☺ → ☺☺
AG Automaten und Formale Sprachen

Preprint AFL-2011-06
Otto-von-Guericke-Universität Magdeburg, Germany

k-Local Internal Contextual Grammars

Radu Gramatovici[(B)] Florin Manea[(A,B)]

[(A)]Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik
PSF 4120, D-39016 Magdeburg, Germany
manea@iws.cs.uni-magdeburg.de

[(B)]Faculty of Mathematics and Computer Science, University of Bucharest,
Str. Academiei 14, RO-010014 Bucharest, Romania,
radu.gramatovici@fmi.unibuc.ro

Abstract

In this paper we propose a generalization of the local internal contextual grammars, introduced by Ilie in 1997, namely the k-local internal contextual grammars. These grammars are, in fact, classical internal contextual grammars, but their only permitted derivations are those that can be described in a restricted manner (that depends on the number k). Within this formalism we define different classes of languages, for which we prove a series generative properties. Also, we propose an efficient algorithmic solution for the membership problem for k-local contextual grammars with polynomial choice. This seems interesting to us, as it shows how the restrictions on the descriptions of the derivations reflect in the possibility of designing a polynomial algorithm for the acceptance of the languages generated by internal contextual grammars using such derivations.

1. Introduction

Contextual grammars were introduced in [10] by Solomon Marcus as an attempt to transform in generative devices some procedures developed within the framework of analytical models. Although a series of linguistic motivations and applications of the contextual grammars formalism are presented in [13, Chapter 5]), the problem of finding classes of Marcus contextual grammars suitable for natural language processing (discussed in [12, Chapter 7]) is still on going, following the introduction and study of contextual tree-like structures [5], parsing algorithms [2, 1, 4, 3, 6, 7] or language properties, such as ambiguity and syntactic complexity, see, e.g., [12] and the references therein, [9, 11].

To this end, one of the most important tasks seems to be the design of efficient acceptors/parsers for the analysis of contextual languages. Such algorithms are extensively used in linguistic applications and their primary condition is to be time-efficient. From the theoretical point of view, this means to have a polynomial (of a reasonable degree) time complexity.

The first approaches to contextual languages recognition/parsing were developed by Harbusch, in a series of papers [6, 7]. However, the parser for internal contextual grammars with context-free selectors reported in these papers is not completely defined and it was not proved

to be correct or to have polynomial complexity. Local internal contextual grammars were introduced by Ilie in [8] as a formalism generating polynomially parsable languages. “Local", in the context of internal contextual grammars, stands for the localization of the derivation of words, with respect to the position of the previously inserted context. A similar condition was defined by Gramatovici in [2], under the name of “derivation that preserves the selectors".

In [8], Ilie showed that the class of languages generated by local internal contextual grammars with regular choice is polynomially parsable. However, this result was not effective: it was obtained indirectly, and no practical algorithm was proposed for the recognition of local internal contextual grammars with regular choice. The result of Ilie was improved in [4] in two directions: first the result was extended for local internal contextual grammars with context-free choice and, second, it was obtained by effectively constructing a polynomial parser that recognizes any word of length N in $\mathcal{O}(N^7)$ time; the time complexity of this algorithm was further improved to $\mathcal{O}(N^4)$ in [3].

In this paper, we generalize local contextual grammars to k-local contextual grammars. Similar to the case of the local contextual grammars, the derivation is localized, but a newly inserted context has to be adjacent to at least to one of the previously inserted k contexts. The localization condition previously defined in [8] is obtained in our model by taking $k = 1$. As an initial result, we show that this generalization is not trivial since k-local internal contextual grammars generate different classes of languages for different values of k. Further, a series of language theoretic results are given for k-local internal contextual grammars, showing how the descriptive restrictions imposed on the derivations reflect in the generative power of the model. Finally, we show that the languages generated by k-local internal contextual grammars, whit polynomial choice, can be recognized in polynomial time.

2. Preliminaries

We assume that the reader is familiar with the basic concepts of formal languages theory (see, e.g., [13]). In this section we only give the definitions, notations and some examples regarding the generative models we use; for further details concerning contextual grammars the reader is referred to the monograph [12].

For an alphabet V, we denote by V^* and V^+ the set of all words and of all non-empty words over V, respectively. The empty word is denoted by λ. Given a word $w \in V^*$ and a letter $a \in V$, we denote by $|w|$ and $|w|_a$ the length of w and the number of occurrences of a in w, respectively. For $w \in V^*$ and V' an alphabet, we obtain $w|_{V'}$ from w by erasing all the symbols not contained in V'; $w|_{V'}$ is called the projection of w to V, .

By *FIN*, *REG*, *CF*, *CS* and *P* we denote the classes of finite, regular, context-free, context-sensitive and deterministic polynomial languages, respectively. Let $\mathcal{F}$ be one of these classes.

An *internal contextual grammar with selection* from $\mathcal{F}$ (or, simpler, with $\mathcal{F}$-choice) is a construct $G = (V, A, (L_1, c_1), \ldots, (L_n, c_n))$, where:

- V is an alphabet,
- A is a finite language over V, called the set of *axioms*,
- for $i \in \{1, \ldots, n\}$, (L_i, c_i) is a *contextual rule* with the *selection language* L_i, where $L_i \subseteq V^*$ and $L_i \in \mathcal{F}$, and the *context* $c_i = (u_i, v_i) \in V^* \times V^*$.

When the class $\mathcal{F}$ is not explicitly mentioned, it is not important or follows from the context.

The derivation in the contextual grammar G is defined as $x \Rightarrow y$ if and only if there is a contextual rule $(L_i,(u_i,v_i))$ such that $x = x_1x_2x_3$ with $x_2 \in L_i$ and $y = x_1u_ix_2v_ix_3$. Let $\overset{*}{\Rightarrow}$ denote the reflexive and transitive closure of $\Rightarrow$. The language generated by G is $L(G) = \{w \in V^* \mid \exists \alpha \in A, \alpha \overset{*}{\Rightarrow} w\}$.

We say that a word w can be derived by a sequence of contexts $(w_1,t_1), (w_2,t_2),\ldots,(w_p,t_p)$ in the contextual grammar G, if there exists the derivation:

$$\alpha \Rightarrow x_{1,1}w_1x_{2,1}t_1x_{3,1} \Rightarrow x_{1,2}w_2x_{2,2}t_2x_{3,2} \Rightarrow \ldots \Rightarrow x_{1,p}w_px_{2,p}t_px_{3,p} = w,$$

such that $\alpha = x_{1,1}x_{2,1}x_{3,1}$ is an axiom from A, $(L_{r_i},(w_i,t_i))$ is a contextual rule of the grammar G, $x_{1,i}w_ix_{2,i}t_ix_{3,i} = x_{1,i+1}x_{2,i+1}x_{3,i+1}$ and $x_{2,i} \in L_{r_i}$ for all $1 \leq i \leq p-1$. The sequence of contexts $(w_1,t_1),(w_2,t_2),\ldots,(w_p,t_p)$ is called the *control sequence* of the above derivation.

It is not hard to see that the derivation of a word is not fully characterized by its control sequence, as one could insert the contexts in various positions in the words obtained during such a derivation. An accurate characterization of a derivation as the above is given by the *description* of the derivation:

$$\alpha; x_{1,1}(w_1)x_{2,1}(t_1)x_{3,1}; x_{1,2}(w_2)x_{2,2}(t_2)x_{3,2}; \ldots; x_{1,p}(w_p)x_{2,p}(t_p)x_{3,p}.$$

For the purposes of this paper, we introduce the marked description of a derivation. We define the infinite set of symbols $\{(_i,)_i,[_i,]_i \mid i \in I\!N\}$, and assume that none of its symbols belongs to V. A *marked description* of the above derivation of w in the grammar G is a sequence:

$$\alpha; y_{1,1}(_1w_1)_1y_{2,1}[_1t_1]_1y_{3,1}; y_{1,2}(_2w_2)_2y_{2,2}[_2t_2]_2y_{3,2}; \ldots; y_{1,p}(_pw_p)_py_{2,p}[_pt_p]_py_{3,p},$$

where $y_{j,i}|_V = x_{j,i}$, for $1 \leq i \leq p$, $\alpha = y_{1,1}y_{2,1}y_{3,1}$, and $y_{1,i}(_iw_i)_iy_{2,i}[_it_i]_iy_{3,i} = y_{1,i+1}y_{2,i+1}y_{3,i+1}$ for any $1 \leq j \leq 3$ and $1 \leq i \leq p-1$. The *marked descriptor* of w with respect to the given derivation is the word $\beta = y_{1,p}(_pw_p)_py_{2,p}[_pt_p]_py_{3,p}$.

A marked descriptor β of a word w, with respect to one of its derivations in G, characterizes completely that derivation. Indeed, one can easily obtain from β the marked description of the respective derivation of w, by the following algorithm.

Algorithm D. *Input: the word β.*

1. **Start** *from the marked descriptor of w.*
2. **While** *the current descriptor contains parentheses, delete the symbols $(_j$, $)_j$, $[_j$, $]_j$ for the maximum j that appears the sequence, together with the symbols from V that are between $(_j$ and $)_j$ but are not between other $(_i$ and $)_i$ with $1 \leq i < j$, and the symbols from V that are between $[_j$ and $]_j$ but are not between other $[_i$ and $]_i$ with $1 \leq i < j$.*
3. **Stop** *when the sequence contains only symbols from V.*

Example 2.1 *Note that a word w may have several different marked descriptors, with respect to the different derivations of that word in an internal contextual grammar with choice, even if we start all the derivations with the same axiom. Indeed, consider the grammar $G = (\{a,b\}, \{a^2b^3\}, (a^*b^*,(a^3,b^3)))$. Then the words:*

1. $a(_1aa(_2aaa)_2a)_1abb[_1bbb[_2bbb]_2]_1b$,
2. $(_1aaa)_1a(_2aaa)_2ab[_1bbb]_1bb[_2bbb]_2$,
3. $(_1aaa(_2aaa)_2)_1aabb[_1bbb]_1[_2bbb]_2b$, *and*

4. $aa({}_1aaa({}_2aaa)_2[_2bbb]_2)_1bbb[_1bbb]_1$

are all marked descriptors of the same word a^8b^9, with respect to four different two-steps derivations of this word from the axiom a^2b^3.

The main notion that we introduce in this paper is defined in the following:

Definition 2.2 *Let k be a positive integer and $G = (V, A, (L_1, c_1), \ldots, (L_n, c_n))$ be an internal contextual grammar with choice. Consider the following derivation of w from α:*

$$\alpha \Rightarrow x_{1,1}w_1x_{2,1}t_1x_{3,1} \Rightarrow x_{1,2}w_2x_{2,2}t_2x_{3,2} \Rightarrow \ldots \Rightarrow x_{1,p}w_px_{2,p}t_px_{3,p} = w,$$

where $\alpha = x_{1,1}x_{2,1}x_{3,1}$, $(w_i, t_i) = c_{r_i}$, with $r_i \in \{1, \ldots, n\}$, $x_{1,i}w_ix_{2,i}t_ix_{3,i} = x_{1,i+1}x_{2,i+1}x_{3,i+1}$ and $x_{2,i} \in L_{r_i}$, for all $1 \leq i \leq p-1$. This derivation is called k-local *if and only if it has the marked description:*

$$\alpha; y_{1,1}({}_1w_1)_1y_{2,1}[_1t_1]_1y_{3,1}; y_{1,2}({}_2w_2)_2y_{2,2}[_2t_2]_2y_{3,2}; \ldots; y_{1,p}({}_pw_p)_py_{2,p}[_pt_p]_py_{3,p},$$

that verifies:

- *for any q, with $k < q \leq p$, there exists r, with $q - k \leq r < q$, such that $({}_r$ is in $y_{1,q}$, $)_r$ and $[_r$ are in $y_{2,q}$, and $]_r$ is in $y_{3,q}$.*

- *for any q and r, with $1 \leq r < q \leq p$, if $({}_r$ is in $y_{1,q}$ and $)_r$ is in $y_{2,q}y_{3,q}$, then $)_r$ and $[_r$ are in $y_{2,q}$ and $]_r$ is in $y_{3,q}$; if $[_r$ is in $y_{1,q}y_{2,q}$ and $]_r$ is in $y_{3,q}$, then $({}_r$ is in $y_{1,q}$, $)_r$ is in $y_{2,q}$ and $[_r$ is in $y_{2,q}$.*

A k-local derivation of w from α will be denoted by $\alpha \overset{*}{\Rightarrow}_{k-loc} w$.

The language generated by the grammar G by k-local derivation is

$$L_{k-loc}(G) = \{w \in V^* \mid \exists \alpha \in A, \alpha \overset{*}{\Rightarrow}_{k-loc} w\}.$$

Intuitively, in a k-local derivation only the first k derivation steps can be performed arbitrarily to the previous steps, while all the other contexts are inserted such that they are adjacent to one of the previously k inserted contexts. Moreover, if the left side of a context is inserted adjacently to the left side of some previously inserted context, then also the right side of the new context is inserted adjacently to the right side of the previously inserted context. For instance, in Example 2.1 the first marked descriptor corresponds to a k-local derivation with $k \geq 1$, the second corresponds to a k-local derivation with $k \geq 2$, while the others do not correspond to k-local derivations for any $k \geq 1$.

For simplicity, we may call an internal contextual grammar k-local internal contextual grammar when we are interested only in the k-local derivation of that grammar.

We denote by $ICC(\mathcal{F})$ the class of languages generated by internal contextual grammars with $\mathcal{F}$-choice by normal derivation, by $ICC_{k-loc}(\mathcal{F})$ the class of languages generated by internal contextual grammars with $\mathcal{F}$-choice by k-local derivation, and by $ICC_{\infty-loc}(\mathcal{F})$ the union of the all these classes $ICC_{\infty-loc}(\mathcal{F}) = \bigcup_{k\geq 1} ICC_{k-loc}(\mathcal{F})$.

Since the definition of 1-local derivation (in our setting) is clearly equivalent to the definition of local derivation, introduced by Ilie in [8], we write $ICC_{loc}(\mathcal{F})$ instead of $ICC_{1-loc}(\mathcal{F})$. .

3. Language theoretic properties

We begin this section with several examples that will become useful in the sequel.

Example 3.1 *Let k be an positive integer, and $G_k = (\{a, b_1, \ldots, b_k\}, \{b_1 \ldots b_k\}, (\{b_1, \ldots, b_k\}, (\epsilon, a)))$ be an internal contextual grammar with FIN-choice. First, the language generated by G_k by normal derivation is the regular language L_k described by the regular expression $b_1a^*b_2a^* \ldots b_ka^*$. Second, it is not hard to see that the language generated by G_k by ℓ-local derivation for $\ell \geq k$, equals L_k. Finally, the language generated by G_k by m-local derivation for $m < k$, denoted $L_{k,m}$, is different from L_k. Indeed,*

$$L_{k,m} = \{b_1a^{n_1} \ldots b_ka^{n_k} \mid n_1, \ldots, n_k \in IN, \text{ and } \exists \ell, \text{ such that } k-m \leq \ell \leq k, \text{ and } \exists r_1, r_2, \ldots, r_\ell \in \{1, \ldots, k\}, \text{ with } n_{r_1} = n_{r_2} = \ldots = n_{r_\ell} = 0\}.$$

That is, in any word from $L_{k,m}$, at most m groups of a symbols are non-empty.

Example 3.2 *Let k be a positive integer, and let $G'_k = (\{a, b_1, b_2, \ldots, b_{2k-1}\}, \{b_1b_2 \ldots b_{2k-1}\}, (\{b_1, b_3 \ldots, b_{2k-1}\}, (a, a)))$ be an internal contextual grammar with FIN-choice. The language generated by G'_k by normal derivation is, clearly, the context-free language:*

$$L'_k = L(G'_k) = \{a^{n_1}b_1a^{n_1}b_2a^{n_2}b_3a^{n_2} \ldots a^{n_k}b_{2k-1}a^{n_k} \mid n_1, \ldots, n_k \geq 0\}.$$

Further, it is not hard to see that all the languages $L'_{k,\ell}$ generated by G'_k by ℓ-local derivation for $\ell \geq k$, are equal to L'_k. On the other hand, the languages $L'_{k,m}$ generated by G'_k by m-local derivation for $m < k$, are different from L'_k. Indeed,

$$L'_{k,m} = \{\ a^{n_1}b_1a^{n_1}b_2a^{n_2}b_3a^{n_2} \ldots a^{n_k}b_{2k-1}a^{n_k} \mid n_1, \ldots, n_k \geq 0, \text{ and } \exists \ell, \text{ such that } k-m \leq \ell \leq k, \text{ and } \exists r_1, r_2, \ldots, r_\ell, \text{ such that } 1 \leq r_1 < r_2 < \ldots < r_\ell \leq k \text{ and } n_{r_1} = n_{r_2} = \ldots = n_{r_\ell} = 0\}.$$

We will see in the following that L'_k cannot be generated by any internal contextual grammar with $\mathcal{F}$-choice by m-local derivation for $m < k$ and $\mathcal{F}$ an arbitrary class of languages.

Example 3.3 *Let $G'' = (\{a, b, c\}, \{abc\}, (\{ab^+\}, (a, bc)))$ be an internal contextual grammar with regular choice. For all $k \geq 1$, the language generated by the grammar G by k-local derivation, denoted L''_k, is non-context-free language. Indeed, it is not hard to see that $L''_k \cap a^*b^*c^* = \{a^nb^nc^n \mid n \geq 1\}$, which is not a context-free language. Thus, L''_k is also a non-context-free.*

Remark 1 *If $x_{1,1}x_{2,1}x_{3,1} \Rightarrow x_{1,1}w_1x_{2,1}t_1x_{3,1} \Rightarrow \ldots \Rightarrow x_{1,p}w_px_{2,p}t_px_{3,p}$ is a k-local derivation in an internal contextual grammar G and t is a positive integer with $1 < t \leq p$, then $x_{1,t}x_{2,t}x_{3,t} \Rightarrow x_{1,t}w_tx_{2,t}t_tx_{3,t} \Rightarrow \ldots \Rightarrow x_{1,p}w_px_{2,p}t_px_{3,p}$ is also a k-local derivation in G.*

Proof.
Let $y_{1,1}y_{2,1}y_{3,1}$, $y_{1,1}({}_1w_1)_1y_{2,1}[{}_1t_1]_1y_{3,1}$, $y_{1,2}({}_2w_2)_2y_{2,2}[{}_2t_2]_2y_{3,2}$, $\ldots$, $y_{1,p}({}_pw_p)_py_{2,p}[{}_pt_p]_py_{3,p}$, be a marked description of the derivation $x_{1,1}x_{2,1}x_{3,1} \overset{*}{\Rightarrow} x_{1,p}w_px_{2,p}t_px_{3,p}$. We define the alphabet $V' = V \cup \{(_j,)_j, [_j,]_j | t \leq j \leq p\}$ and the words $\alpha_i = y_{1,i}({}_iw_i)_iy_{2,i}[{}_it_i]_iy_{3,i}|_{V'}$ for all i such that $t \leq i \leq p$. Then, for all i such that $t \leq i \leq p$, define the words β_{i-t+1} as α_i in which we replace the index j of all parentheses $(_j$, $)_j$, $[_j$, $]_j$ by $j - t + 1$. It is rather

clear that the sequence $x_{1,t}x_{2,t}x_{3,t}, \beta_1, \ldots, \beta_{p-t+1}$ is a marked description of the derivation $x_{1,r}x_{2,r}x_{3,r} \stackrel{*}{\Rightarrow} x_{1,p}w_px_{2,p}t_px_{3,p}$.

According to *Definition 2.2*, it follows that the following holds for the marked description $y_{1,1}y_{2,1}y_{3,1}$, $y_{1,1}({}_1w_1)_1y_{2,1}[{}_1t_1]_1y_{3,1}$, $y_{1,2}({}_2w_2)_2y_{2,2}[{}_2t_2]_2y_{3,2}$, $\ldots$, $y_{1,p}({}_pw_p)_py_{2,p}[{}_pt_p]_py_{3,p}$: for any q, with $k < q \leq p$, there exists r, with $q-k \leq r < q$, such that $({}_r$ belongs to $y_{1,q}$, $)_r$ and $[_r$ belong to $y_{2,q}$ and $]_r$ belongs to $y_{3,q}$.

From the way in which it was constructed and from the fact that the above property relies only on the previously inserted k contexts, it results that the same property is true for the marked description $x_{1,t}x_{2,t}x_{3,t}, \beta_1, \ldots, \beta_{p-t+1}$. Clearly, the second condition from the the definition of k-local derivation holds canonically for this marked description. Thus, the derivation $x_{1,r}x_{2,r}x_{3,r} \stackrel{*}{\Rightarrow} x_{1,p}w_px_{2,p}t_px_{3,p}$, given by the above marked description, is also a k-local. □ □

The following result is immediate:

Proposition 3.4 *For any two classes of languages $\mathcal{F}$ and $\mathcal{F}'$, if $\mathcal{F} \subseteq \mathcal{F}'$ then $ICC_{k-loc}(\mathcal{F}) \subseteq ICC_{k-loc}(\mathcal{F}')$.*

Further, we present a series of generative properties of k-local internal contextual grammars.

Proposition 3.5 *$ICC_{(k+1)-loc}(\mathcal{F}) \setminus ICC_{k-loc}(\mathcal{F}) \neq \emptyset$ for all $k \geq 1$ and all the classes of languages $\mathcal{F}$ that contain FIN.*

Proof.
Consider the language L'_{k+1} defined in Example 3.2,

$$L'_{k+1} = \{a^{n_1}b_1a^{n_1}b_2a^{n_2}b_3a^{n_2}\ldots a^{n_{k+1}}b_{2k+1}a^{n_{k+1}} \mid n_1, \ldots, n_{k+1} \geq 0\}.$$

From Example 3.2 it follows that L'_{k+1} is generated by the grammar G'_{k+1} which has *FIN*-choice by $(k+1)$-local derivation. Thus, $L'_{k+1} \in ICC_{(k+1)-loc}(FIN)$, and, consequently, $L'_{k+1} \in ICC_{(k+1)-loc}(\mathcal{F})$ for all the classes of languages $\mathcal{F}$ that contain *FIN*.

For the second part of the proof, we show that $L'_{k+1} \notin ICC_{k-loc}(\mathcal{F})$ for any class of languages $\mathcal{F}$.

Assume the opposite: there exists a class of languages $\mathcal{F}$ such that $L'_{k+1} \in ICC_{k-loc}(F)$. Consequently, there exists an internal contextual grammar with $\mathcal{F}$-choice $G = (V, A, (L_1, c_1), \ldots, (L_n, c_n))$ such that $L_{k-loc}(G) = L'_{k+1}$.

For $1 \leq i \leq k+1$, denote by $N_i = 1 + \max\{n_i \mid$ there exist $n_1, \ldots, n_{i-1}, n_{i+1}, \ldots, n_{k+1}$, such that $a^{n_1}b_1a^{n_1}b_2a^{n_2}b_3a^{n_2}\ldots a^{n_i}b_{2i-1}a^{n_i}\ldots a^{n_{k+1}}b_{2k+1}a^{n_{k+1}} \in A\}$.

Consider $w = a^{N_1}b_1a^{N_1}b_2a^{N_2}b_3a^{N_2}\ldots a^{N_{k+1}}b_{2k+1}a^{N_{k+1}}$. By the definition of the language L'_{k+1} we have $w \in L'_{k+1} \setminus A$. Therefore, there exists a k-local derivation $\alpha \stackrel{*}{\Rightarrow}_{k-loc} w$, where $\alpha \in A$. Now, it is not hard to see that no context of the grammar, that is used in the derivation of w, contains any of the symbols $b_1, b_2, \ldots, b_k$; otherwise we could easily obtain words that contain more than one b_i-symbol for some $i \in \{1, \ldots, k\}$. Moreover, each of the $2k+2$ groups of a-symbols of w can be obtained only after the insertion of at least one context. Also, the only possibility for the sides of a context to be inserted simultaneously in two such groups of a-symbols would be to insert it around a word of the form $a^\ell b_i a^e$ for some $\ell, r \in IN$ and $i \in \{1, \ldots, k\}$. Now, if w would be derived by a k-local derivation we would obtain that the sides of the contexts could be inserted in at most $2k$ of the a-symbols groups. But this is a contradiction.

Thus $L'_{k+1} \notin ICC_{k-loc}(\mathcal{F})$, for any class of languages F. □ □

Proposition 3.6 *$ICC_{k-loc}(\mathcal{F})$ is incomparable with REG for all $k \geq 1$ and all the classes of languages $\mathcal{F}$ that contain FIN.*

Proof.
Let $k \geq 1$ be a positive integer. Consider the regular language L_{2k+1} defined in Example 3.1. Similarly to the second part of the proof of Proposition 3.5, one can show that $L_{2k+1} \notin ICC_{k-loc}(\mathcal{F})$ for any class of languages $\mathcal{F}$.

According to the previous examples we see that $ICC_{k-loc}(FIN)$ contains some regular languages. Moreover, if $\mathcal{F}$ contains *FIN* then $ICC_{k-loc}(\mathcal{F})$ also contains languages as L'_k, defined in Example 3.2, which is a context-free non-regular language. □ □

In fact, for any $k \geq 1$ and any class of languages F that contains *REG*, $ICC_{k-loc}(\mathcal{F})$ may contain even non-context-free languages. Example 3.3 presents a series of non-context free languages that can be generated by an internal contextual grammar with *REG*-choice by k-local derivation for all $k \geq 1$.

The following result answers an open problem from [8, page 43].

Proposition 3.7 *For any class of languages $\mathcal{F}$, the class of regular languages REG is not included in $ICC_{loc}(\mathcal{F})$.*

Proof. It follows as in the second part of the proof of Proposition 3.5 for $k = 1$, and using the equivalence between the 1-local derivation and the local derivation. □ □

We conclude this section by giving a result on the upper bound of the class of languages generated by k-local internal contextual grammars with at most *CS*-choice.

Proposition 3.8 *For any class of languages $\mathcal{F}$ contained in the class of context-sensitive languages, we have $ICC_{\infty-loc}(\mathcal{F}) \subset CS$.*

Proof.
Given an internal contextual grammar G, with at most context-sensitive choice, one can easily construct a Turing machine working in linear space and accepting $L_{k-loc}(G)$.

Further, recall that a language $L \subseteq V^*$ fulfills the *internal bounded step* property if there is a constant p such that for each $w \in L$ with $|w| > p$, there exists $y \in L$ such that $x = x_1ux_2vx_3$, $y = x_1x_2x_3$ and $0 < |uv| \leq p$ (see [13, page 243]). Clearly, any internal contextual grammar G with at most *CS*-choice generates, by k-local derivation, only languages having the internal bounded step property, while CS contains also languages that do not fulfill the IBS property. Hence, the inclusion $ICC_{\infty-loc}(\mathcal{F}) \subset CS$ is strict. This concludes our proof. □ □

4. Computational Aspects

For the rest of the paper we fix an internal contextual grammar $G = (V, A, (L_1,(u_1,v_1)), \ldots, (L_n,(u_n,v_n)))$ with *P*-choice. The problem that we try to solve in this section is the membership problem for the language generated by the grammar G: given a word $w \in V^*$ and $k \in IN$ we must decide whether w was generated by G or not by k-local derivation.

We may assume, without losing generality, that if (u_i, v_i) is a context of this grammar then $u_iv_i \neq \lambda$. Clearly, the derivation of a word w in such a grammar has at most $|w|$ steps. Also, note that, since L_i is in *P*, the language $u_iL_iv_i$ is in *P* as well for all $i \in \{1, \ldots, n\}$.

Our approach is based on a bottom-up strategy: we try to identify the rules that were used in the possible derivations of w, in the reverse order of their application, and we accept if and only if we obtain in this way a derivation that started with an axiom of G. A sketch of this approach is given in the following.

Generic Parsing Algorithm. Input: the word w.

1. **Let** w be the input word.
2. **Let** $S = \{w\}$ be a queue and $S' = \emptyset$ be a set (also implemented as a queue).
3. **While** S is not void do:
   3.1. **Extract** the word x from S; add x to S'.
   3.2. **If** x is an axiom: STOP; w is accepted.
   3.3. **For** all the contexts (u, v) such that $x = yusvz$, and s selects (u, v), **add** ysz to S, but only if $xyz \notin S \cup S'$.
4. STOP; S is void and no axiom was found, hence, the word w is rejected.

Clearly, a direct implementation of this approach runs in exponential time and it is generally unknown whether the membership problems for languages generated by unrestricted internal contextual grammars with *P*-choice can be solved in polynomial time. We show here that when we are interested in checking whether w can be derived by k-local derivation, we can implement the general idea mentioned above in polynomial time; however, degree of the polynomial bounding the running time of this algorithm depends on k.

First note that, due to the locality of the derivation, it is important to identify and memorize the exact place and derivation step in which each a context was inserted during a derivation, not only to identify and store the contextual rules that were used in that derivation. Thus, we use the following approach: each time a valid context is identified, it is marked; as in the case of the marked description of the derivation, the left side of an identified context is placed between (and), while the right side is placed between [and]. Moreover, we associate with every context we find the number of contexts that we already identified in the input word, to which we add 1. For example, assuming that m derivation steps are performed in order to obtain w from an axiom α of the grammar, the first context we identify (which is, in fact, the last context inserted in the derivation $\alpha \overset{*}{\Rightarrow}_{k-loc} w$) is associated with 1, the second with 2, and so on; the last context identified (which is actually the first context inserted in the derivation) is associated with m. Thus, the sequence of the numbers associated with the identified contexts is the reversed sequence of numbers associated with the contexts in the marked description of the derivation. For simplicity, a symbol a, from a word in which several contexts were identified and marked, is said to be $($-marked (respectively, $[$-marked) if there exists in the given word the symbol $(_q$ (respectively, the $[_q$ symbol) placed to the left of the symbol a and the symbol $)_q$ (respectively, the $]_q$ symbol) placed to the right of the symbol a for some $q \in I\!N$; a factor of the word is said to be $($-marked ($[$-marked) if it is placed between by $(_q$ and $)_q$ (by $[_q$ and $]_q$) for some $q \in I\!N$. A $($-marked ($[$-marked) factor of a word in which several contexts were identified and marked is said to be maximal if it is there is no other $($-marked ($[$-marked) factor of the same word that contains it.

We say that a word x is correctly parenthesized if the following hold:

- the word x contains at most one symbol $(_j$ and one symbol $)_j$ for all $j \in I\!N$; also, x contains at most one symbol $[_j$ and one symbol $]_j$ for all $j \in I\!N$;
- if x contains one symbol from $\{(_j,)_j, [_j,]_j\}$, then it contains all the symbols from this set, and $x = x'(_j y')_j z'[_j u']_j v'$;
- if $(_l$ appears in x between $(_j$ and $)_j$, then $)_l$ appears also between $(_j$ and $)_j$; if $[_l$ appears in x between $[_j$ and $]_j$, so does $]_l$.

It is not hard to see that the marked descriptor of a word z, with respect to a derivation of the grammar G, is correctly parenthesized. On the other hand, the words we obtain, by identifying the contexts inserted in a word and associating numbers with them, are also correctly parenthesized.

In the following we note an important property of the words obtained by identifying and marking contexts as described above. Let w be a word obtained by k-local derivation from α, and let $c_1, \ldots, c_m$ be the contexts inserted in this derivation. We identify and mark successively, in reverse order, the contexts inserted in w starting with the last one. After all these contexts are identified and marked, the unmarked symbols of w form the word α. Further, assume that we have marked the last r contexts inserted in the above derivation of w, for some $1 \leq r \leq m$. We state that in the word obtained by marking these contexts in w there are at most k maximal (-marked ([-marked) factors. It is sufficient to prove this statement for (-marked symbols, a similar argument being valid for [-marked sequences. We use induction: for $r \leq k$, the statement holds canonically. We assume that it holds for a given r, and we prove that it holds for $r+1$. Assume the opposite: after marking the context c_{m-r} we obtain $k+1$ maximal (-marked maximal factors. Since the statement was valid for r, it follows that the symbols of the left sides of the contexts $\{c_{m-r+1}, \ldots, c_m\}$ form exactly k factors, and by marking the left side of c_{m-r} none of these factors is extended. Let N be the number of different maximal (-marked factors, containing the symbols of the left sides of the contexts $\{c_{m-r+1}, \ldots, c_{m-r+k}\}$; from the definition of the k-local derivation it follows that the symbols of the left side of the context $c_{m-t+r+1}$ are contained in one of the factors delimited by $(_q$ and $)_q$, with $m-r+1 \leq q \leq m-r+k$. In the same manner, we prove that the left side of every context from the set $\{c_{m-r+2}, \ldots, c_m\}$ is contained in one of these N factors. Hence, $N = k$ and every left side of the contexts $\{d_{m-r+1}, \ldots, d_{m-r+k}\}$ is in a different maximal (-marked factor. Finally, the symbols of the left side of the context d_{m-r+k} are contained in one of the factors delimited by $(_q$ and $)_q$, with $m-r \leq q \leq m-r+k-1$; but, since these symbols are not contained in one of the factors delimited by $(_q$ and $)_q$, with $m-r+1 \leq q \leq m-r+k-1$, it follows that they are contained in the factor delimited by $(_{m-r}$ and $)_{m-r}$. This is a contradiction, thus, we completed the proof of the induction step. In conclusion, we showed that if the word w is obtained by a k-local derivation from α by inserting the contexts $c_1, \ldots, c_m$, then by marking the symbols of $c_{m-r}, \ldots, c_m$ in reverse order, for $r \geq 0$, we obtain at most k maximal (-marked ([-marked) factors.

According to the above, after identifying and marking the symbols of the last r identified contexts, we obtain a word with at most k (-marked ([-marked) maximal factors.

Now we briefly describe how a context is identified: assume that we have a word x in which $q-1$ contexts were identified so far; we also associated numbers with these contexts, so some of the symbols of x were marked. We search for a word usv as factors of the word

formed by the unmarked symbols of x, provided that $c = (u, v)$ is a context of the grammar and s is a selector of that context; if we find an occurrence of this word we place the symbols corresponding to u between $(_q$ and $)_q$ and the symbols corresponding to v between $[_q$ and $]_q$. Note that when doing this we may also place between $(_q$ and $)_q$ (respectively, between $[_q$ and $]_q$) several already marked symbols, and the corresponding parentheses. However, the insertion of the new parentheses is made such that the obtained word remains correctly parenthesized; if there are more than one possibility to do this then all of them are analyzed.

Once a new context, say the q^{th}, is identified in the word, we check whether the derivation constructed so far is k-local. Formally, we must check whether there exists r with $q > r \geq q - k$, such that $(_r$ and $)_r$ are found between $(_q$ and $)_q$ and $[_r$ and $]_r$ are found between $[_q$ and $]_q$. To this end, we say that the index r is satisfied if there exists an index q, with $r + k \geq q > r$, such that $(_r$ and $)_r$ are found between $(_q$ and $)_q$ and $[_r$ and $]_r$ are found between $[_q$ and $]_q$, in the word obtained after $r + k$ context were identified. Assume that, at some point, we succeeded to identify $r + k$ contexts; these derivation steps form a k-local derivation if and only if after the $(r + k)^{th}$ context was identified, and the corresponding parentheses inserted, all the indices that are not satisfied are strictly greater than r. Clearly, the indices of the parentheses limiting the maximal marked factors are not satisfied.

Now we are able to describe formally the main step of our algorithm.

Remark 2 *Let $\gamma \in (V \cup \{(_j,)_j, [_j,]_j \mid j \leq |w|\})^*$ be a word such that:*

- *γ is correctly parenthesized;*
- *$\gamma \mid_V = w$ and the unmarked symbols of γ form a word y;*
- *γ has m maximal $($-marked factors and m' maximal $[$-marked factors where both m and m' are less or equal to k;*

Then the following equivalence holds:

- *w is obtained from y by a p-steps k-local derivation, and*
- *γ can be obtained from w by marking the symbols of the contexts inserted in the above derivation, in reverse order of their insertion, and*
- *the unsatisfied indices of the parentheses appearing in γ are greater than $p - k$.*

if and only if there exists $\beta \in (V \cup \{(_j,)_j, [_j,]_j \mid j \leq |y|\})^*$ such that:

- *$\beta \mid_V = w$, the unmarked symbols of β form the word z, and w is obtained from z by a $(p-1)$-steps k-local derivation,*
- *β has j maximal $($-marked sequences and j' maximal $[$-marked sequences with $j, j' \leq k$; moreover, β can be obtained from w by marking the contexts inserted in the above derivation, in reverse order of their insertion,*
- *each of not-satisfied indices in β are greater than $p - 1 - k$, and*

- *there exists a context $c = (u, v)$ in X and one of its selectors s, such that γ can be obtained from β by identifying in z an occurrence of the sequence usv (in this order), and marking the symbols corresponding to the context c in β.* □

Proof. The proof of the direct implication is immediate. Assume that there exists a word γ that verifies the above and the contexts inserted during the derivation $y \stackrel{*}{\Rightarrow}_{k-loc} w$ are $c_1, \ldots, c_p$. As stated, γ is obtained from w by identifying and marking the symbols of these contexts. If we unmark the symbols of c_1 in γ, we obtain a word β. This word could have been obtained, also, from w by marking the symbols of $c_2, \ldots, c_t$. From the remarks we made, it follows that β verifies all the listed properties.

To prove the reverse implication, we first observe that $z \stackrel{*}{\Rightarrow} w$, since $y \Rightarrow z$ by the insertion of (u, v) and $z \stackrel{*}{\Rightarrow}_{k-loc} w$. Since β can be obtained from w by marking the symbols of the contexts inserted in the derivation $z \stackrel{*}{\Rightarrow}_{k-loc} w$ in reverse order of their insertion, it follows that γ can be obtained from w by marking the symbols of the contexts inserted in the derivation $x \stackrel{*}{\Rightarrow} w$. Also, it is easy to observe that this derivation is performed in p steps.

To prove that the locality condition holds for the derivation $y \Rightarrow z \stackrel{*}{\Rightarrow}_{k-loc} w$, we note that this condition holds canonically for the first k contexts inserted, and for the contexts inserted in the last $p - 1 - k$ steps (due to the k-locality of the derivation $z \stackrel{*}{\Rightarrow}_{k-loc} w$). Consequently we should prove only that the locality condition holds for the context inserted in the $k + 1$-st step. If this condition does not hold it results that the index $p - k$ is not satisfied in γ, a contradiction with the hypothesis. Hence, $y \stackrel{*}{\Rightarrow}_{k-loc} w$. This concludes our proof. □ □

Coming closer to an effective implementation of the bottom-up strategy mentioned before, note that at any moment of a successful analysis (one that confirms that the w can be generated by G starting from some word by k-local derivation) there exist at most k not-satisfied indices, and the difference between any two such indices is less than k. Thus, we can use, instead of the real value of the indices, their value modulo k. In this setting, the q derivation steps from a derivation of w discovered until a given moment do not form a k-local derivation if and only if there exist two unsatisfied indices, associated with the parentheses used to mark these contexts, that are equal modulo k; that is, the derivation of w is not k-local if and only if the indices of the parentheses associated with the $(q - k)^{th}$ context were not satisfied after the q^{th} context was identified. Also, one should memorize only the unsatisfied indices, the position of the parentheses associated with these indices and the number of contexts identified until that moment in the input word.

According to Remark 2, an algorithm deciding whether w can be generated by G could work as follows: perform a "breadth-first" search in the space of all the words that verify the conditions in Proposition 2, starting from the word w, and trying to find a word α, whose unmarked symbols form an axiom of the grammar. Following Remark 2, the transition, in such a search, between two words β and γ is possible if and only if γ can be obtained from β by marking the sides of a context correctly selected.

The basic data structure used in our algorithmic approach is called *item*, and it is a pair (γ, m) with $m \in I\!N$ such that $m \leq |w|$ and $\gamma \in (V \cup \{(_i,)_i, [_i,]_i \mid 0 \leq i \leq k-1\})^*$ such that γ contains t maximal (-marked sequences and t' maximal [-marked sequences with $t, t' \leq k$. Every word γ which verifies the properties stated in Remark 2 can be encoded as the item $(\overline{\gamma}, m)$ where m is the number of contexts identified in γ so far and $\overline{\gamma}$ is obtained from γ by

deleting the parentheses with satisfied indices and, then, by replacing each index with its value modulo k.

Now we are able to give an exact insight on how our algorithm works.

Remark 3 *In order to decide whether w can be generated by G we run an algorithm that performs a breadth-first search in the space of all the possible items, according to the following guidelines:*

- *The starting item of the search is $(w, 0)$;*
- *The transition from an item (β, a) towards an item (γ, b) is possible if and only if:*
  – $b = a + 1$;
  – γ' is a word obtained from β by identifying and marking (as described before) the sides of a context correctly selected,
  – γ is obtained from γ' by deleting all the parentheses with satisfied indices.
- *If we reach, during this search, an item (α, a), such that the unmarked symbols of α form an axiom of the grammar, then the algorithm stops and accepts. Otherwise, the algorithm stops and rejects the input word.*

It is not hard to see that this algorithm stops after a finite number of steps. Indeed, we note that, for a given word w, the graph defined by the items and the transitions connecting them is a directed acyclic graph, each path in this graph having at most $|w|$ items on it. Also, we have to explore only the items obtained from w by marking the symbols of several contexts, using markings indexed over the set $\{0, \ldots, k-1\}$. □

Note that every item that appears in the above search has at most k (-marked maximal factors and at most k [-maximal marked factors. Thus, we can bijectively associate with every item (α, a) a tuple $(i_0, j_0, \ldots, i_{2k-1}, j_{2k-1}, a)$ such that:

- For $l < k$, i_l and j_l are the positions in the input word where the factor placed between $(_l$ and $)_l$ starts and ends, respectively. More precisely, i_l (respectively, j_l) equals the position of the last symbol from the input word that is placed before $(_l$ $()_l$, respectively); if no such a symbol exists, then $i_l = 0$ ($j_l = 0$, respectively). If $(_l$, $)_l$ does not exist in α, then $i_l = j_l = -1$;
- For $l \geq k$, i_l and j_l are the positions in the input word where the factor placed between $[_{l-k}$ and $]_{l-k}$ starts and ends, respectively. More precisely, i_l (respectively, j_l) equals the position of the last symbol from the input word that is placed before $[_{l-k}$ $(]_{l-k}$, respectively); if no such a symbol exists, then $i_l = 0$ ($j_l = 0$, respectively). If $[_{l-k}$, $]_{l-k}$ does not exist in α, then $i_l = j_l = -1$.

Consequently, we can associate with a set of items I a characteristic function C_I, such that:

- $C_I(i_0, j_0, \ldots, i_{2k-1}, j_{2k-1}, a) = 1$ if the item (α, a), associated with the tuple $(i_0, j_0, \ldots, i_{2k-1}, j_{2k-1}, a)$, is in I;
- $C_I(i_0, j_0, \ldots, i_{2k-1}, j_{2k-1}, a) = 0$, otherwise.

In practice, this function will be implemented as a $4k+1$-dimensional array. Consequently, on a random access machine, both the time needed to modify the value of an element of this matrix and the time needed to compare the value of an element of the array with a given number are constant. The tuple that corresponds to an item that appears in the search algorithm described in Remark 3 can be roughly described as having on each of its components a value between -1 and $|w|$, where w is the input word. Hence, assuming that k is a constant, we state that there are at most $(|w|+2)^{4k+1}$, i.e. $\mathcal{O}(|w|^{4k+1})$ items that verify the restrictions described above.

We denote by $Items$ the set of all the items; we denote by $Items(w)$ the set of all the items that can appear in the analysis of the word w.

The second data structure that we use is needed in order to be able to obtain a description of the derivation of a word w. This structure consists of a partial function $f_w : Items(w) \to Items(w)$ that verifies the following properties:

- $f((\alpha, a)) = (\beta, a-1)$ if and only if there exists a transition from the item $(\beta, a-1)$ to (α, a);
- f is undefined otherwise.

In the implementation of the algorithm, this function can be described as a $4k+1$ dimensional array, containing tuples. The size of such an array is also $\mathcal{O}(|w|^{4k+1})$.

Also, our algorithm produces, if possible, a *final item*: this is an item that verifies the property that its unmarked symbols form an axiom of the grammar.

We can now formally define the parsing algorithm for the internal contextual grammar with context free choice G and k-local derivation, following the guidelines stated in Remark 3. We make the observation that every set used in this algorithm is implemented as a queue. Also, let us state that n is the number of contextual rules and m is the number of axioms in G.

Algorithm A

1. **Denote** by w the input word.
2. **Initialize** the set of items: $S = \{(w, 0)\}$ and the array associated with it, C_S.
3. **Initialize** the function f, described above, leaving it undefined in every point.
4. **Initialize** the set of items $S' = \emptyset$ and the array associated with it, $C_{S'}$.
5. **While** $S \neq \emptyset$ **do**:
   - 5.1. **Extract** the item (β, a) from S; **add** (β, a) to S' and adjust the values corresponding to this item in the arrays C_S and $C_{S'}$.
   - 5.2. **Let** y be the word formed by the unmarked symbols of β.
   - 5.3. **If** there exists $i \in \{1, \ldots, m\}$, such that y equals the axiom a_i **then**: **let** the final item be (β, a); **stop** the algorithm and **output** "YES".
   - 5.4. **For** all $i \in \{1, \ldots, n\}$ **do:**
     - 5.4.1. **Find** all the factors z of y such that $z = u_i s v_i$ with $s \in L_i$.
     - 5.4.2. **Generate** all possible words α obtained from β by marking the words u_i and v_i, found above, with parentheses indexed by $((a+1) \bmod k)$, as follows:

5.4.2.1. **Generate** the tuple associated with the item $(\alpha', a+1)$ from the tuple $(i_0, j_0, \ldots, i_{2k-1}, j_{2k-1}, a)$ associated with (β, a), by setting to -1 the four components corresponding to the indices that are going to be satisfied by the insertion of the parentheses $(_{(a+1) \bmod k}$, $)_{(a+1) \bmod k}$, $[_{(a+1) \bmod k}$, $]_{(a+1) \bmod k}$ in β.

5.4.2.2. **If** α' is correctly parenthesized and the equality $i_{(a+1) \bmod k} = j_{(a+1) \bmod k} = i_{k+((a+1) \bmod k)} = j_{k+((a+1) \bmod k)} = -1$ holds, **then generate** the tuple associated with the item $(\alpha, a+1)$ from the tuple associated with the item $(\alpha', a+1)$, by assigning to the above four components the values of the positions in w of the parentheses $(_{(a+1) \bmod k}$, $)_{(a+1) \bmod k}$, $[_{(a+1) \bmod k}$, $]_{(a+1) \bmod k}$, respectively.

5.4.2.3. **If** $(\alpha, a+1) \notin S \cup S'$ **then insert** $(\alpha, a+1)$ in S and adjust C_S.

5.4.2.4. **Let** $f((\alpha, a+1)) = (\beta, a)$.

6. **Stop** the algorithm and **output** "NO";

As a consequence of the remarks made before the definition of the algorithm, it is clear that the **Algorithm A** ends after a finite number of steps and outputs "YES" iff the word w is generated by the grammar G, and "NO" otherwise.

In the following we compute the time complexity of this algorithm, as a function depending on the length of the input word w. Note that the time bounds that we will prove are valid for random access machines.

It is obvious that the steps 1 and 6 are executed in constant time. In the execution of the steps 2 and 4, the initialization of the arrays corresponding to the sets S and S' is preformed in $\mathcal{O}(|w|^{4k+1})$ computational time, and the initialization of the sets (queues) S and S' is done in constant time. Similarly, the initialization of the function f, in step 3, requires $\mathcal{O}(|w|^{4k+1})$. Consequently, the entire computation in steps $2, 3$ and 4 is carried out in $\mathcal{O}(|w|^{4k+1})$ time. It is not hard to see that the number of times the While cycle from step 5 is executed coincides with the cardinality of the set S' at the end of the algorithm. Consequently, it also equals the number of items reached during the search performed by **Algorithm A**. Hence, this cycle is executed for at most $\mathcal{O}(|w|^{4k+1})$ times. Now, we have to check the running time needed for the operations performed in one iteration of this cycle. The step 5.1 can be executed in constant time while the step 5.2 needs $\mathcal{O}(|w|)$ computational time. The execution of the step 5.3 also needs linear time. First, we have to check if there exists an axiom of the grammar that equals y. Hence, we have to do a constant number of comparisons between two word; every such comparison needs as many symbols comparisons as the length of the shorter of the two words plus a comparison between the length of the two words, and, consequently, every comparison can be implemented in constant time (since all the axioms have their length bounded by a constant). The other operations performed in this step can be executed in linear time. Finally, we check the complexity of the cycle in the step 5.4. This cycle is executed for n times, hence a constant number of times. First, step 5.4.1 requires $\mathcal{O}(|w|^2 P(|w|))$ time, where $P(n)$ is the time complexity needed to recognize any of the languages L_i for $i \in \{ildots, n\}$. The cycle in step 5.4.2 is executed for all the possible words that can be obtained from β by marking an occurrence of a word of the form $u_i s v_i, s \in L_i$, as a subsequence of y. There are $\mathcal{O}(|\beta|^4) = \mathcal{O}(|w|^4)$ possibilities of placing the markings in β; for each of these possibilities, one

may discover in linear time if it verifies the conditions imposed in this step: the markings delimit an occurrence of the context (u_i, v_i), correctly selected, and, the word obtained is correctly parenthesized. Indeed, we can store the occurrences of the words $u_i s v_i, s \in L_i$, in y during the execution of step 5.4.1., and we can check if a word is correctly parenthesized in linear time. Consequently, this cycle is executed for $\mathcal{O}(|w^4|)$ times, for each possibility a linear time verification of the conditions being needed. Also, each of the steps in the cycle 5.4.2 can be executed in linear time. To conclude, the execution of the whole cycle 5.4.2 can be performed in $\mathcal{O}(|w|^5)$ computational time. The complexity of the cycle 5.4 is, hence, $\mathcal{O}(|w|^5)$. Finally, we obtain the complexity of one execution of the operations contained by the While cycle in step 5: $\mathcal{O}(|w|^5)$. Consequently, the overall running time of step 5 is: $\mathcal{O}(|w|^{4k+6})$. Adding the complexity of the steps $1, 2, 3, 4, 5$ and 6, we obtain that total time complexity of the parsing algorithm on the input w: $\mathcal{O}(|w|^{4k+6})$.

By this we have proved:

Theorem 4.1 *For an internal contextual grammar G with P-choice, the membership problem can be solved in polynomial time by* **Algorithm A**. *The complexity of this algorithm is* $\mathcal{O}(N^{4k+6})$. □

Remark 4 *The complexity of the* **Algorithm A** *can be reduced by precomputing the positions where a context could be found in a word obtained from w by ($-marking t sequences and [$-marking t' sequences, $t, t' \leq k$. In particular cases (in the case when $k = 1$, for example) the locality condition can be expressed in a less complicated form, and, also, the identification of the contexts can be carried out in a more efficient way; such things can bring important optimizations to our algorithm (mainly in the implementation of the cycle* 5.4.2.*). See, for instance [3]. In this paper, our main interest was to prove that parsing k-local contextual languages is polynomial, not to provide the most efficient algorithm that does it. Hence, we proposed a more time consuming, yet polynomial, more readable algorithm, rather than presenting a very difficult-to-follow, though optimized, version.*

Our algorithm can be used also as a parsing algorithm. In order to obtain a description of a derivation of the input word w accepted by the **Algorithm A**, we should just follow the path found by **Algorithm A** from the item $(w, 0)$ to the final item, as it is provided by the function f. In the following we present an algorithm that obtains the marked descriptor of w, according to a derivation of w in G, discovered by the **Algorithm A**.

Algorithm B

1. **Let** (α, a) be the final item produced by the **Algorithm A** for the input word w. **Let** $t = a$ be the length of the derivation of w and $\beta = w$ be a word.

2. **While** $a > 0$ **do**:

   2.1. **Let** $(\alpha', a-1) = f((\alpha, a))$.

   2.2. **Construct** the word β' by inserting in β the $(t-a+1)$-indexed parentheses at the positions indicated in $(i_0, j_0, \ldots, i_{2k-1}, j_{2k-1}, a)$, the tuple associated with (α, a). More exactly, the symbol $(_{t-a+1}$ is inserted in β on the position $i_{a \bmod k}$, the symbol $)_{t-a+1}$ is inserted in β on the position $j_{a \bmod k}$, the symbol $[_{t-a+1}$ is inserted in β on the position $i_{k+a \bmod k}$, the symbol $]_{t-a+1}$ is inserted in β on the position $j_{k+a \bmod k}$.

Note that all these positions are relative to the symbols of w. Additionally, the new parentheses should be inserted in such a way to produce a correctly parenthesized word and to satisfy the indices that appear as not-satisfied in α', but appear as satisfied in α.

2.3. **Substitute** $\alpha = \alpha'$, $\beta = \beta'$, $a = a - 1$;

3. **Output** β.

The strategy implemented by this algorithm is very simple. Since w is accepted by **Algorithm A**, a final item (α_0, a) is reached. If w is an axiom in G, then $(\alpha_0, a) = (w, 0)$, and w is its own marked descriptor. Therefore, in this case, the algorithm outputs w.

If w is not an axiom, then starting with the final item (α_0, a) provided by the **Algorithm A**, we obtain a sequence of items $(\alpha_l, a-l)$, with $0 \leq l \leq a$, such that $(\alpha_l, a-l) = f(\alpha_{l-1}, a-l+1)$ for any $1 \leq l \leq a$. From the reverse implication of Proposition 2, such a sequence exists, and the word w is obtained from the axiom $\alpha_0|_V$, by successively inserting a contexts from G.

These contexts are successively marked in the word w, in the order of their application, using the information stored in the items $(\alpha_l, a-l)$, with $0 \leq l \leq a-1$. The positions, in which the l-indexed parentheses are inserted, are extracted from the item $(\alpha_{l-1}, a-l+1)$ for each $1 \leq l \leq a$. More exactly, with the tuple $(i_0, j_0, \ldots, i_{2k-1}, j_{2k-1}, a-l+1)$ associated with $(\alpha_{l-1}, a-l+1)$, the symbol $(_{t-a+1}$ is inserted in w on the position $i_{a \bmod k}$, the symbol $)_{t-a+1}$ is inserted in w on the position $j_{a \bmod k}$, the symbol $[_{t-a+1}$ is inserted in w on the position $i_{k+a \bmod k}$, the symbol $]_{t-a+1}$ is inserted in w on the position $j_{k+a \bmod k}$. Remember, from the definition of a tuple associated with an item produced by the **Algorithm A**, that positioning a parenthesis relative to the positions of the symbols of w means to insert the parenthesis with the position i after the $i-th$ symbol of w (the 0 position being in front of all symbols of w).

Moreover, when several parentheses share the same position in w, we apply the following principles for deciding their relative order:

- The new parentheses should be inserted such that a correctly parenthesized word is produced.

- The new parentheses of an index l, with $2 \leq l \leq a$, should be inserted in order to satisfy the indices that appear as not-satisfied in α_l, but appear as satisfied in α_{l-1}.

To obtain the running time of the **Algorithm B**, we analyze, as in the case of **Algorithm A**, each of its steps. Step 1 can be executed in constant time. The cycle in the step 2 is executed for at most a times, with $a \leq |w|$. The most time-consuming step in this cycle is the step 2.2, and it requires $\mathcal{O}(|w|)$ time. To conclude, this algorithm can be implemented in $\mathcal{O}(|w|^2)$ computational time.

Finally, the marked descriptor of w produced by the **Algorithm B** can be further processed in order to generate the marked description of the derivation of w in G. It is straightforward that the computational complexity of the **Algorithm D**, that does this thing, is also $\mathcal{O}(|w|^2)$ for an input word w.

5. Conclusions

The generalization of local contextual grammars proposed in this paper is supported by the fact that such grammars have rather poor generative properties. Proposition 3.7 answers an open question given by Ilie in [8]: there are regular languages that cannot be generated by local internal contextual grammars with finite or regular choice. A similar result is proved in Proposition 3.6 for any k-local internal contextual grammar with a fixed $k \geq 1$. It remains an open problem whether any regular language can be generated by a k-local internal contextual grammar for some k depending on that regular language.

The problem of recognizing k-local internal contextual languages with only finite, regular or context-free choice should be considered; it seems interesting to see whether the particular form of the selectors permits a more efficient implementation of our entire strategy, or only the identification of the contexts and their selectors can be performed faster. It is worth seeing if our approach works for other types of (linguistically motivated) contextual grammars.

References

[1] G. B. ENGUIX, R. GRAMATOVICI, Parsing with Active P Automata. In: *Workshop on Membrane Computing'03*. Lecture Notes in Computer Science 2933, Springer, 2004, 31–42.

[2] R. GRAMATOVICI, An Efficient Parser for a Class of Contextual Languages. *Fundam. Inform.* **33** (1998) 3, 211–238.

[3] R. GRAMATOVICI, F. MANEA, A CYK-based Parser for Local Internal Contextual Grammars with Context-Free Choice. In: *Proc. AFL'05*. 2005, 31–42.

[4] R. GRAMATOVICI, F. MANEA, Parsing Local Internal Contextual Languages with Context-Free Choice. *Fundam. Inform.* **64** (2005) 1-4, 171–183.

[5] R. GRAMATOVICI, C. MARTÍN-VIDE, Sorted dependency insertion grammars. *Theor. Comput. Sci.* **354** (2006) 1, 142–152.

[6] K. HARBUSCH, An Efficient Online Parser for Contextual Grammars with at Most Context-Free Selectors. In: *CICLing*. Lecture Notes in Computer Science 2588, Springer, 2003, 168–179.

[7] K. HARBUSCH, Parsing Contextual Grammars with Linear, Regular and Context-Free Selectors. In: *Grammars and Automata for String Processing*. Topics in Computer Mathematics 9, Taylor and Francis, 2003, 45–54.

[8] L. ILIE, On Computational Complexity of Contextual Languages. *Theor. Comput. Sci.* **183** (1997) 1, 33–44.

[9] P. JANCAR, F. MRÁZ, M. PLÁTEK, M. PROCHÁZKA, J. VOGEL, Restarting Automata, Marcus Grammars and Context-Free Languages. In: *Developments in Language Theory*. 1995, 102–111.

[10] S. MARCUS, Contextual grammars. *Revue Roum. Math. Pures Appl.* **14** (1969), 1525–1534.

[11] F. MRÁZ, M. PLÁTEK, M. PROCHÁZKA, Restarting automata, deleting and Marcus grammars. In: *Recent Topics in Mathematical and Computational Linguistics*. Romanian Academy Publishing House, 2000, 218–233.

[12] G. PĂUN, *Marcus Contextual Grammars*. Kluwer Publ. House, Doordrecht, 1998.

[13] G. ROZENBERG, A. SALOMAA (eds.), *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997.