On Separations of LR(0)-Grammars by Two Types of Pumping Patterns

Martin Plátek[1,*], František Mráz[1], Dana Pardubská[2,†], Daniel Průša[3,‡], and Jiří Šíma[4,*]

[1] Charles University, Department of Computer Science
Malostranské nám. 25, 118 00 PRAHA 1, Czech Republic
martin.platek@mff.cuni.cz, frantisek.mraz@mff.cuni.cz

[2] Comenius University in Bratislava, Department of Computer Science
Mlynská Dolina, 84248 Bratislava, Slovakia
pardubska@dcs.fmph.uniba.sk

[3] Czech Technical University, Department of Cybernetics
Karlovo nám. 13, 121 35 Prague 2, Czech Republic
prusa@fel.cvut.cz

[4] Institute of Computer Science of the Czech Academy of Sciences,
P. O. Box 5, 18207 Prague 8, Czech Republic
sima@cs.cas.cz

Abstract: We present two types of pumping patterns that allow a total separation inside the class of LR(0)-grammars. Using the same type of pumping patterns, we obtain a total separation inside of linear LR(0)-grammars.

This type of study has a long-term motivation from computational linguistics and the area of syntactic error localization. A recent motivation also comes from the field of formal models of neural networks.

1 Introduction

This paper follows the paper [6], where we have studied restarting automata recognizing deterministic context-free languages (DCFL) with a particular type of pumping patterns. This type of pumping patterns ensures the non-regularity of the recognized languages.

In this paper, we are looking for two types of pumping patterns. The first type of pumping patterns should ensure the non-regularity of the languages recognized by a certain type of restarting automata. In contrast, the second type of pumping patterns should ensure the regularity of the languages recognized by this type of restarting automata. The union of both of these types of pumping patterns should cover all possible pumping patterns defined by the mentioned type of restarting automata.

Any deterministic context-free language L can be accepted by a LR(1)-analyzer (using lookahead of size 1), but the language $L \cdot \{\$\}$, where \$ is a special end-marker not in the alphabet of L) can be accepted by a LR(0)-analyzer [3]. In [4], it was shown that any deterministic context-free language can be accepted by a deterministic monotone restarting automaton. This was proved by simulating any given LR(0)-analyzer by a deterministic monotone restarting automaton. Hence, the computations of such automata were controlled by LR(0)-grammars. Therefore, in what follows, we will use constructions based on properties of phrase structures generated by LR(0)-grammars and we will study pumping patterns based on LR(0)-grammars.

This paper should serve as a step to refine the concepts from [5], where it was shown that restarting automata can serve as a formal model for functional generative description (FGD) of a natural language. Functional generative description is a dependency-based *descriptive system* that has been developed since the 1960's, see esp. [7]. FGD was originally implemented as a generative procedure, but later its authors have been interested in a more declarative representation. The subject of the paper [5] concerns the foundations of a *reduction system* using a complex restarting automaton. The reduction system is more complex than a reduction system for a (shallow) syntactic analyzer since it provides not only the possibility of checking the well-formedness of the (surface) analysis of a sentence, but also its meaning – tectogrammatical representation in terms of FGD. Such a reduction system makes it possible to define the analysis as well as the synthesis of a sentence formally.

A descriptive system [5] *DS* of a natural language L should determine:

(a) The set *LC* of all correct sentences of the language L.

(b) The set *LM* of all correct meaning descriptions of the language L. *LM* represents all meanings of all sentences in *LC*.

(c) A relation $SH \subseteq LC \times LM$ between *LC* and *LM*. The relation describes the ambiguity and the synonymy of L.

Let us note that *LM* is an artificial (formal) unambiguous language, which can be described by a restarting auto-

[*] Supported partially by the grant GA19-05704S of the Czech Science Foundation and by the institutional support RVO: 67985807.

[†] Supported by the grant 1/0601/20 of the Slovak Scientific Grant Agency VEGA.

[‡] Supported by the grant GA19-21198S of the Czech Science Foundation.

maton controlled by an LR(0)-grammar. An important feature modeled by *LM* is valency. More about valency can be found in [1]. There are types of valency features that can be modeled as pumping patterns of non-regular type (e.g., obligatory adjuncts), and there are other valency features that can be modeled as pumping patterns of regular type (e.g., optional adjuncts).

Another recent motivation for this paper comes from a completely different area. The last results of this paper should support the analysis of analog neuron hierarchy of binary-state neural networks (NNs) with an increasing number of extra analog-state neurons, which has been introduced in [8] for studying the power of NNs with realistic weights between integers and rational numbers with respect to the Chomsky hierarchy.

Note that the problem of the regularity of deterministic context-free languages gained attention already in the 60s. Algorithms deciding whether a given deterministic pushdown automaton accepts a regular language were proposed by Stearns [9] and Valiant [10].

This paper is structured as follows. The next section contains basic notions and basic properties of LR(0)-grammars. Section 3 introduces the concepts of our interest and presents the first main result. Section 4 presents the results about total separations. The paper concludes with a summary of future work to be done to fulfill all our plans.

2 Basic Notions

We suppose that the reader is familiar with the basics of formal language and automata theory presented, e.g., in [3]. For technical reasons, we give a list of some notions and definitions related to formal grammars.

Context-free grammars. A context-free grammar G is defined by a 4-tuple $G = (V, \Sigma, R, S)$ where $\Sigma \subset V$, $N = V \setminus \Sigma$, and N, Σ are finite sets of nonterminal and terminal symbols, respectively, $R \subset N \times V^*$ is a finite relation of rewrite (production) rules, and $S \in N$ is the start symbol. We use the usual notation $A \to \beta \in R$ for a production rule $(A, \beta) \in R$. This rule can be applied to $u = u_1 A u_2$ where $u_1, u_2 \in V^*$, yielding $v = u_1 \beta u_2$, which is denoted as $u \Rightarrow v$, or $u \Rightarrow_G v$. We write $u \Rightarrow_R v$ if the nonterminal A substituted by $u \Rightarrow v$ is the rightmost nonterminal in u.

The grammar G generates the context-free language $L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$ where $\Rightarrow^*$ ($\Rightarrow_R^*$) is the reflexive transitive closure of the binary relation $\Rightarrow$ ($\Rightarrow_R$) on V^*. Note that $L(G)$ can also be equivalently defined as $L(G) = \{w \in \Sigma^* \mid S \Rightarrow_R^* w\}$.

We write $u \Rightarrow^{\leq p} v$ for some non-negative integer p if $u \Rightarrow^* v$ and v is derived from u by at most p derivation steps ($\Rightarrow$). A similar meaning has the denotation $u \Rightarrow_R^{\leq p} v$. In what follows, we will primarily work with the rightmost derivations and write $\Rightarrow$ instead of $\Rightarrow_R$.

Any non-empty context-free language $L = L(G) \neq \emptyset$ can be generated by a *reduced* context-free grammar G that excludes unreachable and unproductive symbols, that is, for every $A \in N$, there exist $\alpha, \beta \in V^*$ such that $S \Rightarrow^* \alpha A \beta$, and for every $A \in N$ there is $w \in \Sigma^*$ such that $A \Rightarrow^* w$, respectively. In the following, we work with reduced context-free grammars only.

We say that a context-free grammar $G = (V, \Sigma, R, S)$ is *linear* if the right-hand side of any rule from R contains at most one variable (nonterminal).

For any non-empty word $a_1 \cdots a_n$, with $a_1, \ldots, a_n \in \Sigma$, the derivation $S \Rightarrow^* a_1 \cdots a_n$ can be described by a *derivation tree* which is an ordered tree satisfying:

1. Inner vertices are labeled with nonterminals.
2. The root is labeled with the start symbol S.
3. If inner vertex labeled with nonterminal $A \in N$ has children labeled (from left to right) with $\alpha_1, \alpha_2, \ldots, \alpha_k \in V$ then $A \to \alpha_1 \alpha_2 \cdots \alpha_k \in R$; A together with the subtrees rooted in its children form a complete branch of the tree.
4. The leaves in the tree are labeled (from left to right) with the terminal symbols $a_1, \ldots, a_n \in \Sigma$.

Removing condition 2 we get the definition of a *(computation) derivation sub-tree*.

A grammar G is *unambiguous* if every non-empty string $w \in L(G)$ has a unique derivation tree, or equivalently a unique rightmost derivation.

The results of our paper heavily depend on the characterization of the class of deterministic context-free languages by LR(0)-grammars and corresponding parsers. Following [3], we, therefore, give the necessary definitions and the most relevant results.

Definition 1. *Let $G = (V, \Sigma, R, S)$ be a context-free grammar, $N = V \setminus \Sigma$, $\gamma \in V^*$. A* handle *of γ is an ordered pair (r, i), $r \in R, i \geq 0$ such that there exists $A \in N, \alpha, \beta \in V^*$ and $w \in \Sigma^*$ such that*

(a) $S \Rightarrow_R^* \alpha A w \Rightarrow_R \alpha \beta w = \gamma$,

(b) $r = A \to \beta$, *and*

(c) $i = |\alpha\beta|$.

Lemma 1 ([2])**.** *Let $G = (V, \Sigma, R, S)$ be a reduced context-free grammar. Then G is unambiguous if and only if every $\alpha \in V^*$, such that $S \Rightarrow_R^* \alpha$, has exactly one handle except S, which has none.*

In general, the identification of a handle in a string is not uniquely defined, which is not true for LR(0) grammars.

Definition 2. *Let $G = (V, \Sigma, R, S)$ be a reduced context-free grammar such that $S \Rightarrow_R^+ S$ is not possible in G. We say G is an $LR(0)$-grammar if, for each $w, w', x \in \Sigma^*$, $\eta, \alpha, \alpha', \beta, \beta' \in V^*$, and $A, A' \in V \setminus \Sigma$, if*

(a) $S \Rightarrow_R^* \alpha A w \Rightarrow_R \alpha \beta w = \eta w$, *and*

(b) $S \Rightarrow_R^* \alpha' A' x \Rightarrow_R \alpha' \beta' x = \eta w'$

imply $(A \to \beta, |\alpha\beta|) = (A' \to \beta', |\alpha'\beta'|)$

Note that as a consequence of the above definition we have that $A = A'$, $\beta = \beta'$, $\alpha = \alpha'$, $\gamma = \alpha\beta = \alpha'\beta'$ and $x = w'$. Thus, if G is an LR(0)-grammar, then the rightmost derivation of the word w by G and the left-right analysis is unique (deterministic). In this paper, we consider $LR(0)$ grammars rather as analytical grammars. A language generated by an LR(0)-grammar is called LR(0)-language.

Theorem 1 (LR(0)-language characterization theorem from [3])**.** *Let $L \subseteq \Sigma^*$. The following four statements are equivalent.*

(a) L is an LR(0) language.

(b) L is a deterministic context-free language and for all $x \in \Sigma^+$, $w, y \in \Sigma^*$, if $w \in L$, $wx \in L$, and $y \in L$, then $yx \in L$.

(c) There exists a deterministic pushdown automaton A with a single final state q_f and a pushdown symbol Z_f such that each word $w \in L$ is accepted by A by entering the state q_f with Z_f as the only symbol in its pushdown.

(d) There exist strict deterministic languages L_0 and L_1 such that $L = L_0 L_1^*$.

Note that strict deterministic languages are deterministic context-free languages recognizable by empty pushdown, or equivalently prefix-free deterministic context-free languages.

Let us recall semi-Dyck languages. Let $r \geq 1$, $\Sigma_r = \{a_1, \ldots, a_r\}$, $\bar{\Sigma}_r = \{\bar{a}_1, \ldots, \bar{a}_r\}$ and $\Sigma = \Sigma_r \cup \bar{\Sigma}_r$. The semi-Dyck language D_r is the language generated by the grammar $G_r = (V_r, \Sigma, P, S)$, where $V_r = \Sigma \cup \{S\}$, and R contains the following production rules:

$$S \to S a_i S \bar{a}_i S \mid \lambda, \text{ for each } i, 1 \leq i \leq r.$$

Informally, D_r is the set of all well-balanced parentheses containing r different pairs of brackets [3]. The semi-Dyck languages are examples of context-free languages that are not linear context-free languages. Additionally, they can be used to separate LR(0)-languages from the strict deterministic languages.

Theorem 2 ([3])**.** *Let $D_r \subset \Sigma^*$ be the semi-Dyck language for some $r \geq 1$. Then D_r is an $LR(0)$-language, but not a strict deterministic language.*

LR(0)-analyzer. If L is generated by an $LR(0)$-grammar G, then there exists an $LR(0)$-analyzer $P(G)$ for L with the following important properties (see [3]):

(a) For each word $w \in L$, there is exactly one rightmost derivation of w in G, which corresponds to the LR(0)-analysis of the word w by $P(G)$.

(b) Let u be a prefix of the content of the input tape $w = uv$, which has already been read by a computation of $P(G)$. Then there exists a suffix v' such that $uv' \in L$ iff $P(G)$ did not halt and not reject while reading u.

Let $L \subseteq \Sigma^*$ be a deterministic context-free language (DCFL), and $\$ \notin \Sigma$. Then, according to [3], there are $LR(0)$-grammar $G(0,\$)$ generating the language $L \cdot \{\$\}$, and a corresponding $LR(0)$-analyzer $P(G(0,\$))$ of $L \cdot \{\$\}$; $P(G(0,\$))$ is a deterministic push-down automaton with \$ as the rightmost symbol on its input-tape.

To stress its property, we will sometimes write LR(0,\$)-grammar instead of grammar $G(0,\$)$. We denote the set of LR(0)-grammars by *LRG* and the set of LR(0,\$)-grammars by *LRG*(\$).

Linear LRG. We say that a context-free grammar G is *linear* if the right-hand side of any rule from G contains at most one variable (nonterminal). We also consider linear LR(0)-grammars. We denote the linearity by the prefix *lin*-, and the class of linear LR(0)-grammars as lin-LRG.

Classes of languages. In what follows, $\mathscr{L}(A)$, where A is some class of grammars, denotes the class of languages generated by grammars from A. E.g., the class of languages generated by linear LR(0)-grammars is denoted by $\mathscr{L}(lin\text{-LRG})$.

3 Pumping by LR(0)-grammars

The first goal of our paper is to specify properties of deterministic (linear) context-free grammars that assure that the corresponding (linear) language is non-regular. In this section, we show such properties. We start with several definitions and notations.

Pumping notions. Let $G = (V, \Sigma, R, S)$ be an LR(0)-grammar generating (analyzing) the language $L = L(G)$, and $P(G)$ be the corresponding LR(0)-analyzer for G. Let $w \in L(G)$, $w = xu_1vu_2y$, where $x, u_1, v, u_2, y \in \Sigma^*$, $u_1u_2 \in \Sigma^+$ are given by the derivation tree T_w from Fig. 1. The proper sub-trees T_1 and T_2 of T_w are (computation) subtrees whose roots are labeled with the same nonterminal A, thus by replacing T_1 with T_2 properly inside of T_w, we again get a derivation tree, namely the derivation tree $T_{w(0)}$ for the word $w(0) = xvy \in L(G)$.

Analogously, replacing T_2 with a copy of T_1, we get the derivation tree $T_{w(2)}$ for a longer word $w(2) = xu_1^2vu_2^2y$. If we repeat i times such replacing of T_2 with T_1 we obtain the derivation tree $T_{w(i+1)}$ for the word $w(i+1) = xu_1^{i+1}vu_2^{i+1}y$.

Pumping tree, prefix, infix, pattern and reduction. Let $x, u_1, v, u_2, A, y, T_1$ be as on Fig. 1. Then we say that $P_p = xu_1vu_2$ is an (x, u_1, A, v, u_2)-*pumping prefix* by G, $P_{in} = xu_1vu_2y$ is an (x, u_1, A, v, u_2, y)-*pumping infix* by G, and that T_1 is a pumping tree of P_p. Recall that the LR(0)-analysis by $P(G)$ of the prefix P_p in any word of the form $P_p\gamma$ is the same, for any $\gamma \in \Sigma^*$. We say that (u_1, A, v, u_2) is a *pumping pattern by G (of P_p)*. Let us recall that for any $z \in \Sigma^*$, the LR(0)-analyzer $P(G))$ reduces xu_1vu_2z into xvz. We write $xu_1vu_2z \Leftarrow_{P(G)} xvz$, and say that $xu_1vu_2z \Leftarrow_{P(G)}$

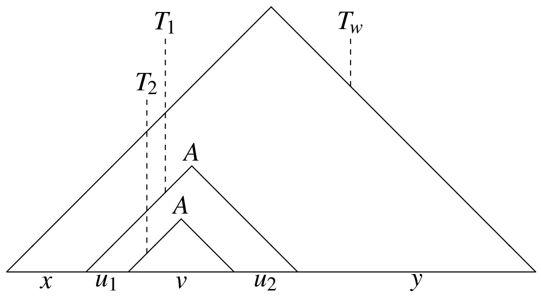

Figure 1: The structure of a derivation tree.

xvz is an (x,u_1,A,v,u_2)-pumping reduction by G. Note that the word xu_1vu_2z needs not be a word from $L(G)$. We will often say in the following that (x,u_1,A,v,u_2) is a pumping prefix and that (x,u_1,A,v,u_2,y) is a pumping infix.

Elementary infix etc. We say that an (x,u_1,A,v,u_2,y)-pumping infix by G is *elementary* if the (x,u_1,A,v,u_2)-pumping reduction is the only and, at the same time, the last pumping reduction by G that can be performed inside of the word xu_1vu_2y, i.e. xvy cannot be reduced by any pumping reduction by G at all. In this case, we say that (x,u_1,A,v,u_2) is an *elementary pumping prefix*, and that (u_1,A,v,u_2) is an *elementary pumping pattern* by G.

Let us note that the corresponding elementally pumping tree T_1 does contain an internal node with the same nonterminal as the root of T_1, and does not contain any other repetition of a nonterminal on any path from its root to a leaf. While the size (the number of terminal symbols) of a pumping infix by G is unbounded, there exists a constant $c > 0$ that depends on the number of terminals and nonterminals, and the length of rules of G such that any elementary pumping infix by G is of size at most c.

We can see the following obvious proposition that summarizes the (pumping) properties of LR(0)-grammars, which we will use in the following text. It is a direct consequence of the previous definitions and the properties of LR(0)-grammars and their analyzers summarized in [3], and inspired by [9].

Proposition 1. *Let $G = (V,\Sigma,R,S)$ be an LR(0)-grammar generating (analyzing) the language $L = L(G)$. Let a word $P_p = xu_1vu_2$ be an (x,u_1,A,v,u_2)-pumping prefix by G, and $xu_1u_1vu_2u_2 \Leftarrow_{P(G)} xu_1vu_2$ be an (xu_1,u_1,A,v,u_2)-pumping reduction by G. Then*

(a) Any $w \in L$ determines its derivation tree T_w by G unambiguously.

(b) P_p determines its pumping sub-tree unambiguously.

(c) $xu_1^m u_1vu_2u_2{}^n z \Leftarrow_{P(G)} xu_1^m vu_2{}^n z$ is an (xu_1^m,u_1,A,v,u_2)-pumping reduction by G for any $m,n > 0$, and any $z \in \Sigma^*$.

(d) $xu_1vu_2z \in L$ iff $xu_1^m vu_2^m z \in L$ for any $m \geq 0$, and any $z \in \Sigma^*$.

(e) An (x,u_1,A,v,u_2)-pumping prefix by G, and the pumping pattern (u_1,A,v,u_2) determine unambiguously a single pumping reduction.

(f) $xu_1^n vu_2^m z \in L$ iff $xu_1^{n+k} vu_2^{m+k} z \in L$ for any $m,n,k \geq 0$.

Assertion (c) is essential for our further considerations. It shows that, for a non-empty u_1, the distance of the place of pumping from the left end is not limited, and that the position of pumping is determined both by the pumping pattern of a pumping reduction and the pumping prefix of the pumping reduction. We use this property in the following definition formalizing a non-regular type of pumping.

Example 1. *Consider the non-regular context-free language $L_{ab} = \{a^n b^n \mid n \in N\}$, that can be generated by the grammar $G = (\{S,S_1,a,b\},\{a,b\},R,S))$, with the following set of rules R:*

$$\begin{array}{lcl} S & \rightarrow & S_1 \\ S_1 & \rightarrow & aS_1b \mid ab \end{array}$$

The grammar is reduced and unambiguous. Consider the sentence $\gamma = aaabbb$.

- *The handle of γ (cf. Definition 1) is the pair $(S_1 \rightarrow ab, 4)$, as*

$$S \Rightarrow_R^* aaS_1bb \Rightarrow_R aaabbb$$

and the division of γ into α,β,w is unique:

$$\gamma = \underbrace{aa}_{\alpha}\underbrace{ab}_{\beta}\underbrace{bb}_{w}$$

- *G is an LR(0)-grammar, moreover, linear as*

(a) $S \Rightarrow_R^* \alpha Aw \Rightarrow_R \alpha\beta w = \eta w$

(b) $S \Rightarrow_R^* \alpha' A' x \Rightarrow_R \alpha'\beta' x = \eta w'$

obviously implies $(A \rightarrow \beta, |\alpha\beta|) = (A' \rightarrow \beta', |\alpha'\beta'|)$

The pumping notions can be illustrated in Fig. 2 with a derivation tree for $w = xaabby \in L(G)$, where $x \in a^*, y \in b^*$.

*We can see, by taking $x = a$, that $P_p = aaabb$ is an (a,a,S_1,ab,b)-*pumping prefix *by G, and that T_1 is a pumping tree of P_p. Pumping pattern of P_p by G is (a,S_1,ab,b) and $aaabb \Leftarrow_{P(G)} aab$ is an (a,a,S_1,ab,b)-pumping reduction by G. Realize that a pumping reduction by G can be applied to any word $a^k b^m$, where $k,m > 1$, including the cases when $k \neq m$.*

Moreover, we can see that $(\lambda,a,S_1,ab,b,\lambda)$ is the single elementary pumping infix by G.

Definition 3. *Let $G = (V,\Sigma,R,S)$ be an LR(0)-grammar generating (analyzing) the language $L = L(G)$. Let*

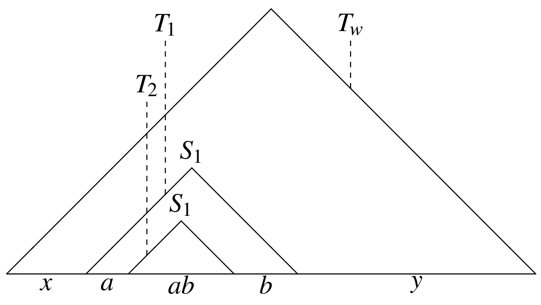

Figure 2: The structure of a derivation tree.

$xu_1vu_2 \Leftarrow_{P(G)} xv$ be an (x,u_1,A,v,u_2)-pumping reduction by G.

We say that A is a distinguishing nonterminal *for G, and $xu_1vu_2 \Leftarrow_{P(G)} xv$ is an (x,u_1,A,v,u_2)*-distinguishing reduction by *G if at least one of the following conditions is true:*

(I) for some $y \in \Sigma^*$ there is a $p > 0$ such that $xvy \in L$ iff $xvu_2^{pj}y \notin L$, for all $j > 0$;

(II) for some $y \in \Sigma^*$, there is a $p > 0$ such that $xvy \in L$ iff $xu_1^{pj}vy \notin L$, for all $j > 0$.

We say that the tuple (u_1,A,v,u_2) is a distinguishing pattern *for G.*

If G contains a distinguishing nonterminal (pattern) then we call G a distinguishing LR(0)-grammar (denoted dist-LRG*).*

For the sake of accuracy, we also say that A is an (x,u_1,A,v,u_2)-distinguishing nonterminal *for G.*

Example 1 (continued)**.** *According to case (I) of Definition 3 the nonterminal S_1 is a distinguishing nonterminal and aabb $\Leftarrow_{P(G)}$ ab is a (λ,a,S_1,ab,b)-distinguishing reduction for $y = \lambda$, and $p = 1$. Thus, (a,S_1,ab,b) is a distinguishing pattern for G.*

Let us recall that the class of LR(0)-languages is not closed under complement (see [2]), but DCFL is closed under complement.

Example 1 (continued)**.** *The language $L_{abc} = c^+ \cdot L_{ab}$ can be generated by LR(0) grammar $G_{abc} = (\{S,S_1,S_2,a,b,c\},\{a,b,c\},R,S)$, with the set of rules R :*

$$\begin{array}{rcl} S & \to & cS_2 \\ S_2 & \to & cS_2 \mid S_1 \\ S_1 & \to & aS_1b \mid ab \end{array}$$

Then, analogously to the situation in L_{ab}, nonterminal S_1 is distinguishing, and there is a (c,a,S_1,ab,b)-distinguishing reduction based on case (I) of Definition 3; here, we can take again $y = \lambda$, and $p = 1$. On the other hand, (c,S_2,ab,λ) is a pumping pattern by G_{abc}, which is not distinguishing for any (x,c,S_2,ab,λ)-reduction by G_{abc}; thus (c,S_2,ab,λ) is a non-distinguishing pumping pattern by G_{abc}.

Lemma 2. *Let $G = (V,\Sigma,R,S)$ be a dist-LR(0)-grammar. Let (u_1,A,v,u_2) be a distinguishing pattern by G. Then $u_1,u_2 \in \Sigma^+$.*

Proof. Assume, for example, that condition (I) from Definition 3 holds. It is then clear that $u_2 \neq \lambda$, otherwise $xvu_2^{pj}y = xvy$ for all $p, j > 0$.

It is also easy to see that $u_1 \neq \lambda$. If $u_1 = \lambda$, we get from Proposition 1 that $xvu_2^{pm}y \in L$ and we have $xvu_2^{pm}u_2^py \notin L$ for all $m \geq 0$ and some $p > 0$. The first fact yields $xvu_2^py \in L$ for $m = 1$, the second fact yields $xvu_2^py \notin L$ for $m = 0$. This is a contradiction. □

The existence of a distinguishing nonterminal serves as a sufficient condition for non-regularity.

Theorem 3. *Let $L = L(G)$ be a language accepted by a dist-LR(0)-grammar $G = (V,\Sigma,R,S)$. Then L is a non-regular language.*

Proof. Assume, for a contradiction, that $G = (V,\Sigma,R,S)$ is a dist-LR(0)-grammar generating a regular language $L = L(G)$ and B be an (x,u_1,B,v,u_2)-distinguishing nonterminal for G.

As L is regular, there exists a deterministic finite automaton $A = (Q,\Sigma,\delta,q_0,F)$ with $n_A = |Q|$ states accepting the language $L = L(A)$. The proof then follows by the case analysis based on properties of the (x,u_1,B,v,u_2)-distinguishing pattern. Since there is an analogy of the case analysis, we will prove the theorem only by the assumption that condition (I) from Definition 3 is fulfilled.

Suppose that for some $y \in \Sigma^*$ and $p > 0$ it holds $xvy \in L$, and, for all $j > 0$, $xvu_2^{p\cdot j}y \notin L$.

Proposition 1 implies that $xu_1^{pm}vu_2^{pm}y \in L$ for all $m \geq 0$ and $xu_1^{pm}vu_2^{p(m+j)}y \notin L$ for all $m \geq 0$, $j > 0$, and Lemma 2 implies that $u_1,u_2 \in \Sigma^+$. Let us inspect states reachable by the automaton A when reading words of the form $xu_1^{pm}vu_2^{pk}$, for $m > n_A$, $k \geq 0$. Since A has n_A states, the pigeonhole principle yields that there are integers r,s, $1 \leq r < s \leq n_A + 1$ and a state q of A such that

$$q = \delta(q_0, xu_1^{pm}vu_2^{pr}) = \delta(q_0, xu_1^{pm}vu_2^{ps}).$$

As $xu_1^{pm}vu_2^{pm}y \in L$, both words

$$xu_1^{pm}vu_2^{pm}y = xu_1^{pm}vu_2^{pr}u_2^{p(m-r)}y$$

and

$$xu_1^{pm}vu_2^{ps}u_2^{p(m-r)}y = xu_1^{pm}vu_2^{pm}u_2^{p(s-r)}y$$

are accepted by A. Then, for $j = s - r$ we have a contradiction with the fact that $xu_1^{pm}vu_2^{pm}u_2^{p\cdot j}y \notin L$, for all $m \geq 0$, $j > 0$.

Next, suppose that for some $y \in \Sigma^*$ and $p > 0$ it holds $xvy \notin L$, and $xvu_2^{p\cdot j}y \in L$, for all $j > 0$.

Using the same reasoning as above, we derive that A rejects $xu_1^{pm}vu_2^{pm}y = xu_1^{pm}vu_2^{pr}u_2^{p(m-r)}y$ as well as $xu_1^{pm}vu_2^{ps}u_2^{p(m-r)}y = xu_1^{pm}vu_2^{pm}u_2^{p(s-r)}y$, which is again a contradiction. □

The previous proof yields the following corollary.

Corollary 1. *Let $G = (V,\Sigma,R,S)$ be a dist-LR(0)-grammar. Let (x,u_1,A,v,u_2) be a distinguishing prefix by G, and P_{in} be an (x,u_1,A,v,u_2,y)-infix by G. Then the language $L(G) \cap \{xu_1^n vu_2^m y \mid n \geq 0, m \geq 0\}$ is not a regular language.*

We will use the following definition to stress the ability of LR(0)-grammars to define DCFL.

Definition 4. *Let G be an $LRG(0,\$)$ grammar which generates $L(G) = L\{\$\}$, where $L \subseteq \Sigma^*$. We say that G is an LRG(1)-grammar which defines L.*
We take

$$\begin{aligned} \mathscr{L}(LRG(1)) &= \{L \mid L\{\$\} \in LRG(\$)\}, \\ \mathscr{L}(dist\text{-}LRG(1)) &= \{L \mid L\{\$\} \in dist\text{-}LRG(\$)\}. \end{aligned}$$

In what follows, the sign $\subset$ means a proper subset.

Theorem 4. *It holds the following:*

$$\begin{array}{ll} (a) & \mathscr{L}(dist\text{-}LRG) \subset \mathscr{L}(LRG), \\ (b) & \mathscr{L}(lin\text{-}dist\text{-}LRG) \subset \mathscr{L}(lin\text{-}LRG), \\ (c) & \mathscr{L}(dist\text{-}LRG(1)) \subset \mathscr{L}(LRG(1)), \\ (d) & \mathscr{L}(lin\text{-}dist\text{-}LRG(1)) \subset \mathscr{L}(lin\text{-}LRG(1)), \\ (e) & \mathscr{L}(LRG(1)) = DCFL. \end{array}$$

Proof. All inclusions (a)–(d) follow from the fact that, according to Theorem 3, all languages generated (accepted) by distinguishing LR(0)-grammars are non-regular, while both $\mathscr{L}$(lin-LRG) and $\mathscr{L}$(lin-LRG(1)) contain all regular languages. The equality (e) is just the fact that for any deterministic context-free language $L \subseteq \Sigma^*$, where $\$ \notin \Sigma$, the language $L\{\$\}$ is accepted by an LR(0)-grammar [3]. □

4 Some total separations inside of LR(0)-languages

While the size of distinguishing pumping patterns is unbounded, the size of elementary pumping patterns for a given grammar is limited. Therefore, below we introduce a condition that implies non-regularity of a language generated by an LR(0)-grammar based on elementary pumping patterns only.

Definition 5. *Let $G = (V,\Sigma,R,S)$ be an LR(0)-grammar and $\alpha = (x,u_1,A,v,u_2,y)$ be an elementary pumping infix by G.*

We say that α is regular *if the language $L(G,\alpha) = L(G) \cap \{xu_1^n vu_2^m y \mid n \geq 0, m \geq 0\}$ is a regular language.*

We say that α is non-regular *if the language $L(G,\alpha) = L(G) \cap \{xu_1^n vu_2^m y \mid n \geq 0, m \geq 0\}$ is a non-regular language.*

We say that G is an elementary-regular grammar *if all elementary pumping infixes by G are regular. We denote the class of elementary-regular grammars by the prefix er-.*

We say that G is an elementary-non-regular grammar *if there exists an elementary non-regular pumping infix by G. We denote the class of elementary non-regular grammars by the prefix enr-.*

We say that a language L is an elementary-non-regular language *if an elementary-non-regular LR(0)-grammar G exists such that $L(G) = L$. The property being elementary-non-regular we mark by the prefix enr- and the class of elementary-non-regular grammars is denoted as enr-LRG. Similarly, the prefix er- will denote the property being elementary-regular.*

We say that a language $L \in \mathscr{L}(LR(0))$ is an elementary-regular language *if there is not any elementary-non-regular LR(0)-grammar G such that $L(G) = L$. We denote the class of elementary-regular languages as $\mathscr{L}$(er-LGR).*

The next corollary follows from the previous definition and Corollary 1.

Corollary 2. *Let $G = (V,\Sigma,R,S)$ be a dist-LR(0)-grammar. Let (x,u_1,A,v,u_2) be a distinguishing elementary prefix by G, and $\alpha = (x,u_1,A,v,u_2,y)$ be a pumping infix by G. Then the language $L(G)$ is an elementary-non-regular language.*

Theorem 5. *Let L be an elementary-non-regular language. Then L is not a regular language.*

Proof. Let G be an elementary-non-regular LR(0)-grammar such that $L(G) = L$. Let $\alpha = (x,u_1,A,v,u_2,y)$ be a non-regular pumping infix by G. The facts that

$$L' = \{xu_1^n vu_2^m y \mid n \geq 0, m \geq 0\}$$

is a regular language and $L \cap L'$ is not a regular language imply that L is not regular, because regular languages are closed under the intersection operation. □

For the opposite direction that each non-regular language L generated by an LR(0)-grammar is elementary-non-regular, we still do not have proof.

The next corollary presents our current results about total separations achieved by LR(0)-grammars. We consider these results important from the point of view of our motivations mentioned in the introduction. The corollary is mainly a consequence of Definition 5.

Corollary 3. *The following equations hold:*

$$\begin{array}{ll} (a) & \mathscr{L}(er\text{-}LRG) \cup \mathscr{L}(enr\text{-}LRG) = \mathscr{L}(LRG), \\ (b) & \mathscr{L}(er\text{-}lin\text{-}LRG) \cup \mathscr{L}(enr\text{-}lin\text{-}LRG) = \\ & \mathscr{L}(lin\text{-}LRG), \\ (c) & \mathscr{L}(er\text{-}LRG) \cap \mathscr{L}(enr\text{-}LRG) = \emptyset, \\ (d) & \mathscr{L}(er\text{-}lin\text{-}LRG) \cap \mathscr{L}(enr\text{-}lin\text{-}LRG) = \emptyset. \end{array}$$

What remains is a further study of relations between the class of regular languages and the classes of languages generated by LR(0)-grammars that are not distinguishing or elementary-regular. We believe that the following conjectures hold.

Conjecture 1. *It holds the following*

$$\begin{array}{ll} (a) & \mathscr{L}(er\text{-}LRG) = Reg. \\ (b) & \mathscr{L}(enr\text{-}LRG) = \mathscr{L}(LRG) - Reg. \end{array}$$

Above, in Corollary 2, we have seen that if an elementary pumping prefix is distinguishing for an LR(0)-grammar G, we can obtain a non-regular pumping infix for G. We conjecture that also the opposite direction holds.

Conjecture 2. *Let $G = (V, \Sigma, R, S)$ be an LR(0)-grammar and (x, u_1, A, v, u_2, y) be a pumping infix by G. If (x, u_1, A, v, u_2, y) is non-regular pumping infix by G then (u_1, A, v, u_2) is a distinguishing pattern for G.*

5 Conclusion and Future Work

It remains yet some work to do in order to fulfill our plans mentioned in the introduction and show our conjectures. It also remains to show the relation between elementary non-regular pumping patterns and distinguishing pumping patterns, and between elementary regular pumping patterns and non-distinguishing pumping patterns. We believe that we will make that in the near future.

We also plan to show that the distinguishing pumping and the non-distinguishing pumping by LR(0)-grammars can be characterized by pumping patterns of limited size (e.g., elementary pumping patterns) depending only on corresponding LR(0)-grammars. Such patterns could help with proving the decidability of the problem whether an LR(0)-grammar is distinguishing or not. After that, we plan to present the mentioned transformation to restarting automata controlled by LR(0)-grammars. This step will open a broad field on studying the descriptional complexity of DCFL and its subclasses. One interesting type of descriptional complexity will be the degree of non-regularity of DCFL and some of its subclasses. Similarly, a degree of regularity of a deterministic context-free language can be measured.

Finally, let us note that total separations of the presented type have essential importance for computational and comparative linguistic, and for expressing of properties of programming languages. We can, in this way, formally express, e.g., the (non-)existence of obligatory adjuncts in a natural language (or in a sentence of it) or the use of different types of parenthesis in a programming language.

References

[1] Bejček, E., Panevová, J., Popelka, J., Straňák, P., Sevčíková, M., Štěpánek, J., Žabokrtský, Z.: Prague dependency treebank 2.5 – a revisited version of PDT 2.0. In: Kay, M., Boitet, C. (eds.) COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8–15 December 2012, Mumbai, India. pp. 231–246. Indian Institute of Technology Bombay (2012), `https://www.aclweb.org/anthology/C12-1015/`

[2] Geller, M.M., Harrison, M.A.: On LR*(k)* grammars and languages. Theor. Comput. Sci. **4**(3), 245–276 (1977). https://doi.org/10.1016/0304-3975(77)90013-5

[3] Harrison, M.A.: Introduction to formal language theory. Series in computer science, Addison-Wesley Longman Publishing Co., Inc. (1978)

[4] Jančar, P., Mráz, F., Plátek, M., Vogel, J.: Restarting automata. In: Reichel, H. (ed.) Fundamentals of Computation Theory, 10th International Symposium, FCT '95, Dresden, Germany, August 22–25, 1995, Proceedings. Lecture Notes in Computer Science, vol. 965, pp. 283–292. Springer (1995). https://doi.org/10.1007/3-540-60249-6_60

[5] Lopatková, M., Plátek, M., Sgall, P.: Towards a formal model for functional generative description: Analysis by reduction and restarting automata. Prague Bull. Math. Linguistics **87**, 7–26 (2007), `http://ufal.mff.cuni.cz/pbml/87/lopatkova-et-al.pdf`

[6] Mráz, F., Pardubská, D., Plátek, M., Šíma, J.: Pumping deterministic monotone restarting automata and DCFL. In: Holena, M., Horváth, T., Kelemenová, A., Mráz, F., Pardubská, D., Plátek, M., Sosík, P. (eds.) Proceedings of the 20th Conference Information Technologies - Applications and Theory (ITAT 2020), Hotel Tyrapol, Oravská Lesná, Slovakia, September 18-22, 2020. CEUR Workshop Proceedings, vol. 2718, pp. 51–58. CEUR-WS.org (2020), `http://ceur-ws.org/Vol-2718/paper13.pdf`

[7] Sgall, P., Goralčíková, A., Nebeský, L., Hajičová, E.: A functional approach to syntax in generative description of language. American Elsevier Publishing Company, Inc., New York (1969)

[8] Šíma, J., Plátek, M.: One analog neuron cannot recognize deterministic context-free languages. In: Gedeon, T., Wong, K.W., Lee, M. (eds.) Neural Information Processing – 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11955, pp. 77–89. Springer (2019). https://doi.org/10.1007/978-3-030-36718-3_7

[9] Stearns, R.E.: A regularity test for pushdown machines. Inf. Control. **11**(3), 323–340 (1967). https://doi.org/10.1016/S0019-9958(67)90591-8

[10] Valiant, L.G.: Regularity and related problems for deterministic pushdown automata. Journal of the ACM **22**, 1–10 (1975)