Leftist Grammars and the Chomsky Hierarchy

Tomasz Jurdziński and Krzysztof Loryś

Institute of Computer Science, Wrocław University
Przesmyckiego 20, 51151 Wrocław, Poland

Abstract. Leftist grammars can be characterized in terms of rules of the form $a \to ba$ and $cd \to d$, without distinction between terminals and nonterminals. They were introduced by Motwani et. al. [9], where the accessibility problem for some general protection system was related to these grammars. This protection system was originally proposed in [3, 10] in the context of Java virtual worlds. Here, the accessibility problem is formulated in the form "Can object p gain (illegal) access to object q by a series of legal moves (as prescribed by the policy)?". It was shown [9] that the membership problem for leftist grammar is decidable what implied the decidability of the accessibility problem for the appropriate protection system.
We study relationships between language classes defined by various types of leftist grammars and classes of the Chomsky hierarchy. We show that general leftist grammars can recognize languages which are not context free, answering in negative the question from [9]. Moreover, we study some restricted variants of leftist grammars and relate them to regular, deterministic context free and context free languages.
Topics: formal languages; accessibility problem in protection systems.

1 Introduction

Leftist grammars were introduced by Motwani et. al. [9] and related to the accessibility problem for some general protection system of computer systems. A protection system is a set of policies that prescribes the ways in which *objects* interact with each other. By objects we mean users, processes or other entities and interactions can include access rights, information sharing privileges and so on. The accessibility problem for the protection system is formulated in the form "Can object p gain (illegal) access to object q by a series of legal moves (as prescribed by the policy)?". A formal treatment of accessibility was first presented by Harrison, Ruzzo and Ullman [5] who showed that the accessibility problem is undecidable for a general access-matrix model of object-resource interaction. This result prompted a broad research on tradeoffs between expressibility and verifiability in protection systems. The work on protection systems took place mainly in the context of operating systems and currently, operating systems have efficient protection mechanisms. However, these mechanisms often fail at the scale necessary for today's Internet [1].

The protection system related to leftist grammars was originally proposed in [3, 10] in the context of Java virtual worlds. The model of this protection

system strictly generalizes grammatical protection systems [2, 7] and the take grant model [8], it is a special case of the general access-matrix model [5]. Its advantage over the general access-matrix model is the fact that accessibility is decidable for this model what was obtained by the reduction to the membership problem of leftist grammars [9].

Formally, the protection system is defined here in the following way. Let $G = (V, E)$ be a directed graph, let $t : V \to T$ be a type function assigning types to vertices. Then, let $R_i, R_e \subseteq T \times T$ be two binary relations. There are following operations on G:

- Insert(v, x) which inserts a new vertex x of any type and an edge (v, x).
- Give(a, b, c) which inserts an edge (b, c), provided $(a, b), (a, c) \in E$ and $(t(b), t(c)) \in R_i$.
- Get(a, b, c) which inserts an edge (a, c), provided $(a, b), (b, c) \in E$ and $(t(b), t(c)) \in R_e$.

Now, we define the accessibility problem. Given the graph G, vertices p, q and relations R_i, R_e, a question is whether there exists a sequence of operations of the above type such that there exists an edge (p, q) after applying this sequence of operations to G. As shown in [9], the accessibility problem for the above set of operations can be reduced to the accessibility problem for a model in which the operations Insert and Give are combined into a new operation Insert(a, x, c) which adds a vertex x and edges $(a, x), (x, c)$, provided $(t(x), t(c)) \in R_i$ and $a, c \in V$.

Leftist grammars can be characterized in terms of rules of the form $a \to ba$ and $cd \to d$ over the alphabet Σ (there is no distinction between terminals and nonterminals). A symbol $x \in \Sigma$ is called a final symbol and a word $w \in \Sigma^*$ belongs to the language defined by a grammar G iff there exists a derivation which starts at wx and ends at x. Intuitively, the rules of type $a \to ba$ correspond to the operation Insert, the rules of type $cd \to d$ correspond to the operation Get, and a derivation of a leftist grammar correspond to a sequence of operations Insert and Get applied to a (sub)graph which is a simple path.

As pointed out above, the membership problem for these grammars is decidable [9]. Moreover, the problem of emptiness of the intersection of the language defined by a leftist grammar and a regular language is decidable. This result implied decidability of the accessibility problem for the protection system from [9]. However, no efficient algorithm for the membership problem of leftist grammars is known. And, there are no nontrivial lower bounds for this problem. In particular, a question if leftist grammars recognize only context-free languages was addressed in [9]. The lack of efficient algorithms for general leftist grammars motivates also exploration of some restricted variants of these grammars, what was addressed by Motwani et al. [9].

From language theoretic point of view, leftist grammars do not even satisfy restrictions of context-sensitivity, as they can have length-reducing rules and length-increasing rules simultaneously. On the other hand, the productions of theses grammars are severely restricted, so one could expect that their complexity is restricted as well. As the leftist grammars are elegantly characterized

and seem to be very simple, the study of their expressiveness is motivated both by their connections to the complexity of the accessibility problem and by itself. As pointed out in [9], slight generalizations of leftist grammars make the membership problem undecidable, so studying their restricted variants is more reasonable than the analysis of generalizations.

2 Our Results

We study relationships between language classes defined by various types of leftist grammars and classes of the Chomsky hierarchy. First, we propose a natural classification according to the restrictions on so called *delete graphs* and *insert graphs*. Here, the insert graph is induced by the rules of type $a \to ba$. The set of vertices of this graph corresponds to symbols of the alphabet Σ of the grammar and each rule $a \to ba$ corresponds to an edge (a, b). Similarly, each rule $cd \to d$ corresponds to an edge (d, c) in the delete graph. We show that general leftist grammars with arbitrary insert graphs and arbitrary delete graphs can recognize languages which are not context free, answering in negative the question from [9]. Further, we show that each time an insert graph or a delete graph is acyclic, a language defined by the grammar is included in CFL. More precisely, we relate these restricted classes of grammars to the set of regular, deterministic context free and context free languages. Our results are summarized in the following table, where FIN, REG, CFL, and DCFL, denote the classes of finite, regular, context-free, and deterministic context-free languages, respectively.

Delete graph	Insert graph	Included in	Not included in
acyclic	arbitrary	REG	FIN
arbitrary	empty	DCFL	REG
arbitrary	acyclic	CFL	DCFL
arbitrary	arbitrary	recursive*	CFL

$\star$ – proved in [9]

In Section 3 we provide some basic definitions and notations. Then, in Section 4, we explore properties of so-called leftmost derivations. Next, in Sections 5, 6, and 7 we investigate expressive power of restricted variants of leftist grammars and relate them to the classes of the Chomsky hierarchy. Section 8 is devoted to the proof of the fact that the set of languages defined by general leftist grammars is not included in CFL. Finally, in Section 9, we summarize our results and state some open problems.

3 Definitions and Notations

For a word x, $|x|$ denotes its length, the ith symbol of x is denoted by $x[i]$ $(0 < i \leq |x|)$, and $x[i, j]$ denotes the factor $x[i] \dots x[j]$ for $0 < i \leq j \leq |x|$. Let $[i, j] = \{l \in \mathbb{N} \mid i \leq l \leq j\}$. Throughout the paper ε denotes the empty word, $\mathbb{N}$, $\mathbb{N}_+$ denote the set of non-negative and positive integers. Sometimes, we will identify regular expressions with regular languages defined by them.

We refer the reader to [6, 4] for a basic knowledge and terminology from formal language theory.

Definition 1. *A leftist grammar $G = (\Sigma, P, x)$ consists of a finite alphabet Σ, a final symbol $x \in \Sigma$, and a set of production rules P of the following two types,*

$$\begin{aligned} ab &\to b \quad \textit{(Delete Rule)} \\ c &\to dc \quad \textit{(Insert Rule)} \end{aligned}$$

where $a, b, c, d \in \Sigma$.

We say that a string $u \in \Sigma^*$ derives a string $v \in \Sigma^*$, denoted by $u \Rightarrow v$, if $u = u_1 y u_2$ and $v = u_1 z u_2$ such that $y \to z$ is a production rule in P. As usual, $\Rightarrow^*$ denotes the reflexive and transitive closure of $\Rightarrow$. If u derives v, then we say that $u \Rightarrow v$ is a derivation step.

A sequence of derivation steps

$$u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_p$$

is called a derivation. A word u_i for $i \in [1, p]$ is called a sentential form in this derivation.

Finally, the language of G is defined to be

$$L(G) = \{w \in \Sigma^* \mid wx \Rightarrow^* x\}.$$

Notice that reversing the directions of all production rules would give a more standard definition of a grammar, where x would be the starting symbol. However, following the convention from [9], we will mainly use the definition stated above.

Throughout the paper, we will implicitely treat symbols of sentential forms as objects which can insert/delete other symbols and can be inserted/deleted. So, we make distinction between different occurences of a symbol $a \in \Sigma$ in a sentential form. However, in order to simplify notations, we will often identify the occurence of the symbol a in a sentential form with its value a. It should be clear from the context whether we say about a symbol as an element of the alphabet or an element of a sentential form.

We say that the symbol b in the delete rule $ab \to b$ is *active*. Similarly, the symbol c is *active* in the insert rule $c \to dc$.

Let $u \Rightarrow v$, where $u = u_1 y u_2$ and $v = v_1 z v_2$ such that $y \to z$ is a production rule in P. We would like to say that a symbol which is active in the production rule $y \to z$ is also *active* in the derivation step $u \Rightarrow v$ (that is, the rightmost symbol of the prefix $u_1 y$). However, it is possible that there are many factorizations $u = u_1 y u_2$ such that $v = u_1 z u_2$ and $y \to z$ is a production. In order to avoid this ambiguity, we associate arbitrary such factorization in each derivation step. And, for a factorization $u = u_1 y u_2$ we say that the rightmost symbol of $u_1 y$ is *active* in the derivation step $u_1 y u_2 \Rightarrow u_1 z u_2$. In this way we will be able to determine uniquely which symbols are inserted/deleted by any particular symbol. Let $u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_p$ be a derivation with a fixed choice of factorizations $u_i = u_{i,1} y_i u_{i,2}$ which determine active symbols in consecutive

derivation steps. A symbol $u_1[i]$ is *active* in u_1 with respect to the derivation $u_1 \Rightarrow^* u_p$ if it is active in any of derivation steps of the derivation $u_1 \Rightarrow^* u_p$.

Despite the above ambiguity of factorizations for general leftist grammars, we show that for each leftist grammar G, there exists a leftist grammar G' with unique position of the active symbol for each possible derivation step and $L(G) = L(G')$.

Proposition 1. *Assume that a leftist grammar G does not contain any production of type $aa \to a$ or $a \to aa$ for $a \in \Sigma$. Then, for each derivation step $u \Rightarrow v$, there exists a unique factorization $u = u_1 y u_2$ such that $v = u_1 z u_2$ and $y \to z$ is a production of G.*

Proof. Assume that $|v| > |u|$. Then, an insert rule is applied in the step $u \Rightarrow v$. Let i be a minimal value such that $u[i] \neq v[i]$. The only rule which could be applied is $u[i] \to v[i]u[i]$ for the factorization $u[1, i-i]u[i]u[i+1, |u|]$. Indeed, a factorization which chooses a symbol to the right of $u[i]$ would imply that $u[i] = v[i]$. A factorization which chooses $u[j]$ for $j < i$ would imply that $u[j] \neq v[j]$ as for each insert rule $a \to ba$, we have $b \neq a$. But this contradicts the assumption that i is minimal such that $u[i] \neq v[i]$.

A similar arguments work also for the case $|v| < |u|$. □

Below, we show that for each language defined by a leftist grammar G, there exists a leftist grammar G' such thaat $L(G) = L(G')$ and G' does not contain rules of type $a \to aa$ for $a \in \Sigma$.

Proposition 2. *Let $G = (\Sigma, P, x)$ be a leftist grammar. Let $P' = P \setminus \{a \to aa \,|\, a \in \Sigma\}$. Then, $L(G) = L(G')$, where $G' = (\Sigma, P', x)$.*

Proof. Let $a \to aa$ for $a \in \Sigma$ be called "cloning productions". We show that for each derivation $wx \Rightarrow^* x$, there exists a derivation $wx \Rightarrow^* x$ which does not use "cloning productions". More precisely, for a derivation $wx \Rightarrow^* x$ with $n > 0$ applications of "cloning productions", we show how to construct a derivation $wx \Rightarrow^* x$ with $n - 1$ applications of cloning productions.

Let $uav \Rightarrow uaav$ be a derivation step in which (according to the choosen factorization) a production $a \to aa$ is applied. Let $uaav \Rightarrow^* y \Rightarrow y'$ be a subderivation such that y is the last sentential form in which the symbol a inserted in the step $uav \Rightarrow uaav$ is not yet deleted. Thus, all derivation steps in the subderivation $uaav \Rightarrow^* y$ which involve symbols to the left of the suffix av (in the sentential form $uaav$) are independent from the derivation steps which involve symbols from this suffix. We can rearrange the derivation $uaav \Rightarrow^* y$ in such a way that first appear the derivation steps which involve the symbols to the left of the suffix av and then the remaining steps. In this way we obtain a derivation

$$uaav \Rightarrow^* u'aav \Rightarrow^* u'av' \Rightarrow u'v',$$

where $u'av' = y$ and $u'v' = y'$.

Now, let us skip the derivation step $uav \Rightarrow uaav$. Then, we can obtain a derivation

$$uav \Rightarrow^* u'av \Rightarrow u'v',$$

as the first subderivation does involve only the symbols from the prefix ua and the second involve only the symbols from the suffix av. Thus, we have obtained a new derivation $wx \Rightarrow^* x$, which contains less applications of the productions of type $aa \to a$ than the original one. In this way, each derivation may be stepwise transformed into a derivation which does not use productions of this type at all. □

Proposition 3. *Let $G = (\Sigma, P, x)$ be a leftist grammar. Let*

$$P' = P \backslash \{aa \to a \mid a \in \Sigma\} \cup \{x' \to ax' \mid (x \to ax) \in P\} \cup \{ax' \to x' \mid (ax \to x) \in P\},$$

where x' is a new symbol, $x' \notin \Sigma$. Then, $L(G) = L(G')$, where $G' = (\Sigma \cup \{x'\}, P', x')$.

Proof. Certainly, by changing the starting symbol x into x' and adding productions

$$\{x' \to ax' \mid (x \to ax) \in P\} \cup \{ax' \to x' \mid (ax \to x) \in P\},$$

one obtains a grammar G'' which defines the same language as G. Note also that G'' does not have a rule $x'x' \to x'$.

For each derivation $wx' \Rightarrow^* x'$ in G'' which applies $n > 0$ times the rules of type $aa \to a$, we construct a derivation $wx' \Rightarrow^* x'$ which applies such rules $n-1$ times. Let $ua^iv \Rightarrow ua^{i-1}v$ be a derivation step in the derivation $wx' \Rightarrow^* x'$ such that a factorization defining the position of the active symbol gives a production $aa \to a$ inside the infix a^i. Moreover assume that:

- the rightmost position of u as well as the leftmost symbol of v are not equal to a and $v \neq \varepsilon$ (such a factorization exists, because there is no rule $x'x' \to x'$).
- there is no application of the rule $aa \to a$ in the subderivation $ua^{i-1}v \Rightarrow^* x'$; that is, we consider the last application of this rule during the whole derivation.

Now, let $ua^{i-1}v \Rightarrow^* u'av' \Rightarrow u'v'$ be a subderivation of the derivation

$$ua^{i-1}v \Rightarrow^* x'$$

such that the derivation step $u'av' \Rightarrow u'v'$ deletes the rightmost a from the sequence a^{i-1}. So, this derivation step applies a rule $ab \to b$ for $b \neq a$. Let us skip the derivation step $ua^iv \Rightarrow ua^{i-1}v$ and follow the derivation $ua^{i-1}v \Rightarrow^* u'av' \Rightarrow u'v'$. Then, we obtain the following subderivation

$$ua^iv \Rightarrow^* u'aav' \Rightarrow u'av'.$$

In the sentential form $u'av'$, we can apply a rule $ab \to b$ once again, deleting the symbol a which is between u' and v'. Thus, we have obtained a new derivation $wx' \Rightarrow^* x'$, which contains less applications of the productions of type $aa \to a$ than the original one. In this way, each derivation may be stepwise transformed into a derivation which does not use productions of this type at all. □

Propositions 1, 2 and 3 have the following implication.

Corollary 1. *For each leftist grammar G, there exists a leftist grammar $G' = (\Sigma', P', x')$ such that $L(G) = L(G')$ and, for each possible derivation step $u \Rightarrow_{G'} v$, there exists only one factorization $u = u_1 y u_2$ such that $v = u_1 z u_2$ where $y \to z$ is a production of G'.*

Based on Corollary 1, we can consider only such grammars, in which each derivation step of the grammar uniquely identifies which symbol of the sentential form is active in this step. In particular, if $uav \Rightarrow ubav$ or $ubav \Rightarrow uab$ for $a \neq b$, then the symbol a preceding v is active in this derivation step. All our results assume that grammars satisfy this condition.

Fact 2 *Let $u \Rightarrow^* x$ be a derivation. Then, one can obtain a derivation $u \Rightarrow^* x$ such that the symbol $u[1]$ is not active in any step of this derivation.*

Let $\mathsf{FIN}_{>0}$ denote a set $\mathsf{FIN} \setminus \{\emptyset, \{\varepsilon\}\}$.

Fact 3 *The set of languages generated by leftist grammars is disjoint with $\mathsf{FIN}_{>0}$.*

Proof. Let G be a leftist grammar such that $L(G) \notin \{\emptyset, \{\varepsilon\}\}$. Let $w \neq \varepsilon$ be a word in $L(G)$, $|w| = n$. Then, there exists a derivation which starts at wx and ends at x. No insert rule is applied in which the leftmost symbol of w (i.e. $w[1]$) is active in this derivation (Fact 2). Further, the symbol $w[1]$ is deleted at some derivation step. So, G derives also a word

$$w[1]^i w[2, n]$$

for each $i > 0$, what implies that the language $L(G)$ is infinite. □

The above fact implies that if the set of languages generated by any type of leftist grammars is included in the set which contains finite languages, then this inclusion is proper.

3.1 Insert Graphs and Delete Graphs

Let $G = (\Sigma, P, x)$ be a leftist grammar, where $\Sigma = \{a_1, \ldots, a_p\}$. An *Insert Graph* of G has p vertices $v_1, \ldots, v_p$. There exists an edge (v_i, v_j) in this graph iff the grammar contains a rule $a_i \to a_j a_i$. Similarly, a *Delete Graph* of G has p vertices $v_1, \ldots, v_p$. There exists an edge (v_i, v_j) in this graph iff the grammar contains a rule $a_j a_i \to a_i$.

We will consider the cases that the insert/delete graph is empty, acyclic, or arbitrary. These cases will be denoted by empty, $\mathsf{acyclic}$ and arb, respectively. Leftist grammars with delete graphs of type A and insert graphs of type B are denoted by $\mathsf{LG}(A, B)$. For example, $\mathsf{LG}(\mathsf{acyclic}, \mathsf{arb})$ denotes leftist grammars with acyclic delete graphs and arbitrary insert graphs. Note that, by Propositions 2, 3, for each grammar G of type $\mathsf{LG}(A, B)$ where $A, B \in \{\mathsf{empty}, \mathsf{acyclic}, \mathsf{arb}\}$, there exists a leftist grammar $G' = (\Sigma', P', x')$ such that $L(G) = L(G')$, G' is of type $\mathsf{LG}(A, B)$ and G' does not contain productions of type $aa \to a$ nor $a \to aa$ for any $a \in \Sigma'$.

4 Leftmost Derivations and their Properties

Let $u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_p$ be a derivation. A symbol $u_1[i]$ is *alive* in u_1 with respect to the derivation $u_1 \Rightarrow^* u_p$ if there exists $j \leq i$ such that $u_1[j]$ is active with respect to $u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_p$. A symbol which is not alive is *gone*.

A derivation $u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_p$ is the *leftmost derivation* if in each derivation step $u_i \Rightarrow u_{i+1}$ the leftmost alive symbol with respect to $u_i \Rightarrow^* u_p$ is active in $u_i \Rightarrow u_{i+1}$.

Definition 4. *Let $u \Rightarrow^* v$ be a leftmost derivation. Assume that $u[i]$ is gone with respect to $u \Rightarrow^* v$. Then, $u[i]$ is firm with respect to this derivation if it is not deleted until v. Otherwise, $u[i]$ is useless in u.*

Proposition 4. *If there exists a derivation $u \Rightarrow^* v$ then there exists a leftmost derivation which starts at u and ends at v.*

Proof. An induction with respect to the number of derivation steps. If there are no derivation steps, then the derivation is certainly leftmost. Let $u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_p$ be a derivation. Assume that $u_1 \Rightarrow^* u_i$ is the leftmost derivation for $i > 1$. If the active symbol in $u_i \Rightarrow u_{i+1}$ is equal to the leftmost alive symbol in u_i then the derivation $u_1 \Rightarrow^* u_{i+1}$ is leftmost as well. Otherwise, assume that the symbol which is active in a derivation step $u_i \Rightarrow u_{i+1}$ is not the leftmost alive symbol in u_i (with respect to the derivation $u_i \Rightarrow^* u_p$). Then, there exists $j > i$ such that the leftmost alive symbol in u_i is active in a derivation step $u_j \Rightarrow u_{j+1}$. Let j be a smallest value which satisfies this condition. According to the choice of j, the derivation $u_i \Rightarrow^* u_j$ does not change anything in the prefix which ends at the leftmost alive symbol in u_i. And, this symbol is active in the step $u_j \Rightarrow u_{j+1}$. Thus, if one applies a derivation rule from $u_j \Rightarrow u_{j+1}$ before the (sub)derivation $u_i \Rightarrow^* u_j$ then, after the derivation corresponding to $u_i \Rightarrow^* u_j$, we obtain u_{j+1}. In this way we constructed another derivation $u \Rightarrow^* v$ in which first $i+1$ derivation steps form a leftmost derivation from a derivation $u \Rightarrow^* v$ in which first i derivation steps form a leftmost derivation. Moreover, the number of derivation steps in the new derivation is equal to the number of derivation steps in the original derivation. □

Proposition 5. *Let G' be a leftist grammar Then, each leftmost derivation $v \Rightarrow^* w$ satisfies the following condition: Each sentential form u in this derivation has a factorization $u = u_1u_2u_3$ such that all symbols in u_3 are alive, all symbols in u_2 are useless, and all symbols in u_1 are firm.*

Proof. The fact that alive symbols form the suffix of a sentential form follows directly from the definition of alive symbols. For the sake of contradiction, assume that a useless symbol a is located directly to the left of a firm symbol b. However, as each firm symbol is not active nor deleted in a further derivation, it is not possible to delete a symbol located directly to the left of it. Contradiction, because a should be deleted (it is useless). □

Next, we make the following observation.

Proposition 6. *Assume that two symbols of the sentential form in the leftmost derivation are inserted by the same symbol. Then, the symbol inserted earlier is gone in this sentential form.*

Proof. Note that, if a and b which occur in a sentential form are inserted by the same symbol and a is inserted earlier than b, then b is to the right of a in the sentential form. But, in the derivation step inserting b, all symbols located to the left of this copy of b are gone. □

We introduce a notion which formally describes the way in which symbols in sentential forms were inserted. Let $U \equiv u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_p$ be a derivation. Let b, d be symbols which appear in some sentential forms of this derivation. We say that b is a *descendant* of d in U if (d, b) belongs to the reflexive and transitive closure of the relation $\{(x, y) \mid vxw \Rightarrow vyxw$ is a derivation step in U for some v and $w\}$.

Further, we define a *history* of each symbol which appears during the derivation $U \equiv u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_p$. A history of a symbol a which appears in u_1 is equal to a word $h(a) = a$. Further, let c be a symbol inserted in a derivation step $vbw \Rightarrow vcbw$ of U. Then, a history of (this copy of) c is equal to a word $h(c) = ch(b)$. So, the history of each symbol (except symbols which appear in the "initial" sentential form) is fixed at a moment when it is inserted.

Proposition 7. *Let $ubav$ be a sentential form in a leftmost derivation. Assume that the symbol a following the prefix ub has descendants in $ubav$ with respect to the considered derivation. Then $u = u_1u_2$ such that u_2b form all descendants of a. Moreover, if the symbol b following the prefix u is alive in $ubav$, then it was inserted by the symbol a which follows it.*

Proof. The first statement follows directly from the form of insert rules and delete rules of leftist grammars.

For the second statement, notice that the only possibility to move to the left the position of the leftmost alive symbol in the leftmost derivation, is to make an insert step. As a has descendants in $ubav$, it has inserted some symbols in the subderivation which ends at $ubav$. It was the leftmost alive symbol in derivation steps in which it inserted symbols. And, as long as the leftmost alive symbol is to the left of a, the rightmost symbol inserted by it is not deleted. □

Proposition 8. *Let*

$$u \Rightarrow^* y_1ay_2y_3 \Rightarrow y_1bay_2y_3 \Rightarrow^* v$$

be a leftmost derivation, let a symbol b following y_1 be an descendant of the rightmost symbol of y_2. Then, a history of this copy of b is equal to bay_2.

Proof. Note that the factor ay_2 contains alive symbols in $y_1ay_2y_3$, because a is active in the derivation step starting from this sentential form. By Proposition 6, each symbol in ay_2 inserted at most one symbol in this factor. So, by Proposition 7, the $y_2[i]$ inserted $y_2[i-1]$ for $i \in [2, |y_2|]$ and $y_2[1]$ inserted a. Thus, the history of b following y_1 is equal to bay_2. □

Proposition 9. *Let $uav \Rightarrow^* w$ be a leftmost derivation. Then, the suffix v remains unchanged as long as the symbol a following u is alive with respect to this derivation. If the symbol a following the prefix u is useless with respect to the derivation $uav \Rightarrow^* w$, then the prefix ua remains unchanged, as long as the symbol a following u is not deleted.*

The statements of the above proposition follow immediately from the definition of alive and useless symbols.

5 Grammars with Empty Insert Graphs

Now, we consider grammars with empty insert graphs.

Theorem 1. *The set of languages recognized by grammars* $\mathsf{LG}(\mathsf{arb}, \mathsf{empty})$ *is included in* DCFL.

Proof. Assume that $uawx \Rightarrow^* x$ is the leftmost derivation for an input word uaw and a at the position $|u| + 1$ is the leftmost alive symbol with respect to this derivation. Thus, all symbols in the prefix u are gone, they are not active in any derivation step. Then, there exists a leftmost derivation $u'awx \Rightarrow^* x$ for each u' which is a subsequence of u. Indeed, as no symbol from u is active in the derivation, we obtain a leftmost derivation $u'aw \Rightarrow^* x$ by removing from the derivation $uawx \Rightarrow^* x$ all steps which delete symbols in u that do not appear in u'.

The above observation implies the following fact. Let $u \in L(G)$ for a grammar G of type $\mathsf{LG}(\mathsf{arb}, \mathsf{empty})$, let $v = ux$. Let $i \in [2, n+1]$ be minimal value such that G contains a production $v[i-1]v[i] \to v[i]$, where $n = |u|$. Then, a word $u' = u[1, i-2]u[i, n]$ belongs to $L(G)$ as well. Using this property, we obtain the following algorithm determining if $w \in \Sigma^*$ belongs to $L(G)$. In each step, the algorithm finds a leftmost position such that an application of the (delete) production rule is possible at this position. Then, an appropriate symbol is deleted from the sentential form. This process is repeated until we obtain x what means that $w \in L(G)$ or $y \neq x$ such that no application of any rule of G is possible in y, what implies that $w \notin L(G)$.

Observe that the above algorithm satisfies the following condition. If it chooses a rewrite $ba \to a$ in a sentential form $wbay$ at some step and replaces $wbay$ into way then no symbol from the prefix w is active in any further derivation steps chosen by the algorithm. So, one can design the following DPDA M which determines if $w \in \Sigma^*$ belongs to $L(G)$. The automaton starts with w as an input word. First, it pushes $w[1]$ on the stack. Next, it applies the following rules. Let b be a symbol on the top of the pushdown and let a be a currently scanned input symbol. If G contains a production $ba \to a$ then M pops b from the pushdown and makes the ϵ-transition (i.e. it does not move its input head). Otherwise, M moves its input head to the right and pushes a on the pushdown. Finally, when the input head achieves a right delimiter, M pops symbols from the pushdown, as long as G contains a production $ax \to x$, where a is a symbol

on the top of the pushdown. If M finishes computation with the empty pushdown, it accepts, otherwise it rejects. One can easily verify that in this way M implements the above algorithm which determines if the input word belongs to $L(G)$. □

Theorem 2. *The set of languages recognized by grammars* $\mathsf{LG}(\mathsf{arb}, \mathsf{empty})$ *is not included in* REG.

Proof. We describe a grammar $G = (\Sigma, P, x)$ of type $\mathsf{LG}(\mathsf{arb}, \mathsf{empty})$ which generates a non-regular language. Let $\Sigma = \{a_0, a_1, b_0, b_1, x\}$ and let P contain the following production rules:

$$\{a_i b_i \to b_i \mid i = 0, 1\} \cup \{b_{1-i} b_i \to b_i \mid i = 0, 1\} \cup \{b_1 x \to x\}.$$

Now, let $w = a_1 a_0$ and $u = b_0 b_1$. We make the following observation

$$w^n u^m \in L(G) \iff m \geq n.$$

The implication $\Leftarrow$ is obvious. The second implication follows from two observations:

- a symbol a_i for $i = 0, 1$ can be deleted only by b_i;
- each copy of b_i for $i = 0, 1$ in the input word from $w^* u^*$ is able to delete at most one copy of a_i.

So, the language $w^* u^* \cap L(G)$ is equal to a non-regular language $\{w^n u^m \mid m \geq n\}$. As the set of regular languages is closed under intersection, the language $L(G)$ is non-regular as well. □

6 Grammars with Acyclic Insert Graphs

Next, we analyze grammars with acyclic insert graphs.

Proposition 10. *Let* G' *be a leftist grammar with acyclic insert graph. Let* $u \Rightarrow^* v$ *be a leftmost derivation in* G'. *Then, each symbol in each sentential form of this derivation has at most* $|\Sigma|$ *descendants which are alive.*

Proof. Let $way \Rightarrow wbay$ be a derivation step which inserts a symbol b. Then, by Proposition 8, $y = y_1 y_2$ such that bay_1 is equal to the history of this b and b is the descendant of the rightmost symbol of y_1. As the insert graph is acyclic, this history is not longer than the depth of the insert graph, which is bounded by $|\Sigma|$. And, because we consider the leftmost derivation, all symbols to the left of this b are gone in $wbay$. Thus, all alive descendants of the rightmost symbol of y_1 are included in bay_1 and $|bay_1| \leq |\Sigma|$. □

Proposition 11. *Let* G' *be a leftist grammar with acyclic insert graph. Then, for each leftmost derivation* $w \Rightarrow^* w'$, *there exists a leftmost derivation which starts at* w, *ends at* w' *and there is no sentential form in this derivation which contains two useless descendants of the same symbol with equal histories.*

Proof. Let $w \Rightarrow^* w'$ be a leftmost derivation. Assume that a sentential form $uavay$ appears in this derivation and copies of a located directly to the left of v and directly to the right of v are useless symbols that are descendants of the same symbol and have equal histories. We show how to shorten this derivation such that one of these a's does not appear in any sentential form. Moreover, we do not introduce any new derivation steps which would insert symbols. Let az be a history of both these copies of a. According to Propositions 8 and 9, the original derivation $w \Rightarrow^* w'$ contains a subderivation

$$w \Rightarrow^* u_1azz' \Rightarrow^* uazz' \Rightarrow^* uav_1azz' \Rightarrow^* uavazz' \Rightarrow^* uavay,$$

where

- u_1azz' is a sentential form obtained after a derivation step which inserts the left copy of a, preceding v (see Proposition 8);
- $uazz'$ is a first sentential form in which the left copy of a is gone (i.e. useless) – (see Propositions 9 and 8);
- uav_1azz' is a sentential form obtained in a derivation step which inserts the right copy of a, following v (see Propositions 8 and 9);
- $uavazz'$ is a first sentential form in which the right copy of a is gone and useless (see Proposition 9);

As the whole av following u is useless in the sentential form $uavazz'$ (what follows from Proposition 5 and the fact that the first and the last symbol in the factor ava are useless), the derivation $uavazz' \Rightarrow^* w'$ implies that there exists also a derivation $uazz' \Rightarrow^* w'$ obtained from the original derivation by deleting the subderivation $uazz' \Rightarrow^* uavazz'$, and all derivation steps which remove symbols from the factor av in the subderivation $uavazz' \Rightarrow^* w'$. Thus, we obtained a new leftmost derivation $w \Rightarrow^* w'$ which avoids the sentential form $uavazz'$ with two useless symbols a that have equal histories and are descendants of the same symbol. In this way, each derivation $w \Rightarrow^* w'$ may be stepwise transformed into a derivation $w \Rightarrow^* w'$ which satisfies conditions of the proposition. □

Theorem 3. *The set of languages recognized by grammars of type* $\mathsf{LG}(\mathsf{arb}, \mathsf{acyclic})$ *is included in* CFL.

Proof. By Proposition 11, if $w \in L(G)$ then there exists a leftmost derivation $wx \Rightarrow^* x$ such that for each sentential form of this derivation and each symbol a in this sentential form, there are no two different descendants of a which are useless and have equal histories. However, as the derivation $wx \Rightarrow^* x$ deletes all symbols except the initial symbol x, each symbol which is gone in any sentential form during the derivation $wx \Rightarrow^* x$, is also the useless symbol.

By Proposition 10, each symbol in each leftmost derivation $wx \Rightarrow^* x$ has at most $|\Sigma|$ alive descendants. So, for each $w \in L(G)$, there exists a leftmost derivation $wx \Rightarrow^* x$ such that each symbol in each sentential form has at most $s = |\Sigma| + |\Sigma|^{|\Sigma|}$ descendants. We construct a PDA A which, for each input word $w \in L(G)$ guesses a leftmost derivation $wx \Rightarrow^* x$ that satisfies this condition. The automaton A starts with w as an input word and the empty pushdown store.

First, it pushes $w[1]$ on the stack. Next, it applies the following rules. Let y be a current content of the pushdown, let $w[i]$ be a currently scanned input symbol. Assume that $yw[i,n]x$ is the current sentential form, all symbols in y are useless. So, as long as $w[i]$ is alive starting from $yw[i,n]x$, the active symbol is equal to $w[i]$ or an descendant of $w[i]$. Next, as long as $w[i]$ is alive, A simulates (guesses) an appropriate subderivation without moving its input head. All descendants of $w[i]$ are stored in the finite control of A. If the particular derivation step changes only the set of descendants of $w[i]$, this step changes only the state of A (describing these descendants). However, if the number of descendant of $w[i]$ is larger than s, A rejects. If the derivation step deletes a symbol from the prefix y, this symbol is removed from the pushdown. If A nondeterministically decides that $w[i]$ becomes gone, the current sequence of its descendants and $w[i]$ itself are pushed on the pushdown store. Finally, when the input head achieves the right delimiter, A continues the above process assuming that the current pushdown store contains useless symbols, and the only active symbol is x. If A is able to remove all symbols from the pushdown and all descendants of x inserted during this process, it accepts (as before, if the number of descendants of x is larger than s, A rejects). Otherwise, it rejects. The problem is that A may loop infinitely without moving its input head. However, it is able to determine such loops in the finite control and avoid them.

One can easily verify that A accepts exactly $L(G)$. Indeed, one can show by induction on i that, after the step in which A moves its input head on the ith position of the input word, the concatenation of the content of the pushdown store and the part of the input which is not read yet form a sentential form obtained in a leftmost derivation after the last derivation step in which an input symbol from $w[1, i-1]$ is alive. Moreover, A is able to simulate each leftmost derivation in which each symbol from the input word has at most s descendants. So, the result follows. □

Theorem 4. *The set of languages recognized by grammars* $\mathsf{LG}(\mathsf{arb}, \mathsf{acyclic})$ *is not included in* DCFL.

Proof. We will define a grammar $G = (\Sigma, P, x)$ with an acyclic insert graph, such that the language $L(G)$ does not belong to DCFL. Let

$$\Sigma = \{a_i \mid i \in [0,3]\} \cup \{b_i, e_i, f_i \mid i \in [0,1]\} \cup \{c, d, x\}.$$

Production rules of G are following, where $i = 0, 1$:

$$
\begin{aligned}
b_i &\to e_i b_i \\
b_i &\to f_i b_i \\
b_{1-i} e_i &\to e_i \\
b_{1-i} f_i &\to f_i \\
e_{1-i} e_i &\to e_i \\
f_{1-i} f_i &\to f_i \\
a_j e_{j \bmod 2} &\to e_{j \bmod 2} \quad \text{for } j \in [0,3] \\
a_j f_{j \operatorname{div} 2} &\to f_{j \operatorname{div} 2} \quad \text{for } j \in [0,3] \\
b_1 c &\to c \\
e_1 c &\to c \\
b_1 d &\to d \\
f_1 d &\to d \\
cx &\to x \\
dx &\to x \qquad .
\end{aligned}
$$

Let $w = a_3a_2a_1a_0$, $u = b_0b_1$. We show that

$$w^n u^m c \in L(G) \iff m \geq 2n.$$

The implication $\Leftarrow$ is simple. In fact, there exists a leftmost derivation in which each b_i inserts e_i. This e_i deletes b_{1-i}, e_{1-i} (if they occur directly to the left of e_i) and one element of $\{a_i, a_{i+2}\}$ (if exists). Finally, c deletes the rightmost b_1 and e_1, and x deletes c.

For the implication $\Rightarrow$, observe that a derivation which starts from a word in w^*u^*cx and ends at x cannot use symbols f_0, f_1. Indeed, these symbols can be deleted only by d and there is no insert rule which inserts d, while there is no occurence of d in any word from w^*u^*cx. So, the only insert rules which are applied in such a derivation introduce symbols e_0, e_1. Let us consider a leftmost derivation for an input word from w^*u^*c. Then, according to the above observations, each sentential form in this derivation belongs to the set

$$w^*\{a_3a_2a_1, a_3a_2, a_3, \varepsilon\}\{b_0, b_1, e_0, e_1\}^*[e_i]b_i\{b_0, b_1\}^*cx,$$

where $i \in [0,1]$, b_i or e_i preceding the suffix $\{b_0, b_1\}^*c$ is the leftmost alive symbol in this sentential form. Let $u_1[e_i]b_iu_2x$ be such a sentential form, where b_i or e_i following u_1 is the leftmost alive symbol. We claim that, as long as this b_i is not deleted, only one symbol from the set $\{a_0, a_1, a_2, a_3\}$ can be deleted. Indeed, b_i is not able to delete any symbol and it can insert only e_i. The symbol e_i can delete a_i and a_{i+2} and cannot insert any symbol. However, each two copies of symbols from the set $\{a_i, a_{i+2}\}$ are separated by a_{i+1} or $a_{(i+3) \bmod 4}$. Thus, b_i and all descendants of b_i are able to delete at most one symbol from the set $\{a_0, a_1, a_2, a_3\}$.

As c and x cannot delete any of a_i's, the above fact implies that, if a word w^nu^mc belongs to $L(G)$, then the number of copies of symbols from the set $\{b_0, b_1\}$ is not smaller than the number of copies of symbols from the set $\{a_0, a_1, a_2, a_3\}$. This condition is satisfied iff $m \geq 2n$.

Using similar arguments as above, one can show that

$$w^n u^m d \in L(G) \iff m \geq n.$$

This property follows from the fact that a derivation for a word $w^n u^m d$ can use symbols f_0, f_1 and cannot use e_0, e_1 what implies that, for each b_i, $i \in [0,1]$, at most two symbols from $\{a_0, a_1, a_2, a_3\}$ can be deleted in the part of the (leftmost) derivation which starts in a step when this b_i is active and finishes in the step in which this copy of b_i is deleted.

The above observations imply that

$$L(G) \cap (w^* u^* c \cup w^* u^* d)$$

is equal to the language

$$L' = \{w^n u^m c \,|\, m \geq 2n\} \cup \{\{w^n u^m d \,|\, m \geq n\}.$$

As the language L' is not in DCFL, and DCFL is closed under intersection with regular languages, $L(G)$ is not in DCFL, either. □

7 Grammars with Restricted Delete Graphs

In this section we show that leftist grammars with acyclic delete graphs define only regular languages.

Let G be a grammar of type $\mathsf{LG}(\mathsf{acyclic}, \mathsf{arb})$. Let G' be a "reversed" grammar with respect to G. That is, for each production $ab \to b$ in G, G' contains a production $b \to ab$; similarly, each production $c \to dc$ of G corresponds to a production $dc \to c$ in G'. Certainly, $L(G)$ is equal to the set

$$\{w \,|\, x \Rightarrow^*_{G'} wx\}.$$

The delete graph of G is equal to the insert graph of G' and the insert graph of G is equal to the delete graph of G'. So, if the delete graph of G is acyclic then the insert graph of G' is acyclic as well.

Proposition 12. *Let G' be a leftist grammar with acyclic insert graph. Then, for each leftmost derivation $x \Rightarrow^* wx$, the number of alive symbols in each sentential form is not larger than the depth of the insert graph.*

Proof. Each symbol in the derivation $x \Rightarrow^* wx$ is an descendant of the rightmost x. So, the statement follows from Proposition 10. □

Proposition 13. *Let G' be a leftist grammar with acyclic insert graph. Then, for each word w such that $x \Rightarrow^* wx$, there exists a leftmost derivation $x \Rightarrow^* wx$, such that the number of useless symbols in each sentential form of this derivation is not larger than $|\Sigma|^{|\Sigma|}$.*

Proof. Note that each symbol in each sentential form of the leftmost derivation $x \Rightarrow^* wx$ is a descendant of the rightmost initial symbol x. And, by Proposition 11, there exists a leftmost derivation $x \Rightarrow^* wx$ such that the number of useless descendants of the rightmost symbol x in each sentential form is not larger than the number of possible histories of symbols. As the insert graph is acyclic, the number of possible histories of symbols is at most $|\Sigma|^{|\Sigma|}$. So, the result follows. □

Note that the leftmost derivation $x \Rightarrow^* wx$ makes firm symbols from the final word w from left to right. That is, first $w[1]$ becomes a status firm, then $w[2], w[3]$ and so on (see Proposition 5). Thus, one can design the following nondeterministic one-way finite automaton A which, for each $w \in \Sigma^*$, determines if $x \Rightarrow^*_{G'} wx$. The automaton A guesses a leftmost derivation $x \Rightarrow^* wx$ such that each sentential form in this derivation contains at most $|\Sigma|$ alive symbols and at most $|\Sigma|^{|\Sigma|}$ useless symbols. A starts the computation with the input head on $w[1]$. Then, it guesses a subderivation until the symbol $w[1]$ is inserted and becomes firm, without moving its head. It stores the current sequence of all alive symbols and all useless symbols in its finite control. After the derivation step in which $w[1]$ is inserted, A moves its head to the right and continues the simulation until $w[2]$ is inserted and becomes firm, still storing active and useless symbols in its finite control. This process in continued for all consecutive symbols of the input word until $w[|w|]$ is inserted or A is not able to "insert" a consecutive symbol of the input word without increasing the number of useless symbols over $|\Sigma|^{|\Sigma|}$. In the latter case, it rejects. In the former case, A continues the simulation until there are no alive nor useless symbols (assuming that the rightmost x is the only symbol which should not be deleted among the sequence of symbols stored in the finite control). If it guesses correctly such a derivation, it accepts. Otherwise, it rejects. Propositions 12 and 13 imply that A accepts exactly the set of words w such that $x \Rightarrow^*_{G'} wx$, i.e. $w \in L(G)$.

So, finally, we obtain the following theorem.

Theorem 5. *The set of languages recognized by grammars of type* $\mathsf{LG}(\mathsf{acyclic}, \mathsf{arb})$ *is included in* REG.

8 General Leftist Grammars

In this section we describe a leftist grammar that defines a language which is not context-free. Let $G = (\Sigma, P, x)$ be a grammar with the alphabet

$$\Sigma = \{a_i, B_i, F_i \mid i = 0, 1\} \cup \{X_{i,j}, D_{i,j} \mid i, j = 0, 1\} \cup \{x\},$$

and the following set of productions, where $i, j, k \in \{0, 1\}$:

$$
\begin{array}{ll}
(10) & a_i \to B_i a_i \\
(20) & a_i \to X_{i,0} a_i \\
(30) & X_{i,j} \to Y_{i,j} X_{i,j} \\
(40) & Y_{i,j} \to D_{i,j} Y_{i,j} \\
(50) & Y_{i,j} \to X_{i,1-j} Y_{i,j}
\end{array}
$$

$$
\begin{array}{ll}
(60) & a_{1-i} B_i \to B_i \\
(70) & B_i D_{i,j} \to D_{i,j} \\
(80) & D_{i,j} D_{i,1-j} \to D_{i,1-j} \\
(83) & X_{1-i,k} D_{i,0} \to D_{i,0} \\
(86) & Y_{1-i,k} D_{i,1} \to D_{i,1}
\end{array}
$$

$$
\begin{array}{ll}
(90) & a_0 F_0 \to F_0 \\
(100) & X_{0,j} F_0 \to F_0 \\
(110) & Y_{0,j} F_1 \to F_1 \\
(120) & F_{1-i} F_i \to F_i
\end{array}
$$

$$
\begin{array}{ll}
(140) & F_1 x \to x \\
(150) & D_{i,j} x \to x
\end{array}
$$

Let

$$
\begin{array}{l}
\mathcal{A} = \{a_0, a_1\} \\
\mathcal{F} = \{F_0, F_1\} \\
\mathcal{D}_i = \{D_{i,0}, D_{i,1}\} \\
\mathcal{X}_i = \{X_{i,0}, X_{i,1}\} \\
\mathcal{Y}_i = \{Y_{i,0}, Y_{i,1}\} \\
\mathcal{Z}_i = \{X_{i,0}, Y_{i,0}, X_{i,1}, Y_{i,1}\}
\end{array}
$$

where $i \in [0, 1]$. For sets of symbols $\mathcal{U}, \mathcal{V}$, by $\mathcal{U}\mathcal{V}$ we mean the set $\{uv \,|\, u \in \mathcal{U} \text{ and } v \in \mathcal{V}\}$.

Proposition 14. *Let $n, m \in \mathbb{N}$. If $n \geq 2^{2m-2}$, then a word $w = (a_1 a_0)^m (F_0 F_1)^n$ belongs to $L(G)$.*

Proof. First, observe that there exist the following derivations in G for $i \in [0, 1]$, $j \in \mathbb{N}$:

$$a_i \Rightarrow^* B_i (D_{i,0} D_{i,1})^j (Y_{i,1} X_{i,1} Y_{i,0} X_{i,0})^j a_i. \tag{1}$$

We obtain such a derivation by applying the rules (10), (20), (30), (40) at the beginning, then the sequence (50), (30), (40), $2j-1$ times (where always the leftmost possible application of the appropriate rule is choosen; each such sequence inserts $D_{i,1} Y_{i,1} X_{i,1}$ or $D_{i,0} Y_{i,0} X_{i,0}$).

Next, observe that for each $j \in \mathbb{N}$, $i \in [0, 1]$, there exists a derivation

$$(\mathcal{Y}_i \mathcal{X}_i)^j a_i B_{1-i} (D_{1-i,0} D_{1-i,1})^j \Rightarrow^* D_{1-i,1}. \tag{2}$$

Indeed, the above derivation is obtained by applying the productions (60), (70) and then the sequence of productions (83), (80), (86), (80) $j-1$ times (where always the leftmost possible application of the appropriate rule is choosen) and finally (83), (80) and (86).

Using the above properties, we can show by induction the following result.
Claim For each $p > 1$, there exist the following derivations

$$(a_1a_0)^{p-1}a_1 \Rightarrow^* D_{0,0}D_{1,1}(D_{0,1}D_{1,1})^{p-2}(Y_{1,1}X_{1,1}Y_{1,0}X_{1,0})^{2^{2p-4}}a_1$$
$$(a_1a_0)^{p} \Rightarrow^* D_{0,0}D_{1,1}(D_{0,1}D_{1,1})^{p-2}D_{0,1}(Y_{0,1}X_{0,1}Y_{0,0}X_{0,0})^{2^{2p-3}}a_0.$$

Proof (of Claim) For $p = 2$ and the word $(a_1a_0)a_1$ we start by the following derivation

$$a_1a_0 \Rightarrow a_1B_0a_0 \Rightarrow B_0a_0 \Rightarrow^3 B_0D_{0,0}Y_{0,0}X_{0,0}a_0 \Rightarrow D_{0,0}Y_{0,0}X_{0,0}a_0.$$

Then, the derivation $a_1a_0a_1 \Rightarrow^* D_{0,0}D_{1,1}Y_{1,1}X_{1,1}Y_{1,0}X_{1,0}a_1$ is obtained by the application of Equation (1) (with $i = 1$, $j = 1$) for the rightmost a_1 and Equation (2) (with $i = 1$, $j = 1$).

For $p \geq 2$ and the word $(a_1a_0)^p$ as well as $p > 2$ and the word $(a_1a_0)^{p-1}a_1$, we use the induction hypothesis and Equations (1) and (2), respectively. (Claim) □

By the above claim, there exists the following derivation for $n, m \in \mathbb{N}$:

$$(a_1a_0)^{m} \Rightarrow^* D_{0,0}D_{1,1}(D_{0,1}D_{1,1})^{m-2}D_{0,1}(Y_{0,1}X_{0,1}Y_{0,0}X_{0,0})^{2^{2m-3}}a_0.$$

Similarly to the recognition of the example language from Theorem 2, one can show that 2^{2m-2} copies of F_0F_1 are sufficient to remove 2^{2m-3} copies of $Y_{0,1}X_{0,1}Y_{0,0}X_{0,0}$ (by application 2^{2m-2} times the sequence of productions (100), (120), (110), (120)). In this way, for each $n \geq 2^{2m-2}$, we obtain a derivation

$$\begin{aligned}(a_1a_0)^m(F_0F_1)^nx &\Rightarrow^* D_{0,0}D_{1,1}(D_{0,1}D_{1,1})^{2m-2}D_{0,1}(Y_{0,1}X_{0,1}Y_{0,0}X_{0,0})^{2^{2m-3}}(F_0F_1)^nx\\ &\Rightarrow^* x,\end{aligned}$$

where the second subderivation is obtained by applying the productions from the set (90) − (150) □

Now, for an input word $w = (a_1a_0)^m(F_0F_1)^nx$, we formulate conditions which are necessary in order to exist a derivation $w \Rightarrow^* x$.

Proposition 15. *Let $(a_1a_0)^m(F_0F_1)^nx \Rightarrow^* x$ be a derivation in G. Then, $n \geq 2^{2m-2}$.*

Proof. First, we specify some necessary conditions satisfied by each derivation $(a_1a_0)^m(F_0F_1)^nx \Rightarrow^* x$. All conditions specified in claims stated below concern such derivations. Observe that no insert rule of G inserts a symbol x, so each sentential form of each derivation which starts from $(a_1a_0)^m(F_0F_1)^nx$ contains only one x, at its rightmost position.
Claim 1 For each $i \in [0, 1]$ and each copy of a_i in the word $(a_1a_0)^m(F_0F_1)^nx$ except the leftmost a_1, a_i inserts a symbol B_i which deletes a_{1-i} located directly

to the left of it. Moreover, its descendants insert $D_{i,j}$ for $j \in [0,1]$ which deletes B_i inserted by it. And, the rightmost element of $\mathcal{D}_i$ which is the descendant of this a_i is deleted by x.

Proof of Claim 1 We prove this statement by induction. Notice that the rightmost a_0 has to insert B_0 which deletes the rightmost a_1. Otherwise, the rightmost a_1 would not be deleted at all, as other (than B_0) possible descendants of a_0 and the elements of $\mathcal{F} \cup \{x\}$ do not delete a_1, nor insert any other symbols. As there are no delete rules in which the elements of $\mathcal{F} \cup \{x\}$ delete B_0, the descendants of the rightmost a_0 have to insert an element of $\mathcal{D}_0$, which deletes B_0. Finally, as the elements of $\mathcal{D}_0$ may be deleted only by other elements of $\mathcal{D}_0$ and by x, the rightmost descendant of the rightmost a_0 which belongs to $\mathcal{D}_0$ remains undeleted until it is just to the left of x.

Now assume that our hypothesis is satisfied for $p-1$ rightmost elements of A, where $2m > p > 1$. Let the pth (rightmost) element of A be a_j for $j \in [0,1]$. The induction hypothesis guarantees that a_{1-j} located directly to the right of this a_j has an descendant from $\mathcal{D}_{1-j}$ which will be deleted by x. So, a_{1-j} to the left of this a_j should be deleted by x, this a_j the next a_{1-j} or any of descendants of these three symbols. As x does not insert anything and it is not possible that the symbol deleting a_{1-j} is an descendant of a_{1-j}, a_j has to insert B_j which will delete a_{1-j} located to the left of it. Similarly, no possible descendant of a_{1-j} nor x can delete B_j, so it is necessary that there is an element of $\mathcal{D}_j$ among descendants of this a_j. The rightmost element of $\mathcal{D}_j$ which is the descendant of a_j will be separated from the rightmost element of $\mathcal{D}_{1-j}$ which is the descendant of a_{1-j} located directly to the right of this a_j (which exists by induction hypothesis) by elements which could be descendants of a_0 or descendants of a_1 except of elements of $\mathcal{D}_j$. As none of these elements can delete the element of $\mathcal{D}_j$ and, by the induction hypothesis, the symbol to the right of this sequence (i.e. the rightmost descendant of a_{1-j} which belongs to $\mathcal{D}_{1-j}$) is deleted by x, the rightmost descendant of the pth element from A which belongs to $\mathcal{D}_j$ is deleted by x. (Claim 1) □

Claim 2 For each $i \in [0,1]$ and each copy of a_i in the word $(a_1a_0)^m(F_0F_1)^n x$ except the leftmost a_1, all descendants of a_{1-i} located directly to the left of this a_i and belong to $\mathcal{Z}_{1-i}$ should be deleted by the descendants of this a_i.

Proof of Claim 2 First, notice that each copy of a_i for $i \in [0,1]$ except the leftmost a_1 has some descendants from $\mathcal{Z}_i$ in some sentential forms during the derivation. Indeed, by Claim 1, each such a_i has descendants from $\mathcal{D}_i$. And, such descendants may be inserted only by elements of $\mathcal{Y}_i$.

Let a_i for $i \in [0,1]$ be a symbol from A which appears in the input word, not the leftmost a_1. By Claim 1, this a_i has an descendant from $\mathcal{D}_i$ which will be deleted by x. As x and possible descendants of a_{1-i} are not able to delete elements of $\mathcal{Z}_{1-i}$, if a_{1-i} located directly to the left of this a_i has descendants from $\mathcal{Z}_{1-i}$, then all these descendants should be deleted by descendants of this a_i. (Claim 2) □

Claim 3 Let a_i for $i \in [0,1]$ be a symbol which appears in the input word. Then, a sequence of its descendants in each sentential form of the derivation

which ends at x belongs to the set

$$(\{B_i\} \cup \mathcal{D}_i)^* \mathcal{Z}_i^* \{B_i\}^*.$$

Proof of Claim 3 For the sake of contradiction, assume that a symbol from $\mathcal{D}_i$ appears between symbols from $\mathcal{Z}_i$ or to the right of them in some sentential form. According to Claim 1, the rightmost descendant of a_i from the set $\mathcal{D}_i$ will be deleted by x. As x does not insert any symbols, and descendants of a_i cannot delete elements of $\mathcal{Z}_i$, the descendants of a_i from $\mathcal{Z}_i$ located to the left of the rightmost descendant from $\mathcal{D}_i$ should be deleted by x. But there are no delete rule in which x deletes any element of $\mathcal{Z}_i$. (Claim 3) □

By Proposition 4, there exists a leftmost derivation $wx \Rightarrow^* x$ for each $w \in L(G)$. So, let us consider only leftmost derivations which start from $w = (a_1 a_0)^m (F_0 F_1)^n x$. Remind that a_i (for $i \in [0,1]$) and its descendants are not able to insert symbols which could delete elements of $\mathcal{Z}_i$. By Claim 1, the factor $Y_{0,0}X_{0,0}$ should appear in the sequence of descendants of the leftmost a_0. Indeed, this a_0 has to insert an element of $\mathcal{D}_0$ (see Claim 1) and the only way to insert such an element is by inserting $X_{0,0}$ which inserts $Y_{0,0}$. Now, we show by induction that the sequence of descendants of the pth symbol from A (for $p > 2$), say a_i, contains (in some sentential form) a subsequence $(Y_{i,1}X_{i,1}Y_{i,0}X_{i,0})^{2^{p-3}}$ which is not deleted as long as this a_i is not deleted. The latter statement follows from the fact that a_i and its descendants are not able to insert symbols which delete the elements of $\mathcal{Z}_i$. Let $p = 3$, that is, we consider the third symbol from A, the second a_1. As the first (leftmost) a_0 inserted the subsequence $Y_{0,0}X_{0,0}$, the descendants of the second a_1 should insert the symbols which delete $Y_{0,0}X_{0,0}$ (see Claim 2). That is, the subsequence $D_{1,0}D_{1,1}$ should appear among the descendants of the second a_1 in some sentential form (as $D_{1,0}$ is the only symbol which deletes elements of $\mathcal{X}_0$, $D_{1,1}$ is the only symbol which deletes elements of $\mathcal{Y}_0$ and $D_{1,1}$ is the only possible descendant of a_1, which deletes $D_{1,0}$). However, in order to insert this sequence to the left of all elements of $\mathcal{Z}_1$ (which are descendants of this a_i) – see Claim 3, a subsequence

$$Y_{1,1}X_{1,1}Y_{1,0}X_{1,0} = (Y_{1,1}X_{1,1}Y_{1,0}X_{1,0})^{2^{3-3}}$$

should be inserted.

Now, assume that the statement is true for $p-1 < 2n$. Thus, the pth element of A, say a_i, and its descendants should insert symbols which are able to delete the sequence $(Y_{1-i,1}X_{1-i,1}Y_{1-i,0}X_{1-i,0})^{2^{p-4}}$ (see Claim 2). Note that elements of $\mathcal{X}_{1-i}$ can be deleted only by $D_{i,0}$ or F_0. However, F_0 cannot be an descendant of a_i. Similarly, elements of $\mathcal{Y}_{1-i}$ can be deleted only by $D_{i,1}$ or F_1, but F_1 cannot be an descendant of a_i. As the descendants of a particular a_i have to delete

$$(Y_{1-i,1}X_{1-i,1}Y_{1-i,0}X_{1-i,0})^{2^{p-4}} \in (\mathcal{Y}_{1-i}\mathcal{X}_{1-i})^{2^{p-3}},$$

the derivation should contain a subsequence of sentential forms $v_1, \ldots, v_{2^{p-2}}$ such that the leftmost descendant of the considered a_i in v_j is equal to $D_{i,(j-1) \bmod 2}$. Indeed, only the leftmost descendant of a symbol a is able to delete a symbol

which is not an descendant of this a. Moreover, by Claim 3, only the leftmost descendants of a_i which belongs to $\mathcal{Z}_i$ is allowed to insert an element of $\mathcal{D}_i$. And, no descendant of a_i which belongs to $\mathcal{Z}_i$ is deleted until this a_i is deleted (because it is not possible that an descendant of a_i is able to delete an element of $\mathcal{Z}_i$). So, in order to obtain first $D_{i,0}$ (as the leftmost descendant of a_i in v_1), a subsequence $Y_{i,0}X_{i,0}$ should be inserted. Further, assume that $D_{i,(j-1) \bmod 2}$ is the leftmost descendant of a_i in v_j. In order to obtain $D_{i,j \bmod 2}$ as the leftmost descendant of a_i, it is needed that at least $Y_{i,(j-1) \bmod 2}$ which inserted $D_{i,(j-1) \bmod 2}$ inserts $X_{i,j \bmod 2}$ which in turn inserts $Y_{i,j \bmod 2}$ and it finally inserts $D_{i,j \bmod 2}$. However, it means that the pth element of A (and its descendants) inserts a subsequence $(Y_{i,1}X_{i,1}Y_{i,0}X_{i,0})^{2^{p-3}}$ which is not deleted by its descendants.

So, finally, the rightmost element of A, that is a_0 which is the $(2m)$th element, inserts a subsequence $(Y_{0,1}X_{0,1}Y_{0,0}X_{0,0})^{2^{2m-3}}$ that will not be deleted by its descendants. Similarly as in Theorem 2, at least one copy of $\mathcal{F}$ from the sequence $(F_0F_1)^n$ is needed to delete one symbol from the sequence $(Y_{0,1}X_{0,1}Y_{0,0}X_{0,0})^{2^{2m-3}}$ (as F_0 deletes only $X_{0,1}, X_{0,0}$ and F_1 deletes only $Y_{0,1}, Y_{0,0}$. Thus, it is required that $n \geq 2^{2m-2}$. □

Theorem 6. *The language $L(G)$ is not a context-free language.*

Proof. For the sake of contradiction, assume that $L(G) \in \mathsf{CFL}$. As CFL is closed under intersection with regular languages, the language $L' = L \cap (a_1a_0)^+(F_0F_1)^+$ is a context-free language as well. However, by Propositions 14 and 15, L' is equal to the non context-free language

$$\{(a_1a_0)^m(F_0F_1)^n \mid n \geq 2^{2m-2}\}.$$

□

9 Conclusions and Open Problems

We have investigated expressive power of leftist grammars, the naturally defined class of grammars, having interesting connections to the accessibility problem in protection systems. Here, we have introduced a classification of theses grammars according to the structure of so-called insert graphs and delete graphs. We have placed all but the most general class of this classification into the Chomsky hierarchy. For general leftist grammars we have shown that they are able to define languages which are not context free, answering the open problem from [9].

From practical point of view, the most interesting further research direction concerns the existence of efficient algorithms for the membership problem of general leftist grammars. More precisely, is this problem in PSPACE? Is it in NP or in P? Concerning the placement of the leftist grammars into the Chomsky hierarchy, a natural question is whether the set of languages defined by leftist grammars is included in the set of context sensitive languages.

References

1. M. Blaze, J. Fegenbaum, J. Ioannidis, A. Keromytis. The role of trust management in distributed security. In J. Vitekand C. Jensen, editors, Secure Internet Programming: Security Issues for Distributed and Mobile Objects, LNCS 1603, Springer, 185–210.
2. T. Budd. Safety in grammatical protection systems. *International Journal of Computer and Information Sciences*, 12(6):413–430, 1983.
3. O. Cheiner, V. Saraswat. Security Analysis of Matrix. Technical report, AT&T Shannon Laboratory, 1999.
4. M. Harrison. Introduction to Formal Language Theory. Addison-Wesley, 1978.
5. M. Harrison, W. Ruzzo, J. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–470, August 1976.
6. J. Hopcroft, R. Motwani, J. Ullman. Introduction to Automata Theory, Languages, and Computation. *Addison-Wesley*, 2000.
7. R. Lipton, T. Budd. On Classes of Protection Systems. In *Foundations of Secure Computation*, Academic Press, 1978, pp. 281–296.
8. R. Lipton, L. Snyder. A linear time algorithm for deciding subject security. *Journal of the ACM*, 24(3):455–464, July 1977.
9. Rajeev Motwani, Rina Panigrahy, Vijay A. Saraswat, Suresh Ventkatasubramanian. On the decidability of accessibility problems (extended abstract). *STOC 2000*, 306–315.
10. V. Saraswat. The Matrix Design. Technical report, AT&T Laboratory, April 1997.