AAZ_bBBA_x & [14] & \Rightarrow AABBBA_x \quad [17] \\
\Rightarrow AA_xBBBA_x & [18] & \Rightarrow AA_xBBB_xA_x \quad [20] \\
\Rightarrow \overline{aa}BBB_xA_x & [26] & \Rightarrow \overline{aab}B_xB_xA_x \quad [29] \\
\Rightarrow \overline{aabbb}A_x & [31] & \Rightarrow \overline{aabbba} \quad [25] \\
\Rightarrow a\overline{abbba} & [32] & \Rightarrow aa\overline{bbba} \quad [32] \\
\Rightarrow aab\overline{bba} & [33] & \Rightarrow aabb\overline{ba} \quad [33] \\
\Rightarrow aabbb\overline{a} & [33] & \Rightarrow aabbba \quad [32]
\end{array}
$$

A graph representing ${}_G\triangle(S \Rightarrow^* aabbba)$ is illustrated on Fig. 3.

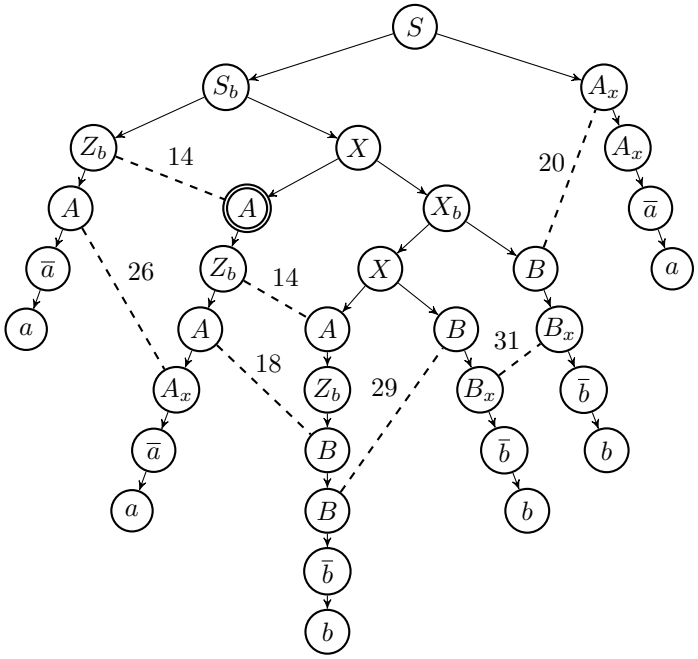

Figure 3: ${}_G\triangle_{aabbba}$

Let us note that dashed lines, numbers, and double-circle contour only denote the context dependencies, applied non-context-free rules, and a specific node, respectively,

and are not the part of the derivation tree.

Pairs of context-dependent nodes are linked with dashed lines, all the other nodes are context-independent. Since $aabbba = \text{frontier}({}_G\triangle_{aabbba})$, all leaves are terminal nodes. Every other node is a nonterminal node. For a pair of neighboring paths $\alpha = S_b Z_b A \overline{a} a$ and $\beta = S_b X A Z_b A A_x \overline{a} a$, a string $\varrho = 14\ 26$ is their context, it is the left context of β and the right context of α. Consider the double circled node A. Then, $\tau = 26$ is the left descendant context of A and $\varphi = 14\ 18$ is the right descendant context of A.

4. Results

Theorem 9. *A language L is context-free iff there is a constant $k \geq 0$ and a general grammar G such that $L = L(G)$ and for every $x \in L(G)$, there is a tree ${}_G\triangle_x \in {}_G\blacktriangle$ that satisfies:*

(I) *any two neighboring paths contain no more than k pairs of context-dependent nodes;*

(II) *out of neighboring paths, every pair of nodes is context-independent.*

Proof. Construction. Consider any $k \geq 0$. Let $G = (V, T, P, S)$ be a GG such that $L(G) = L$. Set $N = V - T$. Let $P_{cs} \subseteq P$ denote the set of all non-context-free rules of G. Set

$$N' = \{A_{l|r} \mid A \in N,\ l, r \in (P_{cs} \cup \{\varepsilon\})^k\}.$$

Construct a grammar $G' = (V', T, P', S_{\varepsilon|\varepsilon})$, where $V' = N' \cup T$. Set $P' = \emptyset$. Construct P' by performing I through IV given next.

(I) For all $A \to B \in P$, $A, B \in N$, and $l, r \in (P_{cs} \cup \{\varepsilon\})^k$, add $A_{l|r} \to B_{l|r}$ to P';

(II) for all $A \to a \in P$, $A \in N$, $a \in (T \cup \{\varepsilon\})$, add $A_{\varepsilon|\varepsilon} \to a$ to P';

(III) for all $A \to BC \in P$, where $A, B, C \in N$, and $r, l, x \in (P_{cs} \cup \{\varepsilon\})^k$, add $A_{l|r} \to B_{l|x} C_{x|r}$ to P';

(IV) for all $p\colon AB \to CD \in P$, $A, B, C, D \in N$, $x, z \in (P_{cs} \cup \{\varepsilon\})^k$, and $y \in (P_{cs} \cup \{\varepsilon\})^{k-1}$, add $A_{x|py} \to C_{x|y}$ and $B_{py|z} \to D_{y|z}$ to P'.

Basic idea. Notice nonterminal symbols. Since every pair of neighboring paths of G contains a limited number of context-dependent nodes, all of its context-dependencies are encoded in nonterminals. G' nondeterministically decides about all context-dependencies while introducing a new pair of neighboring paths by rules III. A new pair of neighboring paths is introduced with every application of

$$A_{l|r} \to B_{l|x} C_{x|r},$$

where x encodes a new descendant context. Context dependencies are realized later by context-free rules IV.

Since P' contains no non-context-free rule, G' is context-free. Next, we proof $L(G) = L(G')$ by establishing Claims 1 through 3. Define the new homomorphism $\gamma : V' \to V$, $\gamma(A_{l|r}) = A$, for $A_{l|r} \in N'$, and $\gamma(a) = a$ otherwise.

Claim 1. *If $S \Rightarrow^m w$ in G, where $m \geq 0$ and $w \in V^*$, then $S_{\varepsilon|\varepsilon} \Rightarrow^* w'$ in G', where $w' \in V'^*$ and $\gamma(w') = w$.*

Proof. We prove this by induction on $m \geq 0$.

Basis. Let $m = 0$. That is $S \Rightarrow^0 S$ in G. Clearly, $S_{\varepsilon|\varepsilon} \Rightarrow^0 S_{\varepsilon|\varepsilon}$ in G', where $\gamma(S_{\varepsilon|\varepsilon}) = S$, so the basis holds.

Induction Hypothesis. Suppose that there exists $n \geq 0$ such that Claim 1 holds for all $0 \leq m \leq n$.

Induction Step. Let $S \Rightarrow^{n+1} w$ in G. Then, $S \Rightarrow^n v \Rightarrow w$, where $v \in V^*$, and there exists $p \in P$ such that $v \Rightarrow w\,[p]$. By the induction hypothesis, $S_{\varepsilon|\varepsilon} \Rightarrow^* v'$, where $\gamma(v') = v$, in G'. Next, we consider the following four forms of p.

(I) Let $p\colon A \to B \in P$, for some $A, B \in N$. Without any loss of generality, suppose l and r are a left descendant context and a right descendant context of A. By the construction of G', there exists a rule $p'\colon A_{l|r} \to B_{l|r} \in P'$, where $A_{l|r}, B_{l|r} \in v'$. Then, there exists a derivation $v' \Rightarrow w'\,[p']$ in G', where $\gamma(w') = w$.

(II) Let $p\colon A \to a \in P$, for some $A \in N$ and $a \in T \cup \{\varepsilon\}$. Since a is a terminal node, it has empty descendant contexts. By the construction of G', there exists a rule $p'\colon A_{\varepsilon|\varepsilon} \to a \in P'$, where $A_{\varepsilon|\varepsilon} \in v'$. Then, there exists a derivation $v' \Rightarrow w'\,[p']$ in G', where $\gamma(w') = w$.

(III) Let $p\colon A \to BC \in P$, for some $A, B, C \in N$. Without any loss of generality, suppose l and r are a left descendant context and a right descendant context of A, and $x \in (P_{cs} \cup \{\varepsilon\})^k$ is a context of neighboring paths beginning at this node. By the construction of G', there exists a rule $p'\colon A_{l|r} \to B_{l|x}C_{x|r} \in P'$, where $A_{l|r}, B_{l|r}, C_{x|r} \in v'$. Then, there exists a derivation $v' \Rightarrow w'\,[p']$ in G', where $\gamma(w') = w$.

(IV) Let $p\colon AB \to CD \in P$, for some $A, B, C, D \in N$. By the assumption stated in Theorem 9, A and B occur in two neighboring paths denoted by α and β, respectively. Without any loss of generality, suppose that a context of α and β is a string $c \in (P_{cs} \cup \varepsilon)^k$, where $c = pc_f$, and l is a left descendant context, r is a right descendant context of A, B, respectively. By the construction of G', there exist two rules

$$p'_l\colon A_{l|pc_f} \to C_{l|c_f}, \quad p'_r\colon B_{pc_f|r} \to D_{c_f|r} \in P',$$

where $A_{l|pc_f}, C_{l|c_f}, B_{pc_f|r}, D_{c_f|r} \in v'$. Then, there exists a derivation $v' \Rightarrow^2 w'\,[p'_l p'_r]$ in G', where $\gamma(w') = w$.

Notice (IV). The preservation of the context is achieved by nonterminal symbols. Since the stored context is reduced symbol by symbol from left to right direction in both α and β, G' simulates the applications of non-context-free rules of G.

We covered all possible forms of p, so the claim holds. □□

Claim 2. *Every $x \in \mathscr{F}(G')$ can be derived in G' as follows.*

$$S_{\varepsilon|\varepsilon} = x_0 \Rightarrow^{d_1} x_1 \Rightarrow^{d_2} x_2 \Rightarrow^{d_3} \cdots \Rightarrow^{d_{h-1}} x_{h-1} \Rightarrow^{d_h} x_h = x,$$

for some $h \geq 0$, where $d_i \in \{1, 2\}$, $1 \leq i \leq h$, so that

(I) *if $d_i = 1$, then $x_{i-1} = uA_{l|r}v$, $x_i = uzv$, $x_{i-1} \Rightarrow x_i\ [A_{l|r} \to z]$, where $u, v \in V'^*$, $z \in \{B_{l|r}, C_{l|x}D_{x|r}, a\}$, for some $A_{l|r}, B_{l|r}, C_{l|x}, D_{x|r} \in N'$, $a \in (T \cup \{\varepsilon\})$;*

(II) *if $d_i = 2$, then $x_{i-1} = uA_{x|py}B_{py|z}v$, $x_i = uC_{x|y}D_{y|z}v$, and*

$$uA_{x|py}B_{py|z}v \Rightarrow uC_{x|y}B_{py|z}v\ [A_{x|py} \to C_{x|y}] \Rightarrow uC_{x|y}D_{y|z}v\ [B_{py|z} \to D_{y|z}],$$

for some $u, v \in V'^*$ and $A_{x|py}, B_{py|z}, C_{x|y}, D_{y|z} \in N'$.

Proof. Since G' is context-free, without any loss of generality in every derivation of G' we can always reorder applied rules to satisfy Claim 2. □□

Claim 3. *Let $S_{\varepsilon|\varepsilon} \Rightarrow^{d_1} x_1 \Rightarrow^{d_2} \cdots \Rightarrow^{d_{m-1}} x_{m-1} \Rightarrow^{d_m} x_m$ in G' be a derivation that satisfies Claim 2, for some $m \geq 0$. Then, $S \Rightarrow^* w$ in G, where $\gamma(x_m) = w$.*

Proof. We prove this by induction on $m \geq 0$.

Basis. Let $m = 0$. That is $S_{\varepsilon|\varepsilon} \Rightarrow^0 S_{\varepsilon|\varepsilon}$ in G'. Clearly, $S \Rightarrow^0 S$ in G. Since $\gamma(S_{\varepsilon|\varepsilon}) = S$, the basis holds.

Induction Hypothesis. Suppose that there exists $n \geq 0$ such that Claim 3 holds for all $0 \leq m \leq n$.

Induction Step. Let $S_{\varepsilon|\varepsilon} \Rightarrow^{d_1} x_1 \Rightarrow^{d_2} \cdots \Rightarrow^{d_{n-1}} x_{n-1} \Rightarrow^{d_n} x_n \Rightarrow^{d_{n+1}} x_{n+1}$ in G' be a derivation that satisfies Claim 2. By the induction hypothesis, $S \Rightarrow^* v$, $v \in V^*$, where $\gamma(x_n) = v$, in G. Divide the proof into two parts according to d_{n+1}.

(A) Let $d_{n+1} = 1$. By the construction of G', there exists a rule $p' \in P'$ such that $x_n \Rightarrow^{d_{n+1}} x_{n+1}\,[p']$. Next, we consider the following three forms of p'.

  (I) Let $p'\colon A_{l|r} \to B_{l|r} \in P'$, for some $A, B \in N$ and $l, r \in (P_{cs} \cup \{\varepsilon\})^k$. By the construction of G', rule p' was introduced by some rule $p\colon A \to B \in P$. Then, there exists a derivation $v \Rightarrow w\,[p]$, where $\gamma(x_{n+1}) = w$.

  (II) Let $p'\colon A_{\varepsilon|\varepsilon} \to a \in P'$, for some $A \in N$ and $a \in T \cup \{\varepsilon\}$. By the construction of G', rule p' was introduced by some rule $p\colon A \to a \in P$. Then, there exists a derivation $v \Rightarrow w\,[p]$, where $\gamma(x_{n+1}) = w$.

  (III) Let $p'\colon A_{l|r} \to B_{l|x}C_{x|r} \in P'$, for some $A, B, C \in N$ and $l, r, x \in (P_{cs} \cup \{\varepsilon\})^k$. By the construction of G', rule p' was introduced by some rule $p\colon A \to BC \in P$. Then, there exists a derivation $v \Rightarrow w\,[p]$, where $\gamma(x_{n+1}) = w$.

(B) Let $d_{n+1} = 2$. Then, $x_n \Rightarrow^{d_{n+1}} x_{n+1}$ is equivalent to

$$u_1 A_{x|py} B_{py|z} u_2 \Rightarrow u_1 C_{x|y} B_{py|z} u_2 \ [p'_1] \Rightarrow u_1 C_{x|y} D_{y|z} u_2 \ [p'_2],$$

where $x_n = u_1 A_{x|py} B_{py|z} u_2$, $x_{n+1} = u_1 C_{x|y} D_{y|z} u_2$, and

$$p'_1 \colon A_{x|py} \to C_{x|y}, \ p'_2 \colon B_{py|z} \to D_{y|z} \in P',$$

for some $u_1, u_2 \in V'^*$ and $A_{x|py}$, $B_{py|z}$, $C_{x|y}$, $D_{y|z} \in N'$. By the construction of G', rules p'_1 and p'_2 were introduced by some rule $p \colon AB \to CD \in P$, Then, there exists a derivation $v \Rightarrow w\,[p]$, where $\gamma(x_{n+1}) = w$.

We covered all possibilities, so the claim holds. □□

By Claims 1 and 3, $S \Rightarrow^* w$ in G iff $S_{\varepsilon|\varepsilon} \Rightarrow^* w'$ in G', where $\gamma(w') = w$. If $S \Rightarrow^* w$ in G and $w \in T^*$, then $w \in L(G)$. Since $\gamma(w') = w' = w$, for $w \in T^*$, $w' \in L(G')$. Therefore, $L(G) = L(G')$ and Theorem 9 holds. □

Consider Theorem 9. Observe that the second condition is superfluous whenever G is context-sensitive. Since a grammar is in the Kuroda normal form and no symbol can be erased, all context dependencies are within pairs of neighboring paths.

Theorem 10. *A language L is context-free iff there is a constant $k \geq 0$ and a context-sensitive grammar G such that $L = L(G)$ and for every $x \in L(G)$, there is a tree ${}_G\triangle_x \in {}_G\blacktriangle$, where any two neighboring paths contain no more than k pairs of context-dependent nodes.*

Proof. Prove this by analogy with the proof of Theorem 9. □

5. Use

In this section, we explain how to apply the results achieved in the previous section in order to demonstrate the contextfreeness of a language, L. As a rule, this demonstration follows the next three-step proof scheme.

(I) Construct a general grammar G in the Kuroda normal form.
(II) Prove $L(G) = L$.
(III) Prove that G satisfies conditions from Theorem 9 or Theorem 10, depending on whether G is context-sensitive.

Reconsider the grammar G from Example 8. Following the proof scheme sketched above, we next prove that $L(G) \in \mathbf{CF}$.

Consider G constructed in Example 8. Next, we show that for G,

$$L(G) = \{w \in (A \cup \{\varepsilon\})(BA)^*(B \cup \{\varepsilon\}) \mid \#_a(w) = \#_b(w),$$
$$A = \{a^i \mid 1 \leq i \leq 5\}, B = \{b^i \mid i \geq 3\}, \text{ and } |w| > 0\}.$$

Without any loss of generality, every terminal derivation of G can be divided into the following 5 phases, where each rule may be used only in a specific phase:

(a) 1–4 (b) 5–11 (c) 12–17 (d) 18–31 (e) 32–33

Next, we describe these phases in a greater detail.

(a) First, we generate one of the following two strings by rules 1 through 4.

$$Z_a X B_x, Z_b X A_x$$

Possibly applicable rule 25 may be postponed for phase (d) without affecting the derivation, since rules in the previous phases cannot rewrite A_x.

(b) The rules 5 through 11 are the only with X, X_a, or X_b on their left-hand sides, therefore we can group all their applications in a sequence to get a sentential form from

$$\{Z_a, Z_b\}\{A, B\}^*\{A_x, B_x\}.$$

(c) The rules 12 through 17 possibly shift Z_a or Z_b to the right and rewrite it to A or B, respectively. Since these rules are the only with Z_a, Z_b on their left-hand sides, they can be always prioritized before the rest of rules without any loss of generality.

$$\{A, B\}^*\{A_x, B_x\}$$

(d) All the remaining rules may be applied in this phase. However, we can exclude rules 32 and 33, so we get a sentential form from

$$\{\overline{a}, \overline{b}\}^*.$$

(e) Since rules 32 and 33 are context-free and produce terminal symbols, they can be always postponed until the end of any successful derivation.

$$\{a, b\}^* = T^*$$

Let us add a few remarks concerning (a) through (e).

Phase (a) is very straightforward. Only notice that it is decided whether the generated string finally ends with a or b and the paired symbol is stored in Z_a or Z_b for phase (c).

In phase (b) an arbitrary string of As and Bs is generated from the initial symbol X. However, for every A, one B is generated and vice versa, so their numbers are always kept equal.

In phase (a) the grammar decides about the last symbol and stores the paired one, which, however, need not to be the first one. Therefore phase (c) determines its final position, while possibly shifting it to the right and finally rewriting to A or B.

Phase (d) is the most tricky. It starts with a sentential form wc, where $w \in \{A, B\}^*$, $c \in \{A_x, B_x\}$. Informally speaking, it consists of the sequences of As which should be at most 5 symbols long, and Bs which should be at least 3 symbols long. Rules 18

through 31 are designed to ensure these restrictions. To give an example, suppose wc is as follows.

$wc = AAAABBBBABBBBAA_x$

First, by rules 18 through 20 the last symbol in every sequence is marked with index x. Otherwise, rules 24 through 28 and rule 31 never become applicable and all the unmarked sequences become permanent resulting into an unsuccessful derivation. The last sequence is already marked.

$$\begin{aligned} & AAAABBBBABBBBAA_x \\ \Rightarrow\ & AAAA_xBBBBABBBBAA_x && [18] \\ \Rightarrow\ & AAAA_xBBBBA_xBBBBAA_x && [18] \\ \Rightarrow\ & AAAA_xBBBB_xA_xBBBBAA_x && [20] \\ \Rightarrow\ & AAAA_xBBBB_xA_xBBB_xBAA_x && [19] \end{aligned}$$

Notice, one symbol sequence of As is legal. Then, every sequence of As is processed in left-to-right direction by rules 21 through 24, but can be successfully rewritten earlier by rules 25 through 28, in the case it consists of less than 5 symbols. Thus, a longer sequence leads to an unsuccessful derivation.

$$\begin{aligned} & AAAA_xBBBB_xA_xBBB_xAA_x \\ \Rightarrow\ & \overline{a}1AA_xBBBB_xA_xBBB_xAA_x && [21] \\ \Rightarrow\ & \overline{aa}2A_xBBBB_xA_xBBB_xAA_x && [22] \\ \Rightarrow\ & \overline{aaaa}BBBB_xA_xBBB_xAA_x && [27] \\ \Rightarrow\ & \overline{aaaa}BBBB_x\overline{a}BBB_xAA_x && [25] \\ \Rightarrow\ & \overline{aaaa}BBBB_x\overline{a}BBB_x\overline{aa} && [26] \end{aligned}$$

If the processing does not start from the leftmost symbol in the current sequence, it remains permanent. Every sequence of Bs is processed by applying rule 29, zero or multiple times rule 30, and finally rule 31. It ensures the lengths of sequences of Bs are at least 3 symbols.

$$\begin{aligned} & \overline{aaaa}BBBB_x\overline{a}BBB_x\overline{aa} \\ \Rightarrow\ & \overline{aaaab}B_xBB_x\overline{a}BBB_x\overline{aa} && [29] \\ \Rightarrow\ & \overline{aaaabb}B_xB_x\overline{a}BBB_x\overline{aa} && [30] \\ \Rightarrow\ & \overline{aaaabbbba}BBB_x\overline{aa} && [31] \\ \Rightarrow\ & \overline{aaaabbbbab}B_xB_x\overline{aa} && [29] \\ \Rightarrow\ & \overline{aaaabbbbabbbaa} && [31] \end{aligned}$$

Notice, it depends on the order of applied rules only within one sequence. Multiple sequences may be processed at random without affecting the derivation.

In phase (e), a resulting terminal string is generated by rules 32 and 33.

$\overline{aaaabbbbabbbaa} \Rightarrow^* aaaabbbbabbbaa$

Therefore, if the derivation is terminating, we achieve a string with an equal number of as and bs, where every sequence of as is at most 5 symbols long and every sequence of bs is at least 3 symbols long.

Grammar G is obviously a context-sensitive grammar in the Kuroda normal form. Let us now show that for any $x \in L(G)$, there is ${}_G\triangle_x \in {}_G\blacktriangle$, where any two neighboring paths contain no more than 2 pairs of context-dependent nodes.

Every pair of context-dependent nodes in ${}_G\triangle_x$ corresponds to one non-context-free rule in $S \Rightarrow^* x$. Consider the six phases sketched above. Observe that phases (a), (b), and (e) contain only context-free rules, so we have only to investigate (c) and (d). On the other hand, (c) and (d) contain no rule of the form $A \to BC$, thus the number of neighboring paths remains unchanged.

In (c) by rules 12 through 17 the derivation may proceed in left-to-right direction through the whole sentence form (except the rightmost symbol) introducing a context dependency between every pair of neighboring paths.

In (d), first, the context dependency is introduced between all neighboring paths representing the borders between the sequences of As and Bs by rules 18 through 20. Second, every sequence of As or Bs is processed in the left-to-right direction by non-context-free rules 21 through 31 introducing a context dependency between all neighboring paths representing symbols inside the sequences of As and Bs.

No other non-context-free rule is applied, therefore, no other context-dependent pair of nodes can occur. Then, every pair of neighboring paths may contain at most one context-dependent pair of nodes introduced in phase (c) and one introduced in phase (d).

Since G is a context-sensitive grammar in the Kuroda normal form, where for every $x \in L(G)$, there is ${}_G\triangle_x \in {}_G\blacktriangle$, where any two neighboring paths contain no more than 2 pairs of context-dependent nodes, by Theorem 10, $L(G) \in \mathbf{CF}$.

As a concluding result, we evaluate the results and discuss the decidability of a problem whether a given general grammar satisfies the properties from Theorem 9. Since Theorem 9 proves the equivalence, it suffices to prove that it is undecidable to say whether a given general grammar generates a context-free language.

Theorem 11. *The problem of the membership to the family of context-free languages for a general grammar is undecidable.*

Proof. Any language generated by a general grammar can be generated by a Turing machine. Therefore, for this nontrivial semantic property, Rice's theorem (see [12]) proves that it is undecidable. □

Observe that the main result does not try to provide a general solution. The algorithm is a tool for proving that a subset of general grammars can be converted to context-free grammars. This algorithm applied to a general grammar with a certain condition met, where proof and conversion are not trivial, provides a powerful tool to show the membership to a context-free languages, which is filling another gap in the theory of formal languages. The results have also an application perspective such as context-free grammars are more compatible with compilers designs, parsing, and natural language processing.

References

[1] A. Aho, J. Ullman, *The Theory of Parsing, Translation, and Compiling*. Prentice-Hall, Series in Automatic Computation, 1972.

[2] B. S. Baker, Non-context-free grammars generating context-free languages. *Information and Control* **24** (1974) 3, 231–246.

[3] R. V. Book, Terminal context in context-sensitive grammars. *SIAM J. Comput.* **1** (1972) 1, 20–30.

[4] R. V. Book, On the structure of context-sensitive grammars. *International Journal of Computer & Information Sciences* **2** (1973), 129–139.

[5] H. Bordihn, V. Mitrana, On the degrees of non-regularity and non-context-freeness. *Journal of Computer and System Sciences* **108** (2020), 104–117.

[6] T. Cormen, C. Leiserson, R. Rivest, *Introduction to Algorithms (Appendix B.5)*. McGraw-Hill, 2002.

[7] M. A. Harrison, *Introduction to Formal Language Theory*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1978.

[8] S.-Y. Kuroda, Classes of languages and linear-bounded automata. *Information and Control* **7** (1964), 207–223.

[9] G. Matthews, A note on asymmetry in phrase structure grammars. *Information and Control* **7** (1964) 3, 360–365.

[10] G. Matthews, Two-way languages. *Information and Control* **10** (1967) 2, 111–119.

[11] A. Meduna, *Formal Languages and Computation: Models and Their Applications*. Taylor & Francis, New York, 2014.

[12] H. G. Rice, Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society* **74** (1953), 358—-366.

[13] G. Rozenberg, A. Salomaa, *Handbook of Formal Languages, Vol. 1: Word, Language, Grammar*. Springer-Verlag, New York, 1997.

[14] A. Salomaa, *Formal Languages*. Academic Press, London, 1973.