Liliana Cojocaru and Erkki Mäkinen

On the Complexity of Szilard languages of Regulated Grammars

DEPARTMENT OF COMPUTER SCIENCES
UNIVERSITY OF TAMPERE

D-2010-12

TAMPERE 2010

UNIVERSITY OF TAMPERE
DEPARTMENT OF COMPUTER SCIENCES
SERIES OF PUBLICATIONS D – NET PUBLICATIONS
D-2010-12, OCTOBER 2010

Liliana Cojocaru and Erkki Mäkinen

On the Complexity of Szilard languages of Regulated Grammars

DEPARTMENT OF COMPUTER SCIENCES
FIN-33014 UNIVERSITY OF TAMPERE

ISBN 978-951-44-8278-6
ISSN 1795-4274

On the Complexity of Szilard Languages of Regulated Grammars

Liliana Cojocaru
University of Tampere
Department of Computer Sciences
Tampere, Finland
cslico@uta.fi

Erkki Mäkinen
University of Tampere
Department of Computer Sciences
Tampere, Finland
em@cs.uta.fi

Abstract

The *regulated rewriting* mechanism is one of the most efficient methods to augment the Chomsky hierarchy with a large variety of language classes that lay within it. In this paper we investigate the derivation process in regulated rewriting grammars such as *matrix grammars*, *random context grammars*, and *programmed grammars* by studying their Szilard languages. We prove that Szilard languages associated with unrestricted derivations in these grammars can be recognized in logarithmic time and space by indexing alternating Turing machines. Hence, these classes of Szilard languages belong to the U_{E^*}-uniform $\mathcal{NC}^1$ class [41]. In general, leftmost Szilard languages of regulated rewriting grammars can be recognized in logarithmic space and square logarithmic time. Hence, these classes of languages belong to the $\mathcal{NC}^2$ class [41].

1 Introduction

When we consider a formal grammar one of the very first tasks is to study the *derivation mechanism* of the system in question. Once derivation properties have been settled on, we can go further by studying closure properties, decidability properties, or the computational power of that generative device. One of the most important tools to investigate the derivation mechanism in formal language theory, is the *Szilard language*. If labels are associated with productions in one-to-one correspondence, then each terminal derivation can be expressed as a word over the set of labels, such that labels in this word are concatenated in the same order they have been used during the derivation. Informally, the Szilard language associated with a generative device is the set of all words obtained in this way.

The concept of Szilard language has been first introduced for Chomsky grammars, under the name of "label language", "associate language", or "derivation language", in [20], [35], [37], and [44]. Roots of Szilard languages come from [2] and [43]. The notion has been extended afterwards for several other generative devices, such as pure context-free grammars [31] and regulated rewriting grammars [14], [16], [36], and [43]. If restrictions are imposed on the derivation order then particular classes of Szilard languages, such as *leftmost Szilard languages* [28], *canonical label languages* [6], *depth-first* and *breadth-first Szilard languages* [30] are obtained.

Hierarchies and closure properties of Szilard languages associated with (pure) context-free grammars are considered in [37], [44], [31], [32], and [33]. Szilard languages of (pure) context-free grammars are very weak in closure properties. They are not closed under union, concatenation, homomorphism and inverse homomorphism, Kleene +, or intersection with regular languages. Hence, none of them form even a *trio family*[1] of languages. In [43] it is proved that the closure of Szilard languages of context-free grammars under the intersection with regular languages equals the family of derivation languages associated with context-free matrix grammars. Another characterization of context-free matrix languages by means of Szilard languages is provided in [11]. There exists a proper hierarchy of Szilard languages of pure context-free grammars with respect to the degree of grammars [31]. There exits also a proper hierarchy of Szilard languages of context-free grammars with respect to a certain homomorphism [32].

Decidability properties of Szilard languages associated with context-free grammars are investigated in [25], [27], [32], [35], and [37]. The emptiness, finiteness, and equivalence problems are decidable for these languages [37]. The inclusion problem for leftmost Szilard languages is decidable [27], [35], and for unrestricted Szilard languages it is NP-complete [32]. The fitting problem [25] and the left fitting problem [26], i.e., whether a given leftmost Szilard language is in the family of Szilard languages, are decidable, too. Several operations on Szilard languages and semilinearity properties of these languages are studied in [21] and [29], respectively.

Time and space bounds of a Turing machine or multicounter machine to recognize Szilard languages associated with Chomsky grammars, are presented in [38] and [24]. In [38] it is proved that (leftmost) Szilard languages of context-free grammars can be recognized by a *linear bounded*[2] (realtime) multicounter machine. Since each realtime multicounter machine can be simulated by a deterministic *off-line*[3] Turing machine with logarithmic space, in terms of the length of the input string [19], it follows that the classes of Szilard languages and (leftmost) Szilard languages associated with context-free grammars are contained[4] in DSPACE($\log n$). In [9] we strengthened this result by proving that the above classes of Szilard languages can be accepted by an indexing alternating Turing machine (henceforth indexing ATM) in logarithmic time and space. Since the class of languages recognizable by an indexing ATM in logarithmic time equals the U_{E^*}-uniform $\mathcal{NC}^1$ class [41], we obtain that the above classes of Szilard languages are strictly contained in $\mathcal{NC}^1$, i.e., the class of Boolean functions computable by polynomial size Boolean circuits, with depth $\mathcal{O}(\log n)$ and constant fan-in [46].

Characterizations of (leftmost) Szilard languages of context-free and phrase-

[1] A family of languages is called *trio* if it is closed under λ-free homomorphism, inverse homomorphism, and intersection with regular languages.

[2] A multicounter machine is *linear bounded* if it works in *realtime*, i.e., there exists a constant k such that during the computation the contents of each counter is less than $k|w|$, where $|w|$ is the length of the input string.

[3] An *off-line* Turing machine is a Turing machine equipped with a read-only input tape and a read-write working tape. It is allowed to shift both heads on both directions, and it works similar to a Turing machine.

[4] DSPACE($\log n$), or the L class, is the class of languages recognizable by an *off-line* deterministic Turing machine using logarithmic space.

structure (unrestricted) grammars in terms of Turing machine resources are provided in [24]. It is proved that $\log n$ is the optimal space bound for an *on-line*[5] deterministic Turing machine to recognize (leftmost) Szilard languages of context-free grammars. It is also an optimal bound for an off-line deterministic Turing machine to recognize leftmost Szilard languages of phrase-structure grammars. However, the optimal bound for an on-line deterministic Turing machine to recognize leftmost Szilard languages of context-free and phrase-structure grammars is n, where n is the length of the input word. Since leftmost Szilard languages of phrase-structure grammars are off-line recognizable by a deterministic Turing machine that uses only logarithmic space, in terms of the input string, leftmost Szilard languages of phrase-structure grammars are included in DSPACE($\log n$). In [9] we proved that the class of leftmost Szilard languages of phrase-structure grammars is strictly included in $\mathcal{NC}^1$ under the U_{E^*}-uniformity restriction.

Regulated grammars are formal grammars composed of Chomsky rules for which the derivation mechanism obeys several filters and controlling constraints that allow or prohibit the use of the rules during the generative process. For formal definitions and results concerning grammars with regulated rewriting the reader is referred to [14]. In this paper we deal with three types of rewriting mechanisms provided by *matrix grammars*, *random context grammars*, and *programmed grammars*. These grammars are equivalent concerning their generative power [14], but they are interesting because each of them uses totally different regulating restrictions in the derivation mechanism that provide good structures to handle a large variety of problems in formal languages, computational linguistics, programming languages, and even graph theory.

This work is dedicated to the complexity of Szilard languages associated with these three types of regulated grammars. The main aim is to relate the corresponding classes of Szilard languages to parallel complexity classes, such as $ALOGTIME$, $\mathcal{NC}^1$, and $\mathcal{NC}^2$, where $ALOGTIME$ is the class of languages recognizable by an indexing ATM in $\log n$ time [4], [7]. Approaching Szilard languages to low complexity classes, such as $\mathcal{NC}^1$ and $\mathcal{NC}^2$, is the most natural way to relate these classes to circuit complexity classes [46], and thus bringing new insights in finding fast parallel algorithms to recognize classes of languages generated by the above regulated mechanisms. Based on the method used in [9] we prove that unrestricted Szilard languages associated with matrix, programmed, and random context grammars are contained in the U_{E^*}-uniform $\mathcal{NC}^1$ class. In general, leftmost Szilard languages of regulated rewriting grammars can be recognized in logarithmic space and square logarithmic time. Hence, these classes of leftmost Szilard languages belong to the $\mathcal{NC}^2$ class [41].

The paper is structured as follows. In Section 2 we introduce the main notions concerning Chomsky grammars and the Chomsky hierarchy. We also present several complexity results of (leftmost) Szilard languages associated with context-free and phrase-structure grammars. In Section 3 we present complexity results for Szilard languages associated with matrix grammars. Section 4 is dedicated to the complexity of Szilard languages of random context grammars, while in Section 5 we investigate

[5] An *on-line* Turing machine is an off-line Turing machine with the restriction that the input head cannot be shifted to the left.

the complexity of Szilard languages associated with programmed grammars. We conclude in Section 6 with some remarks on Szilard languages of regulated grammars with context-sensitive and phrase-structure rules.

2 Chomsky Grammars and Szilard Languages - Prerequisites

Chomsky grammars [8] have played a crucial role in the field of theoretical computer science, especially in formal languages and programming languages. In this section we introduce the main notions and notations that concern Chomsky grammars and the Chomsky hierarchy. We briefly present several complexity results that concern Szilard languages associated with Chomsky grammars. We assume the reader to be familiar with the basic notions of formal language theory [34], [43].

Let X be a finite nonempty alphabet. We denote by λ the empty string, by $|x|_a$ the number of occurrences of the letter a in the string x, and by $|x|$ the length of $x \in X^*$. We denote by $|X|$ the cardinality of the set X.

Definition 1 A *phrase-structure* (PS) or *Chomsky grammar* (CG) is a quadruple $G = (N, T, P, S)$, where N and T, $N \cap T = \emptyset$, are finite sets of *nonterminals* and *terminals*, respectively. $S \in N - T$ is the *axiom*, and P is a finite set of rules of the form $\alpha \to \beta$, $\alpha \in (N \cup T)^* N (N \cup T)^*$ and $\beta \in (N \cup T)^*$.

In the sequel for any *phrase-structure* rule p of the form $\alpha \to \beta$, α and β are called the *left-hand side* and the *right-hand side* of p, respectively. If $\beta \in T^*$, then p is called *terminal* rule. Otherwise, p is called *non-terminal* rule. If $\beta = \lambda$, then p is called *erasing* rule.

Definition 2 Let $G = (N, T, P, S)$ be a phrase-structure grammar (PSG) and let $x, y \in (N \cup T)^*$. We say that x *directly derives* y, written as $x \Rightarrow_G y$, if there exist $\alpha_1, \alpha_2, \alpha, \beta \in (N \cup T)^*$, such that $x = \alpha_1 \alpha \alpha_2$, $y = \alpha_1 \beta \alpha_2$, and $\alpha \to \beta \in P$. We denote by $\Rightarrow_G^*$ the reflexive and transitive closure of $\Rightarrow$. The language generated by G is defined as $L(G) = \{w | w \in T^*, S \Rightarrow_G^* w\}$.

Definition 3 Let $G = (N, T, P, S)$ be a PSG.

1. If no restrictions are imposed on rules in P then G is also called *recursively-enumerable* or *unrestricted (type 0)* grammar.

2. If each rule in P is of the form $\alpha A \gamma \to \alpha \beta \gamma$, where $A \in N$, $\alpha, \gamma \in (N \cup T)^*$, $\beta \in (N \cup T)^+$, then G is a *context-sensitive (type 1)* grammar. Moreover, G may contain the rule $S \to \lambda$, assuming that S does not occur on the right-hand side of any rule in P.

3. If each rule in P is of the form $\alpha \to \beta$, $|\alpha| \leq |\beta|$, then G is a *monotonous (type 1)* grammar. Moreover, the grammar may contain the rule $S \to \lambda$, assuming that S does not occur on the right-hand side of any rule in P.

4. If each rule in P is of the form $\alpha \to \beta$, $\alpha \in N$ and $\beta \in (N \cup T)^*$, then G is a *context-free (type 2)* grammar.

5. If each rule in P is of the form $\alpha \to \beta$, $\alpha \in N$ and $\beta \in T^* \cup T^*N$, then G is a *regular (type 3)* grammar.

Note that, the definitions of a type 1 grammar provided at items 2 and 3 are equivalent, in the sense that the grammars generate the same class of languages. We denote by REG, CFG, CSG, and PSG the set of all regular (type 3), context-free (type 2), context-sensitive or monotonous (type 1), and phrase-structure (type 0) grammars, respectively. The classes of languages generated by $REGs$, $CFGs$, $CSGs$, and $PSGs$ are denoted by $REGL$, CFL, CSL, and PSL, respectively. The class PSL equals the class of recursively enumerable languages, also denoted by RE. Between these classes of languages the next inclusions (*Chomsky hierarchy*) hold $REGL \subset CFL \subset CSL \subset RE$.

If rules in a CG are uniquely labeled, then each terminal derivation[6] in the grammar can be expressed as a unique word over the set of all labels. Informally, the Szilard (control) word associated with a terminal derivation in a CG, is obtained by concatenating the labels of components in the same order they have been used during the derivation. The Szilard language associated with a CG is the set of all words obtained in this way. In the sequel, for the sake of simplicity, we use the same notation both for a rule and the label associated with it.

Definition 4 Let $G = (N, T, S, P)$ be a CG, $P = \{p_1, p_2, ..., p_k\}$ the set of productions, $L(G)$ the language generated by G, and w a word in $L(G)$. The *Szilard word* of w associated with the derivation D: $S = w_0 \Rightarrow_{p_{i_1}} w_1 \Rightarrow_{p_{i_2}} ... \Rightarrow_{p_{i_s}} w_s = w$ is defined as $Sz_D(w) = p_{i_1}p_{i_2}...p_{i_s}$, $p_{i_j} \in P$, $1 \leq j \leq s$. The *Szilard language* of G is $Sz(G) = \{Sz_D(w) | w \in L(G), D \text{ is a derivation of } w\}$.

Definition 5 Let $G = (N, T, S, P)$ be a CG. A terminal derivation D: $S = w_0 \Rightarrow_{p_{i_1}} w_1 \Rightarrow_{p_{i_2}} ... \Rightarrow_{p_{i_s}} w_s = w$ is a leftmost derivation of w, if for each $1 \leq j \leq s$, $w_{j-1} = u_{j-1}\alpha_j v_{j-1} \Rightarrow_{p_{i_j}} u_{j-1}\beta_j v_{j-1} = w_j$, $u_{j-1} \in T^*$, where p_{i_j} is the rule $\alpha_j \to \beta_j$ in P. The *leftmost Szilard language* of a grammar G is $Sz_{left}(G) = \{Sz_D(w) | w \in L(G), D$ is a leftmost derivation of $w\}$.

Consider $SZ(X) = \{Sz(G) | G \text{ is an X-grammar}\}$ and $SZL(X) = \{Sz_{left}(G) | G \text{ is an X-grammar}\}$, the classes of Szilard languages and leftmost Szilard languages associated with X-grammars, where $X \in \{REG, CF, CS, PS\}$. It is well known that $SZ(REG) \subset REGL$, $SZ(CF)$ and CFL are incomparable, $SZ(PS) \subset CSL$ and $SZL(PS) \subset CFL$. Concerning the time and space of Turing machines recognizing Szilard languages, the best upper bounds known so far are $SZ(PS) \subseteq NTIME(n^2)$, $SZ(CF) \subseteq DSPACE(\log n)$, and $SZL(PS) \subseteq DSPACE(\log n)$ [24], [38].

An indexing ATM [7] is an alternating Turing machine that is allowed to write any binary number on a special tape, called *index* tape. This number is interpreted as an address of a location on the input tape. With i, written in binary on the index

[6] That is a derivation that leads to a word in the language.

tape, the machine can read the symbol placed on the i^{th} cell of the input tape. Using universal states to relate different branches on the computation, an indexing ATM can read an input string of length n, in $\mathcal{O}(\log n)$ time. For the formal definition and complexity results on ATMs the reader is referred to [4], [7], and [41].

The next results concerning the Szilard languages of CFGs, CSGs, and PSGs are provided in [9].

Theorem 1 *Each language $L \in X$, $X \in \{SZ(CF), SZL(CF), SZL(CS), SZL(PS)\}$ can be recognized by an indexing ATM in $\mathcal{O}(\log n)$ time and space.*

As a consequence of Theorem 1 and the properties of ATMs [41], we have

Theorem 2 $SZ(CF), SZL(CF), SZL(CS), SZL(PS) \subset \mathcal{NC}^1 \subseteq DSPACE(\log n)$.

Due to the weak restrictions imposed on the types of rules and derivation mechanism, the Chomsky hierarchy is a sparse hierarchy. However, if restrictions are imposed on rules and on the classical derivation mechanism in CGs, this hierarchy can be substantially augmented with a rich variety of language classes. A possibility to achieve this goal is to make use of *regulated rewriting mechanisms* which consists of several filtering and controlling constraints imposed on derivations. These constraints may allow or forbid some derivations to develop, by generating terminal strings. For formal definitions and results concerning the large variety of regulated rewriting mechanisms the reader is referred to [12], [14], and [15].

In the sequel we only deal with *matrix grammars*, *random context grammars*, and *programmed grammars*. We describe the derivation mechanism for these regulated rewriting grammars and we present new results concerning the complexity of the corresponding Szilard languages.

3 Szilard Languages of Matrix Grammars

Matrix grammars (MGs) are regulated rewriting grammars in which rules are grouped into *matrices*. A matrix can be applied if all its rules can be applied one by one according to the order they occur in the matrix sequence. In the case of MGs with appearance checking a rule in a matrix can be passed over if its left-hand side does not occur in the sentential form and the rule belongs to a special set of rules defined within the MG. MGs with context-free rules have been first defined in [1] in order to increase the generative power of CFGs. The definition has been extended for the case of phrase-structure rules in [14]. The generative power of these devices has been studied in [12], [14], and [15]. Formally, a MG is defined as follows.

3.1 Matrix Grammars - Prerequisites

Definition 6 A *matrix grammar with appearance checking* (MG^{ac}) is a quintuple $G = (N, T, S, M, F)$ where S is the axiom, N and T, $N \cap T = \emptyset$, are finite sets of *nonterminals* and *terminals*, respectively, $M = \{m_1, m_2, ..., m_k\}$ is a finite set of finite sequences of rules of the form $m_j = (p_{m_j,1}, p_{m_j,2}, ..., p_{m_j,k_{m_j}})$, where each $p_{m_j,i}$ is an unrestricted rule over $N \cup T$, $1 \leq i \leq k_{m_j}$, $k_{m_j} \geq 1$, $1 \leq j \leq k$, and F is a subset

of the rules occurring in the elements of M, i.e., $F \subseteq \{p_{m_j,r} | 1 \leq j \leq k, 1 \leq r \leq k_{m_j}\}$. A *matrix grammar without appearance checking* has $F = \emptyset$.

Note that, if all rules in M are phrase-structure (PS), context-sensitive (CS), context-free (CF), or regular (REG) rules then G is a PS, CS, CF, or REG matrix grammar, respectively.

Definition 7 Let $G = (N, T, S, M, F)$ be a MG^{ac} and $V = N \cup T$. We say that $x \in V^+$ directly derives $y \in V^*$ in *appearance checking mode* by application of a rule p of the form $\alpha \to \beta$, $\alpha \in (N \cup T)^* N (N \cup T)^*$ and $\beta \in (N \cup T)^*$, denoted by $x \Rightarrow_p^{ac} y$, if one of the following conditions holds *i.* $x = x_1 \alpha x_2$ and $y = x_1 \beta x_2$, or *ii.* rule $\alpha \to \beta$ is not applicable to x, i.e., α is not a substring of x, $p \in F$, and $x = y$.

Note that, if rule p in Definition 7 satisfies condition *i.*, then we say that p is effectively applied. For the case of MGs without appearance checking only condition *i.* has to be checked, and then instead of $x \Rightarrow_p^{ac} y$ the notation $x \Rightarrow_p y$ is used.

Definition 8 Let $G = (N, T, S, M, F)$ be a MG^{ac} (or a MG if $F = \emptyset$) and $V = N \cup T$. For $m_j = (p_{m_j,1}, p_{m_j,2}, ..., p_{m_j,k_{m_j}})$, $k_{m_j} \geq 1$, $1 \leq j \leq k$, and $x, y \in V^*$, we define a derivation step in G, denoted as $x \Rightarrow_{m_j} y$, by $x = x_0 \Rightarrow_{p_{m_j,1}}^{ac} x_1 \Rightarrow_{p_{m_j,2}}^{ac} x_2 \Rightarrow_{p_{m_j,3}}^{ac} ... \Rightarrow_{p_{m_j,k_{m_j}}}^{ac} x_{k_{m_j}} = y$ (and by $x = x_0 \Rightarrow_{p_{m_j,1}} x_1 \Rightarrow_{p_{m_j,2}} x_2 \Rightarrow_{p_{m_j,3}} ... \Rightarrow_{p_{m_j,k_{m_j}}} x_{k_{m_j}} = y$ if $F = \emptyset$). The *language* $L(G)$ generated by G is defined as the set of all words $w \in T^*$ such that there is a derivation $D : S \Rightarrow_{m_{j_1}} y_1 \Rightarrow_{m_{j_2}} y_2 \Rightarrow_{m_{j_3}} ... \Rightarrow_{m_{j_q}} w$, $1 \leq j_i \leq k$, $1 \leq i \leq q$.

If we denote by $L(M, X)$ and $L(M, X, ac)$ the class of languages generated by MGs and MGs with appearance checking, respectively, with X-rules[7], $X \in \{REG, CF, CF - \lambda, CS, PS\}$, then the following inclusions hold [12], [14], [15].
1. $CFL \subset L(M, CF - \lambda) \subset L(M, CF - \lambda, ac) \subset CSL \subset L(M, CF, ac) = RE$,
2. $CFL \subset L(M, CF - \lambda) \subseteq L(M, CF) \subset L(M, CF, ac) = RE$,
3. $L(M, X) = L(M, X, ac) = XL$, $X \in \{REG, CS, PS\}$.

Since rules in a MG are arranged into matrices, and rules inside each matrix are applied in a predefined order, for the case of MGs it is more convenient to associate labels with matrices than with rules. In this manner each terminal derivation in a MG can be expressed as a word over the set of labels associated in one-to-one correspondence with matrices in the grammar, such that labels are concatenated in the same order they have been used during the derivation. Informally, the Szilard language associated with a MG is the set of all words obtained in this way. In the sequel, for the sake of simplicity, we use the same notation both for a matrix and the label associated with it. Formally, we have

Definition 9 Let $G = (N, T, S, M, F)$ be a MG, $M = \{m_1, m_2, ..., m_k\}$ the set of matrices of G, $L(G)$ the language generated by G, and $w \in L(G)$. The *Szilard word* of w associated with the derivation D: $S \Rightarrow_{m_{i_1}} y_1 \Rightarrow_{m_{i_2}} y_2 \Rightarrow_{m_{i_3}} ... \Rightarrow_{m_{i_q}} w$ in G is defined as $Sz_D(w) = m_{i_1} m_{i_2} ... m_{i_q}$, $m_{i_j} \in M$, for some $q \geq 1$, $1 \leq i_j \leq k$, $1 \leq j \leq q$. The *Szilard language* of G is $Sz(G) = \{Sz_D(w) | w \in L(G), D \text{ is a derivation of } w\}$.

[7] By $CF - \lambda$-rule we denote a non-erasing context free rule, i.e., a rule of the form $\alpha \to \beta$, where $\alpha \in N$, $\beta \in (N \cup T)^+$.

We denote by $SZM(X)$ and $SZM^{ac}(X)$ the classes of Szilard languages associated with matrix grammars and matrix grammars with appearance checking with X rules, $X \in \{CF, CS, PS\}$, respectively.

Note that with respect to matrices, MGs are nondeterministic devices. At each step of derivation the grammar nondeterministically chooses which matrix is applied, if this can be applied. Once a matrix becomes active, it works deterministically, in the sense that the order in which the rules are applied is predefined by their order in the matrix sequence. However, the order in which multiple occurrences of a nonterminal in a sentential form are rewritten, is still nondeterministically chosen. A possibility to reduce the high nondeterminism in MGs is to impose an order on which nonterminals occurring in a sentential form can be rewritten. As in the case of CGs, the most significant is the leftmost derivation order [13], [14], [18], [42]. In this paper we focus only on three types of leftmost derivation, defined in [14] for MGs with context-free rules, as follows.

Definition 10 Let $G = (N, T, S, M, F)$ be a MG. A derivation in G is called

- *leftmost-1* if each rule used in the derivation rewrites the leftmost nonterminal occurring in the current sentential form,
- *leftmost-2* if at each step of derivation the leftmost occurrence of a nonterminal which can be rewritten (by first rules of matrices that can be effectively applied, with no restrictions on the other rules) is rewritten,
- *leftmost-3* if each rule used in the derivation rewrites the leftmost occurrence of its left-hand side in the current sentential form.

Note that, the above definition is universally applicable for regulated rewriting grammars such as random context or programmed grammars.

In terms of matrices, for the case of leftmost-1 derivation, in MGs without appearance checking, each rule in the sequence that defines a matrix must rewrite the leftmost nonterminal occurring in the current sentential form (otherwise the matrix cannot be applied in the leftmost-1 derivation manner). In the case of MGs with appearance checking, if a certain rule of a matrix cannot rewrite the leftmost nonterminal, because the nonterminal rewritten by the rule does not occur in the sentential form, then the rule is passed over if it belongs to F. Otherwise, i.e., the nonterminal rewritten by the rule occurs in the sentential form, but this is not the leftmost nonterminal in the sentential form, then the rule, hence the matrix, cannot be applied in the leftmost-1 derivation manner.

A matrix is applicable in leftmost-2 derivation manner if the first rule in the matrix sequence rewrites the leftmost nonterminal that can be rewritten by a matrix. Hence, a matrix m_j can be applied in leftmost-2 derivation manner, if the first rule of m_j (that can be effectively applied, for the case of appearance checking) rewrites the first occurrence of a nonterminal X, and no other matrix $m_{j'}$ exists such that the first rule in $m_{j'}$ (that can be effectively applied) rewrites a nonterminal X', where X' occurs before X in the sentential form on which m_j is applied. No other restrictions are imposed on the other rules of m_j.

A matrix is applicable in leftmost-3 derivation manner if each rule of the matrix rewrites the leftmost occurrence of its left-hand side occurring in the sentential form. If a certain rule in the matrix sequence cannot rewrite the leftmost occurrence of its left-hand side (because this does not occur in the sentential form) then the rule is passed over if this belongs to F.

Szilard languages associated with leftmost-i, $i \in \{1,2,3\}$, derivations are defined in the same way as in Definition 9, with the specification that D is a leftmost-i derivation of w. We denote by $SZML_i(X)$ and $SZML_i^{ac}(X)$ the classes of leftmost-i, $i \in \{1,2,3\}$, Szilard languages associated with MGs and MGs with appearance checking with X rules, $X \in \{CF, CS, PS\}$, respectively.

Henceforth, in any reference to a MG $G = (N, T, A_1, M, F)$, A_1 is considered to be the axiom, $N = \{A_1, A_2, ..., A_m\}$ the ordered finite set of nonterminals, and $M = \{m_1, m_2, ..., m_k\}$ the ordered finite set of labels associated with matrices in M. Each matrix m_j, $1 \leq j \leq k$, is a sequence of the form $m_j = (p_{m_j,1}, p_{m_j,2}, ..., p_{m_j,k_{m_j}})$, $k_{m_j} \geq 1$. Unless otherwise specified (see Chapter 6), each $p_{m_j,r}$, $1 \leq r \leq k_{m_j}$, is a context-free rule of the form $\alpha_{m_j,r} \rightarrow \beta_{m_j,r}$, $\alpha_{m_j,r} \in N$ and $\beta_{m_j,r} \in (N \cup T)^*$. If $\beta_{m_j,r} \in T^*$, then $p_{m_j,r}$ is called a *terminal* rule. Otherwise, $p_{m_j,r}$ is called a *non-terminal* rule.

We define the *net effect* of rule $p_{m_j,r}$, $1 \leq r \leq k_{m_j}$, with respect to nonterminal $A_l \in N$, $1 \leq l \leq m$, by the difference $df_{A_l}(p_{m_j,r}) = |\beta_{m_j,r}|_{A_l} - |\alpha_{m_j,r}|_{A_l}$.

If G is a MG without appearance checking, then the *net effect* of matrix m_j with respect to nonterminal $A_l \in N$, $1 \leq l \leq m$, is the sum $s_{A_l}(m_j) = \Sigma_{r=1}^{k_{m_j}} df_{A_l}(p_{m_j,r})$. To each matrix m_j we associate a vector $V(m_j) \in \mathbf{Z}^m$ defined by $V(m_j) = (s_{A_1}(m_j), s_{A_2}(m_j), ..., s_{A_m}(m_j))$. Depending on the context, the value of $V(m_j)$ taken at the l^{th} place, $1 \leq l \leq m$, i.e., $V_l(m_j)$, is also denoted by $V_{A_l}(m_j) = s_{A_l}(m_j)$.

If G is a MG with appearance checking, then a *policy* of a matrix $m_j \in M$, denoted by ℓ_j^q, is a choice of m_j of using, during the derivation, a certain subsequence of matrix m_j obtained by dropping out some rules that occur in the same time in m_j and F. Hence, if the policy ℓ_j^q is defined by the subsequence $m_j^q = (p_{m_j,1}, p_{m_j,2}, ..., p_{m_j,\xi_{m_j}^q})$ of m_j, then rules in m_j^q occur in the same order they occur in m_j. The *net effect* of matrix m_j, with respect to policy ℓ_j^q and the nonterminal $A_l \in N$, is defined by $s_{A_l}(\ell_j^q) = \Sigma_{r=1}^{\xi_{m_j}^q} df_{A_l}(p_{m_j,r})$. To each policy ℓ_j^q, identified by the sequence m_j^q, we associate a vector $V(\ell_j^q) \in \mathbf{Z}^m$ defined by $V(\ell_j^q) = (s_{A_1}(\ell_j^q), s_{A_2}(\ell_j^q), ..., s_{A_m}(\ell_j^q))$. The value of $V(\ell_j^q)$ taken at the l^{th} place, $1 \leq l \leq m$, is denoted by $V_l(\ell_j^q) = V_{A_l}(\ell_j^q) = s_{A_l}(\ell_j^q)$.

3.2 On the Complexity of Unrestricted Szilard Languages

In this section we focus on Szilard languages of MGs with CF rules, with or without appearance checking. The case of Szilard languages of MGs with CS and PS rules is briefly discussed in Section 6. For Szilard languages associated with MGs without appearance checking and CF rules we have the next result.

Theorem 3 *Each language $L \in SZM(CF)$ can be recognized by an indexing ATM in $\mathcal{O}(\log n)$ time and space ($SZM(CF) \subseteq ALOGTIME$).*

Proof. Let $G = (N, T, A_1, M, F)$ be an arbitrary context-free MG, with $F = \emptyset$. Consider an indexing ATM $\mathcal{A}$ composed of an input tape that stores an input word, $\eta \in M^*$, of length n, $\eta = \eta_1\eta_2...\eta_n$, an index tape to read the input symbols, and a (read-write) working tape, divided into three tracks. The first track is of a fixed and finite dimension and, at the beginning of computation, it stores the Parikh vector of the axiom V^0, i.e., $V_1^0 = V_{A_1}^0 = 1$ and $V_l^0 = V_{A_l}^0 = 0$, $V(m_j)$, and the net effects $df_{A_l}(p_{m_j,r})$ of all rules in m_j, $1 \leq j \leq k$, $1 \leq r \leq k_{m_j}$, $1 \leq l \leq m$. The other two tracks are initially empty.

Level 1 (*Existential*) In an existential state $\mathcal{A}$ guesses the length of η and verifies the correctness of this guess, i.e., writes on the index tape n, and checks whether the n^{th} cell of the input tape contains a terminal symbol and the cell $n+1$ contains no symbol. The correct value of n is recorded in binary on the second track of the working tape. The first and second tracks are parted by a double bar symbol ($\|$). The end of the second track (and the beginning of the third track) is market by another symbol $\|$.

Level 2 (*Universal*) $\mathcal{A}$ spawns n universal processes $\wp_i$, $1 \leq i \leq n$.

• On the first process $\mathcal{A}$ reads $\eta_1 = (p_{\eta_1,1}, p_{\eta_1,2}, ..., p_{\eta_1,k_{\eta_1}})$, and it checks whether $\alpha_{\eta_1,1} = A_1$ and $sdf_{\alpha_{\eta_1,r+1}} = V^0_{\alpha_{\eta_1,r+1}} + \sum_{l=1}^{r} df_{\alpha_{\eta_1,r+1}}(p_{\eta_1,l}) \geq 1$, $1 \leq r \leq k_{\eta_1} - 1$, i.e., whether the nonterminal $\alpha_{\eta_1,r+1}$ rewritten by the $(r+1)^{th}$ rule of η_1, exists in the sentential form generated up to the r^{th} step of derivation in η_1. Process $\wp_1$ returns 1 if these conditions hold. Otherwise, $\wp_1$ returns 0.

• For each $\wp_i$, $2 \leq i \leq n-1$, $\mathcal{A}$ counts the number of occurrences of each matrix $m_j \in M$, $1 \leq j \leq k$, in $\eta^{(i)} = \eta_1\eta_2...\eta_{i-1}$. Let us consider that each m_j occurs in $\eta^{(i)}$ of $c_j^{(i)}$ times, $0 \leq c_j^{(i)} \leq i-1$. Then, for each $1 \leq l \leq m$, $\mathcal{A}$ computes $s_{A_l}^{(i)} = V_l^0 + \sum_{j=1}^{k} c_j^{(i)} V_l(m_j)$, i.e., the number of occurrences of each A_l in the sentential form upon which η_i is applied. Consider $\eta_i = (p_{\eta_i,1}, p_{\eta_i,2}, ..., p_{\eta_i,k_{\eta_i}})$ and $\alpha_{\eta_i,1} = A_{q_i}$, $1 \leq q_i \leq m$. $\mathcal{A}$ checks whether $s_{A_{q_i}}^{(i)} = s_{\alpha_{\eta_i,1}}^{(i)} \geq 1$, i.e., whether the matrix η_i can start the computation. For each $1 \leq r \leq k_{\eta_i} - 1$, $\mathcal{A}$ checks whether[8] $sdf_{\alpha_{\eta_i,r+1}} = s_{\alpha_{\eta_i,r+1}}^{(i)} + \sum_{l=1}^{r} df_{\alpha_{\eta_i,r+1}}(p_{\eta_i,l}) \geq 1$, i.e., whether the rules $p_{\eta_i,r}$, $2 \leq r \leq k_{\eta_i}$, can be applied in the same order they occur in η_i. Process $\wp_i$, $2 \leq i \leq n-1$, returns 1 if these conditions hold. Otherwise, $\wp_i$ returns 0.

• The last process $\wp_n$ counts the number $c_j^{(n)}$ of occurrences of each m_j, $1 \leq j \leq k$, in $\eta^{(n)} = \eta_1\eta_2...\eta_{n-1}$, and computes the sums $s_{A_l}^{(n)} = V_l^0 + \sum_{j=1}^{k} c_j^{(n)} V_l(m_j)$ and $s_{A_l}^{(n,out)} = V_l^0 + \sum_{j=1}^{k} c_j^{(n)} V_l(\eta_j) + V_l(\eta_n)$, $1 \leq l \leq m$. Consider $\eta_n = (p_{\eta_n,1}, p_{\eta_n,2}, ..., p_{\eta_n,k_{\eta_i}})$, and $\alpha_{\eta_n,1} = A_{q_n}$, $1 \leq q_n \leq m$. Process $\wp_n$ returns 1, if $s_{A_{q_n}}^{(n)} \geq 1$, $sdf_{\alpha_{\eta_n,r+1}} = s_{\alpha_{\eta_n,r+1}}^{(n)} + \sum_{l=1}^{r} df_{\alpha_{\eta_n,r+1}}(p_{\eta_n,l}) \geq 1$, for each $1 \leq r \leq k_{\eta_n} - 1$, and $s_{A_l}^{(n,out)} = 0$, for each $1 \leq l \leq m$. Otherwise, $\wp_n$ returns 0.

Each of the above processes uses the third track of the working tape for auxiliary computations, i.e., to record in binary the elements $c_j^{(i)}$, $2 \leq i \leq n$, $1 \leq j \leq k$, and

[8] Here $s_{\alpha_{\eta_i,r+1}}^{(i)}$ is actually the sum $s_{A_l}^{(i)}$ where $\alpha_{\eta_i,r+1}$ is the l^{th} nonterminal in N.

to compute the sums $s_{A_l}^{(i)}$, $2 \leq i \leq n$, $sdf_{\alpha_{\eta_i,r+1}}$, $1 \leq r \leq k_{\eta_i} - 1$, $1 \leq i \leq n$, and $s_{A_l}^{(n,out)}$, $1 \leq l \leq m$. The input η is accepted if all $\wp_i$, $1 \leq i \leq n$, return 1. If at least one of the above processes returns 0, then η is rejected.

The counting procedure used by each process $\wp_i$, $1 \leq i \leq n$, is a function in the U_{E^*}-uniform NC^1. The same observation holds for the summation of a constant number of vectors or multiplication of an integer of at most $\log n$ bits with a binary constant. Hence, all the above operations can be performed by an ATM in $\log n$ time and space. The out-degree of the computation tree at this level is n. By using a divide and conquer procedure the computation tree can be converted into a binary tree of height at most $\log n$. Consequently, the whole algorithm requires $\mathcal{O}(\log n)$ time and space. □

Corollary 1 $SZM(CF) \subset \mathcal{NC}^1$.

Proof. The claim is a direct consequence of Theorem 3 and results in [41]. The inclusion is strict since there exists $L = \{p^n | n \geq 0\} \in \mathcal{NC}^1 - SZM(CF)$. □

Corollary 2 $SZM(CF) \subset DSPACE(\log n)$.

Proof. $SZM(CF) \subset NC^1 \subseteq DSPACE(\log n)$. □

Let G be a context-free MG with appearance checking. When reading a symbol η_i, $1 \leq i \leq n$, as in the proof of Theorem 3, an indexing ATM $\mathcal{A}$ cannot deduce which of the rules in F have been previously applied or not. By using a simulation a of MGs by off-line Turing machines (henceforth TMs) we have the next result.

Theorem 4 *Each language $L \in SZM^{ac}(CF)$ can be recognized by an off-line deterministic Turing machine in $\mathcal{O}(\log n)$ space and $\mathcal{O}(n \log n)$ time ($SZM^{ac}(CF) \subset DSPACE(\log n)$).*

Proof. Let $G = (N, T, A_1, M, F)$ be a context-free MG with appearance checking. Denote by $P = \{p_{\ell_1}, p_{\ell_2}, ..., p_{\ell_s}\}$ the ordered set of productions in M, where ℓ_q is the unique label associated with the q^{th} production in P, and each p_{ℓ_q} is a rule of the form $\alpha_{\ell_q} \rightarrow \beta_{\ell_q}$, with $\alpha_{\ell_q} \in N$, $\beta_{\ell_q} \in (N \cup T)^*$, that may belong to one or more matrices. Let $\mathcal{B}$ be an off-line deterministic Turing machine (with stationary positions) composed of an input tape that stores an input word $\eta \in M^*$, $\eta = \eta_1\eta_2...\eta_n$, of length n, and a (read-write) working tape. For each rule p_{ℓ_q}, $\mathcal{B}$ records on the working tape, the ℓ_q symbol, the left-hand side of p_{ℓ_q} (i.e., the nonterminal α_{ℓ_q}), and the net effects of rule p_{ℓ_q}, with respect to each nonterminal $A_l \in N$. Besides, any rule in $P \cap F$ is marked on the working tape by a symbol $\sharp$. In this way each production in m_j has associated a unique label ℓ_x, and m_j can be seen as a sequence of productions of the form p_{ℓ_x}, whose characteristics, i.e., the rule's left-hand side, the net effects with respect to each nonterminal in N, and the $\sharp$ symbol, can be read from the working tape. In order to be readable all these characteristics are separated by a special symbol. Since each matrix is a finite sequence of rules, and the net effect of a rule, with respect to a certain nonterminal in N, does not depend

on the length of the input string, to record the rules' characteristics $\mathcal{B}$ requires a constant amount of time and space (regarding the length of input).

At the beginning of the computation the working tape is empty. From an initial state q_0, in stationary positions, i.e., by reading no symbol, $\mathcal{B}$ records (in constant time) the characteristics of all rules in P. At the right-hand side of these characteristics $\mathcal{B}$ books m blocks, each of which is composed of $\mathcal{O}(\log n)$ cells. These blocks are used to record, in binary, the Parikh vectors associated with sentential forms. We refer to these blocks as Parikh blocks. For the moment, $\mathcal{B}$ uses the first cell of each Parikh block to record the Parikh vector V^0 of the axiom, i.e., $V_1^0 = V_{A_1}^0 = 1$ and $V_l^0 = V_{A_l}^0 = 0$, $2 \leq l \leq m$. The net effects of the rules in P and the Parikh blocks are each separated by a $\perp$ symbol. Denote by q_c the state reached at the end of this procedure. Then $\mathcal{B}$ continues with the next procedures.

$\mathcal{B}_{\eta_1,j_1}$: $\mathcal{B}$ searches for the very first rule p_{η_1,j_1} in $\eta_1 = (p_{\eta_1,1}, p_{\eta_1,2}, ..., p_{\eta_1,k_{\eta_1}})$, such that p_{η_1,j_1} rewrites the axiom. This can be done by letting $\mathcal{B}$ when reading η_1, pass from state q_c to state $q_{\eta_1,1}$, and from state $q_{\eta_1,j}$ (in stationary positions) to state $q_{\eta_1,j+1}$, $1 \leq j \leq j_1 - 1$, and checking for each rule $p_{\eta_1,j}$, when entering in $q_{\eta_1,j}$, whether $A_1 \neq \alpha_{\eta_1,j}$ and $p_{\eta_1,j} \in F$ (so that $p_{\eta_1,j}$ can be skipped), $1 \leq j \leq j_1 - 1$, and whether $A_1 = \alpha_{\eta_1,j_1}$, when entering in state q_{η_1,j_1}. Suppose that rule p_{η_1,j_1} with the above properties, is labeled by ℓ_{x_1}, i.e., $p_{\eta_1,j_1} = p_{\ell_{x_1}}$. $\mathcal{B}$ adds the net effect of rule $p_{\ell_{x_1}}$, with respect to nonterminal A_l to the value provided by V_l^0, for any $1 \leq l \leq m$. Hence, $\mathcal{B}$ computes the sums $sdf_{A_l}^{(\eta_1,j_1)} = V_l^0 + df_{A_l}(p_{\ell_{x_1}})$. For each $1 \leq l \leq m$, the l^{th} Parikh block used to record V_l^0 is now used to record $sdf_{A_l}^{(\eta_1,j_1)}$ in binary.

$\mathcal{B}_{\eta_1,j_2}$: $\mathcal{B}$ searches for the very first rule p_{η_1,j_2} in η_1, that can be applied after rule $p_{\eta_1,j_1} = p_{\ell_{x_1}}$. This can be done by letting $\mathcal{B}$, in stationary positions, pass form state $q_{\eta_1,j}$ to state $q_{\eta_1,j+1}$, $j_1 \leq j \leq j_2 - 1$, and checking for each rule $p_{\eta_1,j}$, when entering in the state $q_{\eta_1,j}$, whether $sdf_{\alpha_{\eta_1,j}}^{(\eta_1,j_1)} = 0$ and $p_{\eta_1,j} \in F$ (i.e., $p_{\eta_1,j}$ can be skipped, $j_1 \leq j \leq j_2 - 1$), and whether $sdf_{\alpha_{\eta_1,j_2}}^{(\eta_1,j_1)} \geq 1$, when entering in state q_{η_1,j_2}. Suppose that rule p_{η_1,j_2}, with the above properties, is labeled by ℓ_{x_2}, i.e., $p_{\eta_1,j_2} = p_{\ell_{x_2}}$. $\mathcal{B}$ adds the net effect of rule $p_{\ell_{x_2}}$, with respect to nonterminal A_l, to $sdf_{A_l}^{(\eta_1,j_1)}$, for any $1 \leq l \leq m$. Hence, $\mathcal{B}$ computes the sums $sdf_{A_l}^{(\eta_1,j_2)} = sdf_{A_l}^{(\eta_1,j_1)} + df_{A_l}(p_{\eta_1,j_2}) = V_l^0 + df_{A_l}(p_{\ell_{x_1}}) + df_{A_l}(p_{\ell_{x_2}})$. The space used to record $sdf_{A_l}^{(\eta_1,j_1)}$ is now used to record $sdf_{A_l}^{(\eta_1,j_2)}$ in binary, for any $1 \leq l \leq m$.

Suppose that up to the r^{th} step of derivation in appearance checking in η_1, $\mathcal{B}$ has found a subsequence $(p_{\eta_1,j_1}, p_{\eta_1,j_2}, ..., p_{\eta_1,j_r})$ of η_1, composed of rules that can be effectively applied and that the sentential form obtained at the r^{th} step of derivation in η_1 contains $sdf_{A_l}^{(\eta_1,j_r)} = V_l^0 + \sum_{i=1}^{r} df_{A_l}(p_{\eta_1,j_i})$ nonterminals A_l, $1 \leq l \leq m$. The binary value of $sdf_{A_l}^{(\eta_1,j_r)}$, $1 \leq l \leq m$, can be found on the l^{th} Parikh block.

$\mathcal{B}_{\eta_1,j_{r+1}}$: $\mathcal{B}$ searches for the very first rule $p_{\eta_1,j_{r+1}}$ in η_1, that can be applied after rule p_{η_1,j_r}. This can be done by letting $\mathcal{B}$, in stationary positions, to pass form state $q_{\eta_1,j}$ to state $q_{\eta_1,j+1}$, $j_r \leq j \leq j_{r+1} - 1$, and checking for each rule $p_{\eta_1,j}$, when entering in state $q_{\eta_1,j}$, whether $sdf_{\alpha_{\eta_1,j}}^{(\eta_1,j_r)} = 0$ and $p_{\alpha_{\eta_1,j}} \in F$, $j_r \leq j \leq j_{r+1} - 1$, and whether $sdf_{\alpha_{\eta_1,j_{r+1}}}^{(\eta_1,j_r)} \geq 1$. Then $\mathcal{B}$ computes $sdf_{A_l}^{(\eta_1,j_{r+1})} = sdf_{A_l}^{(\eta_1,j_r)} + df_{A_l}(p_{\eta_1,j_{r+1}})$.

The space used to record $sdf_{A_l}^{(\eta_1,j_r)}$ is now used to record (in binary) $sdf_{A_l}^{(\eta_1,j_{r+1})}$, $1 \leq l \leq m$.

$\mathcal{B}$ continues in this way until a subsequence $\ell_{\eta_1}^q = (p_{\eta_1,j_1}, p_{\eta_1,j_2}, ..., p_{\eta_1,j_{\xi_{\eta_1}}^q})$ of η_1, is found such that the rules in $\ell_{\eta_1}^q$ can be effectively applied one after the other in the order they occur in $\ell_{\eta_1}^q$, and all rules in η_1 that do not occur in $\ell_{\eta_1}^q$ are skipped because they cannot be effectively applied according to Definition 7. If $j_{\xi_{\eta_1}}^q \neq k_{\eta_1}$, then $\mathcal{B}$ continues to check, by passing from state $q_{\eta_1,j}$ to $q_{\eta_1,j+1}$, $j_{\xi_{\eta_1}^q} \leq j \leq k_{\eta_1} - 1$, whether rule $p_{\eta_1,j}$ can be passed over because $\alpha_{\eta_1,j}$ does not occur in the sentential form obtained at the j^{th} step of derivation, in appearance checking, in η_1. If no such a subsequence of η_1 can be found, then η is rejected.

Suppose that all rules in η_1 can be applied in appearance checking, and at the end of the checking procedures described above, $\mathcal{B}$ reaches the state $q_{\eta_1,k_{\eta_1}}$. From state $q_{\eta_1,k_{\eta_1}}$, $\mathcal{B}$ enters in state $q_{\eta_2,1}$, by reading $\eta_2 = (p_{\eta_2,1}, p_{\eta_2,2}, ..., p_{\eta_2,k_{\eta_2}})$, and from state $q_{\eta_2,j}$, $\mathcal{B}$ passes to state $q_{\eta_2,j+1}$, $j_1 \leq j \leq k_{\eta_2} - 1$, by checking, in stationary positions, whether all rules in η_2 can be applied in appearance checking. This can be done by consecutively applying procedures of type $\mathcal{B}_{\eta_2,j_r}$, $1 \leq r \leq \xi_{\eta_2}^q$, where $\ell_{\eta_2}^q = (p_{\eta_2,j_1}, p_{\eta_2,j_2}, ..., p_{\eta_2,j_{\xi_{\eta_2}^q}})$ is a subsequence of η_2 composed of rules that can be effectively applied during the derivation inside η_2. The Parikh vector of the sentential form obtained after the application of matrix η_2, in appearance checking, is recorded on the m Parikh blocks. $\mathcal{B}$ proceeds in the same manner for each matrix η_i, $3 \leq i \leq n$.

The input is accepted if, for each η_i, a policy $\ell_{\eta_i}^q = (p_{\eta_i,j_1}, p_{\eta_i,j_2}, ..., p_{\eta_2,j_{\xi_{\eta_i}^q}})$ of η_i can be found, such that all rules in $\ell_{\eta_i}^q$ can be effectively applied, while rules in η_i that are not in $\ell_{\eta_i}^q$ are skipped according to Definition 7. Besides, for the last matrix η_i, in the last state $q_{\eta_n,k_{\eta_n}}$, $\mathcal{B}$ also checks whether the Parikh vector of the last sentential form contains no nonterminal, i.e., whether $sdf_{A_l} = V_l^0 + \sum_{i=1}^{n} \sum_{r=1}^{j_{\xi_{\eta_i}^q}} df_{A_l}(p_{\eta_i,j_r}) = 0$, for any $1 \leq l \leq m$.

As MGs work sequentially, the length of each sentential form is linearly bounded by the length of the input. Hence, $\mathcal{O}(\log n)$ cells are enough in order to record, in binary, the number of occurrences of each nonterminal A_l in a sentential form. Therefore, the space used by $\mathcal{B}$ is $\mathcal{O}(\log n)$. Each time reading an input symbol, $\mathcal{B}$ visits $\mathcal{O}(\log n)$ cells in the working tape, and the constant number of auxiliary operations with binary numbers (performed at each step of derivation) require $\mathcal{O}(\log n)$ time. Hence, $\mathcal{B}$ performs the whole computation in $\mathcal{O}(n \log n)$ time. □

Corollary 3 $SZM^{ac}(CF) \subset DSPACE(\log n)$.

Proof. The strict inclusion of $SZM^{ac}(CF)$ in $DSPACE(\log n)$ follows, e.g., from the existence of the language $L = \{p^n | n \geq 0\} \in DSPACE(\log n) - SZM^{ac}(CF)$. □

3.3 On the Complexity of Leftmost Szilard Languages

MGs are highly nondeterministic rewriting systems. First, due to the nondeterministic manner in which nonterminals can be rewritten, and second, due to the appearance checking restrictions on which rules in a matrix can be passed over. The second type of nondeterminism can be avoided by omitting the appearance checking

mode. The first type of nondeterminism can be reduced by imposing an order on the manner in which nonterminals are rewritten, similar to leftmost derivations in CGs. As in the case of CGs, the leftmost derivation order leads to more interesting results. In this section we focus on the complexity of Szilard languages associated with leftmost-i derivations introduced in [14], $i \in \{1, 2, 3\}$, (Definition 10). However, results provided for these three types of derivations can be generalized for several other leftmost derivations introduced in [13], [18], or [42]. Hence, proofs in this subsection can be considered as "prototypes" for a large variety of complexity results concerning several types of leftmost Szilard languages. For the case of leftmost-1 Szilard languages we have

Theorem 5 *Each language $L \in SZML_1(CF)$ can be recognized by an indexing ATM in $\mathcal{O}(\log n)$ time and space.*

Proof. Let $G = (N, T, A_1, M, F)$ be a MG with CF rules and without appearance checking, working in the leftmost-1 derivation manner. Consider an indexing ATM $\mathcal{A}$ having a similar configuration as the machine used in the proof of Theorem 3, and let $\eta \in M^*$, $\eta = \eta_1\eta_2...\eta_n$ be an input word of length n. In order to guess the length of η, $\mathcal{A}$ proceeds with the procedure described at Level 1 (Existential), Theorem 3. Then $\mathcal{A}$ spawns (Level 2) n universal processes $\wp_i$, $1 \leq i \leq n$.

• On the first process $\mathcal{A}$ reads η_1, where $\eta_1 = (p_{\eta_1,1}, p_{\eta_1,2}, ..., p_{\eta_1,k_{\eta_1}})$, and it checks whether $\alpha_{\eta_1,1} = A_1$ and $sdf_{\alpha_{\eta_1,r+1}} = V^0_{\alpha_{\eta_1,r+1}} + \sum_{l=1}^{r} df_{\alpha_{\eta_1,r+1}}(p_{\eta_1,l}) \geq 1$, $1 \leq r \leq k_{\eta_1} - 1$, i.e., whether the nonterminal $\alpha_{\eta_1,r+1}$ rewritten by the $(r+1)^{th}$ rule of η_1, exists in the sentential form generated up to the r^{th} step of derivation performed by η_1. Then $\mathcal{A}$ checks whether rules in η_1 can be applied in a leftmost-1 derivation manner. In order to check this property, from right-to-left in η_1, $\mathcal{A}$ checks whether each rule $p_{\eta_1,r}$, $2 \leq r \leq k_{\eta_1}$, can rewrite the first nonterminal occurring on the right-hand side of the previous rule $p_{\eta_1,r-1}$, if this is a non-terminal rule. If $p_{\eta_1,r-1}$ is a terminal rule, then $\mathcal{A}$ searches backward in η_1 for the non-terminal rule that produces the nonterminal rewritten by rule $p_{\eta_1,r}$. In this respect $\mathcal{A}$ existentially guesses (Level 3) an integer s (finite in this case) such that the rule $p_{\eta_1,s}$ is a non-terminal rule. $\mathcal{A}$ counts the number of rules existing in η_1 between rule $p_{\eta_1,s}$ and rule $p_{\eta_1,r}$ (excluding $p_{\eta_1,r}$). Suppose that this number is s_v, i.e., $s_v = r - s$. Then, $\mathcal{A}$ counts the number of nonterminals that each rule existing between $p_{\eta_1,s+1}$ and $p_{\eta_1,r-1}$ has on its right-hand side. Suppose that this number is s_q. For s_v and s_q, $\mathcal{A}$ checks whether the $(s_v - s_q)^{th}$ nonterminal existing on the right-hand side of rule $p_{\eta_1,s}$ equals the nonterminal rewritten by rule $p_{\eta_1,r}$, i.e., $\alpha_{\eta_1,r}$.

If $p_{\eta_1,s}$ is the right rule that produces in the sentential form the nonterminal rewritten by rule $p_{\eta_1,r}$, and this is the $\bar{r}^{th}$ nonterminal occurring on the right-hand side of rule $p_{\eta_1,s}$, then for the case of leftmost-1 derivation order, the following relation must hold $\bar{r} + s_q = s_v$. This is because each nonterminal produced in the sentential form by rules used in a leftmost-1 derivation manner, between $p_{\eta_1,s}$ and $p_{\eta_1,r}$ (including nonterminals existing up to the $\bar{r}^{th}$ nonterminal on the right hand side of $p_{\eta_1,s}$), must be fully rewritten by these rules. The nonterminals existing in the sentential form before $p_{\eta_1,s}$ is applied will be rewritten only after the new nonterminals produced between $p_{\eta_1,s}$ and $p_{\eta_1,r}$ are fully rewritten.

However, guessing an integer s that satisfies the above condition is not sufficient, since between $p_{\eta_1,1}$ and $p_{\eta_1,r}$, there may exist several rules $p_{\eta_1,s}$ with this property. This may happen for instance, when the right-hand side of $p_{\eta_1,s}$ has the length greater than $s_v - s_q$, and all nonterminals on the right-hand side of $p_{\eta_1,s}$, are equal with $\alpha_{\eta_1,r}$. In order to eliminate those rules $p_{\eta_1,s}$ that does not produce the real nonterminal rewritten by $p_{\eta_1,r}$, for each s found at Level 3, $\mathcal{A}$ universally branches (Level 4) all rules used between $p_{\eta_1,s}$ and $p_{\eta_1,r}$. On each branch that takes the rule $p_{\eta_1,l}$, $s < l < r$, $\mathcal{A}$ checks whether

1. $\alpha_{\eta_1,l}$ equals $\alpha_{\eta_1,r}$,
2. if $\alpha_{\eta_1,r}$ is the $\bar{r}^{th}$ nonterminal occurring on the right-hand side of rules $p_{\eta_1,s}$, $\bar{s}_q$ is the number of nonterminals produced between rules $p_{\eta_1,s+1}$ and $p_{\eta_1,l-1}$, and $\bar{s}_v = l - s$ is the number of rules used between rule $p_{\eta_1,s}$ and $p_{\eta_1,l}$ (excluding rule $p_{\eta_1,l}$), then the following condition holds $\bar{r} + \bar{s}_q = \bar{s}_v$,
3. the number of nonterminals $\alpha_{\eta_1,r}$ rewritten between rules $p_{\eta_1,s}$ and $p_{\eta_1,l-1}$ are equal with the number of nonterminals $\alpha_{\eta_1,r}$ produced between these rules, up to the $\bar{r}^{th}$ nonterminal occurring on the right-hand side of $p_{\eta_1,s}$ (excluding the $\bar{r}^{th}$ nonterminal).

On each universal branch $\mathcal{A}$ returns 0 if conditions $1-3$ hold, which means that the $\bar{r}^{th}$ nonterminal occurring on the right-hand side of rule $p_{\eta_1,s}$ is not the real nonterminal rewritten by $p_{\eta_1,r}$. Hence, the existential branch that guessed s, must be canceled. Otherwise, $\mathcal{A}$ returns 1. If all universal branches spawned for $p_{\eta_1,s}$ at Level 4, return 1, then the rule $p_{\eta_1,s}$ is the rule that produce the nonterminal rewritten by $p_{\eta_1,s}$ in leftmost-1 derivation manner. In this case $\wp_1$ returns 1.

• For each $\wp_i$, $2 \leq i \leq n$, $\mathcal{A}$ proceeds as follows. $\mathcal{A}$ counts the number of occurrences of each matrix m_j, $1 \leq j \leq k$, in $\eta^{(i)} = \eta_1\eta_2...\eta_{i-1}$. Suppose that this number is $c_j^{(i)}$, $0 \leq c_j^{(i)} \leq i-1$. Then, for each $1 \leq l \leq m$, $\mathcal{A}$ computes the values $s_l^{(i)} = V_l^0 + \sum_{j=1}^{k} c_j^{(i)} V_l(m_j)$, i.e., $\mathcal{A}$ computes the number $s_l^{(i)}$ of occurrences of nonterminal A_l in the sentential form upon which η_i is applied. Consider $\eta_i = (p_{\eta_i,1}, p_{\eta_i,2}, ..., p_{\eta_i,k_{\eta_i}})$ and $\alpha_{\eta_i,1} = A_{q_i}$, $1 \leq q_i \leq m$. Then $\mathcal{A}$ checks whether $s_{q_i}^{(i)} = s_{\alpha_{\eta_i,1}}^{(i)} \geq 1$, i.e., whether the matrix η_i can start the computation. For each $1 \leq r \leq k_{\eta_i} - 1$, $\mathcal{A}$ checks whether[9] $sdf_{\alpha_{\eta_i,r+1}} = s_{\alpha_{\eta_i,r+1}}^{(i)} + \sum_{l=1}^{r} df_{\alpha_{\eta_i,r+1}}(p_{\eta_i,l}) \geq 1$, i.e., whether the rules $p_{\eta_i,r}$, $2 \leq r \leq k_{\eta_i}$, can be applied in the same order they occur in η_i.

Then, $\mathcal{A}$ checks whether rules in η_i can be applied in a leftmost-1 derivation manner. In this respect, $\mathcal{A}$ checks, from right-to-left in the sequence $\eta_i = (p_{\eta_i,1}, p_{\eta_i,2}, ..., p_{\eta_i,k_{\eta_i}})$, whether each rule $p_{\eta_i,r}$, $2 \leq r \leq k_{\eta_i}$, rewrites the first nonterminal occurring on the right-hand side of the previous rule $p_{\eta_i,r-1}$, if this is not a terminal rule. If $p_{\eta_i,r-1}$ is a terminal rule, then $\mathcal{A}$ first searches backward in η_i, as in $\wp_1$, for an integer s such that rule $p_{\eta_i,s}$ produces in the sentential form the nonterminal rewritten by $p_{\eta_i,r}$. If no rule with this property can be found in η_i, $\mathcal{A}$ searches backward in $\eta^{(i)} = \eta_1\eta_2...\eta_{i-1}$ for a matrix η_v such that there exists a non-terminal rule in η_v that produces the nonterminal rewritten by $p_{\eta_i,r}$.

In this order, $\mathcal{A}$ spawns $i-1$ existential branches (Level 3), and each branch

[9]Note that $s_{\alpha_{\eta_i,r+1}}^{(i)}$ is actually the sum $s_l^{(i)}$ where $\alpha_{\eta_i,r+1}$ is the l^{th} nonterminal occurring in $V(m_j)$.

takes the matrix η_v, $1 \leq v \leq i-1$. Suppose that η_v is defined by the sequence $(p_{\eta_v,1}, p_{\eta_v,2}, ..., p_{\eta_v,k_{\eta_v}})$. $\mathcal{A}$ checks whether there exists a non-terminal rule $p_{\eta_v,s}$, $1 \leq s \leq k_{\eta_v}$, in η_v, such that $p_{\eta_v,s}$ produces the nonterminal rewritten by $p_{\eta_i,r}$. This is performed as follows.

Denote by s_v the number of rules used in the derivation process between rule $p_{\eta_v,s}$ of matrix η_v and rule $p_{\eta_i,r-1}$ of matrix η_i (including rules $p_{\eta_v,s}$ and $p_{\eta_v,r-1}$). Suppose that q of these rules (without counting the rule $p_{\eta_v,s}$) are non-terminal. Denote by s_q the total number of nonterminals produced by the q non-terminal rules used between $p_{\eta_i,s+1}$ and $p_{\eta_v,r-1}$. Then, as in process $\wp_1$, $\mathcal{A}$ checks whether $\alpha_{\eta_i,r}$ is the $(s_v - s_q)^{th}$ nonterminal occurring on the right-hand side of rule $p_{\eta_v,s}$. Note that s_v, q, and s_q can be computed by $\mathcal{A}$ through a trivial counting and summation procedure.

Each existential branch spawned at Level 3, is labeled by 1 if there exists a rule $p_{\eta_v,r}$ with the above properties. For each existential branch at Level 3, labeled by 1, $\mathcal{A}$ checks whether the $\bar{r}^{th}$ nonterminal occurring in $\beta_{\eta_v,s}$ is indeed the nonterminal $\alpha_{\eta_i,r}$ rewritten by rule $p_{\eta_i,r}$, i.e., no other rule used between rule $p_{\eta_v,s}$ of matrix $\ell^q_{\eta_v}$ and rule $p_{\eta_i,r}$ of matrix η_i rewrites the $\bar{r}^{th}$ nonterminal $\alpha_{\eta_i,r}$, occurring in $\beta_{\eta_v,s}$. In this respect $\mathcal{A}$ universally branches (Level 4) all symbols occurring between η_{v+1} and η_{i-1}. There are $v-i-1$ such branches. On each branch holding a matrix η_l, defined by $(p_{\eta_l,1}, p_{\eta_l,2}, ..., p_{\eta_l,k_{\eta_v}})$, $v < l < i$, $\mathcal{A}$ settles on a non-terminal rule $p_{\eta_l,\bar{s}}$, $1 \leq \bar{s} \leq k_{\eta_l}$, and it checks whether

1. $\alpha_{\eta_l,\bar{s}}$ equals $\alpha_{\eta_i,r}$,
2. if $\alpha_{\eta_i,r}$ is the $\bar{r}^{th}$ nonterminal occurring on the right-hand side of rule $p_{\eta_v,r}$, $\bar{s}_q$ is the number of nonterminals produced between rules $p_{\eta_v,s+1}$ and $p_{\eta_l,\bar{s}-1}$, and $\bar{s}_v$ is the number of rules used between $p_{\eta_v,s}$ and $p_{\eta_l,\bar{s}}$ (excluding rule $p_{\eta_l,\bar{s}}$), then $\bar{r} + \bar{s}_q = \bar{s}_v$,
3. the number of nonterminals $\alpha_{\eta_i,r}$ rewritten between rules $p_{\eta_v,s}$ and $p_{\eta_l,\bar{s}-1}$ is equal to the number of nonterminals $\alpha_{\eta_i,r}$ produced between these rules, up to the $\bar{r}^{th}$ nonterminal occurring on the right-hand side of rule $p_{\eta_v,s}$ (excluding the $\bar{r}^{th}$ nonterminal).

Besides, for $\wp_n$, as in Theorem 3, $\mathcal{A}$ checks whether at the end of the application of matrix η_n the sentential form contains no nonterminal, i.e., whether condition

4. $s_l^{(n,out)} = 0$, where $s_l^{(n,out)} = V_l^0 + \sum_{j=1}^{k} c_j^{(n)} V_l(\eta_j) + V_l(\eta_n)$, $1 \leq l \leq m$, holds.

On each universal branch (at Level 4) $\mathcal{A}$ returns 0 if conditions $1-3$ hold. Otherwise, it returns 1. If all universal branches spawned for the value s at Level 4 return 1, then rule $p_{\eta_v,s}$ (found at Level 3) is the rule that produces the nonterminal rewritten by $p_{\eta_i,r}$ in the leftmost-1 derivation manner. Then the existential branch spawned at Level 3, corresponding to this s value, will be labeled by 1.

Each process $\wp_i$, $2 \leq i \leq n-1$ returns 1 if there exists a non-terminal rule $p_{\eta_v,s}$ in η_v with the above properties. Otherwise, $\wp_i$ returns 0. If conditions $1-4$ hold, $\wp_n$, returns 1. Otherwise, it returns 0.

Note that, for each $\wp_i$, $1 \leq i \leq n$, $\mathcal{A}$ does not have to check whether matrices η_v and η_l can be applied in a leftmost-1 derivation manner. Nor even if they can be applied, according to the definition of a derivation step in a MG. If η_v and η_l do not satisfy these requirements, then the wrong logical value returned by $\wp_i$ is "corrected" by the 0 value returned by processes $\wp_v$ or $\wp_l$, since all these processes are universally considered.

As in Theorem 3, each of the above processes uses the third track of the working

tape for auxiliary computations, i.e., to record in binary the elements $c_j^{(i)}$, $2 \leq i \leq n$, $1 \leq j \leq k$, and to compute the sums $s_l^{(i)}$, $2 \leq i \leq n$, $sdf_{\alpha_{\eta_i,r+1}}$, $1 \leq r \leq k_{\eta_i} - 1$, $1 \leq i \leq n$, and $s_l^{(n,out)}$, $1 \leq l \leq m$. It is easy to estimate that $\mathcal{A}$ performs the whole computation in logarithmic time and space. □

Corollary 4 $SZML_1(CF) \subset \mathcal{NC}^1$.

Corollary 5 $SZML_1(CF) \subset DSPACE(\log n)$.

The algorithm described in the proof of Theorem 5 cannot be applied for the case of leftmost-1 Szilard languages with appearance checking. The explanation is that, in the proof of Theorem 5, for any matrix η_i, $2 \leq i \leq n$, $\mathcal{A}$ has to guess a policy of a matrix η_v that contains a non-terminal rule that produces the nonterminal rewritten by rule $p_{\eta_i,r}$ of η_i. However, even if process $\wp_v$ returns the true value, which means that at its turn $\wp_v$ can be applied in a leftmost-1 derivation manner on the substring $\eta_1\eta_2...\eta_{v-1}$, the process $\wp_i$ cannot "see" with which policy η_v works in a leftmost-1 derivation manner, since all branches (or processes) spawned at the same level of the computation tree of $\mathcal{A}$ are independent on each other. Hence, the policy of $\wp_v$ that provides the non-terminal rule that produces the nonterminal rewritten by $p_{\eta_i,r}$, may not work in leftmost-1 derivation manner upon $\eta_1\eta_2...\eta_{v-1}$. That is why, for the case of leftmost-1 derivations in matrix grammars with appearance checking another algorithm should be applied.

In the sequel, we focus on the letfmost-i, $i \in \{1, 2, 3\}$, derivation procedures and we describe an ATM that recognizes letfmost-i, $i \in \{1, 2, 3\}$, Szilard languages in logarithmnic space and square logarithmnic time.

In order to simulate letfmost derivations in matrix grammars and to check whether a given word $\eta \in M^*$, $\eta = \eta_1\eta_2...\eta_n$, belongs to the $SZML_i^{ac}(CF)$ class, $i \in \{1, 2, 3\}$, for each matrix η_i, $1 \leq i \leq n$, the ATM must have information concerning the order in which the first occurrence of each nonterminal $A_l \in N$, $1 \leq l \leq m$, occurs in the sentential form at any step of derivation. This can be obtained either by sequentially reproducing the derivation up to the i^{th} step on which η_i is applied, or by letting the ATM to guess the possible order in which the first occurrences of nonterminals in N occur in the sentential form on which η_i is applied. Then the ATM has to check whether the guessed order is correct, in the sense that η_i can be applied in letfmost-i, $i \in \{1, 2, 3\}$, derivation manner on the sentential form built upon this order and whether the computation leads to a terminal derivation. In order to describe the way in which the parallel procedure works we introduce the notion of *ranging vector*. A ranging vector associated with a matrix m_j, $1 \leq j \leq k$, or a policy of this matrix, provides the order in which first occurrences of nonterminals in N occur in the sentential form obtained after m_j has been applied at that step of derivation.

Definition 11 Let $G = (N, T, A_1, M, F)$ be a MG with appearance checking, where $M = \{m_1, m_2, ..., m_k\}$ is the ordered finite set of matrices, $N = \{A_1, A_2, ..., A_m\}$ the ordered finite set of nonterminals, and $SF_{\ell_j^q}$ the sentential form obtained after matrix m_j, $1 \leq j \leq k$, with policy ℓ_j^q, has been applied at a certain step of derivation in G.

The *ranging vector* associated with the sentential form $SF_{\ell_j^q}$ and policy ℓ_j^q, denoted[10] by $S(\ell_j^q)$, is a vector in $\mathbf{N}^m$ defined as

$$S_l(\ell_j^q) = \begin{cases} 0, & \text{if } A_l \in N \text{ does not occur in } SF_{\ell_j^q}, \text{ i.e., } |SF_{\ell_j^q}|_{A_l} = 0, \\ i, & \text{if the first occurrence of } A_l \text{ in } SF_{\ell_j^q} \text{ is the } i^{th} \text{ element in the} \\ & \text{order of first occurrences of nonterminals from } N \text{ in } SF_{\ell_j^q}. \end{cases}$$

Note that, if matrix $m_{j'}$ with policy $\ell_{j'}^q$, is applied in the Szilard word before matrix m_j with the policy ℓ_j^q, then the ranging vector $S(\ell_j^q)$ can be nondeterministically computed knowing the ranging vector $S(\ell_{j'}^q)$, for all leftmost-i, $i \in \{1, 2, 3\}$, derivation cases.

Example 1 Let $S(\ell_{j'}^q) = (3, 0, 2, 1, 0) \in \mathbf{N}^5$ be the ranging vector associated with the sentential form obtained after the policy $\ell_{j'}^q$ of matrix $m_{j'}$ has been applied at a certain step of derivation, i.e., $SF_{\ell_{j'}^q} = A_4X_4A_3X_{3,4}A_1\bar{X}_{3,4}$, $X_4 \in (\{A_4\} \cup T)^*$, $X_{3,4}, \bar{X}_{3,4} \in (\{A_3, A_4\} \cup T)^*$ (such that $|SF_{\ell_{j'}^q})|_{A_1} = 1$). If ℓ_j^q is identified by the sequence $m_j^q = (A_4 \to tA_5, A_3 \to A_2)$, $t \in T^*$, then if m_j^q rewrites the first occurrence of A_4 in $SF_{\ell_{j'}^q}$ and the second occurrence of A_3, then the sentential form obtained after ℓ_j^q has been applied, in the leftmost-2 derivation manner, may look like

- $SF_{\ell_j^q} = tA_5A_4X_4A_3A_2X_{3,4}A_1\bar{X}_{3,4}$, $X_4 \in (\{A_4\} \cup T)^*$, $X_{3,4}, \bar{X}_{3,4} \in (\{A_3, A_4\} \cup T)^*$, i.e., $S(\ell_j^q) = (5, 4, 3, 2, 1)$,
- $SF_{\ell_j^q} = tA_5A_4X_4A_3\bar{X}_4A_1A_2X_{3,4}$, $X_4, \bar{X}_4 \in (\{A_4\} \cup T)^*$, $X_{3,4} \in (\{A_3, A_4\} \cup T)^*$, i.e., $S(\ell_j^q) = (4, 5, 3, 2, 1)$,
- $SF_{\ell_j^q} = tA_5A_3A_4X_4A_2X_{3,4}A_1\bar{X}_{3,4}$, $X_4 \in (\{A_4\} \cup T)^*$, $X_{3,4}, \bar{X}_{3,4} \in (\{A_3, A_4\} \cup T)^*$, i.e., $S(\ell_j^q) = (5, 4, 2, 3, 1)$, or like
- $SF_{\ell_j^q} = tA_5A_3A_2X_3A_1A_4X_{3,4}$, $X_3 \in (\{A_3\} \cup T)^*$, $X_{3,4} \in (\{A_3, A_4\} \cup T)^*$, i.e., $S(\ell_j^q) = (4, 3, 2, 5, 1)$.

Thus the sentential form $SF_{\ell_j^q}$ depends on the second occurrence of A_3 and A_4 in $SF_{\ell_{j'}^q}$. Note that, for the case of leftmost-2 derivations, if say, the first rule in ℓ_j^q rewrites A_3, then ℓ_j^q is eligible to be applied in leftmost-2 derivation manner if and only if there is no other policy of m_j and no other matrix, distinct of m_j, for which the first rule in the matrix sequence rewrites A_4. The same observations hold when erasing rules are applied.

For instance, if ℓ_j^q is identified by the sequence $m_j^q = (A_4 \to \lambda, A_3 \to t)$, $t \in T^*$, then if m_j^q rewrites the first occurrence of A_4 in $SF_{\ell_{j'}^q}$ and the second occurrence of A_3 in $SF_{\ell_{j'}^q}$, then the sentential form obtained after ℓ_j^q has been applied, in the leftmost-2 derivation manner, may look like

[10] If ℓ_j^q is not yet "decided" or $F = \emptyset$, then instead of $S(\ell_j^q)$ the notation $S(m_j)$ is used.

- $SF_{\ell_j^q} = A_4X_4A_3tX_{3,4}A_1\bar{X}_{3,4}$, $X_4 \in (\{A_4\} \cup T)^*$, $X_{3,4}, \bar{X}_{3,4} \in (\{A_3, A_4\} \cup T)^*$, i.e., $S(\ell_j^q) = (3, 0, 2, 1, 0)$,
- $SF_{\ell_j^q} = A_4X_4A_3\bar{X}_4A_1tX_{3,4}$, $X_4, \bar{X}_4 \in (\{A_4\} \cup T)^*$, $X_{3,4} \in (\{A_3, A_4\} \cup T)^*$, i.e., $S(\ell_j^q) = (3, 0, 2, 1, 0)$,
- $SF_{\ell_j^q} = A_3A_4X_4tX_{3,4}A_1\bar{X}_{3,4}$, $X_4 \in (\{A_4\} \cup T)^*$, $X_{3,4}, \bar{X}_{3,4} \in (\{A_3, A_4\} \cup T)^*$, i.e., $S(\ell_j^q) = (3, 0, 1, 2, 0)$, or like
- $SF_{\ell_j^q} = A_3tX_3A_1A_4X_{3,4}$, $X_3 \in (\{A_3\} \cup T)^*$, $X_{3,4} \in (\{A_3, A_4\} \cup T)^*$, i.e., $S(\ell_j^q) = (2, 0, 1, 3, 0)$,

depending on the second occurrence of A_3 and A_4 in $SF_{\ell_{j'}^q}$.

If the matrix m_j with the policy ℓ_j^q, defined by the sequence $m_j^q = (A_4 \to tA_5, A_3 \to A_2)$, is applied in the leftmost-3 derivation manner on SF', then after rewriting the nonterminal A_4, ℓ_j^q must rewrite only the first occurrence of A_3 in SF'. Hence, there are fewer possibilities than in the case of leftmost-2 derivation manner, to build the ranging vector associated with $SF_{\ell_j^q}$.

In this case, by applying ℓ_j^q defined by $m_j^q = (A_4 \to tA_5, A_3 \to A_2)$, on $SF_{\ell_{j'}^q} = A_4X_4A_3X_{3,4}A_1\bar{X}_{3,4}$, $X_4 \in (\{A_4\} \cup T)^*$, $X_{3,4}, \bar{X}_{3,4} \in (\{A_3, A_4\} \cup T)^*$, we may obtain

- $SF_{\ell_j^q} = tA_5A_4X_4A_2A_3X_{3,4}A_1\bar{X}_{3,4}$, $X_4 \in (\{A_4\} \cup T)^*$, $X_{3,4}, \bar{X}_{3,4} \in (\{A_3, A_4\} \cup T)^*$, i.e., $S(\ell_j^q) = (5, 3, 4, 2, 1)$,
- $SF_{\ell_j^q} = tA_5A_2A_3X_3A_4X_{3,4}A_1\bar{X}_{3,4}$, $X_3 \in (\{A_3\} \cup T)^*$, $X_{3,4}, \bar{X}_{3,4} \in (\{A_3, A_4\} \cup T)^*$, i.e., $S(\ell_j^q) = (5, 2, 3, 4, 1)$, or
- $SF_{\ell_j^q} = tA_5A_2A_4X_4A_3X_{3,4}A_1\bar{X}_{3,4}$, i.e., $S(\ell_j^q) = (5, 2, 4, 3, 1)$.

For the leftmost-1 derivation case matrix m_j with the policy ℓ_j^q, defined by the sequence $m_j^q = (A_4 \to tA_5, A_3 \to A_2)$, cannot be applied, since there is no possibility to obtain a new sentential form such that A_3, rewritten by the second rule, to be the leftmost nonterminal occurring in it. Matrix m_j with the policy ℓ_j^q, defined by the sequence $(A_4 \to \lambda, A_3 \to t)$, $t \in T^*$, can be applied on $SF_{\ell_{j'}^q} = A_4X_4A_3X_{3,4}A_1\bar{X}_{3,4}$, $X_4 \in (\{A_4\} \cup T)^*$, $X_{3,4}, \bar{X}_{3,4} \in (\{A_3, A_4\} \cup T)^*$, if and only if $X_4 = \lambda$.

In the sequel we briefly describe an ATM $\mathcal{A}$ that checks whether an input word $\eta \in M^*$, $\eta = \eta_1\eta_2...\eta_n$, belongs to $SZML_i^{ac}(CF)$, $i \in \{1, 2, 3\}$. First $\mathcal{A}$ guesses an n-tuple $\Re = (S(\eta_1), S(\eta_2), ..., S(\eta_n))$, where each $S(\eta_v)$ is the ranging vector associated with the matrix η_v, $1 \leq v \leq n$. There may exist $\mathcal{O}(c^n)$ such n-tuples of ranging vectors, where c is a constant that depends on the number of vectors in $\mathbf{N}^m$ that can be built upon the set $\{0, 1, ..., m\}$. For instance, if we have the information that a certain sentential form has only $m - s$ distinct nonterminals, then there are $(m - s + 1)^m$ guesses that may provide the ranging vector associated with this sentential form. Hence, $c = \mathcal{O}(\sum_{s=1}^{m-1}(m - s + 1)^m)$. According to this

observation, $\mathcal{A}$ spawns $\mathcal{O}(c^n)$ existential branches, each of them holding an n-tuple of type $\Re$. A branch will be labeled by 1 if each vector in $\Re$, i.e., $\Re_v = S(\eta_v)$, $1 \leq v \leq n-1$, provides[11] a possible order of first occurrences of nonterminals in N in the sentential form on which η_v ends the v^{th} step of derivation, the matrix η_{v+1} can be applied upon $S(\eta_v)$ in the leftmost-i, $i \in \{1,2,3\}$, derivation manner, and whether the derivation performed in the leftmost-i manner by using all ranging vectors in $\Re$, leads to a word in the language.

On each existential branch, $\mathcal{A}$ proceeds with an universal and existential level as follows. $\mathcal{A}$ spawns n universal processes $\wp_i$, $1 \leq i \leq n$. On each process $\mathcal{A}$ spawns a polynomial number of existential branches, each of them holding a possible configuration of policies used by matrices occurring in the input word up to the matrix η_i, and computes the net effect according to this configuration. $\mathcal{A}$ guesses a policy $\ell^q_{\eta_i}$ and, based on this net effect, checks whether matrix η_i with the policy $\ell^q_{\eta_i}$ can be applied, in the leftmost-i, $i \in \{1,2,3\}$, derivation manner, on the current sentential form for which the order of first occurrences of nonterminals in N is provided by the vector $S(\eta_{i-1})$ in $\Re$. Then $\mathcal{A}$ checks whether $S(\eta_i)$ is a ranging vector on which $\ell^q_{\eta_i}$ may complete the i^{th} step of derivation, in leftmost-i, $i \in \{1,2,3\}$, derivation manner.

Recall that the policy $\ell^q_{\eta_i}$ can be applied, in leftmost-1 derivation manner on the ranging vector $S(\eta_{i-1})$ if the first rule in $\ell^q_{\eta_i}$ rewrites the nonterminal A_l for which $S_l(\eta_{i-1}) = 1$, and each rule in $\ell^q_{\eta_i}$ rewrites the leftmost nonterminal occurring in the sentential form built according to the information provided by $S(\eta_{i-1})$ after applying the first rule in $\ell^q_{\eta_i}$.

The policy $\ell^q_{\eta_i}$ can be applied, in leftmost-2 derivation manner on the ranging vector $S(\eta_{i-1})$ if there exists an index l, $1 \leq l \leq m$, such that the first rule of $\ell^q_{\eta_i}$ rewrites A_l and there is no matrix m_j, $m_j \neq \eta_i$, and no policy $\ell^q_{m_j}$ of m_j, $1 \leq j \leq k$, such that the first rule in $\ell^q_{m_j}$ rewrites a nonterminal $A_{l'}$ with $S_{l'}(\eta_{i-1}) < S_l(\eta_{i-1})$.

For the case of leftmost-3 derivation manner $\mathcal{A}$ does not have to check the above leftmost-2 condition, since the first rule of the policy $\ell^q_{\eta_i}$ is allowed to rewrite the first occurrence of its left-hand side, i.e., A_l, even if there exist several other matrices for which the left-hand side of the first rule, say $A_{l'}$, may be placed in the sentential form before A_l, i.e., $S_{l'}(\eta_{i-1}) < S_l(\eta_{i-1})$. Hence, for the leftmost-3 derivation case $\ell^q_{\eta_i}$ can be applied if the first rule in $\ell^q_{\eta_i}$ rewrites any nonterminal A_l for which $S_l(\eta_{i-1}) \neq 0$.

Then $\mathcal{A}$ checks whether $S(\eta_i)$ is a ranging vector on which $\ell^q_{\eta_i}$ may complete the i^{th} step of derivation, in leftmost-i, $i \in \{1,2,3\}$, derivation manner.

Note that the ranging vector $S(\eta_{i-1})$ does not provide complete information concerning the shape of the sentential form obtained after the application of matrix η_{i-1}, since $S(\eta_{i-1})$ provides only the order of the first occurrences of each nonterminal in N. Hence, the position of the second, third, and so on, occurrence of a nonterminal must be considered according to the order provided by $S(\eta_{i-1})$.

To verify whether $S(\eta_i)$ is a possible ranging vector on which $\ell^q_{\eta_i}$ may complete the i^{th} step of derivation, $\mathcal{A}$ builds all possible ranging vectors that can be computed starting from $S(\eta_{i-1})$ in the leftmost-i, $i \in \{1,2,3\}$, derivation manner. Then $\mathcal{A}$ checks whether $S(\eta_i)$ is one of the ranging vectors computed in this way.

[11] $S(\eta_n)$ must be the null vector.

Each process $\wp_i$ returns 1 if there exists at least one configuration of policies used by matrices occurring in the input word up to the matrix η_i, and at least one policy $\ell^q_{\eta_i}$ of η_i, that satisfies the above leftmost-i, $i \in \{1,2,3\}$, requirements.

If all processes η_i, $1 \leq i \leq n$, return 1 then $\Re$ is a correct guess and the existential branch holding this tuple is labeled by 1. The input is accepted if there exists at least one existential branch, holding an n tuple $\Re$, labeled by 1. Otherwise, the input is rejected.

Note that guesses yielded by different branches at a certain level of the computation tree of an ATM are independent on each other. If the ranging vectors composing $\Re$ are separately guessed by each process $\wp_i$, $1 \leq i \leq n-1$, then $\mathcal{A}$ cannot check whether the policy $\ell^q_{\eta_i}$ that works in a leftmost-i, $i \in \{1,2,3\}$, derivation manner on the ranging vector $S(\eta_{i-1})$ yields the same ranging vector for which the policy $\ell^q_{\eta_{i+1}}$ is guessed by process $\wp_{i+1}$ to work in a leftmost-i, $i \in \{1,2,3\}$, derivation manner on the ranging vector $S(\eta_i)$. Therefore, $\mathcal{A}$ has to guess from the very beginning an n-tuple $\Re$ of ranging vectors associated with each matrix in η and to universally check the correctness of this guess through the processes $\wp_i$. In other words, the whole n-tuple $\Re$ must be seen by all the universal processes $\wp_i$, $1 \leq i \leq n$.

It is easy to observe that the first level of the computation tree associated with $\mathcal{A}$ can be "unfolded", by using a divide and conquer procedure, into a computation tree of height $\mathcal{O}(\log c^n) = \mathcal{O}(n)$ in which each node has the out-degree 2. To record the $\Re$ vector $\mathcal{A}$ needs $\mathcal{O}(n)$ space. Hence, this algorithm cannot be related to the parallel complexity classes $\mathcal{NC}^1$ and $\mathcal{NC}^2$. In order to improve the linear time and space resources to logarithmic (the logarithmic uniformity assumptions required by the $\mathcal{NC}$ classes) we divide the input string of length n, into $(\log n)^{\log n}$ substrings of length $\log n$, and apply the above algorithm for each substring. Briefly, the new algorithm works as follows.

The ATM $\mathcal{A}$ performs a number of $\log n$ "iterated" divisions, where n is the length of the input word. Dividing n by $[\log n]$ we obtain[12] a quotient Q_1 and a remainder R_1, i.e., $n = Q_1 [\log n] + R_1$, where $0 \leq R_1 < \log n$. Dividing the quotient Q_1 by $[\log n]$ we obtain a new quotient Q_2 and a remainder R_2, i.e., $n = (Q_2 [\log n] + R_2) [\log n] + R_1$, with $0 \leq R_2 < \log n$. We continue this procedure until the resulted quotient can be no longer divided by $[\log n]$. Suppose that Q_ℓ is this quotient, then $n = ((...((Q_\ell [\log n] + R_\ell) [\log n] + R_{\ell-1}) [\log n] + ...) [\log n] + R_2) [\log n] + R_1$, with $1 \leq Q_l < [\log n]$ and $0 \leq R_l < [\log n]$, $l \in \{1,2,...,\ell\}$. It is easy to prove that $\ell < \log n$.

$\mathcal{A}$ guesses an R_1-tuple of ranging vectors associated with the first R_1 matrices occurring in $\eta = \eta_1\eta_2...\eta_n$ and checks, similar as in the algorithm described above, whether the substring $\eta_1\eta_2...\eta_{R_1}$ is valid, according to the leftmost-i, $i \in \{1,2,3\}$, derivation procedure. Then $\mathcal{A}$ guesses a $[\log n]$-tuple of ranging vectors associated with matrices placed at the $[\log n]$ cutting points in η obtained by dividing the interval $[R_1 + 1...n]$ into $[\log n]$ intervals of length Q_1. $\mathcal{A}$ continues with this routine for each interval of length Q_1 as follows.

$\mathcal{A}$ checks, in parallel, whether the first R_2 matrices in each Q_1-interval forms a valid substring of a leftmost-i, $i \in \{1,2,3\}$, Szilard word. Then, in parallel for

[12] By $[a]$ we denote the largest integer not greater than a, where a is a real number.

each Q_1-interval, $\mathcal{A}$ guesses another $[\log n]$-tuple of ranging vectors associated with matrices placed at the $[\log n]$ cutting points in η obtained by dividing each interval of length $Q_1 - R_2$ into $[\log n]$ intervals of length Q_2. This procedure is repeated until intervals of length $Q_\ell < \log n$ are obtained. At this point, $\mathcal{A}$ checks whether the substring of η corresponding to the Q_ℓ-intervals, are valid according to the leftmost-i, $i \in \{1,2,3\}$, derivation order. It can be proved that all cutting points are right edges of these intervals. If correct ranging vectors can be found for all intervals and all cutting points, then η is a correct leftmost-i, $i \in \{1,2,3\}$, Szilard word.

On the other hand, the division operation is a function in the $\mathcal{NC}^1$ class[13] [5]. Since $\mathcal{A}$ performs a number of $\log n$ divisions, the computation tree associated with $\mathcal{A}$ has at least $\log n$ levels. At each level $\mathcal{A}$ needs $\mathcal{O}(\log c^{\log n}) = \mathcal{O}(\log n)$ time to check the correctness of a substring of length at most $\log n$, $\mathcal{O}(\log n)$ time to perform the division operation, and $\mathcal{O}(\log n)$ space (which is reused at each level) to record the ranging vectors. Hence, the above algorithm requires $\log^2 n$ time and $\log n$ space. More precisely, we have

Theorem 6 *Each language $L \in SZML_i^{ac}(CF)$, $i \in \{1,2,3\}$, can be recognized by an indexing ATM in $\mathcal{O}(\log^2 n)$ time and $\mathcal{O}(\log n)$ space.*

Proof. We prove the claim for the leftmost-2 derivation. For the leftmost-1 and leftmost-3 cases the proof is almost the same. Let $G = (N, T, A_1, M, F)$ be a MG with appearance checking, and $\mathcal{A}$ an indexing ATM with a similar configuration as in the proof of Theorem 3. Let $\eta \in M^*$, $\eta = \eta_1\eta_2...\eta_n$, be an input word of length n. To guess the length of η, $\mathcal{A}$ proceeds with the Level 1 (*Existential*), Theorem 3.

Level 2 (*Existential*) Consider the quotient Q_1 and the remainder R_1 of the division of n by $[\log n]$, where $0 \leq R_1 < [\log n]$. $\mathcal{A}$ spawns $\mathcal{O}(c^{\log n})$ existential branches, each branch holding an R_1-tuple of ranging vectors $\Re_{R_1} = (S(\eta_1), S(\eta_2), ..., S(\eta_{R_1}))$, where[14] $c = \mathcal{O}(\sum_{s=1}^{m-1}(m-s+1)^m)$ and $S(\eta_v)$ is the ranging vector associated with matrix η_v, $1 \leq v \leq R_1$. Then $\mathcal{A}$ checks whether all vectors in $\Re_{R_1}$ are correct, according to the leftmost-2 derivation order. This can be done in $\mathcal{O}(\log n)$ space and $\mathcal{O}(\log n)$ parallel time through Levels 3-4.

Levels 3-4 (*Universal-Existential*) $\mathcal{A}$ spawns (Level 3) R_1 universal processes $\wp_v^{(R_1)}$, $1 \leq v \leq R_1$.

• On $\wp_1^{(R_1)}$ $\mathcal{A}$ checks whether there exists a policy for η_1 that can be applied in leftmost-2 derivation manner on the axiom A_1 and ends this step of derivation with the ranging vector $S(\eta_1)$. Process $\wp_1^{(R_1)}$ returns 1 if these conditions hold.

• On each $\wp_v^{(R_1)}$, $2 \leq v \leq R_1$, $\mathcal{A}$ counts the number of occurrences of each matrix $m_j \in M$, $1 \leq j \leq k$, in $\eta^{(v)} = \eta_1\eta_2...\eta_{v-1}$. Suppose that each m_j occurs $c_j^{(v)}$ times, $0 \leq c_j^{(v)} \leq v-1$, in $\eta^{(v)}$. $\mathcal{A}$ guesses k tuples of integers[15] $t_j^{(v)} =$

[13] It is actually a function in the logspace-uniform $\mathcal{TC}^0$ class [22].

[14] The constant c depends on the number of vectors in $\mathbf{N}^m$ that can be built upon the set $\{0, 1, ..., m\}$. Here and throughout the paper, $c = \mathcal{O}(\sum_{s=1}^{m-1}(m-s+1)^m)$. At page 20 we have explained the manner in which this constant can be computed.

[15] Here and throughout this proof, by using an abuse of notation, we denote by $c_j = |m_j \cap F|$,

$(c_{j,1}^{(v)}, c_{j,2}^{(v)}, ..., c_{j,2^{c_j}-1}^{(v)}, c_{j,2^{c_j}}^{(v)})$, where $c_{j,q}^{(v)}$ with $0 \leq c_{j,q}^{(v)} \leq c_j^{(v)}$ and $\sum_{q=1}^{2^{c_j}} c_{j,q}^{(v)} = c_j^{(v)}$, represents the number of times the policy ℓ_j^q of matrix m_j, $1 \leq q \leq 2^{c_j}$, can be used when m_j is activated on $\eta^{(v)}$. Then $\mathcal{A}$ spawns (Level 4) $\mathcal{N}^{(R_1)}=\mathcal{O}(R_1^{\sum_{j=1}^{k} 2^{c_j}})$ existential branches, each of which holds k tuples $t_j^{(v)}$ (one tuple for each matrix). On each branch, $\mathcal{A}$ computes $s_l^{(v)} = V_l^0 + \sum_{j=1}^{k} \sum_{q=1}^{2^{c_j}} c_{j,q}^{(v)} V_l(\ell_j^q)$, $1 \leq l \leq m$. Suppose that η_v is a matrix with $2^{c_{\eta_v}}$ policies, where $c_{\eta_v} = |\eta_v \cap F|$, and that each policy $\ell_{\eta_v}^q$, $1 \leq q \leq 2^{c_{\eta_v}}$, is identified by the sequence $m_{\eta_v}^q$=($p_{\eta_v,1}^q$, $p_{\eta_v,2}^q$, ..., $p_{\eta_v,\xi_{\eta_v}^q}^q$), $1 \leq r \leq \xi_{\eta_v}^q$, $|\eta_v - F| \leq \xi_{\eta_v}^q \leq |\eta_v|$. Then $\mathcal{A}$ computes $sdf_{\alpha_{\eta_v,r+1}^q} = s_{\alpha_{\eta_v,r+1}^q}^{(v)} + \sum_{l=1}^{r} df_{\alpha_{\eta_v,r+1}^q}(p_{\eta_v,l}^q)$, $1 \leq r \leq \xi_{\eta_v}^q - 1$, and it checks whether

1. $s_{\alpha_{\eta_v,1}^q}^{(v)} \geq 1$, i.e., $p_{\eta_v,1}^q$ can be applied on $\eta^{(v)} = \eta_1\eta_2...\eta_{v-1}$,
2. $sdf_{\alpha_{\eta_v,r+1}^q} \geq 1$, $1 \leq r \leq \xi_{\eta_v}^q - 1$, i.e., rules of policy $\ell_{\eta_v}^q$ can be applied one by one in the order defined by the sequence $m_{\eta_v}^q$,
3. $S(\eta_{v-1})$ is a possible ranging vector with which η_{v-1} ends the $(v-1)^{th}$ step of derivation, i.e., $S_l(\eta_{v-1}) = 0$, if $s_l^{(v)} = 0$, and $S_l(\eta_{v-1}) > 0$, if $s_l^{(v)} > 0$, $1 \leq l \leq m$. Then $\mathcal{A}$ checks whether policy $\ell_{\eta_v}^q$ of η_v, can be applied on $S(\eta_{v-1})$ in the leftmost-2 derivation manner, i.e., there exists an index l, $1 \leq l \leq m$, such that $p_{\eta_v,1}^q$, the first rule in $m_{\eta_v}^q$, rewrites A_l, i.e., $S_l(\eta_{v-1}) \neq 0$, and there is no matrix m_j, $m_j \neq \eta_v$, and no policy $\ell_{m_j}^q$ of m_j, such that the first rule in $\ell_{m_j}^q$ rewrites a nonterminal $A_{l'}$ with $S_{l'}(\eta_{v-1}) < S_l(\eta_{v-1})$. Then $\mathcal{A}$ verifies whether $S(\eta_v)$ is a possible ranging vector on which $\ell_{\eta_v}^q$ ends the v^{th} step of derivation in leftmost-2 manner. Note that $S(\eta_v)$ can be (nondeterministically) computed knowing the rules of the policy $\ell_{\eta_v}^q$ applied in leftmost-2 derivation manner on $S(\eta_{v-1})$ (Example 1).

Each $\wp_v^{(R_1)}$, $2 \leq v \leq R_1$, returns 1 if there exist at least one $t_j^{(v)}$-tuple and at least one policy $\ell_{\eta_v}^q$ of η_v, that satisfy the above leftmost-2 requirements. If each $\wp_v^{(R_1)}$, $1 \leq v \leq R_1$, returns 1 then $\Re_{R_1}$ is a correct guess and the existential branch holding the $[\log n]$-tuple, spawned at Level 2, is labeled by 1.

Level 5 (*Existential*) Let Q_2 be the quotient and R_2 the remainder of Q_1 divided by $[\log n]$, $0 \leq R_2 < [\log n]$. $\mathcal{A}$ spawns $\mathcal{O}(c^{\log n})$ existential branches, each of them holding a $2\,[\log n]$-tuple of ranging vectors $\Re_{R_2}^c = (S(\eta_{R_1}), S(\eta_{R_1+R_2}), S(\eta_{R_1+Q_1}),$ $S(\eta_{R_1+Q_1+R_2}), ..., S(\eta_{R_1+([\log n]-1)Q_1}), S(\eta_{R_1+([\log n]-1)Q_1+R_2}))$, where $S(\eta_{R_1})$ is the ranging vector belonging to the $\Re_{R_1}$-tuple found correct at Levels 3-4, and each $S(\eta_j)$ is a guessed ranging vector associated with matrix η_j, $j \in \{R_1+R_2, R_1+Q_1, R_1+Q_1+R_2, R_1+2Q_1, ..., R_1+([\log n]-1)Q_1, R_1+([\log n]-1)Q_1+R_2, R_1+[\log n]\,Q_1\}$. Because $\Re_{R_1}$ is not useful anymore, the space used by $\mathcal{A}$ to record $\Re_{R_1}$ is allocated now to record $\Re_{R_2}^c$.

---

i.e., the number of rules existing in the same time in m_j and F. Then each matrix m_j, $1 \leq j \leq k$, may have $\binom{c_j}{0} + \binom{c_j}{1} + ... + \binom{c_j}{c_j} = 2^{c_j}$ policies. Namely, there exist $\binom{c_j}{0}$ choices of using no rule from $m_j \cap F$, $\binom{c_j}{1}$ choices of passing over only one rule from $m_j \cap F$, $\binom{c_j}{2}$ choices of passing over two rules from $m_j \cap F$, and so on.

Level 6 (*Universal*) On each existential branch from Level 5, $\mathcal{A}$ spawns $[\log n]$ universal processes $\wp_{i_1}^{(Q_1)}$, $0 \leq i_1 \leq [\log n]-1$. Each process $\wp_{i_1}^{(Q_1)}$ takes the interval $[R_1+i_1Q_1...R_1+i_1Q_1+R_2]$, and checks whether the ranging vectors $S(\eta_{R_1+i_1Q_1})$ and $S(\eta_{R_1+i_1Q_1+R_2})$, $1 \leq i_1 \leq [\log n]-1$, provide a correct order in which the leftmost-2 derivation can be performed between matrices $\eta_{R_1+i_1Q_1}$ and $\eta_{R_1+i_1Q_1+R_2}$. Besides $S(\eta_{R_1+i_1Q_1})$ and $S(\eta_{R_1+i_1Q_1+R_2})$, each $\wp_{i_1}^{(Q_1)}$ also keeps, from the previous level, the ranging vector $S(\eta_{R_1+(i_1+1)Q_1})$. In this way each $S(\eta_{R_1+i_1Q_1})$, $1 \leq i_1 \leq [\log n]-1$, guessed at Level 5, is redirected to only one process, i.e., to $\wp_{i_1-1}^{(Q_1)}$.

Level 7 (*Existential*) For each process $\wp_{i_1}^{(Q_1)}$, $0 \leq i_1 \leq [\log n]-1$, $\mathcal{A}$ spawns $\mathcal{O}(c^{\log n})$ existential branches (guesses), each branch holding an (R_2+1)-tuple of ranging vectors $\Re_{R_2} = (S(\eta_{R_1+i_1Q_1}), S(\eta_{R_1+i_1Q_1+1}), ..., S(\eta_{R_1+i_1Q_1+R_2-1}), S(\eta_{R_1+i_1Q_1+R_2}))$. Then $\mathcal{A}$ checks whether all vectors in $\Re_{R_2}$ are correct according to the leftmost-2 derivation requirements. This can be done, for each process $\wp_{i_1}^{(Q_1)}$, $1 \leq i_1 \leq [\log n]-1$, in $\mathcal{O}(\log n)$ time and space, through Levels 8-9 as follows.

Levels 8-9 (*Universal-Existential*) For each branch spawned at Level 7, i.e., for each $0 \leq i_1 \leq [\log n]-1$, $\mathcal{A}$ spawns R_2 universal processes $\wp_v^{(R_2)}$, $1 \leq v \leq R_2$. On each $\wp_v^{(R_2)}$, $\mathcal{A}$ checks whether each substring $\eta_{R_1+i_1Q_1}\eta_{R_1+i_1Q_1+1}...\eta_{R_1+i_1Q_1+v}$ is correct according to the leftmost-2 derivation requirements, and whether each ranging vector in $\Re_{R_2}$ is correct. This is performed as follows. For each $\wp_v^{(R_2)}$, $1 \leq v \leq R_2$, $\mathcal{A}$ counts the number of occurrences of each matrix $m_j \in M$, $1 \leq j \leq k$, in $\eta^{(i_1,v)} = \eta_1\eta_2...\eta_{R_1+i_1Q_1+v-1}$. Denote by $x_{i_1} = R_1 + i_1Q_1$. Suppose that each m_j occurs $c_j^{(i_1,v)}$ times, $0 \leq c_j^{(i_1,v)} \leq x_{i_1} + v - 1$, in $\eta^{(i_1,v)}$. Then $\mathcal{A}$ guesses a $t_j^{(i_1,v)}$-tuple of integers of the form $(c_{j,1}^{(i_1,v)}, c_{j,2}^{(i_1,v)}, ..., c_{j,2^{c_j}-1}^{(i_1,v)}, c_{j,2^{c_j}}^{(i_1,v)})$, where $c_{j,q}^{(i_1,v)}$ with $0 \leq c_{j,q}^{(i_1,v)} \leq c_j^{(i_1,v)}$ and $\sum_{q=1}^{2^{c_j}} c_{j,q}^{(i_1,v)} = c_j^{(i_1,v)}$, represents the number of times the policy ℓ_j^q of matrix m_j, $1 \leq q \leq 2^{c_j}$, can be used when m_j is activated on $\eta^{(i_1,v)}$. $\mathcal{A}$ spawns (Level 9) $\mathcal{N}^{(R_2)} = \mathcal{O}(c_j^{(i_1,v)^{\sum_{j=1}^{k} 2^{c_j}}}) = \mathcal{O}(n^{\sum_{j=1}^{k} 2^{c_j}})$ existential branches, each of which holds k tuples $t_j^{(i_1,v)}$, $1 \leq j \leq k$. On each existential branch, $\mathcal{A}$ computes the sums $s_l^{(i_1,v)} = \sum_{j=1}^{k}\sum_{q=1}^{2^{c_j}} c_{j,q}^{(i_1,v)} V_l(\ell_j^q)$, $1 \leq l \leq m$. Suppose that $\eta_{x_{i_1}+v}$ is a matrix with $2^{c_{\eta_{x_{i_1}+v}}}$ policies, where $c_{\eta_{x_{i_1}+v}} = |\eta_{x_{i_1}+v} \cap F|$, and that each policy $\ell_{\eta_{x_{i_1}+v}}^q$, $1 \leq q \leq 2^{c_{\eta_{x_{i_1}+v}}}$, is identified by the sequence $m_{\eta_{x_{i_1}+v}}^q = (p_{\eta_{x_{i_1}+v},1}^q, p_{\eta_{x_{i_1}+v},2}^q, ..., p_{\eta_{x_{i_1}+v},\xi_{\eta_{x_{i_1}+v}}^q}^q)$, where $|\eta_{x_{i_1}+v} - F| \leq \xi_{\eta_{x_{i_1}+v}}^q \leq |\eta_{x_{i_1}+v}|$.

$\mathcal{A}$ computes the net effect of each rule in $m_{\eta_{x_{i_1}+v}}^q$, i.e., $sdf_{\alpha_{\eta_{x_{i_1}+v},r+1}^q} = s_{\alpha_{\eta_{x_{i_1}+v},r+1}^q}^{(i_1,v)} + \sum_{l=1}^{r} df_{\alpha_{\eta_{x_{i_1}+v},r+1}^q}(p_{\eta_{x_{i_1}+v},l}^q)$, $1 \leq r \leq \xi_{\eta_{x_{i_1}+v}}^q - 1$, and it checks whether

1. $s_{\alpha_{\eta_{x_{i_1}+v},1}^q}^{(v)} \geq 1$, i.e., $p_{\eta_{x_{i_1}+v},1}^q$ the first rule of $\ell_{\eta_{x_{i_1}+v}}^q$, can be applied on $\eta^{(i_1,v)}$,
2. $sdf_{\alpha_{\eta_{x_{i_1}+v},r+1}^q} \geq 1$, $1 \leq r \leq \xi_{\eta_{x_{i_1}+v}}^q - 1$, i.e., rules of policy $\ell_{\eta_{\eta_{x_{i_1}+v}}}^q$ can be applied one by one in the order defined by the sequence $m_{\eta_{x_{i_1}+v}}^q$. Furthermore,

$\mathcal{A}$ checks the following leftmost-2 conditions:

3. $S(\eta_{x_{i_1}+v-1})$ is a possible ranging vector with which matrix $\eta_{x_{i_1}+v-1}$ ends the $(x_{i_1}+v-1)^{th}$ step of derivation, i.e., $S_l(\eta_{x_{i_1}+v-1})=0$, if $s_l^{(i_1,v)}=0$, and $S_l(\eta_{x_{i_1}+v-1})>0$, if $s_l^{(i_1,v)}>0$, $1\leq l\leq m$. The policy $\ell^q_{\eta_{x_{i_1}+v}}$ of $\eta_{x_{i_1}+v}$, can be applied on $S(\eta_{x_{i_1}+v-1})$ in a leftmost-2 manner, i.e., $\mathcal{A}$ checks whether there exists an l, $1\leq l\leq m$, such that $p^q_{\eta_{x_{i_1}+v},1}$, the first rule of $\ell^q_{\eta_{x_{i_1}+v}}$, rewrites A_l and there is no matrix m_j, $m_j\neq\eta_{x_{i_1}+v}$, and no policy $\ell^q_{m_j}$ of m_j, such that the first rule in $\ell^q_{m_j}$ rewrites a nonterminal $A_{l'}$ with $S_{l'}(\eta_{x_{i_1}+v-1})<S_l(\eta_{x_{i_1}+v-1})$, $S_l(\eta_{x_{i_1}+v-1})\neq 0$. Then $\mathcal{A}$ checks whether $S(\eta_{x_{i_1}+v})$ is a possible ranging vector on which $\ell^q_{\eta_{x_{i_1}+v}}$ ends the $(x_{i_1}+v)^{th}$ step of derivation. Note that $S(\eta_{x_{i_1}+v})$ can be nondeterministically computed knowing $S(\eta_{x_{i_1}+v-1})$ and the rules composing $\ell^q_{\eta_{x_{i_1}+v}}$.

Each $\wp_v^{(R_2)}$, $1\leq v\leq R_2$, is said *partially correct* if there exist at least one $t_j^{(i_1,v)}$-tuple (guessed at Level 9) and at least one policy $\ell^q_{\eta_{x_{i_1}+v}}$ of $\eta_{x_{i_1}+v}$, that satisfy conditions $1-3$. If $\wp_v^{(R_2)}$ is not partially correct, it is labeled by 0. Note that, at this moment we cannot decide whether $\wp_v^{(R_2)}$ can be labeled by 1, since we do not know whether $S(\eta_{x_{i_1}})$ is valid, i.e., whether matrix $\eta_{x_{i_1}}$ indeed ends the $x_{i_1}^{th}$ step of derivation with the ranging vector $S(\eta_{x_{i_1}})$, and whether $\eta_{x_{i_1}}$ can be applied in the leftmost-2 derivation manner upon the ranging vector $S(\eta_{x_{i_1}-1})$ (which is not yet guessed[16]). The logical value of each $\wp_v^{(R_2)}$ will be decided at the end of computation, when it will be known whether $S(\eta_{R_1+i_1Q_1})$ is a valid ranging vector with respect to the matrices that compose the subword $\eta_{R_1+(i_1-1)Q_1}...\eta_{R_1+i_1Q_1-1}$. A partially correct process $\wp_v^{(R_2)}$ is labeled by a symbol $\diamond$. If all processes $\wp_v^{(R_2)}$ are labeled by $\diamond$, then the existential branch holding the $\Re_{R_2}$-tuple, provided at Level 7, is labeled by $\diamond$. Otherwise, this branch is labeled by 0. A process $\wp_{i_1}^{(Q_1)}$, yielded at Level 6, will be labeled by $\diamond$ if there exists at least one existential branch labeled by $\diamond$ at Level 7. Otherwise, $\wp_{i_1}^{(Q_1)}$ returns 0.

Suppose that we have run the algorithm up to the $(\ell-1)^{th}$ "iterated" division of n by $[\log n]$, i.e., we know the quotient $Q_{\ell-1}$ and the remainder $R_{\ell-1}$ of $Q_{\ell-2}$ divided by $[\log n]$, i.e., $Q_{\ell-2}=Q_{\ell-1}[\log n]+R_{\ell-1}$. More precisely, $Q_{\ell-2}=Q_{\ell-1}[\log n]+R_{\ell-1}$ and $n=((...((Q_{\ell-1}[\log n]+R_{\ell-1})[\log n]+R_{\ell-2})[\log n]+...)[\log n]+R_2)[\log n]+R_1$, with $Q_{\ell-1}\geq[\log n]$, $0\leq R_l<[\log n]$, $l\in\{1,2,...,\ell-1\}$, and $\ell\leq[\log n]$.

Level 5$(\ell-1)$ (*Existential*) Consider the quotient Q_ℓ and the remainder R_ℓ of $Q_{\ell-1}$ divided by $[\log n]$, $0\leq Q_\ell,R_\ell<[\log n]$. Since $Q_{\ell-2}$, $R_{\ell-2}$ and $R_{\ell-1}$ are no more needed, the space used to record them is now used to record Q_ℓ and R_ℓ, still keeping $Q_{\ell-1}$. Denote by $x_{i_{\ell-2}}=\sum_{l=1}^{\ell-1}R_l+\sum_{l=1}^{\ell-2}i_lQ_l$. For each

[16] $S(\eta_{x_{i_1}-1})$ will be guessed at the last level of the computation tree associated with $\mathcal{A}$, when all the remainders of the "iterated" division of n by $[\log n]$ will be spent, and when $\eta_{R_1+i_1Q_1-1}$ will actually be the last matrix occurring in the suffix of $\eta_{R_1+(i_1-1)Q_1}...\eta_{R_1+i_1Q_1-1}$ of length Q_ℓ, the last quotient of the "iterated" division.

existential branch labeled by $\diamond$ at Level $5\ell-8$, $\mathcal{A}$ spawns $\mathcal{O}(c^{\log n})$ existential branches (guesses), each of which holds a $2\lceil\log n\rceil$-tuple of ranging vectors $\Re^c_{R_\ell} = (S(\eta_{x_{i_{\ell-2}}}), S(\eta_{x_{i_{\ell-2}}+R_\ell}), S(\eta_{x_{i_{\ell-2}}+Q_{\ell-1}}), S(\eta_{x_{i_{\ell-2}}+Q_{\ell-1}+R_\ell}), ..., S(\eta_{x_{i_{\ell-2}}+(\lceil\log n\rceil-1)Q_{\ell-1}}), S(\eta_{x_{i_{\ell-2}}+(\lceil\log n\rceil-1)Q_{\ell-1}+R_\ell}))$, such that $S(\eta_{x_{i_{\ell-2}}})$ is the ranging vector belonging to the tuple $\Re_{R_{\ell-1}}$ found correct at Level $5\ell-8$. Because $\Re_{R_{\ell-1}}$ is no more needed, the space used by $\mathcal{A}$ to record $\Re_{R_{\ell-1}}$ is allocated now to record the tuple $\Re^c_{R_\ell}$. Then $\mathcal{A}$ proceeds with the Level $5\ell-4$, similar to Levels 6, 11, ..., $5\ell-9$.

Level $5\ell-4$ (*Universal*) On each existential branch spawned at Level $5(\ell-1)$, $\mathcal{A}$ spawns $\lceil\log n\rceil$ universal processes $\wp^{(Q_{\ell-1})}_{i_{\ell-1}}$, $0 \le i_{\ell-1} \le \lceil\log n\rceil - 1$. Denote by $x_{i_{\ell-1}} = \sum_{l=1}^{\ell-1} R_l + \sum_{l=1}^{\ell-1} i_l Q_l = x_{i_{\ell-2}} + i_{\ell-1}Q_{\ell-1}$, $0 \le i_{\ell-1} \le \lceil\log n\rceil - 1$. Each process $\wp^{(Q_{\ell-1})}_{i_{\ell-1}}$ takes the interval $[x_{i_{\ell-1}}...x_{i_{\ell-1}}+R_\ell]$, and checks whether the ranging vectors (guessed at Level $5(\ell-1)$) $S(\eta_{x_{i_{\ell-1}}})$ and $S(\eta_{x_{i_{\ell-1}}+R_\ell})$, $0 \le i_{\ell-1} \le \lceil\log n\rceil - 1$, provides a correct order in which the leftmost-2 derivation can be performed between matrices $\eta_{x_{i_{\ell-1}}}$ and $\eta_{x_{i_{\ell-1}}+R_\ell}$. Besides $S(\eta_{x_{i_{\ell-1}}})$ and $S(\eta_{x_{i_{\ell-1}}+R_\ell})$, each $\wp^{(Q_{\ell-1})}_{i_{\ell-1}}$, also keeps, from the previous level, the ranging vector $S(\eta_{x_{i_{\ell-2}}+(i_{\ell-1}+1)Q_{\ell-1}})$. Then $\mathcal{A}$ continues with Level $5\ell-3$, similar to Levels 7, 12, ..., $5\ell-8$.

Level $5\ell-3$ (*Existential*) For each process $\wp^{(Q_{\ell-1})}_{i_{\ell-1}}$, $0 \le i_{\ell-1} \le \lceil\log n\rceil - 1$, $\mathcal{A}$ spawns $\mathcal{O}(c^{\log n})$ existential branches, each branch holding an $(R_\ell+1)$-tuple $\Re_{R_\ell} = (S(\eta_{x_{i_{\ell-1}}}), S(\eta_{x_{i_{\ell-1}}+1}), ..., S(\eta_{x_{i_{\ell-1}}+R_\ell-1}), S(\eta_{x_{i_{\ell-1}}+R_\ell}))$ of ranging vectors. Then $\mathcal{A}$ checks whether all vectors in $\Re_{R_\ell}$ are correct. This can be done, for each process $\wp^{(Q_{\ell-1})}_{i_{\ell-1}}$, $0 \le i_{\ell-1} \le \lceil\log n\rceil - 1$, in $\mathcal{O}(\log n)$ time and space, through Levels $5\ell-2$ (Universal) and $5\ell-1$ (Existential) similar to Levels 3-4, 8-9, ..., $(5\ell-7)$-$(5\ell-6)$.

Levels $(5\ell-2)$-$(5\ell-1)$ (*Universal-Existential*) For each existential branch spawned at Level $5\ell-3$, $\mathcal{A}$ spawns R_ℓ universal processes $\wp^{(R_\ell)}_v$, $1 \le v \le R_\ell$. On each $\wp^{(R_\ell)}_v$, $1 \le v \le R_\ell$, $\mathcal{A}$ spawns $\mathcal{N}^{(R_\ell)} = \mathcal{O}((x_{i_{\ell-1}}+v)^{\sum_{j=1}^{k} 2^{c_j}}) = \mathcal{O}(n^{\sum_{j=1}^{k} 2^{c_j}})$ existential branches, each of which holds a possible configuration of policies used by matrices occurring in $\eta^{(i_{\ell-1},v)} = \eta_1\eta_2...\eta_{x_{i_{\ell-1}}+v-1}$, and computes the net effect according to this configuration. $\mathcal{A}$ guesses a policy $\ell^q_{\eta_{x_{i_{\ell-1}}+v}}$ and, based on the net effect computed before, checks whether $\eta_{x_{i_{\ell-1}}+v}$ with the policy $\ell^q_{x_{i_{\ell-1}}+v}$ can be applied, in leftmost-2 derivation manner, on the sentential form having the associated ranging vector $S(\eta_{x_{i_{\ell-1}}+v-1})$ in $\Re_{R_\ell}$. Then $\mathcal{A}$ checks whether $S(\eta_{x_{i_{\ell-1}}+v})$ in $\Re_{R_\ell}$ is a possible ranging vector on which $\ell^q_{\eta_{x_{i_{\ell-1}}+v}}$ ends the $(x_{i_{\ell-1}}+v)^{th}$ step of derivation. Note that $S(\eta_{x_{i_{\ell-1}}+v})$ can be nondeterministically computed, knowing the ranging vector $S(\eta_{x_{i_{\ell-1}}+v-1})$ and the sequence of rules that defines $\ell^q_{\eta_{x_{i_{\ell-1}}+v}}$.

Each process $\wp^{(R_\ell)}_v$, $1 \le v \le R_\ell$, that satisfies the above conditions is partially correct, and it is labeled by $\diamond$. Otherwise, $\wp^{(R_\ell)}_v$ is labeled by 0. If all $\wp^{(R_\ell)}_v$ are labeled by $\diamond$, then the existential branch holding the tuple $\Re_{R_\ell}$, provided at Level $5\ell-3$, is labeled by $\diamond$. Otherwise, this branch is labeled by 0. The process $\wp^{(Q_{\ell-1})}_{i_{\ell-1}}$, yielded at Level $5\ell-4$, will be labeled by $\diamond$ if there exists at least one existential branch labeled by $\diamond$ at Level $5\ell-3$. Otherwise, $\wp^{(Q_{\ell-1})}_{i_{\ell-1}}$ is labeled by 0.

At this level the only substrings of η left unchecked are those substrings that corresponds to intervals $I_{Q_{\ell-1}} = [\sum_{l=1}^{\ell-1} R_l + \sum_{l=1}^{\ell-2} i_l Q_l + i_{\ell-1} Q_{\ell-1} + R_\ell ... \sum_{l=1}^{\ell-1} R_l + \sum_{l=1}^{\ell-2} i_l Q_l + (i_{\ell-1}+1) Q_{\ell-1}] = [x_{i_{\ell-2}} + i_{\ell-1} Q_{\ell-1} + R_\ell ... x_{i_{\ell-2}} + (i_{\ell-1}+1) Q_{\ell-1}]$, $0 \leq i_l \leq [\log n] - 1$, $1 \leq l \leq \ell - 1$, and besides the cutting points $P_\ell^u = \sum_{l=1}^{u} R_l + \sum_{l=1}^{u-1} i_l Q_l + (i_u + 1) Q_u$, $1 \leq u \leq \ell - 1$. On each interval of type $I_{Q_{\ell-1}}$, $\mathcal{A}$ proceeds with Level 5ℓ.

Level 5ℓ (*Existential*) Each interval $I_{Q_{\ell-1}}$ can be divided into $[\log n]$ subintervals of length $1 \leq Q_\ell < [\log n]$. Hence, $\mathcal{A}$ spawns $\mathcal{O}(c^{\log n})$ existential branches each of which holds a $[\log n]$-tuple of ranging vectors $\Re^c_{Q_\ell} = (S(\eta_{x_{i_{\ell-1}} + R_\ell}), S(\eta_{x_{i_{\ell-1}} + R_\ell + Q_\ell}), ..., S(\eta_{x_{i_{\ell-1}} + R_\ell + ([\log n]-1) Q_\ell}))$, where $S(\eta_{x_{i_{\ell-1}} + R_\ell})$ is the ranging vector found valid at Level $5\ell - 3$.

Level $5\ell+1$ (*Universal*) For each existential branch spawned at Level 5ℓ, $\mathcal{A}$ spawns $[\log n]$ universal processes $\wp_{i_\ell}^{(Q_\ell)}$, $0 \leq i_\ell \leq [\log n] - 1$. Each $\wp_{i_\ell}^{(Q_\ell)}$ takes an interval of length Q_ℓ of the form $[\sum_{l=1}^{\ell} R_l + \sum_{l=1}^{\ell-1} i_l Q_l + i_\ell Q_\ell ... \sum_{l=1}^{\ell} R_l + \sum_{l=1}^{\ell-1} i_l Q_l + (i_\ell + 1) Q_\ell]$. Denote by $x_{i_\ell} = \sum_{l=1}^{\ell} R_l + \sum_{l=1}^{\ell-1} i_l Q_l + i_\ell Q_\ell$, $0 \leq i_\ell \leq [\log n] - 1$. For each interval $[x_{i_\ell} ... x_{i_\ell + 1}]$, $\mathcal{A}$ checks whether the substring $\eta_{x_{i_\ell}} ... \eta_{x_{i_\ell+1}}$, $0 \leq i_\ell \leq [\log n] - 1$, is valid according to the leftmost-2 derivation order.

Level $5\ell + 2$ (*Existential*) For each $\wp_{i_\ell}^{(Q_\ell)}$, $0 \leq i_\ell \leq [\log n] - 1$, $\mathcal{A}$ spawns $\mathcal{O}(c^{\log n})$ existential branches, each branch holding an $(Q_\ell + 1)$-tuple of ranging vectors $\Re_{Q_\ell} = (S(\eta_{x_{i_\ell}}), S(\eta_{x_{i_\ell}+1}), ..., S(\eta_{x_{i_\ell}+Q_\ell-1}), S(\eta_{x_{i_\ell+1}}))$. In each $\Re_{Q_\ell}$-tuple the first vector $S(\eta_{x_{i_\ell}})$ and the last vector $S(\eta_{x_{i_\ell+1}})$ have been guessed at Level 5ℓ. They are ranging vectors associated with matrices placed in cutting points, i.e., end points of intervals of length at most $\log n$. They are also overlapping points of two consecutive intervals of type $[x_{i_\ell} ... x_{i_\ell+1}]$. Hence, each ranging vector $S(\eta_{x_{i_\ell}})$ is checked two times. Once if it is a valid vector on which $\eta_{x_{i_\ell}+1}$ can be applied in leftmost-2 derivation manner, and twice if by applying $\eta_{x_{i_\ell}}$ on the sentential form built by using the ranging vector $S(\eta_{x_{i_\ell}-1})$ a sentential form with the ranging vector $S(\eta_{x_{i_\ell}})$ is obtained.

As all intervals of type $[x_{i_\ell} ... x_{i_\ell+1}]$ are universally checked, the tuple $\Re^c_{Q_\ell}$ spawned at Level 5ℓ is labeled by 1, if all ranging vectors $S(\eta_{x_{i_\ell}})$ and all vectors composing $\Re_{Q_\ell}$ are correct. To check whether all ranging vectors in $\Re_{Q_\ell}$ are correct, for each process $\wp_{i_\ell}^{(Q_\ell)}$, $0 \leq i_\ell \leq [\log n] - 1$, $\mathcal{A}$ follows the same procedure, that requires $\mathcal{O}(\log n)$ time and space, described at Levels $5\ell - 2$ (Universal) and $5\ell - 1$ (Existential).

For the last substring of length Q_ℓ in η, i.e., the suffix of η of length Q_ℓ of the form $\eta_{\sum_{l=1}^{\ell} R_l + \sum_{l=1}^{\ell-1} ([\log n]-1) Q_l + ([\log n]-1) Q_\ell} ... \eta_{\sum_{l=1}^{\ell} R_l + \sum_{l=1}^{\ell-1} ([\log n]-1) Q_l + [\log n] Q_\ell}$, on $\wp_{[\log n]-1}^{(Q_\ell)}$ $\mathcal{A}$ must check whether the matrix $\eta_{\sum_{l=1}^{\ell} R_l + \sum_{l=1}^{\ell-1} ([\log n]-1) Q_l + [\log n] Q_\ell} = \eta_n$ ends the computation. This can be checked as in process $\wp_n$, Theorem 3.

Each cutting point $P_\ell^u = \sum_{l=1}^{u} R_l + \sum_{l=1}^{u-1} i_l Q_l + (i_u + 1) Q_u$ can be equivalently rewritten as $\sum_{l=1}^{u+1} R_l + \sum_{l=1}^{u} i_l Q_l + [\log n] Q_{u+1}$, due to the equality $Q_u = [\log n] Q_{u+1} + R_{u+1}$, for any $1 \leq u \leq \ell - 1$. Furthermore, $\sum_{l=1}^{u+1} R_l + \sum_{l=1}^{u} i_l Q_l + [\log n] Q_{u+1}$ is equal with $\sum_{l=1}^{u+1} R_l + R_{u+2} + \sum_{l=1}^{u} i_l Q_l + ([\log n]-1) Q_{u+1} + [\log n] Q_{u+2}$, due to the equality $Q_{u+1} = [\log n] Q_{u+2} + R_{u+2}$, for any $1 \leq u \leq \ell - 2$. By applying

this transformation k times, where $k = \ell - u$, each P_ℓ^u can be equivalently rewritten as $\sum_{l=1}^{u+1} R_l + R_{u+2} + ... + R_{u+k} + \sum_{l=1}^{u} i_l Q_l + ([\log n] - 1)(Q_{u+1} + ... + Q_{u+k-1}) + [\log n]\, Q_{u+k}$, where $u + k = \ell$. In this way each P_ℓ^u, yielded at Level $5u$ by the $\Re^c_{R_{u+1}}$-tuple, $1 \leq u \leq \ell - 1$, is in fact the end point of an interval of the form $[\sum_{l=1}^{\ell} R_l + \sum_{l=1}^{\ell-1} i_l Q_l + i_\ell Q_\ell ... \sum_{l=1}^{\ell} R_l + \sum_{l=1}^{\ell-1} i_l Q_l + (i_\ell + 1) Q_\ell] = [x_{i_\ell} ... x_{i_\ell + 1}]$, for which $0 \leq i_l \leq [\log n] - 1$, $1 \leq l \leq \ell - 1$, $i_\ell = [\log n] - 1$. Hence, the decision on the correctness of each ranging vector $S(\eta_{\sum_{l=1}^{u} R_l + \sum_{l=1}^{u-1} i_l Q_l + (i_u + 1) Q_u}) = S(\eta_{P_\ell^u})$ will be actually taken by a process of type $\wp^{(Q_\ell)}_{[\log n]-1}$.

Since the validity of each cutting point is decided by a process of type $\wp^{(Q_\ell)}_{[\log n]-1}$, the logical value returned by this process is "propagated" up to the level of the computation tree that has spawned the corresponding cutting point, and thus each $\diamond$ symbol receives a logical value. The input is accepted, if going up in the computation tree, with all $\diamond$'s changed into logical values, the root of the tree is labeled by 1.

The $\Re_{R_\hbar}$, $\Re^c_{R_\hbar}$, $\Re_{Q_\ell}$, and $\Re^c_{Q_\ell}$-tuples of ranging vectors, $1 \leq \hbar \leq \ell$, the sequences m_j^q, the vectors $V(\ell_j^q)$, $1 \leq j \leq k$, and auxiliary net effects computed by $\mathcal{A}$ during the algorithm, are stored by using $\mathcal{O}(\log n)$ space. It is easy to observe that $\mathcal{A}$ has $\mathcal{O}(\log n)$ levels. Since at each level $\mathcal{A}$ spawns either $\mathcal{O}(n^c)$ or $\mathcal{O}(c^{\log n})$ existential branches, where c is a constant (independent of the length of the input), each level is thus convertible into a binary tree with $\mathcal{O}(\log n)$ levels. Moreover, at each Level $5\hbar$, $1 \leq \hbar \leq \ell$, $\mathcal{A}$ performs a division operation, which requires $\mathcal{O}(\log n)$ time and space. Consequently, $\mathcal{A}$ performs the whole computation in $\mathcal{O}(\log^2 n)$ parallel time and $\mathcal{O}(\log n)$ space. □

Corollary 6 *Each language $L \in SZML_i(CF)$, $i \in \{1, 2, 3\}$, can be recognized by an indexing ATM in $\mathcal{O}(\log n)$ space and $\mathcal{O}(\log^2 n)$ time.*

Proof. The proof is similar to the proof provided for Theorem 6. The main difference is that at each Level $5\hbar + 4$, $0 \leq \hbar \leq \ell$, $\mathcal{A}$ does not have to spawn $\mathcal{N}^{(R_{\hbar+1})} = \mathcal{O}(R_{\hbar+1}^{\sum_{j=1}^{k} 2^{c_j}})$ existential branches in order to guess the $t_j^{(i_\hbar, v)}$-tuples of integers that provide the number of times each policy of m_j can be used in the substring $\eta^{(i_\hbar, v)}$, for each matrix m_j, $1 \leq j \leq k$. However, this does not decrease the time resources needed, since $\mathcal{A}$ has to perform $\log n$ division operations, each of which requiring $\mathcal{O}(\log n)$ time and space. Hence, the parallel time is still $\mathcal{O}(\log^2 n)$. □

Corollary 7 $SZML_i(CF) \cup SZML_i^{ac}(CF) \subset \mathcal{NC}^2$, $i \in \{1, 2, 3\}$.

Corollary 8 $SZML_i(CF) \cup SZML_i^{ac}(CF) \subset DSPACE(\log^2 n)$, $i \in \{1, 2, 3\}$.

4 Szilard Languages of Random Context Grammars

Random Context Grammars (RCGs) are regulated rewriting grammars in which the application of a rule is enabled by the existence in the current sentential form of some nonterminals that provide the *context* under which the rule in question can be applied. These nonterminals are listed by the so called *permitting context* of that rule. The use of a rule may be disabled by the existence, in the current sentential

form, of some nonterminals that provide the forbidden context under which the rule in question cannot be applied. These nonterminals are listed by the so called *forbidding context* of that rule.

RCGs with context-free rules have been first introduced in [47] to cover the gap existing between the classes CFLs and CSLs. A generalization of RCGs for phrase-structure rules can be found in [14]. The generative capacity and several descriptional properties of RCGs can be found in [10], [12], [14], [17], [47], and [48].

4.1 Random Context Grammars - Prerequisites

Definition 12 A *random context grammar* (RCG) is a quadruple $G = (N, T, S, P)$ where S is the axiom, N and T, $N \cap T = \emptyset$, are finite sets of *nonterminals* and *terminals*, respectively. P is a finite set of *triplets* (random context rules) of the form $r = (p_r, Q_r, R_r)$, where $p_r : \alpha_r \rightarrow \beta_r$ is a phrase structure rule over $N \cup T$, i.e., $\alpha_r \in (N \cup T)^* N (N \cup T)^*$ and $\beta_r \in (N \cup T)^*$, Q_r and R_r are subsets of N, called the *permitting* and *forbidding context* of r, respectively. If $R_r = \emptyset$, for any $r \in P$, then G is a *permitting* RCG. If $Q_r = \emptyset$, for any $r \in P$, then G is a *forbidding* RCG.

A *permitting* RCG is a RCG without appearance checking, i.e., $R_r = \emptyset$ for any $r \in P$. If there exists at least one $r \in P$ such that $R_r \neq \emptyset$, then G is a RCG with appearance checking (henceforth RCG^{ac}). If $Q_r = \emptyset$ for any $r \in P$, then G is called *forbidding* RCG. If all rules p_r are phrase-structure (PS), context-sensitive (CS), context-free (CF), or regular (REG) rules then G is a PS, CS, CF, or REG random context grammar.

Definition 13 Let $G = (N, T, S, P)$ be a RCG or a RCG^{ac}, and $V = N \cup T$. The language $L(G)$ generated by G is defined as the set of all words $w \in T^*$ for which there exists a derivation D: $S = w_0 \Rightarrow_{r_{i_1}} w_1 \Rightarrow_{r_{i_2}} w_2 \Rightarrow_{r_{i_3}} ... \Rightarrow_{r_{i_s}} w_s = w$, $s \geq 1$, where $r_{i_j} = (\alpha_{i_j} \rightarrow \beta_{i_j}, Q_{i_j}, R_{i_j})$, $1 \leq j \leq s-1$, $w_{j-1} = w'_{j-1}\alpha_{i_j}w''_{j-1}$, $w_i = w'_{j-1}\beta_{i_j}w''_{j-1}$ for some w'_{j-1}, $w''_{j-1} \in V^*$, such that *i.* all symbols in Q_{i_j} occur in $w'_{j-1}w''_{j-1}$, and *ii.* no symbol of R_{i_j} occurs in $w'_{j-1}w''_{j-1}$.

Denote by $L(RC, X)$ and $L(RC, X, ac)$ the class of languages generated by RCGs without appearance checking and RCGs with appearance checking, respectively, with X-rules, $X \in \{REG, CF, CF - \lambda, CS, PS\}$, then $L(RC, X, ac) = L(M, X, ac)$, $L(RC, Y) = L(M, Y)$, $Y \in \{REG, CS, PS\}$, $L(RC, CF) \subseteq L(M, CF)$, $L(RC, CF - \lambda) \subseteq L(M, CF - \lambda)$ [12], [14]. Hence
1. $CFL \subset L(RC, CF - \lambda) \subset L(RC, CF - \lambda, ac) \subset CSL \subset L(RC, CF, ac) = RE$,
2. $CFL \subset L(RC, CF - \lambda) \subseteq L(RC, CF) \subset L(RC, CF, ac) = RE$,
3. $L(RC, X) = L(RC, X, ac) = XL$, $X \in \{REG, CS, PS\}$.

Let $G = (N, T, S, P)$ be a RCG. If labels are associated with triplets[17] $r = (p, Q, R) \in P$, in one-to-one correspondence, then the Szilard language associated with a *RCG* is defined as follows.

[17]For the sake of simplicity, we use the same notation both for a triple and the label associated with it.

Definition 14 Let $G = (N, T, S, P)$ be a RCG, $P = \{r_1, r_2, ..., r_k\}$ the set of productions, $L(G)$ the language generated by G, and w a word in $L(G)$. The *Szilard word* of w associated with the derivation D: $S = w_0 \Rightarrow_{r_{i_1}} w_1 \Rightarrow_{r_{i_2}} w_2 \Rightarrow_{r_{i_3}} ... \Rightarrow_{r_{i_s}} w_s = w$, $s \geq 1$, is defined as $Sz_D(w) = r_{i_1} r_{i_2} ... r_{i_s}$, $r_{i_j} \in P$, $1 \leq j \leq s$. The *Szilard language* of G is $Sz(G) = \{Sz_D(w) | w \in L(G), D \text{ is a terminal derivation of } w\}$.

Denote by $SZRC(X)$ and $SZRC(X)^{ac}$ the classes of Szilard languages associated with RCGs without appearance checking and RCGs with appearance checking and X rules, $X \in \{CF, CS, PS\}$, respectively.

Definition 10 is applicable also for leftmost-i, $i \in \{1, 2, 3\}$, derivations in RCGs with CF rules [14]. In terms of triplets $r = (p_r, Q_r, R_r) \in P$, where p_r is a CF rule of the form $\alpha_{p_r} \rightarrow \beta_{p_r}$, $\alpha_{p_r} \in N$, $\beta_{p_r} \in (N \cup T)^*$, and Q_r and R_r are the permitting and forbidding context of r, respectively, these derivations can be explained as follows.

A production $r = (p_r, Q_r, R_r) \in P$ can be applied in leftmost-1 derivation manner if p_r rewrites the leftmost nonterminal occurring in the sentential form, as long as the sentential form on which r is applied contains all nonterminals in Q_r and no nonterminal in R_r.

A production $r = (p_r, Q_r, R_r) \in P$ can be applied in leftmost-2 derivation manner if the rule p_r rewrites the leftmost nonterminal that can be rewritten by any rule in P eligible to be applied on the current sentential form, in the sense that if any other rule $r' = (p_{r'}, Q_{r'}, R_{r'}) \in P$ can be applied, because the sentential form contains all nonterminals in $Q_{r'}$ and no nonterminal in $R_{r'}$, then the nonterminal rewritten by r' follows in the sentential form the nonterminal rewritten by r.

A production $r = (p_r, Q_r, R_r) \in P$ can be applied in leftmost-3 derivation manner if the rule p_r rewrites the leftmost nonterminal that can be rewritten by r, as long as the sentential form on which r is applied contains all nonterminals in Q_r and no nonterminal in R_r.

Szilard languages associated with leftmost-i, $i \in \{1, 2, 3\}$, derivations can be defined in the same way as in Definition 14, with the specification that D is a leftmost-i derivation of w. We denote by $SZRCL_i(X)$ and $SZRCL_i^{ac}(X)$ the classes of leftmost-i, $i \in \{1, 2, 3\}$, Szilard languages associated with RCGs and RCGs with appearance checking with X rules, $X \in \{CF, CS, PS\}$, respectively.

Let $G = (N, T, P, A_1)$ be an arbitrary RCG with CF rules, where A_1 is the axiom, $N = \{A_1, A_2, ..., A_m\}$ and $P = \{r_1, r_2, ..., r_k\}$ are the finite sets of ordered nonterminals and labels associated in one-to-one correspondence, respectively. For each production $r = (p_r, Q_r, R_r) \in P$, where p_r is a rewriting rule of the form $\alpha_{p_r} \rightarrow \beta_{p_r}$, $\alpha_{p_r} \in N$, and $\beta_{p_r} \in (N \cup T)^*$, its *net effect* during the derivation D with respect to each nonterminal $A_l \in N$, $1 \leq l \leq m$, is given by $df_{A_l}(p_r) = |\beta_{p_r}|_{A_l} - |\alpha_{p_r}|_{A_l}$. To each rule r we associate a vector $V(r) \in \mathbf{Z}^m$ defined by $V(r) = (df_{A_1}(p_r), df_{A_2}(p_r), ..., df_{A_m}(p_r))$, where $\mathbf{Z}$ is the set of integers. The value of $V(r)$ taken at the l^{th} place, $1 \leq l \leq m$, is denoted by $V_l(r)$.

4.2 On the Complexity of Unrestricted Szilard Languages

Theorem 7 *Each language $L \in SZRC(CF) \cup SZRC^{ac}(CF)$ can be recognized by an indexing ATM in $\mathcal{O}(\log n)$ time and space ($SZRC(CF) \cup SZRC^{ac}(CF) \subseteq ALOGTIME$).*

Proof. We give the proof for the class $SZRC^{ac}(CF)$. For the class $SZRC(CF)$ the proof is simpler. Let $G = (N, T, P, A_1)$ be an arbitrary RCG^{ac} with CF rules. We describe an indexing ATM that decides in logarithmic time and space whether an input word $\gamma = \gamma_1\gamma_2...\gamma_n \in P^*$ of length n, belongs to $Sz(G)$. Let $\mathcal{A}$ be an indexing ATM composed of an input tape that stores γ, an index tape, and a working tape composed of three tracks. Here and throughout this paper, each label γ_i corresponds to a triplet in P of the form $(p_{\gamma_i}, Q_{\gamma_i}, R_{\gamma_i})$, where p_{γ_i} is a CF rule of the form $\alpha_{\gamma_i} \rightarrow \beta_{\gamma_i}$, $\alpha_{\gamma_i} \in N$, and $\beta_{\gamma_i} \in (N \cup T)^*$, $1 \leq i \leq n$. At the beginning of the computation the first track of the working tape of $\mathcal{A}$ stores $k+1$ vectors, V^0 corresponding to the axiom, i.e., $V_1^0 = 1$ and $V_l^0 = 0$, $2 \leq l \leq m$, and $V(r_j)$, $1 \leq j \leq k$. The other two tracks are initially empty.

Level 1 (*Existential*) In an existential state $\mathcal{A}$ guesses the length of γ, i.e., writes on the index tape n, and checks whether the n^{th} cell of the input tape contains a terminal symbol and the cell $n+1$ contains no symbol. The correct value of n is recorded in binary on the second track of the working tape.

Level 2 (*Universal*) $\mathcal{A}$ spawns n universal processes $\wp_i$, $1 \leq i \leq n$.

• On $\wp_1$, $\mathcal{A}$ checks whether $\alpha_{\gamma_1} = A_1$. Process $\wp_1$ returns 1 if this equality holds.

• For each $\wp_i$, $2 \leq i \leq n$, $\mathcal{A}$ counts the number of occurrences of each rule $r_j \in P$, $1 \leq j \leq k$, in $\gamma^{(i)} = \gamma_1\gamma_2...\gamma_{i-1}$. Suppose that each r_j occurs $c_j^{(i)}$ times, $0 \leq c_j^{(i)} \leq i-1$, in $\gamma^{(i)}$. $\mathcal{A}$ computes $s_{A_l}^{(i)} = V_l^0 + \sum_{j=1}^{k} c_j^{(i)} V_l(r_j)$, i.e., the number of times each nonterminal A_l, $1 \leq l \leq m$, occurs in the sentential form obtained at the i^{th} step of derivation. Besides, for $\wp_n$, for each $1 \leq l \leq m$, $\mathcal{A}$ computes $s_{A_l}^{(n,out)} = V_l^0 + \sum_{j=1}^{k} c_j^{(n)} V_l(r_j) + V_l(\gamma_n)$. Each $\wp_i$, $2 \leq i \leq n-1$, returns 1 if only one of the conditions $1-3$ holds. Process $\wp_n$ returns 1, if one of the conditions $1-3$ holds, and besides $s_{A_l}^{(n,out)} = 0$, for each $1 \leq l \leq m$.

1. $s_{\alpha_{\gamma_i}}^{(i)} \geq 1$, $\alpha_{\gamma_i} \notin Q_{\gamma_i} \cup R_{\gamma_i}$, $s_X^{(i)} \geq 1$, for each $X \in Q_{\gamma_i}$, and $s_Y^{(i)} = 0$ for each $Y \in R_{\gamma_i}$,
2. $s_{\alpha_{\gamma_i}}^{(i)} \geq 2$ if $\alpha_{\gamma_i} \in Q_{\gamma_i} - R_{\gamma_i}$, $s_X^{(i)} \geq 1$, for each $X \in Q_{\gamma_i}$, $X \neq \alpha_{\gamma_i}$, and $s_Y^{(i)} = 0$ for each $Y \in R_{\gamma_i}$,
3. $s_{\alpha_{\gamma_i}}^{(i)} = 1$ if $\alpha_{\gamma_i} \in R_{\gamma_i} - Q_{\gamma_i}$, $s_X^{(i)} \geq 1$, for each $X \in Q_{\gamma_i}$, and $s_Y^{(i)} = 0$ for each $Y \in R_{\gamma_i}$, $Y \neq \alpha_{\gamma_i}$.

The computation tree of $\mathcal{A}$ has only two levels, in which each node has unbounded out-degree. By using a divide and conquer algorithm each of these levels can be converted into a binary tree of height $\mathcal{O}(\log n)$. All functions used in the algorithm, such as counting and addition, are in $\mathcal{NC}^1$, which is equal to ALOGTIME under the U_{E^*}-uniformity restriction [41]. In order to store, on the third track of the working tape, the binary value of $c_j^{(i)}$, and to compute in binary $s_{A_l}^{(i)}$ and $s_{A_l}^{(n,out)}$, $1 \leq i \leq n$, $1 \leq j \leq k$, $1 \leq l \leq m$, $\mathcal{A}$ needs $\mathcal{O}(\log n)$ space. Hence, for the whole computation $\mathcal{A}$ uses $\mathcal{O}(\log n)$ time and space. □

Corollary 9 $SZRC(CF) \cup SZRC^{ac}(CF) \subset \mathcal{NC}^1$.

Corollary 10 $SZRC(CF) \cup SZRC^{ac}(CF) \subset DSPACE(\log n)$.

4.3 On the Complexity of Leftmost Szilard Languages

Theorem 8 *Each language $L \in SZRCL_1^{ac}(CF)$ can be recognized by an indexing ATM in $\mathcal{O}(\log n)$ time and space ($SZRCL_1^{ac}(CF) \subseteq ALOGTIME$).*

Proof. Let $G = (N, T, P, A_1)$ be a RCG^{ac} with CF rules working in leftmost-1 derivation manner. Consider an indexing ATM $\mathcal{A}$ having a similar structure as in the proof of Theorem 7. Let $\gamma = \gamma_1\gamma_2...\gamma_n \in P^*$, be an input word of length n. In order to guess the length of γ, $\mathcal{A}$ proceeds with the procedure described at Level 1-*Existential*, Theorem 7. Then $\mathcal{A}$ spawns (Level 2-*Universal*) n universal processes $\wp_i$, $1 \leq i \leq n$, and (briefly) proceeds as follows.

For each $\wp_i$, $1 \leq i \leq n$, $\mathcal{A}$ checks as in Theorem 7, whether each triplet γ_i can be applied on $\gamma^{(i)} = \gamma_1\gamma_2...\gamma_{i-1}$ according to Definition 13. Then $\mathcal{A}$ checks whether rule p_{γ_i} can be applied in a leftmost-1 derivation manner on $\gamma^{(i)}$. To do so, $\mathcal{A}$ spawns at most $i-1$ existential branches (Level 3-*Existential*) each branch corresponding to a label γ_v, $1 \leq v \leq i-1$, such that p_{γ_v} in $(p_{\gamma_v}, Q_{\gamma_v}, R_{\gamma_v})$ is a non-terminal rule. Denote by q the number of non-terminal rules used in γ between γ_{v+1} and γ_{i-1} (including γ_{v+1} and γ_{i-1}), and by s_q the total number of nonterminals produces by these rules, and let $s = i - v - s_q$. $\mathcal{A}$ checks whether α_{γ_i} is the s^{th} nonterminal occurring on the right-hand side[18] of rule p_{γ_v}.

An existential branch spawned at Level 3, is labeled by 1 if p_{γ_v} satisfies these properties. For each existential branch labeled by 1 at Level 3, $\mathcal{A}$ checks whether the s^{th} nonterminal occurring in β_{γ_v} is indeed the α_{γ_i} nonterminal rewritten by rule p_{γ_i}, i.e., no other rule used between rule p_{γ_v} of γ_v and rule p_{γ_i} of γ_i rewrites the s^{th} nonterminal, equal to α_{γ_i}, in β_{γ_v} (for which a relation of type "$s + s_q = i - v$" may also hold). Hence, $\mathcal{A}$ universally branches (Level 4-*Universal*) all symbols occurring between rules γ_{v+1} and γ_{i-1}. On each branch holding a triplet $\gamma_l = (p_{\gamma_l}, Q_{\gamma_l}, R_{\gamma_l})$, $v < l < i$, $\mathcal{A}$ checks whether 1. α_{γ_l} equals α_{γ_i},

2. $s + \bar{s}_q = l - v$, providing that α_{γ_i} is the s^{th} nonterminal occurring on the right-hand side of rule p_{γ_v} (found at Level 3) and $\bar{s}_q$ is the number of nonterminals produced between rules p_{γ_v} and p_{γ_l},

3. the number of nonterminals α_{γ_i} rewritten between p_{γ_v} and p_{γ_l} is equal to the number of nonterminals α_{γ_i} produced between these rules, up to the s^{th} nonterminal occurring on the right-hand side of rule p_{γ_v}.

On each universal branch (Level 4) $\mathcal{A}$ returns 0 if conditions $1-3$ hold. Otherwise, it returns 1. Note that, for each $\wp_i$, $1 \leq i \leq n$, $\mathcal{A}$ does not have to check whether γ_v and γ_l, can be applied in leftmost-1 derivation manner. This condition is checked by each of the processes $\wp_v$ and $\wp_l$, since all of them are universally considered. It is easy to estimate that $\mathcal{A}$ performs the whole computation in logarithmic time and space. □

[18] If p_{γ_v} is the rule that produces the nonterminal rewritten by rule p_{γ_i}, and this is the s^{th} nonterminal occurring on the right-hand side of p_{γ_v}, then for the case of leftmost-1 derivation order, we must have $s + s_q = i - v$. This is because each nonterminal produced in the sentential form by rules used in a leftmost-1 derivation manner, between p_{γ_v} and p_{γ_i} (including nonterminals existing up to the s^{th} nonterminal on the right-hand side of p_{γ_v}), must be fully rewritten by these rules. The nonterminals existing in the sentential form before p_{γ_v} has been applied, will be rewritten only after the new nonterminals produced between p_{γ_v} and p_{γ_i} are fully rewritten.

Corollary 11 $SZRCL_1(CF) \cup SZRCL_1^{ac}(CF) \subset \mathcal{NC}^1$.

Corollary 12 $SZRCL_1(CF) \cup SZRCL_1^{ac}(CF) \subset DSPACE(\log n)$.

In order to simulate letfmost-i derivations, $i \in \{2,3\}$, and to check whether $\gamma = \gamma_1\gamma_2...\gamma_n \in P^*$ belongs to $SZRCL_i^{ac}(CF)$, for each triplet γ_i, $1 \leq i \leq n$, an ATM must have information concerning the order in which the first occurrence of each nonterminal $A_l \in N$, $1 \leq l \leq m$, occurs in the sentential form at any step of derivation. In this respect we introduce the notion of *ranging vector* for a RCG.

Definition 15 Let $G = (N, T, P, A_1)$ be a RCG^{ac} with CF rules, where $P = \{r_1, r_2, ..., r_k\}$ is the ordered finite set of triplets in P. Let SF_{r_j} be the sentential form obtained after the triplet $r_j = (p_j, Q_j, R_j)$, $1 \leq j \leq k$, has been applied at a certain step of derivation in G. The *ranging vector* associated with SF_{r_j}, denoted by $S(r_j)$, $1 \leq j \leq k$, is a vector in $\mathbf{N}^m$ defined as

$$S_l(r_j) = \begin{cases} 0, & \text{if } A_l \in N \text{ does not occur in } SF_{r_j}, \text{ i.e., } |SF_{r_j}|_{A_l} = 0, \\ i, & \text{if the first occurrence of } A_l \text{ in } SF_{r_j} \text{ is the } i^{th} \text{ element in the} \\ & \text{order of first occurrences of nonterminals from } N \text{ in } SF_{r_j}. \end{cases}$$

Depending on the context, the value of $S(r_j)$ taken at the l^{th} place, $1 \leq l \leq m$, i.e., $S_l(r_j)$, is also denoted by $S_{\alpha_{p_j}}(r_j)$ if p_j in $r_j = (p_j, Q_j, R_j)$ is a CF rule of the form $\alpha_{p_j} \to \beta_{p_j}$ and $\alpha_{p_j} = A_l$.

Note that, if $r_{j'} = (p_{j'}, Q_{j'}, R_{j'})$ is applied in the Szilard word before $r_j = (p_j, Q_j, R_j)$ then the ranging vector $S(r_j)$ can be computed knowing $S(r_{j'})$. This observation holds for both leftmost-2 and leftmost-3 derivations (Example 2).

Example 2 Consider $S(r_{j'}) = (3, 0, 2, 1, 0) \in \mathbf{N}^5$ the ranging vector associated with the sentential form $SF_{r_{j'}}$, obtained after rule $r_{j'}$ has been applied, at the i^{th} step of derivation. Suppose that $SF_{r_{j'}}$ contains one occurrence of A_1, three occurrences of A_3, and arbitrary number of A_4. According to Definition 15, $SF_{r_{j'}}$ looks like $SF_{r_{j'}} = tA_4X_4A_3X_{3,4}A_1\bar{X}_{3,4}$, where $t \in T^*$, $X_4 \in (\{A_4\} \cup T)^*$, $X_{3,4}, \bar{X}_{3,4} \in (\{A_3, A_4\} \cup T)^*$. If in $r_j = (p_j, Q_j, R_j)$, p_j is the rule $A_3 \to tA_5$, $Q_j = \{A_3, A_4\}$ and $R_j = \{A_5\}$, then r_j can be applied in leftmost-2 derivation manner after $r_{j'}$, if there is no other RC rule $r_{j''} = (p_{j''}, Q_{j''}, R_{j''}) \in P$, such that $p_{j''}$ rewrites A_4, $SF_{r_{j'}}$ contains all nonterminals in $Q_{j''}$ and no nonterminal in $R_{j''}$. Depending on the position of the second occurrence of A_3 in $SF_{r_{j'}}$, the sentential form obtained after p_j has been applied on $SF_{r_{j'}}$ may look like

- $SF_{r_j} = tA_4X_4A_5A_3X_{3,4}A_1\bar{X}_{3,4}$ or $SF_{r_j} = tA_4X_4A_5\bar{X}_4A_3X_{3,4}A_1\bar{X}_{3,4}$, $t \in T^*$, $X_4, \bar{X}_4 \in (\{A_4\} \cup T)^*$, $X_{3,4}, \bar{X}_{3,4} \in (\{A_3, A_4\} \cup T)^*$, i.e., $S(r_j) = (4, 0, 3, 1, 2)$, or like
- $SF_{r_j} = tA_4X_4A_5\bar{X}_4A_1A_3X_{3,4}$, or $SF_{r_j} = tA_4X_4A_5\bar{X}_4A_1\tilde{X}_4A_3X_{3,4}$, $t \in T^*$, $X_4, \bar{X}_4, \tilde{X}_4 \in (\{A_4\} \cup T)^*$, $X_{3,4} \in (\{A_1, A_3, A_4\} \cup T)^*$, i.e., $S(r_j) = (3, 0, 4, 1, 2)$.

For the case of leftmost-3 derivation, rule r_j can be applied in leftmost-3 manner after $r_{j'}$, by rewriting the leftmost occurrence of A_3 in $S(r_{j'})$, even if there exist a RC rule $r_{j''} \in P$ able to rewrite A_4.

Next we sketch an ATM $\mathcal{A}$ that decides whether an input word $\gamma = \gamma_1\gamma_2...\gamma_n$ belongs to $SZRCL_i^{ac}(CF)$, $i \in \{2,3\}$. Let Q_1 be the quotient, and R_1 the remainder of n divided by $[\log n]$. Dividing Q_1 by $[\log n]$ a new quotient Q_2 and remainder R_2 are obtained. If this "iterated" division is performed until the resulted quotient, denoted by Q_ℓ, can be no longer divided by $[\log n]$, then n (written in the base $[\log n]$) is $n = ((...((Q_\ell\,[\log n]+R_\ell)\,[\log n]+R_{\ell-1})\,[\log n]+...)\,[\log n]+R_2)[\log n]+R_1$, $1 \leq Q_\ell < \log n$, $0 \leq R_l < \log n$, $l \in \{1,...,\ell\}$, and $\ell < \log n$.

Knowing R_1, $\mathcal{A}$ guesses an R_1-tuple of ranging vectors associated with the first R_1 triplets (RC rules) occurring in γ and checks whether $\gamma_1\gamma_2...\gamma_{R_1}$ is valid, according to the leftmost-i derivation manner, $i \in \{2,3\}$. Then $\mathcal{A}$ guesses a $[\log n]$-tuple of ranging vectors associated with triplets placed at the $[\log n]$ cutting points in γ obtained by dividing $[R_1+1...n]$ in $[\log n]$ intervals of length Q_1. $\mathcal{A}$ continues with this routine for each interval of length Q_1 as follows. First $\mathcal{A}$ checks, in parallel, whether the first R_2 triplets in each Q_1-interval forms a valid substring of a leftmost-i, $i \in \{2,3\}$, Szilard word. Then, in parallel for each Q_1-interval, $\mathcal{A}$ guesses another $[\log n]$-tuple of ranging vectors associated with triplets placed at the $[\log n]$ cutting points in γ obtained by dividing each interval of length $Q_1 - R_2$ into $[\log n]$ intervals of length Q_2. This procedure is repeated until intervals of length $Q_\ell < \log n$ are obtained. At this point, $\mathcal{A}$ checks whether the substrings of γ corresponding to Q_ℓ-intervals, are valid according to the leftmost-i derivation order, $i \in \{2,3\}$. It can be proved that all cutting points are right edges of these intervals. If correct ranging vectors can be found for all intervals and all cutting points, then γ is a correct leftmost-i, $i \in \{2,3\}$, Szilard word. Hence, we have

Theorem 9 *Each language $L \in SZRCL_i^{ac}(CF)$, $i \in \{2,3\}$, can be recognized by an indexing ATM in $\mathcal{O}(\log n)$ space and $\mathcal{O}(\log^2 n)$ time.*

Proof. We prove the claim for the leftmost-2 derivation. For the leftmost-3 case the proof is almost the same. Let $G = (N, T, P, A_1)$ be an arbitrary RCG^{ac} working in leftmost-2 derivation manner, and $\mathcal{A}$ be an indexing ATM with a similar configuration as in the proof of Theorem 7. Let $\gamma = \gamma_1\gamma_2...\gamma_n \in P^*$, be an input of length n. To guess the length of γ, $\mathcal{A}$ proceeds with the procedure described at Level 1 (*Existential*), Theorem 7.

Level 2 (*Existential*) Consider the quotient Q_1 and the remainder R_1 of the division of n by $[\log n]$, where $0 \leq R_1 < [\log n]$. $\mathcal{A}$ spawns $\mathcal{O}(c^{\log n})$ existential branches, each branch holding an R_1-tuple $\Re_{R_1} = (S(\gamma_1), S(\gamma_2), ..., S(\gamma_{R_1}))$ of ranging vectors, where[19] $c = \mathcal{O}(\sum_{s=1}^{m-1}(m-s+1)^m)$ and $S(\gamma_v)$ is the ranging vector associated with γ_v, $1 \leq v \leq R_1$. $\mathcal{A}$ checks (Levels 3) in $\mathcal{O}(\log n)$ time and space, whether all vectors in $\Re_{R_1}$ are correct, in the sense that $S(\gamma_v)$ can be obtained from $S(\gamma_{v-1})$ by applying rule γ_v in leftmost-2 derivation manner on the sentential form built from $S(\gamma_{v-1})$.

Level 3 (*Universal*) $\mathcal{A}$ spawns R_1 universal processes $\wp_v^{(R_1)}$, $1 \leq v \leq R_1$.

[19]The constant c depends on the number of vectors in $\mathbf{N}^m$ that can be built upon the set $\{0,1,...,m\}$. If a certain sentential form has only $m-s$ distinct nonterminals, then there are $(m-s+1)^m$ guesses that provide the ranging vector associated with this sentential form. Hence, here and throughout this proof, $c = \mathcal{O}\sum_{s=1}^{m-1}(m-s+1)^m)$, see also the explanations at page 22.

• Process $\wp_1^{(R_1)}$ reads $\gamma_1 = (p_{\gamma_1}, Q_{\gamma_1}, R_{\gamma_1})$ and it checks whether γ_1 can be applied on A_1, i.e., $\alpha_{\gamma_1}{=}A_1$, and whether $S(\gamma_1)$ is the ranging vector associated with β_{γ_1}. If these conditions hold, $\wp_1^{(R_1)}$ returns 1. Otherwise, it returns 0.

• For each $\wp_v^{(R_1)}$, $2 \leq v \leq R_1$, $\mathcal{A}$ counts the number of occurrences of each RC rule $r_j \in P$, $1 \leq j \leq k$, in $\gamma^{(v)} = \gamma_1\gamma_2...\gamma_{v-1}$. Suppose that each r_j occurs $c_j^{(v)}$ times in $\gamma^{(v)}$, $0 \leq c_j^{(v)} \leq v-1$. For each $1 \leq l \leq m$, $\mathcal{A}$ computes $s_{A_l}^{(v)} = V_l^0 + \sum_{j=1}^{k} c_j^{(v)} V_l(r_j)$, i.e., the number of times nonterminal A_l occurs in the sentential form obtained at the v^{th} step of derivation. Each $\wp_v^{(R_1)}$, $2 \leq v \leq R_1$, returns 1 if only one of the conditions in $\mathbf{I}_1^{(v)}$ and all conditions in $\mathbf{I}_2^{(v)}$ hold.

$\mathbf{I}_1^{(v)}$
1. $s_{\alpha_{\gamma_v}}^{(v)} \geq 1$, $\alpha_{\gamma_v} \notin Q_{\gamma_v} \cup R_{\gamma_v}$, $s_X^{(v)} \geq 1$, for each $X \in Q_{\gamma_v}$, and $s_Y^{(v)} = 0$ for each $Y \in R_{\gamma_v}$,
2. $s_{\alpha_{\gamma_v}}^{(v)} \geq 2$ if $\alpha_{\gamma_v} \in Q_{\gamma_v} - R_{\gamma_v}$, $s_X^{(v)} \geq 1$, for each $X \in Q_{\gamma_v}$, $X \neq \alpha_{\gamma_v}$, and $s_Y^{(v)} = 0$ for each $Y \in R_{\gamma_v}$,
3. $s_{\alpha_{\gamma_v}}^{(v)} = 1$ if $\alpha_{\gamma_v} \in R_{\gamma_v} - Q_{\gamma_v}$, $s_X^{(v)} \geq 1$, for each $X \in Q_{\gamma_v}$, and $s_Y^{(v)} = 0$ for each $Y \in R_{\gamma_v}$, $Y \neq \alpha_{\gamma_v}$.

$\mathbf{I}_2^{(v)}$
1. $S(\gamma_{v-1})$ is a possible ranging vector on which γ_{v-1} ends the $(v-1)^{th}$ step of derivation, i.e., $S_l(\gamma_{v-1}) = 0$ if $s_{A_l}^{(v)} = 0$, and $S_l(\gamma_{v-1}) \neq 0$ if $s_{A_l}^{(v)} > 0$, for each $1 \leq l \leq m$,
2. for any RC rule $r = (p, Q, R) \in P$, p is of the form $\alpha_p \rightarrow \beta_p$, $\alpha_p \neq \alpha_{\gamma_v}$, that can be applied on $\gamma^{(v)}$ (because it satisfies one of the conditions of type $1-3$ in $\mathbf{I}_1$) we have $S_{\alpha_{\gamma_v}}(\gamma_{v-1}) < S_{\alpha_p}(\gamma_{v-1})$, i.e., p_{γ_v} can be applied in leftmost-2 manner on $\gamma^{(v)}$ with the ranging vector $S(\gamma_{v-1})$,
3. $S(\gamma_v)$ is a possible ranging vector with which γ_v ends the v^{th} step of derivation, i.e., $S_l(\gamma_v) = 0$ if $s_{A_l}^{(v)} + V_l(\gamma_v) = 0$, and $S_l(\gamma_v) \neq 0$ if $s_{A_l}^{(v)} + V_l(\gamma_v) > 0$, for each $1 \leq l \leq m$.

If all processes $\wp_v^{(R_1)}$, $1 \leq v \leq R_1$, return 1 then $\Re_{R_1}$ is a correct guess and the existential branch holding the $[\log n]$-tuple, spawned at Level 2, is labeled by 1.

Level 4 (*Existential*) Let Q_2 be the quotient and R_2 the remainder of Q_1 divided by $[\log n]$, $0 \leq R_2 < [\log n]$. $\mathcal{A}$ spawns $\mathcal{O}(c^{\log n})$ existential branches, each of which holding a tuple $\Re_{R_2}^c{=}(S(\gamma_{R_1}), S(\gamma_{R_1+R_2}), S(\gamma_{R_1+Q_1}), S(\gamma_{R_1+Q_1+R_2}), ..., S(\gamma_{R_1+([\log n]-1)Q_1}), S(\gamma_{R_1+([\log n]-1)Q_1+R_2}))$, where $S(\gamma_{R_1})$ is the ranging vector belonging to the R_1-tuple found correct at Level 3. Because the tuple $\Re_{R_1}$ is not useful anymore, the space used by $\mathcal{A}$ to record $\Re_{R_1}$ is allocated now to record $\Re_{R_2}^c$.

Level 5 (*Universal*) On each existential branch from Level 4, $\mathcal{A}$ spawns $[\log n]$ universal processes $\wp_{i_1}^{(Q_1)}$, $0 \leq i_1 \leq [\log n] - 1$. Each process $\wp_{i_1}^{(Q_1)}$ takes the interval $[R_1 + i_1Q_1...R_1 + i_1Q_1 + R_2]$, and checks whether the ranging vectors $S(\gamma_{R_1+i_1Q_1})$ and $S(\gamma_{R_1+i_1Q_1+R_2})$, $1 \leq i_1 \leq [\log n] - 1$, provide a correct order in which the leftmost-2 derivation can be performed between $\gamma_{R_1+i_1Q_1}$ and $\gamma_{R_1+i_1Q_1+R_2}$. Besides $S(\gamma_{R_1+i_1Q_1})$ and $S(\gamma_{R_1+i_1Q_1+R_2})$, each $\wp_{i_1}^{(Q_1)}$ also keeps, from the previous level, the ranging vector $S(\gamma_{R_1+(i_1+1)Q_1})$. In this way each ranging vector $S(\gamma_{R_1+i_1Q_1})$,

$1 \leq i_1 \leq [\log n] - 1$, guessed at Level 4, is redirected to only one process, i.e., $\wp_{i_1-1}^{(Q_1)}$. Denote by $x_{i_1} = R_1 + i_1 Q_1$, $0 \leq i_1 \leq [\log n] - 1$.

Level 6 (*Existential*) For each universal process $\wp_{i_1}^{(Q_1)}$, $0 \leq i_1 \leq [\log n] - 1$, $\mathcal{A}$ spawns $\mathcal{O}(c^{\log n})$ existential branches, each branch holding an (R_2+1)-tuple of ranging vectors $\Re_{R_2} = (S(\gamma_{x_{i_1}}), S(\gamma_{x_{i_1}+1}), ..., S(\gamma_{x_{i_1}+R_2-1}), S(\gamma_{x_{i_1}+R_2}))$. Then $\mathcal{A}$ checks whether all vectors in $\Re_{R_2}$ are correct ranging vectors according to the leftmost-2 derivation requirements. This can be done, for each process $\wp_{i_1}^{(Q_1)}$, $0 \leq i_1 \leq [\log n] - 1$, in $\mathcal{O}(\log n)$ time and space, through Level 7 as follows.

Level 7 (*Universal*) For each existential branch spawned at Level 6, $\mathcal{A}$ spawns R_2 universal processes $\wp_v^{(R_2)}$, $1 \leq v \leq R_2$. On each $\wp_v^{(R_2)}$, $\mathcal{A}$ checks whether each substring $\gamma_{x_{i_1}}\gamma_{x_{i_1}+1}...\gamma_{x_{i_1}+v}$ is correct according to the leftmost-2 derivation order. In this respect, for each $\wp_v^{(R_2)}$, $1 \leq v \leq R_2$, $\mathcal{A}$ counts the number of occurrences of each RC rule $r_j \in P$, $1 \leq j \leq k$, in $\gamma^{(i_1,v)} = \gamma_1\gamma_2...\gamma_{x_{i_1}+v-1}$. Suppose that each r_j occurs $c_j^{(i_1,v)}$ times, $0 \leq c_j^{(i_1,v)} \leq x_{i_1} + v - 1$, in $\gamma^{(i_1,v)}$. For each $1 \leq l \leq m$, $\mathcal{A}$ computes $s_{A_l}^{(i_1,v)} = V_l^0 + \sum_{j=1}^{k} c_j^{(i_1,v)} V_l(r_j)$, i.e., the number of times A_l occurs in the sentential form obtained at the $(x_{i_1}+v)^{th}$ step of derivation. Then $\mathcal{A}$ checks conditions of type $\mathbf{I}_1^{(v)}$ and $\mathbf{I}_2^{(v)}$ (Level 3) for the RC rule $\gamma_{x_{i_1}+v}$, i.e., instead of v, $x_{i_1}+v$ is considered. Denote by $\mathbf{I}_1^{(i_1,v)}$ and $\mathbf{I}_2^{(i_1,v)}$ these conditions.

Each $\wp_v^{(R_2)}$, $1 \leq v \leq R_2$, is said *partially correct* if one of the conditions in $\mathbf{I}_1^{(i_1,v)}$ and all conditions in $\mathbf{I}_2^{(i_1,v)}$ hold. If $\wp_v^{(R_2)}$ is not partially correct, it is labeled by 0. Note that, at this moment we cannot decide whether $\wp_v^{(R_2)}$ can be labeled by 1, since we do not know whether $S(\gamma_{x_{i_1}})$ is valid, i.e., whether $\gamma_{x_{i_1}}$ indeed ends the $x_{i_1}^{th}$ step of derivation with the ranging vector $S(\gamma_{x_{i_1}})$, and whether $\gamma_{x_{i_1}}$ can be applied in the leftmost-2 derivation manner upon the ranging vector $S(\gamma_{x_{i_1}-1})$ (which is not yet guessed[20]). The logical value of each $\wp_v^{(R_2)}$ will be decided at the end of computation, when it will be known whether $S(\gamma_{x_{i_1}})$ is a valid ranging vector with respect to the rules that compose the subword $\gamma_{R_1+(i_1-1)Q_1}...\gamma_{R_1+i_1Q_1-1} = \gamma_{x_{i_1-1}}...\gamma_{x_{i_1}-1}$. A partially correct process $\wp_v^{(R_2)}$ is labeled by $\diamond$. If all processes $\wp_v^{(R_2)}$ are labeled by $\diamond$, then the existential branch holding the tuple $\Re_{R_2}$, provided at Level 6, is labeled by $\diamond$. Otherwise, this branch is labeled by 0. Process $\wp_{i_1}^{(Q_1)}$, yielded at Level 5, will be labeled by $\diamond$ if there exists at least one existential branch labeled by $\diamond$ at Level 6. Otherwise, $\wp_{i_1}^{(Q_1)}$ returns 0.

Suppose that we have run the algorithm up to the $(\ell-1)^{th}$ "iterated" division of n by $[\log n]$, i.e., we know the quotient $Q_{\ell-1}$ and the remainder $R_{\ell-1}$ of $Q_{\ell-2}$ divided by $[\log n]$. More precisely, $Q_{\ell-2} = Q_{\ell-1}[\log n] + R_{\ell-1}$ and $n = ((...((Q_{\ell-1}[\log n] + R_{\ell-1})[\log n] + R_{\ell-2})[\log n] + ...)[\log n] + R_2)[\log n] + R_1$, with $Q_{\ell-1} > [\log n]$, $0 \leq R_l < [\log n]$, $l \in \{1, 2, ..., \ell-1\}$, and $\ell \leq [\log n]$.

Level 4$(\ell-1)$ (*Existential*) Let Q_ℓ be the quotient and R_ℓ the remainder of $Q_{\ell-1}$ di-

[20] $S(\gamma_{x_{i_1}-1})$ will be guessed at the last level of the computation tree of $\mathcal{A}$, when all the remainders of the "iterated" division of n by $[\log n]$ will be spent, and when $\gamma_{x_{i_1}-1}$ will be the last rule occurring in the suffix of length Q_ℓ of the substring $\gamma_{R_1+(i_1-1)Q_1}...\gamma_{R_1+i_1Q_1-1} = \gamma_{x_{i_1-1}}...\gamma_{x_{i_1}-1}$ of γ.

vided by $[\log n]$, $0 \leq Q_\ell, R_\ell < [\log n]$. Since $Q_{\ell-2}$, $R_{\ell-2}$ and $R_{\ell-1}$ are no more needed, the space used to record them is now used to record Q_ℓ and R_ℓ in binary, still keeping $Q_{\ell-1}$. Denote by $x_{i_{\ell-2}} = \sum_{l=1}^{\ell-1} R_l + \sum_{l=1}^{\ell-2} i_l Q_l$. For each existential branch labeled by $\diamond$ at Level $4\ell-6$, $\mathcal{A}$ spawns $\mathcal{O}(c^{\log n})$ existential branches, each branch holding a $2\,[\log n]$-tuple $\Re^c_{R_\ell} = (S(\gamma_{x_{i_{\ell-2}}}), S(\gamma_{x_{i_{\ell-2}}+R_\ell}), S(\gamma_{x_{i_{\ell-2}}+ Q_{\ell-1}}), S(\gamma_{x_{i_{\ell-2}}+ Q_{\ell-1}+R_\ell}), ...,$ $S(\gamma_{x_{i_{\ell-2}}+([\log n]-1)Q_{\ell-1}}), S(\gamma_{x_{i_{\ell-2}}+([\log n]-1)Q_{\ell-1}+R_\ell}))$, where $S(\gamma_{x_{i_{\ell-2}}})$ is the ranging vector belonging to tuple $\Re_{R_{\ell-1}}$ found correct at Level $4\ell-5$. Because $\Re_{R_{\ell-1}}$ is no more needed the space used to record $\Re_{R_{\ell-1}}$ is allocated now to record $\Re^c_{R_\ell}$. Then $\mathcal{A}$ proceeds with Level $4\ell-3$, similar to Levels 5,..., $4\ell-7$.

Level $4\ell-3$ (*Universal*) On each existential branch spawned at Level $4(\ell-1)$, $\mathcal{A}$ spawns $[\log n]$ universal processes $\wp^{(Q_{\ell-1})}_{i_{\ell-1}}$, $0 \leq i_{\ell-1} \leq [\log n]-1$. Denote by $x_{i_{\ell-1}} = \sum_{l=1}^{\ell-1} R_l + \sum_{l=1}^{\ell-1} i_l Q_l = x_{i_{\ell-2}} + i_{\ell-1} Q_{\ell-1}$, $0 \leq i_{\ell-1} \leq [\log n]-1$. Each process $\wp^{(Q_{\ell-1})}_{i_{\ell-1}}$ takes the interval $[x_{i_{\ell-1}} ... x_{i_{\ell-1}} + R_\ell]$, and checks whether the ranging vectors (guessed at Level $4(\ell-1)$) $S(\gamma_{x_{i_{\ell-1}}})$ and $S(\gamma_{x_{i_{\ell-1}}+R_\ell})$, $0 \leq i_{\ell-1} \leq [\log n]-1$, provide a correct order in which the leftmost-2 derivation can be performed between $\gamma_{x_{i_{\ell-1}}}$ and $\gamma_{x_{i_{\ell-1}}+R_\ell}$. Besides $S(\gamma_{x_{i_{\ell-1}}})$ and $S(\gamma_{x_{i_{\ell-1}}+R_\ell})$, each $\wp^{(Q_{\ell-1})}_{i_{\ell-1}}$, also keeps from the previous level $S(\gamma_{x_{i_{\ell-2}}+(i_{\ell-1}+1)Q_{\ell-1}})$. Then $\mathcal{A}$ continues with Level $4\ell-2$, similar to Levels 6, ..., $4\ell-6$.

Level $4\ell-2$ (*Existential*) For each universal process $\wp^{(Q_{\ell-1})}_{i_{\ell-1}}$, $0 \leq i_{\ell-1} \leq [\log n]-1$, $\mathcal{A}$ spawns $\mathcal{O}(c^{\log n})$ existential branches, each branch holding an $(R_\ell+1)$-tuple of ranging vectors $\Re_{R_\ell} = (S(\gamma_{x_{i_{\ell-1}}}), S(\gamma_{x_{i_{\ell-1}}+1}), ..., S(\gamma_{x_{i_{\ell-1}}+R_\ell-1}), S(\gamma_{x_{i_{\ell-1}}+R_\ell}))$. Then $\mathcal{A}$ checks whether all vectors composing $\Re_{R_\ell}$ are correct. This can be done, for each process $\wp^{(Q_{\ell-1})}_{i_{\ell-1}}$, $0 \leq i_{\ell-1} \leq [\log n]-1$, in $\mathcal{O}(\log n)$ time and space, through Level $4\ell-1$ similar to Levels 3, 7, ..., $4\ell-5$.

Level $4\ell-1$ (*Universal*) For each existential branch spawned at Level $4\ell-2$, $\mathcal{A}$ spawns R_ℓ universal processes $\wp^{(R_\ell)}_v$, $1 \leq v \leq R_\ell$. On each $\wp^{(R_\ell)}_v$, $\mathcal{A}$ checks whether each substring $\gamma_{x_{i_{\ell-1}}} ... \gamma_{x_{i_{\ell-1}}+v}$ and each ranging vector in $\Re_{R_\ell}$ is correct according to the leftmost-2 derivation order. In this respect $\mathcal{A}$ checks conditions of type $\mathbf{I}_1^{(v)}$ and $\mathbf{I}_2^{(v)}$ (Level 3) for the rule $\gamma_{x_{i_{\ell-1}}+v}$, i.e., instead of v, $x_{i_{\ell-1}} + v$ is considered. Denote by $\mathbf{I}_1^{(i_{\ell-1},v)}$ and $\mathbf{I}_2^{(i_{\ell-1},v)}$ these conditions.

Each process $\wp^{(R_\ell)}_v$, $1 \leq v \leq R_\ell$, that satisfies only one of the conditions in $\mathbf{I}_1^{(i_{\ell-1},v)}$ and all conditions in $\mathbf{I}_2^{(i_{\ell-1},v)}$ is partially correct, and it is labeled by $\diamond$. Otherwise, $\wp^{(R_\ell)}_v$ is labeled by 0. If all processes $\wp^{(R_\ell)}_v$ are labeled by $\diamond$, then the existential branch holding the tuple $\Re_{R_\ell}$, provided at Level $4\ell-2$, is labeled by $\diamond$. Otherwise, this branch is labeled by 0. Process $\wp^{(Q_{\ell-1})}_{i_{\ell-1}}$, yielded at Level $4\ell-1$, is labeled by $\diamond$ if there exists at least one existential branch labeled by $\diamond$ at Level $4\ell-2$. Otherwise, $\wp^{(Q_{\ell-1})}_{i_{\ell-1}}$ is labeled by 0.

At this level the only substrings of γ left unchecked are those substrings that corresponds to the intervals of the form $I_{Q_{\ell-1}} = [\sum_{l=1}^{\ell-1} R_l + \sum_{l=1}^{\ell-2} i_l Q_l + i_{\ell-1} Q_{\ell-1} +$ $R_\ell ... \sum_{l=1}^{\ell-1} R_l + \sum_{l=1}^{\ell-2} i_l Q_l + (i_{\ell-1}+1) Q_{\ell-1}]$, $0 \leq i_l \leq [\log n]-1$, $1 \leq l \leq \ell-1$, and besides the cutting points $P^u_\ell = \sum_{l=1}^{u} R_l + \sum_{l=1}^{u-1} i_l Q_l + (i_u+1) Q_u$, $1 \leq u \leq \ell-1$.

On each interval of type $I_{Q_{\ell-1}}$, $\mathcal{A}$ proceeds with Level 4ℓ.

Level 4ℓ (*Existential*) Each interval $I_{Q_{\ell-1}}$ can be divided into $[\log n]$ subintervals of length $1 \leq Q_\ell < [\log n]$. Hence, $\mathcal{A}$ spawns $\mathcal{O}(c^{\log n})$ existential branches each of which holds a $[\log n]$-tuple $\Re^c_{Q_\ell} = (S(\gamma_{x_{i_{\ell-1}}+R_\ell}), S(\gamma_{x_{i_{\ell-1}}+R_\ell+Q_\ell}), ..., S(\gamma_{x_{i_{\ell-1}}+R_\ell+([\log n]-1)Q_\ell}))$, where $S(\gamma_{x_{i_{\ell-1}}+R_\ell})$ is the ranging vector found valid at Level $4\ell-1$.

Level $4\ell+1$ (*Universal*) For each existential branch spawned at Level 4ℓ, $\mathcal{A}$ spawns $[\log n]$ universal processes $\wp^{(Q_\ell)}_{i_\ell}$, $0 \leq i_\ell \leq [\log n]-1$. Each process $\wp^{(Q_\ell)}_{i_\ell}$ takes an interval of length Q_ℓ of the form $[\sum_{l=1}^{\ell} R_l + \sum_{l=1}^{\ell-1} i_l Q_l + i_\ell Q_\ell ... \sum_{l=1}^{\ell} R_l + \sum_{l=1}^{\ell-1} i_l Q_l + (i_\ell+1)Q_\ell]$. Denote by $x_{i_\ell} = \sum_{l=1}^{\ell} R_l + \sum_{l=1}^{\ell-1} i_l Q_l + i_\ell Q_\ell$, $0 \leq i_\ell \leq [\log n]-1$. For each interval $[x_{i_\ell} ... x_{i_\ell+1}]$, $\mathcal{A}$ checks whether the substring $\gamma_{x_{i_\ell}} ... \gamma_{x_{i_\ell+1}}$ is valid according to the leftmost-2 derivation order (Level $4\ell+2$).

Level $4\ell+2$ (*Existential*) For each $\wp^{(Q_\ell)}_{i_\ell}$, $0 \leq i_\ell \leq [\log n]-1$, $\mathcal{A}$ spawns $\mathcal{O}(c^{\log n})$ existential branches, each branch holding an $(Q_\ell+1)$-tuple of ranging vectors $\Re_{Q_\ell} = (S(\gamma_{x_{i_\ell}}), S(\gamma_{x_{i_\ell}+1}), ..., S(\gamma_{x_{i_\ell}+Q_\ell-1}), S(\gamma_{x_{i_\ell+1}}))$. In each $\Re_{Q_\ell}$ the vectors $S(\gamma_{x_{i_\ell}})$ and $S(\gamma_{x_{i_\ell+1}})$ have been guessed at Level 4ℓ. They are ranging vectors associated with triplets placed in cutting points, i.e., edges of intervals of length $[\log n]$. They are also overlapping points of two consecutive intervals of type $[x_{i_\ell} ... x_{i_\ell+1}]$. Hence, each ranging vector $S(\gamma_{x_{i_\ell}})$ is checked two times. Once if it is a valid vector on which $\gamma_{x_{i_\ell}+1}$ can be applied in leftmost-2 derivation manner (checked by process $\wp^{(Q_\ell)}_{i_\ell}$). Then, if by applying $\gamma_{x_{i_\ell}}$ on the sentential form built upon the ranging vector $S(\gamma_{x_{i_\ell}-1})$ a sentential form with an associated ranging vector equal to $S(\gamma_{x_{i_\ell}})$ is obtained (which is checked by $\wp^{(Q_\ell)}_{i_\ell-1}$).

As all intervals of type $[x_{i_\ell} ... x_{i_\ell+1}]$ are universally checked by processes $\wp^{(Q_\ell)}_{i_\ell}$, the tuple $\Re^c_{Q_\ell}$ spawned at Level 4ℓ is labeled by 1, if all ranging vectors $S(\gamma_{x_{i_\ell}})$ and all vectors in $\Re_{Q_\ell}$ are correct. To check whether all ranging vectors in $\Re_{Q_\ell}$ are correct, for each process $\wp^{(Q_\ell)}_{i_\ell}$, $0 \leq i_\ell \leq [\log n]-1$, $\mathcal{A}$ follows the same procedure, that requires $\mathcal{O}(\log n)$ time and space, described at Levels 3, 7, ..., $4\ell-1$ (Universals). For the last substring of length Q_ℓ in γ, i.e., the suffix of γ of length Q_ℓ of the form $\gamma_{\sum_{l=1}^{\ell} R_l + \sum_{l=1}^{\ell-1}([\log n]-1)Q_l + ([\log n]-1)Q_\ell} \cdots \gamma_{\sum_{l=1}^{\ell} R_l + \sum_{l=1}^{\ell-1}([\log n]-1)Q_l + [\log n]Q_\ell}$, $\wp^{(Q_\ell)}_{[\log n]-1}$ must check whether the triplet $\gamma_{\sum_{l=1}^{\ell} R_l + \sum_{l=1}^{\ell-1}([\log n]-1)Q_l + [\log n]Q_\ell} = \gamma_n$ ends the computation. This is done as for process $\wp_n$, Theorem 7.

Each cutting point $P^u_\ell = \sum_{l=1}^{u} R_l + \sum_{l=1}^{u-1} i_l Q_l + (i_u+1)Q_u$ can be equivalently rewritten as $\sum_{l=1}^{u+1} R_l + \sum_{l=1}^{u} i_l Q_l + [\log n]\, Q_{u+1}$, due to the equality $Q_u = [\log n]\, Q_{u+1} + R_{u+1}$, for any $1 \leq u \leq \ell-1$. Furthermore, $\sum_{l=1}^{u+1} R_l + \sum_{l=1}^{u} i_l Q_l + [\log n]\, Q_{u+1}$ is equal with $\sum_{l=1}^{u+1} R_l + R_{u+2} + \sum_{l=1}^{u} i_l Q_l + ([\log n]-1)Q_{u+1} + [\log n]\, Q_{u+2}$, due to the equality $Q_{u+1} = [\log n]\, Q_{u+2} + R_{u+2}$, for any $1 \leq u \leq \ell-2$. By applying this transformation k times, where $k = \ell - u$, each P^u_ℓ can be equivalently rewritten as $\sum_{l=1}^{u+1} R_l + R_{u+2} + ... + R_{u+k} + \sum_{l=1}^{u} i_l Q_l + ([\log n]-1)(Q_{u+1} + ... + Q_{u+k-1}) + [\log n]\, Q_{u+k}$, where $u+k = \ell$.

In this way each P^u_ℓ, yielded at Level $4u$ by $\Re^c_{R_{u+1}}$, $1 \leq u \leq \ell-1$, is in fact the right edge of an interval of the form $[\sum_{l=1}^{\ell} R_l + \sum_{l=1}^{\ell-1} i_l Q_l + i_\ell Q_\ell ... \sum_{l=1}^{\ell} R_l +$

$\sum_{l=1}^{\ell-1} i_l Q_l + (i_\ell + 1)Q_\ell] = [x_{i_\ell} ... x_{i_\ell+1}]$, for which $0 \leq i_l \leq [\log n] - 1$, $1 \leq l \leq \ell - 1$, $i_\ell = [\log n] - 1$. Hence, the decision on the correctness of each ranging vector $S(\gamma_{\sum_{l=1}^{u} R_l + \sum_{l=1}^{u-1} i_l Q_l + (i_u+1)Q_u}) = S(\gamma_{P_\ell^u})$ will be actually taken by a process of type $\wp^{(Q_\ell)}_{[\log n]-1}$. Since the validity of each cutting point is decided by a process of type $\wp^{(Q_\ell)}_{[\log n]-1}$, the logical value returned by this process is "propagated" up to the level of the computation tree that has spawned the corresponding cutting point, and thus each $\diamond$ symbol receives a logical value. The input is accepted, if going up in the computation tree, with all $\diamond$'s changed into logical values, the root of the tree is labeled by 1.

The tuples $\Re_{R_\hbar}$, $\Re^c_{R_\hbar}$, $\Re_{Q_\ell}$, $\Re^c_{Q_\ell}$, $1 \leq \hbar \leq \ell$, vectors $V(r_j)$, $1 \leq j \leq k$, and auxiliary net effects computed by $\mathcal{A}$ during the algorithm, are stored by using $\mathcal{O}(\log n)$ space, in a similar manner as in Theorems 7 and 8.

It is easy to observe that $\mathcal{A}$ has $\mathcal{O}(\log n)$ levels. Since at each level $\mathcal{A}$ spawns either $\mathcal{O}(n^{c_1})$ or $\mathcal{O}(c_2^{\log n})$ existential branches, where c_1 and c_2 are constants, (each level being thus convertible into a binary tree with $\mathcal{O}(\log n)$ levels), and at each Level $4\hbar$, $1 \leq \hbar \leq \ell$, $\mathcal{A}$ performs a division operation, which requires $\mathcal{O}(\log n)$ time and space [22], $\mathcal{A}$ will perform the whole computation in $\mathcal{O}(\log^2 n)$ parallel time and $\mathcal{O}(\log n)$ space. □

Corollary 13 *$SZRCL_i(CF) \cup SZRCL_i^{ac}(CF) \subset \mathcal{NC}^2$, $i \in \{2, 3\}$.*

Corollary 14 *$SZRCL_i(CF) \cup SZRCL_i^{ac}(CF) \subset DSPACE(\log^2 n)$, $i \in \{2, 3\}$.*

5 Szilard Languages of Programmed Grammars

Programmed grammars (PGs) are regulated rewriting grammars in which the application of a rule is conditioned by its occurrence in the so called *success field* associated with the rule previously applied in the derivation. If a rule is effectively applied, in the sense that its left-hand side occurs in the current sentential form, then the next rule to be used is chosen from its success field. For programmed grammars working in appearance checking mode, if the left-hand side of a rule (chosen to be applied) does not occur in the current sentential form then, at the next derivation step, a rule from its *failure field* must be used.

Programmed grammars have been introduced in [39] and [40], as a generalization of phrase-structure grammars with applications in natural language processing. This is possible due to the success and failure fields that prescribe an order in which productions can be used during the derivation. Extended versions of context-free programmed grammars, namely stochastic context-free programmed grammars, have been effectively used in pattern recognition [23] and [45]. For more results on the generative capacity of PGs the reader is referred to [39], [40], [18], [12], [13], and [14]. From [14] we have the following definitions.

5.1 Programmed Grammars - Prerequisites

Definition 16 A *programmed grammar* (PG) is a quadruple $G = (N, T, S, P)$ where S is the axiom, N and T, $N \cap T = \emptyset$, are finite sets of *nonterminals* and *terminals*,

respectively. P is a finite set of *triplets* (programmed grammar rules) of the form $r = (p_r, \sigma_r, \varphi_r)$, where p_r is a rewriting rule of the form $\alpha_r \to \beta_r$, with $\alpha_r \in (N \cup T)^* N (N \cup T)^*$ and $\beta_r \in (N \cup T)^*$, σ_r and φ_r are subsets of P, called the *success field* and *failure field* of r, respectively. If $\varphi_r = \emptyset$, for any $r \in P$, then G is a programmed grammar without *appearance checking*, otherwise G is a programmed grammar with *appearance checking* (henceforth PG^{ac}).

If all rules in P are phrase-structure (PS), context-sensitive (CS), context-free (CF), or regular (REG) rules then G is a PS, CS, CF, or REG programmed grammar, respectively.

Definition 17 Let $G = (N, T, S, P)$ be a PG or a PG^{ac} and $V = N \cup T$. The language $L(G)$ generated by G is defined as the set of all words $w \in T^*$ such that there is a derivation D: $S = w_0 \Rightarrow_{r_{i_1}} w_1 \Rightarrow_{r_{i_2}} w_2 \Rightarrow_{r_{i_3}} ... \Rightarrow_{r_{i_s}} w_s = w$, $s \geq 1$, and for $r_{i_j} = (p_{r_{i_j}}, \sigma_{r_{i_j}}, \varphi_{r_{i_j}})$, where $p_{r_{i_j}}$ is a rule of the form $\alpha_{i_j} \to \beta_{i_j}$, $1 \leq j \leq s-1$, we have either *i.* $w_{j-1} = w'_{j-1}\alpha_{i_j}w''_{j-1}$, $w_j = w'_{j-1}\beta_{i_j}w''_{j-1}$ for some $w'_{j-1}, w''_{j-1} \in V^*$ and $r_{i_{j+1}} \in \sigma_{r_{i_j}}$, or *ii.* α_{i_j} does not occur in w_{j-1}, $w_{j-1} = w_j$ and $r_{i_{j+1}} \in \varphi_{r_{i_j}}$.

Note that, for the case of PGs without appearance checking we have, in Definition 17, $\varphi_{r_{i_j}} = \emptyset$ for any r_{i_j}, and therefore there is no reason to check condition *ii.*

Denote by $L(P, X)$ and $L(P, X, ac)$ the class of languages generated by PGs and PGs with appearance checking, respectively, with X-rules, $X \in \{REG, CF, CF - \lambda, CS, PS\}$, then $L(M, X) = L(P, X)$ and $L(M, X, ac) = L(P, X, ac)$ [14]. Hence
1. $CFL \subset L(P, CF - \lambda) \subset L(P, CF - \lambda, ac) \subset CSL \subset L(P, CF, ac) = RE$,
2. $CFL \subset L(P, CF - \lambda) \subseteq L(P, CF) \subset L(P, CF, ac) = RE$,
3. $L(P, X) = L(P, X, ac) = XL$, $X \in \{REG, CS, PS\}$.

Let $G = (N, T, S, P)$ be a PG. If labels are associated with triplets[21] $r = (p, \sigma, \varphi) \in P$, in one-to-one correspondence, then the Szilard language associated with G is defined as follows.

Definition 18 Let $G = (N, T, S, P)$ be a programmed grammar, $P = \{r_1, r_2, ..., r_k\}$ the set of productions, $L(G)$ the language generated by G, and w a word in $L(G)$. The *Szilard word* of w associated with the derivation D: $S = w_0 \Rightarrow_{r_{i_1}} w_1 \Rightarrow_{r_{i_2}} w_2 \Rightarrow_{r_{i_3}} ... \Rightarrow_{r_{i_s}} w_s = w$, $s \geq 1$, is defined as $Sz_D(w) = r_{i_1} r_{i_2} ... r_{i_s}$, $r_{i_j} \in P$, $1 \leq j \leq s$. The *Szilard language* of G is $Sz(G) = \{Sz_D(w) | w \in L(G), D$ is a terminal derivation of $w\}$.

Let $SZP(X)$ and $SZP^{ac}(X)$ be the classes of Szilard languages associated with programmed grammars and programmed grammars with appearance checking, respectively, with X rules, $X \in \{CF, CS, PS\}$.

Definition 10 is applicable also for leftmost-i, $i \in \{1, 2, 3\}$, derivations in PGs with CF rules [14]. In terms of triplets $r = (p_r, \sigma_r, \varphi_r) \in P$, where p is a CF rule of the form $\alpha_{i_j} \to \beta_{i_j}$, $\alpha_{i_j} \in N$, these derivations can be explained as follows.

For the case of leftmost-1 derivations, after r has been effectively applied in leftmost-1 manner, the rule from σ_r that rewrites the leftmost nonterminal occurring

[21] As in the case of RCGs, for the sake of simplicity, we use the same notation both for a triple and the label associated with it.

in the current sentential form must be applied. If no rule in σ_r can rewrite the leftmost nonterminal occurring in the sentential form, then the grammar cannot be applied in the leftmost-1 derivation manner. If no rule in σ_r can be applied (because the left-hand sides of the rules do not occur in the sentential form) then a rule in $\varphi_{r'}$, where $r' = (p_{r'}, \sigma_{r'}, \varphi_{r'})$ is an arbitrary rule in σ_r, must be applied in leftmost-1 manner.

For the case of leftmost-2 derivations, after r has been effectively applied in leftmost-2 manner, the rule from σ_r that rewrites the leftmost nonterminal that can be rewritten by rules in σ_r (not necessary the leftmost nonterminal occurring in the sentential from) must be applied. If no rule in σ_r can be applied in leftmost-2 manner, then a rule in $\varphi_{r'}$, where $r' = (p_{r'}, \sigma_{r'}, \varphi_{r'})$ is an arbitrary rule in σ_r, that rewrites the leftmost nonterminal that can be rewritten by rules in $\varphi_{r'}$, must be applied.

For the case of leftmost-3 derivations, after r has been effectively applied in leftmost-3 manner, a rule from σ_r that rewrites the leftmost occurrence of its left-hand side in the current sentential form must be applied. If no rule in σ_r can be applied in leftmost-3 manner, then a rule in $\varphi_{r'}$, where $r' = (p_{r'}, \sigma_{r'}, \varphi_{r'})$ is an arbitrary rule in σ_r, must be applied in leftmost-3 manner.

Szilard languages associated with leftmost-i, $i \in \{1, 2, 3\}$, derivations can be defined in the same way as in Definition 18, with the specification that D is a leftmost-i derivation of w.

We denote by $SZPL_i(X)$ and $SZPL_i^{ac}(X)$ the classes of leftmost-i, $i \in \{1, 2, 3\}$, Szilard languages associated with PGs and PGs with appearance checking with X rules, $X \in \{CF, CS, PS\}$, respectively.

Let $G = (N, T, P, A_1)$ be an arbitrary programmed grammar with appearance checking, where A_1 is the axiom, $N = \{A_1, A_2, ..., A_m\}$ and $P = \{r_1, r_2, ..., r_k\}$ are the finite sets of ordered nonterminals and labels, respectively.

For each production $r = (p_r, \sigma_r, \varphi_r) \in P$, where p_r is a rewriting rule of the form $\alpha_{p_r} \rightarrow \beta_{p_r}$, $\alpha_{p_r} \in (N \cup T)^* N (N \cup T)^*$, and $\beta_{p_r} \in (N \cup T)^*$, its *net effect* during the derivation D with respect to each nonterminal $A_l \in N$, $1 \leq l \leq m$, is given by the difference $df_{A_l}(p_r) = |\beta_{p_r}|_{A_l} - |\alpha_{p_r}|_{A_l}$. To each rule r we associate a vector $V(r) \in \mathbf{Z}^m$ defined by $V(r) = (df_{A_1}(p_r), df_{A_2}(p_r), ..., df_{A_m}(p_r))$, where $\mathbf{Z}$ is the set of integers. The value of $V(r)$ taken at the l^{th} place, $1 \leq l \leq m$, is denoted by $V_l(r)$.

5.2 On the Complexity of Unrestricted Szilard Languages

In this subsection we focus on unrestricted Szilard languages of PGs with CF rules. Leftmost Szilard languages are studied in Subsection 5.3. The case of Szilard languages of PGs with CS and PS rules is briefly discussed in Section 6.

Theorem 10 *Each language $L \in SZP(CF) \cup SZP^{ac}(CF)$ can be recognized by an indexing ATM in $\mathcal{O}(\log n)$ time and space ($SZP(CF) \cup SZP^{ac}(CF) \subseteq ALOGTIME$).*

Proof. We give the proof for the class $SZP^{ac}(CF)$. For the class $SZP(CF)$ the proof is simpler. Let $G = (N, T, P, A_1)$ be a PG^{ac} with CF rules, and $\mathcal{A}$ an indexing ATM composed of an input tape that stores an input $\gamma = \gamma_1\gamma_2...\gamma_n \in P^*$, an index tape, and a working tape composed of three tracks. Here and throughout

this section, each label γ_i corresponds to a triplet in P of the form $(p_{\gamma_i}, \sigma_{\gamma_i}, \varphi_{\gamma_i})$, where p_{γ_i} is a CF rule of the form $\alpha_{\gamma_i} \rightarrow \beta_{\gamma_i}$, $1 \leq i \leq n$. At the beginning of the computation the first track stores $k+1$ vectors, V^0 corresponding to the axiom, i.e., $V_1^0 = 1$ and $V_l^0 = 0$, $2 \leq l \leq m$, and $V(r_i)$, $1 \leq i \leq k$. The other two tracks are initially empty. In order to guess the length of γ, $\mathcal{A}$ proceeds with the procedure described at Level 1 (*Existential*), Theorem 3 or Theorem 7.

Levels 2-3 (*Universal-Existential*) $\mathcal{A}$ spawns n universal processes $\wp_i$, $1 \leq i \leq n$ (Level 2). On $\wp_1$ $\mathcal{A}$ checks whether $\alpha_{\gamma_1} = A_1$, while on $\wp_2$, $\mathcal{A}$ checks whether $\gamma_2 \in \sigma_{\gamma_1}$. For each $\wp_i$, $3 \leq i \leq n$, $\mathcal{A}$ counts the number of occurrences of each $r_j \in P$, $1 \leq j \leq k$, in $\gamma^{(i-1)} = \gamma_1\gamma_2...\gamma_{i-2}$. Suppose that r_j occurs $c_j^{(i-1)}$ times in $\gamma^{(i-1)}$, $0 \leq c_j^{(i-1)} \leq i-2$. Since for some occurrences of $r_j = (p_j, \sigma_j, \varphi_j)$ in $\gamma^{(i-1)}$, p_j may be either effectively applied (because its left-hand side α_{γ_j} occurs in the sentential form) or it is a "dummy" rule (because p_j cannot be applied), for each $1 \leq j \leq k$, $\mathcal{A}$ guesses a pair of arbitrarily large integers $t_j^{(i-1)} = (c_{j,a}^{(i-1)}, c_{j,d}^{(i-1)})$ such that $c_{j,a}^{(i-1)} + c_{j,d}^{(i-1)} = c_j^{(i-1)}$, where $c_{j,a}^{(i-1)}$ is the number of times r_j is effectively applied up to the $(i-1)^{th}$ step of derivation, and $c_{j,d}^{(i-1)}$ is the number of times r_j is a dummy rule in $\gamma^{(i-1)}$. Since there exist $\mathcal{O}(n^2)$ guesses, $\mathcal{A}$ spawns $\mathcal{O}(n^2)$ existential branches (Level 3). On each existential branch holding a pair $t_j^{(i-1)}$, $\mathcal{A}$ computes the sums $s_{A_l}^{(i-1)} = V_l^0 + \sum_{j=1}^{k} c_{j,a}^{(i-1)} V_l(r_j)$, $1 \leq l \leq m$, i.e., the number of occurrences of each A_l in the sentential form obtained at the $(i-1)^{th}$ step of derivation. Then, $\mathcal{A}$ checks whether one of the following conditions holds:

1. $s_{\alpha_{\gamma_{i-1}}}^{(i-1)} \geq 1$ and $\gamma_i \in \sigma_{i-1}$, i.e., γ_{i-1} is effectively applied and the next rule must be chosen from its success field,
2. $s_{\alpha_{\gamma_{i-1}}}^{(i-1)} = 0$ and $\gamma_i \in \varphi_{i-1}$, i.e., γ_{i-1} is a dummy rule and the next rule must be chosen from its failure field.

Besides, for the last process $\wp_n$, $\mathcal{A}$ computes $s_{A_l}^{(n,out,a)} = s_{A_l}^{(n-1)} + df_{A_l}(p_{\gamma_{n-1}}) + df_{A_l}(p_{\gamma_n})$ and $s_{A_l}^{(n,out,d)} = s_{A_l}^{(n-1)} + df_{A_l}(p_{\gamma_n})$, $1 \leq l \leq m$, and it checks whether one of the following conditions holds:

1. $s_{\alpha_{\gamma_{n-1}}}^{(n-1)} \geq 1$, $\gamma_n \in \sigma_{n-1}$, $s_{\alpha_{\gamma_n}}^{(n)} \geq 1$, $s_{A_l}^{(n,out,a)} = 0$, $1 \leq l \leq m$,
2. $s_{\alpha_{\gamma_{n-1}}}^{(n-1)} = 0$, $\gamma_n \in \varphi_{n-1}$, $s_{\alpha_{\gamma_n}}^{(n)} \geq 1$, $s_{A_l}^{(n,out,d)} = 0$, $1 \leq l \leq m$.

Each process $\wp_i$, $3 \leq i \leq n$, returns 1, if one of the conditions $1-2$ holds. Otherwise it returns 0. Finally, γ is accepted if all $\wp_i$, $1 \leq i \leq n$, return 1, i.e., all n universal branches are labeled by 1.

Each of the above processes uses the third track of the working tape for auxiliary computations, i.e., to record in binary the elements $c_j^{(i-1)}$, $c_{j,a}^{(i-1)}$, and $c_{j,d}^{(i-1)}$), $3 \leq i \leq n$, $1 \leq j \leq k$, and to compute the sums $s_{A_l}^{(i-1)}$, $3 \leq i \leq n$, and $s_{A_l}^{(n,out,a)}$, $1 \leq l \leq m$.

The counting procedure used by each process $\wp_i$, $1 \leq i \leq n$, is a function in the U_{E^*}-uniform NC^1 class. The same observation holds for the summation of a constant number of vectors or multiplication of an integer of at most $\log n$ bits long with a binary constant. Hence, all the above operations can be performed by an ATM in $\log n$ time and space. The out-degree of the computation tree at this level is

n. By using a divide and conquer procedure the computation tree can be converted into a binary tree of height at most $\log n$. Consequently, for the whole computation $\mathcal{A}$ uses $\mathcal{O}(\log n)$ time and space. □

Corollary 15 $SZP(CF) \cup SZP^{ac}(CF) \subset \mathcal{NC}^1$.

Corollary 16 $SZP(CF) \cup SZP^{ac}(CF) \subset DSPACE(\log n)$.

5.3 On the Complexity of Leftmost Szilard Languages

The algorithm described in the proof of Theorem 8 cannot be applied for the case of leftmost-1 SZLs of PGs with appearance checking. The explanation is that, in the proof of Theorem 8, even if process $\wp_v$ returns the true value, which means that at its turn γ_v can be applied in a leftmost-1 derivation manner on $\gamma_1\gamma_2...\gamma_{v-1}$, the process $\wp_i$ cannot "see" whether γ_v has been effectively applied in the derivation, or it is only a dummy rule, since all branches spawned at the same level of the computation tree of $\mathcal{A}$ are independent on each other. Hence, for the case of leftmost-1 derivation in PGs with appearance checking an algorithm similar to that described in the proof of Theorem 6 or Theorem 9 must be applied. Using a similar method as for Theorem 5 or Theorem 8 we have

Theorem 11 *Each language $L \in SZPL_1(CF)$ can be recognized by an indexing ATM in $\mathcal{O}(\log n)$ time and space.*

Corollary 17 $SZPL_1(CF) \subset \mathcal{NC}^1$.

Corollary 18 $SZPL_1(CF) \subset DSPACE(\log n)$.

In order to simulate derivations of type letfmost-i, $i \in \{1, 2, 3\}$, and to check whether a given word $\gamma \in P^*$, $\gamma = \gamma_1\gamma_2...\gamma_n$, belongs to $SZPL_i^{ac}(CF)$, as in the case of RCGs, for each triplet γ_i, $1 \leq i \leq n$, the ATM must have information concerning the order in which the first occurrence of each nonterminal $A_l \in N$, $1 \leq l \leq m$, occurs in the sentential form at any step of derivation. In this respect we redefine the notion of a *ranging vector* for PGs. A ranging vector associated with a triple $r_j = (p_j, \sigma_j, \varphi_j) \in P$, $1 \leq j \leq k$, provides the order in which first occurrences of nonterminals in N occur in the sentential form obtained after r_j has been applied at that step of derivation. Similar to the Definition 15, for the case of PGs we have

Definition 19 Let $G = (N, T, S, P)$ be a PG with appearance checking and CF rules, where $P = \{r_1, r_2, ..., r_k\}$ is the ordered finite set of triples in P. Let SF_{r_j} be the sentential form obtained after triplet $r_j = (p_j, \sigma_j, \varphi_j)$, $1 \leq j \leq k$, has been applied at a certain step of derivation in G. The *ranging vector* associated with SF_{r_j}, denoted by $S(r_j)$, $1 \leq j \leq k$, is a vector in $\mathbf{N}^m$ defined as

$$S_l(r_j) = \begin{cases} 0, & \text{if } A_l \in N \text{ does not occur in } SF_{r_j}, \text{ i.e., } |SF_{r_j}|_{A_l} = 0, \\ i, & \text{if the first occurrence of } A_l \text{ in } SF_{r_j} \text{ is the } i^{th} \text{ element in the} \\ & \text{order of first occurrences of nonterminals from } N \text{ in } SF_{r_j}. \end{cases}$$

Note that if $r_{j'} = (p_{j'}, \sigma_{j'}, \varphi_{j'})$ is applied in the Szilard word before $r_j = (p_j, \sigma_j, \varphi_j)$ then the ranging vector $S(r_j)$ can be computed knowing $S(r_{j'})$. This observation holds for all leftmost-i, $i \in \{1, 2, 3\}$, derivations.

Example 3 Consider the ranging vector $S(r_{j'}) = (3, 0, 2, 1, 0) \in \mathbf{N}^5$, associated with the sentential form $SF_{r_{j'}}$ obtained, at a certain step of derivation, after the application of rule $r_{j'}$, i.e., $SF_{r_{j'}} = A_4X_4A_3X_{3,4}A_1\bar{X}_{3,4}$, $X_4 \in (\{A_4\} \cup T)^*$, $X_{3,4}, \bar{X}_{3,4} \in (\{A_3, A_4\} \cup T)^*$.

If in $r_j = (p_j, \sigma_j, \varphi_j)$, p_j is the rule $A_3 \to tA_5$, then r_j can be applied in leftmost-2 derivation manner after $r_{j'}$, if either $r_{j'}$ has been effectively applied in leftmost-2 manner, $r_j \in \sigma_{j'}$, and no rule in $\sigma_{j'}$ rewrites A_4, or $r_{j'}$ is a dummy rule (case in which the shape of the sentential form $SF_{r_{j'}}$ is actually borrowed from the very last PG rule effectively applied before $r_{j'}$), $r_j \in \varphi_{j'}$, and no rule in $\varphi_{j'}$ rewrites A_4.

The triplet $r_j = (p_j, \sigma_j, \varphi_j)$ can be applied in leftmost-3 derivation manner after $r_{j'}$, if either $r_{j'}$ has been effectively applied in leftmost-3 manner, $r_j \in \sigma_{j'}$, and the rule p_j rewrites the first occurrence of A_3 in $SF_{r_{j'}}$ (even if there may exist rules in $\sigma_{j'}$ that rewrites the first occurrence of A_4 in $SF_{r_{j'}}$), or $r_{j'}$ is a dummy rule, $r_j \in \varphi_{j'}$, and the rule p_j rewrites the first occurrence of A_3 in $SF_{r_{j'}}$.

Depending on the position of the second occurrence of A_3 in $SF_{r_{j'}}$, the sentential form obtained after p_j has been applied on $SF_{r_{j'}}$ may look like

- $SF_{r_j} = A_4X_4A_5A_3X_{3,4}A_1X_{1,3,4}$, $X_4 \in (\{A_4\} \cup T)^*$, $X_{3,4} \in (\{A_3, A_4\} \cup T)^*$, $X_{1,3,4} \in (\{A_1, A_3, A_4\} \cup T)^*$, i.e., $S(r_j) = (4, 0, 3, 1, 2)$,
- $SF_{r_j} = A_4X_4A_5\bar{X}_4A_3X_{3,4}A_1X_{1,3,4}$, $X_4, \bar{X}_4 \in (\{A_4\} \cup T)^*$, $X_{3,4} \in (\{A_3, A_4\} \cup T)^*$, $X_{1,3,4} \in (\{A_1, A_3, A_4\} \cup T)^*$, i.e., $S(r_j) = (4, 0, 3, 1, 2)$, or like
- $SFr_j = A_4X_4A_5\bar{X}_4A_1X_{1,4}A_3X_{1,3,4}$, $X_4, \bar{X}_4 \in (\{A_4\} \cup T)^*$, $X_{1,4} \in (\{A_1, A_4\} \cup T)^*$, $X_{1,3,4} \in (\{A_1, A_3, A_4\} \cup T)^*$, i.e., $S(r_j) = (3, 0, 4, 1, 2)$.

Using a similar method as in the proof of Theorem 6, for MGs, or Theorem 9, for RCGs, for the case of leftmost-i derivations, $i \in \{1, 2, 3\}$, in PGs, we have

Theorem 12 *Each language $L \in SZPL_i^{ac}(CF)$, $i \in \{1, 2, 3\}$, can be recognized by an indexing ATM in $\mathcal{O}(\log n)$ space and $\mathcal{O}(\log^2 n)$ time.*

Corollary 19 $SZPL_i(CF) \cup SZPL_i^{ac}(CF) \subset \mathcal{NC}^2$, $i \in \{1, 2, 3\}$.

Corollary 20 $SZPL_i(CF) \cup SZPL_i^{ac}(CF) \subset DSPACE(\log^2 n)$, $i \in \{1, 2, 3\}$.

6 Remarks on Szilard Languages of Regulated Rewriting Grammars with PS Rules

The derivation mechanism in MGs, PGs, or RCGs is quite similar to the derivation mechanism in Chomsky grammars. In the case of MGs, the only difference is that productions are grouped into *matrices* composed of a finite number of rules obeying a *predefined order* and some constraints that prohibit the use of some of the rules composing the matrix sequence. For the case of PGs constraints are imposed by the *success* and *failure* lists that prescribe the rules eligible to be applied at a certain step of derivation, while for the case or RCGs constraints are provided by the *permitting* and *forbidding* contexts that enable or disable a rule to be applied.

These restrictions do increase the generative power of MGs, PGs, or RCGs [14] but they do not change the complexity of the corresponding Szilard languages.

On the other hand Definition 10 of leftmost-i, $i \in \{1,2,3\}$, derivations in MGs, PGs, or RCGs with CF rules, can be naturally generalized for PS rules as follows.

Let $G = (N, T, S, M, F)$ be a MG with PS rules, where $M = \{m_1, m_2, ..., m_k\}$, each m_j is a finite sequence of the form $m_j = (p_{j_1}, p_{j_2}, ..., p_{j_{k(j)}})$, $k(j) \geq 1$, and each rule $p_{j_i} \in m_j$, $1 \leq i \leq k(j)$, is of the form $\alpha_{p_{j_i}} \rightarrow \beta_{p_{j_i}}$, $\alpha_{p_{j_i}} \in (N \cup T)^* N (N \cup T)^*$, $\beta_{p_{j_i}} \in (N \cup T)^*$. Consider $P_\alpha = \{\alpha_{p_{j_i}} | 1 \leq j \leq k, 1 \leq i \leq k(j)\}$ the set of the left-hand sides of all rules in m_j, $1 \leq j \leq k$.

Consider $G = (N, T, S, P)$ a PG or RCG with PS rules, where $P = \{r_1, r_2, ..., r_k\}$ and each rule $p_j \in P$, $1 \leq j \leq k$, is of the form $\alpha_{p_j} \rightarrow \beta_{p_j}$, $\alpha_{p_j} \in (N \cup T)^* N (N \cup T)^*$ and $\beta_{p_j} \in (N \cup T)^*$. Consider $P_\alpha = \{\alpha_{p_j} | 1 \leq j \leq k\}$ the set of the left-hand sides of all rules in P.

Definition 20 A derivation in G, where G is a MG, PG or RCG is called

- *leftmost-1* if each rule used in the derivation rewrites the leftmost substring α occurring in the current sentential form, such that if $\alpha_0 \alpha$ is a prefix of the current sentential form, then $\alpha_0 \in T^*$ and $\alpha \in P_\alpha$,
- *leftmost-2* if at each step of derivation, the leftmost occurrence of $\alpha \in P_\alpha$ that can be rewritten is rewritten,
- *leftmost-3* if each rule used in the derivation rewrites the leftmost occurrence of its left-hand side in the current sentential form.

In [9] we proved that the class of leftmost Szilard languages of PS (and particularly of CS) grammars can be recognized in logarithmic time and space by indexing ATMs. The case of leftmost-1 derivation in MGs, PGs, or RCGs with CF or PS rules is in fact a generalization of the leftmost derivation in CGs (Defintion 5). Using a similar method as in [9] it can be proved that Theorems 5, 8, and 11 hold for classes of leftmost-1 Szilard languages of MGs, PGs, or RCGs (with or without appearance checking) with CS or PS rules, too. Hence, we have

Theorem 13 *Each* $L \in SZML_1(X) \cup SZPL_1(X) \cup SZRCL_1(X) \cup SZRCL_1^{ac}(X)$, $X \in \{CS, PS\}$ *can be recognized by an indexing ATM in* $\mathcal{O}(\log n)$ *time and space.*

Corollary 21 $SZML_1(X) \cup SZPL_1(X) \cup SZRCL_1(X) \cup SZRCL_1^{ac}(X) \subset \mathcal{NC}^1$, $X \in \{CS, PS\}$.

Corollary 22 $SZML_1(X) \cup SZPL_1(X) \cup SZRCL_1(X) \cup SZRCL_1^{ac}(X) \subset DSPACE(\log n)$, $X \in \{CS, PS\}$.

For the moment we have no results concerning the complexity of leftmost-1 Szilard languages of MGs and PGs with appearance checking and PS rules, or leftmost-i, $i \in \{2,3\}$, Szilard languages of MGs, PGs, and RCGs, with or without appearance checking, and PS rules.

References

[1] S. Ábrahám, *Some Questions of Phrase-Structure Grammars.* Computational Linguistic 4, 61–70, 1965.

[2] E. Altman and R. Banerji, *Some Problems of Finite Representability.* Information and Control, 8, 251–263, 1965.

[3] B.S. Baker, *Non-Context-Free Grammars Generating Context-Free Languages.* Information and Control, **24**, 231–246, 1974.

[4] J.L. Balcázar, J. Díaz, and J. Gabarró, *Structural Complexity.* Vol. II. Springer-Verlag, Berlin-Heidelberg, 1990.

[5] P.W. Beame, S.A. Cook, and H.J. Hoover, *Log Depth Circuits for Division and Related Problems.* SIAM Journal on Computing, **15**(4), 994–1003, 1986.

[6] R.L. Cannon, *Notes on Canonical Label Languages.* International Journal of Computer and Information Sciences, **8**(2), 141–148, 1979.

[7] A. Chandra, D. Kozen, and L. Stockmeyer, *Alternation.* Journal of Association for Computing Machinery, **28**(1), 114–133, 1981.

[8] N. Chomsky, *Three Models for the Description of Language.* IRE Transactions on Information Theory **2**(3), 113-124, 1956.

[9] L. Cojocaru, E. Mäkinen, and F.L. Ţiplea, *Classes of Szilard Languages in NC^1.* Proceedings of the 11^{th} International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 299–306, 2009.

[10] A.B. Cremers, H.A. Maurer, and O. Mayer, *A Note On Leftmost Restricted Random Context Grammars.* Information Processing Letters, **2**(2), 31–33, 1973.

[11] S. Crespi-Reghizzi and D. Mandrioli, *Petri Nets and Szilard Languages.* Information and Control, **33**(2), 177–192, 1977.

[12] J. Dassow, *Grammars With Regulated Rewriting.* Formal Languages and Applications, C. Martn-Vide, V. Mitrana, and G. Păun Eds., Springer-Verlag Berlin Heidelberg, 249-274, 2004.

[13] J. Dassow, H. Fernau, and Gh. Păun, *On the Leftmost Derivation in Matrix Grammars.* International Journal of Foundations of Computer Science, **10**(1), 61–80, 1999.

[14] J. Dassow and Gh. Păun, *Regulated Rewriting in Formal Language Theory.* Springer-Verlag Berlin Heidelberg, 1989.

[15] J. Dassow, Gh. Păun, and A. Salomaa, *Grammars with Controlled Derivations.* Handbook of Formal Languages, Volume II, Chapter 3, G. Rozenberg and A. Salomaa Eds., Springer-Verlag Berlin Heidelberg, 101–154, 1997.

[16] J. Duske, R. Parchmann, and J. Specht, *Szilard Languages of IO-Grammars*. Information and Control, **40**(3), 319–331, 1979.

[17] S. Ewert and A. P. J. van der Walt, *A Pumping Lemma for Random Permitting Context Languages*. Theoretical Computer Science, **270**(1-2) 959–967, 2002.

[18] H. Fernau, *Regulated Grammars under Leftmost Derivation*. Grammars, **3**(1), 37–62, 2000.

[19] P. Fischer, A. Meyer, and A. Rosenberg, *Counter Machines and Counter Languages*. Theory of Computing Systems, **2**(3), 265–283, 1968.

[20] A.C. Fleck, *An Analysis of Grammars by Their Derivation Sets*. Information and Control, 24, 389–398, 1974.

[21] M. Höpner and M. Opp, *Renaming and Erasing in Szilard Languages*. Proceedings of the Fourth Colloquium on Automata, Languages and Programming, ICALP 1977, LNCS 52, 244–257, 1977.

[22] W. Hesse, *Division is in uniform TC^0*. Automata, Languages and Programming 28^{th} International Colloquium, ICALP 2001, LNCS 2076, 104–114, 2001.

[23] T. Huang, K. S. Fu, *Stochastic Syntactic Analysis for Programmed Grammars and Syntactic Pattern Recognition*, Computer Graphics and Image Processing, **1**(3), 257–283, 1972.

[24] Y. Igarashi, *The Tape Complexity of Some Classes of Szilard Languages*. SIAM Journal of Computing, 6, 460–466, 1977.

[25] H.P. Kriegel and H.A. Maurer, *Formal Translations and the Containment Problem for Szilard Languages*. Automata Theory and Formal Languages 2^{nd} GI Conference Kaiserslautern, LNCS 33, 233–238, 1975.

[26] H.P. Kriegel and Th. Ottmann, *Left-Fitting Translations*. Automata, Languages and Programming, LNCS 52, 309–322, 1977.

[27] M. Linna, *Two Decidability Results for Deterministic Pushdown Automata*. Mathematical Foundations of Computer Science, LNCS 53, 365–373, 1977.

[28] E. Mäkinen, *On Certain Properties of Left Szilard Languages*. Elektronische Informationsverarbeitung und Kybernetik **19**(10/11), 497–501, 1983.

[29] E. Mäkinen, *On Context-Free and Szilard Languages*. BIT Numerical Mathematics, **24**(2), 164–170, 1984.

[30] E. Mäkinen, *A Note on Depth-First Derivations*. BIT Numerical Mathematics, **25**, 1, 293–296, 1985.

[31] E. Mäkinen, *On Szilard Languages of Pure Context-Free Grammars*. Elektronische Informationsverarbeitung und Kybernetik EIK (Journal of Information Processing and Cybernetics), **22**(10/11), 527–532, 1986.

[32] E. Mäkinen, *On Homomorphic Images of Szilard Languages*. International Journal of Computer Mathematics, **18**(3/4), 239–245, 1986.

[33] E. Mäkinen, *A Note on the Inclusion Problem for Szilard Languages*. International Journal of Computer Mathematics, **21**(3/4), 291–295, 1987.

[34] A. Mateescu and A. Salomaa, *Aspects of Classical Language Theory*. Handbook of Formal Languages, Volume I, Chapter 4, G. Rozenberg and A. Salomaa Eds., Springer-Verlag Berlin Heidelberg, 175–251, 1997.

[35] E. Moriya, *Associate Languages and Derivational Complexity of Formal Grammars and Languages*. Information and Control, **22**(2), 139–162, 1973.

[36] Gh. Păun, *On Szilard's Languages Associated to a Matrix Grammar*. Information Processing Letters, **8**(2), 104–105, 1979.

[37] M. Penttonen, *On Derivation Language Corresponding to Context-Free Grammars*. Acta Informatica, 3, 285–291, 1974.

[38] M. Penttonen, *Szilard Languages Are log n Tape Recognizable*. Elektronische Informationsverarbeitung und Kybernetik EIK, **13**(11), 595–602, 1977.

[39] D. J.Rosenkrantz, *Programmed Grammars - a New Device for Generating Formal Languages*. PhD Thesis, Columbia University, New York, 1967.

[40] D. J.Rosenkrantz, *Programmed Grammars and Classes of Formal Languages*. Journal of the ACM (JACM), **16**(1), 107–131, 1969.

[41] W. Ruzzo, *On Uniform Circuit Complexity*. Journal of Computer and System Sciences, **22**(3), 365–383, 1981.

[42] A. Salomaa, *Matrix Grammars with a Leftmost Restriction*. Information and Control, **20**(2), 143–149, 1972.

[43] A. Salomaa, *Formal Languages*. Academic Press, London-New York, 1973.

[44] E.D. Stotskij, *Some Restriction on Derivations in Context-Free Grammars*. Nauchno-Techn. Inform. Ser. **2**(7), 35-38, 1967.

[45] P. H. Swain and K. S. Fu, *Stochastic Programmed Grammars for Syntactic Pattern Recognition*. Pattern Recognition, **4**(1), 83–100, 1972.

[46] H. Vollmer, *Introduction to Circuit Complexity A Uniform Approach*. Springer-Verlag Berlin Heidelberg, 1999.

[47] A. P. J. van der Walt, *Random Context Languages*. Information Processing Letters, 71, 66–68, 1972.

[48] A. P. J. van der Walt and S. Ewert, *A Shrinking Lemma for Random Forbidding Context Languages*. Theoretical Computer Science, **237**(1-2), 149–158, 2000.